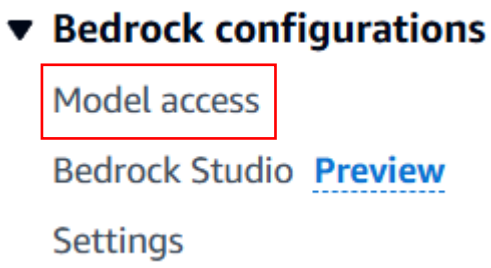
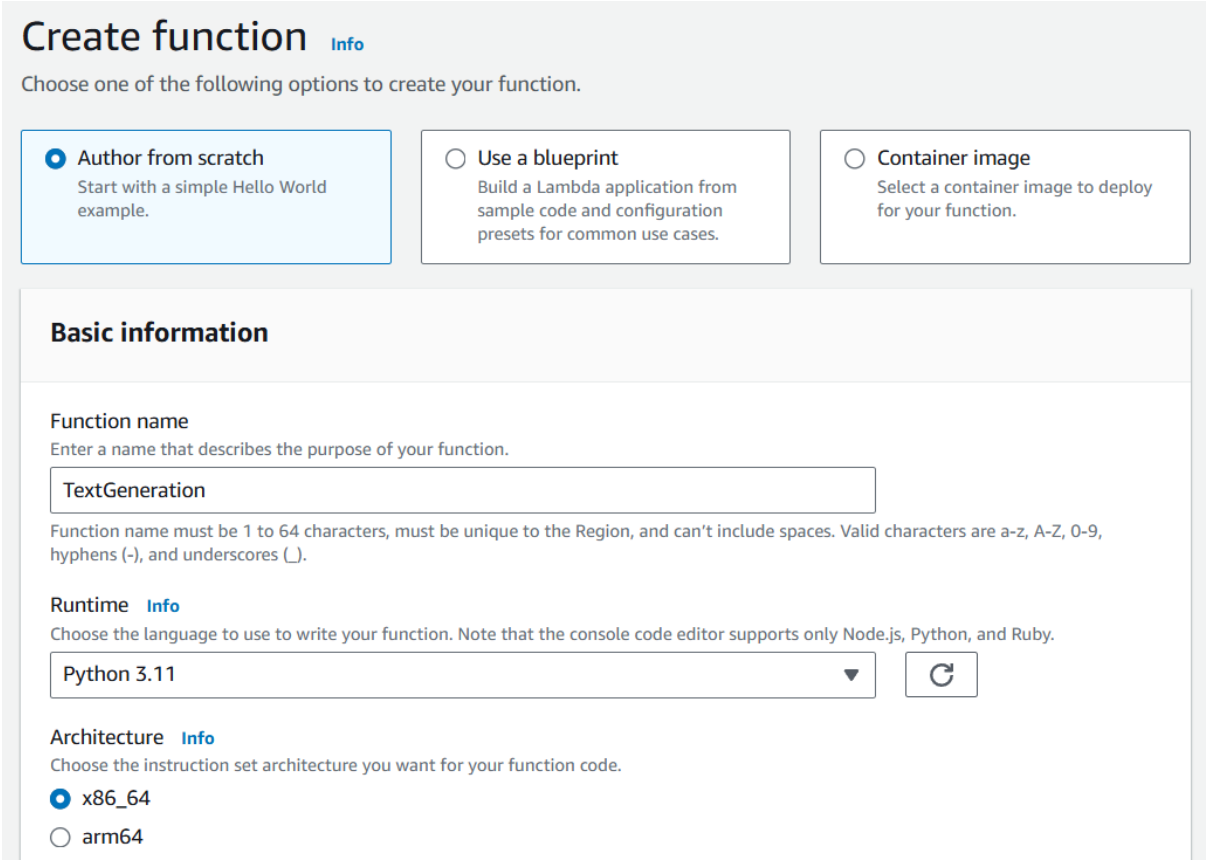
😊 Amazon Bedrock (Text Generation)

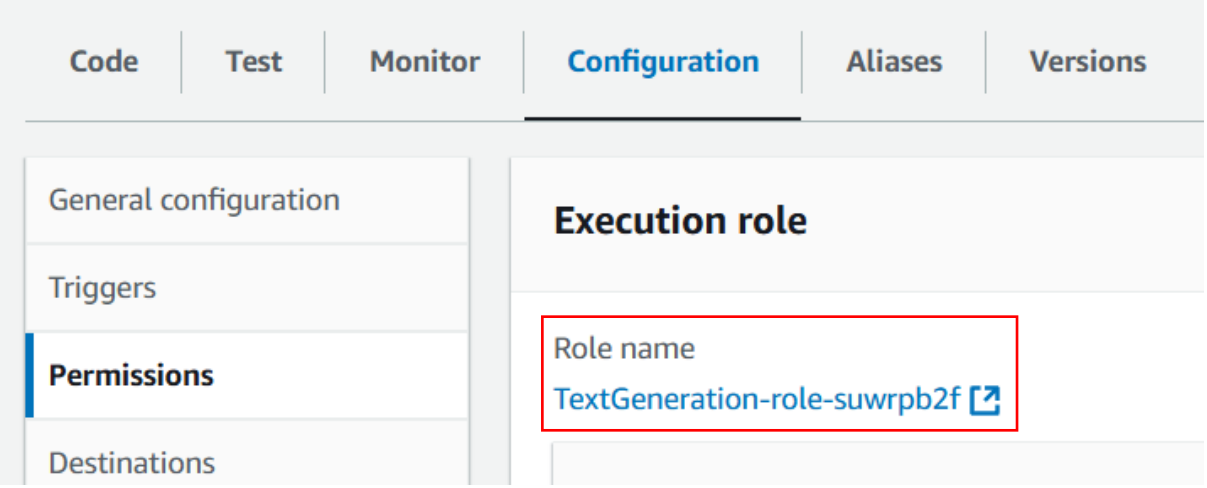
1. In this lab we will use the Text Generation feature of Amazon Bedrock using the Cohere foundational model.
2. If you have completed the previous lab then you will have access to this model. If not, open Amazon Bedrock, and from the left pane, scroll down to the bottom to expand bedrock configurations and open Model Access. Here you need to enable all the models you will see. Enabling them takes time.



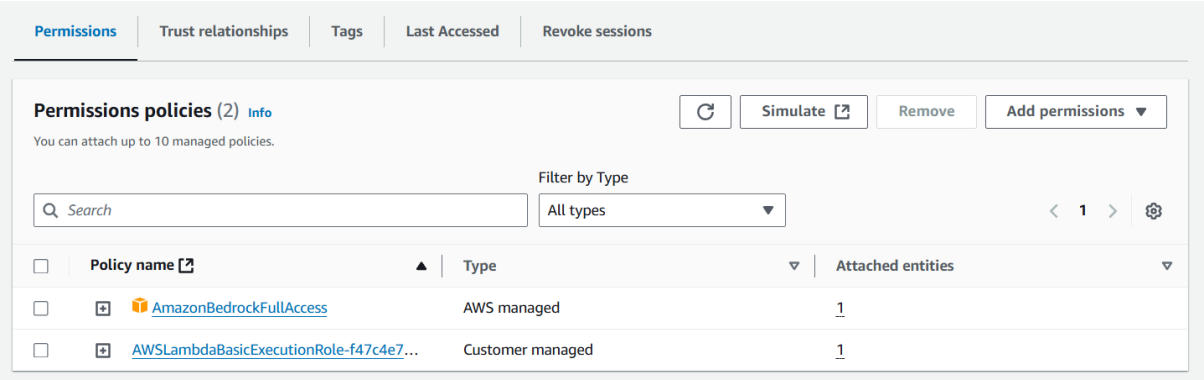
3. Now you need to search for AWS Lambda and navigate to it. Then click to create a function. Here you need to choose the author from scratch and give it a function name then create your function.



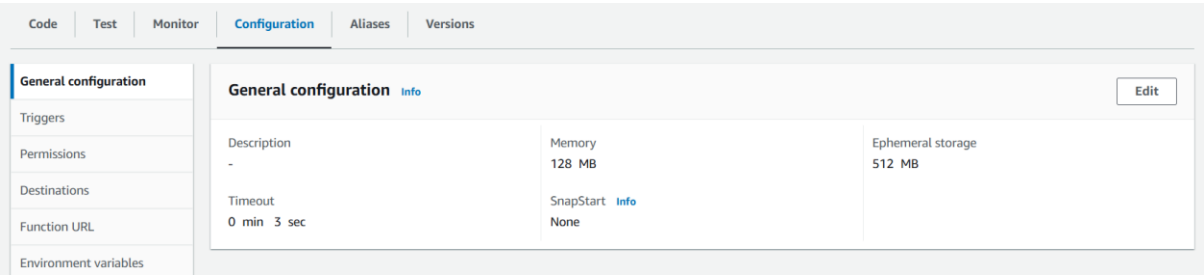
4. Since your lambda function has been created, you need to change some of its properties. You need to go to the configuration tab, then to Permission, and click on the IAM role attached to your lambda function.

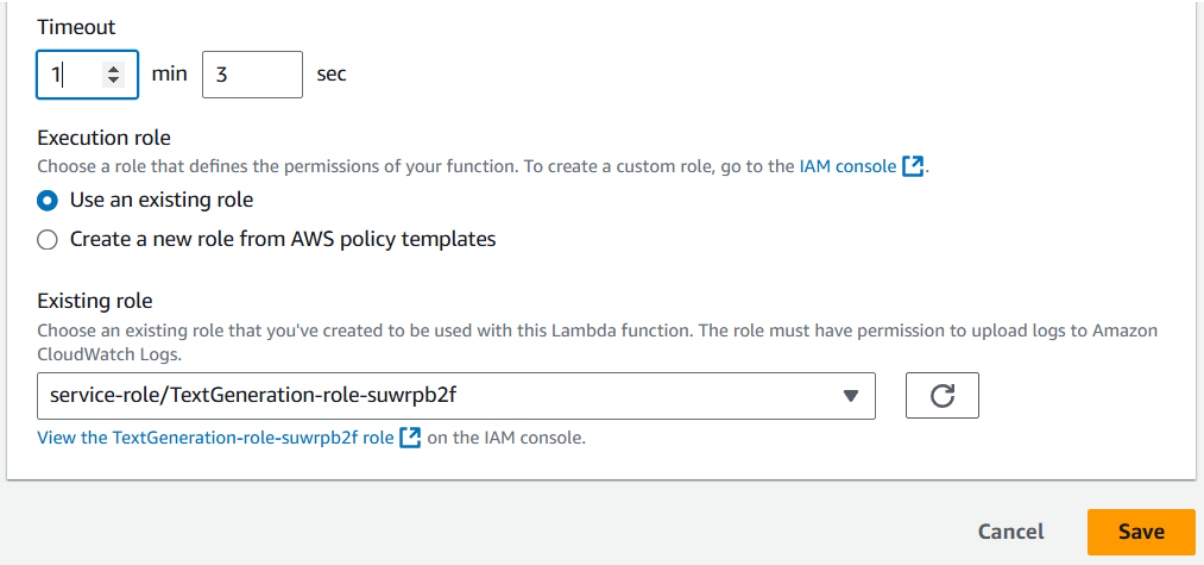


5. Then you need to provide Amazon bedrock full access permission to your role and come back.



6. Then come to general configuration and click on edit. Here we are going to increase the timeout limit to 1 minute from just 3 seconds. Click on save to save your changes.



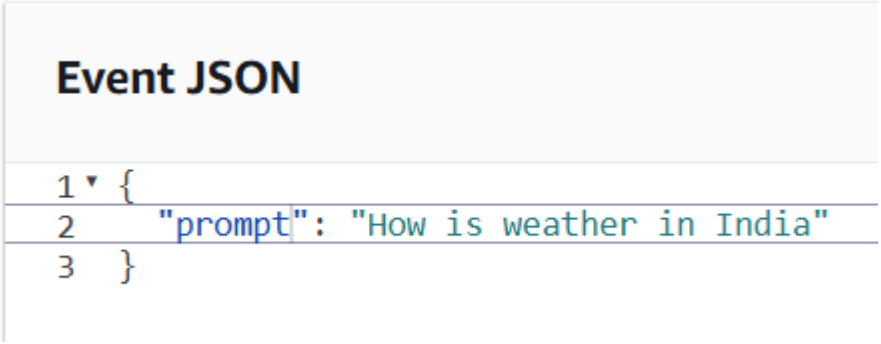


7. Then in your lambda function come to the code are and paste the code shown below.

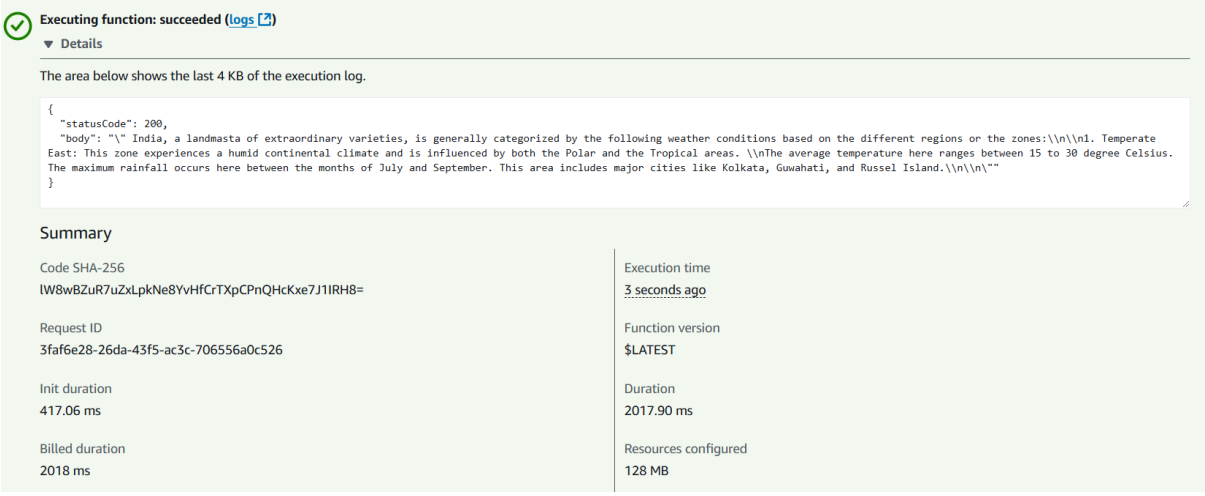
```
import json
#1 Import boto3 and create client connection with bedrock
import boto3
client_bedrock=boto3.client('bedrock-runtime')
#print(boto3.__version__)

def lambda_handler(event, context):
#2 a. Store the input in a variable, b. print the event
   input_prompt=event['prompt']
   print(input_prompt)

#3. Create  Request Syntax - Get details from console & body should be json object - use   json.dumps for body

   client_bedrockrequest=client_bedrock.invoke_model(
     contentType='application/json',
     accept='application/json',
     modelId='cohere.command-light-text-v14',
     body=json.dumps( {
      "prompt": input_prompt,
      "temperature": 0.9,
      "p": 0.75,
      "k": 0,
      "max_tokens": 100}))
   #print(client_bedrockrequest)

#4. Convert Streaming Body to Byte(.read method) and then Byte to String using json.loads#

   client_bedrock_byte=client_bedrockrequest['body'].read()
```

```
    #print(client_bedrock_byte)
    #print(type(client_bedrock_byte))
#5 a. Print the event and type , b. Store the input in a variable

    client_bedrock_string=json.loads(client_bedrock_byte)
    #print(client_bedrock_string)
#6. Update the 'return' by changing the 'body'
    client_final_response=client_bedrock_string['generations'][0]['text']
    print(client_final_response)

    return {
        'statusCode': 200,
        'body': json.dumps(client_final_response)
    }
```

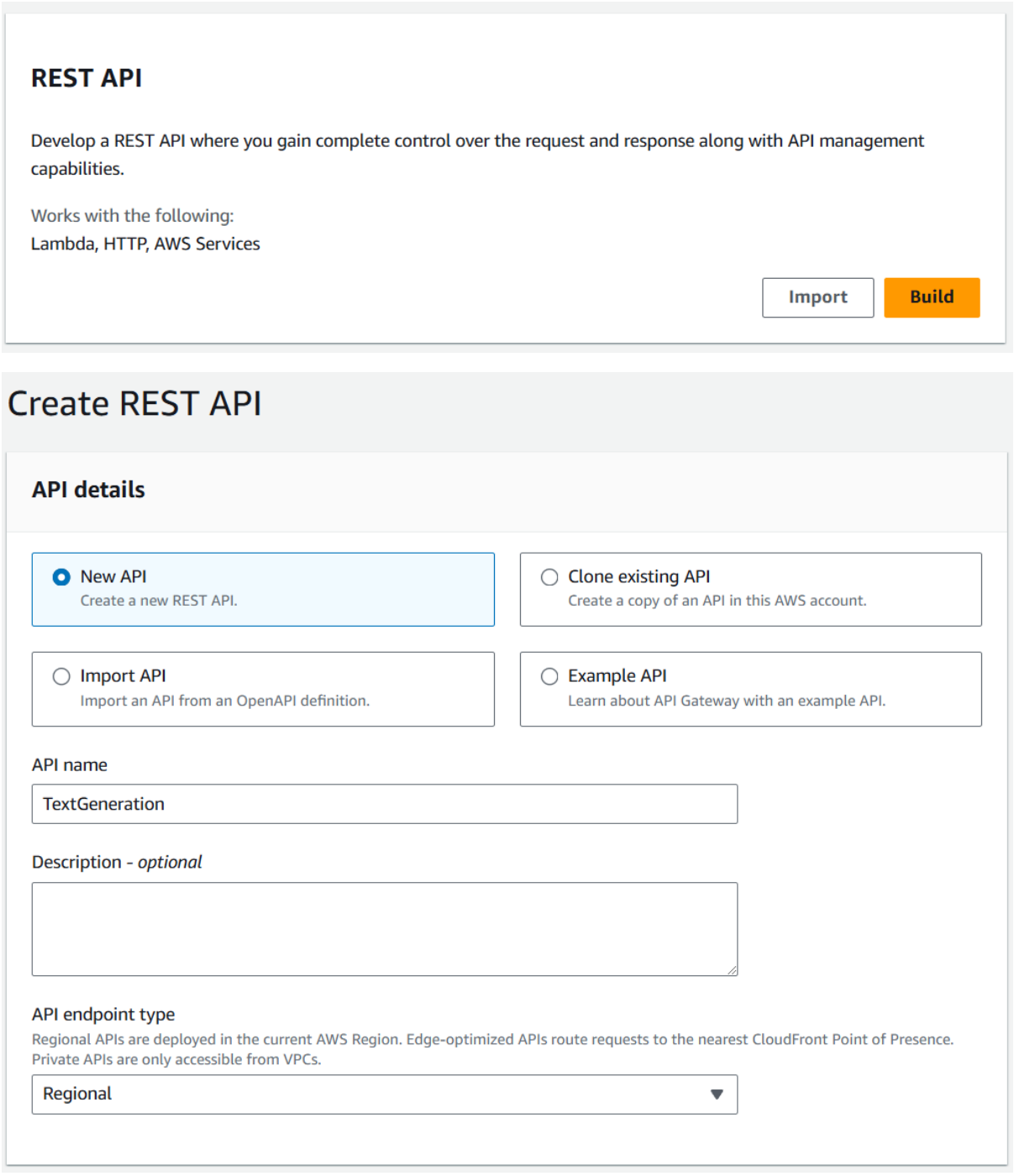
8. Now you need to click on deploy your code and create a new test event.
9. Then you need to create a test event and you need to write the same prompt or a prompt of your choice. Save it then click on test.



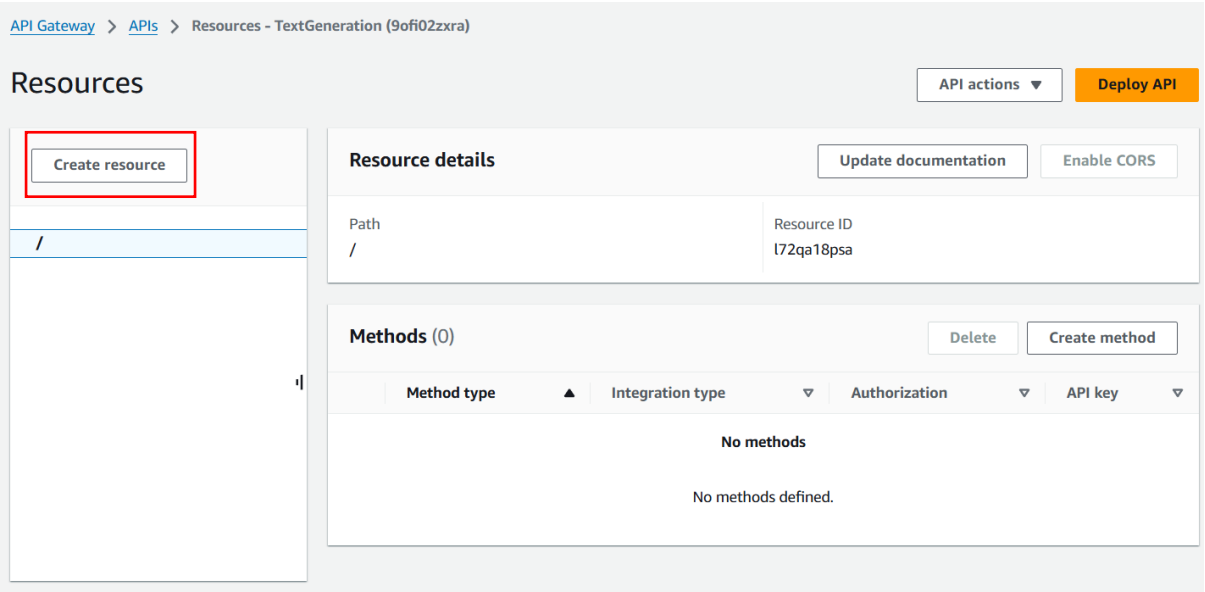
10. Below you can see that you get the status code 200 which means that we succeeded and also, we get the response as you can see below.



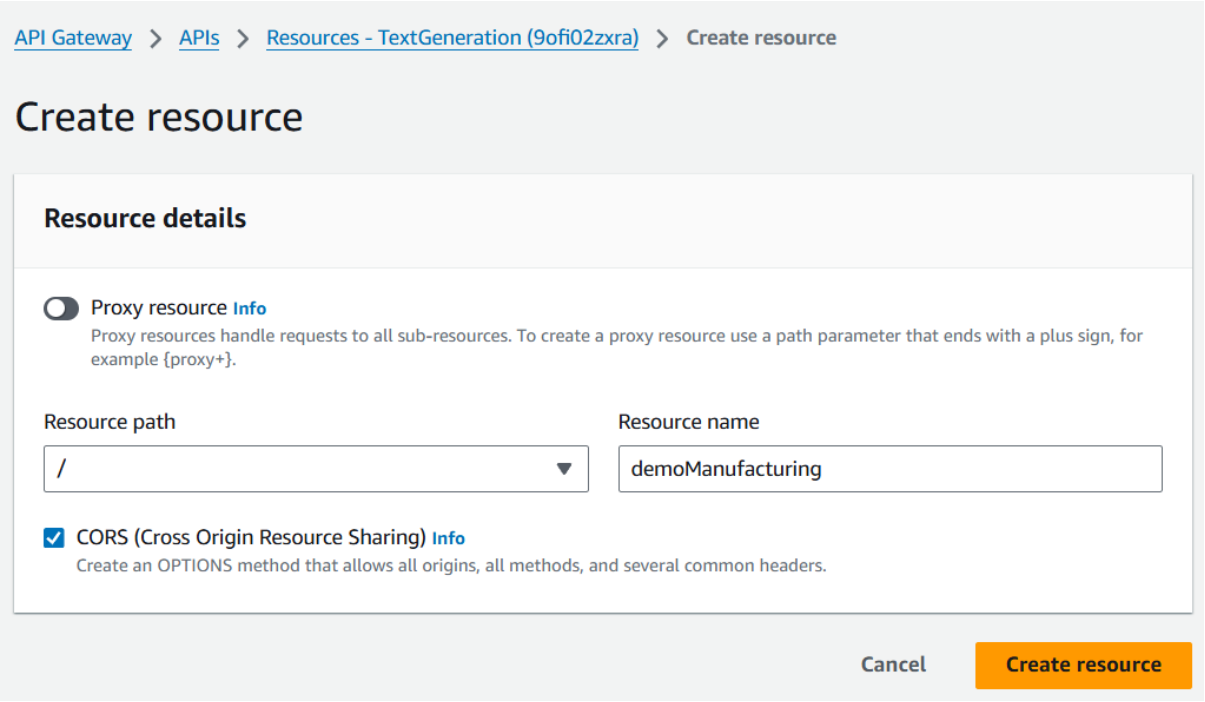
11. Now navigate to the API gateway and choose to create a REST API. Here you need to give your API a name and then click on create.



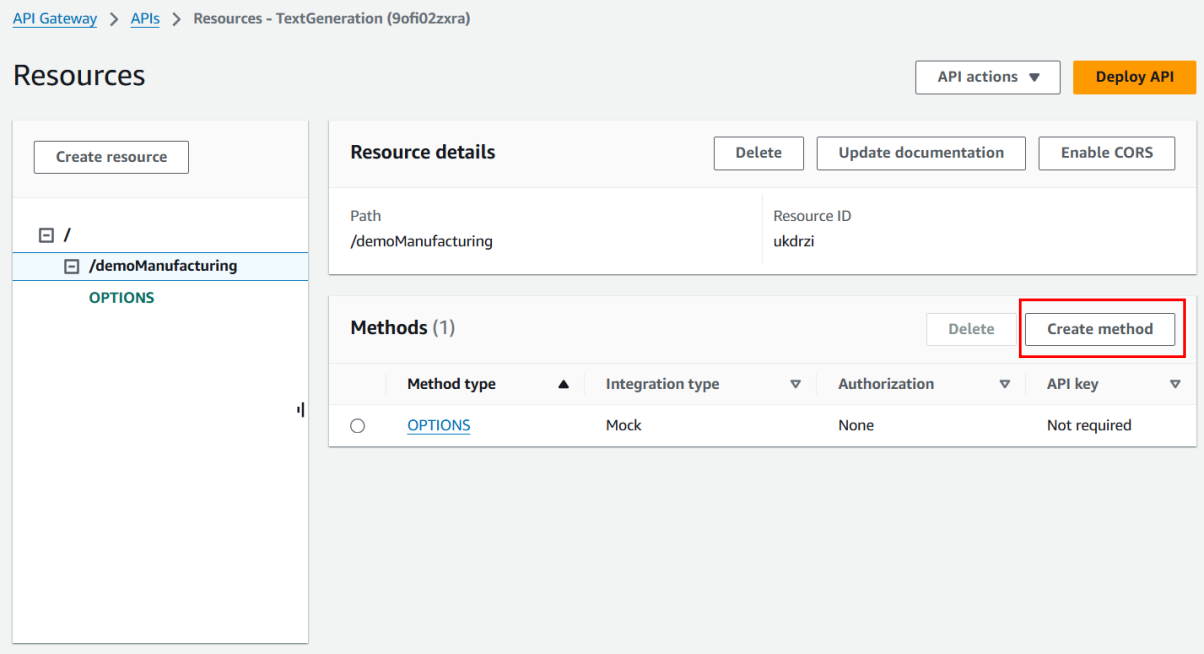
12. First, we need to create a resource. Click on Create resource.



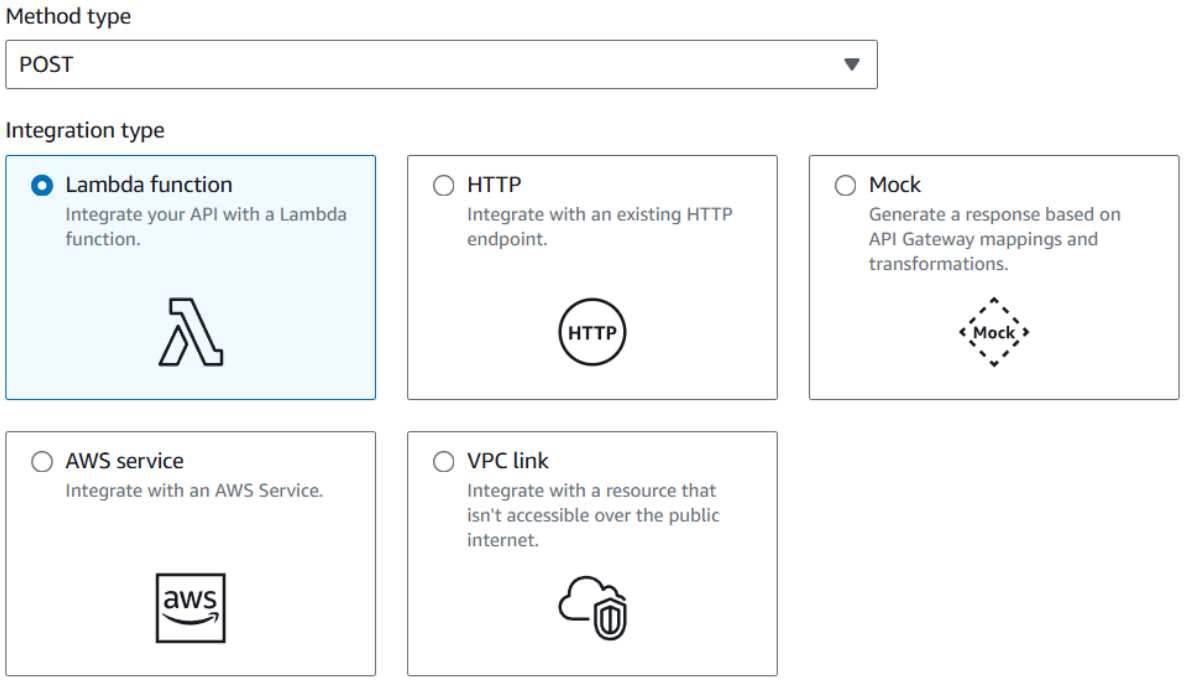
13. After that give your resource a name and enable CORS then click on Create Resource.



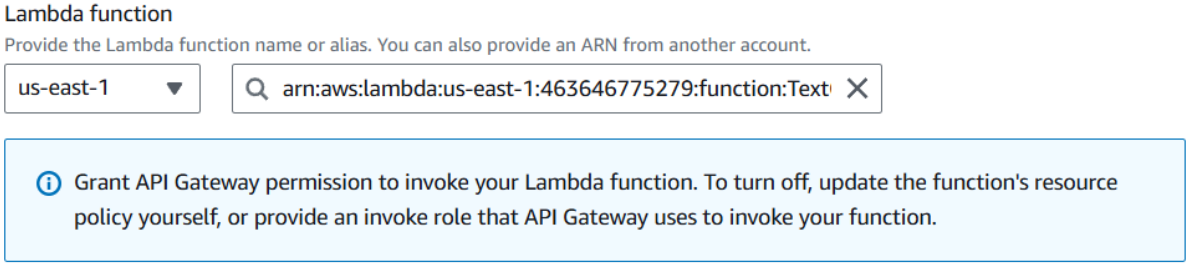
14. Then you need to choose your resource and click on create method.



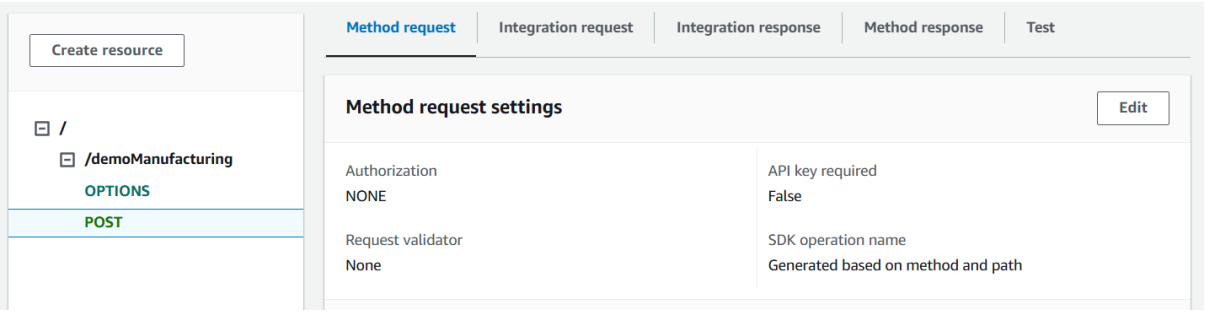
15. Choose the method type as POST and choose lambda in the integration type.



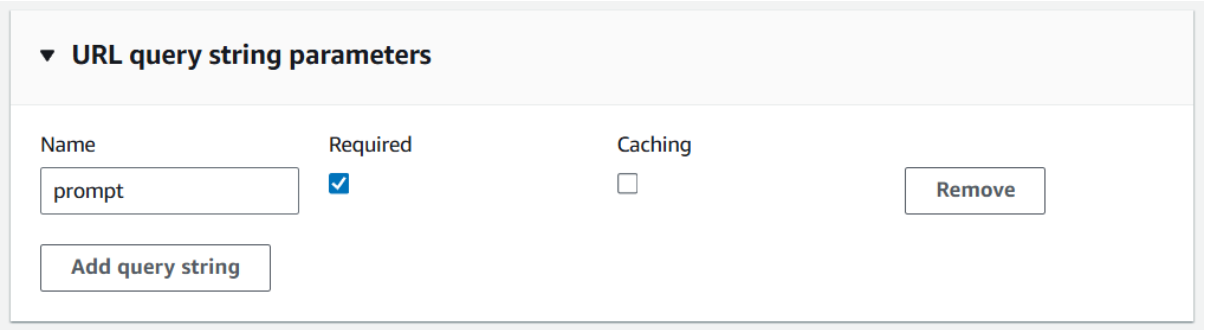
16. Then choose your lambda function and click on create method.



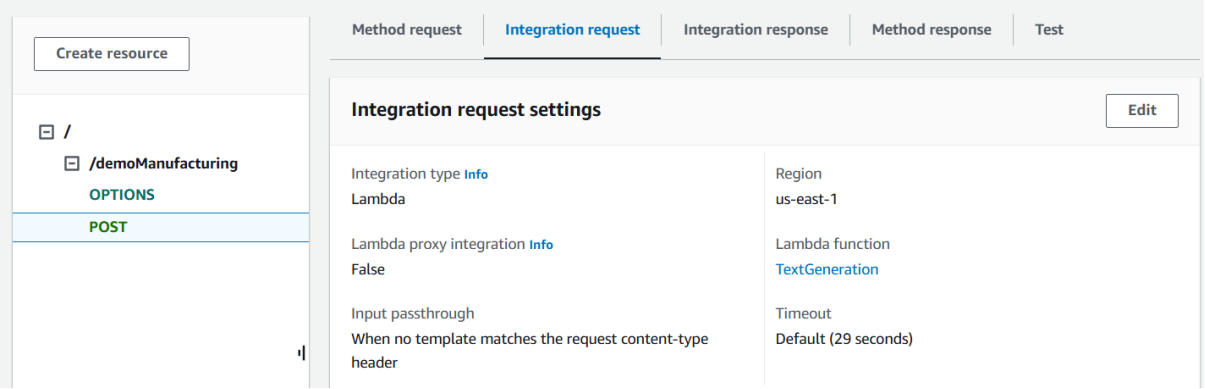
17. In your POST method choose method request and click on edit.



18. Here you need to expand URL query string parameters and click on add query string, in the name write prompt check the box for required, and click on save.

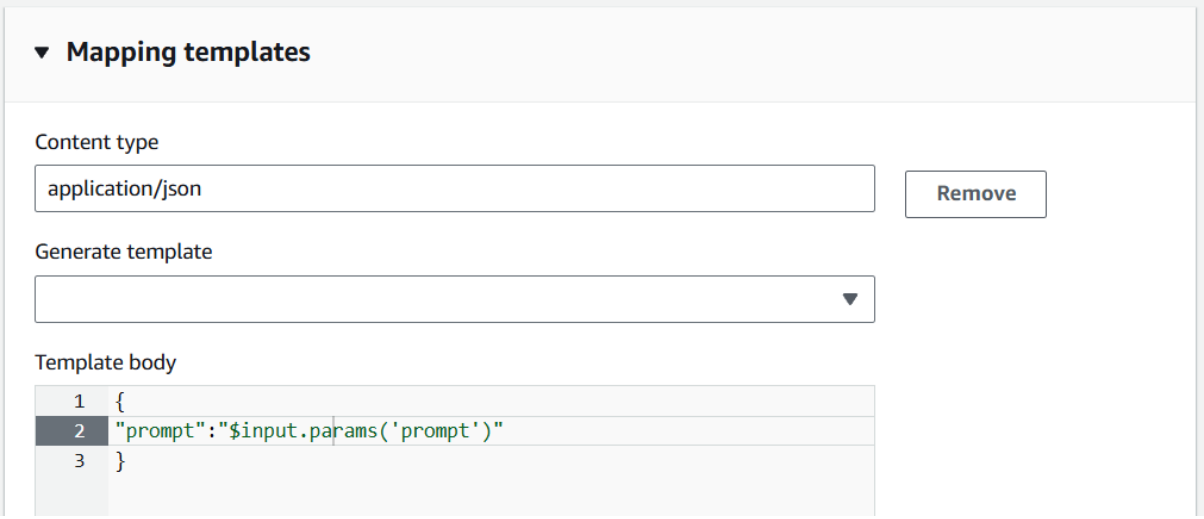


19. After that go to the integration request and click on edit.

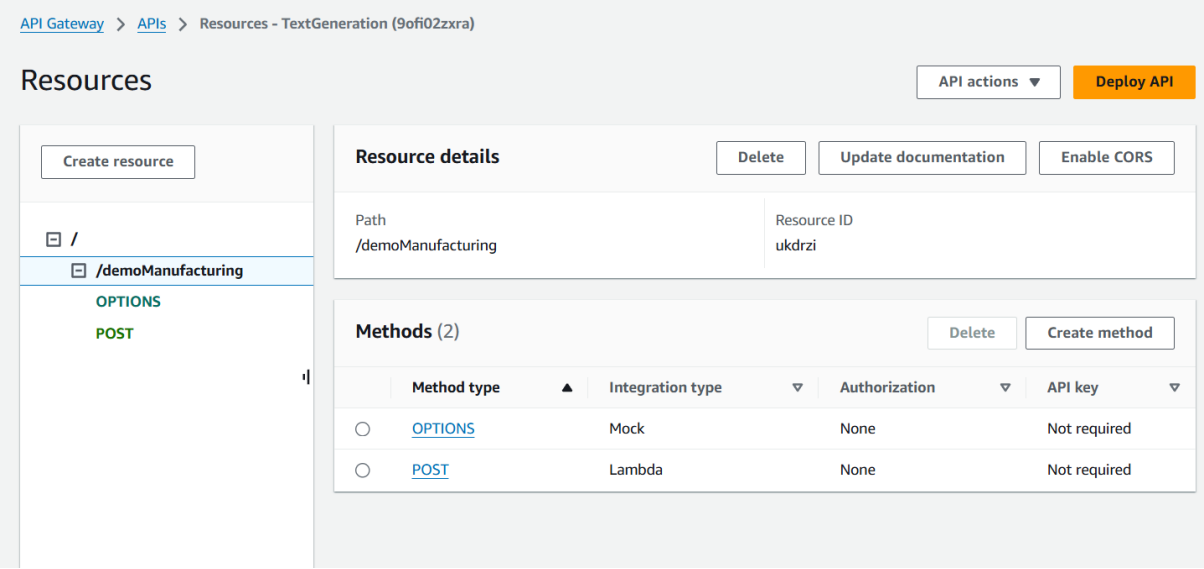


20. Scroll down to the bottom and expand mapping templates then write the same text as shown below.

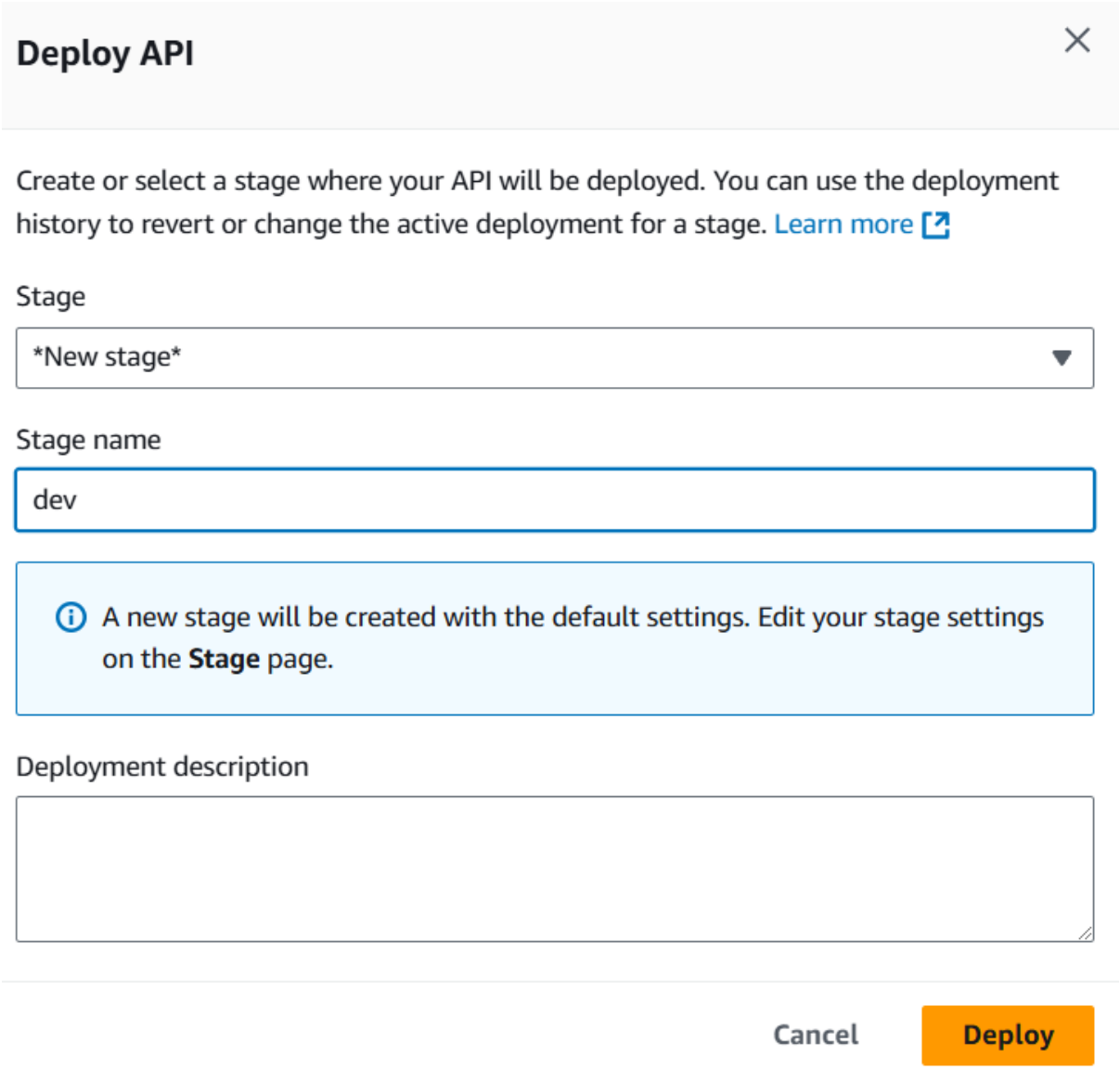
{
"prompt":"$input.params('prompt')"
}



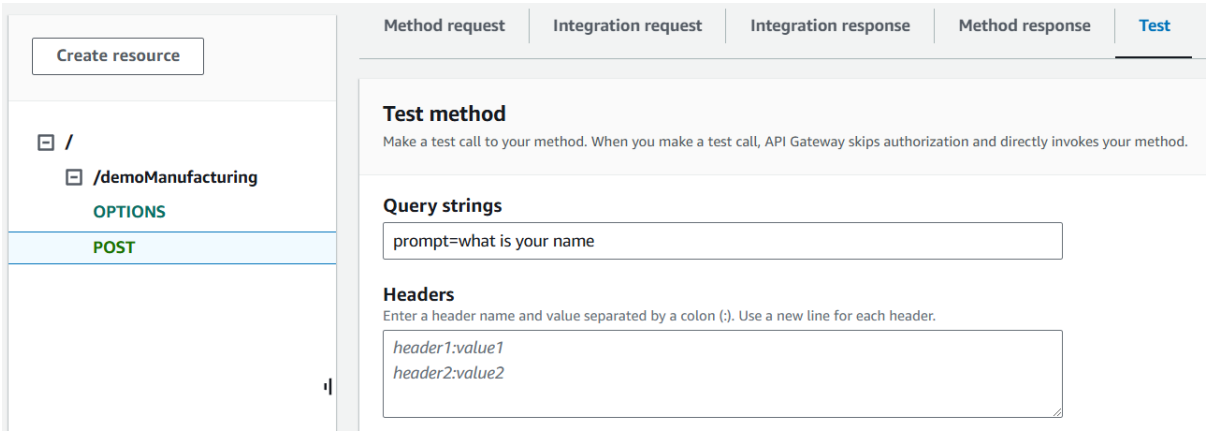
21. After that we are going to deploy our API so click on resource and then click on Deploy API.



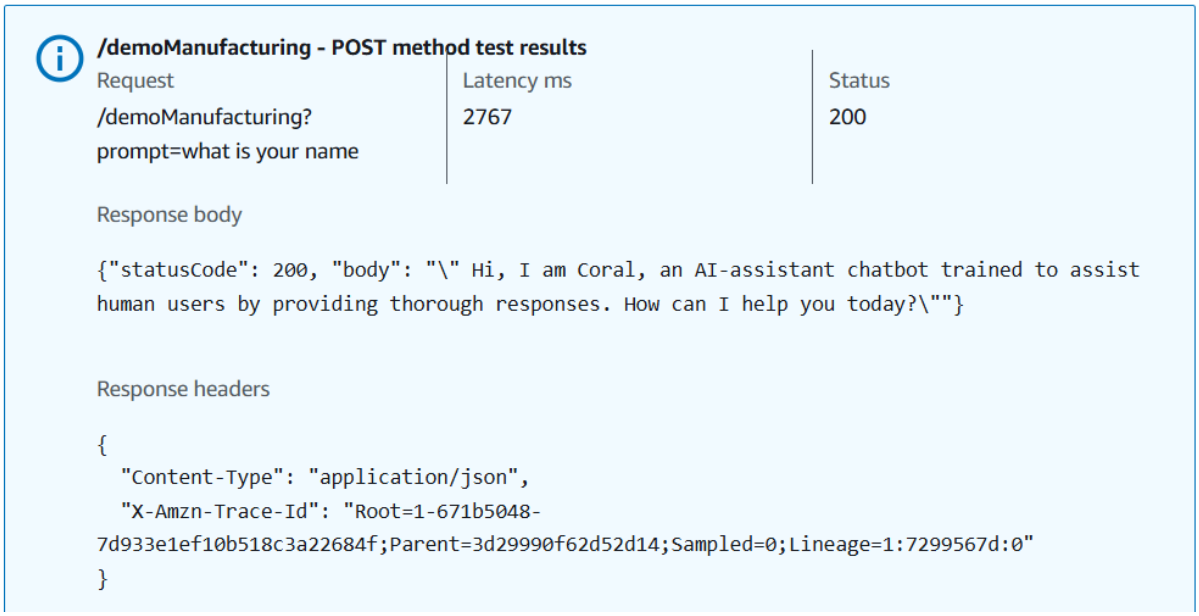
22. Then choose a new stage and give your stage a name then click on deploy.



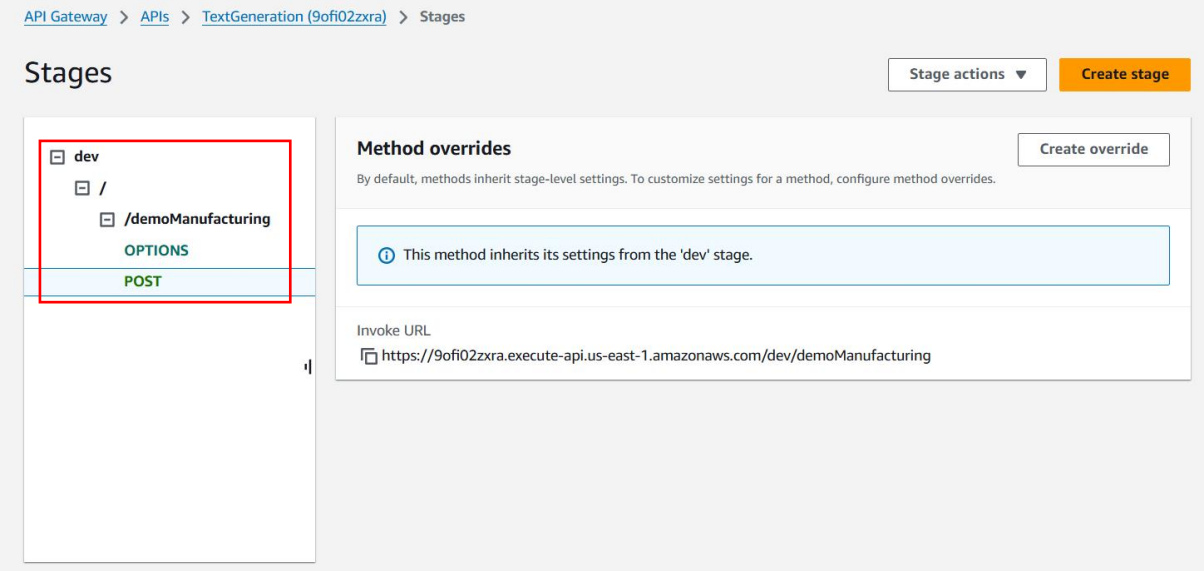
23. Once your API has been deployed you need to come to the POST method and click on Test then in the query strings you need to write the prompt in the same way. Then click on the test.



24. Below you can see that you get the response as expected.



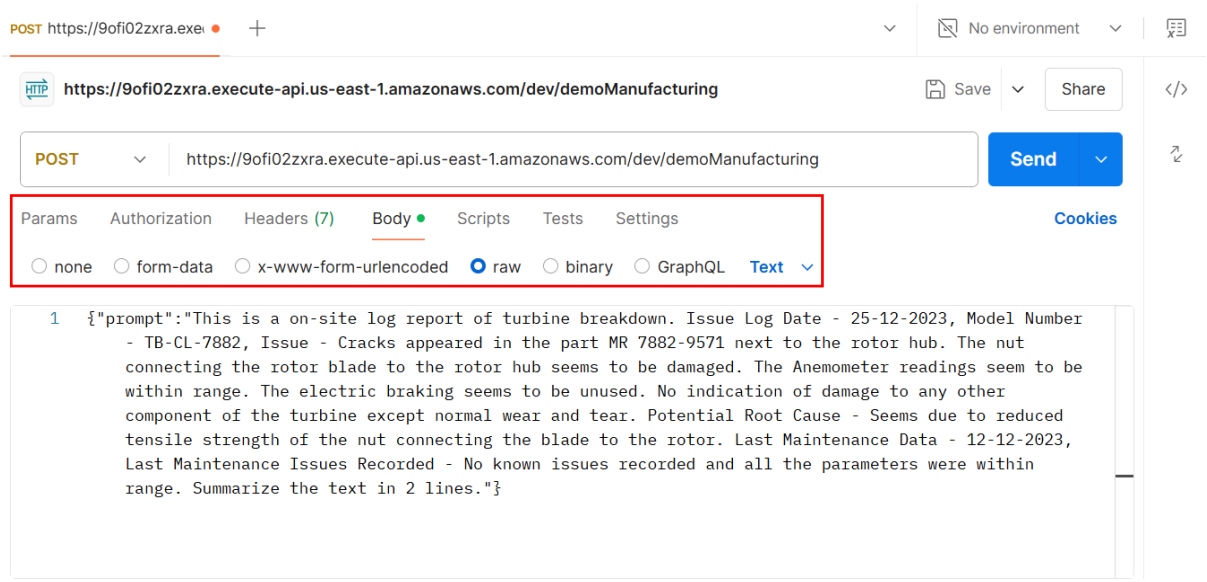
25. Now in your API gateway from the left pane choose stages and expand the dev stage until you get the POST method. Select it and copy the invoke URL.



26. Then open the Post Man tool on your laptop and create a new workspace. Here you need to POST method as you can see then paste the invoke URL in it.
27. Here you need to choose body then raw and change it text. After that past the same text mentioned below and click on Send.

{"prompt":"This is a on-site log report of turbine breakdown. Issue Log Date - 25-12-2023, Model Number - TB-CL-7882, Issue - Cracks appeared in the part MR 7882-9571 next to the rotor hub. The nut connecting the rotor blade to the rotor hub seems to be damaged. The Anemometer readings seem to be within range. The electric braking seems to be unused. No indication of damage to any other component of the turbine except normal wear and tear. Potential Root Cause - Seems due to reduced tensile strength of the nut connecting the blade to the rotor. Last Maintenance Data - 12-12-2023, Last Maintenance Issues Recorded - No

known issues recorded and all the parameters were within range. Summarize the text in 2 lines."}



28. Here you can see that you get the response in two lines.