



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.

▼ Bedrock configurations

Model access

Bedrock Studio [Preview](#)

Settings

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.

The screenshot shows the 'Create function' wizard. It starts with a summary step: 'Choose one of the following options to create your function.' Three options are listed:

- Author from scratch**: Start with a simple Hello World example. This option is selected.
- Use a blueprint**: Build a Lambda application from sample code and configuration presets for common use cases.
- Container image**: Select a container image to deploy for your function.

Below this, the 'Basic information' step is shown. It asks for a 'Function name' (e.g., 'TextGeneration') and specifies runtime and architecture settings. The runtime is set to 'Python 3.11' and the architecture to 'x86_64'.

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.

The screenshot shows the AWS Lambda Configuration page. The top navigation bar includes tabs for Code, Test, Monitor, Configuration (which is selected), Aliases, and Versions. On the left, a sidebar lists General configuration, Triggers, Permissions (selected), Destinations, and Environment variables. The main content area is titled "Execution role" and displays a "Role name" field containing "TextGeneration-role-suwrpb2f" with a blue edit icon. A red box highlights this role name field.

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

The screenshot shows the AWS Lambda Permissions policies page. The top navigation bar includes tabs for Permissions (selected), Trust relationships, Tags, Last Accessed, and Revoke sessions. Below the tabs, it says "Permissions policies (2) Info" and "You can attach up to 10 managed policies." A search bar and a "Filter by Type" dropdown are present. The main table lists two policies:

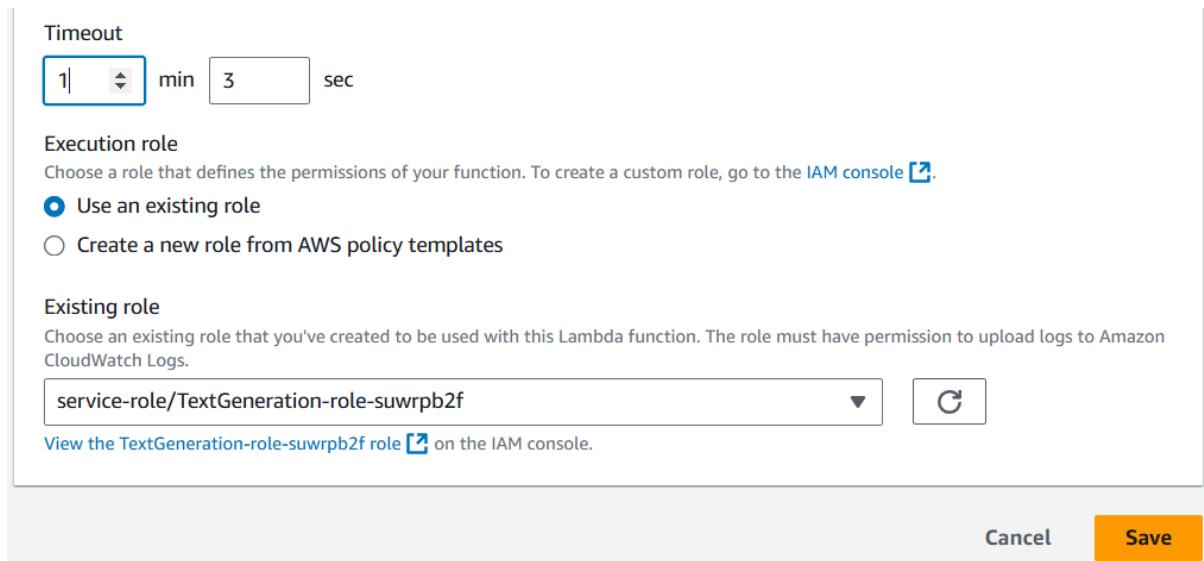
- AmazonBedrockFullAccess (AWS managed)
- AWSLambdaBasicExecutionRole-f47c4e7... (Customer managed)

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.

The screenshot shows the AWS Lambda General configuration page. The top navigation bar includes tabs for Code, Test, Monitor, Configuration (selected), Aliases, and Versions. On the left, a sidebar lists General configuration (selected), Triggers, Permissions, Destinations, Function URL, and Environment variables. The main content area is titled "General configuration" and shows the following configuration details:

| | | |
|-------------|-----------|-------------------|
| Description | Memory | Ephemeral storage |
| - | 128 MB | 512 MB |
| Timeout | SnapStart | |
| 0 min 3 sec | Info | |
| | None | |

An "Edit" button is located in the top right corner of the configuration table.



7. Then in your lambda function come to the code area 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.

Event JSON

```

1 ▾ {
2   "prompt": "How is weather in India"
3 }

```

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.

Executing function: succeeded ([logs](#) [?](#))

▼ Details

The area below shows the last 4 KB of the execution log.

```

{
  "statusCode": 200,
  "body": "\"India, a landmass of extraordinary varieties, is generally categorized by the following weather conditions based on the different regions or the zones:\\n\\n1. Temperate East: This zone experiences a humid continental climate and is influenced by both the Polar and the Tropical areas. \\n\\n2. The average temperature here ranges between 15 to 30 degree Celsius. The maximum rainfall occurs here between the months of July and September. This area includes major cities like Kolkata, Guwahati, and Russel Island.\\n\\n\""
}

```

| Summary | Execution time |
|--|----------------------|
| Code SHA-256 IW8wBZuR7uZxLpkNe8YvHfCrTXpCPnQHcKxe7J1IRH8= | 3 seconds ago |
| Request ID | Function version |
| 3faf6e28-26da-43f5-ac3c-706556a0c526 | \$LATEST |
| Init duration | Duration |
| 417.06 ms | 2017.90 ms |
| Billed duration | Resources configured |
| 2018 ms | 128 MB |

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.

REST API

Develop a REST API where you gain complete control over the request and response along with API management capabilities.

Works with the following:
Lambda, HTTP, AWS Services

[Import](#)

[Build](#)

Create REST API

API details

New API

Create a new REST API.

Clone existing API

Create a copy of an API in this AWS account.

Import API

Import an API from an OpenAPI definition.

Example API

Learn about API Gateway with an example API.

API name

TextGeneration

Description - *optional*

API endpoint type

Regional APIs are deployed in the current AWS Region. Edge-optimized APIs route requests to the nearest CloudFront Point of Presence.
Private APIs are only accessible from VPCs.

Regional

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

The screenshot shows the AWS API Gateway Resources page. At the top, there is a breadcrumb navigation: API Gateway > APIs > Resources - TextGeneration (9ofi02zxra). Below the breadcrumb, the word "Resources" is displayed. On the left, there is a sidebar with a "Create resource" button highlighted by a red box. The main area is titled "Resource details" and contains the following information:

- Path: /
- Resource ID: l72qa18psa

Below this is a section titled "Methods (0)" with a "Create method" button. A table header for methods includes columns for Method type, Integration type, Authorization, and API key. The body of the table says "No methods" and "No methods defined."

At the top right of the main area are "API actions" and "Deploy API" buttons.

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

The screenshot shows the "Create resource" dialog. At the top, there is a breadcrumb navigation: API Gateway > APIs > Resources - TextGeneration (9ofi02zxra) > Create resource. The title of the dialog is "Create resource".

The "Resource details" section contains the following fields:

- Proxy resource [Info](#): A note explains that proxy resources handle requests to all sub-resources. To create a proxy resource, use a path parameter that ends with a plus sign, for example {proxy+}.
- Resource path: /
- Resource name: demoManufacturing
- CORS (Cross Origin Resource Sharing) [Info](#): A note explains that this creates an OPTIONS method allowing all origins, all methods, and several common headers.

At the bottom right are "Cancel" and "Create resource" buttons.

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

API Gateway > APIs > Resources - TextGeneration (9ofi02zxra)

Resources

[Create resource](#)

| Resource details | | Delete | Update documentation | Enable CORS |
|--|--------------------|------------------------|--------------------------------------|-----------------------------|
| Path | /demoManufacturing | Resource ID | ukdrzi | |
| Methods (1) Delete Create method | | | | |
| Method type | Integration type | Authorization | API key | |
| <input type="radio"/> OPTIONS | Mock | None | Not required | |

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

Method type

POST

Integration type

| | | |
|---|--|---|
| <input checked="" type="radio"/> Lambda function Integrate your API with a Lambda function.  | <input type="radio"/> HTTP Integrate with an existing HTTP endpoint.  | <input type="radio"/> Mock Generate a response based on API Gateway mappings and transformations.  |
| <input type="radio"/> AWS service Integrate with an AWS Service.  | <input type="radio"/> VPC link Integrate with a resource that isn't accessible over the public internet.  | |

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

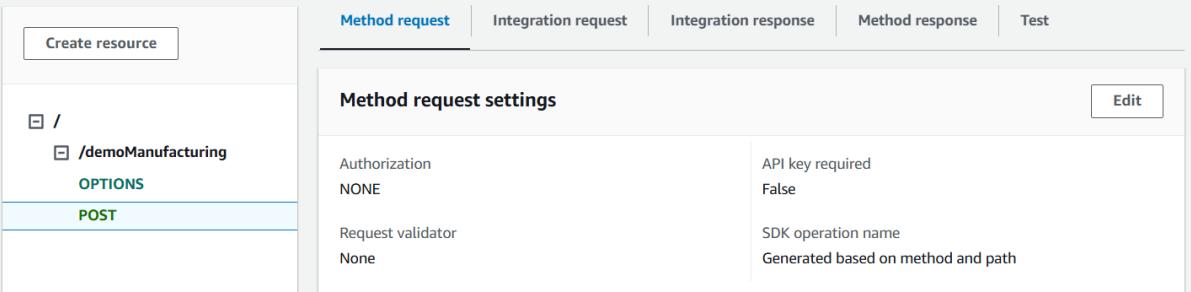
Lambda function

Provide the Lambda function name or alias. You can also provide an ARN from another account.

us-east-1  

 Grant API Gateway permission to invoke your Lambda function. To turn off, update the function's resource policy yourself, or provide an invoke role that API Gateway uses to invoke your function.

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

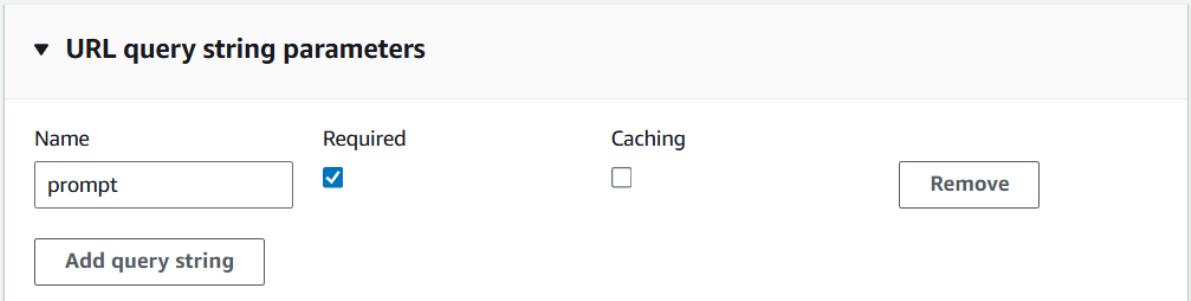


The screenshot shows the 'Method request' tab selected in the top navigation bar. On the left, a tree view shows a root node with a child node '/demoManufacturing' containing 'OPTIONS' and 'POST' methods. The 'POST' method is selected. On the right, the 'Method request settings' section is displayed with the following configuration:

| Setting | Value |
|--------------------|------------------------------------|
| Authorization | NONE |
| Request validator | None |
| API key required | False |
| SDK operation name | Generated based on method and path |

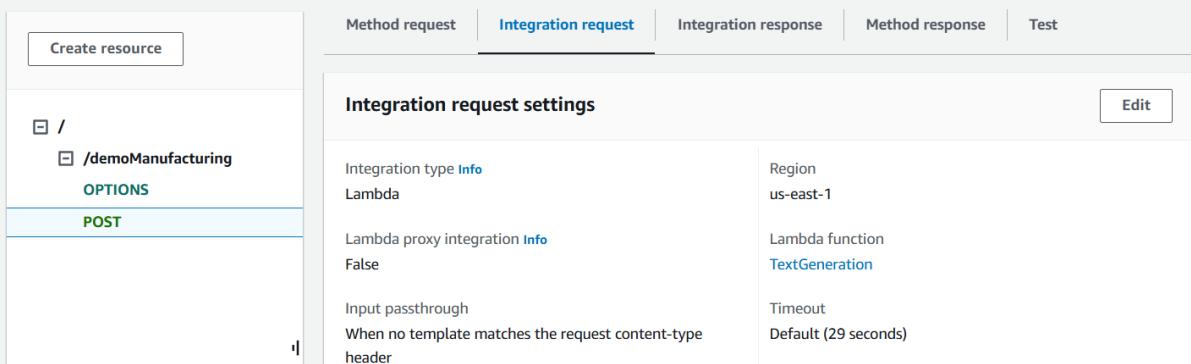
An 'Edit' button is located in the top right corner of the settings area.

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.



The screenshot shows the 'URL query string parameters' section. It contains one entry: 'prompt' with 'Required' checked and 'Caching' unchecked. A 'Remove' button is available to the right of the entry. Below the table is a 'Add query string' button.

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



The screenshot shows the 'Integration request' tab selected in the top navigation bar. On the left, the same tree view is shown with the 'POST' method selected. On the right, the 'Integration request settings' section is displayed with the following configuration:

| Setting | Value |
|--|--------------------------|
| Integration type | Info |
| Lambda | Lambda proxy integration |
| False | Region |
| Input passthrough | us-east-1 |
| When no template matches the request content-type header | Lambda function |
| | TextGeneration |
| | Timeout |
| | Default (29 seconds) |

An 'Edit' button is located in the top right corner of the settings area.

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

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

▼ Mapping templates

Content type

application/json

Remove

Generate template



Template body

```
1 {
2 "prompt": "$input.params('prompt')"
3 }
```

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

The screenshot shows the AWS API Gateway Resources page. The URL in the address bar is [API Gateway > APIs > Resources - TextGeneration \(9ofi02zxra\)](#). The main area is titled "Resources". On the left, there's a tree view showing a root node "/" with a child node "/demoManufacturing". Under "/demoManufacturing", there are two methods listed: "OPTIONS" and "POST". On the right, the "Resource details" section shows the path `/demoManufacturing` and the resource ID `ukdrzi`. Below this, the "Methods (2)" section lists the two methods with their respective configurations. A prominent orange button labeled "Deploy API" is located at the top right of the main content area.

| Method type | Integration type | Authorization | API key |
|-------------|------------------|---------------|--------------|
| OPTIONS | Mock | None | Not required |
| POST | Lambda | None | Not required |

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

Deploy API



Create or select a stage where your API will be deployed. You can use the deployment history to revert or change the active deployment for a stage. [Learn more](#)

Stage

New stage



Stage name

dev

i A new stage will be created with the default settings. Edit your stage settings on the [Stage](#) page.

Deployment description

Cancel

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.

The screenshot shows the AWS API Gateway 'Test method' interface. On the left, there's a sidebar with a 'Create resource' button and a tree view showing a root node with a child node '/demoManufacturing'. Underneath it, the 'OPTIONS' and 'POST' methods are listed, with 'POST' being the currently selected method (indicated by a blue background). The main content area has five tabs at the top: 'Method request', 'Integration request', 'Integration response', 'Method response', and 'Test'. The 'Test' tab is active and highlighted in blue. Below the tabs, there's a section titled 'Test method' with the sub-instruction 'Make a test call to your method. When you make a test call, API Gateway skips authorization and directly invokes your method.' There are two input fields: 'Query strings' containing 'prompt=what is your name' and 'Headers' containing 'header1:value1' and 'header2:value2'.

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

/demoManufacturing - POST method test results

| Request | Latency ms | Status |
|---|------------|--------|
| /demoManufacturing? prompt=what is your name | 2767 | 200 |

Response body

```
{"statusCode": 200, "body": "\" Hi, I am Coral, an AI-assistant chatbot trained to assist human users by providing thorough responses. How can I help you today?\""} 
```

Response headers

```
{
  "Content-Type": "application/json",
  "X-Amzn-Trace-Id": "Root=1-671b5048-7d933e1ef10b518c3a22684f;Parent=3d29990f62d52d14;Sampled=0;Lineage=1:7299567d:0"
}
```

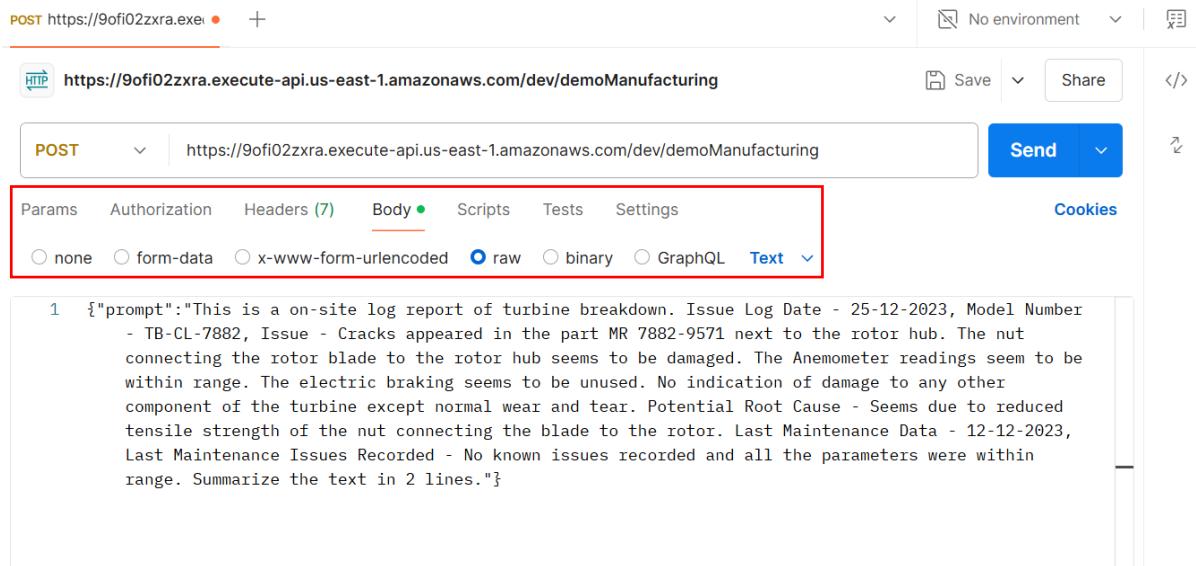
- Now in your API gateway from the left pane choose stages and expand the dev stage until you get the POST method. Select it and copy the invoke URL.

The screenshot shows the AWS API Gateway interface. In the top navigation bar, the path is: API Gateway > APIs > TextGeneration (9ofi02zxra) > Stages. On the right, there are buttons for 'Stage actions' and 'Create stage'. The main area is titled 'Stages' and shows a tree view of stages: dev, /, and /demoManufacturing. Under /demoManufacturing, the 'OPTIONS' and 'POST' methods are listed. A red box highlights the /demoManufacturing node. To the right, there's a 'Method overrides' section with a note that the method inherits settings from the 'dev' stage. Below it is an 'Invoke URL' field containing the URL: https://9ofi02zxra.execute-api.us-east-1.amazonaws.com/dev/demoManufacturing.

- 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.
- 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."}



POST https://9ofi02zxra.execute-api.us-east-1.amazonaws.com/dev/demoManufacturing

Params Authorization Headers (7) Body Scripts Tests Settings Cookies

none form-data x-www-form-urlencoded raw binary GraphQL Text

```
1 {"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.



Body Cookies Headers (7) Test Results

200 OK • 1037 ms • 524 B •

Pretty Raw Preview Visualize JSON

```
1 {
2   "statusCode": 200,
3   "body": "\\" The Model Number TB-CL-7882 has reported issues of cracks in the part, potentially due to reduced tensile strength, and is waiting on further analysis.\\""
4 }
```