Architecture

Services used

1. API Gateway
2. Lambda
3. Bedrock
4. IAM
5. Amplify

API Gateway will serve as the front door to your application. Every request to our application will go through API Gateway

Lambda function will serve as the backend for our application. This is where we will have our code, call other services, and return a response to the user.

Bedrock is the LLM Catalog for Large Language Models (LLMs). We will use Bedrock interfaces to generate a response to our user.

IAM is the service that controls the permissions and user roles. We will use this to ensure our Lambda will have the necessary permissions to invoke our Large Language Models

Amplify is the service we will use to deliver our frontend to the users.

Tutorial

Lambda

1. Start by creating an AWS Skill Builder account.
    a. https://skillbuilder.aws/
2. After the sign up is complete, navigate to AWS console.
    a. https://us-east-1.console.aws.amazon.com/console/home?region=us-east-1
3. Search for Lambda
    a. This service will be used to create the logic of the application.
4. Create a lambda function
    a. Give the function a name (ex, svlsttrl)
    b. Select python 3.14 for the runtime.
    c. Leave everything default and click **Create Function**
5. Click on **Configuration** tab
    a. Edit **General configuration** and increase the timeout to 30 seconds
    b. Select **Permissions** from the left menu

i. Click on the **role name** (blue link)
ii. This will open the IAM role in a new tab
iii. Click **Add permissions** and select **attach policies**
iv. Search for **AmazonBedrockLimitedAccess** and attach it to the role
v. Feel free to close this tab now

API Gateway

1. Search for **API Gateway** and open it in a new page
2. Click **create API** and select **Build** for **REST API**
    a. For **API details** select **New API**
    b. Give the API a name
    c. Leave everything else as default and click build
3. In the new console, click **Create Method** on the right side
    a. For **Method type** select **ANY**
    b. For **Integration type** select **Lambda function**
    c. Toggle **Lambda proxy integration** to turn it on
    d. For **Lambda function** select the lambda we created in the first step.
    e. Click **Create Method**
4. Under **Resources** click on the **slash (/)**
    a. Click enable CORS button on the top right
    b. Checkmark the **Default 4XX**
    c. Checkmark the **Default 5XX**
    d. Checkmark GET
    e. Delete everything from **Access-Control-Allow-Headers** box and replace it with ***
    f. Click Save
5. In the new page click **Deploy API**
    a. For the **Stage** select **New stage**
    b. Enter string **Prod**
6. In the new page, note the **Invoke URL. We will need this for later**

Test One (optional)

1. Open the **Invoke URL** link in a new tab.
2. You should see a message the says **Hello from Lambda!**
3. This proves that so far we are able to reach the backend from our API Gateway

Invoke a model

1. Navigate to the lambda function again
2. Replace the lambda code with the code from the GitHub
    a. https://github.com/aws-gmu/serverless/blob/main/lambda-basic
3. Click on **Configuration** tab
    a. Select **Environment Variables**
    b. Click Edit > Add Environment Variable
    c. For the **Key**, insert BEDROCK_MODEL_ID
    d. For the **Value,** insert **us.anthropic.claude-opus-4-20250514-v1:0**
    e. Click **Save**

Test 2 (Optional)

1. On MAC, Open Terminal
    a. For Windows: Open PowerShell
2. Paste the following command and change the **<<Invoke URL>>** to your API Gateway URL

```
==========================================================================
curl -X POST https://mqwy5lemvl.execute-api.us-east-1.amazonaws.com/prod \
 -H "Content-Type: application/json" \
 -d '{
  "prompt": "Explain what AWS Lambda is in simple terms."
 }'
==========================================================================
```

3. You should get a response that'd contain the model response

Frontend

Now that our backend works, we need to create a frontend to process the response of the API Gateway. Thus, we'll create a chat interface.

1. Download the file - https://github.com/aws-gmu/serverless/blob/main/index-basic.html
2. Edit the file using a text editor
    a. Find **<<INVOKE_URL>>** and **replace** it with your API Gateway Invoke URL
    b. Save the file
    c. Open the file using a web browser
3. On AWS Console navigate to **Lambda**
4. Open the lambda you created earlier

5. Update its code to be based on the GitHub file - https://github.com/aws-gmu/serverless/blob/main/Lambda

6. Start chatting with the bot you just created

If you note, the bot does not maintain chat history so it can't keep a conversation up. Thus, we need to update the frontend and backend to maintain a chat history.

Update Bot

1. On AWS Console navigate to **Lambda**
2. Open the lambda you created earlier
3. Update its code to be based on the GitHub file - https://github.com/aws-gmu/serverless/blob/main/lambda2
4. Download the updated HTML file from here - https://github.com/aws-gmu/serverless/blob/main/index.html
5. Edit the **index.html** file you just downloaded using a text editor
    a. Find **<<INVOKE_URL>>** and **replace** it with your API Gateway Invoke URL
    b. Save the file
    c. Open the file using a web browser
6. Start chatting with the Bot

Congrats! You just built yourself a chatbot