Serverless Real-time Chat Application with Global Distribution

A Serverless Real-time Chat Application with Global Distribution is a chat app that allows users from all over the world to send and receive messages instantly, without you having to manage any servers.

What is a Serverless Real-time Chat Application?

- Serverless: You don't manage any servers. AWS handles the infrastructure, so you only focus on your code.
- Real-time: Messages are sent and received instantly, without delays.
- Global Distribution: The chat app can be accessed from anywhere in the world, with low latency

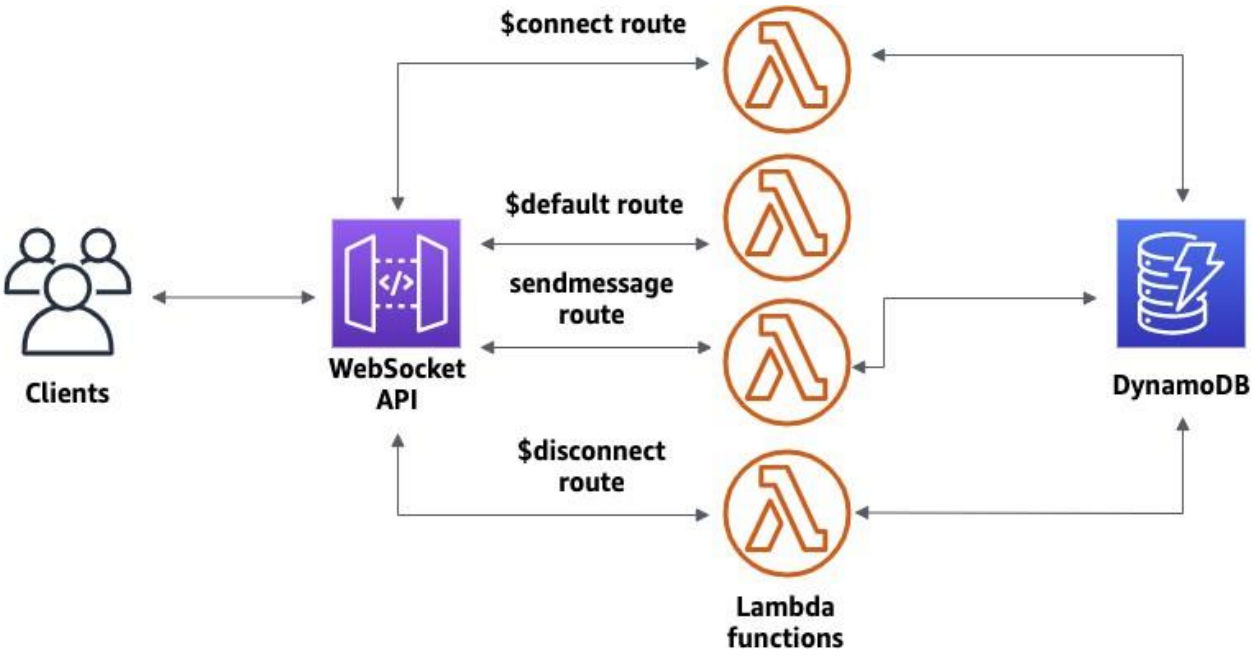
Prerequisites

- An AWS account
- Basic understanding of AWS Lambda, DynamoDB, and API Gateway
- AWS CLI and SAM CLI installed (optional, if you're using CloudFormation)

Key Components Involved

- **API Gateway (WebSocket)**: Manages the connections between clients (chat users). It enables two-way communication.
- **AWS Lambda**: Small pieces of code (functions) that execute when something happens (like when a user sends a message).
- **Amazon DynamoDB**: A fast, NoSQL database to store data like user connections or chat messages.
- **Amazon CloudFront**: A Content Delivery Network (CDN) that speeds up access to your app by serving it from servers around the world.
- **Amazon S3**: Storage for your web app files (HTML, JavaScript, etc.), which CloudFront can distribute globally.

First, you'll use an AWS CloudFormation template to create Lambda functions that will handle API requests, as well as a DynamoDB table that stores your client IDs. Then, you'll use the API Gateway console to create a WebSocket API that integrates with your Lambda functions. Lastly, you'll test your API to verify that messages are sent and received.



Step 1: Create Lambda Functions and a DynamoDB Table

In this step, we will create the necessary backend components for your serverless chat application. These components include Lambda functions to manage client connections and a DynamoDB table to store each client's unique ID.

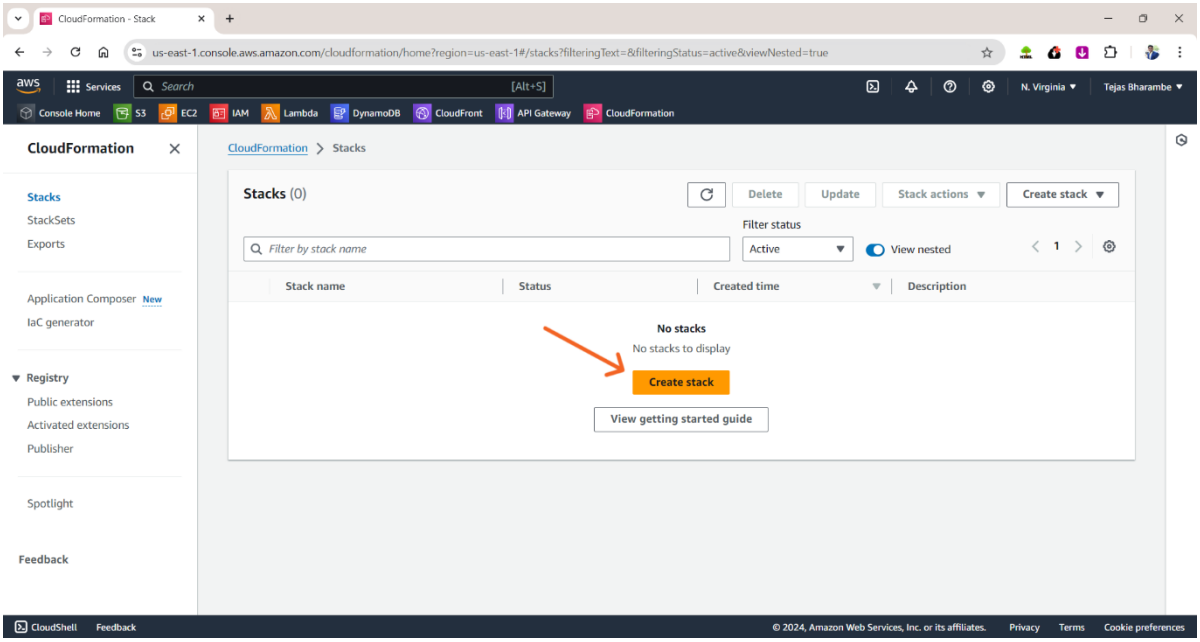
Download and Unzip the CloudFormation Template

https://docs.aws.amazon.com/apigateway/latest/developerguide/samples/ws-chat-app-starter.zip

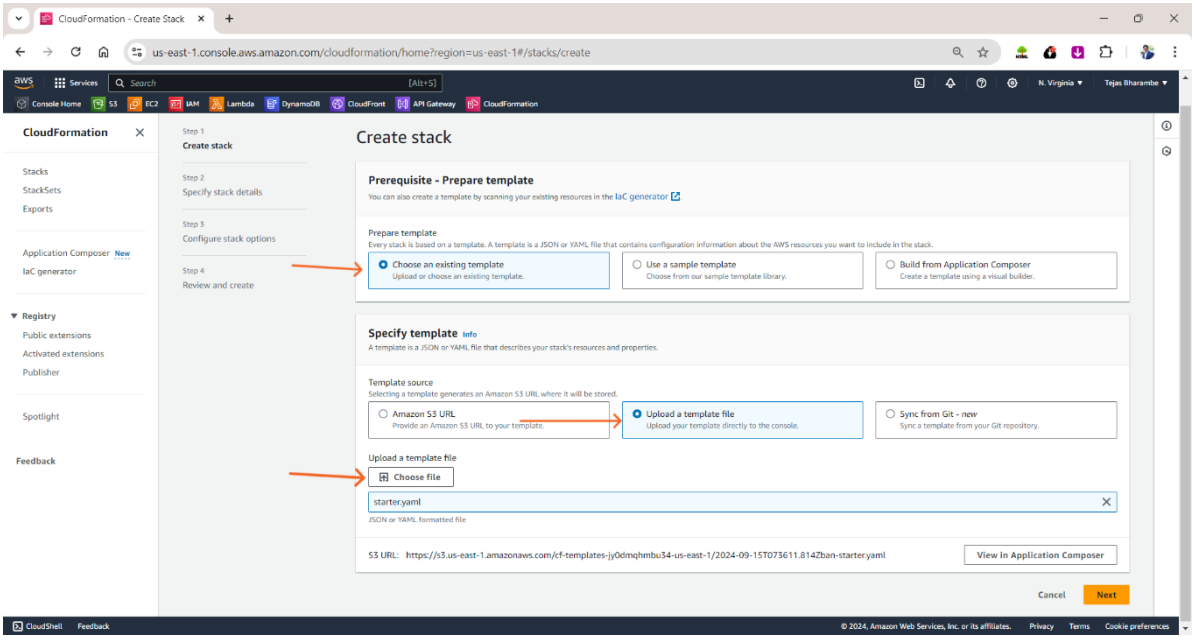
1. **Download the Template**:
   - Obtain the CloudFormation template file that will set up your DynamoDB table and Lambda functions. This file is usually provided in .yaml and .json format.
2. **Unzip the Template**:
   - If the template is in a compressed format (e.g.zip), unzip it to access the .yaml or .json file.

Deploy the CloudFormation Stack

1. Open AWS Management Console**:**
   - Go to the AWS Management Console.
2. Navigate to CloudFormation**:**
   - In the AWS Management Console, search for and select "CloudFormation".
3. Create a New Stack**:**
   - Click on **"Create stack"** and then choose **"With new resources (standard)"**.



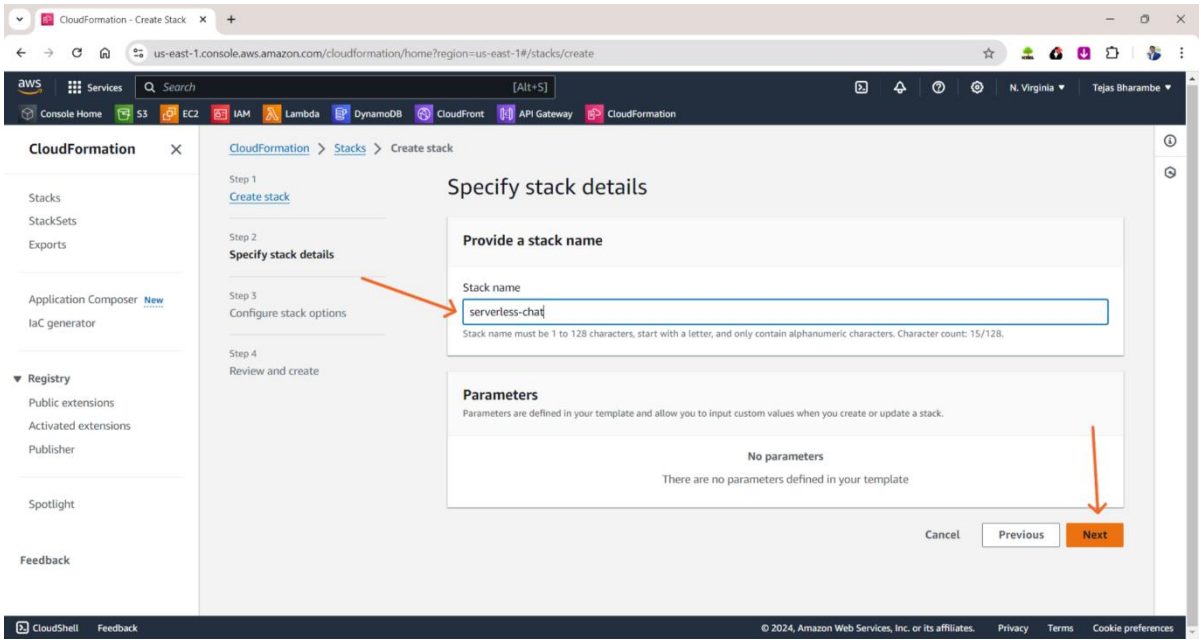
4. Upload the Template File**:**
   - In the "Specify template" section, choose **"Upload a template file"**.



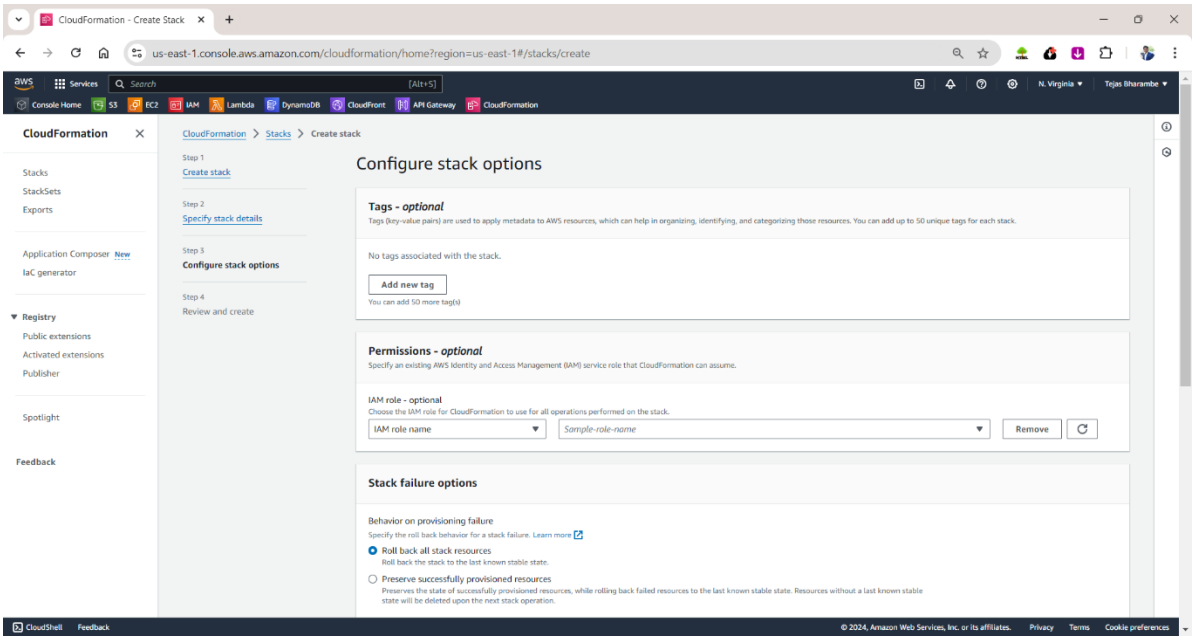
   - Select the .yaml or .json file you unzipped earlier.
   - Choose Next.

5. Configure the Stack:

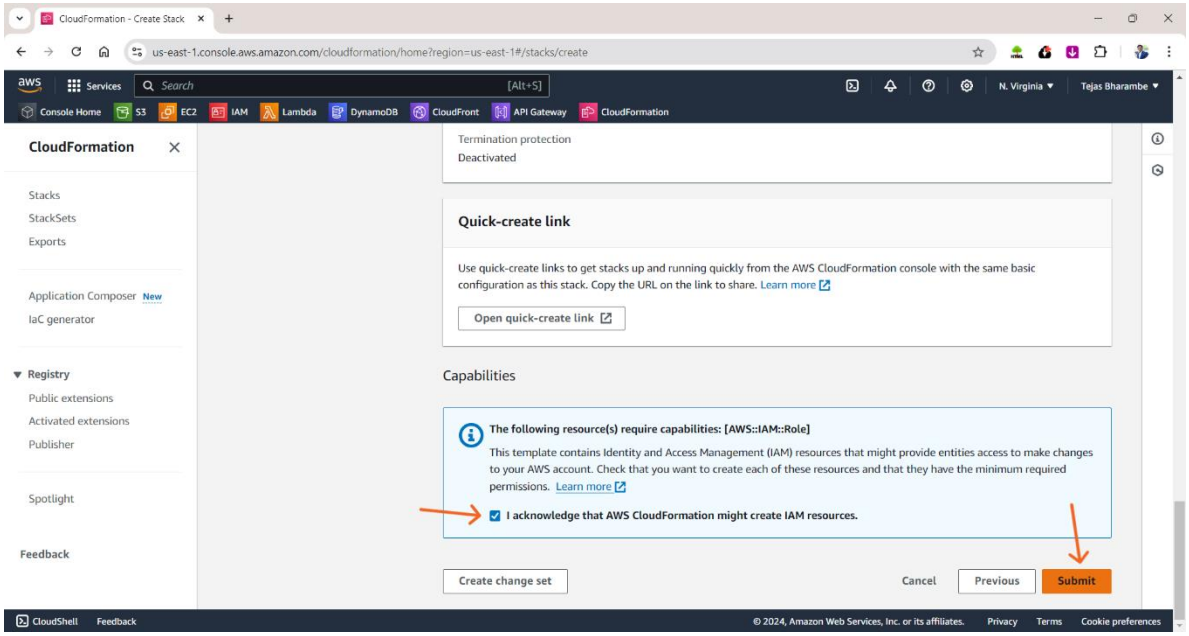
- Give your stack a name, such as serverless-chat and then choose Next



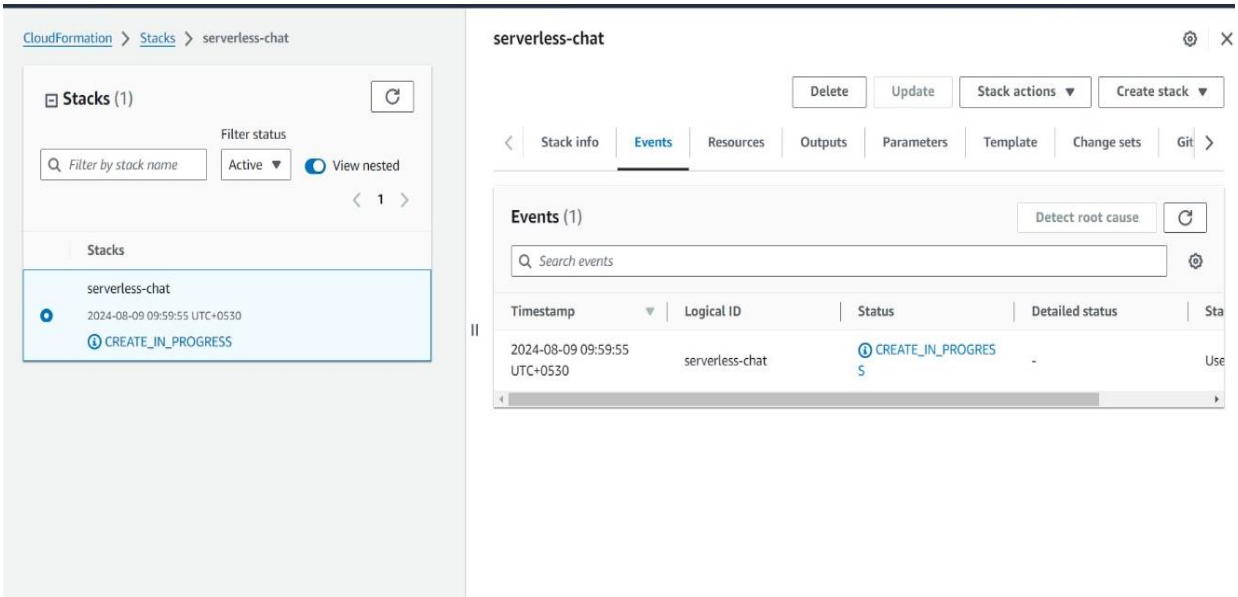
- For Configure stack options no changes, choose Next.



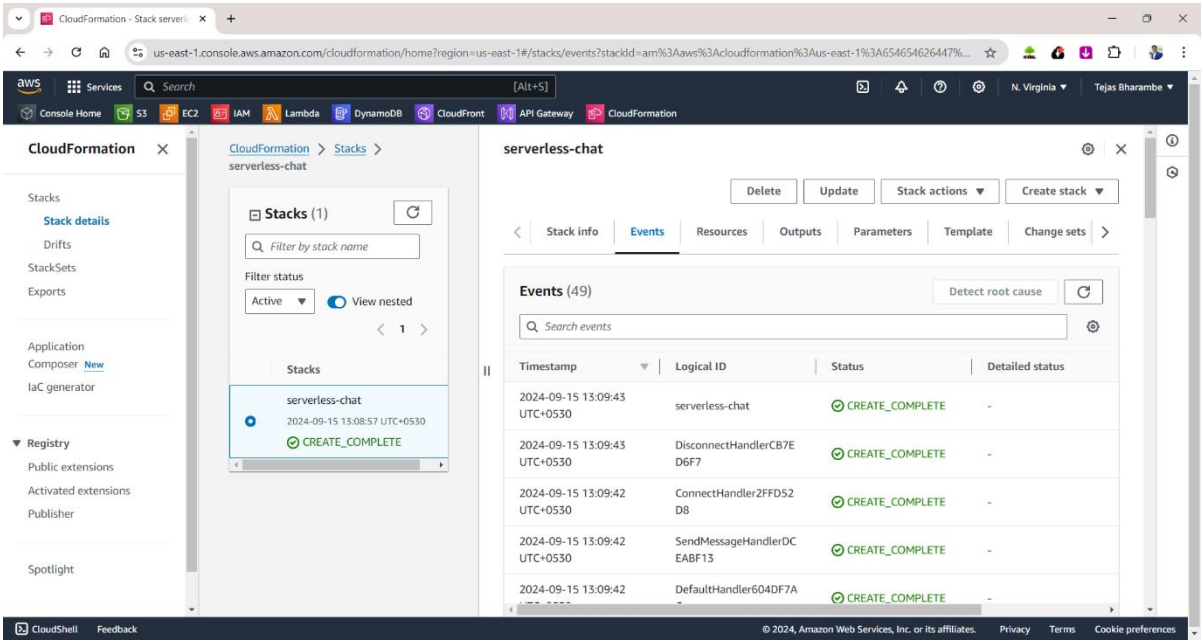
- For Capabilities, acknowledge that AWS CloudFormation can create IAM resources in your account.
- Choose submit



- AWS CloudFormation provisions the resources specified in the template. It can take a few minutes to finish provisioning your resources.



- When the status of your AWS CloudFormation stack is CREATE_COMPLETE, you're ready to move on to the next step.



Step 2: Create a WebSocket API

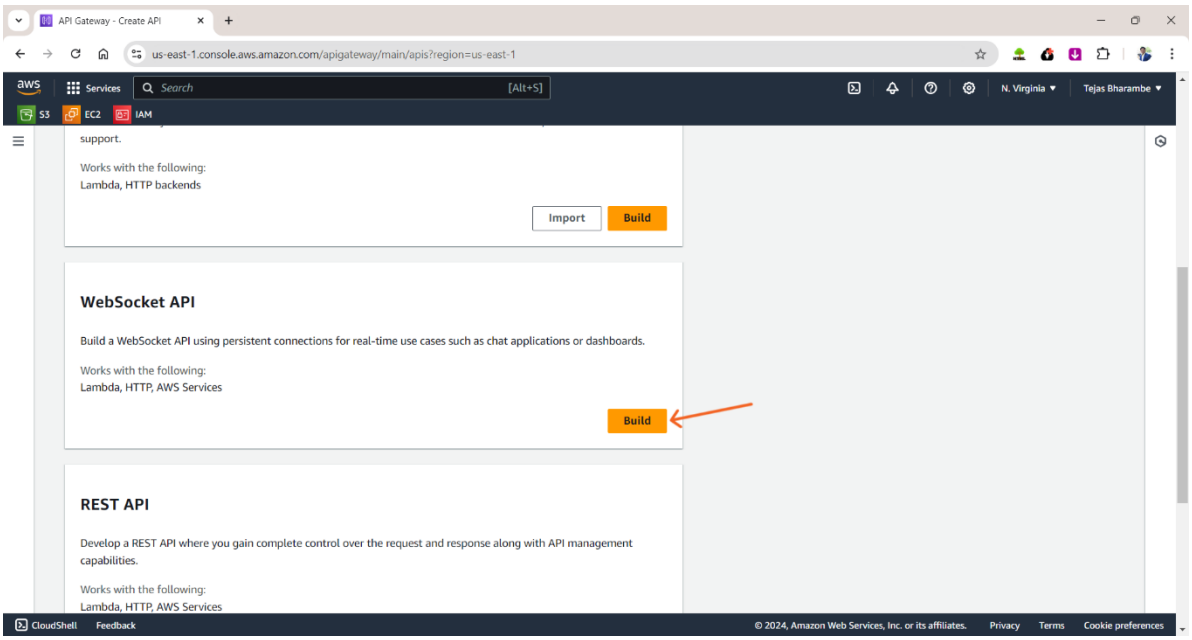
- You'll create a WebSocket API to handle client connections and route requests to the Lambda functions that you created in Step 1.

1. Sign In to the API Gateway Console

- Access API Gateway:
  - Go to the API Gateway console: API Gateway Console.

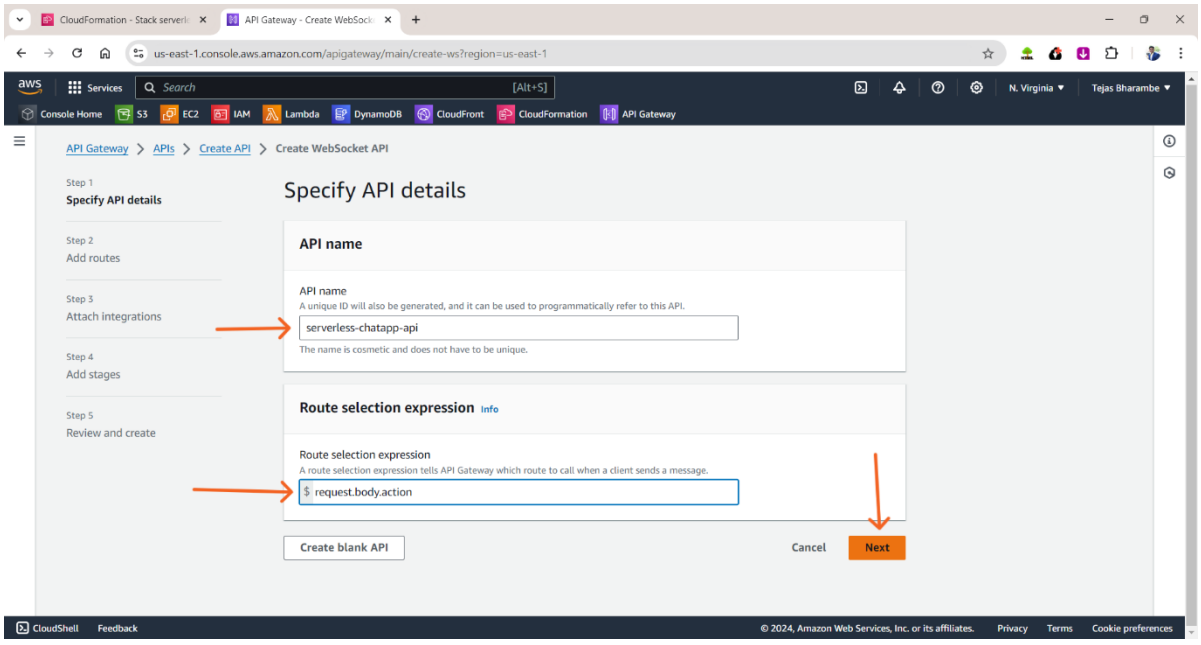
Create a New WebSocket API

1. Choose "Create API":
   - On the API Gateway homepage, select **"Create API"**.
2. Select WebSocket API:
   - Under **"Choose an API type"**, select **"WebSocket API"**.
   - Click **"Build"** to start building your WebSocket API.



3. Enter API Details:

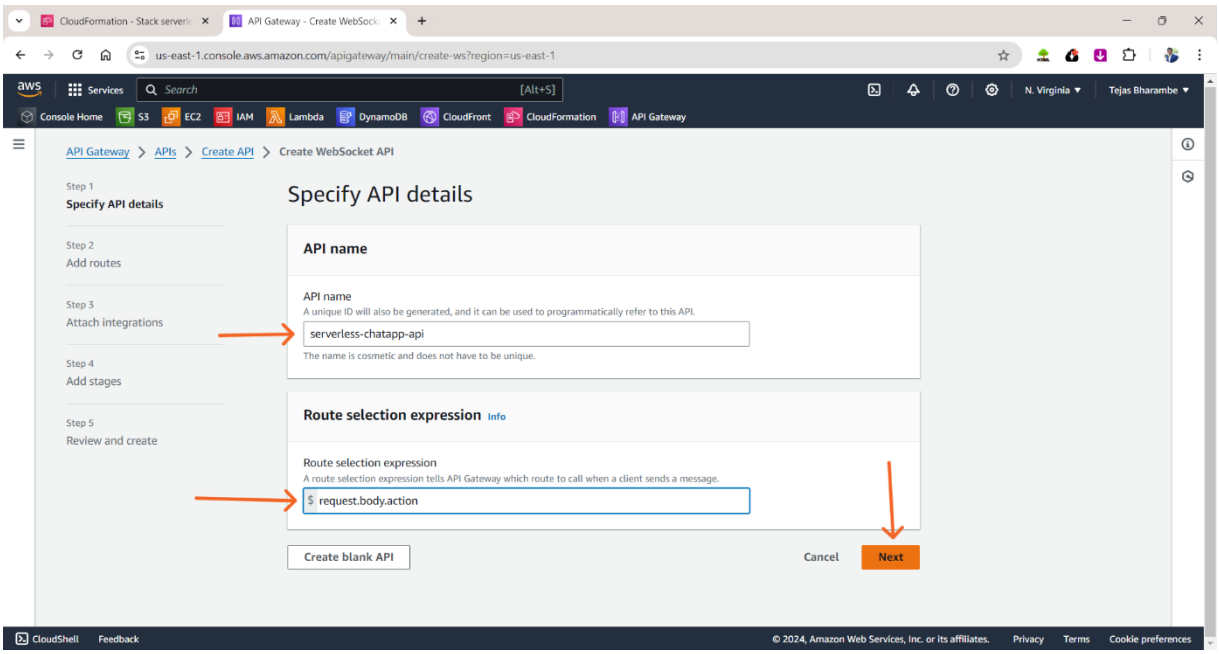
- **API Name**: Enter serverless-chatapp-api as the name of your API.
- **Route Selection Expression**: Enter request.body.action.



This expression determines how API Gateway routes incoming messages to the appropriate Lambda functions. It looks for an action field in the message body to decide which route to invoke.

4. Choose "Next":

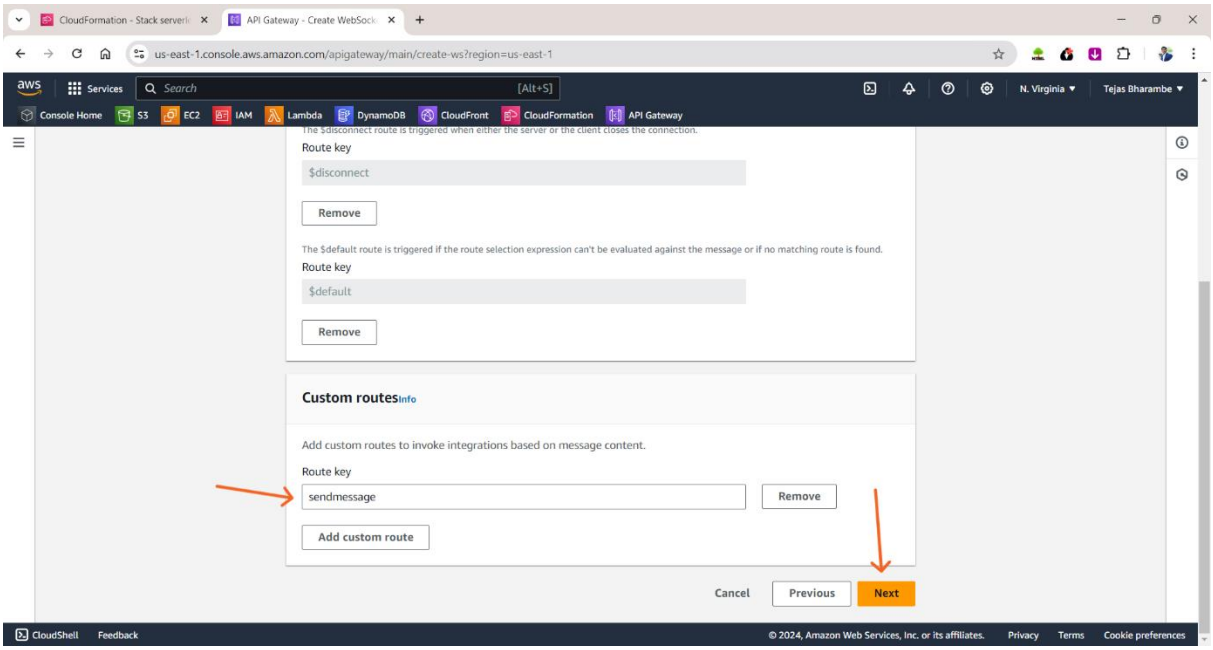
- After entering the API details, click "**Next**" to proceed**.**



3. Define Routes

1. Add Predefined Routes**:**
   - **$connect**: This route is automatically triggered when a client connects to the WebSocket API.
   - **$disconnect**: This route is triggered when a client disconnects from the WebSocket API.
   - **$default**: This route is triggered when no other route matches an incoming request.

For Predefined routes, choose Add $connect, Add $disconnect, and Add $default.
The $connect and $disconnect routes are special routes that API Gateway invokes automatically when a client connects to or disconnects from an API. API Gateway invokes the $default route when no other routes match a request.



2. Add a Custom Route:
    - Click **"Add custom route"**.

3. Route Key: Enter sendmessage.
   This route will handle messages sent by clients during a chat session.

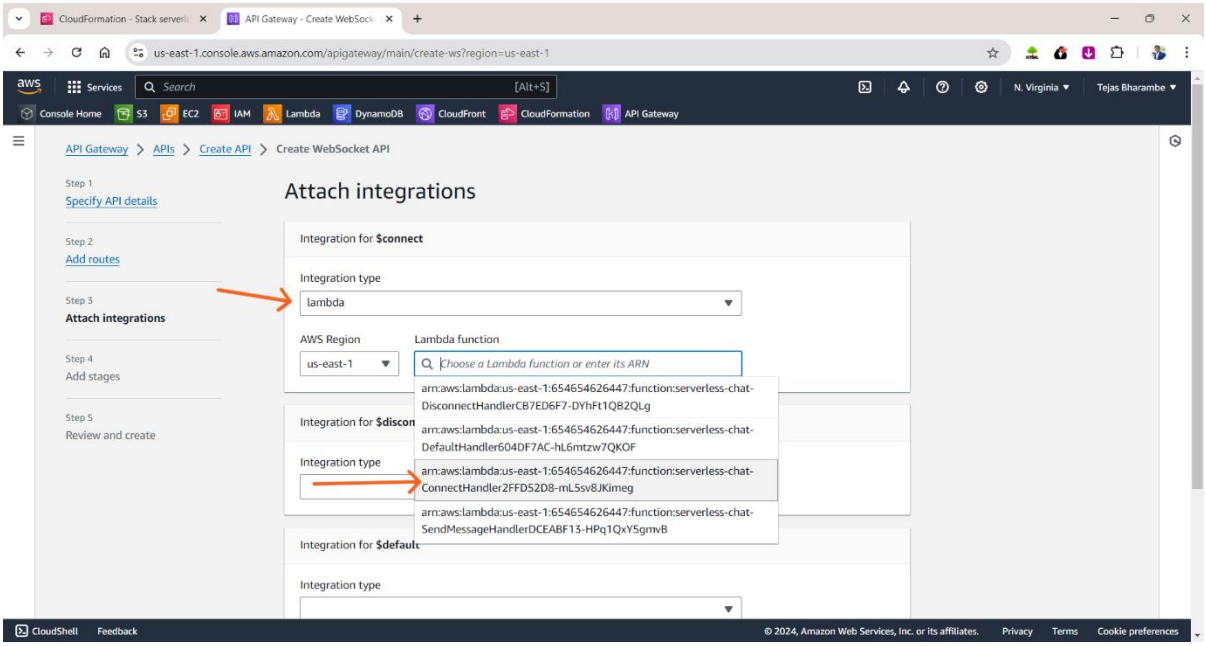
4. Review and Continue:
    - After adding the predefined and custom routes, review your settings.
    - Click **"Next"** to continue.

4. Integrate Routes with Lambda Functions

Once you have created the WebSocket API and defined your routes, the next steps involve integrating these routes with the appropriate Lambda functions that were created earlier. This will allow your API to handle client connections, disconnections, and message transmissions.

Attach Integrations

1. For **Each Route**, Choose Integration Type as Lambda:
2. In the API Gateway console, you'll see the routes you defined earlier ($connect, $disconnect, sendmessage, and $default).



5.Attach Lambda Functions to Routes:

- **$connect Route:**

- Set the Integration type to Lambda.
- For Lambda function, choose the function named *serverless-chat-ConnectHandler.*

- **$disconnect Route**:

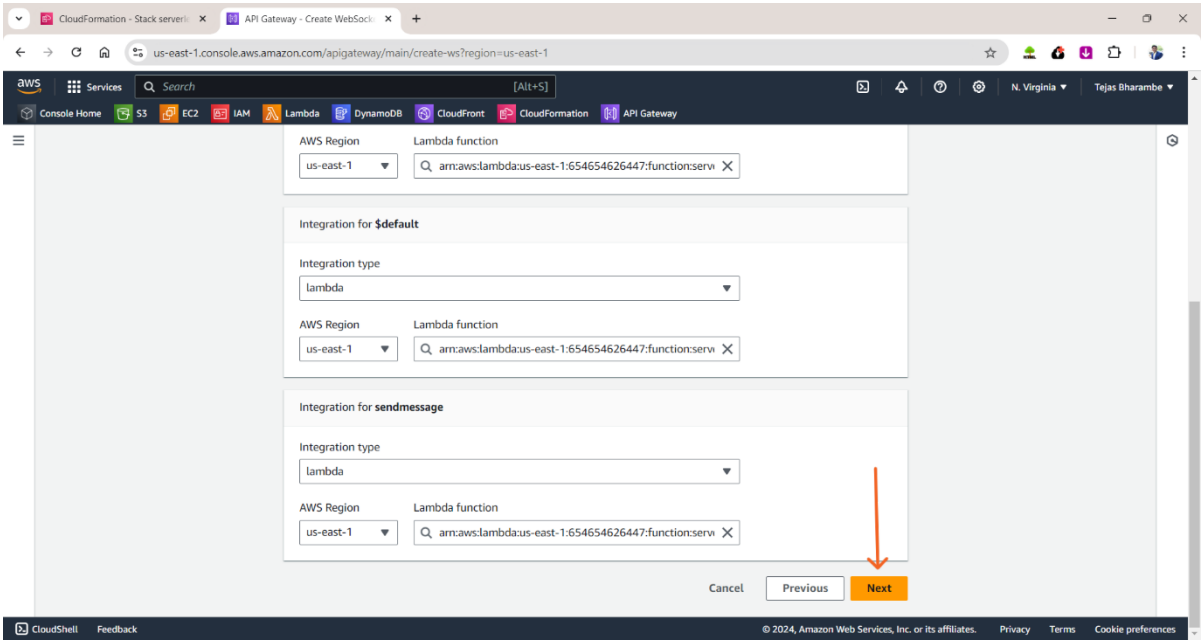
- Set the Integration type to Lambda.
- For Lambda function, choose the function named *serverless-chat-DisconnectHandler*.

- **sendmessage Route**:

- Set the Integration type to Lambda.
- For Lambda function, choose the function named *serverless-chat-SendMessageHandler.*

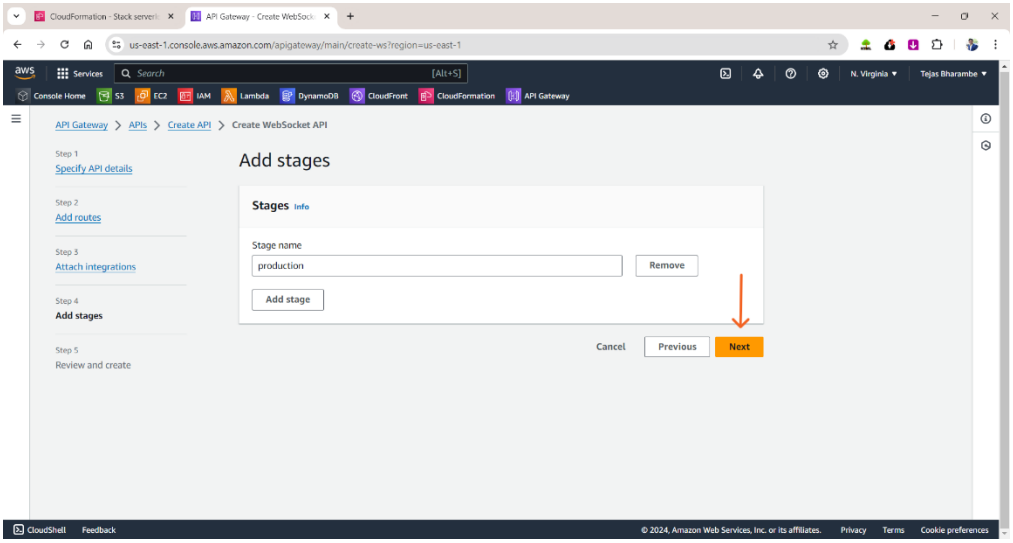
- **$default Route**:

- You can also attach the default route to a Lambda function or leave it as is for handling unmatched requests.



1. **Review the Attached Integrations:**
   - Make sure that each route has the correct Lambda function attached. This setup ensures that when a client connects, disconnects, or sends a message, the appropriate Lambda function is triggered.

2. **Review the Automatically Created Stage:**
   - API Gateway automatically creates a deployment stage for you. By default, this stage is named production.
   - This stage represents the environment where your API is deployed and can be accessed by users.

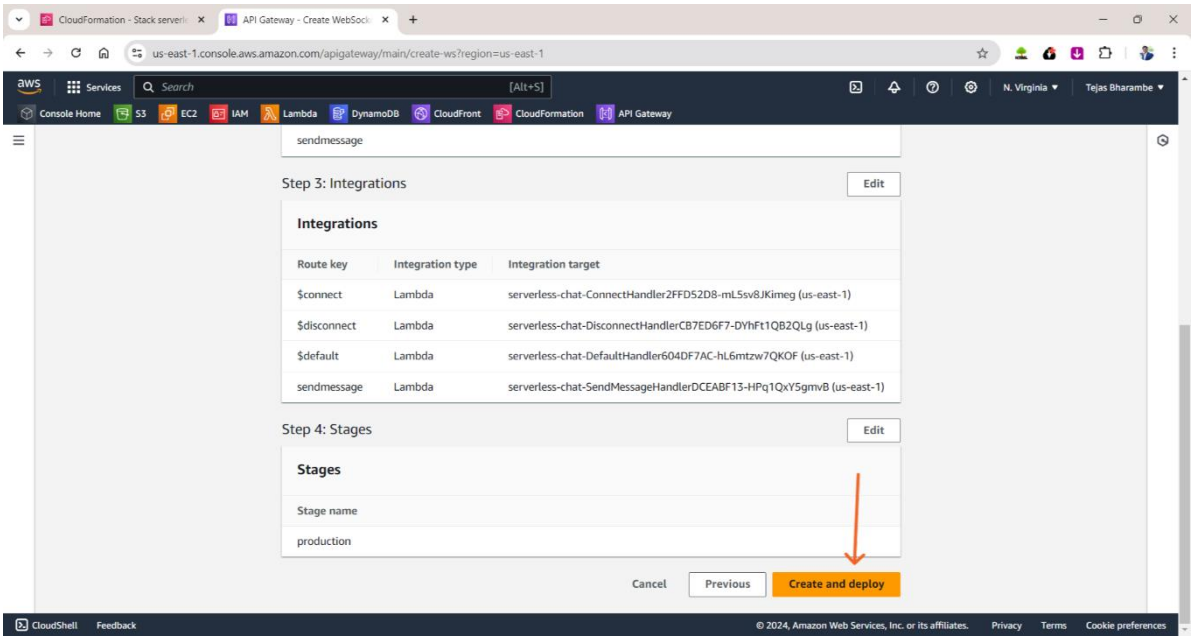


3. **Stage Name:**

    - The stage name production is typical for a live, production-ready environment. You can change this if you prefer (e.g., to dev or test), but for now, using production is fine.

4. **Create and Deploy the WebSocket API**

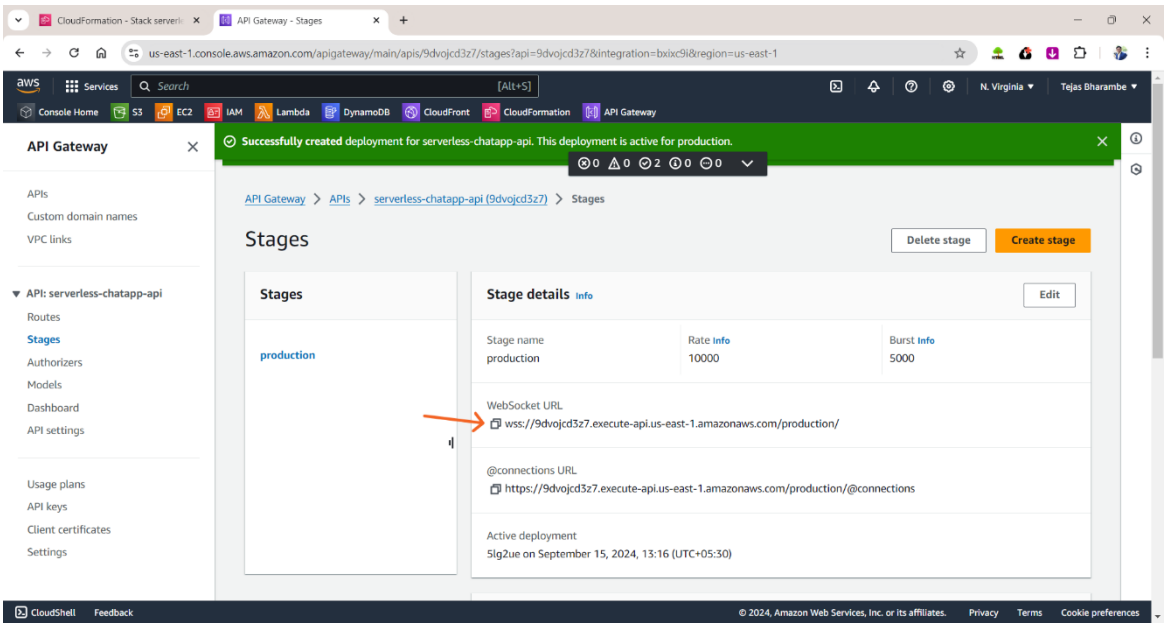
    - Choose "Next"
    - After reviewing the stage, click "Next" to proceed.
    - Finally, click "Create and deploy" to deploy your WebSocket API.
    - API Gateway will deploy the API, making it available at a specific WebSocket URL.



Step 3: Test your API

Next, you'll test your API to make sure that it works correctly. Use the wscat command to connect to the API.

- To to get the invoke URL for your API
- Sign in to the API Gateway console at *https://console.aws.amazon.com/apigateway.*
- Choose your API.
- Choose Stages, and then choose production.



Note your API's **WebSocket URL**. The URL should look like *wss://9dvojcd3z7.execute-api.us-east-1.amazonaws.com/production/*

To connect to your API

1. Use the following command to connect to your API.for that you can use command prompt.
2. When you connect to your API, API Gateway invokes the `$connect` route. When this route is invoked, it calls a Lambda function that stores your connection ID in DynamoDB.

```
wscat -c wss://9dvojcd3z7.execute-api.us-east-1.amazonaws.com/production/
```

3. Open a new command prompt terminal and run the wscat command again with the following parameters.

```
wscat -c wss://9dvojcd3z7.execute-api.us-east-1.amazonaws.com/production/
```

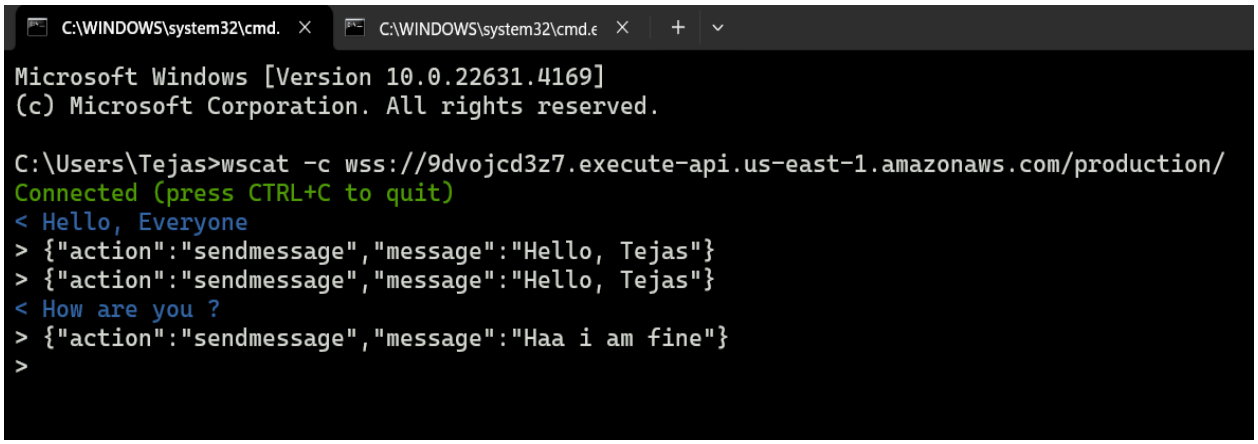
To send a message

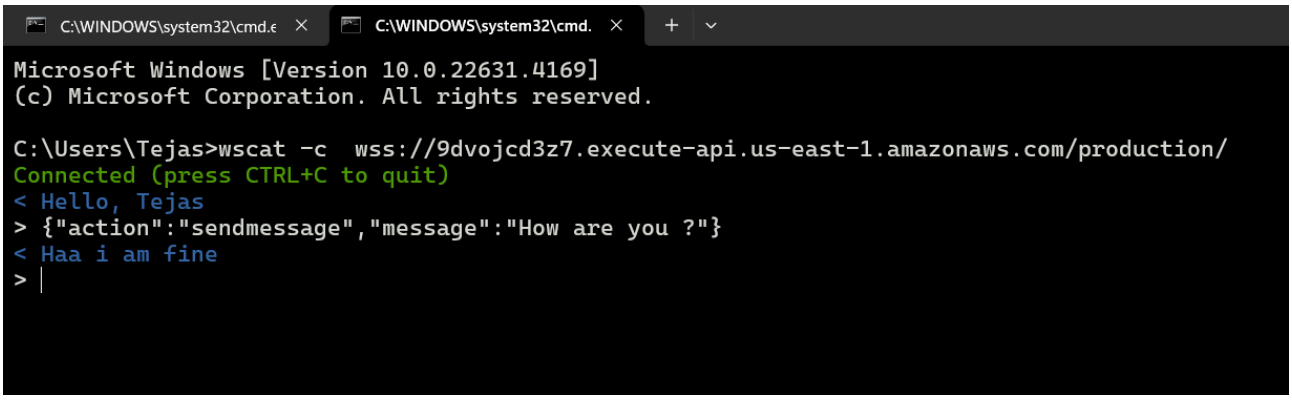
- API Gateway determines which route to invoke based on your API's route selection expression. Your API's route selection expression is `$request.body.action`. As a result, API Gateway invokes the `sendmessage` route when you send the following message:

```
{"action": "sendmessage", "message": "hello, everyone!"}
```

- The Lambda function associated with the invoked route collects the client IDs from DynamoDB. Then, the function calls the API Gateway Management API and sends the message to those clients. All connected clients receive the following message:

- `< hello, everyone!`

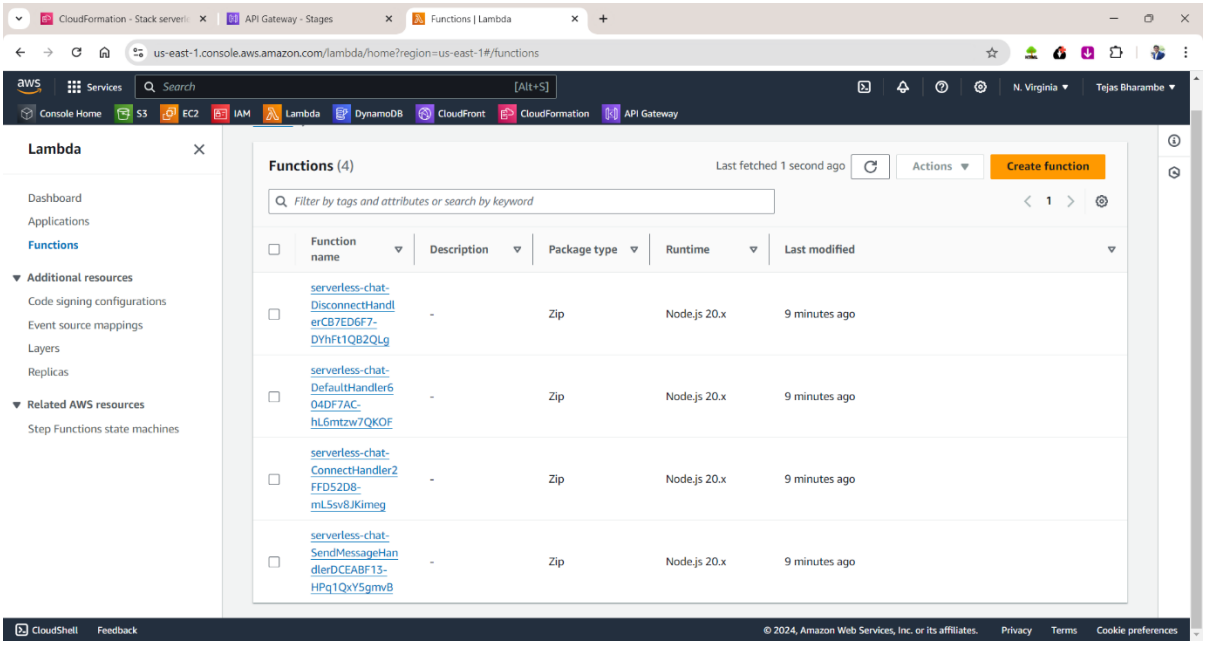


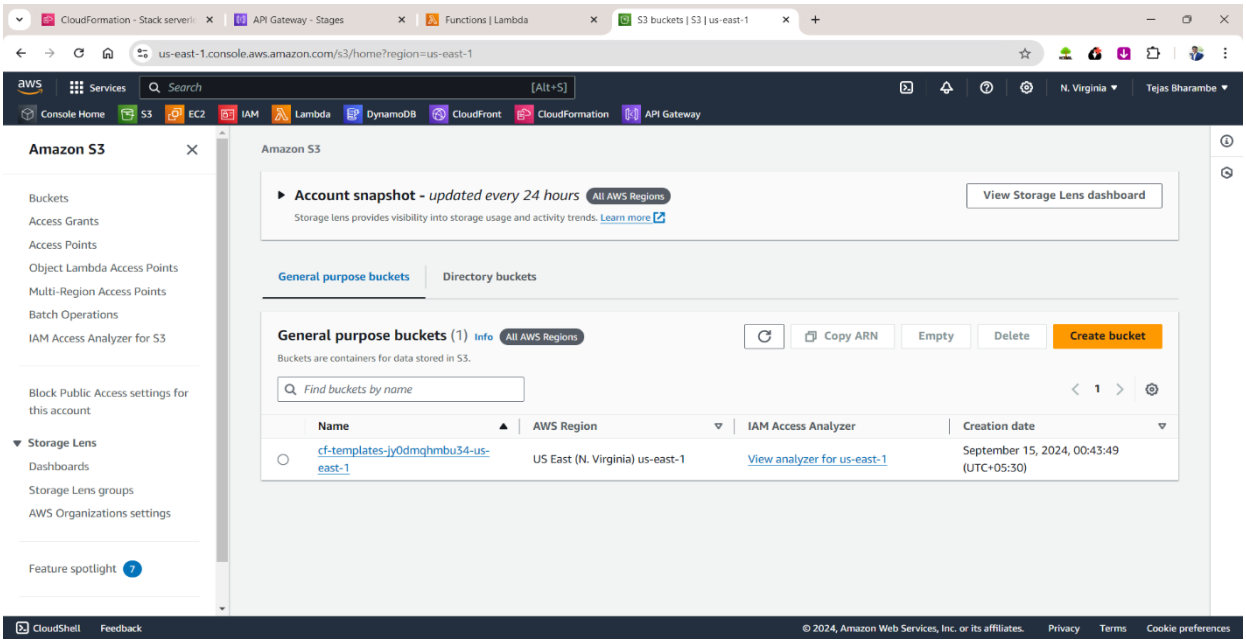


To disconnect from your API

- Press **CTRL+C** to disconnect from your API. When a client disconnects from your API, API Gateway invokes your API's `$disconnect` route. The Lambda integration for your API's `$disconnect` route removes the connection ID from DynamoDB.

Kindly check Lambda function and S3 bucket below.





Step 4: Clean up

To prevent unnecessary costs, delete the resources that you created as part of this tutorial. The following steps delete your AWS CloudFormation stack and WebSocket API.

To delete a WebSocket API

1. Sign in to the API Gateway console at https://console.aws.amazon.com/apigateway.

2. On the **APIs** page, select your serverless-chatapp-api API. Choose **Actions**, choose **Delete**, and then confirm your choice.

To delete an AWS CloudFormation stack

1. Open the AWS CloudFormation console at https://console.aws.amazon.com/cloudformation.
2. Select your AWS CloudFormation stack.
3. Choose **Delete** and then confirm your choice.

Credit Attribution :

I'm grateful to AWS for providing an excellent blog that served as the foundation for this project. This ->Tutorial: Create a WebSocket chat app with a WebSocket API, Lambda and DynamoDB - Amazon API Gateway was instrumental in guiding the implementation of the base architecture.