



AWS Serverless Real-Time Chat Application



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.

Prerequisites:

- An AWS account
- Basic understanding of AWS Lambda, AWS CloudFormation, and AWS API Gateway

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).
- AWS CloudFormation: For Creating a stack by uploading a YAML file.

First, you'll use an AWS CloudFormation template to create Lambda functions that will handle API requests. Then, you'll use the API Gateway console to create a

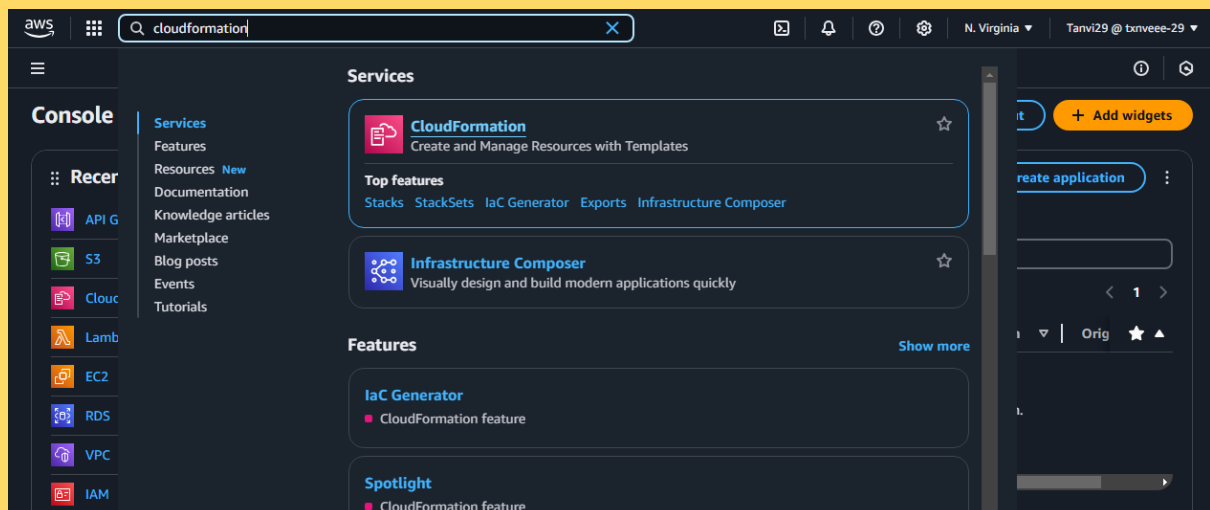
WebSocket API that integrates with your Lambda functions. Lastly, you'll test your API to verify that messages are sent and received.

Download and Unzip the CloudFormation Template:

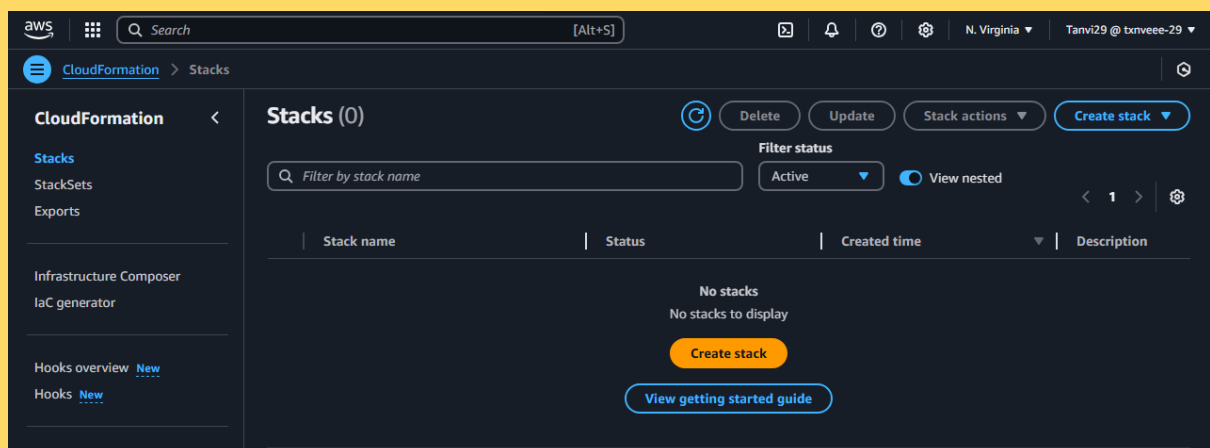
<https://docs.aws.amazon.com/apigateway/latest/developerguide/samples/ws-chat-app-starter.zip>

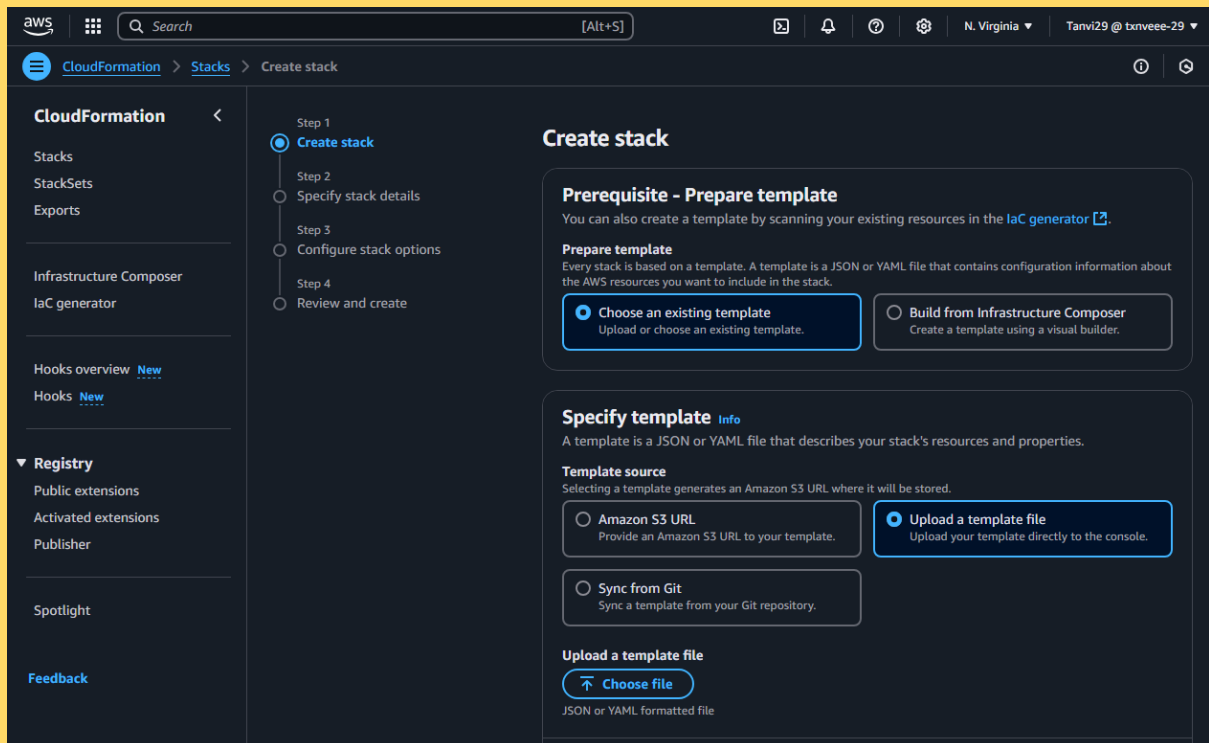
Deploy the CloudFormation Stack

1. Open AWS Management Console:
 - o Go to the [AWS Management Console](#).
2. Navigate to CloudFormation:
 - o In the AWS Management Console, search for and select **"CloudFormation"**.



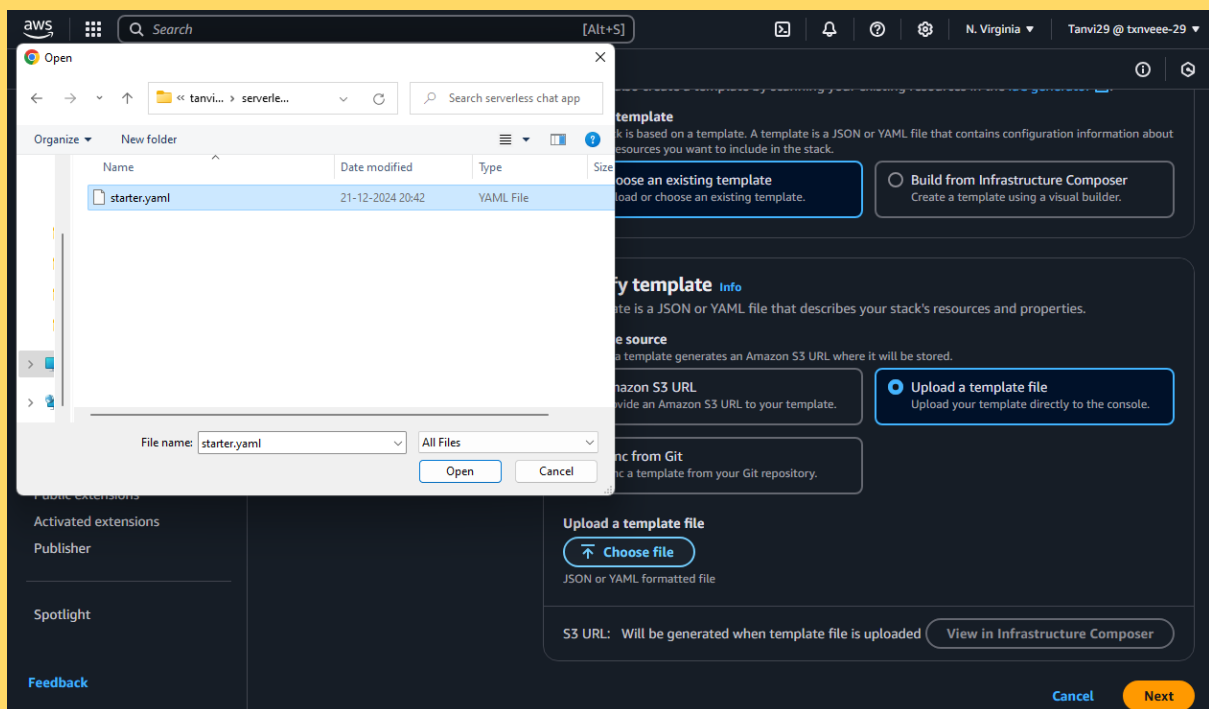
3. Create a New Stack:
 - o Click on **"Create stack"** and then choose **"With new resources (standard)"**





4. Upload the Template File:

- o In the "Specify template" section, choose **"Upload a template file"**.
- o Select the **.yaml** or **.json** file you unzipped earlier.
- o Choose Next.



Upload a template file

Choose file

starter.yaml

JSON or YAML formatted file

5. Configure the Stack:

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

[Alt+S] N. Virginia ▼ Tanvi29 @ txnveee-29 ▼

Create stack

Step 1
● Create stack

Step 2
● **Specify stack details**

Step 3
○ Configure stack options

Step 4
○ Review and create

Specify stack details

Provide a stack name

Stack name

serverless-chat

Stack name must be 1 to 128 characters, start with a letter, and only contain alphanumeric characters. Character count: 15/128.

Parameters

Parameters are defined in your template and allow you to input custom values when you create or update a stack.

No parameters

There are no parameters defined in your template

[Cancel](#) [Previous](#) [Next](#)

- For Configure stack options no changes, choose Next.
- For Capabilities, acknowledge that AWS CloudFormation can create IAM resources in your account.
- Choose submit

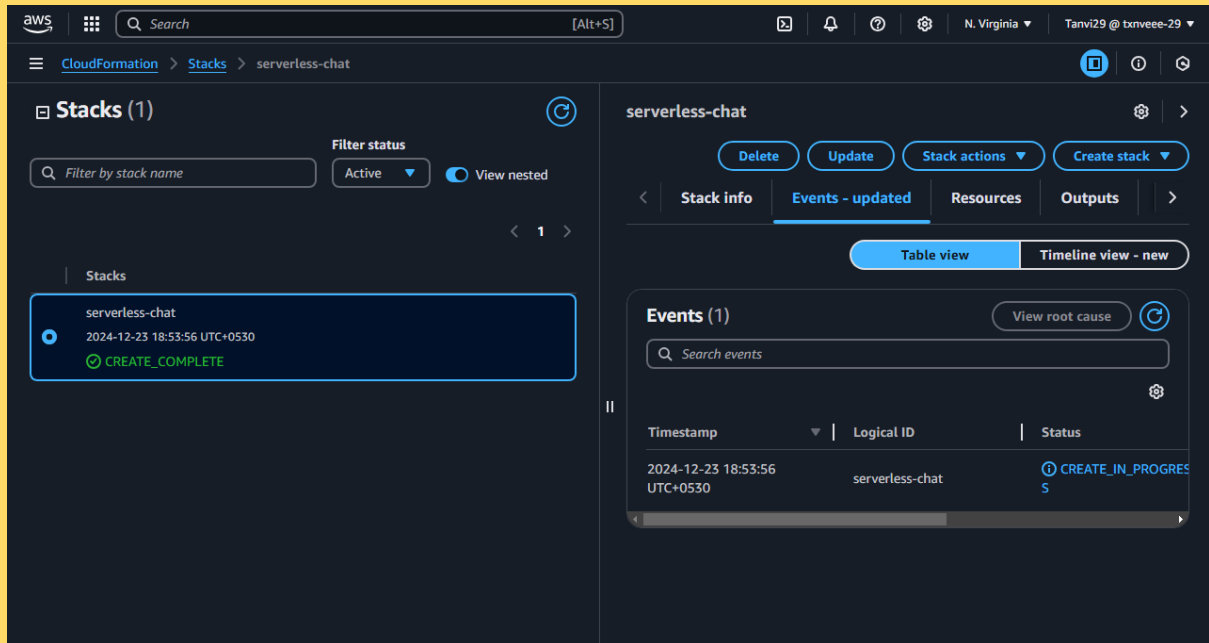
The following resource(s) require capabilities: [AWS::IAM::Role]

This template contains Identity and Access Management (IAM) resources that might provide entities access to make changes to your AWS account. Check that you want to create each of these resources and that they have the minimum required permissions. [Learn more](#)

☒ **I acknowledge that AWS CloudFormation might create IAM resources.**

[Cancel](#) [Previous](#) [Next](#)

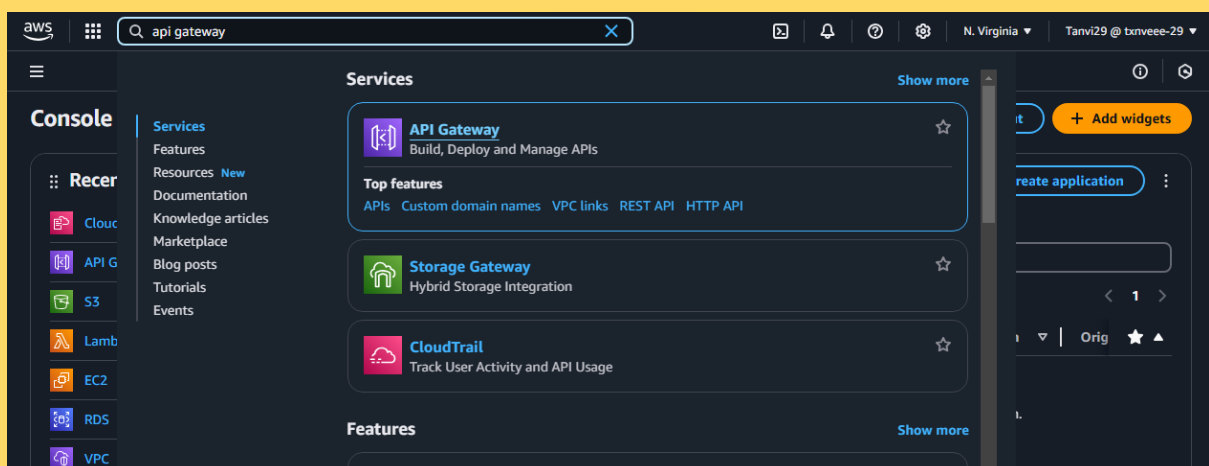
- o AWS CloudFormation provisions the resources specified in the template. It can take a few minutes to finish provisioning your resources.
- o When the status of your AWS CloudFormation stack is `CREATE_COMPLETE`, you're ready to move on to the next step.



Create a WebSocket API:

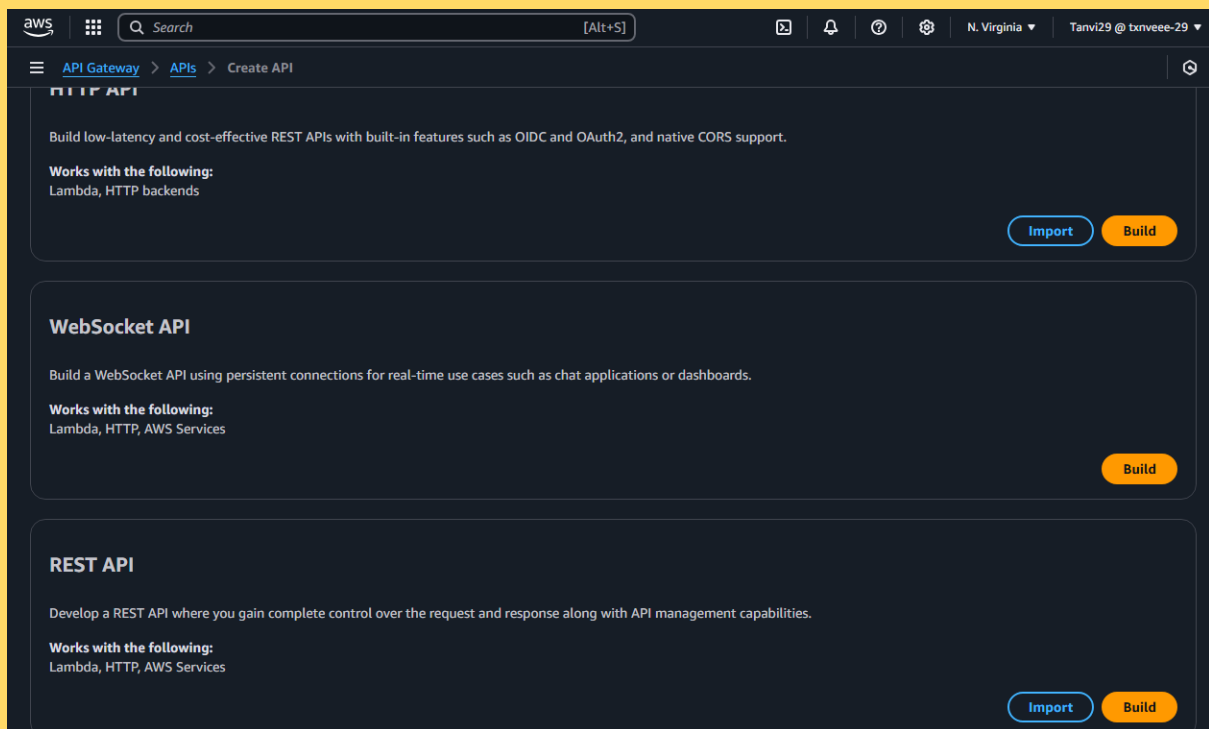
You'll create a WebSocket API to handle client connections and route requests to the Lambda functions that you created in Step 1.

1. Open the API Gateway Console:
 - o Go to the API Gateway console: API Gateway Console.



- o On the API Gateway homepage, select **"Create API"**.
- o Under "Choose an API type", select **"WebSocket API"**.

- o Click **"Build"** to start building your WebSocket API.

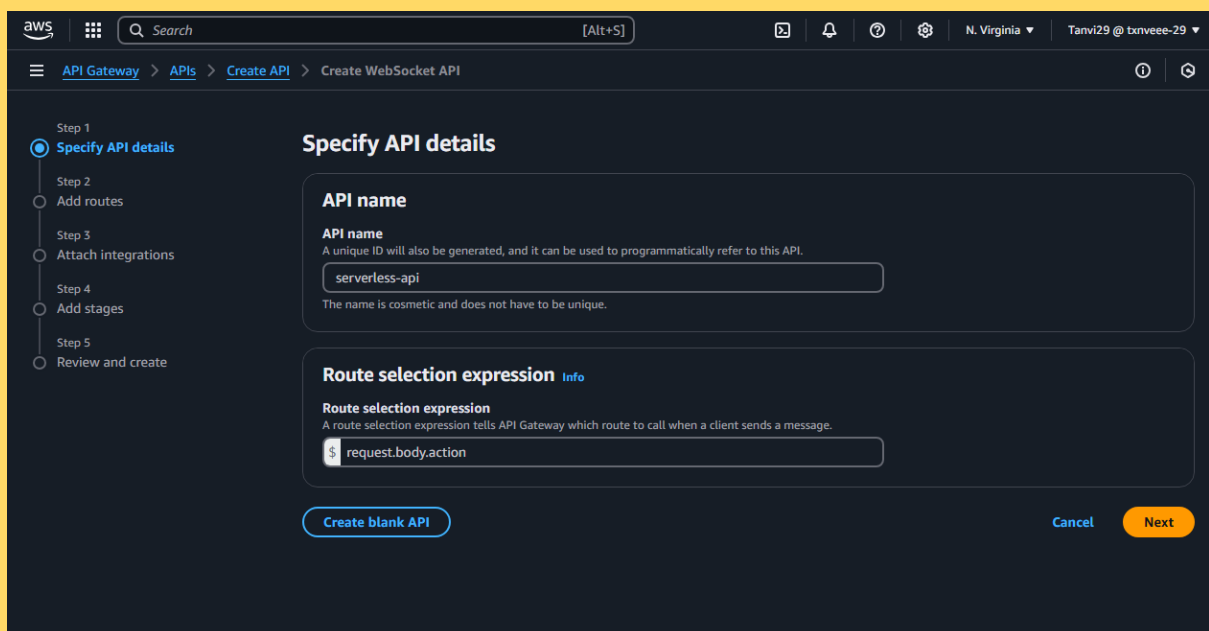


2. Enter API Details:

- o API Name: Enter **"serverless-api"** as the name of your API.
- o 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

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



3. Define Routes

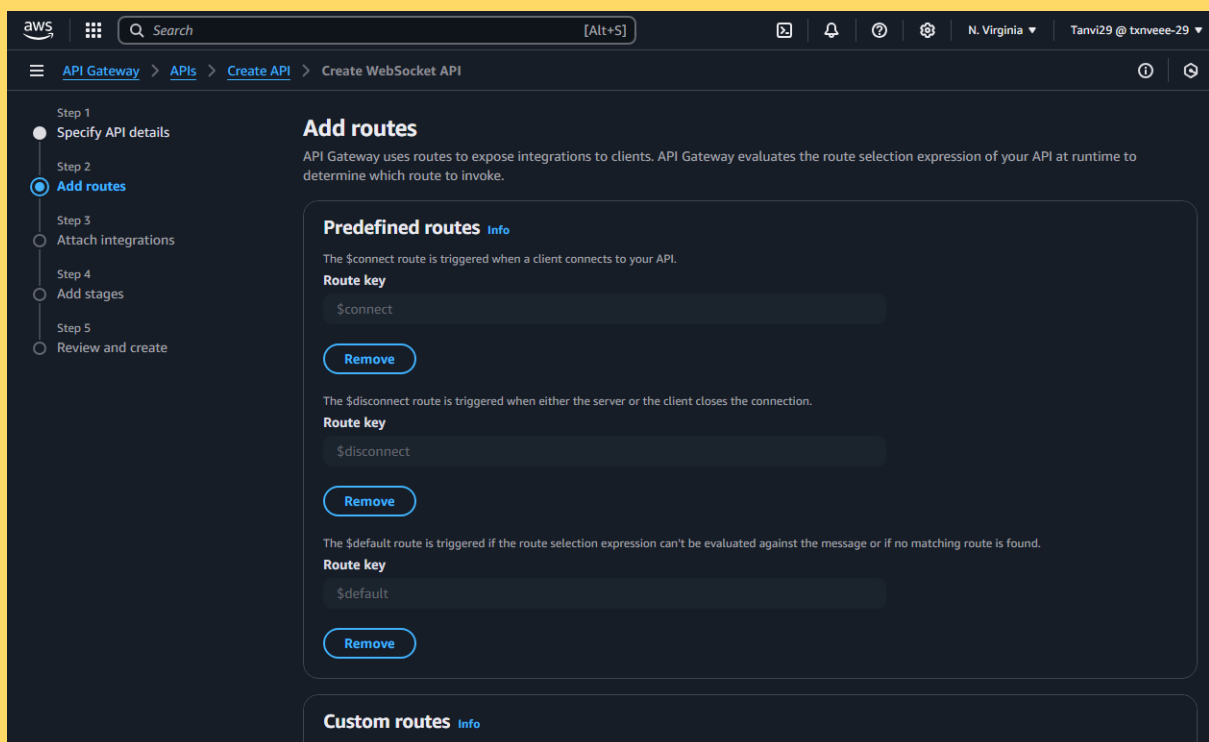
a. Add Predefined Routes:

- o **\$connect**: This route is automatically triggered when a client connects to the WebSocket API.

- o **\$disconnect**: This route is triggered when a client disconnects from the WebSocket API.

- o **\$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.



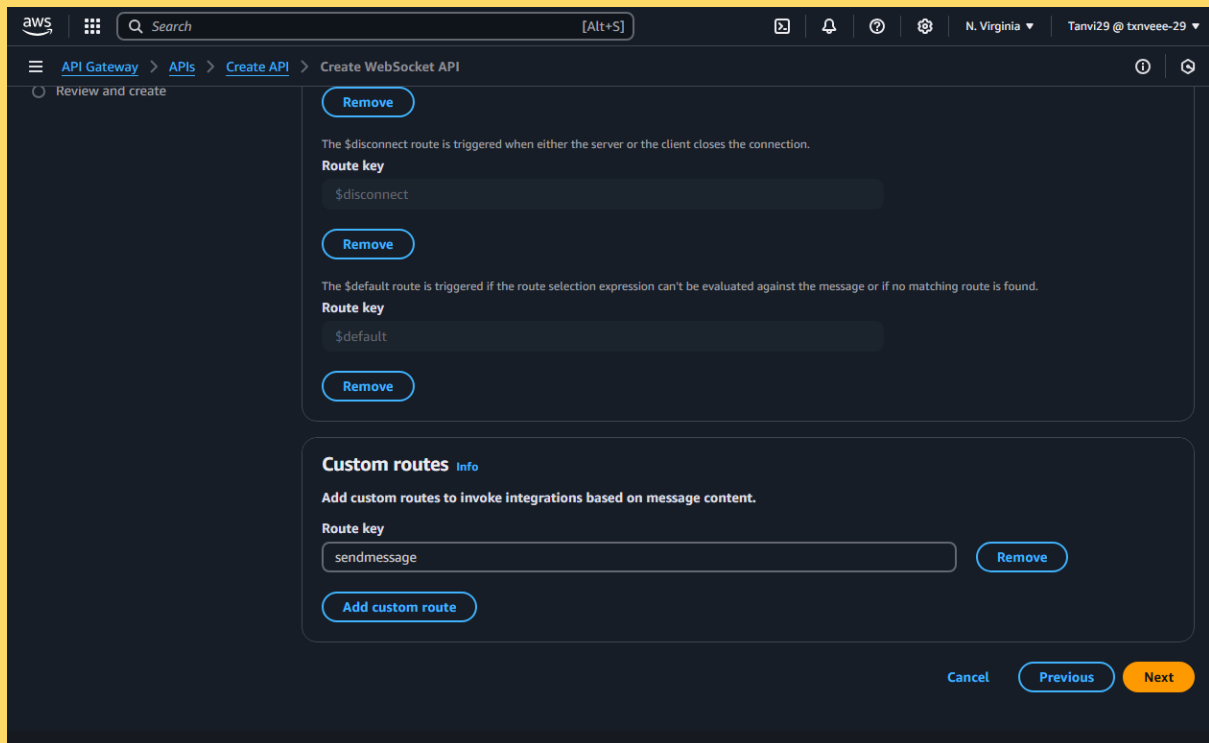
b. Add a **Custom Route**:

- o Click "**Add custom route**".

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

d. Review and Continue:

- o After adding the predefined and custom routes, review your settings.
- o 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

- a. For Each Route, Choose Integration Type as Lambda:
 - b. In the API Gateway console, you'll see the routes you defined earlier (\$connect, \$disconnect, sendmessage, and \$default).
 - c.
5. Attach Lambda Functions to Routes:
- o **\$connect Route:** ▪ Set the Integration type to Lambda. ▪ For Lambda function, choose the function named serverless-chat-ConnectHandler.
 - o **\$disconnect Route:** ▪ Set the Integration type to Lambda. ▪ For Lambda function, choose the function named serverless-chat-DisconnectHandler.
 - o **sendmessage Route:** ▪ Set the Integration type to Lambda. ▪ For Lambda function, choose the function named serverless-chat - SendMessageHandler.
 - o **\$default Route:** ▪ You can also attach the default route to a Lambda function or leave it as is for handling unmatched requests.

aws [Search] [Alt+S] N. Virginia Tanvi29 @ bnveee-29

API Gateway > APIs > Create API > Create WebSocket API

Step 2 Add routes
Step 3 **Attach integrations**
Step 4 Add stages
Step 5 Review and create

Integration for \$connect

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Integration for \$disconnect

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Integration for \$default

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Integration for sendmessage

aws [Search] [Alt+S] N. Virginia Tanvi29 @ bnveee-29

API Gateway > APIs > Create API > Create WebSocket API

Integration for \$disconnect

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Integration for \$default

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Integration for sendmessage

Integration type: Lambda

AWS Region: us-east-1 Lambda function: arn:aws:lambda:us-east-1:975050208869:function:serve

Cancel Previous Next

6. Stage Name:

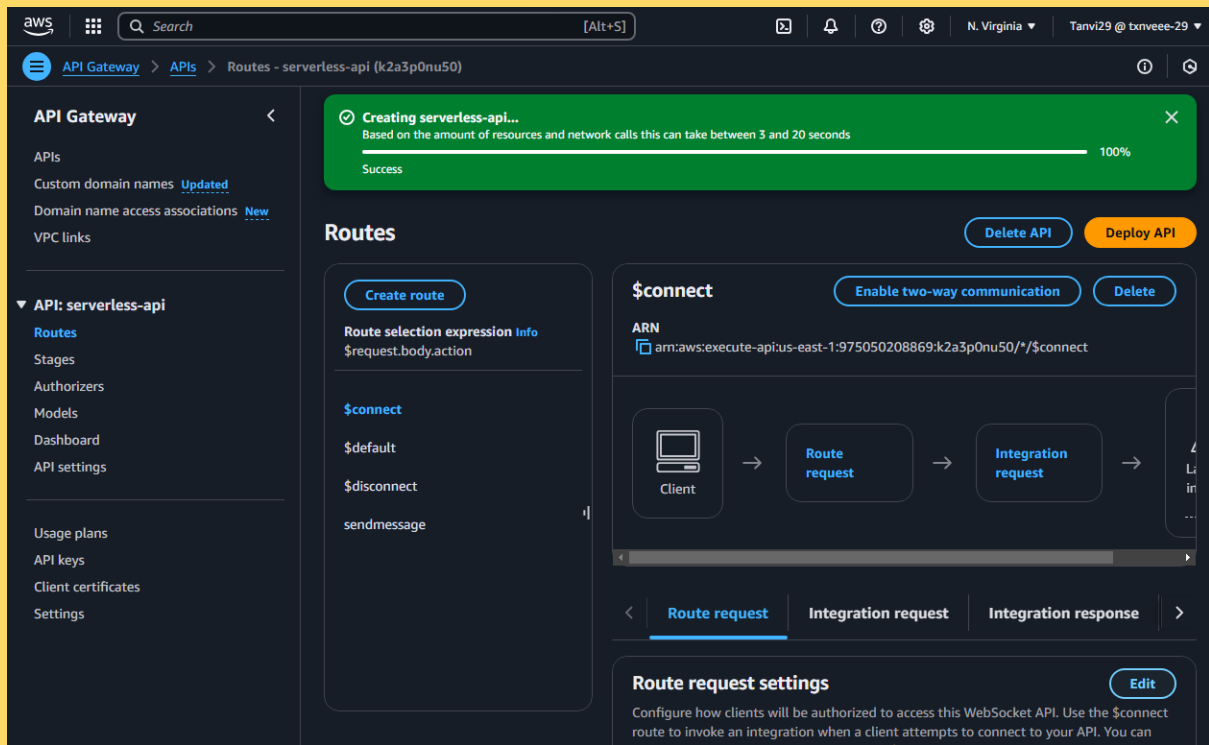
- o The stage name production is typical for a live, production-ready environment. You can change this if you prefer (e.g., to dev or test), but for now, using production is fine.

The screenshot shows the AWS API Gateway console with the 'Create WebSocket API' wizard. The 'Add stages' step is selected in the left-hand navigation pane. The main area displays a form for adding a new stage. The stage name is 'production'. There are 'Add stage', 'Remove', 'Previous', and 'Next' buttons. The 'Next' button is highlighted in orange.

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

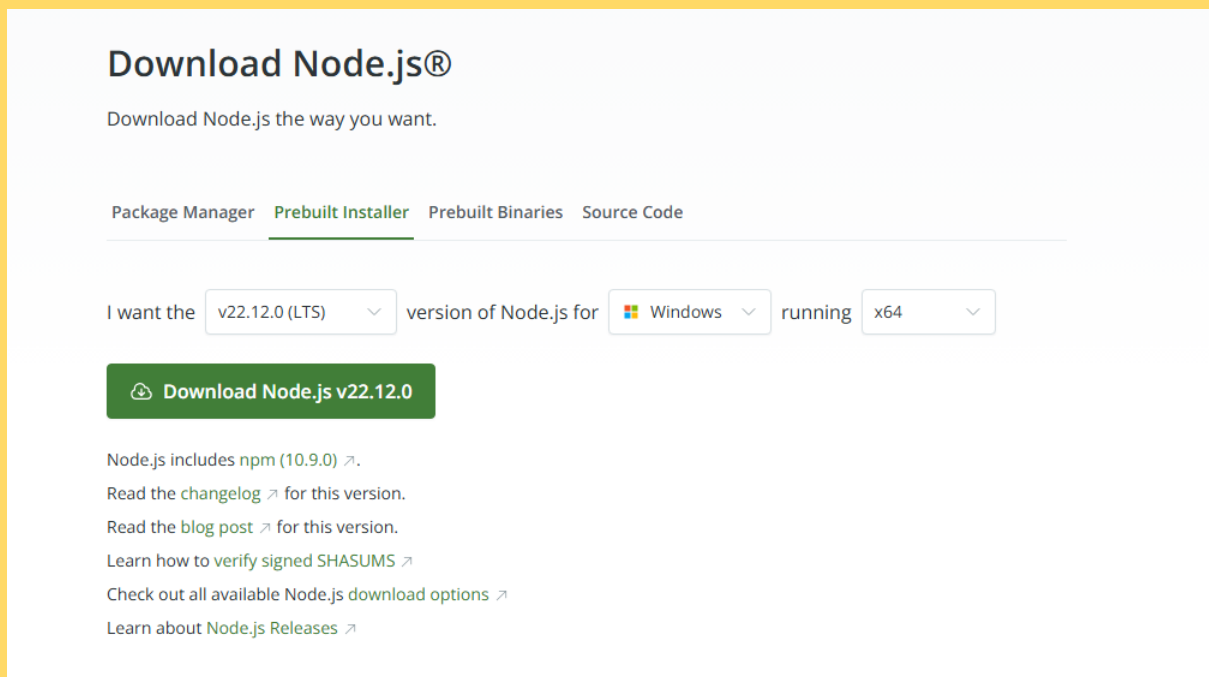
The screenshot shows the AWS API Gateway console with the 'Create WebSocket API' wizard. The 'Integrations' step is selected in the left-hand navigation pane. The main area displays a table of integrations and a section for adding stages. The integrations table lists four routes: \$connect, \$disconnect, \$default, and sendmessage, each with a Lambda integration type and a specific integration target. The 'Stages' section shows a stage named 'production'. There are 'Edit', 'Previous', and 'Create and deploy' buttons. The 'Create and deploy' button is highlighted in orange.

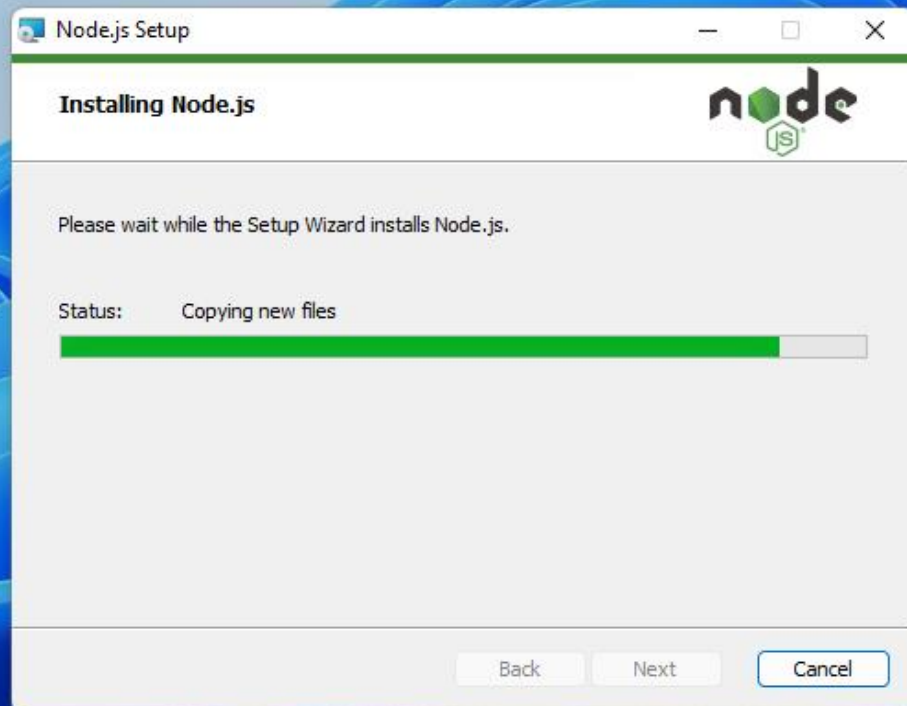
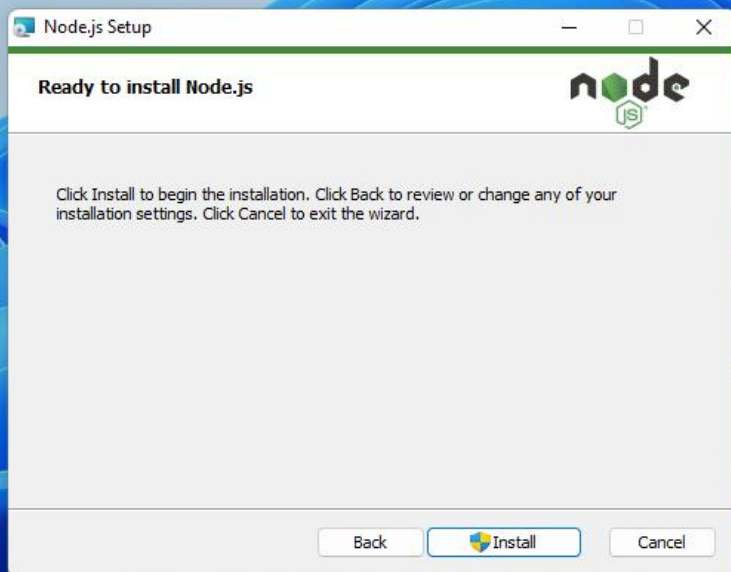
Route key	Integration type	Integration target
\$connect	Lambda	serverless-chat-ConnectHandler2FFD52D8-mTnmWbQrYUmW (us-east-1)
\$disconnect	Lambda	serverless-chat-DisconnectHandlerCB7ED6F7-YMDGv5l0BeBQ (us-east-1)
\$default	Lambda	serverless-chat-DefaultHandler604DF7AC-g1UxKQJz22uN (us-east-1)
sendmessage	Lambda	serverless-chat-SendMessageHandlerDCEABF13-DxQcnaKwEjce (us-east-1)



Install Node.js

1. Search “**download Node.js**” in google and install it

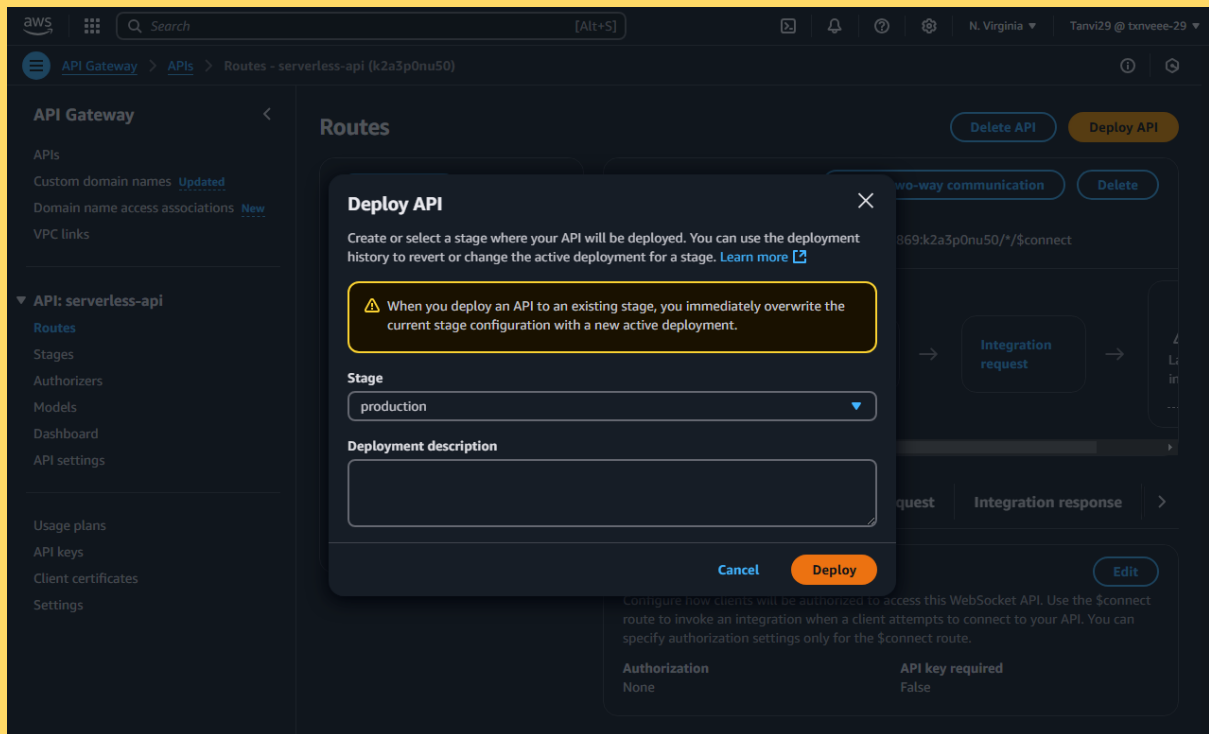




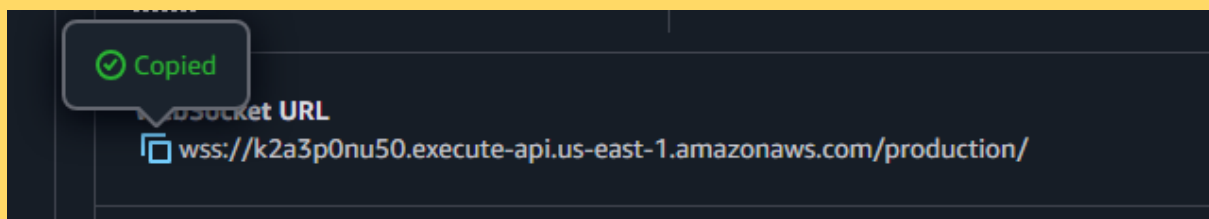


Deploying the API

1. Select "**Deploy API**" as shown below in the image.



2. Copy “WebSocket URL”



3. Now, open two windows cmd side by side:

- a. In first cmd type the commands in the following order:

```
node -v
npm -v
npm install -g wscat
```

- b. After this, type the command `wscat -c` and then paste the copied WebSocket URL as follows:

```
wscat -c wss://k2a3p0nu50.execute-api.us-east-1.amazonaws.com/production/
```

- c. Press “Enter”

- d. You can see that it is now connected and ready to chat and you also have option to disconnect by clicking **Ctrl+Q**

4. Now, in the second windows cmd, type this command again:

```
wscat -c wss://k2a3p0nu50.execute-api.us-east-1.amazonaws.com/production/
```

- a. We now created a connection between the two cmd
- b. Now to check whether we can chat use this command to send message

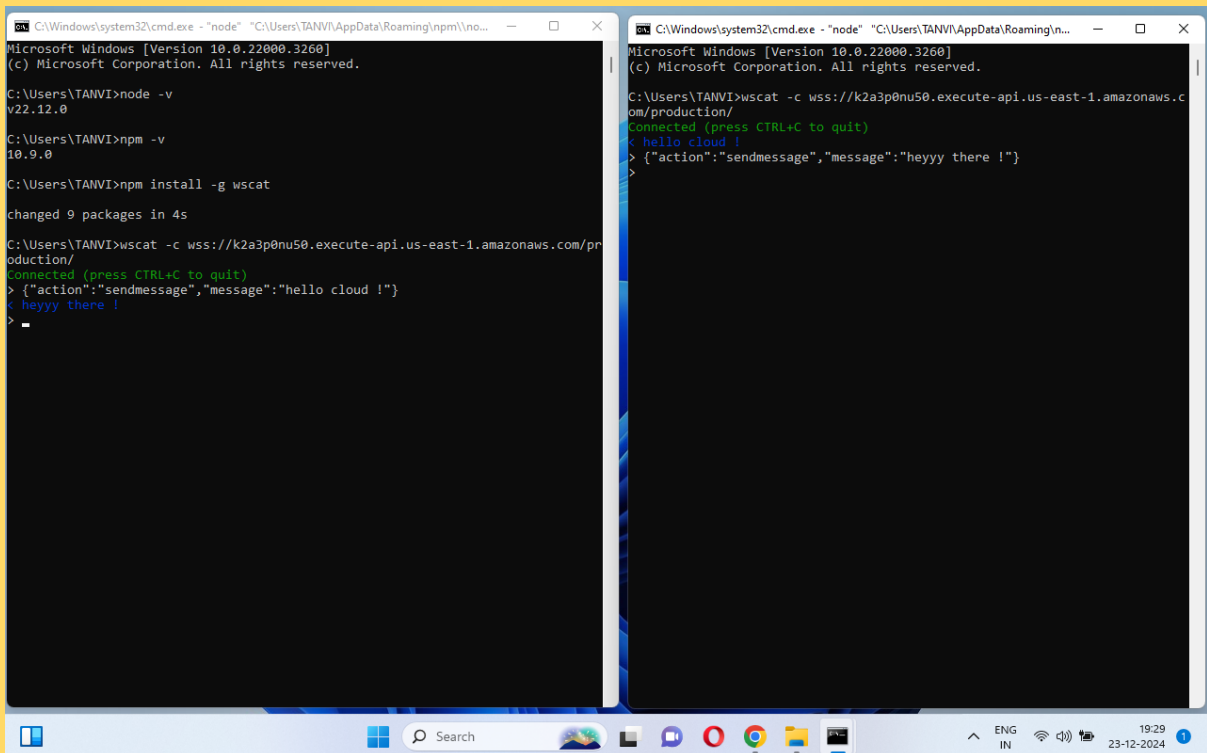
To send a message

• 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 cloud !"}
```

• And then try sending a message from the second cmd using the same command as follows:

```
{"action": "sendmessage", "message": "heyyy there !"}
```



```
C:\Windows\system32\cmd.exe - "node" "C:\Users\TANVI\AppData\Roaming\npm\node_modules\npm\bin\node-cli.js"
Microsoft Windows [Version 10.0.22000.3260]
(c) Microsoft Corporation. All rights reserved.

C:\Users\TANVI>node -v
v22.12.0

C:\Users\TANVI>npm -v
10.9.0

C:\Users\TANVI>npm install -g wscat
changed 9 packages in 4s

C:\Users\TANVI>wscat -c wss://k2a3p0nu50.execute-api.us-east-1.amazonaws.com/production/
Connected (press CTRL+C to quit)
> {"action":"sendmessage","message":"hello cloud !"}
< hello cloud !
> {"action":"sendmessage","message":"heyyy there !"}
< heyyy there !
>
```

5. 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.

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-api**”. Choose Actions, choose **Delete**, and then confirm your choice.
3. To delete an AWS CloudFormation stack
 - a. Open the AWS CloudFormation console at <https://console.aws.amazon.com/cloudformation>.
 - b. Select your AWS CloudFormation “**serverless-chat**” stack, Choose **Delete** and then confirm your choice.