# **COMMUNICATION BRIDGE FOR WEBSITE DEVELOPMENT TEAM**

**Abstract**

This project, titled **“COMMUNICATION BRIDGE FOR WEBSITE DEVELOPMENT TEAM,”** is developed by the **Connection Team** to automate and streamline collaboration between the **Meetly Team** and the **Website Development Team**.

The **Connection Team** acts as an intelligent bridge — receiving meeting transcripts from the Meetly Team via Gmail, analyzing them with **AWS Bedrock (Claude 3.5 Sonnet)** through an automated **n8n workflow**, and converting them into standardized **Website Instruction JSONs**. These structured instructions are then sent to the Website Team through a webhook for immediate use in website creation.

If the Website Team requires additional assets such as images or voice scripts, their JSON-based requests are automatically formatted into readable emails and sent back to the Meetly Team. Every step, including AI analysis and data exchange, is securely logged into **AWS DynamoDB** via **Lambda** and **API Gateway**.

By implementing this workflow, the **Connection Team** eliminates manual communication gaps, improves coordination, and establishes an efficient, AI-driven automation pipeline for website development projects.

**Project Overview**

The **Connection Team Workflow** automates and streamlines communication between three key teams involved in website development — **Meetly Team**, and **Website Development Team**.

* **Meetly Team** conducts client meetings and sends the project transcript (in JSON format) via Gmail.
* The **Connection Team**, through an automated **n8n workflow**, extracts the JSON attachment and analyzes it using an **AI Agent powered by AWS Bedrock (Claude 3.5 Sonnet)**.
* The AI interprets the client’s use case, identifies the website category (e.g., hospital, education, restaurant), and generates a structured **Website Instruction JSON** containing sections, themes, and required content.
* The **Website Team** receives this instruction automatically through a **webhook endpoint**, enabling them to begin website creation instantly.
* If the Website Team needs additional media assets (images, voice scripts, etc.), their JSON request is automatically converted to a plain-text email and sent back to the **Meetly Team** for fulfillment.
* Every step of the workflow — from receiving transcripts to AI analysis, data exchange, and email communication — is logged into **AWS DynamoDB** using **Lambda** and **API Gateway** for traceability and monitoring.

This system transforms what was once a **manual, error-prone coordination process** into an **intelligent, automated, and fully traceable communication bridge**, ensuring faster project turnaround and improved collaboration among all teams.

**Objective**

The primary objective of the “Communication Bridge for Website Development Team” project is to design and implement an intelligent automation workflow that eliminates manual coordination between the Meetly Team and the Website Development Team.

Developed by the Connection Team, this system acts as a smart intermediary layer that:

* Automatically receives client meeting transcripts shared by the Meetly Team in JSON format via Gmail.
* Uses AI (AWS Bedrock – Claude 3.5 Sonnet) to interpret and transform the transcripts into a standardized website instruction format, enabling the Website Team to begin development without delays.
* Automatically handles return communication, such as image or voice script requests, by converting technical JSON responses into readable emails for the Meetly Team.
* Ensures transparency and reliability by logging every workflow step (AI output, data exchange, and communication flow) into AWS DynamoDB through Lambda and API Gateway integration.

By achieving these objectives, the Connection Team’s automation workflow establishes a seamless, AI-powered communication channel, minimizing errors, improving clarity, and significantly reducing the time taken to transfer client requirements between teams.

**Technologies and Tools Used**

|  |  |  |
| --- | --- | --- |
| **Category** | **Tool / Service** | **Purpose** |
| Workflow Automation | **n8n** | Acts as the main orchestration tool to automate the entire workflow between Gmail, AI, and APIs. |
| Cloud AI | **AWS Bedrock (Claude 3.5 Sonnet)** | Analyzes meeting transcripts and converts them into structured website instruction JSONs. |
| Email Service | **Gmail API (n8n Node)** | Used to automatically receive client transcripts and send formatted emails to the Meetly Team. |
| Database | **AWS DynamoDB** | Stores detailed logs of workflow executions, AI analysis, and team communications for auditing. |
| Serverless Function | **AWS Lambda** | Writes and processes log data into DynamoDB via API Gateway. |
| API Management | **AWS API Gateway** | Provides a secure HTTP endpoint for communication between n8n and AWS Lambda. |
| Webhooks | **n8n Cloud Webhook / Website API** | Enables real-time data exchange between the workflow and the Website Development Team. |
| Scripting Languages | **JavaScript & Python (n8n Code Nodes)** | Used within the workflow for JSON parsing, transformation, and plain-text message generation. |

**Architecture Diagram**

Meetly Team (Gmail)

│

▼

[n8n Workflow Trigger: Gmail]

│

▼

Parse Transcript (Binary→JSON)

│

▼

AI Agent (Claude 3.5 Sonnet - Bedrock)

│

▼

Clean & Parse JSON

│

▼

Send to Website Team (Webhook)

│

▼

Website Team Response (Image/Voice)

│

▼

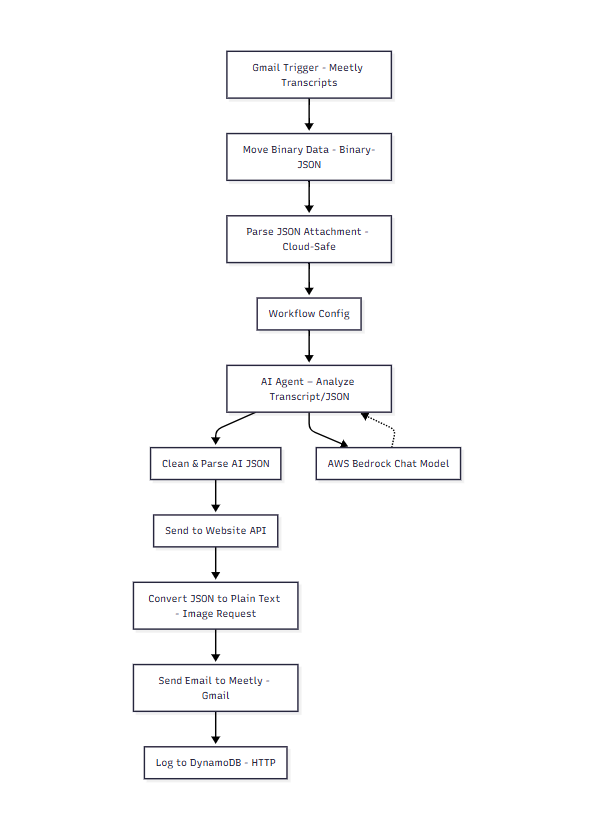
Format → Send Email to Meetly

│

▼

Log Results to AWS DynamoDB

**Mermaid diagram**



**Step-by-Step Implementation Process**

### **Step 1️: Gmail Trigger – Meetly Transcripts**

* Automatically monitors Gmail for new emails from the Meetly Team.
* Downloads attached transcript files (trans.json) in real-time.
* Initiates the workflow whenever a valid transcript email is received.

### **Step 2️: Move Binary Data (Binary → JSON)**

* Converts the attached binary JSON file into text-readable format.
* Makes the data accessible for parsing and AI processing.
* Bridges raw Gmail attachments with structured data handling in n8n.

### **Step 3️: Parse JSON Attachment (Cloud-Safe)**

* Validates the JSON file to ensure it’s properly structured.
* Removes unwanted spaces or invalid characters.
* Prepares the data for accurate AI analysis in later steps.

### **Step 4️: Workflow Config**

* Stores dynamic URLs like websiteApiUrl and logApiUrl.
* Centralizes configuration for easy updates or environment changes.
* Ensures all connected APIs stay flexible and maintainable.

### **Step 5️: AI Agent – Analyze Transcript / JSON**

* Uses **Claude 3.5 Sonnet (AWS Bedrock)** to interpret the meeting transcript.
* Extracts use case details such as website type, theme, and sections.
* Generates structured “Website Instruction JSON” for the Website Team.

### **Step 6️: Clean & Parse AI JSON**

* Cleans the AI output by removing unwanted characters or formatting.
* Extracts valid JSON blocks and normalizes key fields.
* Ensures a consistent structure like website\_type, business\_name, and sections\_required.

### **Step 7️: Send to Website API**

* Sends the processed JSON to the Website Team’s webhook endpoint.
* Shares complete website instructions automatically.
* Waits for and receives their response about required assets or voice scripts.

### **Step 8️: Convert JSON to Plain Text (Image Request)**

* Converts the Website Team’s asset request JSON into readable email text.
* Lists image and voice script requirements clearly for communication.
* Generates formatted subject and body for the Meetly Team email.

### **Step 9️: Send Email to Meetly (Gmail)**

* Sends the formatted message to the Meetly Team via Gmail.
* Notifies them about all required images and voice scripts.
* Completes the feedback loop between the Website and Meetly Teams.

### **Step 10: Log to DynamoDB (HTTP)**

* Provides a central dashboard for Connection Team activity tracking.
* Posts workflow logs to AWS DynamoDB through API Gateway + Lambda.
* Enables audit tracking and process monitoring for the Connection Team.

**Code Snippets for Key Components**

* **AI Agent Prompt (LangChain Node)**

You are an intelligent website analyzer working in the Connection Team.

You receive structured JSON input that describes a website use case (e.g., hospital, restaurant, education platform, e-commerce, NGO, or corporate site).

Your job is to:  
 1. Detect the main domain or topic of the website based on its title, sections, and keywords.  
 2. Summarize the website’s purpose.  
 3. Produce a standardized “website instruction JSON” that can be sent directly to the Website Building Team.

Title rule:  
 If the input JSON already includes a title or business\_name, keep it EXACTLY as written — do not shorten, modify, or reformat.

Domain detection guidance:  
 - If words like “menu”, “chef”, “dining”, “food”, “restaurant”, or “reservation” appear → topic = “Restaurant”.  
 - If words like “hospital”, “doctor”, “patient”, “healthcare”, or “medical” → topic = “Hospital”.  
 - If words like “course”, “education”, “student”, “learning”, “academy” → topic = “Education”.  
 - If words like “product”, “shop”, “cart”, “store”, “e-commerce” → topic = “E-commerce”.  
 - If unclear, set topic = “Business/Organization”.

Output rules:  
 - Return only valid JSON (no explanations, markdown, or code fences).  
 - If some details are missing, infer reasonable defaults.  
 - Keep section names descriptive but concise.  
 - Always include 3–6 relevant images in “images\_required”.

**Output Format:**

{  
 "mode": "website\_instruction",  
 "topic": "<detected website topic>",  
 "summary": "<one-sentence overview>",  
 "website": {  
 "title": "<exact title from input>",  
 "theme": "<color scheme or design style>",  
 "sections": [  
 { "name": "<section name>", "details": "<brief description>" }  
 ],  
 "images\_required": [  
 "<list of image ideas>"  
 ]  
 }  
 }

Input JSON:  
 {{$json}}

* **Clean & Parse AI JSON**

// Step 1: Extract the raw string from AI output

let raw = $json.output || $json.text || $json.data || JSON.stringify($json);

// Step 2: Find and extract only the JSON portion

const firstBrace = raw.indexOf('{');

const lastBrace = raw.lastIndexOf('}');

if (firstBrace === -1 || lastBrace === -1) {

return [{ json: { error: 'No JSON block found in AI output', raw\_output: raw } }];

}

const jsonString = raw.slice(firstBrace, lastBrace + 1);

// Step 3: Parse the extracted JSON safely

let parsed;

try {

parsed = JSON.parse(jsonString);

} catch (err) {

return [{ json: { error: 'Invalid JSON format', details: err.message, raw\_json: jsonString } }];

}

// Step 4: Extract the fields

const website = parsed.website || {};

const topicRaw = parsed.topic || '';

const title = website.title || parsed.title || 'Unnamed Website';

const sections = (website.sections || []).map(s => s.name || '').filter(Boolean);

// Step 5: Normalize the topic

let website\_type = topicRaw.toLowerCase();

if (website\_type.includes('hospital') || website\_type.includes('healthcare')) website\_type = 'hospital';

else if (website\_type.includes('education')) website\_type = 'education';

else if (website\_type.includes('restaurant')) website\_type = 'restaurant';

else website\_type = 'business';

// Step 6: Build final Website API payload

const result = {

website\_type,

business\_name: title,

sections\_required: sections

};

// Step 7: Return

return [{ json: result }];

### **Log to DynamoDB (HTTP)**

Js code:

{{ JSON.stringify({

executionId: $execution.id,

workflowName: $workflow.name || 'Connection-Team-Workflow',

timestamp: new Date().toISOString(),

website\_type: $node["Clean & Parse AI JSON"].json.website\_type,

business\_name: $node["Clean & Parse AI JSON"].json.business\_name,

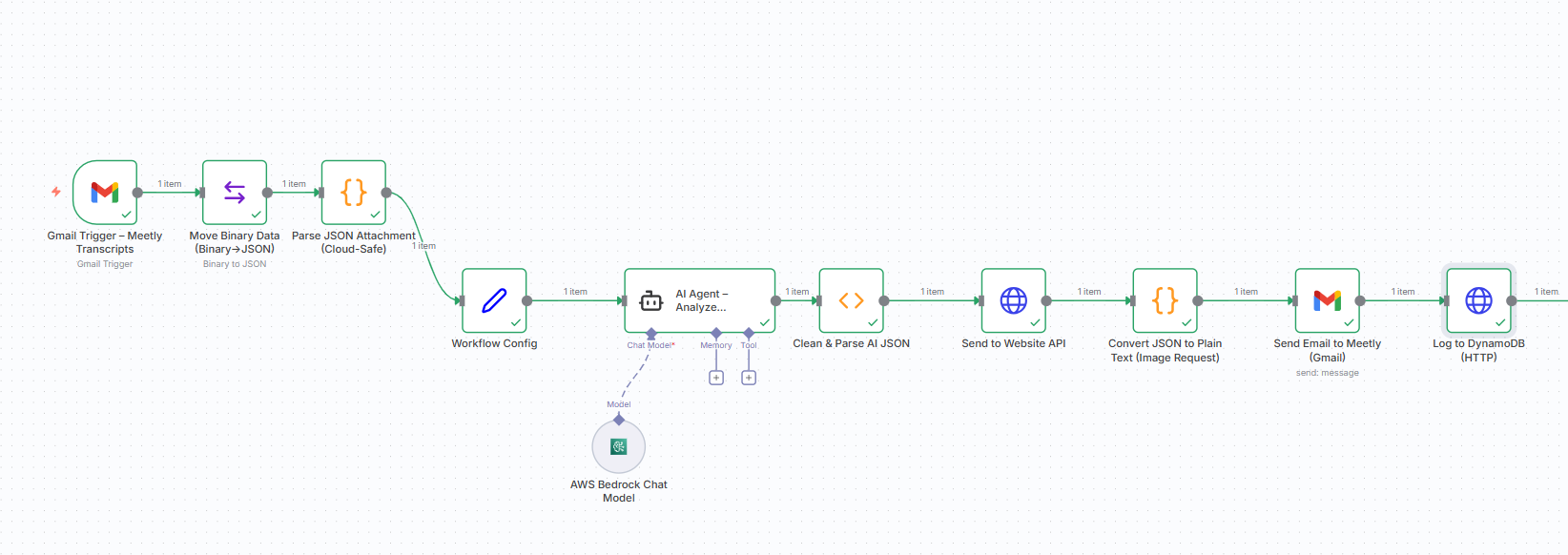
bedrock\_model: $node["AWS Bedrock Chat Model"].json.model,

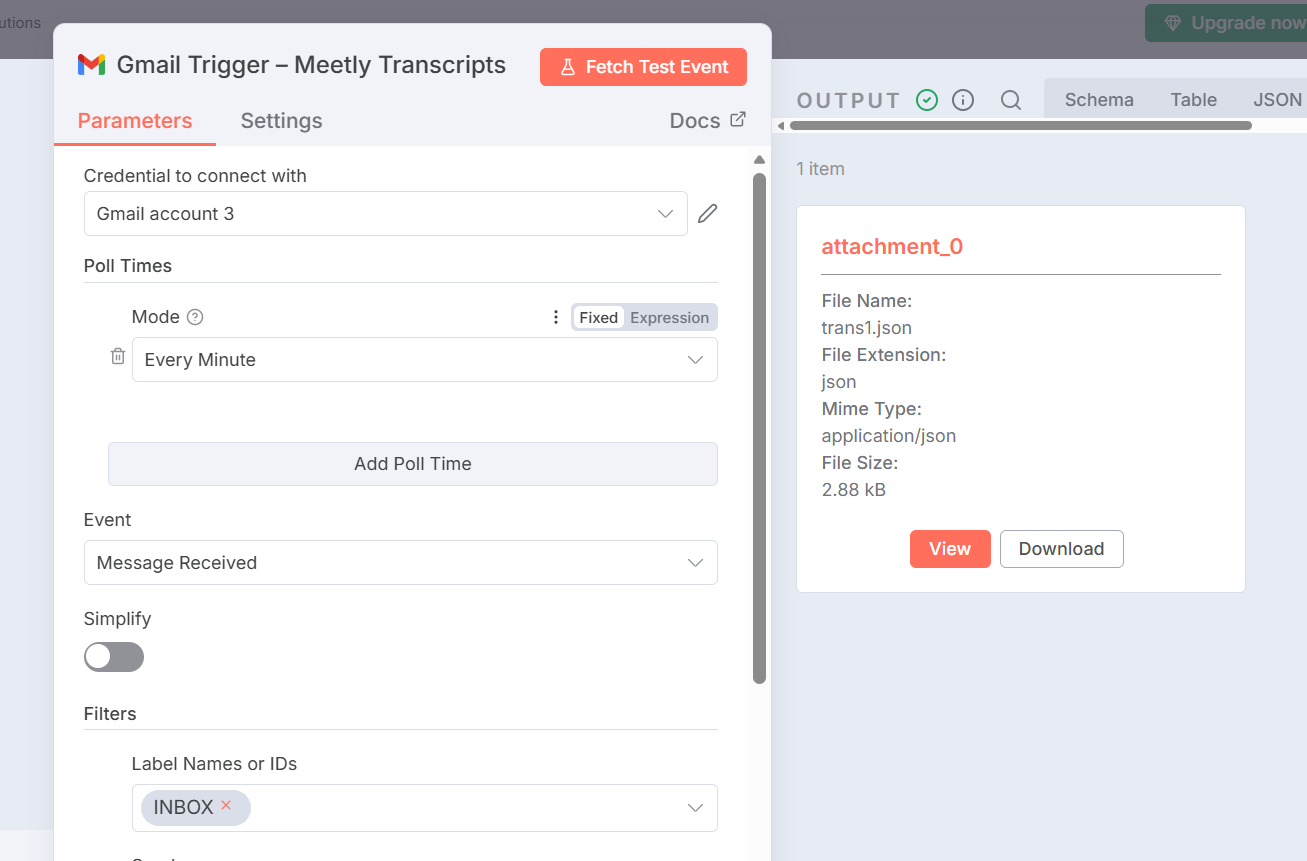
ai\_output\_summary: $node["AI Agent – Analyze Transcript / JSON"].json.output,

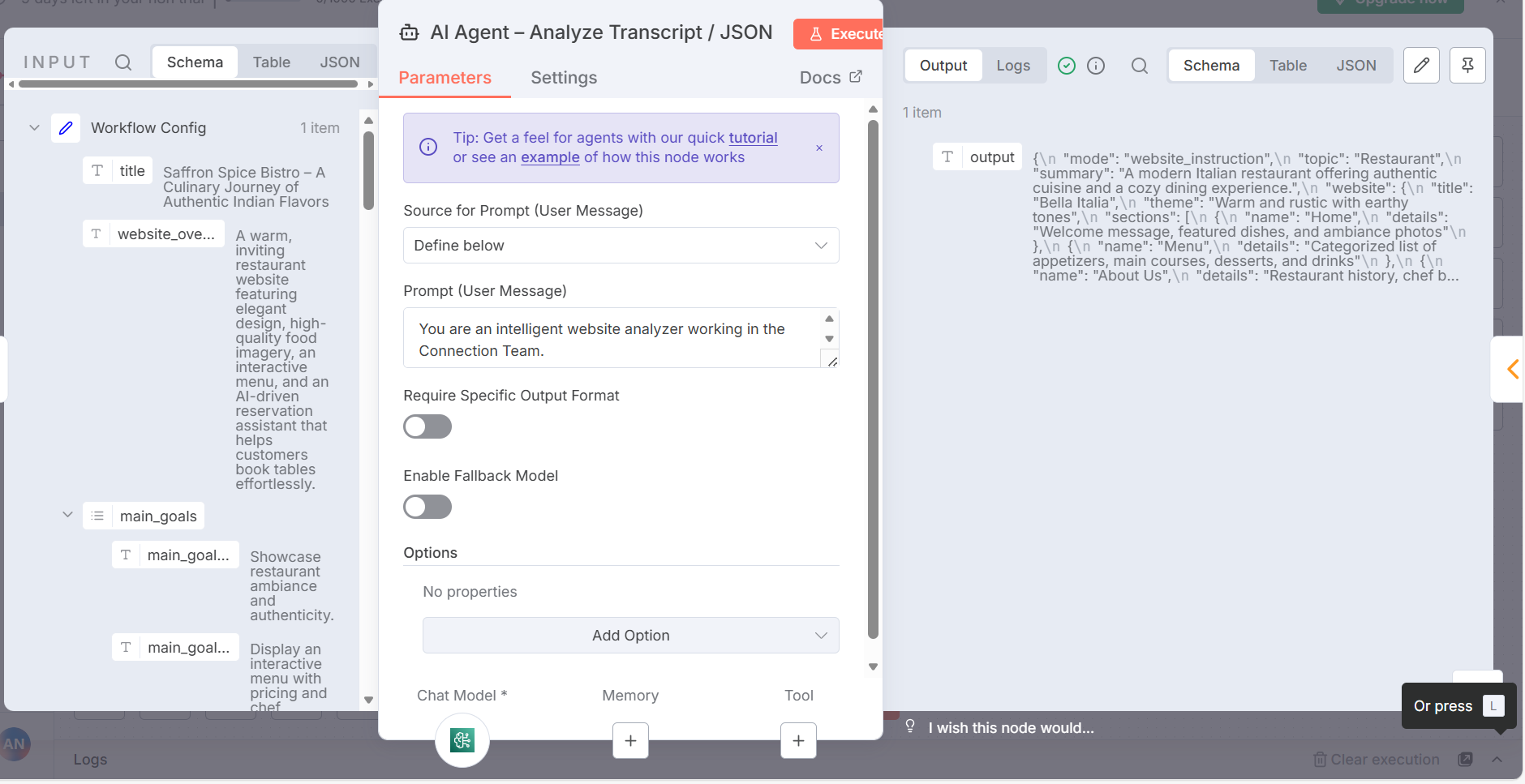
status: 'completed'

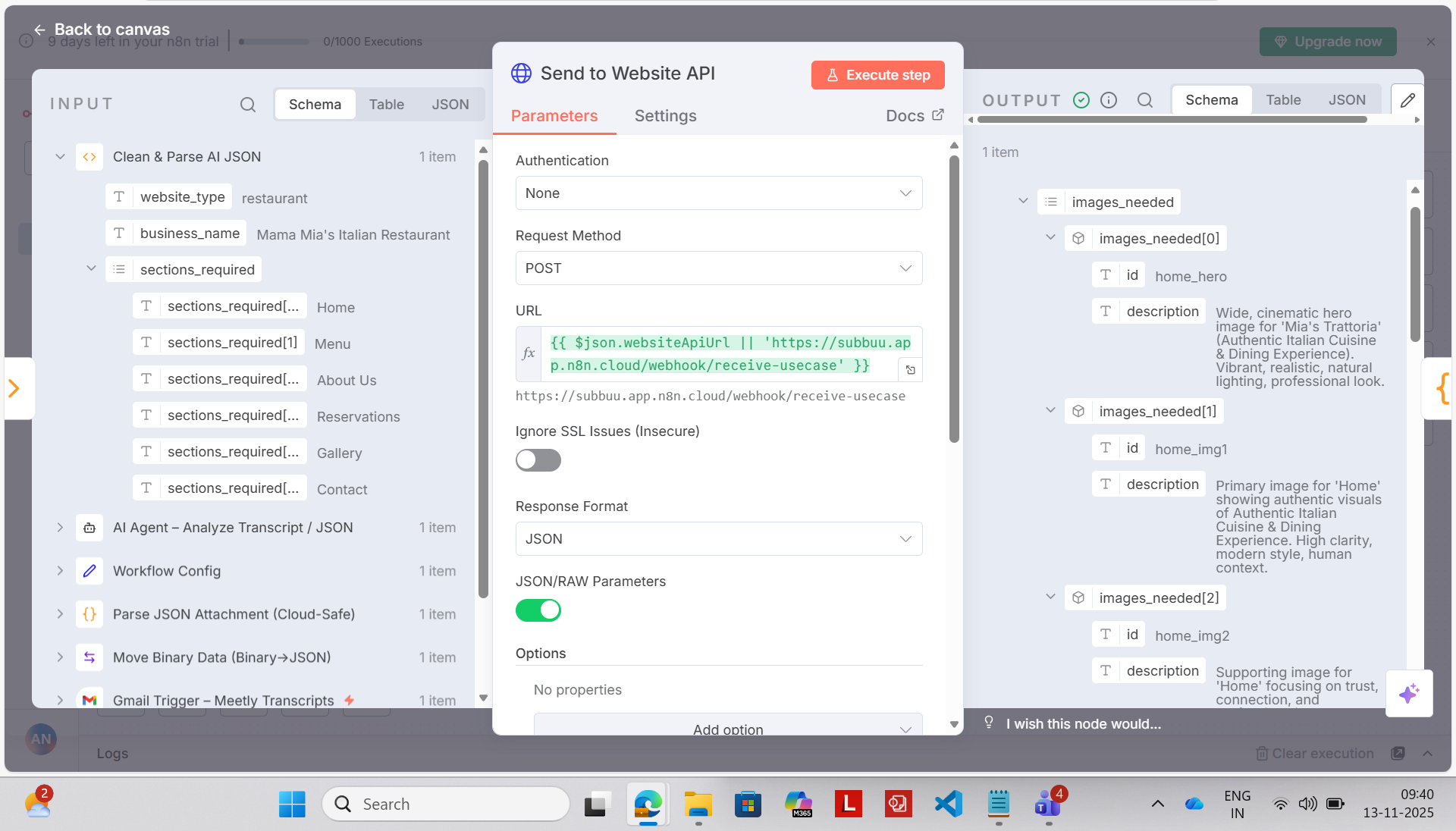
}) }}

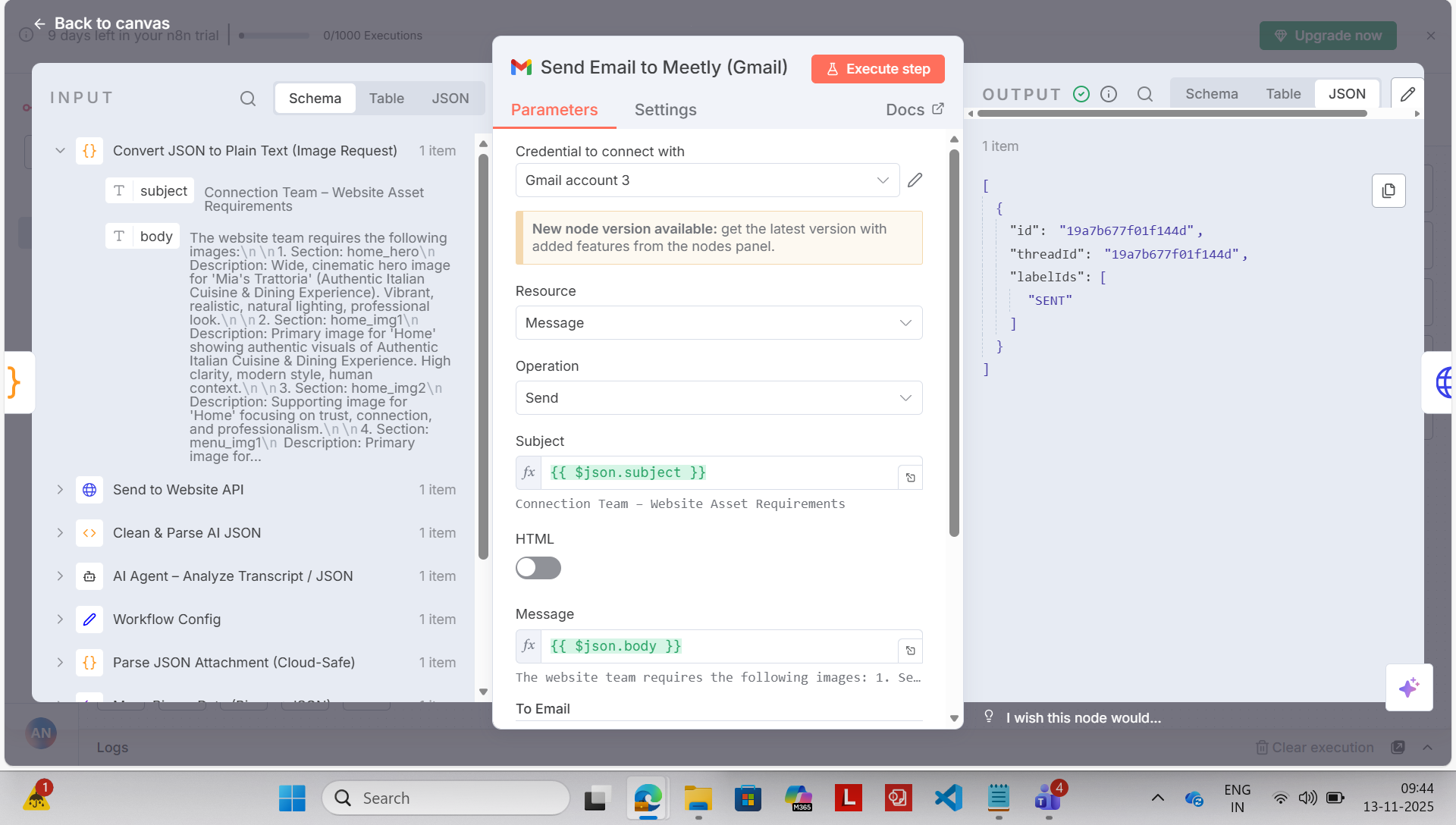
**Screenshots**

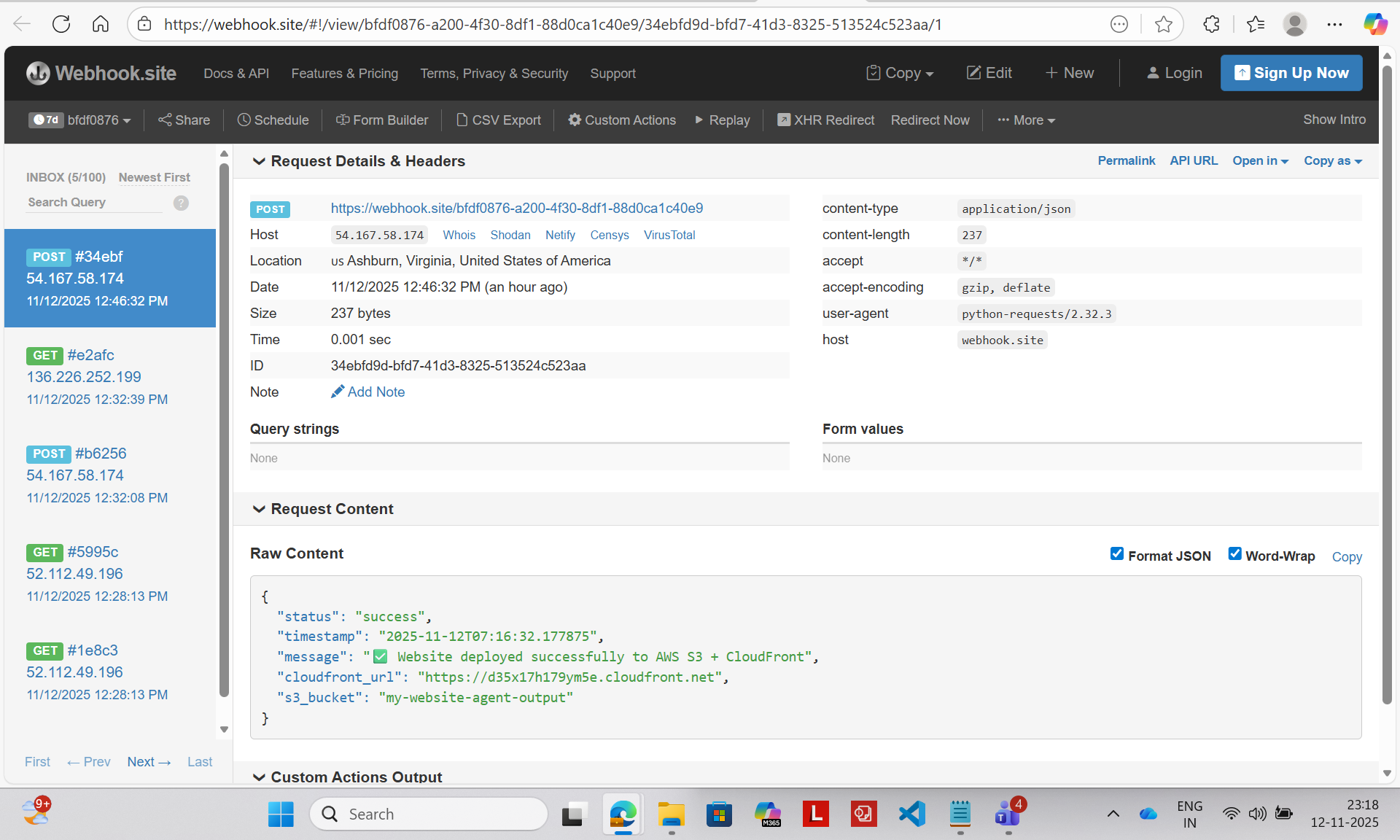
Workflow  




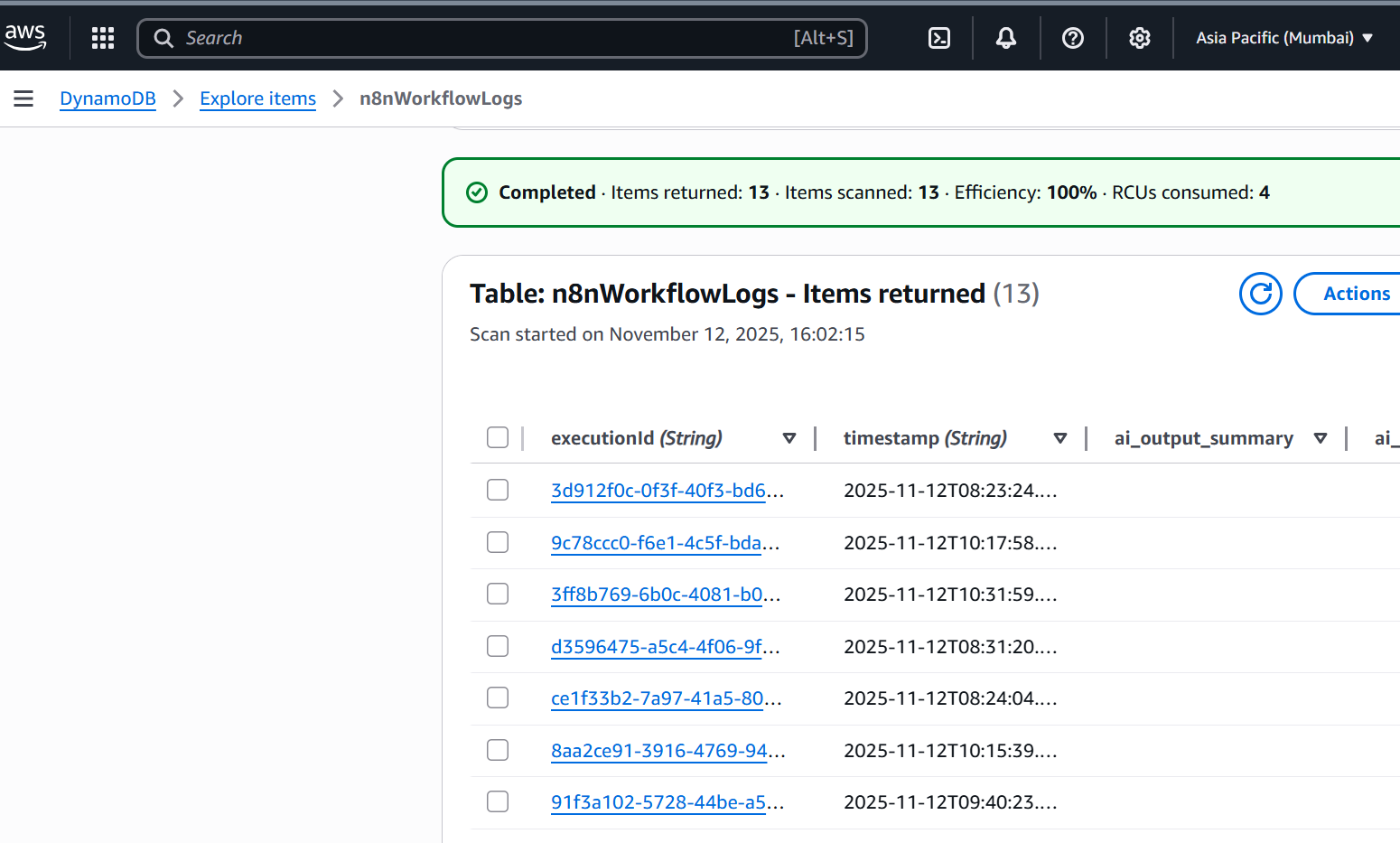


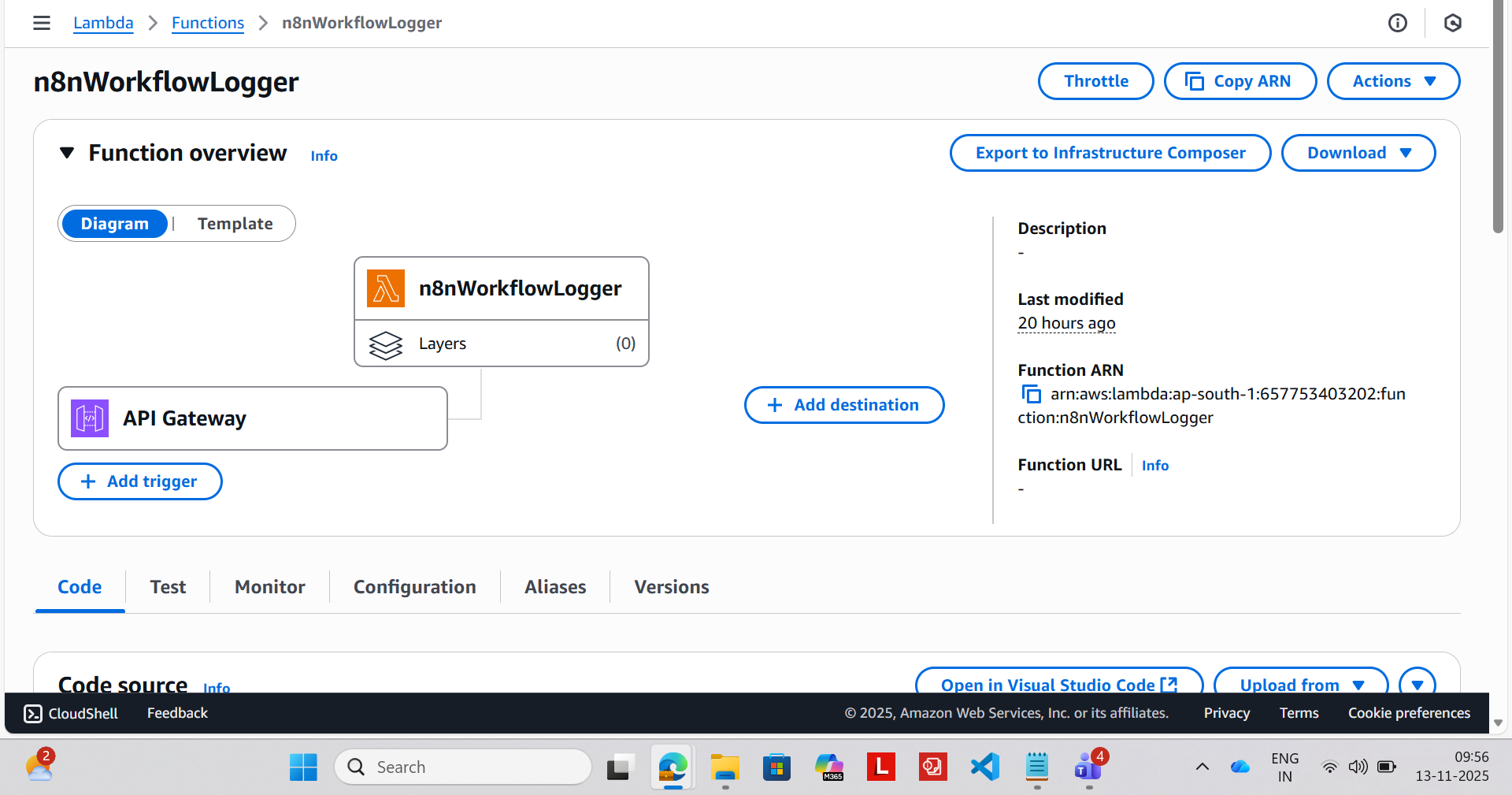


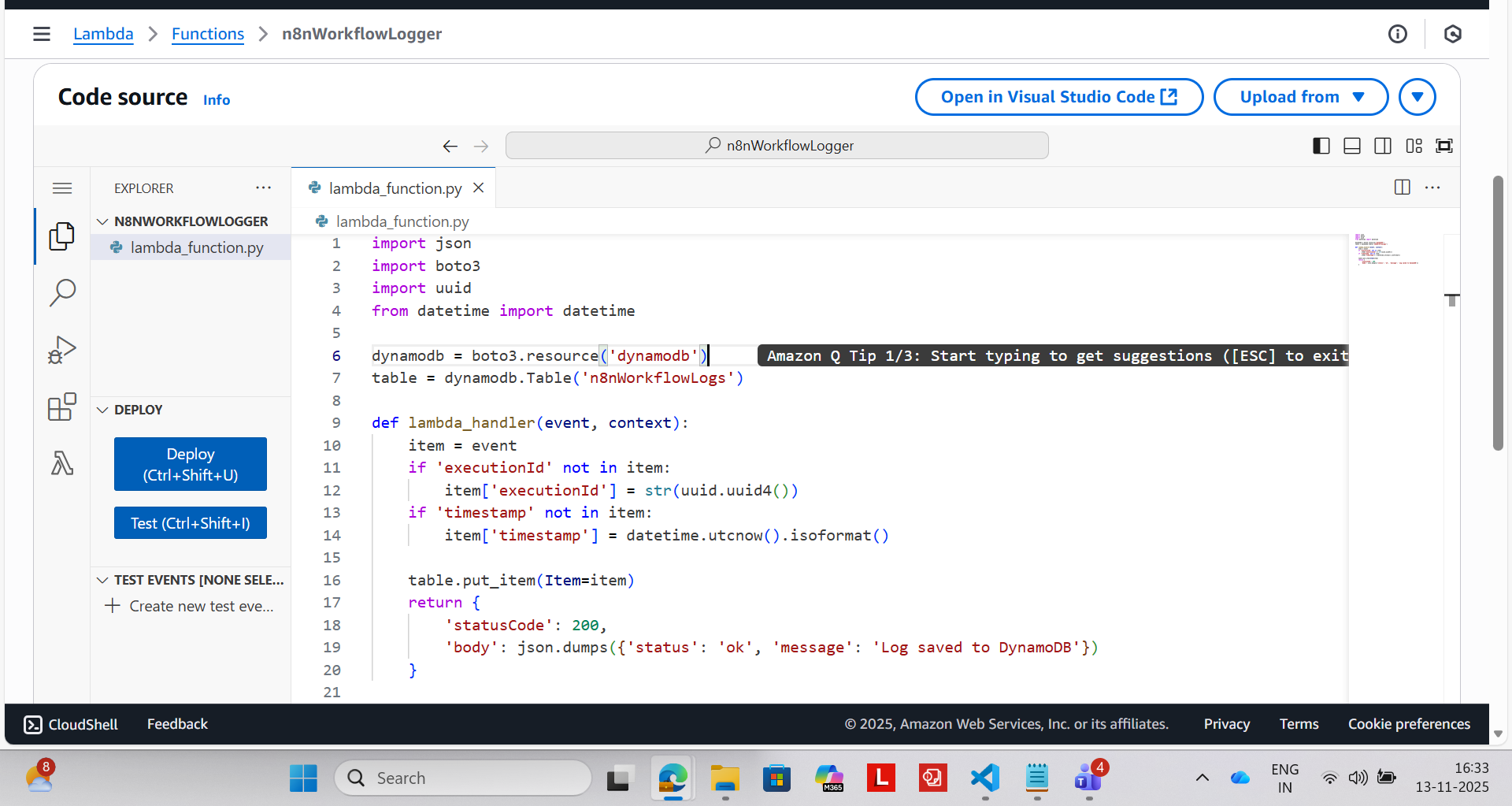


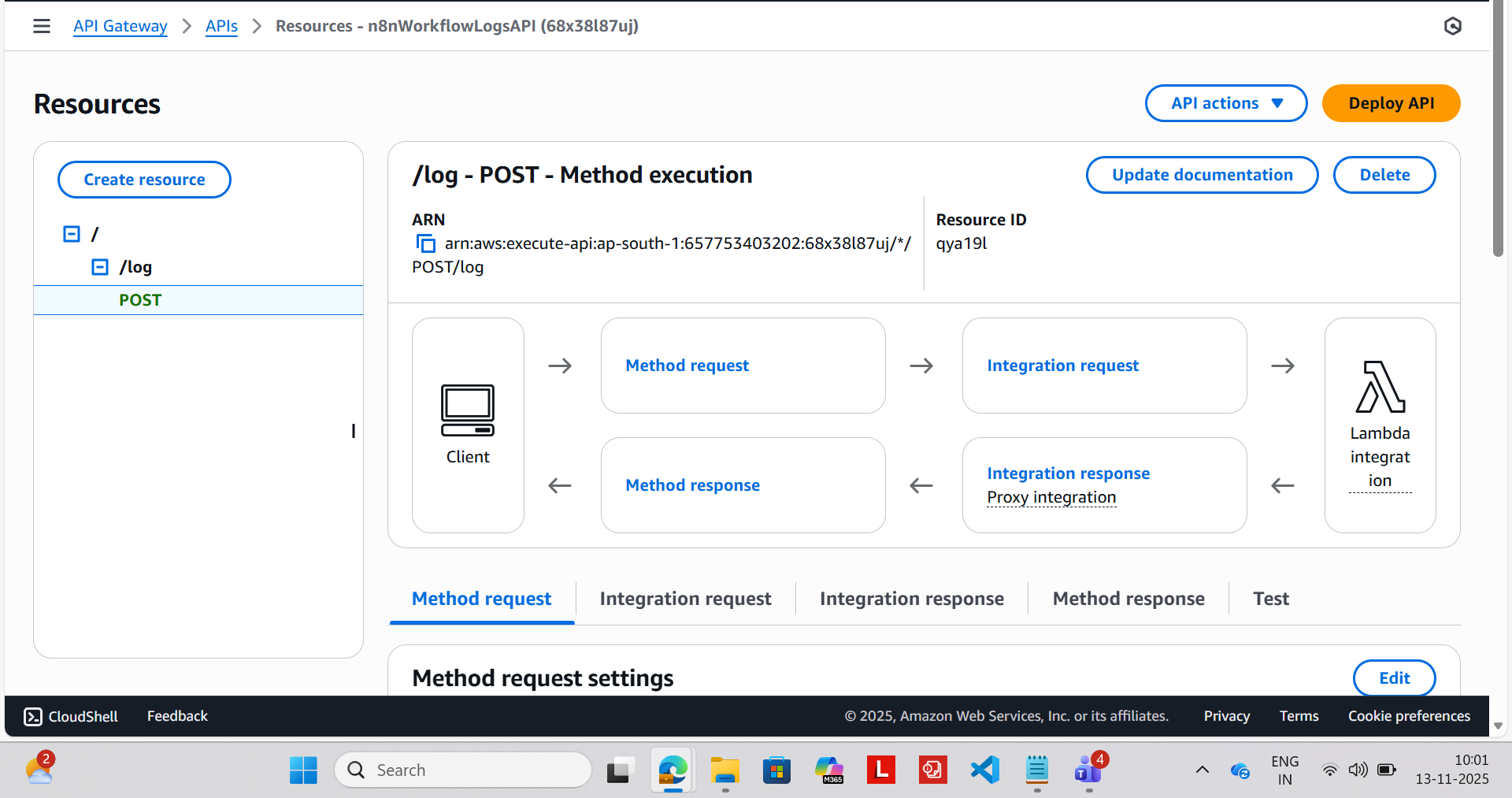


AWS Configuration









**Conclusion**

The **Communication Bridge for Website Development Team** successfully automates the end-to-end collaboration between the Meetly Team and the Website Development Team through an intelligent, AI-powered workflow.

By acting as the **automation and intelligence layer**, the Connection Team’s workflow ensures that meeting transcripts are instantly analyzed, transformed into structured website instructions, and shared seamlessly with the development team.

This system eliminates manual intervention, reduces communication delays, and provides complete visibility through automated logging in AWS DynamoDB.  
 Using tools like **n8n**, **AWS Bedrock**, **Lambda**, and **API Gateway**, the workflow delivers a scalable, reliable, and efficient coordination framework.

Overall, the project demonstrates how **AI-driven automation can streamline cross-team communication**, improve accuracy, and accelerate the website development lifecycle.

Future Enhancements

* Add **automatic prompt optimization** based on historical AI results.
* Extend DynamoDB logging for AI token usage cost tracking.
* Deploy workflow as **Docker container** via AWS ECS/ECR for higher scalability.

### **Final Thoughts “AI won’t just automate workflows — it will manage entire ecosystems of communication.**

**Github Repository :** [**Sujitha-976/COMMUNICATION-BRIDGE-FOR-WEBSITE-DEVELOPMENT-TEAM**](https://github.com/Sujitha-976/COMMUNICATION-BRIDGE-FOR-WEBSITE-DEVELOPMENT-TEAM)