amazon-bedrock-samples

[root@ip-172-31-94-192 ~]# git clone https://github.com/aws-samples/amazon-bedrock-samples.git

Cloning into 'amazon-bedrock-samples'...

remote: Enumerating objects: 16171, done.

remote: Counting objects: 100% (2389/2389), done.

remote: Compressing objects: 100% (753/753), done.

remote: Total 16171 (delta 1941), reused 1680 (delta 1633), pack-reused 13782 (from 3)

Receiving objects: 100% (16171/16171), 1.05 GiB | 38.69 MiB/s, done.

Resolving deltas: 100% (8645/8645), done.

Updating files: 100% (1954/1954), done.

[root@ip-172-31-94-192 ~]# ll

total 8

drwxr-xr-x 20 root root 4096 Jul 8 07:42 amazon-bedrock-samples

drwxr-xr-x 2 root root 6 Jul 7 16:16 efs

-rwxrwxrwx 1 root root 301 Jul 2 02:30 toggle-maintenance.sh

[root@ip-172-31-94-192 ~]# cd /root/amazon-bedrock-samples/agents-and-function-calling/bedrock-agents/use-case-examples/insurance-claim-lifecycle-automation/shell

-bash: cd: cd: No such file or directory

[root@ip-172-31-94-192 ~]# cd amazon-bedrock-samples/agents-and-function-calling/bedrock-agents/use-case-examples/insurance-claim-lifecycle-automation/shell

[root@ip-172-31-94-192 shell]# ll

total 8

-rwxr-xr-x 1 root root 3334 Jul 8 07:42 create-customer-resources.sh

-rwxr-xr-x 1 root root 938 Jul 8 07:42 delete-customer-resources.sh

[root@ip-172-31-94-192 shell]# vi create-customer-resources.sh

[root@ip-172-31-94-192 shell]# chmod +x create-customer-resources.sh

[root@ip-172-31-94-192 shell]# aws configure

Default region name [None]: us-east-1

Default output format [None]:

[root@ip-172-31-94-192 shell]# sh create-customer-resources.sh

Creating S3 bucket: insurance-app-demo-resources-172982781615-us-east-1

make\_bucket: insurance-app-demo-resources-172982781615-us-east-1

Uploading agent files to S3...

upload: ../agent/api-schema/create\_claim.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/api-schema/create\_claim.json

upload: ../agent/api-schema/send\_reminder.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/api-schema/send\_reminder.json

upload: ../agent/api-schema/gather\_evidence.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/api-schema/gather\_evidence.json

upload: ../agent/knowledge-base-assets/AccidentImages\_file\_requirements.docx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/AccidentImages\_file\_requirements.docx

upload: ../agent/knowledge-base-assets/Claim\_3b45c-9d\_Amounts.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_3b45c-9d\_Amounts.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_3b45c-9d\_RepairEstimate.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_3b45c-9d\_RepairEstimate.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_2s34w-8x\_Amounts.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_2s34w-8x\_Amounts.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_5t16u-7v\_Amounts.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_5t16u-7v\_Amounts.pdf

upload: ../agent/knowledge-base-assets/Claim\_3b45c-9d\_RepairEstimate.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_3b45c-9d\_RepairEstimate.pdf

upload: ../agent/knowledge-base-assets/Driverlicense\_file\_requirements.docx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Driverlicense\_file\_requirements.docx.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_5t16u-7v\_RepairEstimate.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_5t16u-7v\_RepairEstimate.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Driverlicense\_file\_requirements.docx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Driverlicense\_file\_requirements.docx

upload: ../agent/knowledge-base-assets/External-Insurance-FAQs.xlsx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/External-Insurance-FAQs.xlsx.metadata.json

upload: ../agent/knowledge-base-assets/External-Insurance-FAQs.xlsx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/External-Insurance-FAQs.xlsx

upload: ../agent/knowledge-base-assets/Claim\_2s34w-8x\_RepairEstimate.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_2s34w-8x\_RepairEstimate.pdf

upload: ../agent/knowledge-base-assets/Internal-Insurance-FAQs.xlsx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Internal-Insurance-FAQs.xlsx.metadata.json

upload: ../agent/knowledge-base-assets/AccidentReport\_file\_requirements.docx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/AccidentReport\_file\_requirements.docx.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_2s34w-8x\_RepairEstimate.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_2s34w-8x\_RepairEstimate.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_5t16u-7v\_Amounts.pdf.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_5t16u-7v\_Amounts.pdf.metadata.json

upload: ../agent/knowledge-base-assets/Claim\_3b45c-9d\_Amounts.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_3b45c-9d\_Amounts.pdf

upload: ../agent/knowledge-base-assets/VehicleRegistration\_file\_requirements.docx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/VehicleRegistration\_file\_requirements.docx.metadata.json

upload: ../agent/knowledge-base-assets/VehicleRegistration\_file\_requirements.docx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/VehicleRegistration\_file\_requirements.docx

upload: ../agent/lambda/action-groups/send\_reminder.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/send\_reminder.py

upload: ../agent/lambda/action-groups/create\_claim.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/create\_claim.py

upload: ../agent/knowledge-base-assets/AccidentReport\_file\_requirements.docx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/AccidentReport\_file\_requirements.docx

upload: ../agent/lambda/action-groups/gather\_evidence.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/gather\_evidence.py

upload: ../agent/knowledge-base-assets/Claim\_5t16u-7v\_RepairEstimate.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_5t16u-7v\_RepairEstimate.pdf

upload: ../agent/lambda/action-groups/create\_claim.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/create\_claim.zip

upload: ../agent/lambda/data-loader/claims.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/data-loader/claims.json

upload: ../agent/lambda/action-groups/send\_reminder.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/send\_reminder.zip

upload: ../agent/lambda/data-loader/loader\_deployment\_package.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/data-loader/loader\_deployment\_package.zip

upload: ../agent/knowledge-base-assets/AccidentImages\_file\_requirements.docx.metadata.json to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/AccidentImages\_file\_requirements.docx.metadata.json

upload: ../agent/streamlit/bedrock\_streamlit.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/streamlit/bedrock\_streamlit.py

upload: ../agent/streamlit/bedrock\_logo.png to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/streamlit/bedrock\_logo.png

upload: ../agent/streamlit/setup-streamlit-env.sh to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/streamlit/setup-streamlit-env.sh

upload: ../agent/streamlit/requirements.txt to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/streamlit/requirements.txt

upload: ../agent/knowledge-base-assets/Claim\_2s34w-8x\_Amounts.pdf to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Claim\_2s34w-8x\_Amounts.pdf

upload: ../agent/lambda/data-loader/index.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/data-loader/index.py

upload: ../agent/knowledge-base-assets/Internal-Insurance-FAQs.xlsx to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/knowledge-base-assets/Internal-Insurance-FAQs.xlsx

upload: ../agent/lambda/action-groups/gather\_evidence.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/action-groups/gather\_evidence.zip

upload: ../agent/lambda/lambda-layer/cfnresponse-layer.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/lambda-layer/cfnresponse-layer.zip

upload: ../agent/streamlit/sigv4.py to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/streamlit/sigv4.py

upload: ../agent/lambda/lambda-layer/bedrock-agents-layer.zip to s3://insurance-app-demo-resources-172982781615-us-east-1/agent/lambda/lambda-layer/bedrock-agents-layer.zip

Publishing Bedrock Agents Lambda Layer...

Publishing cfnresponse Lambda Layer...

Deploying CloudFormation Stack: insurance-app-demo

Waiting for changeset to be created..

Waiting for stack create/update to complete

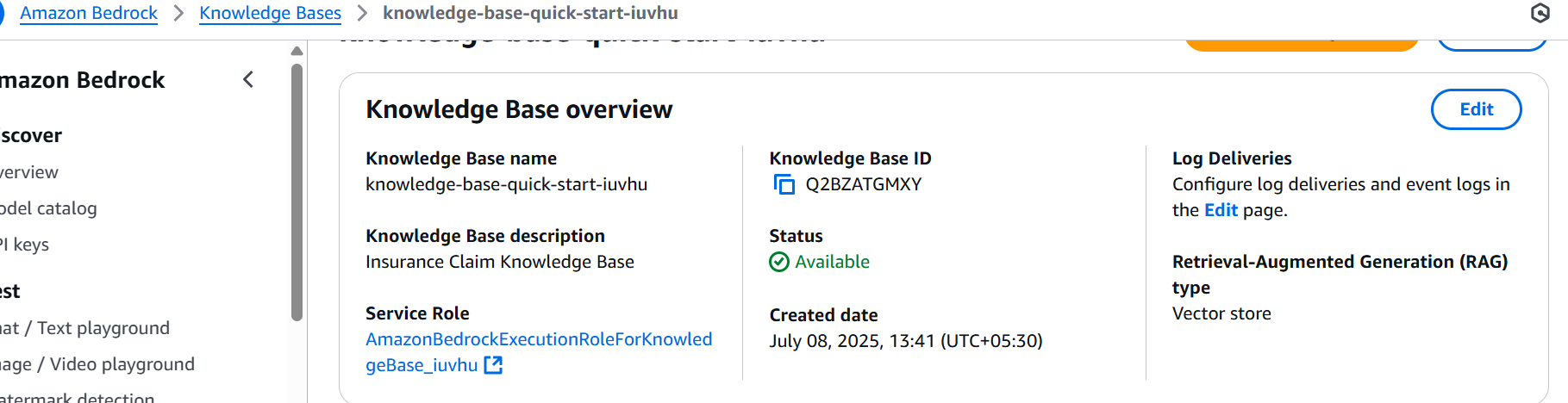
Successfully created/updated stack - insurance-app-demo

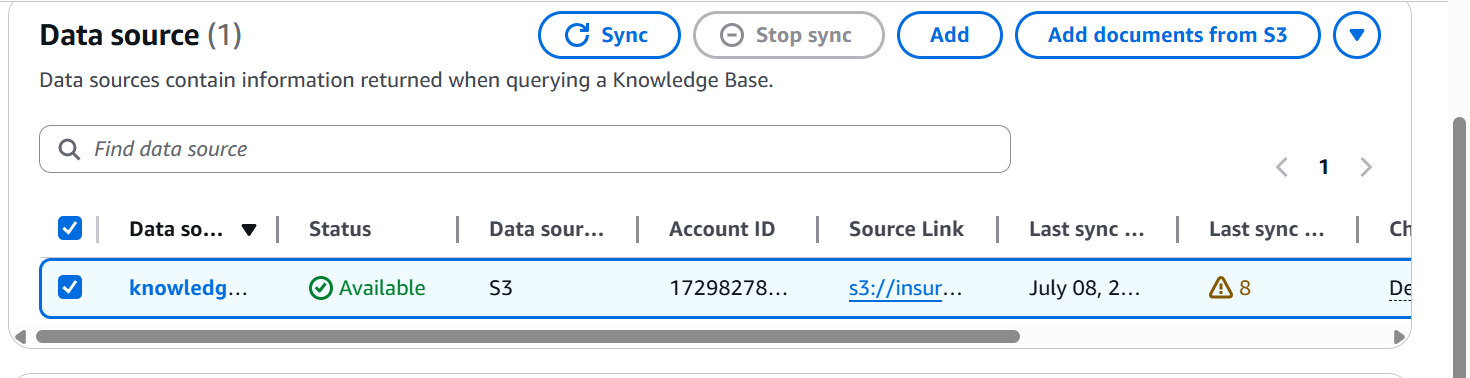
✅ Stack deployment initiated. Check status in the AWS CloudFormation console or use the command below:

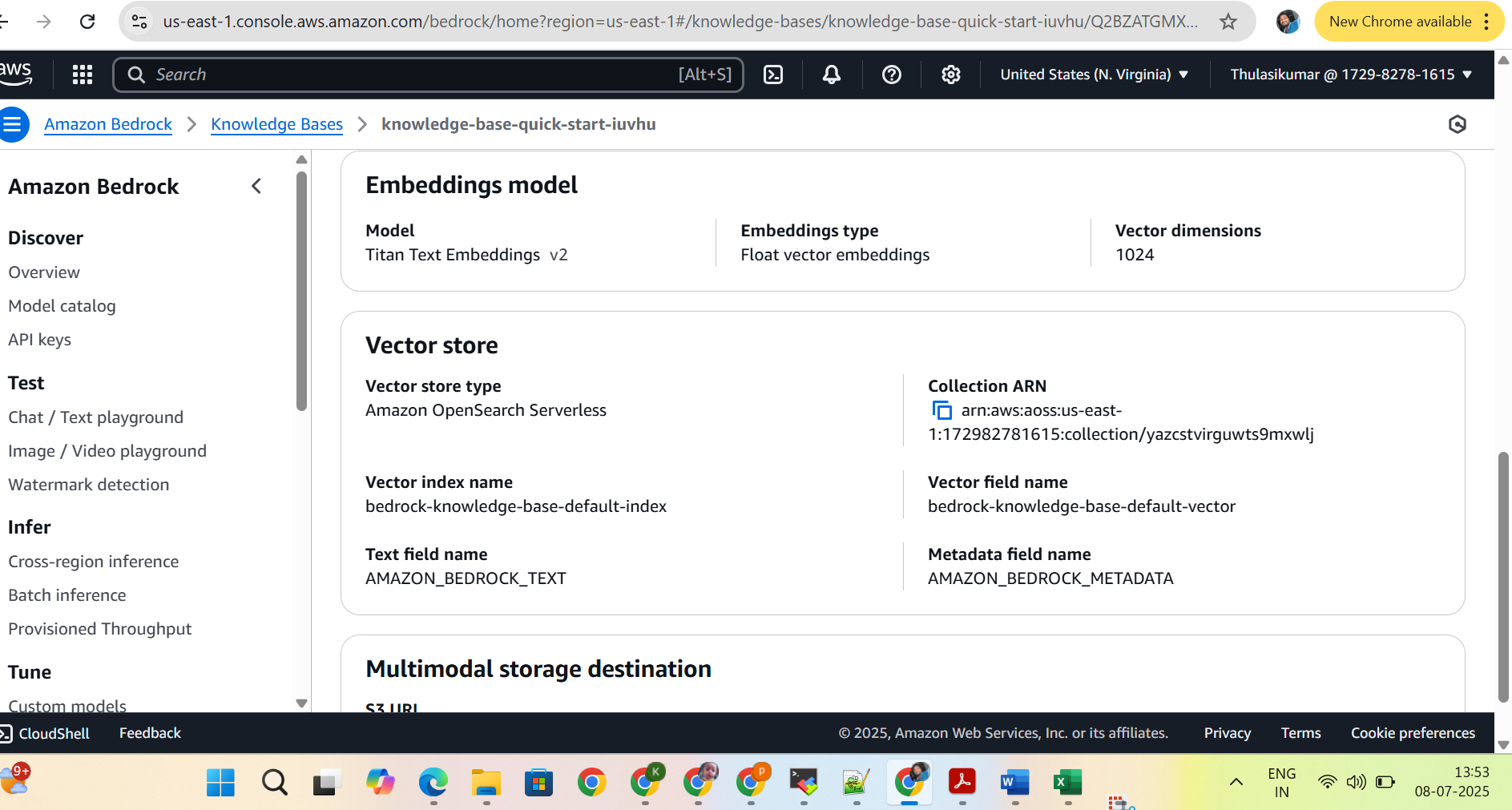
aws cloudformation describe-stacks --stack-name insurance-app-demo --region us-east-1 --query "Stacks[0].StackStatus"

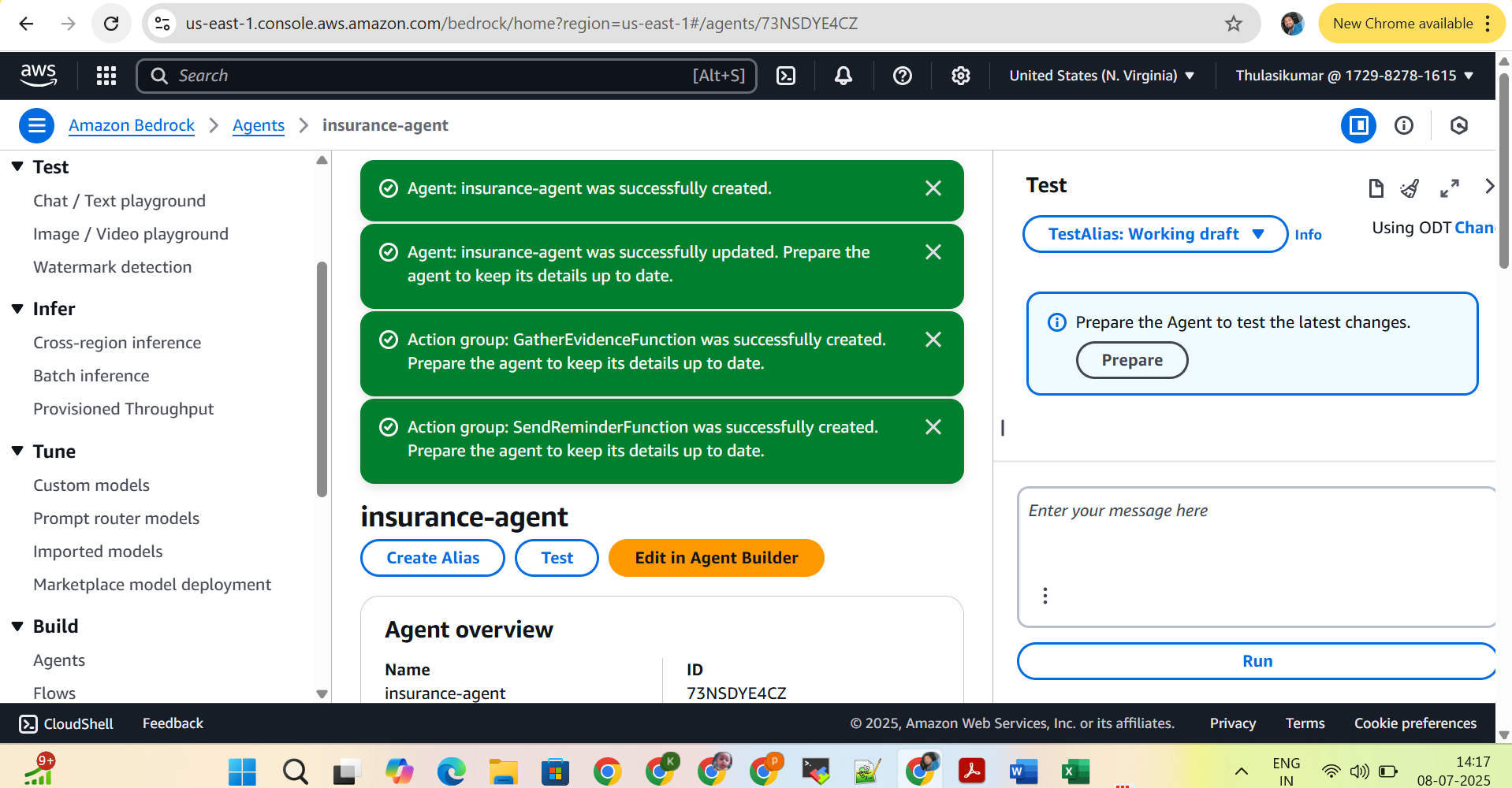
[root@ip-172-31-94-192 shell]# cd

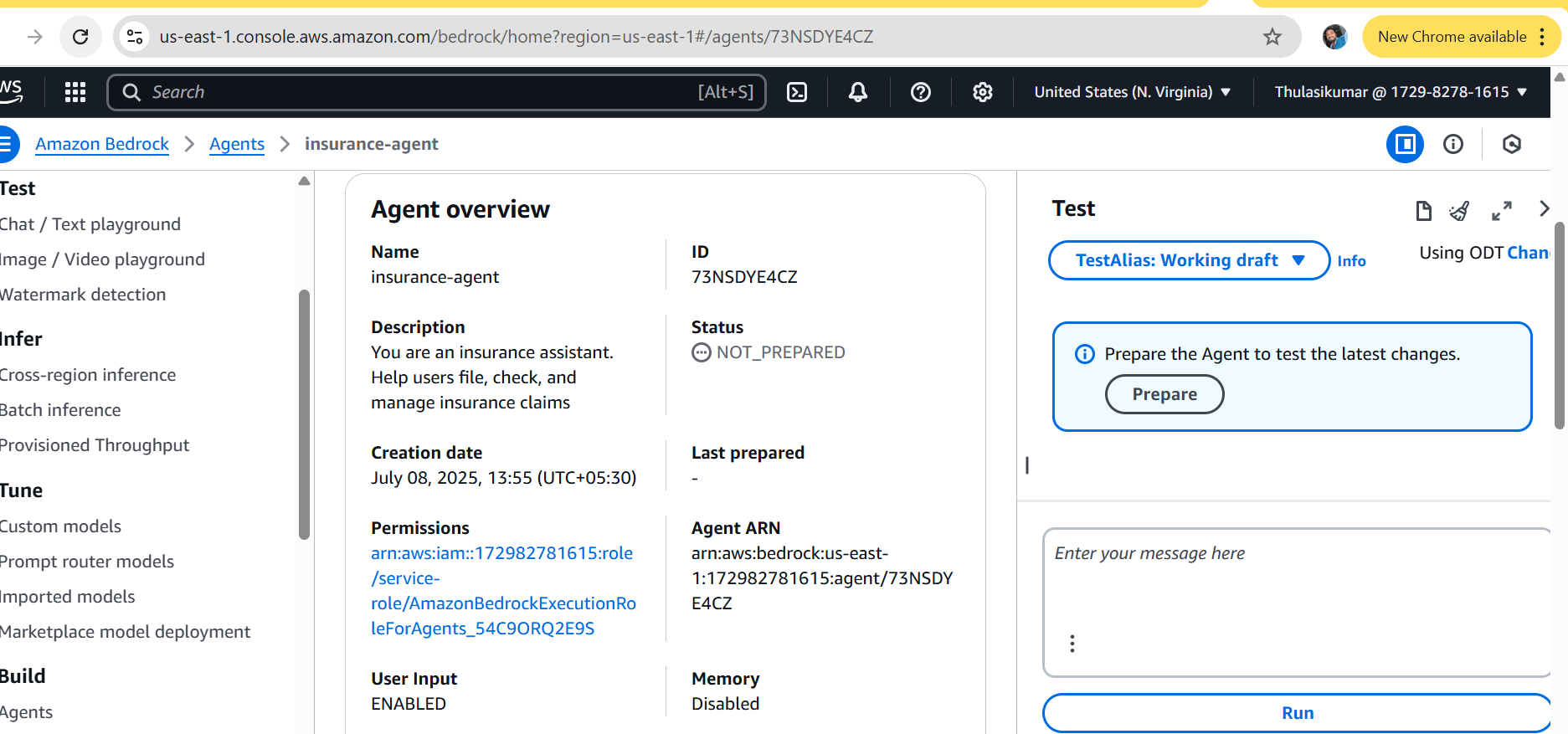
[root@ip-172-31-94-192 ~]# cd amazon-bedrock-samples/agents-and-function-calling/bedrock-agents/use-case-examples/insurance-claim-lifecycle-automation/agent/streamlit/



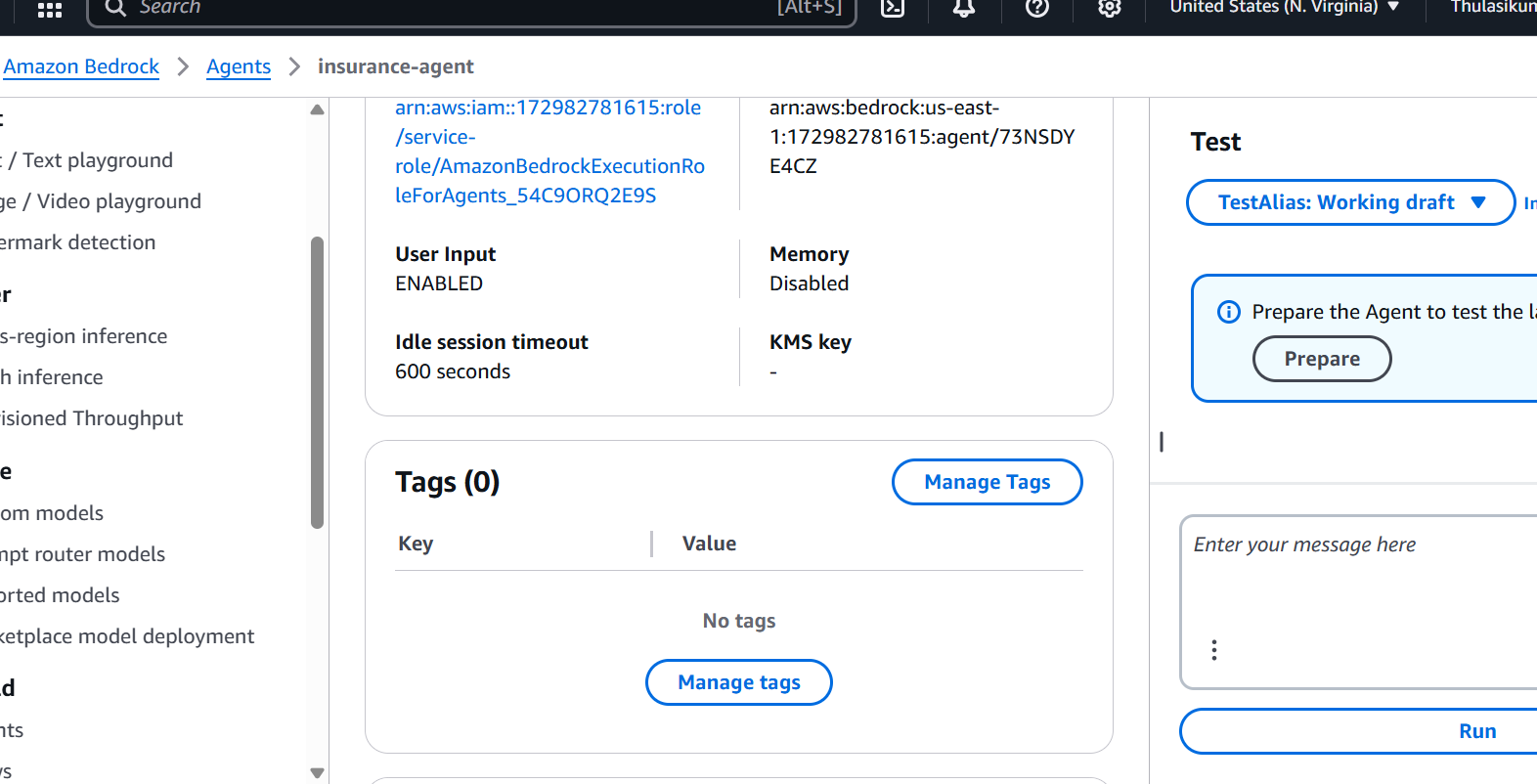








You are an insurance assistant. Help users file, check, and manage insurance claims



User ───▶ Bedrock Agent

│

┌─────┴────────┐

Query KB Call Action Group

(S3+Embed) (via Lambda)

│

┌──────┼─────────┐

CreateClaim GatherEvidence SendReminder

│ │ │

Write to DB Read from DB Send SNS

│

DynamoDB

**Step 1: Create the Agent (save it once)**

1. Go back to the **Agent creation screen**.
2. Fill the fields:

| **Field** | **Value** |
| --- | --- |
| **Agent name** | insurance-agent |
| **Description** | Helps customers file, manage, and check insurance claims. |
| **Agent resource role** | ✅ **Select AmazonBedrockExecutionRoleForAgents\_XXXX** (already created in your case) |
| **Foundation model** | Choose any supported model (e.g. Anthropic Claude 3 Sonnet) |
| **Instructions** | You must enter a minimum of 40 characters. Example: |
| *"You are an intelligent insurance assistant that helps users create, manage, and retrieve claims using APIs and documents."* |  |
| **User input** | ✅ Enabled (recommended) |
| **Idle session timeout** | Default 600 (10 mins) is okay |

1. Click **“Save and Exit”** (not just “Save”)

**🔹 Step 2: Now Go Back and Add Action Groups**

Now that the Agent has a saved IAM role, you can **edit the agent** and:

* Go to the **Action Groups** section
* Click **Add**
* Use “**Define with API schemas**”
* Select **existing Lambda function**
* Attach the **OpenAPI schema** from S3

You will now **no longer see the error**, and you can add all 3 action groups.

## NEXT ****Pending Steps****

### 🔹 1. **Complete Agent Configuration**

You're currently editing the agent screen. Now finish these items:

#### ✅ Fill "Instructions for the Agent" (required)

Provide a clear instruction, minimum 40 characters. Example:

“You are an insurance assistant. Help users file claims, gather documents, and send reminders using provided APIs and knowledge base.”

#### ✅ Attach the Knowledge Base

1. Scroll to the **Knowledge Bases** section.
2. Click **“Add”**
3. Select your **synced Knowledge Base** from the list (e.g. knowledge-base-quick-start-\*)
4. Save it.

#### 🔘 (Optional) Enable Session Summarization

* You can keep it **Disabled** for now unless you're using a model that supports memory and summarization (like Claude 3).

### 🔹 2. **Save and Exit the Agent**

Once all required fields are filled:

👉 Click **"Save and Exit"** (not just "Save") to finalize your agent setup.

### 🔹 3. **Test the Agent**

Now go to:

**Bedrock Console → Agents → Select your agent → Test Agent**

#### 🔹 Try sample utterances:

text

CopyEdit

- "Create a new claim"

- "Send reminder for missing documents"

- "Gather evidence for claim ABC123"

- "What is the total amount claimed for [9x87y-3z](https://us-east-1.console.aws.amazon.com/dynamodbv2/home?region=us-east-1#edit-item?itemMode=2&pk=9x87y-3z&route=ROUTE_ITEM_EXPLORER&table=insurance-app-demo-ExistingClaims)?"

### **How to Add the Knowledge Base**

1. **Knowledge Base**  
   ✔️ Select: knowledge-base-quick-start-n5usi  
   (Already done — ✅ good)
2. **Knowledge Base instructions for Agent**  
   ➕ Provide a sentence to tell the agent **how to use the KB**.

#### 📘 Example instruction:

“Use this knowledge base to answer questions about claim documents, process steps, and policy-related information.”

You can also try:

“Refer to this knowledge base when users ask about claims, evidence requirements, and internal FAQs.”

1. ✔️ Click **Add**

### **Step 1: Prepare the Agent**

Right now, the agent status is NOT\_PREPARED, which means it hasn't been compiled for use.

1. Go to the **Agent page**.
2. Click the **“Prepare”** button (top bar near “Test” and “Save”).
3. Wait a few seconds until the status changes to **PREPARED**.

Once prepared, a new version will be created automatically.

### ✅ **Step 2: Create a Version**

After preparation completes:

1. Go to the **Versions** tab.
2. You’ll see version **v1** or a similar label.
3. Click to **select** the version and optionally give it a name (or leave default).

### ✅ **Step 3: Create an Alias**

Now, link a test alias to your version:

1. Click **“Create”** under the **Aliases** tab.
2. Provide:
   * **Alias name:** e.g., dev, test, or live
   * Select the version (e.g., v1)
3. Click **Create**.

You now have an **active agent with a version + alias**, ready to test or integrate!

### ✅ **Step 4: Test the Agent**

Now, click **“Test”** from the top bar.

Try utterances like:

* “Create a new insurance claim.”
* “Send reminders for missing documents.”
* “Gather documents for claim ID 123.”
* “What is the status of claim 456?”

You'll see:

* How your instructions work.
* Action group invocations.
* Knowledge Base retrievals.

### **Form Values to Use:**

* **Company name**:  
  Sparsh Insurance Solutions Pvt Ltd  
  (Or use your real company name)
* **Company website URL**:  
  https://sparshinsurance.example.com  
  (Use your actual site if available; if not, placeholder is fine.)
* **What industry do you operate in?**  
  Insurance
* **Who are your intended users?**  
  Internal employees  
  (Select this if only your team uses the system. If it’s customer-facing, also check *External users*.)

### ✅ Use Case Description (Recommended Copy):

"We are building an internal automation system to streamline the insurance claim process using Amazon Bedrock Agents. The system uses Lambda functions and a knowledge base to guide users through filing and managing claims, retrieving required documents, and providing summaries. The goal is to reduce manual work and improve response accuracy."

## Current Setup Status

| **Component** | **Status** | **Notes** |
| --- | --- | --- |
| **Agent Name** | insurance-agent | Ready |
| **Model** | Claude 3 Haiku | Access granted |
| **Instructions** | ✅ Provided | Solid |
| **Action Groups** | ✅ 3 connected | Via Lambda |
| **Knowledge Base** | ✅ Attached & synced | knowledge-base-quick-start-n5usi |
| **IAM Roles** | ✅ Configured | AmazonBedrockExecutionRoleForAgents\_... |
| **Agent Prepared** | ✅ Done | Success message seen |

Watch how it uses:

* ✅ Action groups → via Lambda
* ✅ Knowledge base → to give policy or document info

## ****Insurance Claims Application — High-Level Flow****

### 🔁 1. **User Interaction**

* User types a request:  
  “I want to file a new claim” or “Show me documents for claim 2s34w-8x”
* This goes to the **Amazon Bedrock Agent**

### 🧠 2. **Bedrock Agent (Main Brain)**

* Orchestrates the request
* Chooses whether to:
  + Call a **Lambda function** (via **Action Group**) for dynamic tasks
  + Query the **Knowledge Base** for static info (policy/process/etc.)

### 🧩 3. **Action Groups (via Lambda Functions)**

Each Action Group does a specific task:

* **CreateClaimFunction** → writes new claim to **DynamoDB**
* **GatherEvidenceFunction** → retrieves claim details from **DynamoDB**
* **SendReminderFunction** → sends SNS notification if documents are pending

### 📚 4. **Knowledge Base**

* Uses **S3 files (PDF, DOCX, XLSX)** + embedding model
* Answers FAQs like:
  + “What documents are needed?”
  + “What is the claim process?”

### 🗄️ 5. **DynamoDB (Structured Data Storage)**

* Stores **claim records**
* Used by Lambda to:
  + Look up status
  + Fetch amounts, repair info
  + Track progress of claim submission

### 📤 6. **S3 Bucket**

* Hosts:
  + Knowledge base documents
  + OpenAPI specs for Lambdas
  + Streamlit frontend (optional)

### 📣 7. **SNS (Notifications)**

* Sends **email or SMS** reminders via Lambda
* Used by SendReminderFunction

### 🧪 8. **Testing & Deployment**

* You test via **Bedrock Agent Test Console**
* You deploy agent using **Alias + API Gateway** (for integration into apps)

git clone https://github.com/aws-samples/amazon-bedrock-samples.git

cd amazon-bedrock-samples/agents-and-function-calling/bedrock-agents/use-case-examples/insurance-claim-lifecycle-automation/agent/streamlit/

chmod u+x setup-streamlit-env.sh

### **Streamlit Web UI Deployment Summary**

#### 1. **Clone the Bedrock Sample Repository**

If you haven't already:

bash

CopyEdit

git clone https://github.com/aws-samples/amazon-bedrock-samples.git

cd amazon-bedrock-samples/agents-and-function-calling/bedrock-agents/use-case-examples/insurance-claim-lifecycle-automation/agent/streamlit/

#### 2. **Make the Setup Script Executable**

bash

CopyEdit

chmod u+x setup-streamlit-env.sh

#### 3. **Run the Setup Script**

This will create a Python virtual environment and install required packages:

bash

CopyEdit

source ./setup-streamlit-env.sh

#### 4. **Set Environment Variables**

Replace values accordingly:

bash

CopyEdit

export AWS\_REGION=ap-south-1

export BEDROCK\_AGENT\_ID=PH2UTSJLRC

export BEDROCK\_AGENT\_ALIAS=YOUR\_ALIAS\_ID # You need to create one

export KNOWLEDGE\_BASE\_ID=knowledge-base-quick-start-n5usi

export DATA\_SOURCE\_ID=knowledge-base-quick-start-yff87-data-source

export KB\_BUCKET\_NAME=insurance-app-demo-customer-resources

#### 5. **Run the Streamlit App**

bash

CopyEdit

streamlit run bedrock\_streamlit.py

It will launch a browser with a frontend where users can:

* Chat with the Agent using the alias.
* Ask questions to the Knowledge Base.
* Upload documents to S3 and sync them to the KB.

export AWS\_REGION= **us-east-1**

export BEDROCK\_AGENT\_ID= 73NSDYE4CZ

export BEDROCK\_AGENT\_ALIAS\_ID= 3JIL7VITZL

export BEDROCK\_KB\_ID= Q2BZATGMXY

export BEDROCK\_KB\_DATA\_SOURCE\_ID= MNFSQVCGJU

export BEDROCK\_KB\_S3\_BUCKET\_NAME= **insurance-app-demo-resources-172982781615-us-east-1**

python3 -m venv "$VENV\_NAME"

### Next Steps: Run the Streamlit Web UI

Now launch the frontend:

bash

CopyEdit

source agent-env/bin/activate

streamlit run bedrock\_streamlit.py

### 🌐 To Access the Web UI from Browser

If you're running this on an **EC2 instance**, open your browser and visit:

cpp

CopyEdit

http://<your-ec2-public-ip>:8501

🔐 Make sure TCP port **8501** is **open in your EC2 security group** (inbound rule) — otherwise, the page won't load.