Generative AI Unicorn Party Runbook

Wiki

Supported region
us-west-2

Maintenance/Issue Reporting
Report bugs, submit feature requests, seek support assistance by visiting the AWS GameDay Support page and selecting the Standalone Quest template. Then, assign the ticket to the content owner covering the appropriate time zone. You can also reach content owners or ask questions in **genai-gameday-interest**.

Quest	Content Developer
Mystic Code	bobayomi
UniCare	simcher
Voice of the Unicorn	jhvishes
Unicorn Party	bobayomi

GameDay Operator Advisors

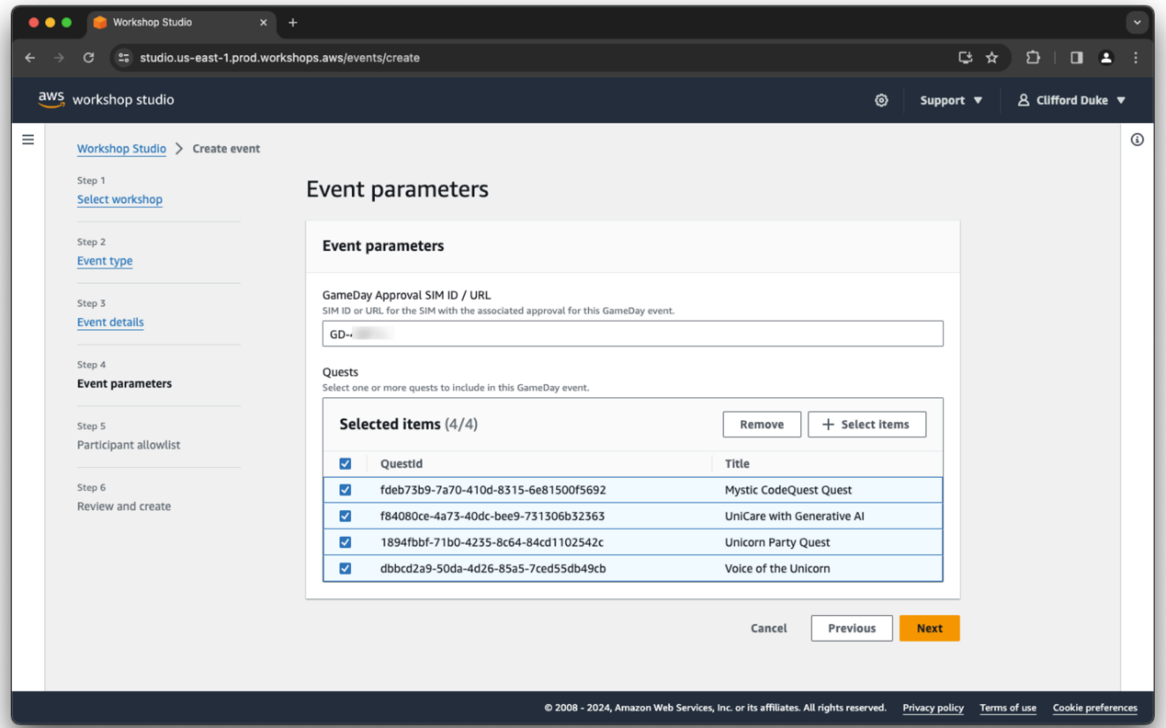
- jddehar
- clifduke

Operators Guide

Use the following guide to deploy the GameDay
Running a GameDay in Workshop Studio
GameDay Workshop: https://studio.us-east-1.prod.workshops.aws/workshops/public/0b6b2563-974f-44ba-ad2b-e6f670f5838b

IMPORTANT: Ensure you limit your GameDay to the following settings
Deployment Region: `us-west-2`
Maximum teams: 50

IMPORTANT: Ensure you enable all 4 associated quests when creating the event. Each quest is timed by QDK. Remind participants only start the quest that they are ready to play.



Event parameters

Event parameters

GameDay Approval SIM ID / URL
SIM ID or URL for the SIM with the associated approval for this GameDay event.
GD-[redacted]

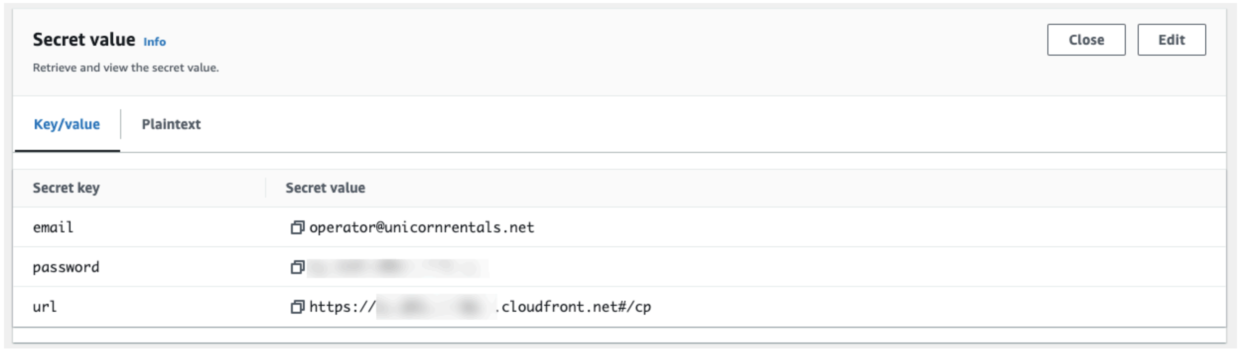
Quests
Select one or more quests to include in this GameDay event.

Selected items (4/4) — Remove — Select items

☑	QuestId	Title
☑	fdeb73b9-7a70-410d-8315-6e81500f5692	Mystic CodeQuest Quest
☑	f84080ce-4a73-40dc-bee9-731306b32363	UniCare with Generative AI
☑	1894fbbf-71b0-4235-8c64-84cd1102542c	Unicorn Party Quest
☑	dbbcd2a9-50da-4d26-85a5-7ced55db49cb	Voice of the Unicorn

Cancel
Previous
Next
© 2008 - 2024, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy policy
Terms of use
Cookie preferences

Once the central account has been provisioned, federate into it and go to Secrets Manager to find the initial control panel credentials
Secret Name: `gdQuests/ControlPanel/default_operator`



Secret value Info
Retrieve and view the secret value.

Secret key	Secret value
email	operator@unicornrentals.net
password	[redacted]
url	https://[redacted].cloudfront.net#/cp

Login to the AWS GameDay Control Panel and enable all 4 quests to begin the quest deployment

Team name approvals are now controlled under the AWS GameDay Control Panel > Teams

Upload a PDF version of your trip report HERE

Supporters Guide

This section outlines solutions for each quest. Challenge yourself before looking at the answer. Similarly, guide players how to solve problems first.

Mystic Code

Main learning objectives: Prompt Engineering

TASK 1: DEPLOY THE LAMBDA FUNCTION

Full Code

Customers can consider using Amazon Q to fill in the TODO at line 155 and 158

Here is the Full code:

```python
import json
import boto3

# Setup bedrock
bedrock_runtime = boto3.client(
    service_name="bedrock-runtime",
    region_name="us-west-2",
)

def call_claude_3_sonnet(prompt):

    prompt_config = {
        "anthropic_version": "bedrock-2023-05-31",
        "max_tokens": 4096,
        "messages": [
            {
                "role": "user",
                "content": [
                    {"type": "text", "text": prompt},
                ],
            }
        ],
    }

    body = json.dumps(prompt_config)

    modelId = "anthropic.claude-3-sonnet-20240229-v1:0"
    accept = "application/json"
    contentType = "application/json"

    response = bedrock_runtime.invoke_model(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    response_body = json.loads(response.get("body").read())

    results = response_body.get("content")[0].get("text")
    return results


# Call Titan model
def call_titan(prompt):
    prompt_config = {
        "inputText": prompt,
        "textGenerationConfig": {
            "maxTokenCount": 4096,
            "stopSequences": [],
            "temperature": 0.7,
            "topP": 1,
        },
    }

    body = json.dumps(prompt_config)

    modelId = "amazon.titan-text-lite-v1"
    accept = "application/json"
    contentType = "application/json"

    response = bedrock_runtime.invoke_model(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    response_body = json.loads(response.get("body").read())

    print(response_body)

    results = response_body.get("results")[0].get("outputText")
    return results

def claude_2_prompt_format(prompt: str) -> str:
    # Add headers to start and end of prompt
    return "\n\nHuman: " + prompt + "\n\nAssistant:"


def call_claude_2(prompt):
    prompt_config = {
        "prompt": claude_2_prompt_format(prompt),
        "max_tokens_to_sample": 4096,
        "temperature": 0.7,
        "top_k": 250,
        "top_p": 0.5,
        "stop_sequences": [],
    }

    body = json.dumps(prompt_config)

    modelId = "anthropic.claude-v2"
    accept = "application/json"
    contentType = "application/json"

    response = bedrock_runtime.invoke_model(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    response_body = json.loads(response.get("body").read())

    results = response_body.get("completion")
    return results


# Bedrock api call to stable diffusion
def generate_image(text, style):
    """
    Purpose:
        Uses Bedrock API to generate an Image
    Args/Requests:
         text: Prompt
         style: style for image
    Return:
        image: base64 string of image
    """
    body = {
        "text_prompts": [{"text": text}],
        "cfg_scale": 10,
        "seed": 0,
        "steps": 50,
        "style_preset": style,
    }

    if style == "None":
        del body["style_preset"]

    body = json.dumps(body)

    modelId = "stability.stable-diffusion-xl-v1"
    accept = "application/json"
    contentType = "application/json"

    response = bedrock_runtime.invoke_model(
        body=body, modelId=modelId, accept=accept, contentType=contentType
    )
    response_body = json.loads(response.get("body").read())

    results = response_body.get("artifacts")[0].get("base64")
    return results


def lambda_handler(event, context):

    data_string = event["body"]
    input_data = json.loads(data_string)
    model = input_data["model"]
    prompt = input_data["prompt"]

    # check for stable diffusion style
    if "style" in input_data:
        style = input_data["style"]
    else:
        style = "None"

    # You will need to write prompts to fill out the code to call the correct model bo

    if model == "titan":
        generated_text = call_titan(prompt)
        response_body = json.dumps({"text": generated_text})
    elif model == "claude":
        generated_text = call_claude_3_sonnet(prompt)
```

```
            response_body = json.dumps({"text": generated_text})
        elif model == "stable_diffusion":
            # Call stable diffusion model
            generated_image = generate_image(prompt, style)
            response_body = json.dumps({"image": generated_image})
        else:
            body = f"Error {model} is not a valid model, try another one."
            response_body = json.dumps({"error": body})

        # DO NOT CHANGE CODE BELOW
        return {"statusCode": 200, "body": response_body}
```

Customers will then need to create an API Gateway by clicking `Add trigger`

Customers will need to create a REST API with an API Key

The API Key can be found in configuration then “Triggers” then see below

Customers then submit the API_URL,API_KEY

NOTE: If you see an error like the following, CHECK model access for Claude, Amazon Titan and Stability Models (model names/IDs, mentioned in the code)

TASK 2: FRONT END DEPLOYMENT

Customers will need to launch a new Task using the BedrockApp Task

They will need to use the bedrock_app_sg

They will need to update the env vars with their API_KEY and LAMBDA_API

Once Task is up, customers will need to enter the IP address of their app. Example: http://[Public_IP]

TASK 3.1: RIDDLES OF THE ELDERS

Using the deployed app you can send Prompts to get back answers

Here is the Prompt i used to get a working answer

```
Given the following style guide and code,  identify the lines of code and comments tha

Style Guide:
1. Variable names and function names should be in snake_case (lowercase letters with u
2. Class names should be in CamelCase (no underscores; each word begins with a capital
3. There should be spaces around operators and after commas.
4. There should be no space between a function name and the following opening parenthe
```

```
5. Lines should not be longer than 79 characters.
6. Comments should start with a # followed by a single space and should clearly descri

Code:
class my_class:
    def __init__(self, var1,var2):
        self.Var1=var1
        self.Var2=var2

    def Add_Numbers(self):
        return self.Var1+self.Var2

    myObject=my_class(3,5)
    result = myObject.Add_Numbers()
    print ('The result is:', result) #This adds the numbers
```

Lines that need to be fixed

```
Here are the lines that need to be fixed:

    class my_class: - Class name should be MyClass

    def init(self, var1,var2): - Variable names should be var_1, var_2

    self.Var1=var1 - Variable names should be self.var_1 = var_1

    self.Var2=var2 - Variable names should be self.var_2 = var_2

    def Add_Numbers(self): - Function name should be add_numbers

    return self.Var1+self.Var2 - Variable names should be self.var_1 + self.var_2

    myObject=my_class(3,5) - Variable name should be my_object = MyClass(3, 5)

    result = myObject.Add_Numbers() - Function name should be my_object.add_numbers()

    print ('The result is:', result) #This adds the numbers - Comment should have spac
```

The answer is 9

TASK 3.2 WALL OF TEXT

You can use the following prompt to get the answers, for each Question, swapping in the question asked.

```
Answer the question below using the provided Context. Think carefully and make sure yo

Question:
Marquis Mainframe had numerous encounters, but how many times did he specifically ente

Context:
During the zenith of the Code Council's power, three luminaries stood at its helm: Lad
```

```
Lady Elara, as the chief architect of the Council, penned the 'Codex of Unity'. But sh

Sir Bytehart, an advocate for organic coding, introduced the Tree of Tuples, which he

Amidst the shimmering epoch of Silicon Sands, the Computation Cove was a sanctuary of

Magus Codic was a visionary who embraced the ancient and the avant-garde. He often sha

Seeress Silica, with her diaphanous digital divinations, often gazed deep into the Dat

Marquis Mainframe, a master of multi-threaded mysteries, had his own set of epics. In

Mistress Codella, the youngest of the three, had an affinity for aquatic algorithms. H

In the radiant epoch of Cypherdom, the Enchanted Empires flourished under the stewards

Lord Algon, the sage of sequences, safeguarded the Spire's most precious Parchments. E

Dame Databyte, famed for her feats in the fragmented frontiers, often oscillated betwe

Baroness Bitweave, the duchess of datastreams, was renowned for her finesse with flowi
```

Baroness Bitweave's greatest work expounded on a number of transformative techniques. How many techniques did she discuss in the 'Bitweave Book of Binds'?
9 is correct.

Lady Elara is credited with introducing a coding pattern that greatly impacted Mythica. What is the name of this pattern?
Codex of Unity is correct.

How many times did Magus Codic venture into the Matrix Marshes?
5 is correct.

Combining all her journeys, how many distinct adventures did Seeress Silica embark on?
7 is correct.

Marquis Mainframe had numerous encounters, but how many times did he specifically enter the Fortresses of Function?
4 is correct.

TASK 3.3 UNICORN PICTURE

This is subjective and we are using this asthetic predictor to give customers a score https://github.com/LAION-AI/aesthetic-predictor

Customers will enter a prompt like `A magical unicorn in an amazing forest.`

Download the file, then upload it to s3

They will need to use the Unicorn Bucket predeployed into the account.
After customers upload the file, they submit the Object URL to get a score.

Note: if a score doesn't show up, they may have to try again (cold start with loading the model)

My test Unicorn get a 7.34/10

UniCare

Main learning objectives:

- Learn how to integrate Bedrock Agents and Knowledge
- Learn how to secure Bedrock Agents execution with backend resources using action group, for example, invoking DynamoDB through lambda function in this quest
- There are 3 tasks in this quest:
  - UniSum Health: Summarizing patient transcript using Claude Instant
  - WisdomCare: Create Bedrock Knowledge Base with S3
  - Healing Horn: Integrate Bedrock Agents with Knowledge Base from Task 2
- The streamlit page for WisdomCare and HealingHorn are useful for participants to validate if their knowledge base and agents are setup correctly before providing IDs for knowledge base, agent, and agent alias, for scoring.
- [Optional] Event operator can grant extra bonus points (operator can decide on how much extra to reward for your event) if participants can bring up streamlit for WisdomCare and Healing Horn

ENVIRONMENT SETUP

Bedrock

- In Amazon Bedrock console, select **Model access**, choose **Manage model access**, select **Claude Instant** and **Titan Embeddings G1 – Text**. Choose **Request model access**

Cloud9

- Go to Cloud9 console, open the Cloud9 instance and clone the gameday repo
- Select **Manage EC2 instance** for the Cloud9 instance which takes you to the EC2 console
- Select the EC2 instance ID, select **Security** tab, open **Security groups**
- Edit inbound rule to allow port **8501** from "My IP" (Recommended) or Anywhere-IPv4 (0.0.0.0/0).
- Disable VPN on local laptop if running

Patient Table

- In the Cloud9 environment, open a terminal, run "**sh setup.sh**" to install dependencies. You can ignore any dependency warning in the terminal that are not defined in requirements.txt.
- The **setup.sh** will also create a **UnicornPatientTable** in DynamoDB. You can validate the table setup in DynamoDB console, **Explore items**, select **UnicornPatientTable**. There should be 100 items inserted into UnicornPatientTable which

you will use later in Healing Horn section.

Streamlit App

- To start the streamlit app, run "**sh run.sh**" in Cloud9 terminal
- In a browser, access streamlit app over http://[ec2_public_ip]:8501 or simply click to open the External URL in the terminal.

> Note: There is an automated 15 minutes timeout using AWS managed temporary credential. If you get an error throughout the game, "*The security token included in the request is expired*", just kill the existing streamlit process in Cloud9 terminal and rerun "**sh run.sh**" to refresh the session. Double check the port is 8501 since we only allow this port in the security group despite port can increase in streamlit for each run.

TASK 1: UNISUM HEALTH

You will summarize a medical transcript with Amazon Bedrock in this section.

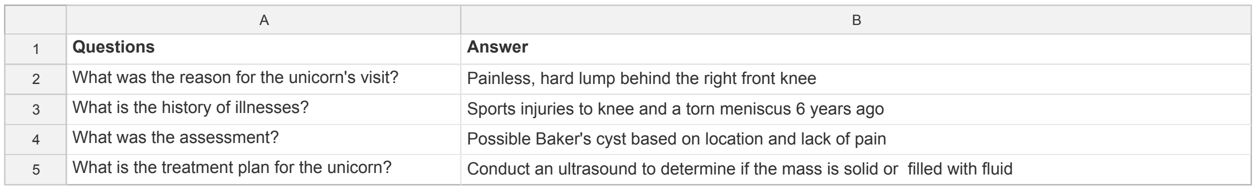
- In Cloud9, fill out sections marked with TODO in **1_UniSum_Health.py** and answer questions on the gameday dashboard to earn points.

```
llm = Bedrock(
    model_id=modelId,
    model_kwargs={
        "max_tokens_to_sample":300,
        "temperature":0.5,
        "top_k":250,
        "top_p":1,
        "stop_sequences":[]
    },
    client=bedrock_runtime,
)
```

```
map_prompt = """Write a concise medical transcript summary of the following:
"{text}"
CONCISE SUMMARY:"""
```

- Start the streamlit app, upload **/data/transcript1.pdf** to **UniSum Health**. Wait for the summarization task to complete. You should see a bulleted summary as shown in the image below. Take a note of the s3 bucket name, you will use this later in the game.

- Return to the gameday dashboard and answer multiple-choice questions to complete this task:



	A	B
1	**Questions**	**Answer**
2	What was the reason for the unicorn's visit?	Painless, hard lump behind the right front knee
3	What is the history of illnesses?	Sports injuries to knee and a torn meniscus 6 years ago
4	What was the assessment?	Possible Baker's cyst based on location and lack of pain
5	What is the treatment plan for the unicorn?	Conduct an ultrasound to determine if the mass is solid or filled with fluid

- Congratulations, you completed UniSum Health!

TASK 2: WISDOMCARE

In this section, you will create a knowledge base in Amazon Bedrock using the provided datasets.

- Go to Amazon S3 console, upload **/data/health** folder to the S3 bucket that you noted earlier
- Go to Amazon Bedrock console, select **Knowledge base** from the left navigation pane. Select **Create knowledge base**. Enter a name, e.g. *knowledge-base-health-xxxx*
- Choose **Create and use a new service role**. Next.
- Enter a data source name, e.g. *knowledge-base-health-xxxxx-data-source*
- Browse to the s3 path where you uploaded the PDFs.
- Expand **Advanced** settings, keep **Default Chunking** as Chunking strategy. Next.
- Select an embeddings model, e.g. **Titan Embeddings G1 – Text v1.2**
- Select **Quick create a new vector store – Recommended**. Next and select **Create knowledge base**. It should take 5-10 minutes to create the knowledge base. Take a note of Knowledge base ID when it's ready.
- [Important] Choose the Knowledge base data source and select **Sync**
- You can test knowledge base in the Bedrock console. Choose **Select model**, choose Claude Instant 1.2, type a question and **Run**.
- In Cloud9 environment, update the following sections in /pages/2_WisdomCare.py

```
retriever = AmazonKnowledgeBasesRetriever(
        knowledge_base_id="YOUR_KNOWLEDGE_BASE_ID",
        retrieval_config={
            "vectorSearchConfiguration": {"numberOfResults": 4}},
    )
```

```
qa = RetrievalQA.from_chain_type(
    llm=llm,
    retriever=retriever,
    return_source_documents=True
    )
```

- Save the file and rerun the streamlit app. You should be able to interact with the WisdomCare Q&A bot.

- Return to the gameday dashboard. Enter Knowledge Base ID for evaluation.
    - Note: *Double check no space when copy and paste the knowledge base ID to gameday dashboard. Occassionally, GameDay dashboard might experience latency causing knowledge base ID not being recognized the first time. If this happens, refreshing the gameday dashboard should resolve the problem.*
- Operator can grant extra 5000 points for bringing up the streamlit WisdomCare page
- Congratulations, you completed WisdomCare!

TASK 3: HEALING HORN

In this section, you will create an agent with Amazon Bedrock to retrieve patient records and answer health questions based on WisdomCare knowledge base. Lambda code (healinghorn_lambda.py) and OpenAPI schema yaml (healinghorn_openapi.yaml) are available under /lambda

Configure Agent

- Upload the **healinghorn_openapi.yml** to the s3 bucket you noted earlier from UniSum
- Go to Amazon Bedrock console, then **Agents**, choose **Create Agent**
- Enter **Agent name**, choose **Create**
- Choose **Anthropic Claude Instant V1**. Provide instructions for the agent. Example: *You are an AI agent that retrieves patient records and provide health information with associated knowledge base, respectfully say I don't know if requested information is not available.*
- Choose **Create and use a new service role**. Select **Save**.
- In the **Action groups** section:
    - Select **Add**.
    - Select **Define with API schemas** and select **Quick create a new Lambda function - recommended**.
    - Choose **Select an existing API schema**.
    - Browse to the s3 path of **healinghorn_openapi.yaml**, it should be something like *s3://sagemaker-us-west-2-{Account_ID}/healinghorn_openapi.yml* . NOTE: This is the same bucket you noted earlier from UniSum.
    - Choose **Create**
- Go to Lambda console, you should see a new Lambda function created with the corresponding agent name.
    - Replace the lambda code with sample code from **healinghorn_lambda.py**
    - Grant **DynamoDBFullAccess** to Lambda function execution role. You can find the lambda role name in **Permissions** under **Configuration** tab
    - File → Save → Deploy.
- Back in the **Agent Builder**, select **Add** under **Knowledge bases** section. Choose the knowledge base you created in WisdomCare. Provide instructions for the agent, e.g. *"You are a helpful agent that provides health information using the associated knowledge base, respectfully say I don't know if the requested info is not available"*.
- Choose **Save** and **Prepare** to test the Agent configuration with a sample query such as "patient_id 21".

- You can guide players to explore **Show trace** for pre-processing and orchestration rationale.

- Create an alias for the agent. Choose **Create Alias**. Take a note of the agent ID and agent alias ID
- Update **3_Healing_Horn.py** in Cloud9 with your agent ID and agent alias ID

```
agentId='YOUR_AGENT_ID'
agentAliasId='YOUR_AGENT_ALIAS'
```

- Execute **run.sh** to test the Healing Horn page. You should see the following result if the Bedrock agent is setup correctly to retrieve patient records from UnicornPatientTable in DynamoDB and lookup health information from WisdomCare knowledge base for other prompts.

- Provide the **Agent ID** and **Agent Alias ID** in the gameday dashboard for scoring
- OPTIONAL: Operator can grant extra 5000 points for bringing up the streamlit HealingHorn page
- Congratulations, you completed Healing Horn!

Voice of the Unicorn

Main learning objectives:

- This quests involves transcribing a contact center interaction and answering questions based on the content and metadata. For that participants are expected to use a LLM (in Bedrock).
- There are 3 tasks in this quest. Task 2 and 3 load once task 1 is complete.

TASK 1: CONVERT VOICE TO TEXT

1. Open AWS management console and Go to Amazon Transcribe Call Analytics > Post-call Analytics.
2. Create Job >
    a. Give the job a name (e.g. pca-1), Leave Language settings, Model Type as defaults (US English, General Model)
    b. For input data select the file whose location is mentioned in the UI banner above this → "Welcome New Hires"
    d. For Agent audio channel select **Channel 0**
    e. For Output data leave it as Service-managed S3 bucket
    f. For IAM Role, Select Use an existing IAM role, and select the available one from the dropdown. The name of the role should be similar to - StackSet-gdQuests-XXXXXXX-TranscribeIAMRole-XXXXX
    g. Hit next, leave rest settings as is and hit Create Job. The Post call Analytics Job would take 5 minutes to complete.
3. Once the job is Complete and Successful, hit ready in the Quest UI to finish this task

TASK 2: INSIGHTS INTO THE CALL

1. Open AWS management and go to the successful Post call analytics Job and open it. You will be able to see what was the beautified transcription output under transcription preview. Give that a read, however for this task, hit Download transcript to download the JSON output of the Job
2. Open that JSON file in a tab of your browser
3. Now to go to Amazon Bedrock's console
    a. Go to Bedrock's Model Access page to check whether access been granted for various models. If not, select the models you want to use (Suggest Using Anthropic Claude for this Quest) and Manage Model Access
    b. For Claude models you have to enter some extra details, but you will get access instantly. You can use AWS GameDay as your use-case.
4. Once Model Access has been Granted, Go to Chat under Playgrounds and select Anthropic from model category and then Claude V2 for model (Claude performs best for this use-case, and also has the biggest context window [100K token] which is useful in this scenario as we're passing a whole conversation log JSON)
5. In the Chat box, paste the following prompt. Be sure to paste the Whole JSON output between the <transcript> </transcript> XML tags. This is the reason Claude V2 and higher is advised as it has a 100k Token context window to allow for such use-cases (The Questions open up sequentially, e.g. Q2 opens up after Q1. Therefore, participants will have to run this prompt multiple times with 1 question each. The prompt below is for supporters to get through the task quickly. Following is an example for getting the answer to 1st question

> Human: Answer all the questions below, based on the contents of the JSON in <transcript></transcript>. If you cannot answer the question, reply with 'n/a'.
> <question>
> What was the overall sentiment score for the CUSTOMER?
> </question>
>
> <transcript>
> [Paste transcribe call analytics output JSON here]
> </transcript>

1. The Model will return a response similar to this format. (PLEASE NOTE - Actual Answers would be different then below because Transcribe's underlying model will change over time)

2. Mark the answers as the above in the respective questions and hit Submit. After Submitting all 4 answers, the task will finish

TASK 3: AUTOMATE!

In this quest, we need to fix a lambda function that interacts with Bedrock to get back answers for certain questions and stores in a Database. At a high-level it does the following:

a. It generates plain text transcript from the Transcribe output,
b. Passes that along with instructions, on what information needs to be extracted, in a form of a Prompt to Bedrock (ClaudeV2 Model)
c. Uses the output from Bedrock to update a DynamoDB table, with partition key 'call_analytics_job_name' (value of which will be present in event['JobName'])

For this let's Open the Lambda function.

1. Go to Lambda Console, and Search for BedrockOrchestrator. Open the Function.
2. Essentially, we need to replace the values of variables which have been assigned None values, under update_ddb() function. Replace the update_ddb() function by pasting the following snippet:

```python
def update_DDB(bedrock_op, job_name, table_name):
    #Todo: extract values from bedrock_op corresponding to each variable
    bedrock_op_summary = bedrock_op['Summary']
    bedrock_op_topic = bedrock_op['Topic']
    bedrock_op_product = bedrock_op['Product']
    bedrock_op_resolved = bedrock_op['Resolved']
    bedrock_op_callback = bedrock_op['Callback']
    bedrock_op_politeness = bedrock_op['Politeness']
    bedrock_op_actions = bedrock_op['Actions']

    #Todo: Init boto3 dynamoDB client in ddb variable
    ddb = boto3.client('dynamodb')
    response = ddb.put_item(
        TableName=table_name,
        Item={
            'call_analytics_job_name':{'S':job_name},
            'summary': {'S':bedrock_op_summary},
            'topic': {'S':bedrock_op_topic},
            'product': {'S':bedrock_op_product},
            'resolved': {'S':bedrock_op_resolved},
            'callback': {'S':bedrock_op_callback},
            'politeness': {'S':bedrock_op_politeness},
            'actions': {'S':bedrock_op_actions}
            }
        )
    return response
```

1. Deploy the changes by hitting Deploy
2. Once deployed, go back to the Quest UI, and hit Ready. This will kick off a Step Function execution by dropping a new conversation file in the designated S3 bucket.
3. Go to the Step Functions Console and Explore the State machine with the name “TranscriptionOrchestrator-XXXXXX”, and open its latest execution.
4. You’d notice that the execution is in process. The Entire execution takes approx 5 minutes. Once done all steps will be green
    a. The successful lambda invoke step meant that the function passed the transcript and related questions as a prompt to a ClaudeV2 model and got stored the response in a formatted way in DynamoDB. Feel Free to explore this item, by Going to DynamoDB Console and searching for “TeamDDBTabble”. Explore its items and search for call_analytics_job_name as the name of the step functions state machine **execution (not State machine name)**, from the previous step. In the screenshot below, the Step functions execution name was “75f6fe68-5f83-cd46-12c8-167998efa844_707f16b0-67b7-5594-85f6-ca5f1ad0eef7”
5. Once this execution is complete, go back to the Quest UI and hit EVALUATE. This would finish the task and wrap up this Quest.

Unicorn Party

Main learning objectives: Explore creative ways to build generative AI experience using PartyRock

TASK 1: PARTYROCK APP

Create The Party Style Widget

- Placeholder: Enter a type of party for the unicorns
- Example: A party at sea on a magical boat

Party Scene Widget:

Describe a lavish party for unicorns based on the following style: `@party style`

Painter

Given the following scene, create a prompt for an image generation model that explains in detail how to paint a picture that depicts the party.

Scene: `@party scene`

Finished App

TASK 2: REMIX APP

- Fix the Summary to use `Chat` instead of FIXME

- Add text like
    - Based on the following summary, create a prompt for an image generation model, that captures the essence of the current scene.
- Update the Story Image to just use `@nextimage`

Example of Final Output

Customers must submit their URLs with `/published` (this confirms the URL is public)