EC2 Operational Activities Management via Chatbot

Overview

In the current dynamic environment, effective management of cloud assets is vital for organizations. This guide will show you how to set up a serverless chatbot utilizing OpenAI's ChatGPT (API), AWS Lambda, and a Chrome Extension to oversee your Amazon EC2 instances. By adhering to these straightforward instructions, you can develop a robust chatbot capable of executing actions like initiating, halting, and monitoring the condition of your instances, in addition to providing basic information about EC2 and AWS resources.

Step 1: Set up your OpenAI API key
To begin, you'll need to sign up for an OpenAI API key if you haven't already. Once you've obtained your key, store it securely as you'll need it later to interact with the ChatGPT API.

Step 2: Create an AWS Lambda function
Next, create an AWS Lambda function that will serve as the backend for chatbot. The Lambda function is written in Python and includes the necessary dependencies of AWS SDK (Boto3) and OpenAi for managing EC2 instances.

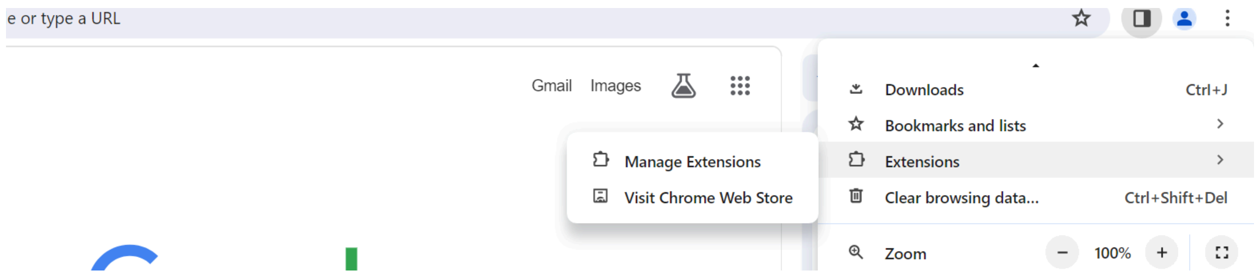
Step 3: Create an API Gateway
To link the Lambda function to the front end, first create a REST API Gateway. This will allow the Chrome Extension to interact with the Lambda function. Make sure to configure the API Gateway to use Lambda Integration and secure it using appropriate authentication methods.

Step 4: Develop the Chrome Extension
Create the Chrome Extension (Front End) that will act as the user interface to chatbot. Design a simple popup that includes an input field for user messages and an area to display the chat history. When a user submits a message, the extension should send a POST request to the API Gateway, which will trigger the Lambda function.
Make a new folder '**extension**'. Inside this folder, create these four files: HTML, CSS, JavaScript, and manifest and add an icon.png.

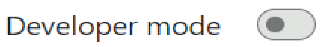
Folder structure

```
├── extension/
│   ├── index.html
│   ├── script.js
│   ├── styles.css
│   └── icon.png
```

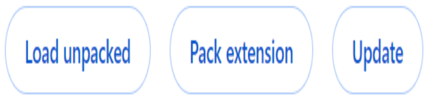
How to configure Chrome Extensions



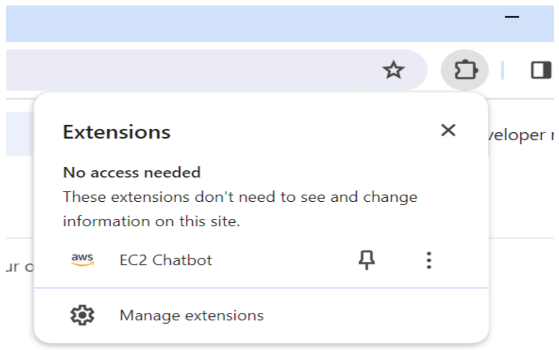
Open a browser, click the three dots on the right side and click the Extension and choose the Manage extension.

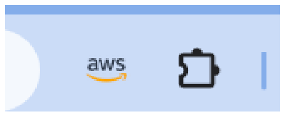


On the right side you have Developer mode you enable the mode to on. After enabling you get the below buttons



Now click the Load unpacked and choose the extension folder created in step4. After choosing the **extension** folder then on the right side you get Extension icon displayed click that icon you get

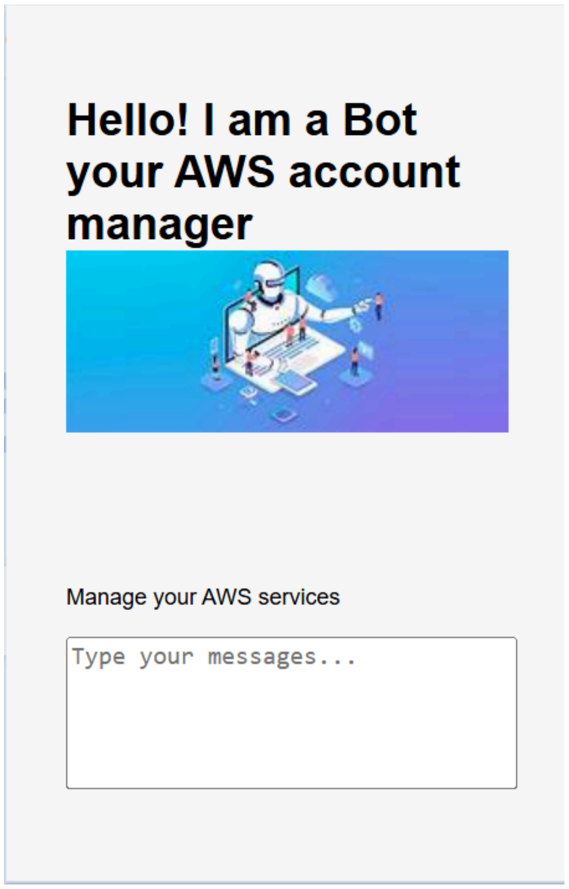

Extensions
No access needed
These extensions don't need to see and change information on this site.
EC2 Chatbot
Manage extensions



Pin the EC2 Chatbot it will appear next to the address bar. Then click the aws icon and the Chatbot window pops up.

Step 5: Test and deploy

Thoroughly test your chatbot by deploying the Lambda function, API Gateway, and Chrome Extension to ensure that everything is working correctly. Make any necessary adjustments and optimizations to improve your chatbot's user experience and functionality.


Hello! I am a Bot your AWS account manager

Manage your AWS services

Type your messages...

Usage

To execute a command, type the relevant phrase into the chatbot interface. The chatbot will process the input, send it to the Lambda function via the REST API, and execute the corresponding operation in AWS.

Operational Commands

<u>EC2</u>

List EC2 Instances:

- Input: "**list ec2 instances”**
- Action: Lists all EC2 instances with their ID, state, and type.

Get EC2 Instances Count:

- Input: "**get ec2 count**"
- Action: Returns the total count of EC2 instances.

List On-Demand EC2 Instances:

- Input: "***list on demand instances***"
- Action: Lists on-demand EC2 instances with their ID, owner, and type.

Get CPU Utilization:

- Input: "**get cpu utilization <IP_ADDRESS>**"
- Action: Returns the CPU utilization for the instance associated with the specified IP address.

Start an EC2 Instance:

- Input: "**start instance <INSTANCE_ID>**"
- Action: Starts the specified EC2 instance.

Stop an EC2 Instance:

- Input: "**stop instance <INSTANCE_ID>**"
- Action: Stops the specified EC2 instance.

Get Instance type and count:

- Input: "**get instance type count**"
- Action: List all the instance types with count .

Get machines list owned by the owner:

- Input: "**get list of machines owned by <owner name>**"
- Action: List all the machines tagged under the owner name.

Get cost optimizer alerts:

- Input: "**get cost alerts**"
- Action: List out the alerts flagged by cost optimizer

Get underutilized instances :

- Input: "**get underutilized machines**"
- Action: List out all underutilized machines.

Add tag value to all instances:

- Input: "**add tag <keyname:keyvalue>**"
- Action:The above command adds a new tag value to all instances in the account.

Remove tag value from all instances :

- Input: "**remove tag <keyname>**"
- Action: The above command will remove the tag from all instances in the account.

List Spot EC2 instances

- Input: "**list spot instances**"
- Action: The above command will list all spot instances.

Count of running instances

- Input: "**get running instance count**"
- Action: The above command will give the count of running instances.

Count of stopped instances

- Input: "**get stopped instance count**"
- Action: The above command will give the count of stopped instances.

Count of snapshots

- Input: "**get snapshot count**"

- Action: The above command will give the count of the snapshot.

Count of AMIs

- Input: "**get ami count**"
- Action: The above command will give the count of AMIs

List of instances created in the past given hours

- Input: "**find the instances created in the last <number> hours**"
- Action: The above command will list the instances created in the last number of hours.

List of instances created in the past given week

- Input: "**find the instances created in the last <number> week**"
- Action: The above command will list the instances created in the last number of weeks.

List of instances created in the past given months

- Input: "**find the instances created in the last <number> months**"
- Action: The above command will list the instances created in the last number of months.

List the instances since stopped for the given input months

- Input: "**get instances stopped for last <number> months**"
- Action: The above command will list the instances stopped in the last input months.

Add security group to instance

- Input: "**add security group <sg-name> to <ip-address>**"
- Action: The above command will add the given security group to the given instance.

Remove security group from the instance

- Input: "**remove security group <sg-name> from <ip-address>**"
- Action: The above command will remove the given security group from the given instance.

S3

List all s3 buckets

- Input: "**list s3 buckets**"
- Action: The above command will list all the s3 buckets for that account.

Get s3 bucket size

- Input: "**get s3 bucket <bucketname> size**"
- Action: The above command will get the size of the given bucket

Create s3 bucket

- Input: "**create s3 bucket <bucketname>**"
- Action: The above command will create the bucket

Delete s3 bucket

- Input: "**drop s3 bucket <bucketname>**"

- Action: The above command will delete the bucket

List files in s3 bucket

- Input: “**list files in s3 bucket <bucketname>**”
- Action: The above command will list all files in the bucket

EBS

Count of EBS volumes

- Input: "**get ebs volume count**"
- Action: The above command will give the count of EBS Volumes.

Find the Maximum volume size

- Input: “**find the maximum size of volume**”
- Action: The above command will list the maximum volume size

Find the Minimum volume size

- Input: “**find the minimum size of volume**”
- Action: The above command will list the minimum volume size

Find the Total volume size

- Input: “**find the total volume size**”
- Action: The above command will find the total volume size.

Find the volume type wise count

- Input: “**find volume type count**”
- Action: The above command will find the volume type wise count.

Find the volume size wise count

- Input: “**find volume size count**”
- Action: The above command will find the volume size wise count.

Find the unattached volumes

- Input: “**find unattached volumes**”
- Action: The above command will find the volumes which are not attached to ec2 instances.

Billing

Get AWS Billing specific to month

- Input: "**get billing for the month <month name> <year>**"
- Action: Retrieves AWS billing information for the specified month and year.

Get AWS Billing specific to a year

- Input: "**get billing for the year <year>**"

- Action: Retrieves AWS billing information for the specified month and year.

Get instance type wise billing for specific to month and year

- Input: "**get instancetype billing cost <month number> <year>**"
- Action: Retrieves instance type wise billing cost for the specified month and year.

DynamoDB

Get list of all dynamodb tables

- Input: **"list dynamodb tables**"

Get table metadata

- Input: **"get table metadata <table name>"**

Extension and Customization

- To add new functionalities, extend the manage_ec2_instances function in lambda with additional conditions and corresponding actions.
- For improved natural language processing, adjust the GPT-3 prompt or experiment with different OpenAI models.

Maintenance

- Keep the AWS SDK boto3 and OpenAI Client libraries up to date to ensure compatibility and security.
- Monitor AWS service quotas to ensure the operations do not exceed limits.
- Regularly review IAM permissions for principle of least privilege.

Conclusion

By using these steps, you can make a chatbot without a server using OpenAI's chatGPT (API), and AWS Lambda to use as Chrome Extension. This helps manage your Amazon EC2 instances easily. This chatbot will streamline your AWS management tasks and allow you to interact with your infrastructure in a more natural and intuitive way. As a result,you'll save time and effort while maintaining better control over your resources.