

MINI PROJECT (SEM V)

BANKING CHATBOT

Using **IBM Cloud** & **AWS**

NAME: TANYA CHETNA VAISH

ROLL NO: 2013553

Project Link: [Trinket Bank](#)

Introduction:

A chatbot (called bot for short) is a computer program designed to mimic conversation with human users on the internet, according to Oxford Dictionaries. Using robotics and **Artificial Intelligence** (AI), a chatbot can assist customers without the need for a customer service agent on the other end. Customers started to see chatbots in banking in the early 2000s through text messaging. These bots could do simple tasks like show an account balance when given a specific command. Now, AI vendors like **Abe** are making chatbot interactions about banking as natural as chatting with a friend across platforms like Facebook Messenger, Slack and in banks' mobile apps.

Taking an inspiration from revolutionary banking chatbots like Bank of America's Erica and Capital-One, I tried to create a banking chatbot in this project.

Objective:

Create a Banking Chatbot using [IBM Watson Assistant](#) and embed it on a website hosted on [AWS](#) Cloud.

Cloud Service Providers Used:

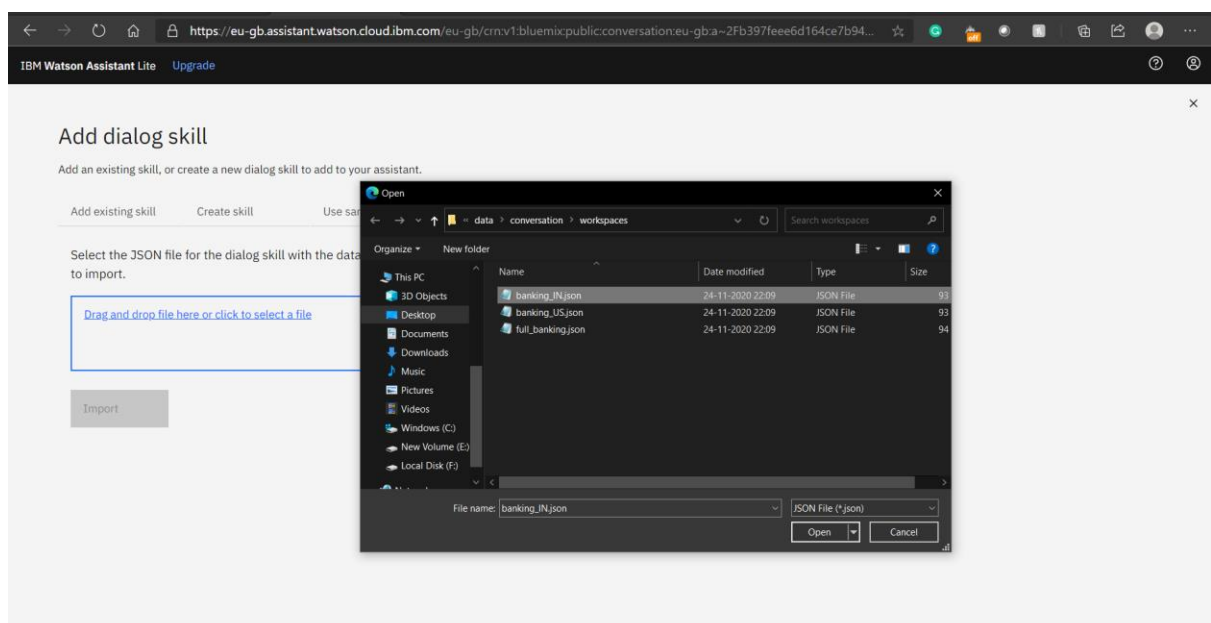
IBM Cloud and AWS

Tools Used:

[Watson Assistant](#) from IBM Cloud, [Bucket Storage](#) Service from IBM Cloud & [EC2](#) from AWS.

STEPS & CODE USED:

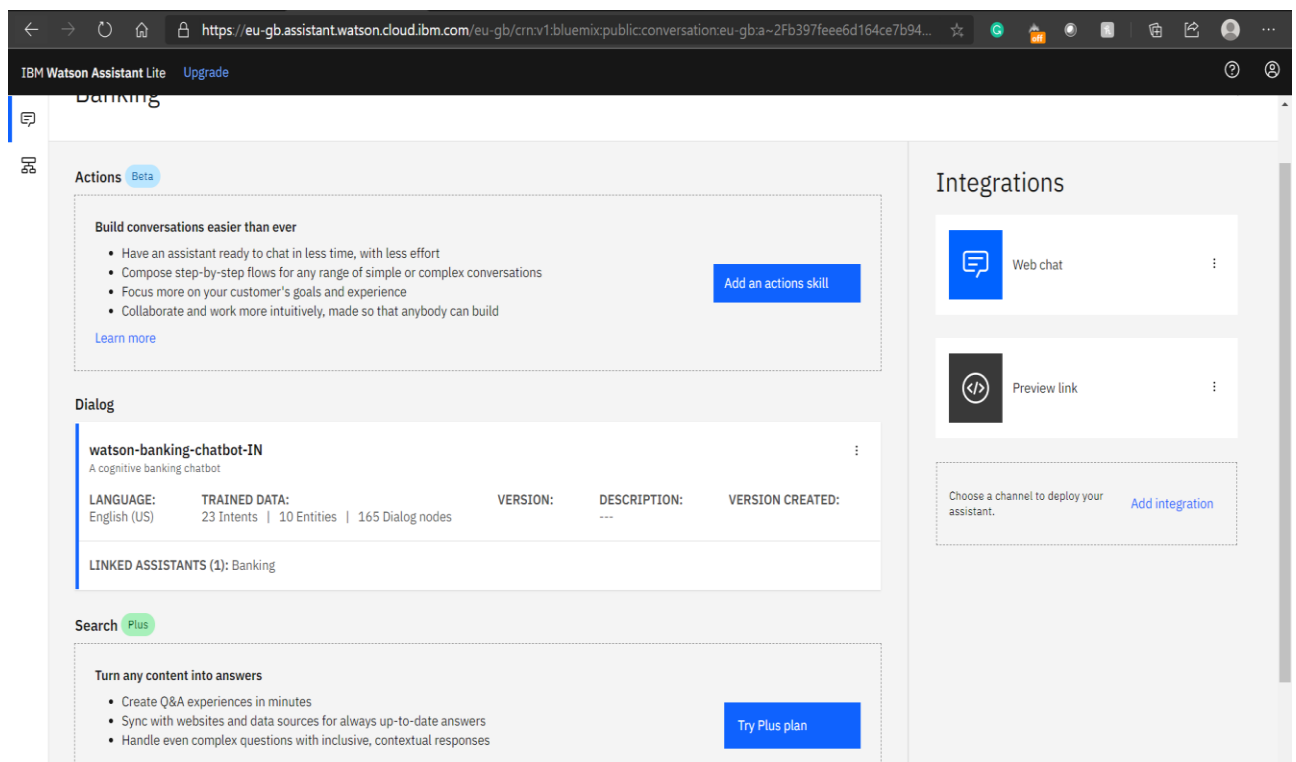
1. Login into IBM Cloud: <https://cloud.ibm.com>
2. Click the CatLog tab
3. Search for the Watson Assistant service and click that tile under the AI heading.
4. Fill out the necessary information and click Create
5. Click Launch Watson Assistant. If you're prompted to log in, provide your IBM Cloud credentials.
6. An assistant named My first assistant is created for you automatically. An assistant is a cognitive bot to which you add skills that enable it to interact with your customers in useful ways.
7. A dialog skill named My first skill is added to the assistant for you automatically. A dialog skill is a container for the artifacts that define the flow of a conversation that your assistant can have with your customers.
8. Go to the Skills tab.
9. Click Create skill
10. Select the Dialog Skill option and then click Next
11. Click Create Skill
12. Select the Dialog Skill option and then click Next.
13. Click the Import Skill tab.
14. Click Choose JSON file and copy the banking_IN.json file which contains the content for banking in India. Click Import.

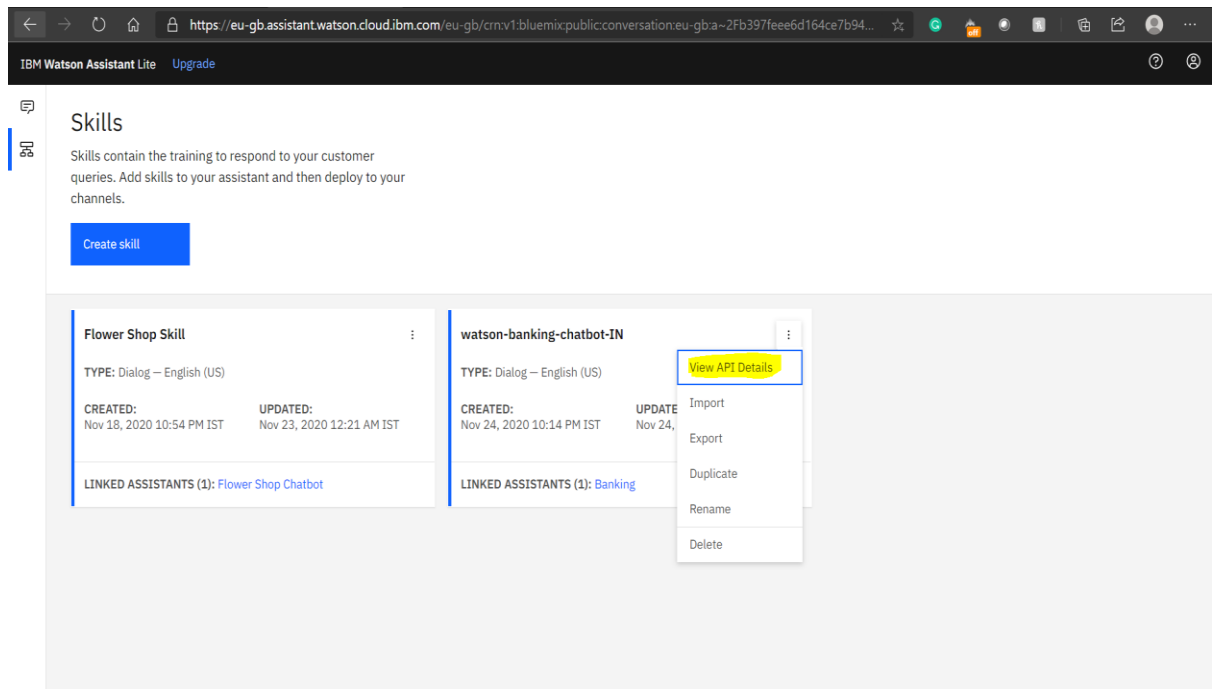


banking_IN.json file:

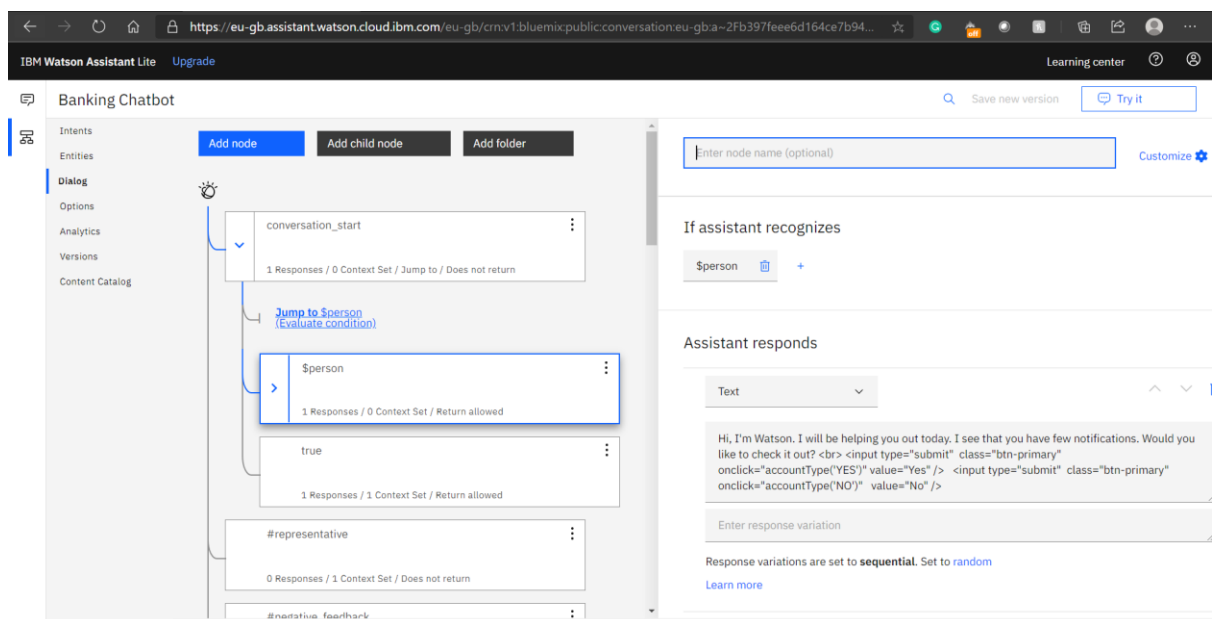
This file contains the information about intents and entities in json format. This information is what is going to make up the chatbot. Only by importing this JSON file, one can create a whole chatbot and do necessary customizations if any. This can be done manually- the creation of entities and intents. But the JSON not only increases automation but also provides a ready-to-use infrastructure and is faster. *I have attached the GitHub repository link at the end of this doc containing the necessary files used in this project.*

15. By default, the application will import and use the skill named **watson-banking-chatbot-IN**, but we can change this by clicking on three dots at the top right of the dialog tab and clicking on rename option. Here, rename it to **Banking Chatbot**.





To view the Assistant dialog, click on the skill and choose the Dialog tab. Here's a snippet of the dialog:



16) Make the required customisations as per your requirement or add one or more intents and entities.

For adding an intent:

An intent is specified using #.

- Click on Intents tab, then click on Create intent +
- Type the intent name and click Add
- Give a few examples based on the intent
For example: #Greetings intent had 5 examples: Hello, Hi, Howdy, Hi how are you and how are you.
- Click Add example after adding every example.
- Now you can use these intents anywhere as required in dialogs while creating the chatbot

For adding entities:

An entity is specified using @.

- Click on Entities tab, then click on Create entity +
- Give the name and click Create entity
- Give Value and Synonyms based on the entity provided
For example, if the entity is @location, value will be Mumbai and the synonyms can be Andheri, Juhu and so on.
- Click Add Value.

For this chatbot, we have:

Banking Chatbot

⋮

A cognitive banking chatbot

LANGUAGE:
English (US)

TRAINED DATA:
24 Intents | 11 Entities | 165 Dialog nodes

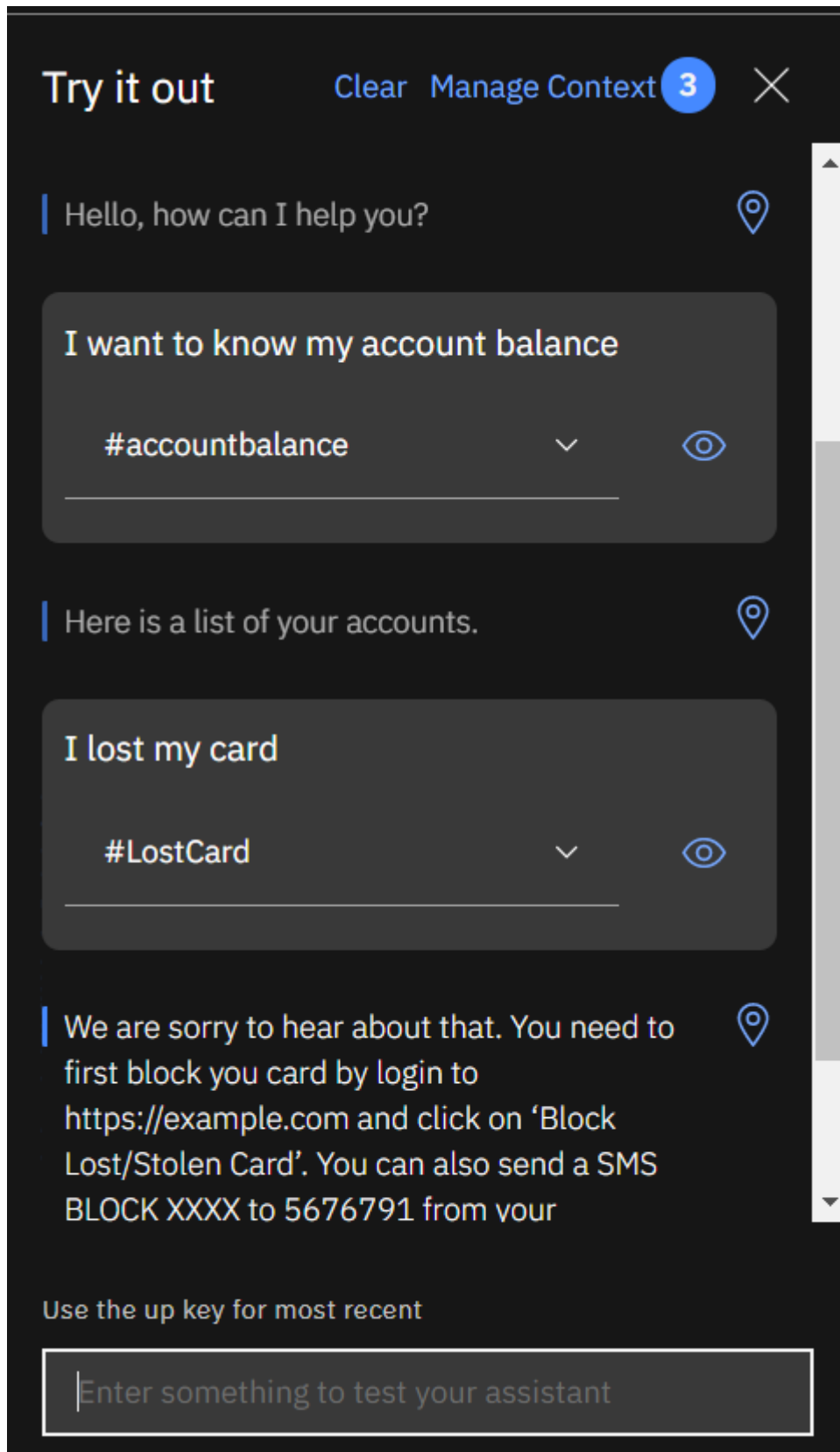
VERSION:

DESCRIPTION:

VERSION CREATED:

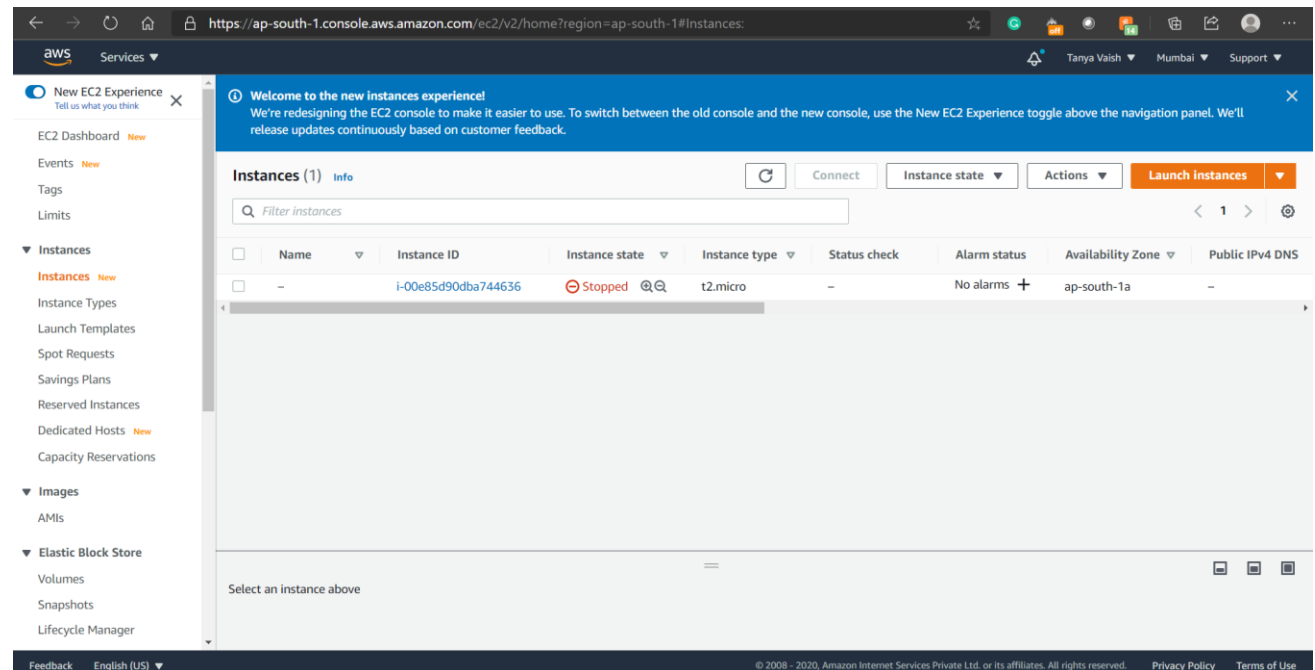
LINKED ASSISTANTS (1): Banking

17) Testing the Chatbot:

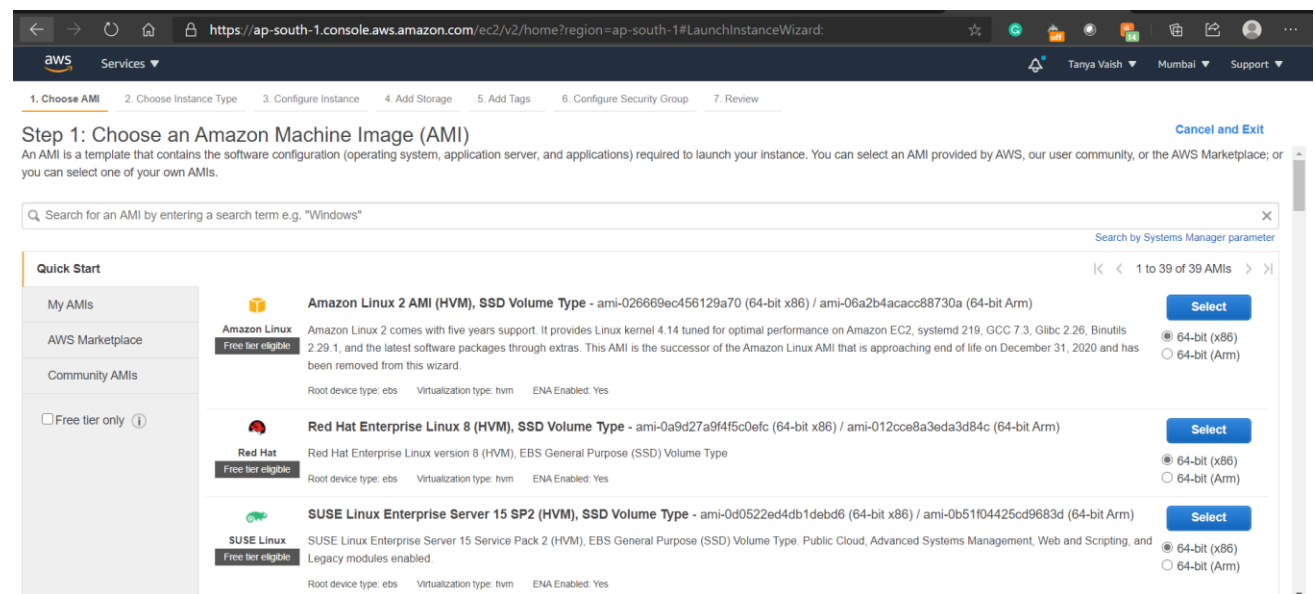


18) Now we have to integrate this chatbot to make it accessible to users. Log in AWS account (here using AWS Free Tier Account)
[Amazon Web Services \(AWS\) - Cloud Computing Services](#)

19) Go to EC2 Service



20) Click on Launch Instance. Select AMI and other necessary specs needed for instance:



Used t2. micro instance as part of free tier account.

← → ↻ 🏠 🔒 https://ap-south-1.console.aws.amazon.com/ec2/v2/home?region=ap-south-1#LaunchInstanceWizard: Services ▼ Tanya Vaish ▼ Mumbai ▼ Support ▼

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 2: Choose an Instance Type

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instances are virtual servers that can run applications. They have varying combinations of CPU, memory, storage, and networking capacity, and give you the flexibility to choose the appropriate mix of resources for your applications. [Learn more](#) about instance types and how they can meet your computing needs.

Filter by: All instance families ▼ Current generation ▼ Show/Hide Columns

Currently selected: t2.micro (- ECUs, 1 vCPUs, 2.5 GHz, -, 1 GiB memory, EBS only)

	Family	Type	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance	IPv6 Support
<input type="checkbox"/>	t2	t2.nano	1	0.5	EBS only	-	Low to Moderate	Yes
<input checked="" type="checkbox"/>	t2	t2.micro <small>Free tier eligible</small>	1	1	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.small	1	2	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.medium	2	4	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.large	2	8	EBS only	-	Low to Moderate	Yes
<input type="checkbox"/>	t2	t2.xlarge	4	16	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t2	t2.2xlarge	8	32	EBS only	-	Moderate	Yes
<input type="checkbox"/>	t3	t3.nano	2	0.5	EBS only	Yes	Up to 5 Gigabit	Yes

Cancel Previous Review and Launch Next: Configure Instance Details

Here I selected 2 security groups to allow networking, so that we can access the webpage hosted on this instance-one is SSH using which we can connect to this instance using our bare systems, and other is HTTP to allow this instance to be over Internet from anywhere. Specify the source as Anywhere (0.0.0.0/0, ::/0)

aws Services ▼ Tanya Vaish ▼ Mumbai ▼ Support ▼

1. Choose AMI 2. Choose Instance Type 3. Configure Instance 4. Add Storage 5. Add Tags 6. Configure Security Group 7. Review

Step 6: Configure Security Group

A security group is a set of firewall rules that control the traffic for your instance. On this page, you can add rules to allow specific traffic to reach your instance. For example, if you want to set up a web server and allow Internet traffic to reach your instance, add rules that allow unrestricted access to the HTTP and HTTPS ports. You can create a new security group or select from an existing one below. [Learn more](#) about Amazon EC2 security groups.

Assign a security group: ☒ Create a new security group ☐ Select an existing security group

Security group name:

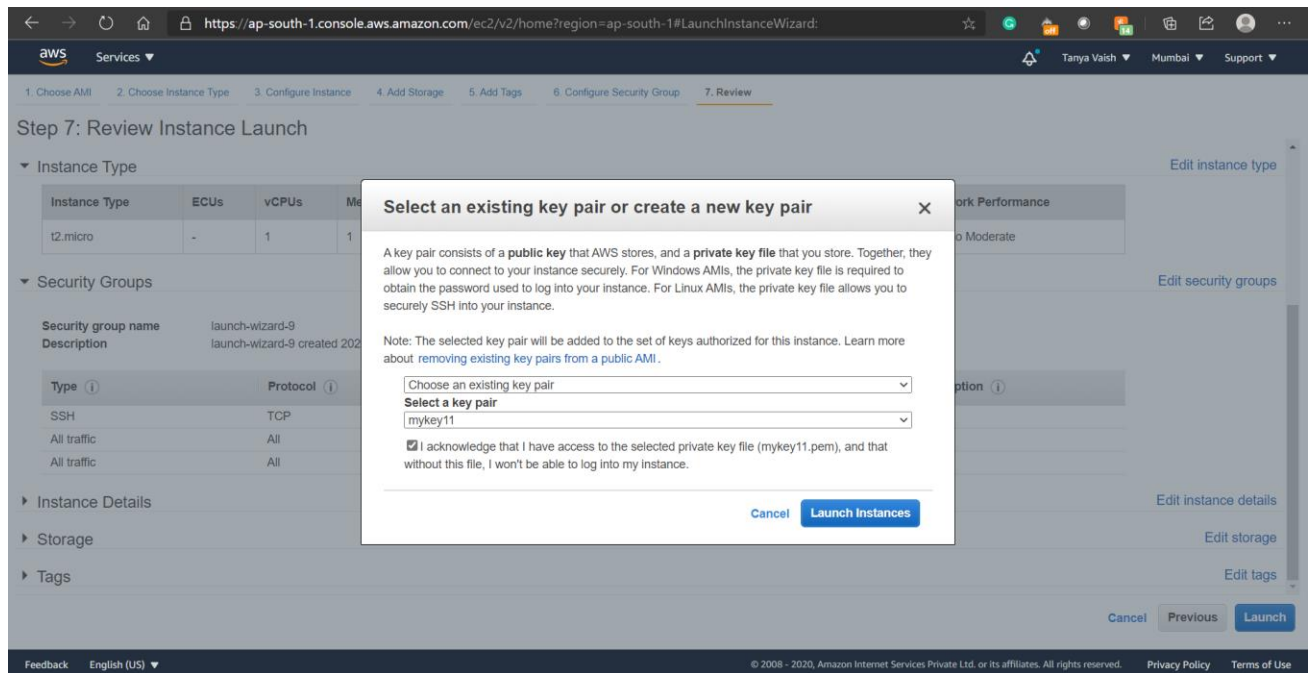
Description:

Type	Protocol	Port Range	Source	Description
SSH	TCP	22	Custom 0.0.0.0/0	e.g. SSH for Admin Desktop
HTTP	TCP	80	Anywhere 0.0.0.0/0, ::/0	e.g. SSH for Admin Desktop

Add Rule

Warning
Rules with source of 0.0.0.0/0 allow all IP addresses to access your instance. We recommend setting security group rules to allow access from known IP addresses only.

Cancel Previous Review and Launch



Make sure to download the key to access the instance locally.
Click on Launch Instance and after a while the instance will be in running state.

21) Now SSH connect with this instance using key on terminal. Make sure you have AWS CLI setup already to access AWS using CLI/terminal.

```
C:\Users\TANYA\Downloads>ssh -i "mykey11.pem" ec2-user@ec2-65-0-12-214.ap-south-1.compute.amazonaws.com
The authenticity of host 'ec2-65-0-12-214.ap-south-1.compute.amazonaws.com (65.0.12.214)' can't be established.
ECDSA key fingerprint is SHA256:p0PK5n/Z+KNsHPQzhPsesMOyIayKQ9gPgyUw/VP3Dgo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'ec2-65-0-12-214.ap-south-1.compute.amazonaws.com,65.0.12.214' (ECDSA) to the list of known hosts.

 _ | _ | _ )
 _ | ( _ | /   Amazon Linux 2 AMI
 _ | \ _ | _ |
 _ | ( _ | /   Amazon Linux 2 AMI
 _ | \ _ | _ |

https://aws.amazon.com/amazon-linux-2/
No packages needed for security; 3 packages available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-40-226 ~]$
```

21) After logging into the instance, run the following commands:

```
sudo su – root [To login as root user]
yum update -y [To update yum]
yum install httpd -y [To setup Apache webserver]
service httpd start
chkconfig httpd on
```

```
[ec2-user@ip-172-31-43-25 ~]$ yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
You need to be root to perform this command.
[ec2-user@ip-172-31-43-25 ~]$ sudo su - root
[root@ip-172-31-43-25 ~]# yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
--> Package ec2-instance-connect.noarch 0:1.1-12.amzn2 will be updated
--> Package ec2-instance-connect.noarch 0:1.1-13.amzn2 will be an update
--> Package iptables.x86_64 0:1.8.4-10.amzn2.1.1 will be updated
--> Package iptables.x86_64 0:1.8.4-10.amzn2.1.2 will be an update
--> Package iptables-libs.x86_64 0:1.8.4-10.amzn2.1.1 will be updated
--> Package iptables-libs.x86_64 0:1.8.4-10.amzn2.1.2 will be an update
--> Finished Dependency Resolution
```

Dependencies Resolved

Package	Arch	Version	Repository	Size
Updating:				
ec2-instance-connect	noarch	1.1-13.amzn2	amzn2-core	22 k
iptables	x86_64	1.8.4-10.amzn2.1.2	amzn2-core	476 k
iptables-libs	x86_64	1.8.4-10.amzn2.1.2	amzn2-core	93 k

22) Now after successful installation, head into /var/www/html directory to create the HTML page here. Use the pre-installed nano editor:

```
[root@ip-172-31-39-123 ~]# cd /var/www/html
[root@ip-172-31-39-123 html]# ls
[root@ip-172-31-39-123 html]# nano index.html
```

Here, begins the necessary steps of IBM Watson Assistant and EC2 AWS integration

Here's the HTML code used:

```
root@ip-172-31-36-6:/var/www/html
GNU nano 2.9.8 index.html

<!DOCTYPE html>
<html>
<head>
<title>Trinket Bank</title>
</head>
<style>
body {
background-image: url('https://bank-donotdelete-pr-oms5kwixzqptok.s3.us.cloud-object-storage.appdomain.cloud/Pin-on-Super-gif.gif')
}
</style>
<body>
<center>
<h1 style="font-size:60px;color:white">Welcome to Trinket Bank</h1>

<p style="font-size:40px;color:lightblue">Chat with your virtual assistant here</p>

</center>
</body></html>

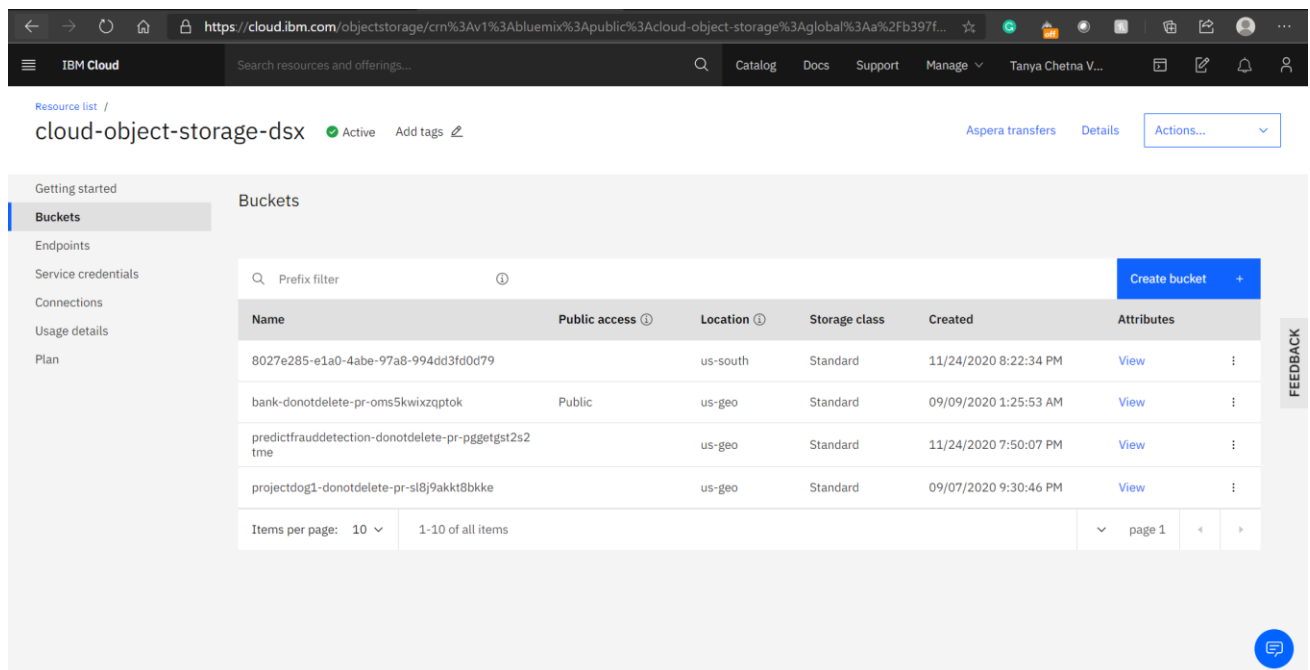
<script>
window.watsonAssistantChatOptions = {
integrationID: "567c4e21-c2f7-4765-bd44-9935eef2fd72", // The ID of this integration.
region: "eu-gb", // The region your integration is hosted in.
serviceInstanceID: "fc08724b-a457-4838-936e-e6b76678190c", // The ID of your service instance.
onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
const t=document.createElement('script');
t.src="https://web-chat.global.assistant.watson.appdomain.cloud/loadWatsonAssistantChat.js";
document.head.appendChild(t);
});
</script>
```

Let's discuss the parts of this code one by one:

- The **HTML <doctype> tag** is used for specifying which version of **HTML** the document is using. This is referred to as the document type declaration (DTD).
- For the background-image and img src specified URLs to the images stored in IBM's Bucket Storage Service and made it publicly accessible.

Steps for storing image in IBM Bucket Storage:

a) Go to your Cloud Object Storage



The screenshot shows the IBM Cloud Object Storage console. The top navigation bar includes the IBM Cloud logo, a search bar, and links to Catalog, Docs, Support, and Manage. The main content area is titled "cloud-object-storage-dsx" and shows a list of buckets. A sidebar on the left contains links to Getting started, Buckets, Endpoints, Service credentials, Connections, Usage details, and Plan. The Buckets table has columns for Name, Public access, Location, Storage class, Created, and Attributes. A "Create bucket" button is visible in the top right of the table area.

Name	Public access	Location	Storage class	Created	Attributes
8027e285-e1a0-4abe-97a8-994dd3fd0d79		us-south	Standard	11/24/2020 8:22:34 PM	View
bank-donotdelete-pr-oms5kwixzqptok	Public	us-geo	Standard	09/09/2020 1:25:53 AM	View
predictfrauddetection-donotdelete-pr-pggetgst2s2tme		us-geo	Standard	11/24/2020 7:50:07 PM	View
projectdog1-donotdelete-pr-sl8j9akkt8bkke		us-geo	Standard	09/07/2020 9:30:46 PM	View

b) Either create a new bucket here or use a pre-defined bucket.
Click on Upload files(objects) + and upload a file here.

Storage / cloud-object-storage-dsx / bank-donotdelete-pr-oms5kwixzqptok

Details Actions...

Objects

Configuration

Access policies

Endpoints

Service credentials

Connections

Usage details

Plan

Objects

Reminder: Objects are uploaded in multiple parts to optimize transfer performance. If an upload is interrupted before the transfer is completed, then the parts of the incomplete object that were uploaded prior to the interruption will count towards billable storage. While the console will alert users to incomplete multipart uploads, it is encouraged to routinely check for and clear out incomplete uploads using the REST API or an SDK. [Learn more](#)

Upload files (objects)

Object name	Size	Last modified
48062.gif	829.7 KB	11/24/2020 11:14:26 PM
Banking_loss_events.xlsx	890.2 KB	09/09/2020 1:28:22 AM
Bouquet_love-scaled.jpg	496.1 KB	11/23/2020 12:03:37 AM
Pin-on-Super-gif.gif	237.2 KB	11/24/2020 11:25:52 PM
aaaaaa.PNG	111.2 KB	11/24/2020 11:18:03 PM
c8a8cf28e2bc418808f2488043aac2e2.jpg	100.6 KB	11/23/2020 12:50:04 AM

I uploaded 2 images here which I will be using for my webpage. Also, you can upload images you want to use in your chatbot as part of response as image.

- c) To make the image publicly accessible make the bucket and its object public by updating the access policies and granting public access.

Storage / cloud-object-storage-dsx / bank-donotdelete-pr-oms5kwixzqptok

Details Actions...

Getting started

Buckets

Objects

Configuration

Access policies

Endpoints

Service credentials

Connections

Usage details

Plan

Bucket access policies

Users Service IDs Access groups **Public Access** Authorized IPs

Want to disable public access for this bucket?

Warning: Granting Public access to this bucket will allow anyone to access the bucket. [Learn more](#)

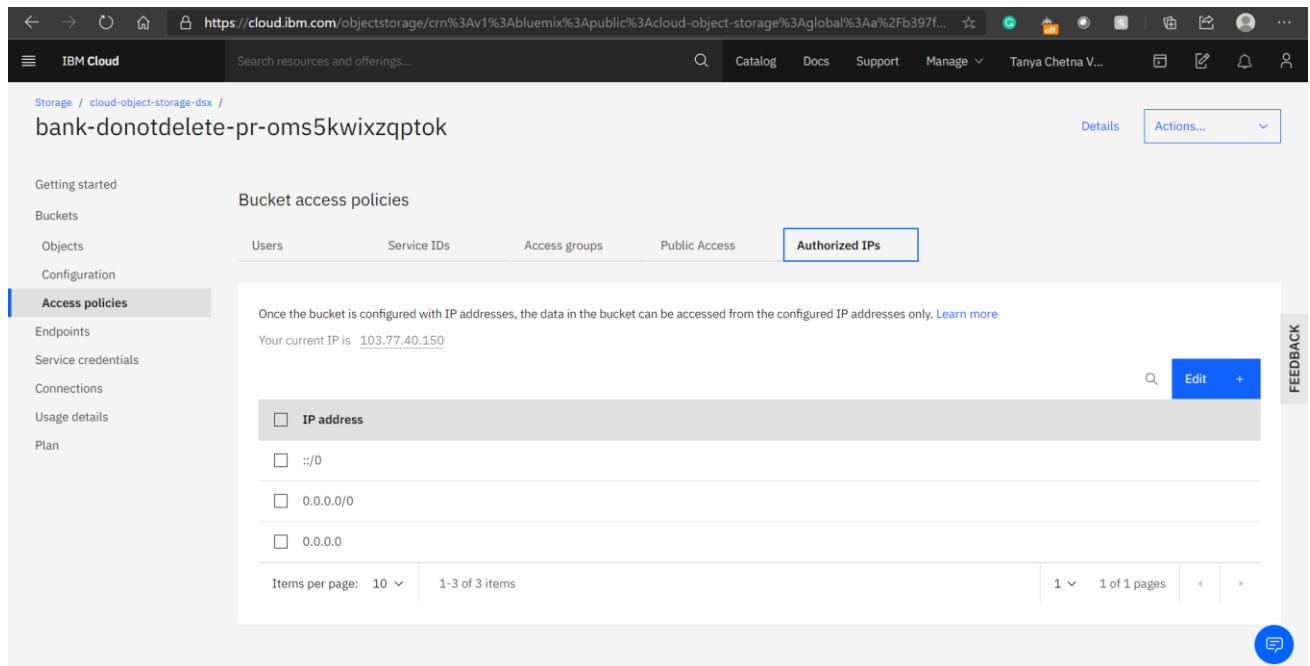
Access group: Public Access

Role for this bucket: Content Reader

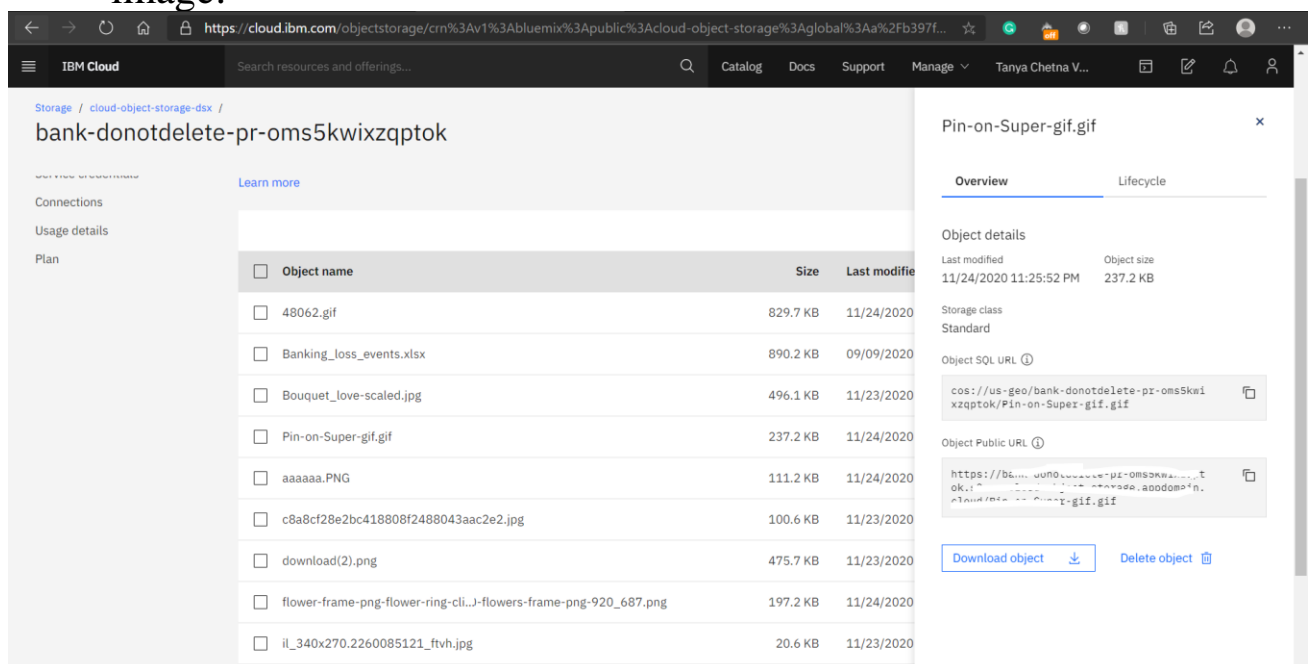
Public Access group will have the role of **Content Reader**.

As a Content Reader, one can read and list objects in the bucket.

Create access policy



d) After granting public access, click on the image details of the image you want to use to get the object public URL of the image:



- Embedded the chatbot to this HTML page using code.

Steps to Embed Chatbot:

a) Go to IBM Cloud and Launch Watson Assistant

b) Head into Banking Chatbot and select Webchat tab from right.

Now head into the Embed section and copy the code written there and paste it below the html code you wrote on your EC2 instance.

The screenshot displays the IBM Watson Assistant Lite console. The top navigation bar includes the IBM logo and the text 'IBM Watson Assistant Lite Upgrade'. The main content area is divided into several sections:

- Actions:** A section titled 'Build conversations easier than ever' with a list of bullet points and a 'Learn more' link. A blue button labeled 'Add an actions skill' is present.
- Dialog:** A section titled 'Banking Chatbot' with a subtitle 'A cognitive banking chatbot'. It includes a table with columns: LANGUAGE (English (US)), TRAINED DATA (24 Intents | 11 Entities | 165 Dialog nodes), VERSION, DESCRIPTION, and VERSION CREATED. Below the table is a section 'LINKED ASSISTANTS (1): Banking'.
- Search:** A section titled 'Turn any content into answers' with a list of bullet points and a 'Watch a brief demonstration' link. A blue button labeled 'Try Plus plan' is present.
- Integrations:** A section on the right side of the console. It includes a 'Preview link' button, a 'Web chat' button, and a section titled 'Choose a channel to deploy your assistant.' with a blue button labeled 'Add integration'.

Below the main content area, there is a 'Web chat' section. It includes a 'Close' button and a 'Saved' button. The 'Web chat' section has a 'Web chat' integration name. Below this, there is a tabbed interface with tabs: Style, Home Screen, Live agent, Suggestions, Security, and Embed. The 'Embed' tab is selected, showing a code snippet for embedding the chatbot on a website. The code is as follows:

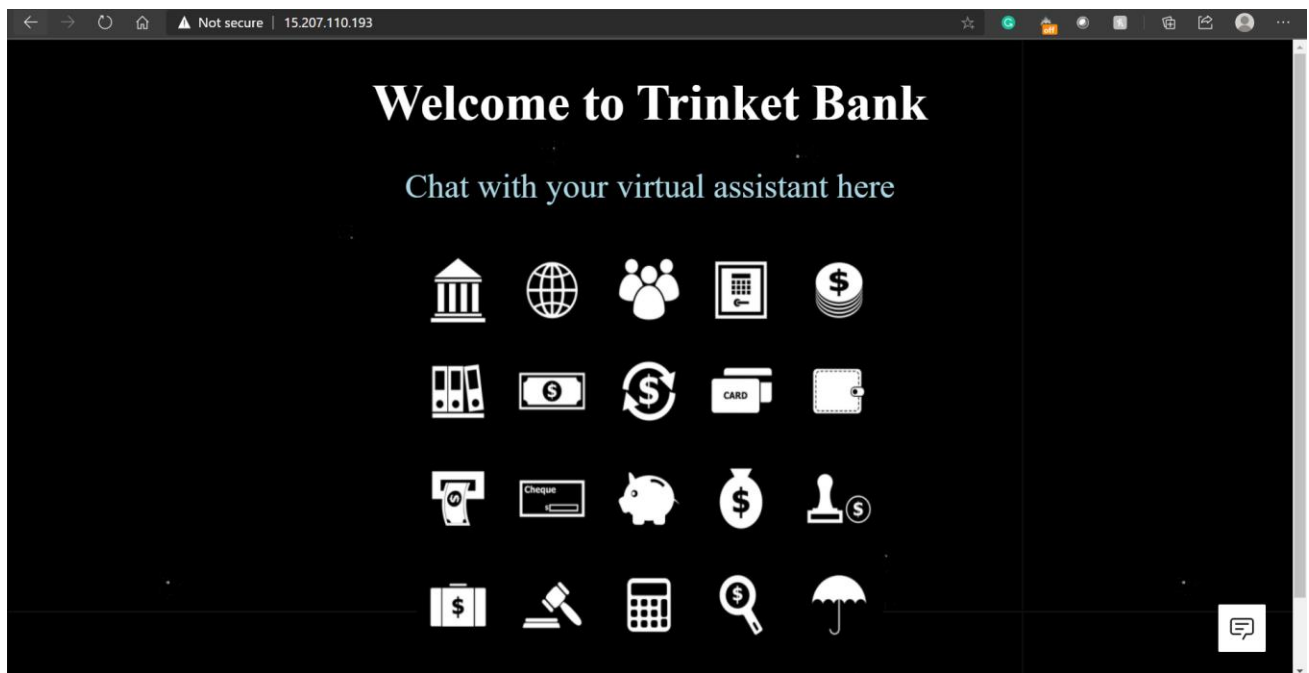
```
<script>
window.watsonAssistantChatOptions = {
  integrationID: " ", // The ID of this integration.
  region: "eu-gb", // The region your integration is hosted in.
  serviceInstanceID: " ", // The ID of your service instance.
  onLoad: function(instance) { instance.render(); }
};
setTimeout(function(){
  const t=document.createElement('script');
  t.src="https://web-chat.global.assistant.watson.cloud.ibm.com/instances/";
  document.head.appendChild(t);
});
</script>
```

Also , do some customisation if you want to change the color of the bar of Watson Assitant from default blue color to red or pink.I changed it to white.

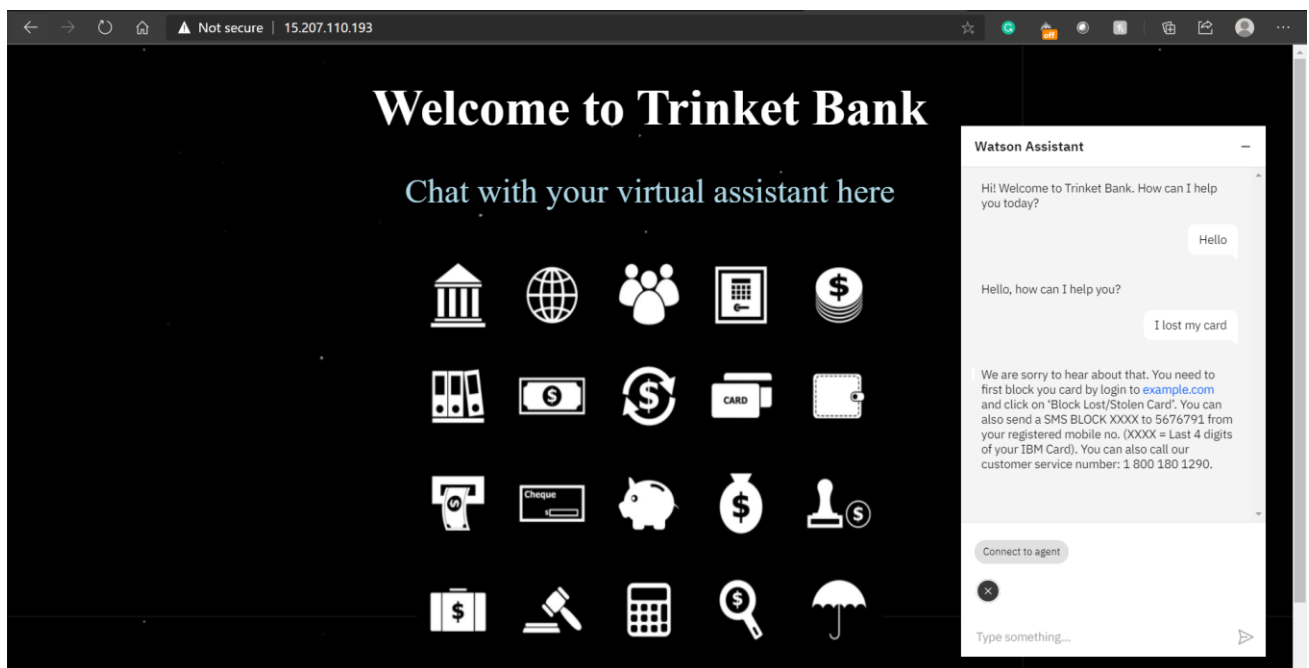
23)Write this code into nano editor and after writing press Cntrl+S to save and Cntrl+X to exit the nano editor.

24)Now go to the instances page of AWS and get the public IP of the Banking Chatbot instance.In my case its : <http://13.233.114.230/>

25) Paste this IP onto browser and that's it the page is accessible with the floating IBM Watson Assistant icon at the bottom right.



Click on the Watson Assistant icon to access and use it.

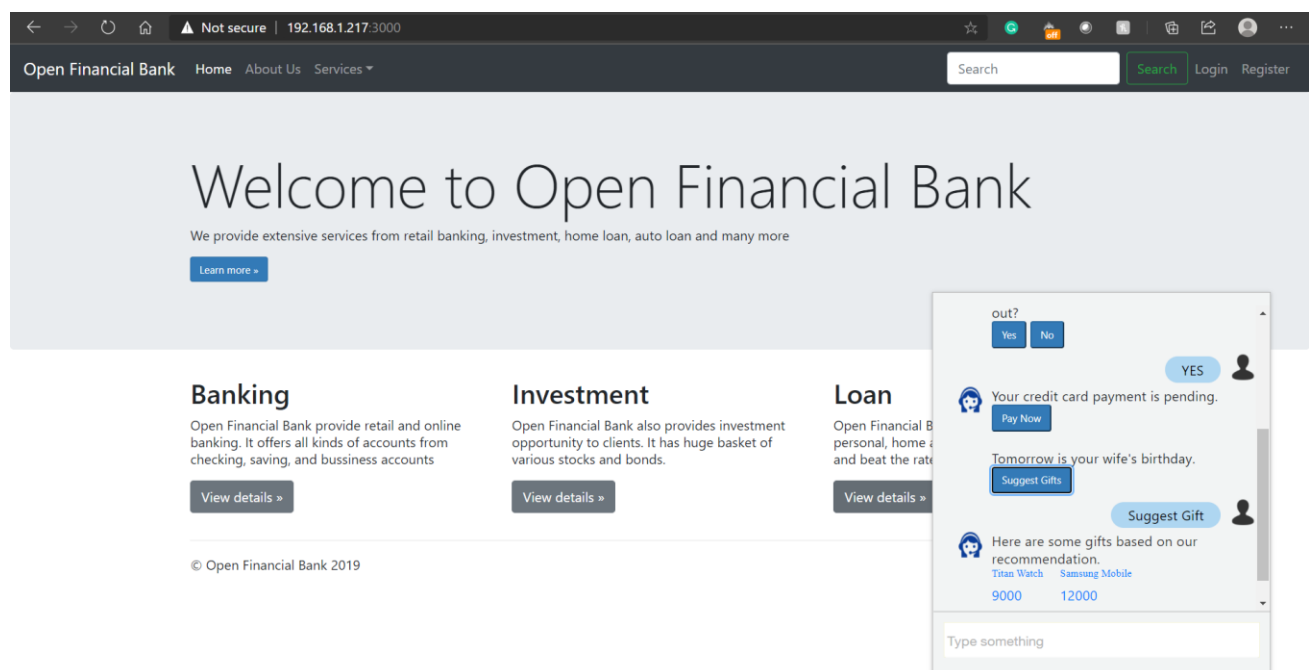


INFERENCE:

This was however a basic Banking Chatbot, but more customisations can be done and the Watson Assistant can be integrated with Watson Discovery or Natural Language Understanding services to provide a wider outlook to this chatbot. This can be easily used by banks to aid customers by providing data. As a future scope of this project, I will try to add more dialogs making it more customer-friendly and try to host it using a more informative webpage with greater number of functionalities.

In this project I integrated the services from 2 Cloud Service Providers- IBM Cloud and AWS, and build a publicly accessible chatbot.

One can use Nodejs and deploy locally this chatbot on there systems and integrate it with Watson Discovery and Natural Language Understanding so that not only the chatbot can iterate over greater set of data and then reply but also understand questions posed to it more properly. The steps for which are given in IBM docs and is pretty straightforward to apply. I deployed the app locally too to get hands on that as well:



For doing this just setup the Watson Discovery using necessary csv files (also uploaded to GitHub repository) and Natural Language Understanding services on IBM cloud and save the credentials. Create the .env file providing the credentials for each of the 3 services-Watson Assistant, Watson Discovery and Natural Language Understanding. Also install

node.js. Run the npm install and then npm start command in the same directory as the .env file and that's it. Your app is running locally now, while also fetching information from internet:

```
C:\Users\TANYA\Desktop\Untitled Folder\watson-banking-chatbot>npm audit fix
removed 4 packages, changed 2 packages, and audited 406 packages in 15s
found 0 vulnerabilities

C:\Users\TANYA\Desktop\Untitled Folder\watson-banking-chatbot>npm start
> watson-banking-chatbot@0.0.1 start
> node server.js

locale = en_IN
Using configured SKILL_ID: 77bfa7c8-3889-486c-a75d-712457119637
Watson Assistant is ready!
Server running on port: 3000
environments[0]: {
  "environment_id": "system",
  "name": "Watson System Environment",
  "description": "Shared system data sources",
  "read_only": true
}
environment: {
  "environment_id": "system",
  "name": "Watson System Environment",
  "description": "Shared system data sources",
  "read_only": true
}
environment: {
  "environment_id": "89be6596-8f4b-4b8c-a038-6f758c935d95",
  "name": "byod",
  "description": "",
  "created": "2020-11-24T16:53:57.525Z",
  "updated": "2020-11-24T16:53:57.525Z",
  "read_only": false
}
Found Discovery environment using DISCOVERY_ENVIRONMENT_ID.
{
  environment_id: '89be6596-8f4b-4b8c-a038-6f758c935d95',
  name: 'byod',
  description: '',
  created: '2020-11-24T16:53:57.525Z',
  updated: '2020-11-24T16:53:57.525Z',
  read_only: false
}
Found Discovery collection using DISCOVERY_COLLECTION_ID.
{
```

Chatbot Link:

Webpage: Trinket Bank

Preview Link: <https://web-chat.global.assistant.watson.cloud.ibm.com/preview.html?region=eu-gb&integrationID=63656b1f-8922-4104-a0c5-b2879a23ac25&serviceInstanceID=fc08724b-a457-4838-936e-e6b76678190c>

GitHub: [TanyaChetnaVaish/MiniProjectSemV \(github.com\)](https://github.com/TanyaChetnaVaish/MiniProjectSemV)

Thankyou!