Retrieval Augmented Generation (RAG)

RAG (Retrieval-Augmented Generation) is a hybrid model architecture that combines **retrieval-based methods** and **generative models** to enhance the performance of AI systems, particularly in tasks such as answering questions, generating text, or providing information. This technique aims to improve the quality and accuracy of generated text by incorporating external knowledge, rather than relying solely on a model's internal knowledge.

Key Components of RAG:

1. **Retrieval Component (Retriever)**:
    - The retrieval phase involves identifying and fetching relevant documents or passages from an external knowledge base (e.g., databases, the internet, or custom corpora).
    - This step enhances the model's ability to generate factually accurate responses by grounding the generation process in real-world information.
2. **Generative Component (Generator)**:
    - After retrieving relevant data, a generative model (like GPT, BERT, or another transformer-based language model) uses this information to produce a coherent and contextually appropriate response. ○ This allows for a more informed and contextual output compared to purely generative models, which rely only on learned patterns from training data.

Working Process:

1. **Query**: The user provides an input query.
2. **Retrieval**: The system searches for the most relevant documents or pieces of information from a knowledge source using the retriever.
3. **Generation**: The retrieved information is then passed to the generative model, which synthesizes it with the query to produce a response.

Benefits of RAG:

- **Improved Accuracy**: The retrieval step helps ground the generated output in factual data, reducing the risk of "hallucination" (i.e., when a generative model invents incorrect information).
- **Knowledge Scalability**: Instead of being limited to the knowledge stored within a model's parameters, RAG models can access vast amounts of up-to-date information from external sources.

- **Adaptability**: RAG models can be easily updated by changing the retrieval source, without retraining the entire model, which is particularly useful for rapidly changing domains like news or research.

Use Cases:

1. **Question Answering (QA)**: Providing fact-based answers to user queries by fetching information from sources like Wikipedia or custom databases.
2. **Customer Support**: Assisting users with up-to-date product or service information by accessing a dynamic knowledge base.
3. **Legal and Research**: Supporting legal professionals or researchers by retrieving relevant case studies, research papers, or documents.
4. **Content Generation**: Creating factually correct and contextually rich articles, reports, or summaries.

Example Models:

- **RAG Model from Facebook AI**: One of the most well-known implementations of RAG, designed for tasks like QA by combining retrieval and generation to achieve better results.

Comparison with Traditional Models:

- **Purely Generative Models**: Can generate fluent text but may produce factually incorrect or inconsistent information.
- **Retrieval-based Models**: Provide accurate data but may struggle with generating coherent and contextually appropriate responses.
- **RAG**: Combines the strengths of both approaches—factual accuracy from retrieval and contextual fluency from generative models.

In summary, RAG is a powerful architecture that leverages external knowledge sources to improve the quality and reliability of generated text, making it suitable for applications where factual correctness is critical.

To begin with the Lab:

1. **Use Case**: The goal is to build an **Employee FAQ** system using **Retrieval-Augmented Generation (RAG)**, specifically focusing on HR policies.
2. **Amazon Bedrock**: The application will utilize the **Amazon Bedrock Cloud Foundation model** from **Anthropic** for generating responses.
3. **Orchestration with LangChain**: The system will be orchestrated using **LangChain**, which integrates different components like retrieval and generation.

4. **Data Source**: The Bedrock model will be integrated with an **internal data source**, specifically a **PDF document** containing the company's HR policies (such as leave policies).
5. **Geographical Scope**: Large enterprises often have **region-specific HR policies** documented in separate PDFs, uploaded on the organization's intranet (e.g., **SharePoint**).
6. **Objective**: The system will allow employees to ask questions related to HR policies, and it will retrieve relevant information from the uploaded PDF documents using RAG.
7. There are some prerequisites for this lab. First, you should have VS Code on your laptop and in this, you should have these two extensions installed.

7. Create a new folder
8. Open VSCode and create a virtual environment
   Use command – python -m venv . (this dot represents your current raw directory)

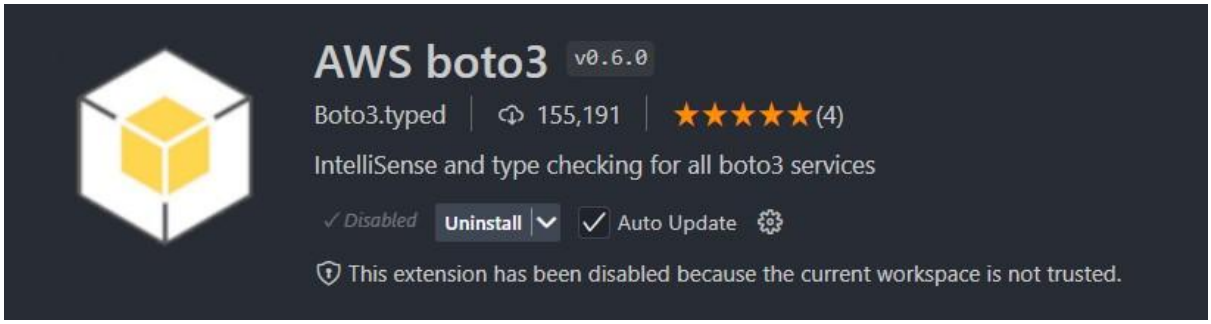
```
PS C:\Users\amanr\Desktop\HR_QA_RAG> python -m venv .
```

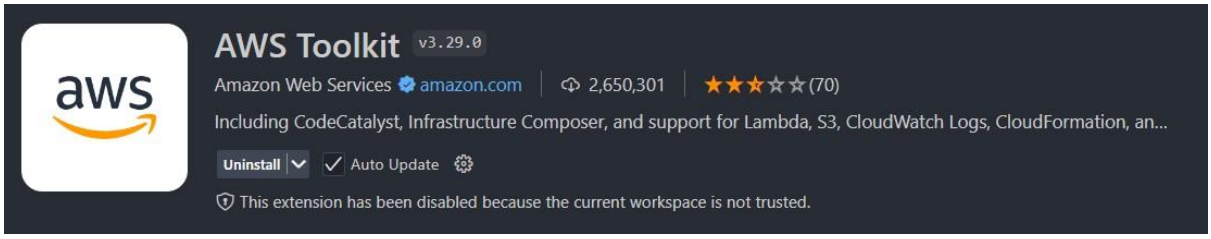
9. Now activate the virtual environment
   Use command - Scripts\activate

```
PS C:\Users\amanr\Desktop\HR_QA_RAG> Scripts\activate
```

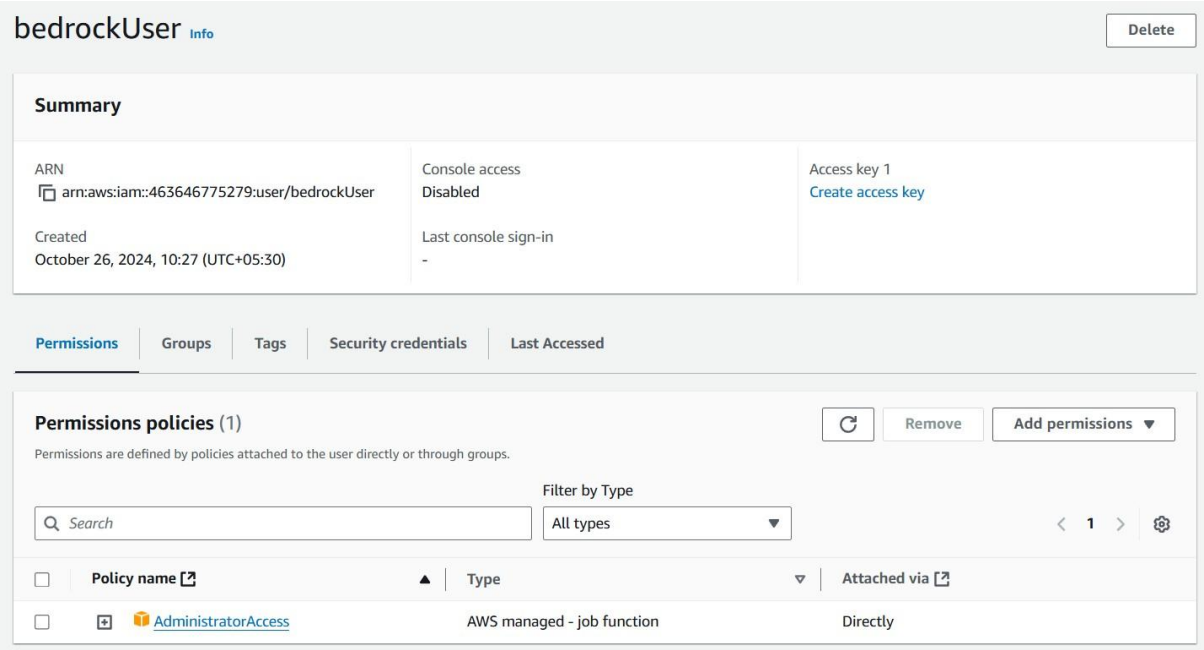
10. After activating the virtual environment, configure the aws account.
    Use command – aws configure

11. Now copy and paste the access Id, key and region.
12.





8. Next you should have Python on your laptop with that you should also have the AWS CLI installed.
9. Then you need to come to your console, create a User from IAM, and attach Administrator access to that user.
10. After that you need to go to the security credentials of this user and create access and secret access key for this user. Then configure this user in AWS CLI on your laptop.




```
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> aws configure
AWS Access Key ID [****************4JMH]: AKIAWX44AK7XSJHLZLPA
AWS Secret Access Key [****************UYdX]: BvL+AcBc5Z7kFLEzCG6ANQBaa0xHgISSOC3SHNAJ
Default region name [ap-south-1]: us-east-1
Default output format [json]: json
```

11. In your VS Code you need to open the terminal and install boto3, langchain, and streamlit using the below commands.

    pip install boto3

    pip install langchain

    pip install streamlit

    pip install -U langchain-aws

    pip install anthropic

    pip install flask-sqlalchemy

    pip install pypdf

    pip install faiss-cpu

    pip install faiss-gpu

    pip install langchain_community

Note - if not cpu use gpu instead of cpu

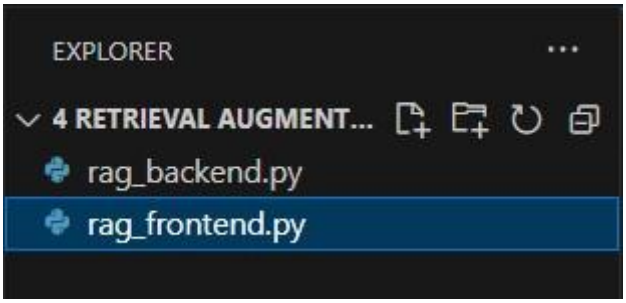
Direct Command to install all the libraries in one go:

pip install boto3 langchain streamlit -U langchain-aws anthropic flask-sqlalchemy pypdf faiss-cpu

```
PROBLEMS 1  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  AZURE  POSTMAN CONSOLE                    pwsh

PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show boto3
Name: boto3
Version: 1.35.49
Summary: The AWS SDK for Python
Home-page: https://github.com/boto/boto3
Author: Amazon Web Services
Author-email:
License: Apache License 2.0
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: botocore, jmespath, s3transfer
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show langchain
Name: langchain
Version: 0.3.4
Summary: Building applications with LLMs through composability
Home-page: https://github.com/langchain-ai/langchain
Author:
Author-email:
License: MIT
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: aiohttp, langchain-core, langchain-text-splitters, langsmith, numpy, pydantic, PyYAML, requests, SQLAlchemy, tenacity
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot> pip show streamlit
Name: streamlit
Version: 1.39.0
Summary: A faster way to build and share data apps
Home-page: https://streamlit.io
Author: Snowflake Inc
Author-email: hello@streamlit.io
License: Apache License 2.0
Location: C:\Users\PULKIT\anaconda3\Lib\site-packages
Requires: altair, blinker, cachetools, click, gitpython, numpy, packaging, pandas, pillow, protobuf, pyarrow, pydeck, requests, rich, tenacity, toml, tornado, typing-ex
tensions, watchdog
Required-by:
PS D:\AWS Bedrock\3 Bedrock Creating Chatbot\Chatbot>
```

14. Now you will get a folder that contains the code files with this lab. So, you need to open that folder in VS Code and check the code.



15. So, below is the explanation of the backend code.
    This code outlines how to build a **Retrieval-Augmented Generation (RAG)** system for HR policies using **LangChain** and **Amazon Bedrock**:
    - It loads and splits an **HR policy PDF** using **PyPDFLoader** and **RecursiveCharacterTextSplitter** for efficient processing.
    - **Embeddings** are created using **Amazon Titan Text Embeddings**, and these embeddings are stored in **FAISS Vector DB** via **VectorstoreIndexCreator**.
    - A function connects to the **Claude Foundation Model** from **Amazon Bedrock** to generate responses.
    - The **hr_rag_response()** function queries the **vector DB**, retrieves relevant information, and sends both the query and context to the LLM for answer generation.

16. Below is the explanation of frontend code.

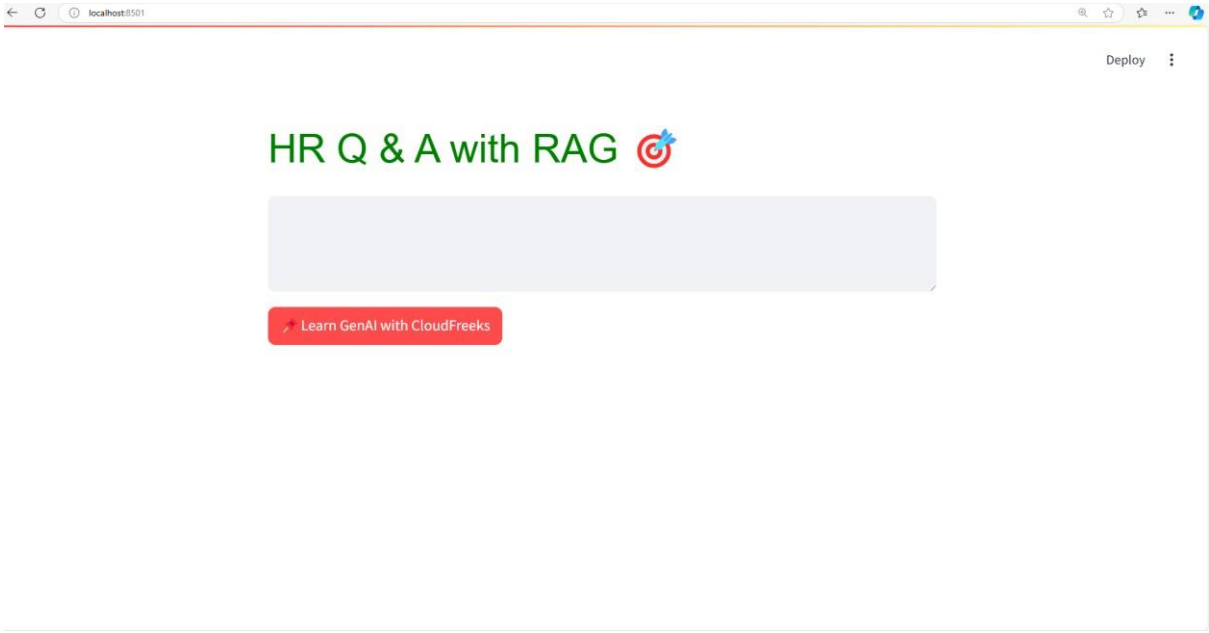
    This **Streamlit** frontend code provides a user interface for an **HR Q&A system** powered by **RAG**:

    - It sets up a page titled "HR Q & A with RAG " with a custom heading and uses a **spinner** while the **vector index** (HR policy data) is being loaded from the backend (hr_index()).
    - Users can input their questions via a **text area**, and a button labeled " Learn GenAI with Rahul Trisal" triggers the query.
    - Upon clicking the button, another spinner appears while the system processes the query by calling the **hr_rag_response()** function from the backend.
    - Finally, the **retrieved answer** is displayed using st.write().

17. Now to run this you need to the command given below in your terminal.

    streamlit run rag_frontend.py

18. Below you can see that our RAG is running perfectly.



19. Here we asked some questions from it and based on the document it responded. To access the document from which it is answering you can open this link.

    https://www.upl-ltd.com/images/people/downloads/Leave-Policy-India.pdf

HR Q & A with RAG 🎯

how many casual leaves in a year?

📌 Learn GenAI with CloudFreeks

Based on the provided context, the number of casual leaves in a year is 7. The text states "7 casual leaves in a year."

HR Q & A with RAG 🎯

how many privilege leaves in a year?

📌 Learn GenAI with CloudFreeks

Based on the provided context, employees are eligible for 21 working days of privilege leave in a calendar year.

HR Q & A with RAG 🎯

how many sick leaves?

📌 Learn GenAI with CloudFreeks

Based on the provided context, I do not have enough information to determine how many sick leaves are allowed. The context mentions sick leave for more than 3 days requires a medical certificate, but does not specify the total number of sick leaves allowed. Since the question asks for a specific number and the context does not provide that, I don't know the answer.