RAG POC How-to Guide

Deploy a Generative AI chatbot for your private knowledge base using Retrieval Augmented Generation (RAG)

February 2024

H19943

White Paper

Abstract

The RAG POC How-to Guide is your step-by-step tutorial for deploying a Generative AI chatbot use case Proof-of-Concept (POC) with Retrieval Augmented Generation (RAG).

Copyright

The information in this publication is provided as is. Dell Inc. makes no representations or warranties of any kind with respect to the information in this publication, and specifically disclaims implied warranties of merchantability or fitness for a particular purpose.

Use, copying, and distribution of any software described in this publication requires an applicable software license.

Dell Inc. believes the information in this document is accurate as of its publication date. The information is subject to change without notice.

Contents

Executive summary

What to expect

This POC is meant to help you explore what is possible with Generative AI. This is built with small models in mind and will require a bit more tuning on your side. It is, therefore, intended for use cases that keep people the loop.

Revisions

Date	Part number/ revision	Description
February 2024	H19943	Initial release

We value your feedback

Dell Technologies and the authors of this document welcome your feedback on this document. Contact the Dell Technologies team by email.

Author: David O'Dell

Note: For links to other documentation for this topic, see the AI Info Hub

Introduction to RAG POC

Important notes

Note: The software and sample files are provided “as is” and are to be used only in conjunction with this POC application. They should not be used in production and are provided without warranty or guarantees. Please use them at your own discretion.

Note: As you begin to develop and experiment with your POC, we recommend hosting the project in a secure on-premises environment to protect sensitive data.

This guide

The guide includes directions and links to the Dell Technologies Generative AI repository on GitHub, along with instructions for building the POC with Jupyter Notebooks, Conda, Miniconda, Python, and other tools. It also contains information about the value of building Generative AI use cases with NVIDIA AI Enterprise.

What is covered:

- RAG process diagram
- Chatbot software stack diagram
- Hardware and software requirements
- Detailed step-by-step instructions to build your POC
- Guidance on running and using the POC
- Prompt engineering tips

Other resources

Table 1. Other resources in the RAG POC kit

RAG POC prompting emails	Topical use case solution briefs	Topical use case “cheat sheets”	RAG POC persona guide
These emails are pre-written, to help you execute a training program using your new POC. They include pre-launch informative emails, launch details, and post-launch exercises for your organization. Email text marked in red [lorem ipsum] will need to be completed by your team.	The solution briefs outline potential RAG POC use cases and how to cluster them for efficiency in your Generative AI practice. • Content creation • Digital assistant • Natural language search	1. The solution briefs outline potential RAG POC use cases and how to cluster them for efficiency in your Generative AI practice. • Content creation • Digital assistant • Natural language search •	This one-page guide details the importance of the different leaders and personas within your organization for achieving success with your RAG POC. It takes a village to find success with Generative AI; use this guide to help guide your onboarding.

Retrieval Augmented Generation (RAG)

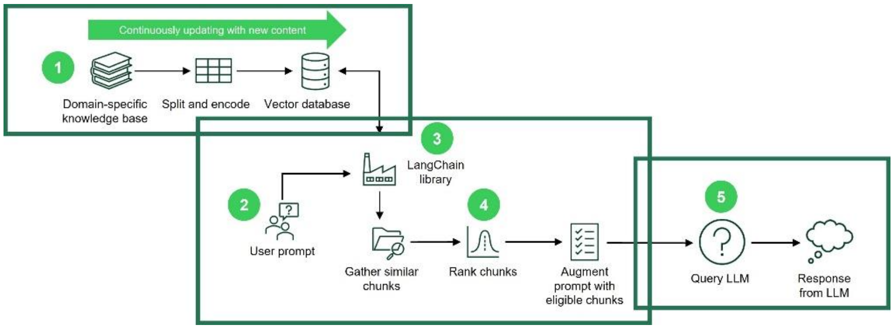
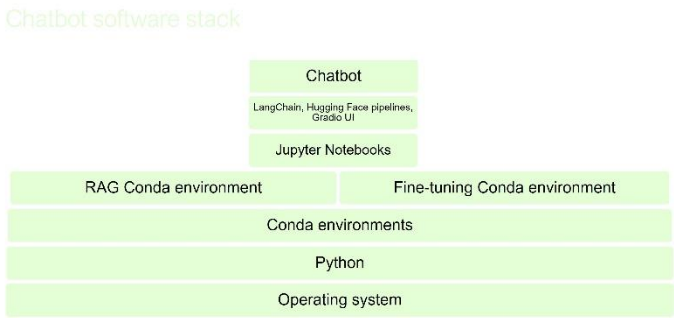


Figure 1. Reference diagram of RAG

In figure 1 there are several parts. 1) Knowledge base content is split and encoded into vector database. 2) User prompts the application. 3) LangChain library tools ingest the prompt, analyze it and gather similar chunks of content. 4) Chunks of related content are ranked and then attached to the prompt. 5) Prompt and eligible chunks are fed to the LLM for response.

Chatbot software stack


Chatbot software stack

Chatbot

LangChain, Hugging Face pipelines, Gradio UI

Jupyter Notebooks

RAG Conda environment | Fine-tuning Conda environment

Conda environments

Python

Operating system

Figure 2. Representation of a chatbot software stack

Hardware and software requirements

What is required to run your RAG POC?

Hardware

- On-premises compute (VM or bare metal) 32 GB RAM, 4vCPU
- GPU access, 16Gb or higher

Software

- Ubuntu 22.04 or comparable Linux environment (WSL is also possible)
- Miniconda environments
- Python 3.10 or higher
- Jupyter notebooks installed.

Storage

- 500-1000 GB for combined model cache and PDF dataset storage
- As you experiment, this will grow.

Network

- Access to public internet

Note: Test environment: Dell PowerEdge with Nvidia T4 and Dell Precision Mobile Workstation 7780 with Nvidia RTX5000.

Step-by-step instructions

First steps

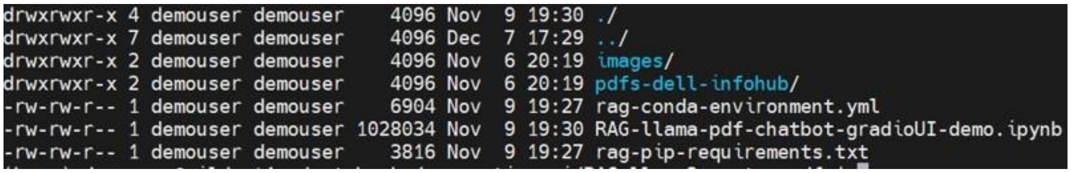
Gather essential account access and API keys

- Hugging Face – free open-source AI datasets and models, your first and most important resource. www.huggingface.co
- Weights and biases – free training monitoring package for competitive analysis comparisons. wandb.ai/
- GitHub – free notebooks, lots of libraries. www.github.com
- You'll need to set up your account AND set up a simple API token for each. You'll definitely use these on a regular basis as will the customer.
- Meta (Facebook) Llama model usage permission form: https://ai.meta.com/resources/models-and-libraries/llama-downloads/

Clone the Dell-examples generative AI repo

- In GitHub, simply git clone our repo located at the URL below:
- Notebook and dataset files available at https://github.com/dell-examples/generative-ai
- Everything is stored inside the repo directory. You don't have to move or adjust anything.

Dell example file structure



```
drwxrwxr-x 4 demouser demouser    4096 Nov  9 19:30 ./
drwxrwxr-x 7 demouser demouser    4096 Dec  7 17:29 ../
drwxrwxr-x 2 demouser demouser    4096 Nov  6 20:19 images/
drwxrwxr-x 2 demouser demouser    4096 Nov  6 20:19 pdfs-dell-infohub/
-rw-rw-r-- 1 demouser demouser    6904 Nov  9 19:27 rag-conda-environment.yml
-rw-rw-r-- 1 demouser demouser 1028034 Nov  9 19:30 RAG-llama-pdf-chatbot-gradioUI-demo.ipynb
-rw-rw-r-- 1 demouser demouser    3816 Nov  9 19:27 rag-pip-requirements.txt
```

Figure 3. File structure of Dell-examples generative AI repo

Next steps

The PDF datastore

Using PDF datastore

- The PDF files that are included in the repo are located in the "pdfs-dell-infohub" directory. This is a random assortment of freely available PDF whitepapers, blogs, and tech documents from https://infohub.delltechnologies.com/
- You can use these as-is or delete and replace them with other PDF files.

- The directory name is hardcoded into the notebook. You can easily make adjustments to the name or location—you just need to modify the code references.

The vector database

More info on vector databases

- When the notebook is run, the PDF files are encoded and embedded into a ChromaDB vector database in the directory “db” located alongside the same directory as the notebook.
- You can easily change the location of this in the notebook.
- For demo purposes and experimentation, the code deletes the “db” directory every time the notebook is run, and a new chatbot session is created so the content is up to date. You can change this as well if you like.

Install Miniconda

Creating use-case-based~~based~~ Conda environments

- When working with Python tools and libraries it is highly recommended to use virtual environments.
- As you experiment with different notebooks, compatibility issues become very evident as different libraries are installed or uninstalled for certain tools.
- Miniconda is used to segregate your work area into use-case-based Conda environments.
- RAG workloads are very similar to other RAG workloads, so the libraries and tools won’t fluctuate much. What you use for one notebook you’ll probably use for another. Fine-tuning workloads is also similar; however, both of these workloads are different enough from each other that they should have their own Conda environments. Try to avoid mixing your RAG notebook kernel with your FT notebook kernel.
- Installer link: https://docs.conda.io/projects/miniconda/en/latest/index.html
- Conda environment YAML file is available in the GenAI GitHub repo.
- Create the RAG environment from the included file: https://conda.io/projects/conda/en/latest/user-guide/tasks/manage-environments.html#creating-an-environment-from-an-environment-yml-file

Install Jupyter Notebooks

Visualizing and sharing Python code with Jupyter Notebooks

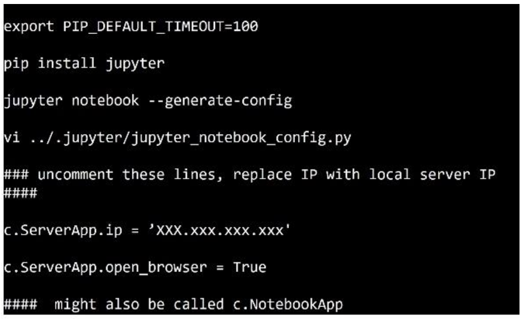
- After installing the base Jupyter Notebooks, it helps to install nb_conda_kernels so that each Conda environment you create will become a unique Python kernel to choose from inside Jupyter Notebooks.
- https://jupyter.org/install
- https://anaconda.org/conda-forge/nb_conda_kernels

Starting the Jupyter Notebook server

In the (base) Conda environment

- Edit the Jupyter config file and replace “localhost” with your Linux host IP address.
- At the command line, start the notebook server from the same directory as the GenAI GitHub repo.
- Jupyter Notebooks will create a small web server on the Linux host to serve the UI from. You’ll need a Windows jump box or browser with network access to your Linux host to reach that URL.

Example of Jupyter Notebook server



```
export PIP_DEFAULT_TIMEOUT=100

pip install jupyter

jupyter notebook --generate-config

vi ../.jupyter/jupyter_notebook_config.py

### uncomment these lines, replace IP with local server IP
####

c.ServerApp.ip = 'XXX.xxx.xxx.xxx'

c.ServerApp.open_browser = True

####  might also be called c.NotebookApp
```

Figure 4. Example of how to start the Jupyter Notebook server

Running the RAG chatbot notebook

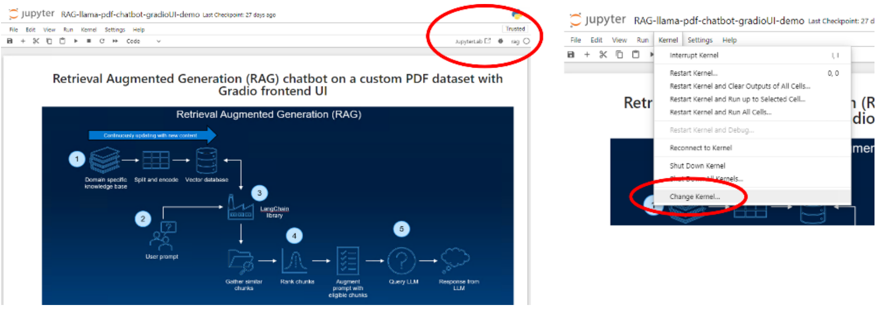
Preparing the POC for use

First step

Verify that the RAG Python kernel is selected and displayed on the upper-right kernel menu.

- If not, go to Kernel and Change Kernel.
- If RAG is not shown, redo your Conda environments.

Visual of RAG Python kernel



Retrieval Augmented Generation (RAG) chatbot on a custom PDF dataset with Gradio frontend UI

jupyter
RAG-llama-pdf-chatbot-gradioUI-demo Last Checkpoint: 27 days ago
File
Edit
View
Run
Kernel
Settings
Help
Code
Trusted
JupyterLab
rag
Retrieval Augmented Generation (RAG)
Continuously updating with new content
1
Domain specific knowledge base
Split and encode
Vector database
2
User prompt
3
LangChain library
4
Gather similar chunks
Rank chunks
Augment prompt with eligible chunks
5
Query LLM
Response from LLM
Interrupt Kernel
Restart Kernel...
Restart Kernel and Clear Outputs of All Cells...
Restart Kernel and Run up to Selected Cell...
Restart Kernel and Run All Cells...
Restart Kernel and Debug...
Reconnect to Kernel
Shut Down Kernel
Shut Down All Kernels...
Change Kernel...

Figure 5. Visual of selecting RAG Python kernel

Run the RAG chatbot notebook

The Notebook will build the deployment anywhere from 3-5 minutes.

- Navigate to the notebook file and double-click to open.
- Select RUN and then RUN ALL CELLS.

Images of running RAG chatbot notebook

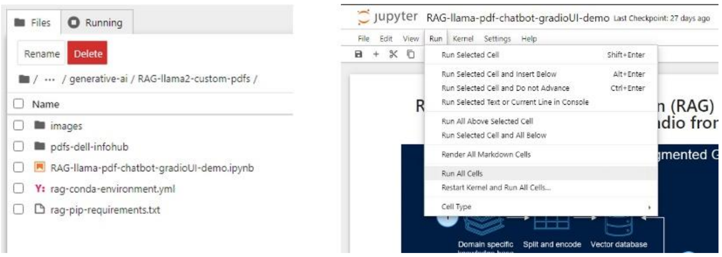

Files
Running
Rename
Delete
/ ··· / generative-ai / RAG-llama2-custom-pdfs /
Name
images
pdfs-dell-infohub
RAG-llama-pdf-chatbot-gradioUI-demo.ipynb
rag-conda-environment.yml
rag-pip-requirements.txt
jupyter
RAG-llama-pdf-chatbot-gradioUI-demo
Last Checkpoint: 27 days ago
File
Edit
View
Run
Kernel
Settings
Help
Run Selected Cell Shift+Enter
Run Selected Cell and Insert Below Alt+Enter
Run Selected Cell and Do not Advance Ctrl+Enter
Run Selected Text or Current Line in Console
Run All Above Selected Cell
Run Selected Cell and All Below
Render All Markdown Cells
Run All Cells
Restart Kernel and Run All Cells...
Cell Type
Domain specific
Split and encode
Vector database

Figure 6. Illustration of how to run notebook

Using the RAG chatbot

Simply type your question into the prompt field or select from the examples below. Then click submit. Time will vary depending on many factors such as question complexity, document retrieval, and GPU speed.

Images of how to use chatbot

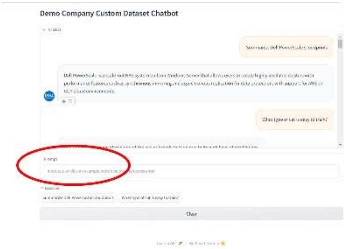


Figure 7. Illustration of how to ruse RAG chatbot

Prompt engineering tips

Keys to prompt engineering

How to find success with prompt engineering

- LLMs depend greatly on context and having guidance on the tone and behavior they should respond in. A few tips on getting more out of your prompt:
  - Tell the LLM what role to assume, such as “advertising copywriter” or “sales representative” or “medical doctor.”
  - Ask the LLM to “be concise.”
  - Let the LLM know about the audience preferences: “this audience prefers technical language.”
  - Be specific about formatting: “3-item list with bullet points” or “product brief with persuasive headline and product description.”

References

Dell Technologies documentation

The following Dell Technologies documentation provides other information related to this document. Access to these documents depends on your login credentials. If you do not have access to a document, contact your Dell Technologies representative.

- Generative AI in the Enterprise
- *Using Retrieval Augmented Generation on a Custom PDF Dataset*
- *Document 2*

NVIDIA documentation

The following NVIDIA documentation provides other information related to this document. Access to these documents depends on your login credentials.

- *Nvidia Developer version RAG Microservices Workflow*
- *Document 2*