# Agentic RAG

- Introduction
- Amazon Bedrock Agent
- Simple 3-Node Agent using LangGraph (Executor → Judge → Router)
- Complex 6-Node Agent using LangGraph (Planner → Retriever → Analyzer → Reflector → Router → Finalizer)
- Agentic RAG implemented as Step Function
- Agents implementations
  - Planner
  - Retriever
  - Analyzer
  - Reflector

## Introduction

For implementing Agentic RAG, we will use 3 different approaches:

- **Amazon Bedrock Agent** implementing a sequence of:
  - Plan
  - Retrieve
  - Analyse & Execute
  - Reflect
- **Simple 3-Node Agent using LangGraph** implementing a sequence of Executor → Judge → Router
- **Complex 6-Node Agent using LangGraph** implementing a sequence of Planner → Retriever → Analyzer → Reflector → Router → Finaliser
- **Agentic RAG implemented as Step Function** as a sequence of Planner → Retriever → Analyzer → Reflector → Router → Finaliser

## Amazon Bedrock Agent

The implementation of Agentic RAG with an Amazon Bedrock Agent will require:he implementation of Agentic RAG with an Amazon Bedrock Agent will require:

- System prompt to steer the Plan → Retrieve → Analyse & Execute → Reflect sequence
- Change of default instructions for Orchestration of the Amazon Bedrock Agent

Advantages:

- Simple to implement
- Built-in testing interface
- Existent orchestration, pre and post-processing

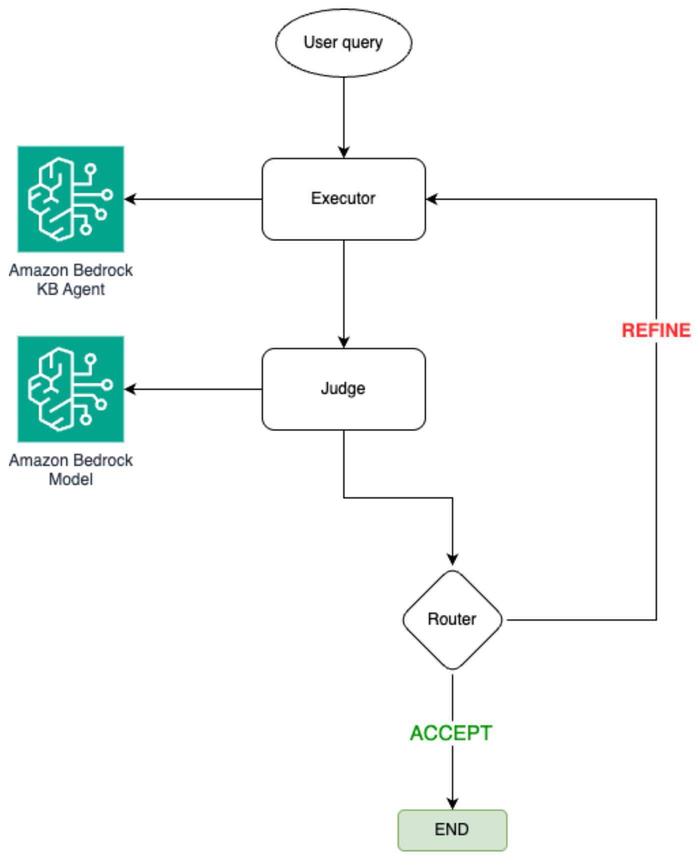
Drawbacks:

- Citations are not included in the final answer
- Directives in the system prompt are not consistently applied

## Simple 3-Node Agent using LangGraph (Executor → Judge → Router)

The steps are as following:

- **Executor** - it takes the user query and calls an **Amazon Bedrock Knowledge Base Agent** to retrieve citations + answer from the Knowledge Base
- **Judge** -  it takes the system state from the **Executor** node and use a “LLM as a Judge” to evaluate if the answer (and citation) are aligned with the initial query. If the answer is considered complete, it returns either **ACCEPT** or **REVISE**, as well as a justification.
- **Router** - it takes the system state from the **Judge** node and either exit (END state) or returns to the **Executor**. A maximum of 3 steps will be run.

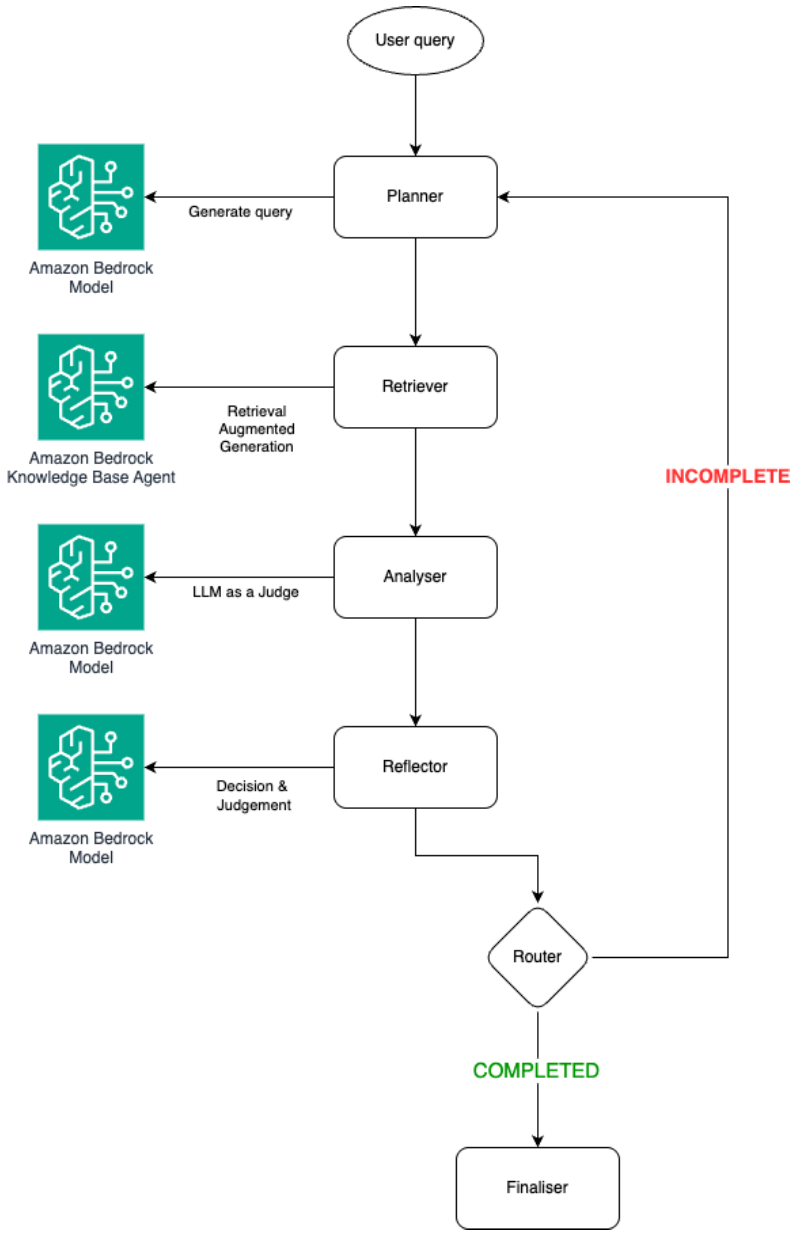


## Complex 6-Node Agent using LangGraph (Planner → Retriever → Analyzer → Reflector → Router → Finalizer)

The steps are as following:

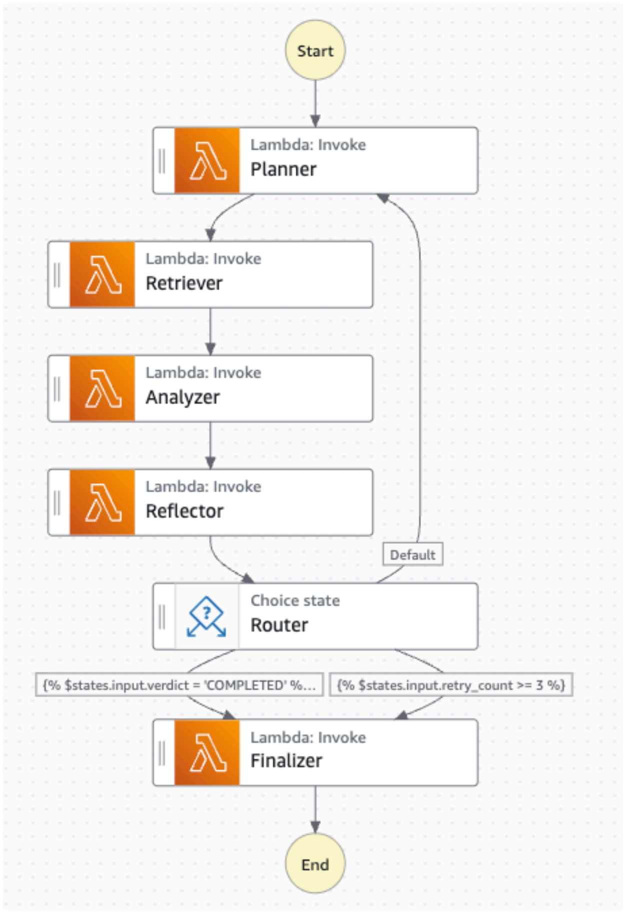
- **Planner** - it takes the user query and the history of queries, answers & citations and generates a comprehensive query that is passed to the retriever
- **Retriever** - calls the Amazon Bedrock Knowledge Agent with the query received from Planner and returns the answer and citations.
- **Analyzer** - receives the query, and the history of answers and citations and uses a “LLM as a Judge” to evaluate if the citations and answer are aligned with the initial query.
- **Reflector** - receives the result of analyser and decide if the answer is complete or incomplete. It also provide a justification.
- **Router** - routes to the Finaliser or Planner: If the answer is complete, will go to the Finaliser node/step, otherwise, if the answer is evaluated as incomplete, will get back to Planner. An additional condition is that if the number of routings exceeds or is equal with 3, will exit with an partial answer (routing max number = 3).

- **Finaliser** - end step. Output the current state of the system (containing query, history of queries, answers and citations, as well as decision and justification).



## Agentic RAG implemented as Step Function

The diagram of the Step Function is presented in the next figure.



The Planner, Retriever, Analyzer, Reflector steps are implemented as lambda functions as following:

- **Planner** - invoke a **Bedrock Model** to generate current query; add the query to query history list; if route count > 0, will use for generating current query also the retrieval history
- **Retriever** - invoke a **Bedrock Agent** (Knowledge Base) to retrieve the current content (summary + citations) based on current query; add the retrieved data to the retrieval history
- **Analyzer** - use the current query, and the retrieval information and invoke a **Bedrock Model** to evaluate the quality of retrieval relative to the current query. Add a synthesis.
- **Reflector** - uses the synthesis and the retrieval history and present through invoking a **Bedrock Model** to provide a verdict and a justification.
- **Router** - implemented as a choice state - decide based on routing number and decision if the next step is Finalizer or reroute to Planner. Condition is:
  - If verdict is COMPLETED or route_count >=3 will route to Finalizer
  - If verdict is INCOMPLETE and route_count < 3 will route to Planner
- **Finalizer** - reformat the input to prepare the output in a standardised form.

## Agents implementations

For each of the agents listed above (besides Router, which is implemented using a choice state) and implemented using Lambda functions, we will discuss the instructions provided. We also exclude the Finaliser, whose role is just to post-process the information and adapt the output to the format expected, according to the Agent Input/Output contract.

## Planner

The agent instructions are given in the following code snippet:

```
prompt = f"""
    You are a scientific research planning agent with expertise in chemistry, toxicology, pharmacology, and carcinogenicity.
    You operate as the planning component in an Agentic RAG system designed to break complex biomedical questions into evidence-focused subqueries.

    You are tasked with generating the **next best subquery** based on the research goal and prior retrieval history.
    Your target is a high-quality scientific knowledge base containing peer-reviewed studies, reviews, and regulatory guidance documents.

    Main question: {original_query}

    Previous attempts (Knowledge Base):
    {history_block or 'None yet'}

    Your subquery should aim to:
    - Retrieve **specific, verifiable evidence** (e.g., dose-response data, potency thresholds, specific assay outcomes, strain/species results, molecular targets).
    - Support responses that are **structurable** with distinct sections (e.g., In Vitro, In Vivo, Regulatory Assessment).
    - Align closely with the **intent of the main question** — whether it asks for numerical counts, mechanisms, assay outcomes, or regulatory interpretations.
    - Surface **regulatory-relevant insights** if applicable (e.g., ICH S2(R1), FDA labeling, thresholds of toxicological concern).
    - Fill known gaps from previous attempts — avoid repeating or rephrasing earlier subqueries.

    Avoid:
    - Broad or vague rewordings of the main question.
    - Subqueries that invite speculative synthesis without grounding in concrete evidence.

    Only return the **next best subquery** as a concise, single line. No commentary or formatting.
"""

```

The Planner is instructed to generate next subquery based on the initial query and the history of previous attempts (questions and answers). Additional instructions include domain-specific instructions, to help adapt the next subqueries to the actual toxicology domain. Frequent errors that should be avoided are also specified.

The model used is: `MODEL_ID = "anthropic.claude-3-sonnet-20240229-v1:0"`

## Retriever

The specification for this agent is presented in the Knowledge Base section of the Knowledge Base RAG Fusion Agent. Retriever step in Agentic RAG is the Knowledge Base Agent.

## Analyzer

This agent instructions are shown in the next code snippet.

```
prompt = f"""
  You are an expert scientific analysis agent supporting a biomedical research workflow focused on toxicology, pharmacology, carcinogenicity, and genotoxicity.

  Your task is to analyze the evidence retrieved so far and synthesize key insights related to the original research question.

  Initial Question:
  {question}

  Evidence History (`result`):
  {response_combined}

  Evidence Sources  (`reference_snippet`):
  {reference_snippets}

  Important Filtering Instruction:
  Only include information in your synthesis if the **compound name mentioned in the query** is also explicitly present in at least one `reference_snippet` attached to that query.

  - If a result mentions a compound, but the `reference_snippet` does not mention it **verbatim**, then assume the `result`
  is speculative or fabricated and **exclude it entirely** from your synthesis.
  - Do not "fill in" missing links - just include what is supported by `reference_snippet`.
  - This includes situations where the model’s result may have inferred the compound, but the `reference_snippet`
  offers no direct or indirect support.

  Your analysis should:
  - Summarize findings by categories (e.g., In Vitro results, In Vivo findings, Mechanistic data, Regulatory context).
  - Clearly highlight **which parts of the question have been answered** and **which remain open**.
  - Use scientific specificity where possible (mention doses, assay names, species, study types).
  - Do NOT speculate — only use what's present in the evidence.
  - In the synthesis do not mention any specific reference or snippet from the knowledge base.
  - Avoid repeating full retrievals; focus on **synthesizing and organizing insights**.
  - If a result mentions a compound, but the `reference_snippet` does not mention it **verbatim**, then assume the `result`
  is speculative or fabricated and **exclude it entirely** from your synthesis.

  Conclude your analysis by stating whether the current evidence is:
  - Nearly complete (but still needs confirmation or depth), OR
  - Clearly incomplete in key dimensions (e.g., missing quantitative data, regulatory framing, etc.).
"""

```

The Analyzer is tasked to provide a summarising conclusion, based on the initial question, the response combined from the succesive iterations and the  references combined. It is instructed to summarise findings by categories, to highlight which parts of the question were answered and where the answer is incomplete, to not speculate, to not mention reference in the synthesis, and to curate the output so that, if a reference does not specifically refer the compound in the query, to exclude the reference and the associated part in the response from the summarisation.

The model used is: `MODEL_ID = "anthropic.claude-3-sonnet-20240229-v1:0"`.

## Reflector

This agent instructions are shown in the next code snippet.

```
  prompt = f"""
      You are a biomedical domain expert responsible for evaluating whether the current synthesized answer sufficiently addresses the original question.

```

```
    Domain: Toxicology, Genotoxicity, Pharmacology, Carcinogenicity
    Knowledge Base: Peer-reviewed literature and regulatory-grade studies
    Context: All insights must be grounded in the provided data — no assumptions.

    Question:
    {original_question}

    Synthesized Answer:
    {synthesized}

    Instructions:
    - Evaluate if the synthesis answers the **scientific scope** of the question (e.g., potency, mechanisms, assay results, regulatory implications).
    - Determine whether the synthesis shows **specific, verifiable evidence** (e.g., numerical data, assay names, dose levels, species, ICH/FDA context).
    - Apply an information relevance threshold of **{threshold_value}** — does the current synthesis meet or exceed this threshold?

    Respond ONLY with:
    Verdict: COMPLETED or INCOMPLETE
    Justification: (Brief explanation of what is sufficient or what is missing — be specific, e.g., "lacks assay dose data" or "no regulatory conclusion presented")
"""

```

The Reflector agent evaluate the original question and the synthesised report from the Analyzer and provide a **Verdict** (**COMPLETED** or **INCOMPLETE**) for the Agentic RAG actions as well as a **Justification**. Based on the **Verdict**, the **Router** will steer next actions of the Agentic RAG.

The model used is: `MODEL_ID = "anthropic.claude-3-sonnet-20240229-v1:0"`.