

Building an Multimodal Knowledge Assistant

Jerry Liu

September 23, 2024



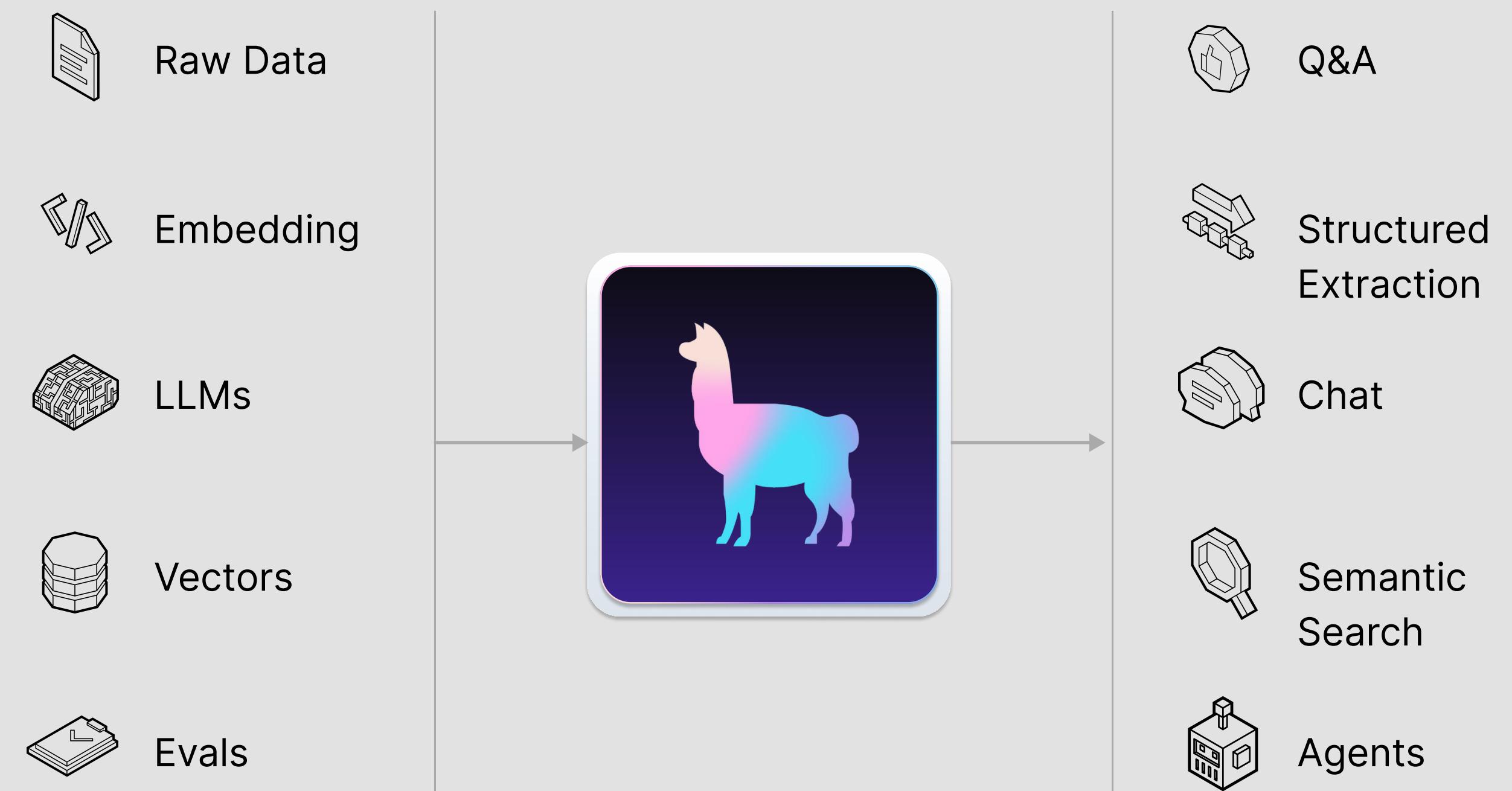
Llamaindex: Build Production LLM Apps over Enterprise Data

Llamaindex helps any developer build **context-augmented LLM apps from prototype to production.**

Open-Source: Leading developer toolkit for building production LLM apps over data.
Docs: <https://docs.llamaindex.ai/>
Repo: https://github.com/run-llama/llama_index

LlamaCloud: A centralized knowledge interface for your production LLM application.

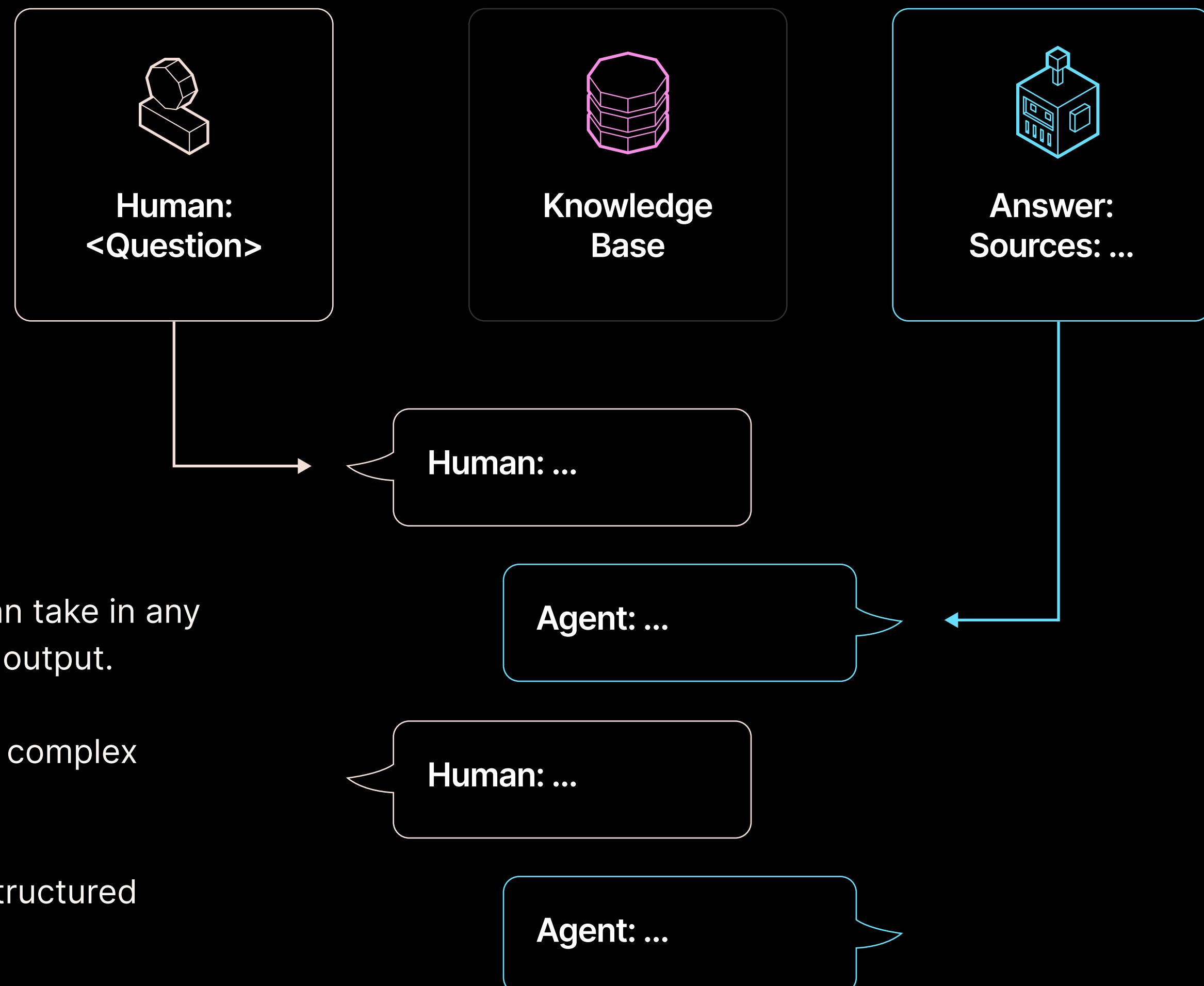
Link: <https://cloud.llamaindex.ai/>



GUNDERSON DETTMER



Building a Knowledge Assistant



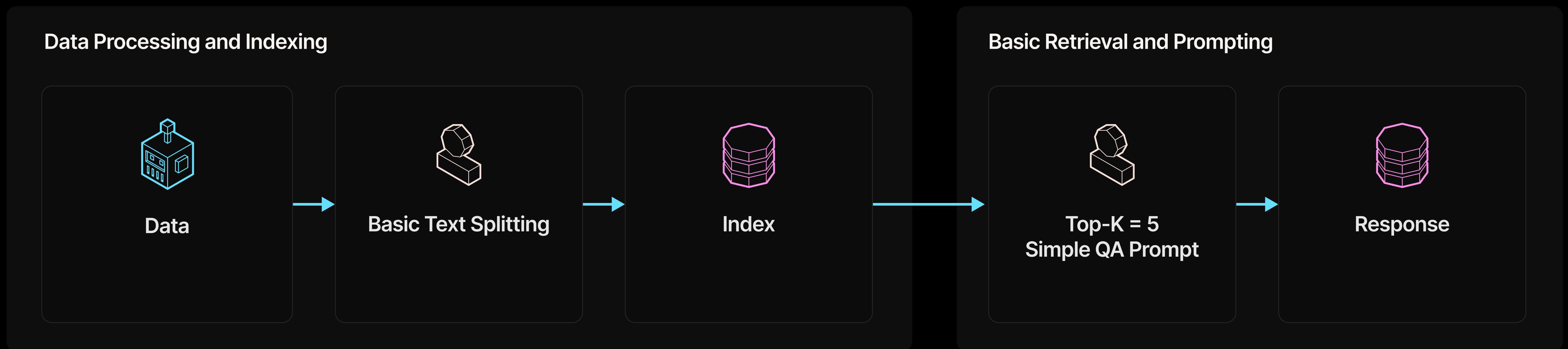
Goal: Build an interface that can take in any task as input and give back an output.

Input forms: simple questions, complex questions, research tasks

Output forms: short answer, structured output, research report

Knowledge Assistant with Basic RAG

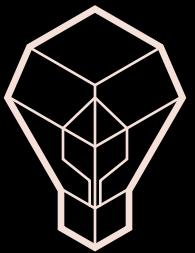
- ⚠ Naive data processing, primitive retrieval interface
- ⚠ Poor query understanding/planning
- ⚠ No function calling or tool use
- ⚠ Stateless, no memory



Can we do more?

There's many questions/tasks that naive RAG can't give an answer to

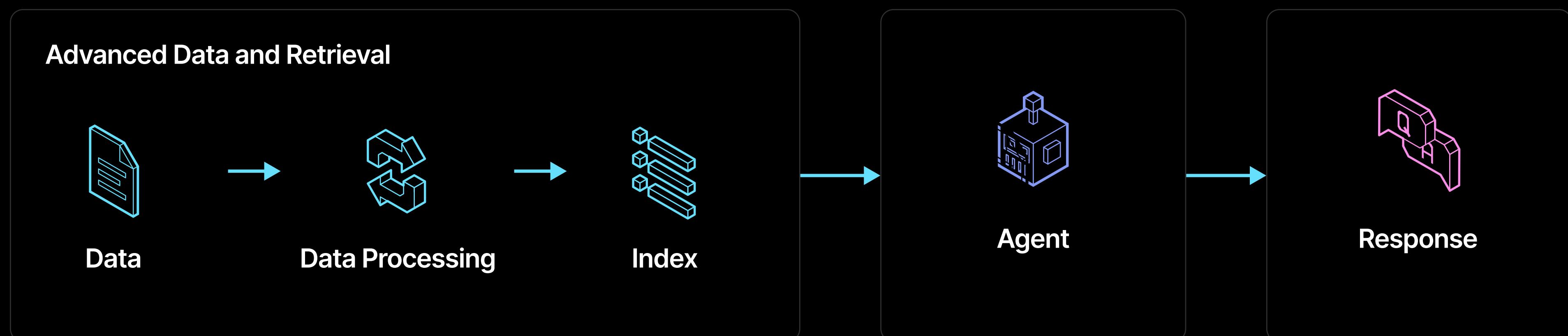
- 🚫 Hallucinations
- 🚫 Limited time savings
- 🚫 Limited decision-making enhancement



How do we aim to build
a production-ready
knowledge assistant?

A Better Knowledge Assistant

1. High-quality Multimodal RAG
2. Complex output generation
3. Agentic reasoning over complex inputs
4. Towards a scalable, full-stack application



A Better Knowledge Assistant

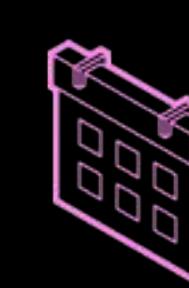
1. High-quality Multimodal RAG
2. **Complex output generation**
3. Agentic reasoning over complex inputs
4. Towards a scalable, full-stack application



Report
Generation



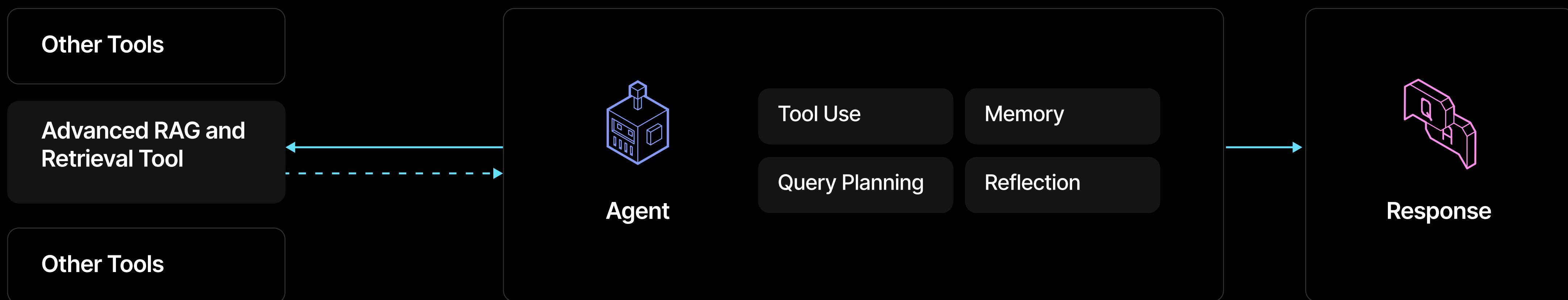
Data Analysis



Action-Taking

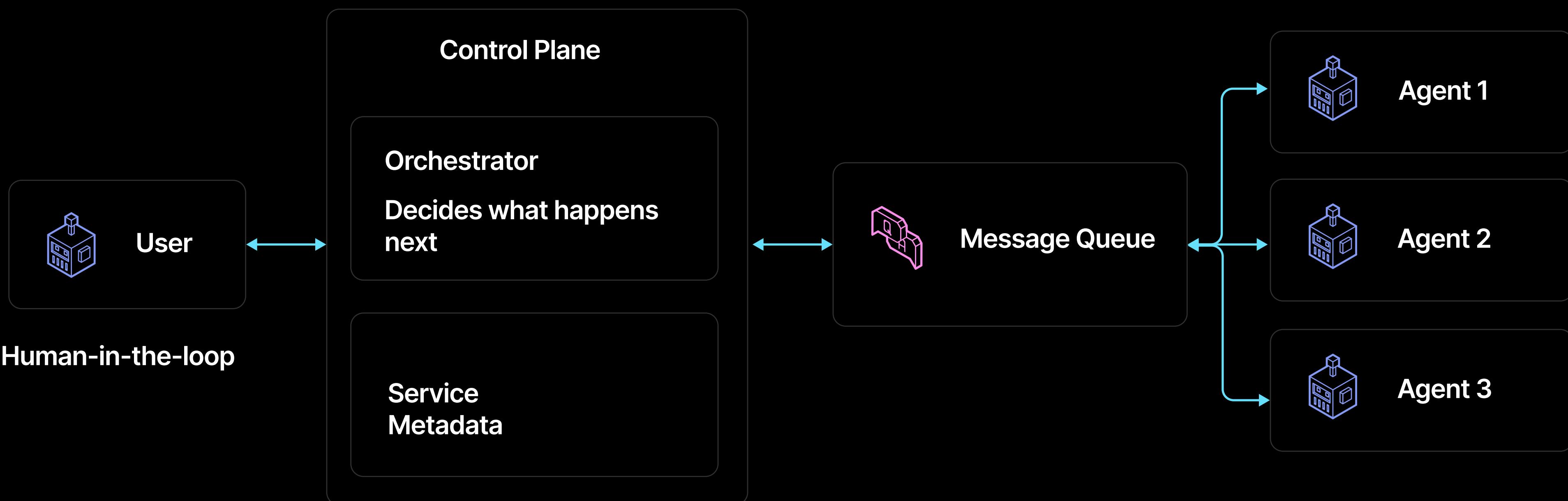
A Better Knowledge Assistant

1. High-quality Multimodal RAG
2. Complex output generation
3. **Agentic reasoning over complex inputs**
4. Towards a scalable, full-stack application



A Better Knowledge Assistant

1. High-quality Multimodal RAG
2. Complex output generation
3. Agentic reasoning over complex inputs
4. **Towards a scalable, full-stack application**

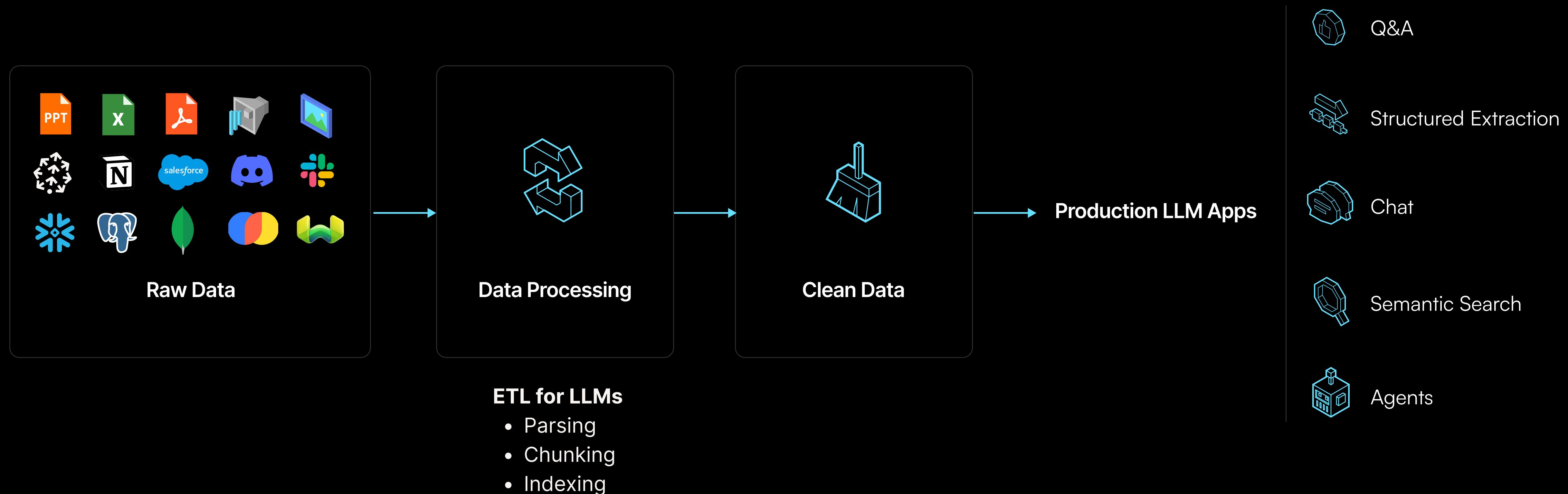


Setting up Multimodal RAG

Any LLM App is only as Good as your Data

Garbage in = garbage out

Good data quality is a **necessary** component of any production LLM app.



Case Study: Complex Documents

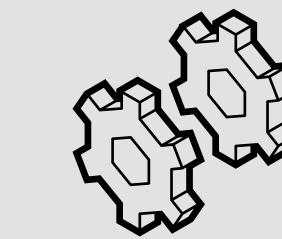
A lot of documents can be classified as **complex**:

- Embedded Tables, Charts, Images
- Irregular Layouts
- Headers/Footers

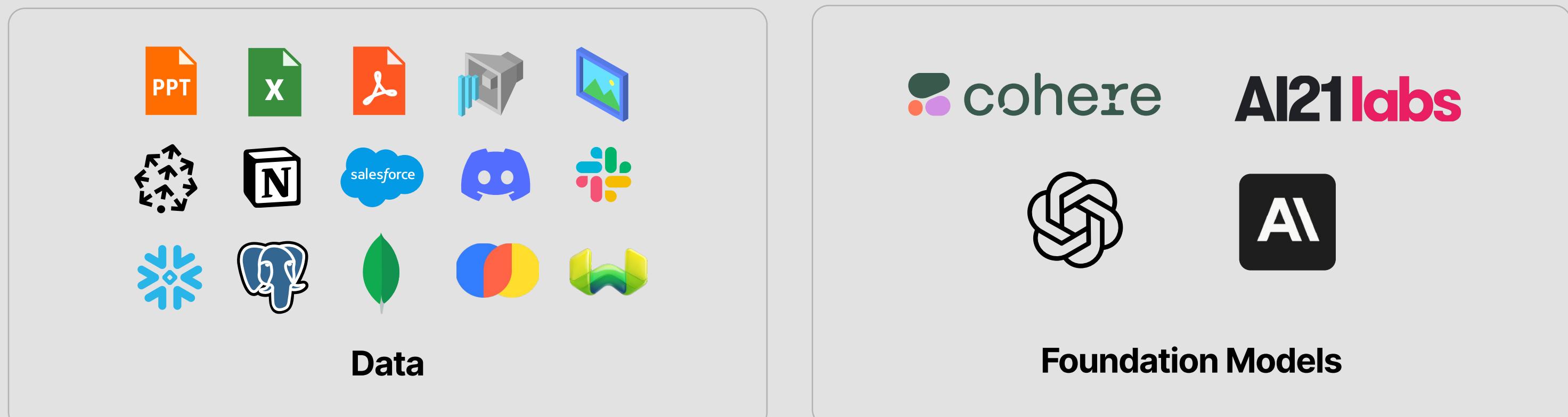
Users want to ask research questions over this data:

- Simple pointed questions
- Multi-document comparisons
- Research tasks

Building a production-ready knowledge assistants over this complex data is **challenging**.



Developers



An LLM-Native Document Parser

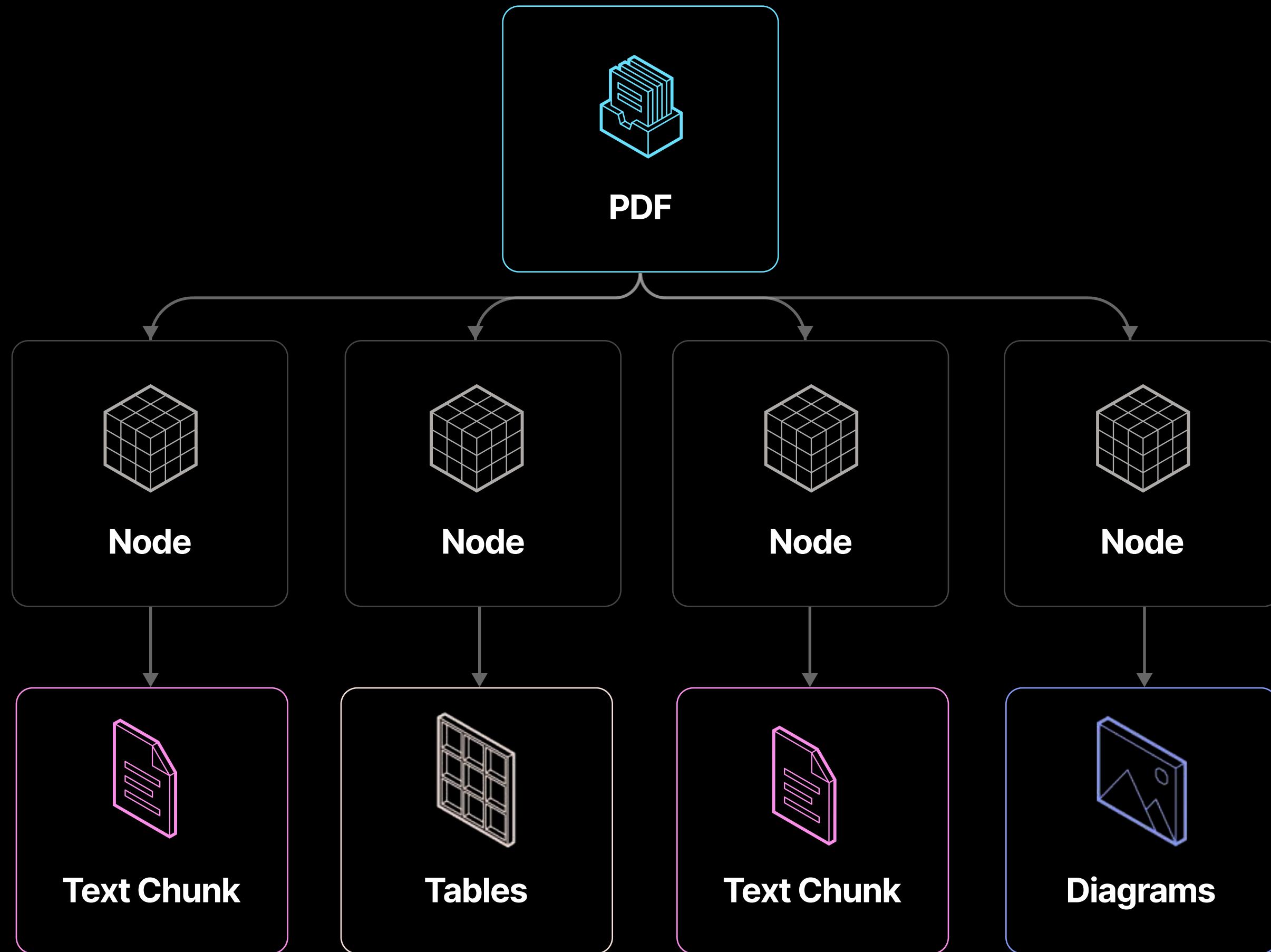
An ideal **GenAI-native parser** can structure complex document data for any downstream use case.

Requirements

- Parse tables accurately into text and semi-structured representations
- Parse text into semantically coherent chunks
- Extract visual elements (images/diagrams/charts) into structured formats *and* return image chunks.
- Automated metadata extraction

Non-Requirements

- Extract detailed JSONs for every element
- Extract bounding boxes



LlamaParse

"As an AI Applied Data Scientist who was granted one of the first ML patents in the U.S., and who is building cutting-edge AI capabilities at one of the world's largest Private Equity Funds, I can confidently say that LlamaParse from LlamalIndex is currently the best technology I have seen for parsing complex document structures for Enterprise RAG pipelines. Its ability to preserve nested tables, extract challenging spatial layouts, and images is key to maintaining data integrity in advanced RAG and agentic model building."

Dean Barr, Applied AI Lead at Carlyle



LlamaParse

Advanced document parser specifically for reducing LLM hallucinations

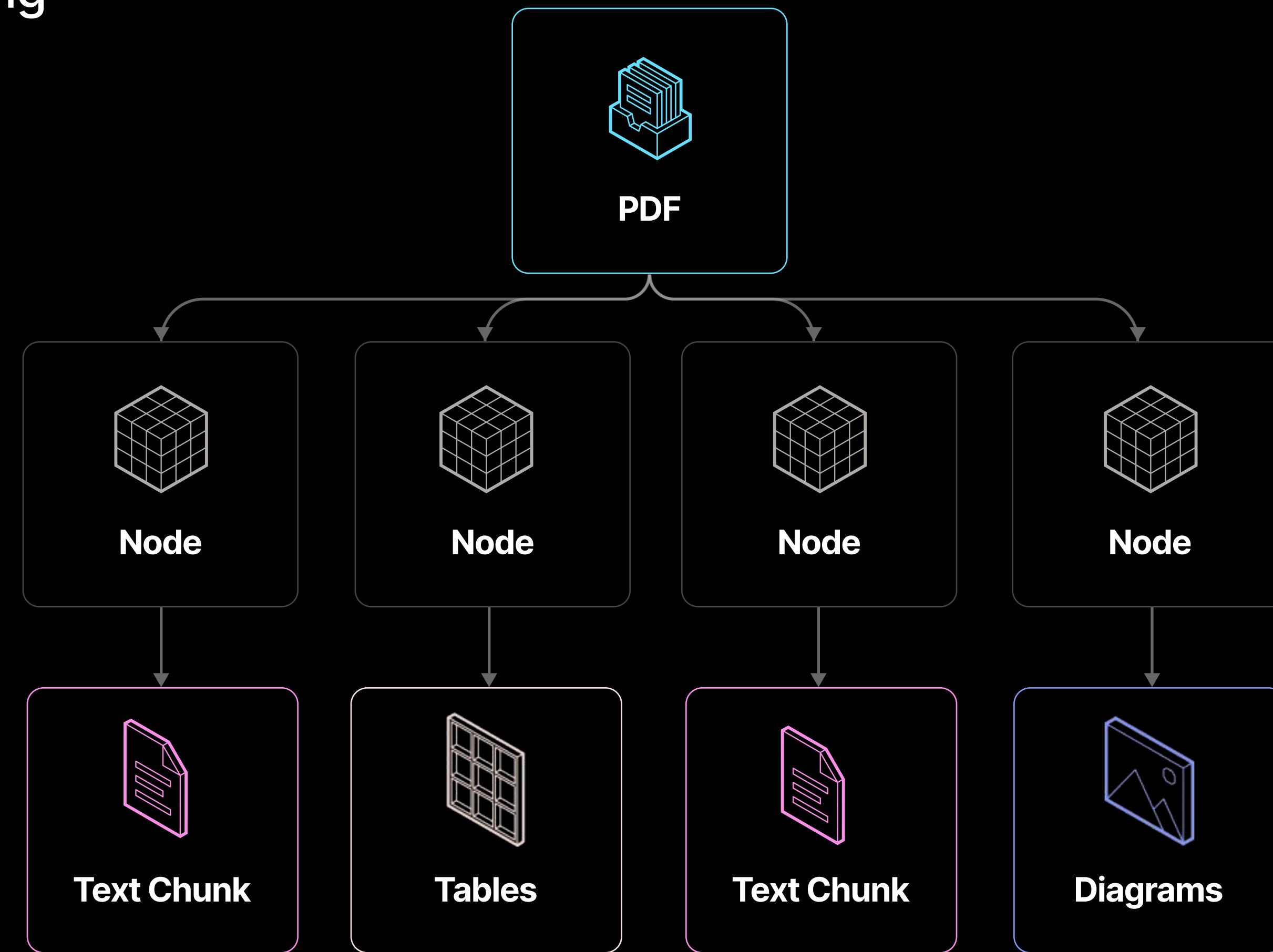
20k+
unique users

25M+
pages processed

Advanced Parsing + Advanced Indexing

You can combine parsing with **hierarchical indexing and retrieval** to model heterogeneous unstructured/tabular/multimodal data within a document.

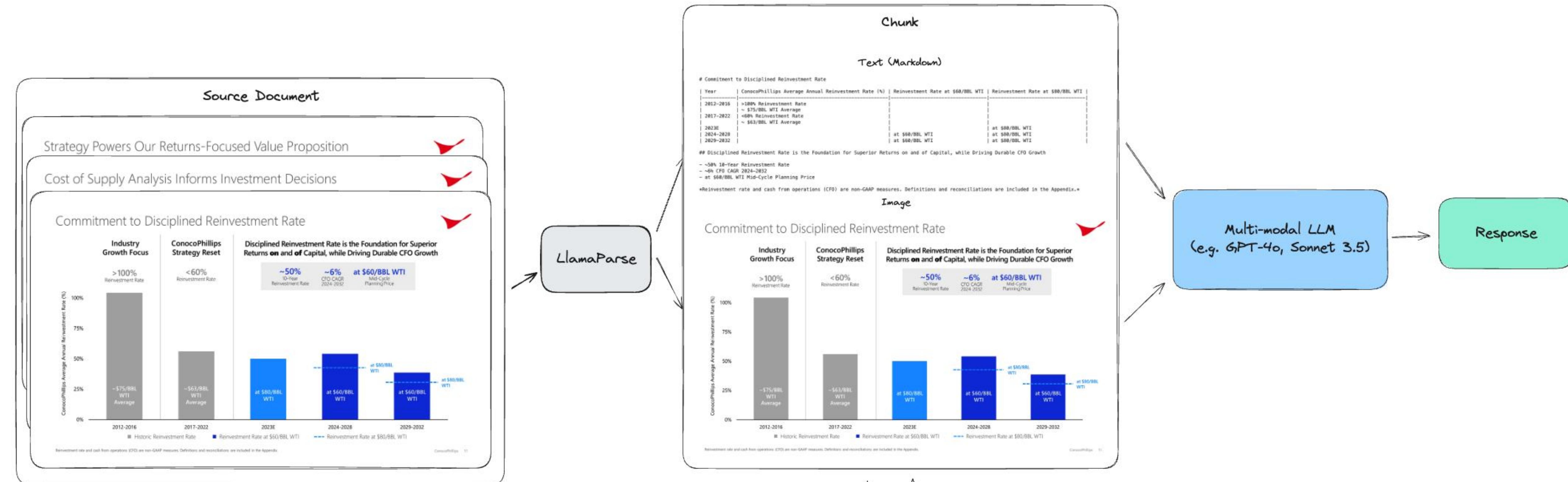
1. Parse documents into elements: text chunks, tables, images, and more.
2. For each element, extract **one or more** text representations that can be indexed.
3. Do **recursive retrieval**



Multimodal RAG Pipeline

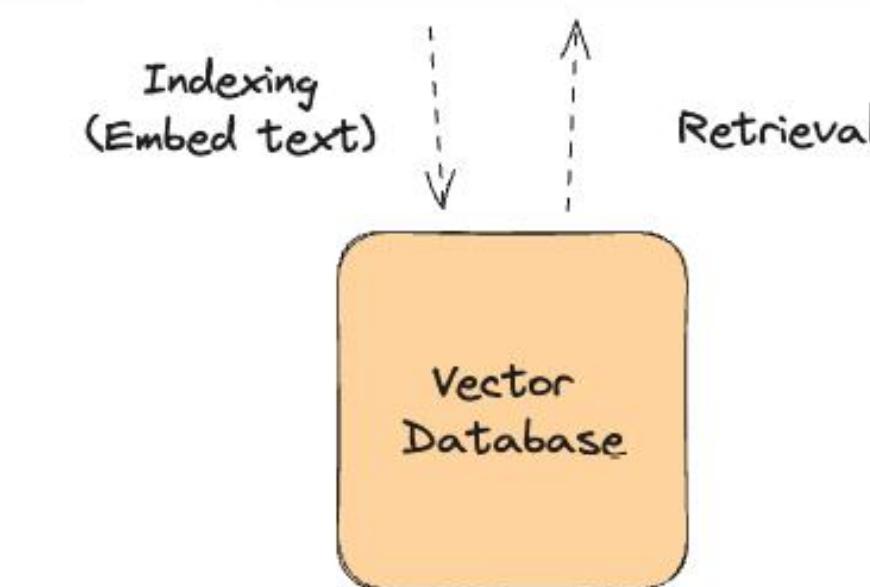
A true multimodal RAG pipeline stores both text and image chunks for use within a multi-modal LLM

Multi-modal RAG over a Slide Deck



Indexing

1. Parse document into text and image chunks with LlamaParse
2. Link each text chunk to image chunk through metadata
3. Embed and index text chunks



Retrieval

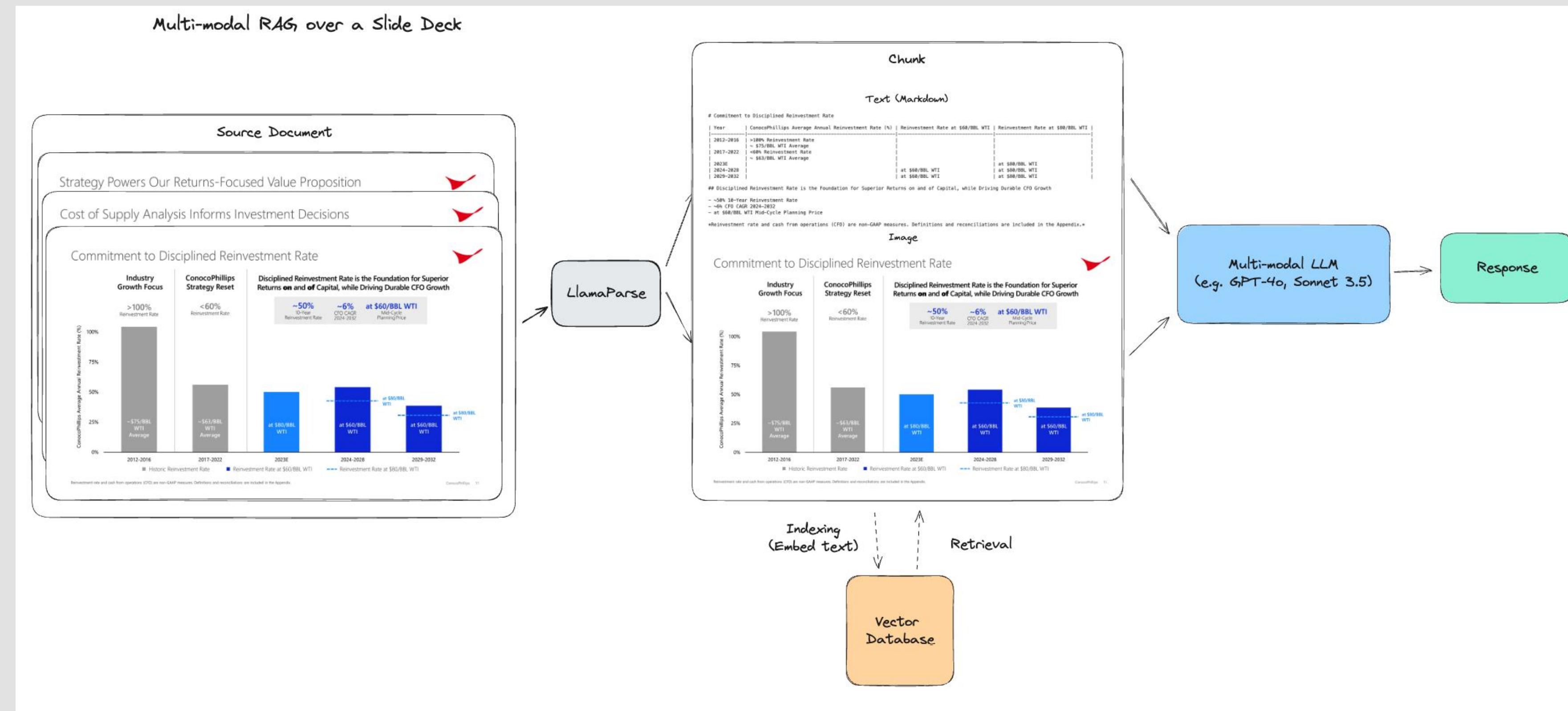
1. Retrieve text chunks by text embeddings
2. Feed in both text and image to multimodal LLM during synthesis.

Multimodal RAG Pipeline

Let's run through a demo example of building multimodal RAG over a complex slide deck!

The end result is you're able to ask questions over visual data in the document.

https://github.com/run-llama/llama_parse/blob/main/examples/multimodal/multimodal_rag_slide_deck.ipynb



LlamaCloud: An Enterprise RAG Platform

A production-ready RAG platform that allows developers to easily connect their unstructured data sources to LLM agent systems.

Instant Time-to-Value for building knowledge assistants

- Out-of-the-box advanced RAG capabilities
- Free up developer time to rapidly iterate on higher-level agent use cases

State-of-the-Performance leads to increased

satisfaction and reduced compliance risk

Reduced maintenance cost once application is deployed

Enterprise-ready security like access controls

Signup: <https://cloud.llamaindex.ai/>

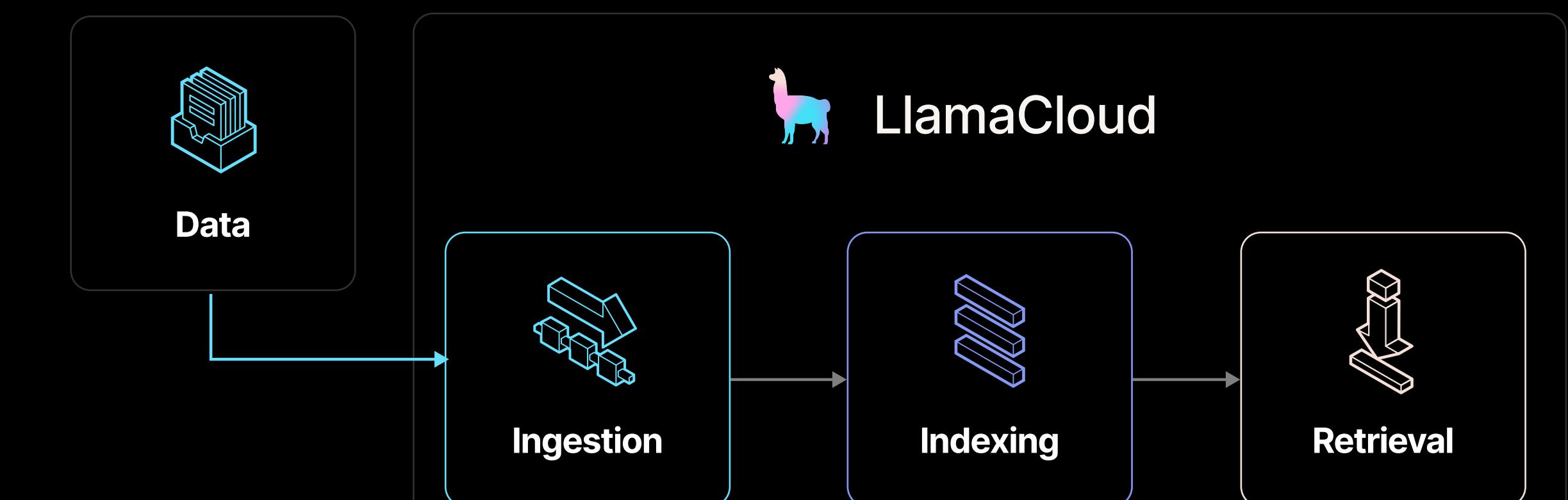
The screenshot shows the LlamaCloud web interface. At the top right, there are 'Rerun' and 'Delete' buttons. The main area displays a project titled 'uber_and_lyft'. On the left sidebar, there are sections for 'PROJECT' (test_projects_03_28_2024), 'PROJECT NAVIGATION' (Parse, Index, Evals), and 'RESOURCES' (API Key). The 'YOUR PLAN' section shows a 'Free Plan' with an 'Upgrade Plan' button. The 'USAGE' section indicates 0 / 1000 pages per day (750 Pages per file). The 'Retrieval Endpoint' section shows the Endpoint URL: <https://api.cloud.llamaindex.ai/api/pipeline/edf2f149-b8b9-4b5b-9806-b5d4a182fe86/retrieve>. Below it is the 'Endpoint API' section with a code snippet:

```
from llama_index.indices.managed.llama_cloud import LlamaCloudIndex
# pip install llama-index-indices-llama-cloud

index = LlamaCloudIndex(
    name="uber_and_lyft",
    project_name="test_projects_03_28_2024",
    api_key="llx-..."
)

nodes = index.as_retriever().retrieve(query)
response = index.as_query_engine().query(query)
```

The 'Ingestion Configuration' section includes 'Transformations' (SentenceSplitter, OpenAIEmbedding).



E2E Multimodal RAG Capabilities

Setup multimodal indexing and retrieval in minutes

Signup here: <https://cloud.llamaindex.ai/>

Node 1 - Page Screenshot

Our Differentiated Portfolio: Deep, Durable and Diverse

~20 BBOE of Resource
Under \$40/BBL Cost of Supply

~\$32/BBL
Average Cost of Supply

WTI Cost of Supply (\$/BBL)

Resource (BBOE)

Legend: Lower 48, Canada, Alaska, EMENA, Asia Pacific

Diverse Production Base
10-Year Plan Cumulative Production (BBOE)

Lower 48

Alaska

GKA

GWA

GPA

WNS

EMENA

Norway

Qatar

Libya

Asia Pacific

Canada

Montney

Eagle Ford

Bakken

Other

APLNG

Malaysia

China

Surmont

Costs assume a mid-cycle price environment of \$40/BBL WTI.

File: 2023-conocophillips-aim-presentation (1).pdf

File ID: d199c200-8ccb-4fa7-aab1-18c185afa389

Page: 14

Projects > llamacloud_demo > multimodal_test > playground

multimodal_test

Sync Edit Copy Delete

Overview Playground Data Sources

Chat Retrieval

Chat Options

tell me about the diverse geographies which represent the production bases

Show events >

ConocoPhillips has a diverse and extensive production base spread across various geographic regions. Here's a detailed breakdown of the regions where ConocoPhillips operates:

Lower 48 States (USA) - This includes significant operations in the Permian Basin, which is further divided into the Delaware Basin and the Midland Basin. The Delaware Basin, for instance, has shown improved performance from 2019 to 2022, and the company has a vast inventory in this area.

Alaska (USA) - ConocoPhillips has a strong presence in Alaska, where it engages in conventional oil production. The company's operations in this region are expected to leverage existing infrastructure for future growth.

Canada - While specific production figures are not detailed in the documents, Canada is listed as one of the regions in ConocoPhillips' diverse production base.

Norway - In Norway, ConocoPhillips has a production of 115 MBOED, with developments like the four subsea tie backs in the Greater Ekofisk Area, which are on track for onstream in 2024. The license for this area has been extended through 2048.

Libya - The company has increased its working interest to approximately 20% in the Waha Concession, with a production of 50 MBOED.

Malaysia - ConocoPhillips reports a production of 40 MBOED

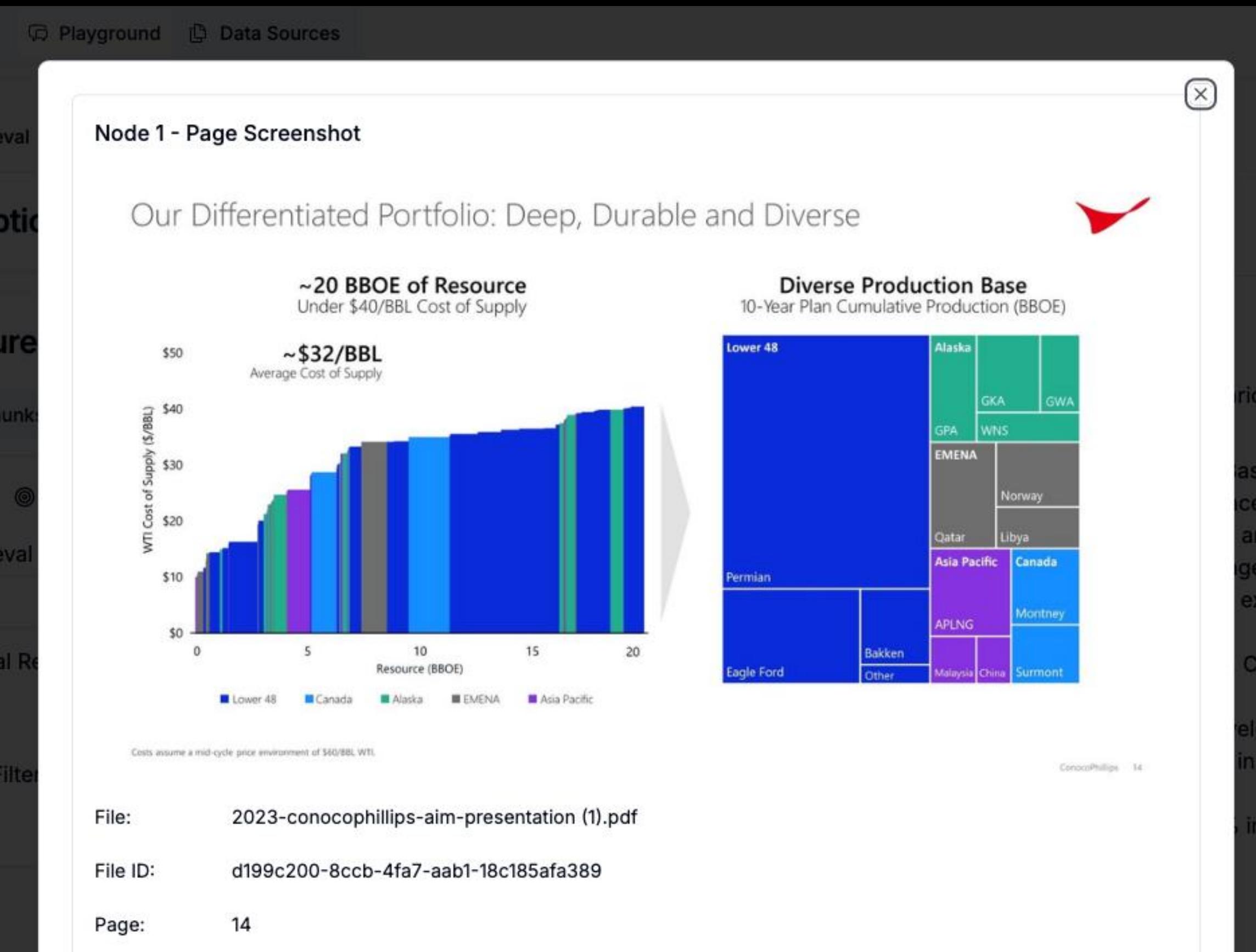
Sources: 1 2 3

Page Screenshots: 1 2 3 4 5 6

Regenerate

Type a message

Send message



Projects > llamacloud_demo > multimodal_test > playground

multimodal_test

Sync Edit Copy Delete

Overview Playground Data Sources

Chat Retrieval

Chat Options

Configure Retrieval

- Chunks Files
- Fast Accurate Advanced
- Base Retrieval 30
- Multi-Modal Retrieval
- Metadata Filtering

tell me about the diverse geographies which represent the production bases

Show events >

ConocoPhillips has a diverse and extensive production base spread across various geographic regions. Here's a detailed breakdown of the regions where ConocoPhillips operates:

- Lower 48 States (USA)** - This includes significant operations in the Permian Basin, which is further divided into the Delaware Basin and the Midland Basin. The Delaware Basin, for instance, has shown improved performance from 2019 to 2022, and the company has a vast inventory in this area.
- Alaska (USA)** - ConocoPhillips has a strong presence in Alaska, where it engages in conventional oil production. The company's operations in this region are expected to leverage existing infrastructure for future growth.
- Canada** - While specific production figures are not detailed in the documents, Canada is listed as one of the regions in ConocoPhillips' diverse production base.
- Norway** - In Norway, ConocoPhillips has a production of 115 MBOED, with developments like the four subsea tie backs in the Greater Ekofisk Area, which are on track for onstream in 2024. The license for this area has been extended through 2048.
- Libya** - The company has increased its working interest to approximately 20% in the Waha Concession, with a production of 50 MBOED.
- Malaysia** - ConocoPhillips reports a production of 40 MBOED

Sources: 1 2 3

Page Screenshots: 1 2 3 4 5 6

Type a message

Regenerate

Report Generation

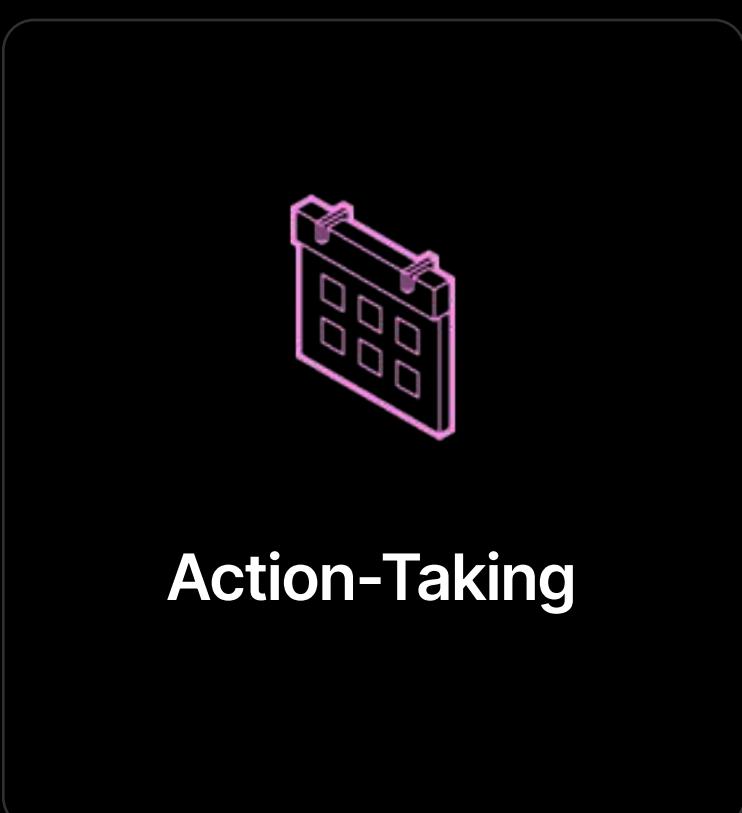
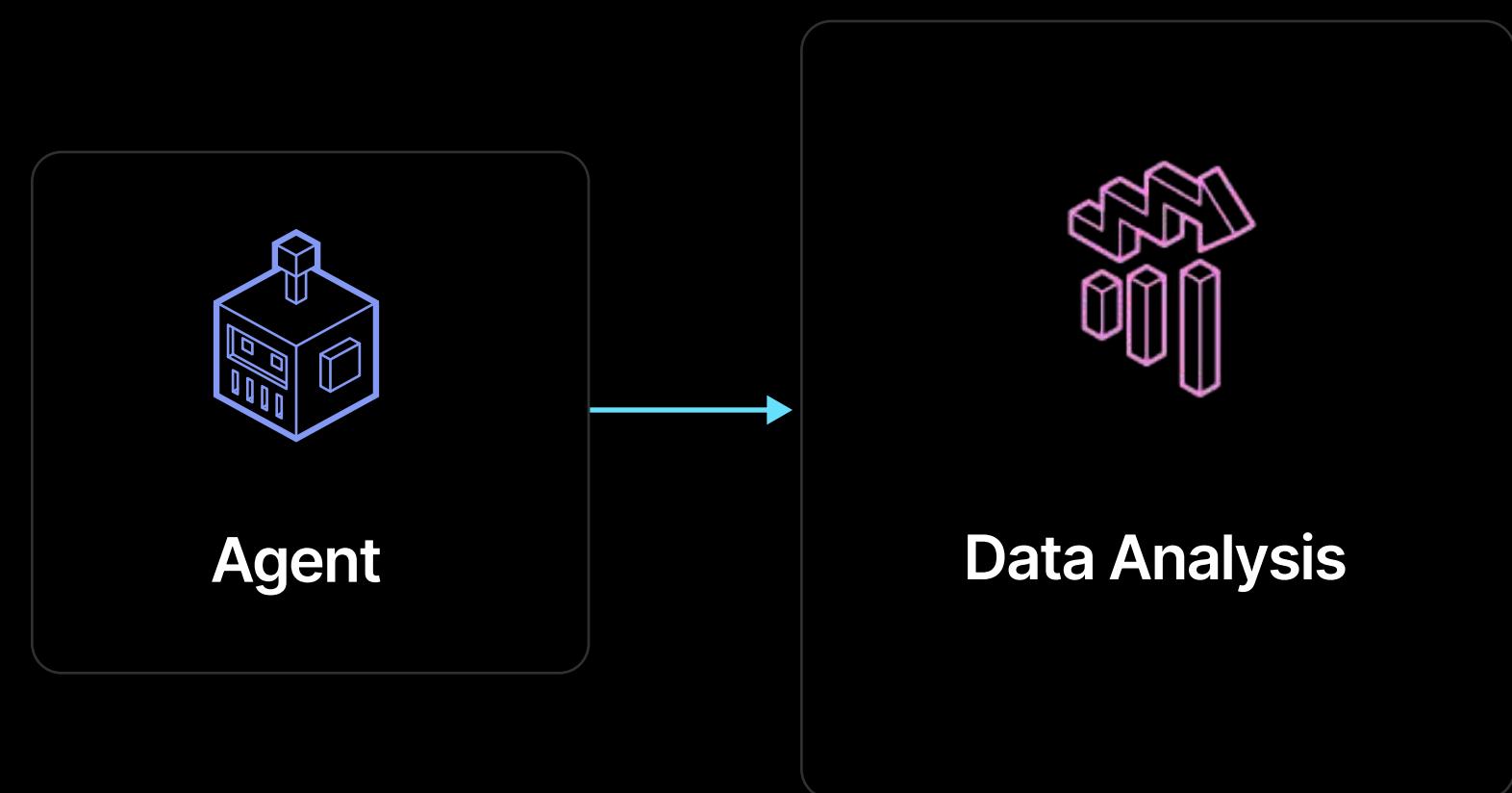
Automating Decision Making

Agents should have the capability to not only generate chatbot responses, but also

1. Produce knowledge work
2. Take actions

✓ Action-taking and Output Generation potentially lead to **much greater ROI** in terms of time savings and capability improvement

Solution💡: Structured Outputs and Function Calling



Multimodal Report Generation

Generate interleaving text-and-image responses with the help of **structured outputs**.

https://github.com/run-llama/llama_parse/blob/main/examples/multimodal/multimodal_report_generation.ipynb

Output Schema

```
class TextBlock(BaseModel):
    text: str

class ImageBlock(BaseModel):
    file_path: str

class ReportOutput(BaseModel):
    blocks: ListBlock | ImageBlock]
```

The financial performance of ConocoPhillips' Alaska/International segment and the Lower 48 segment can be summarized as follows:

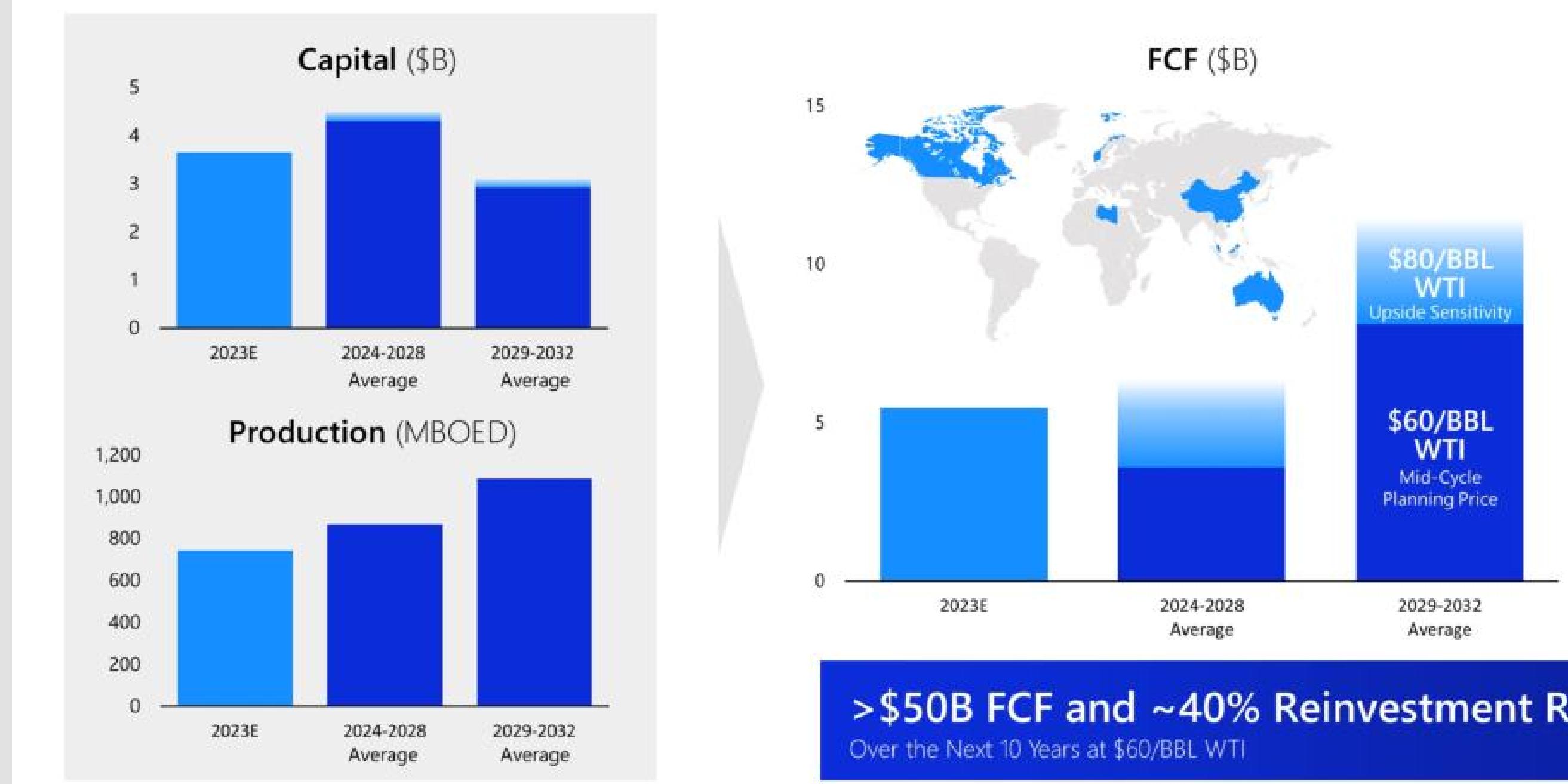
Alaska/International Segment

- **Capital Expenditures:** Expected to average *3.7 billion in 2023*, *4.4 billion from 2024-2028*, and *\$3.0 billion from 2029-2032*.
- **Production:** Projected to be around 750 MBOED in 2023, increasing to 870 MBOED on average from 2024-2028, and reaching 1080 MBOED on average from 2029-2032.
- **Free Cash Flow (FCF):** Estimated at *5.5 billion in 2023*, *averaging 6.5 billion from 2024-2028*, and *\$15.0 billion from 2029-2032*.
- **Key Projects:** Includes significant investments in LNG, Surmont, Montney, and conventional international assets.

Lower 48 Segment

- **Capital Expenditures:** Expected to be *6.3 billion in 2023*, *averaging 6.5 billion from 2024-2028*, and *\$8.1 billion from 2029-2032*.
- **Production:** Projected to be around 1050 MBOED in 2023, increasing to 1220 MBOED on average from 2024-2028, and reaching 1530 MBOED on average from 2029-2032.
- **Free Cash Flow (FCF):** Estimated at *7 billion in 2023*, *averaging 5.5 billion from 2024-2028*, and *\$8 billion from 2029-2032*.
- **Key Projects:** Focused on the Permian Basin, Eagle Ford, and Bakken, with significant investments in technology and emissions reductions.

Alaska and International: Our Unique Diversification Advantage



Agentic Reasoning over Complex Inputs

Complex Inputs

Naive RAG works well for pointed questions, but fails on more complex tasks.

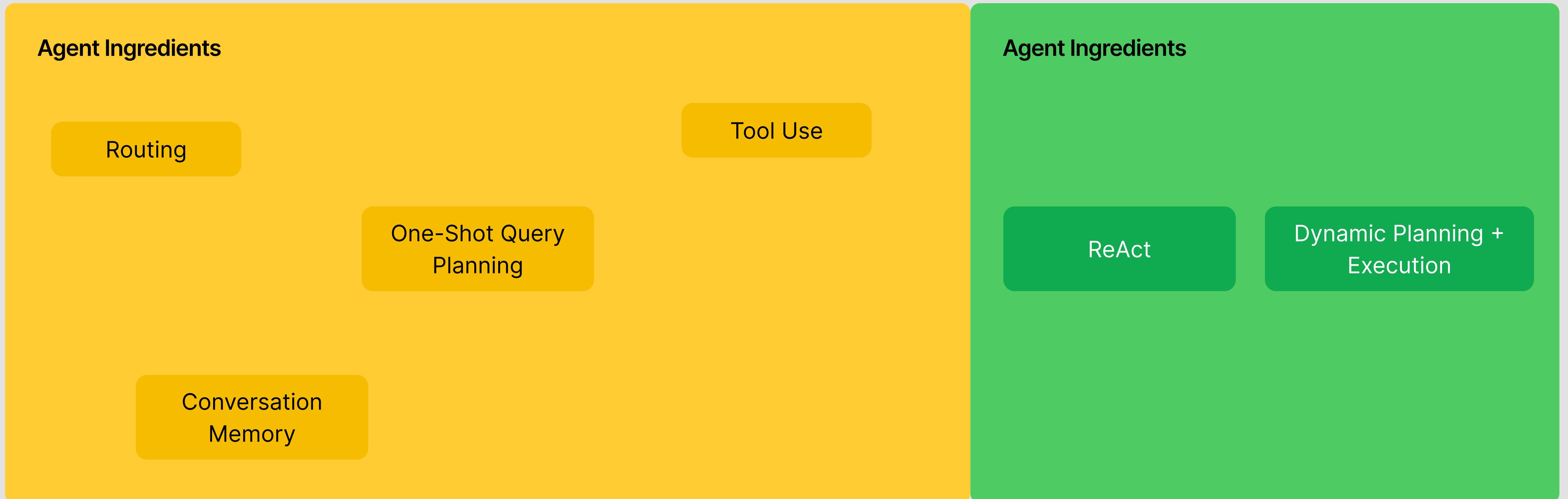
Summarization Questions: “Give me a summary of the entire <company> 10K annual report”

Comparison Questions: “Compare the open-source contributions of candidate A and candidate B”

Multi-part Questions: “Tell me about the pro-X arguments in article A, and tell me about the pro-Y arguments in article B, make a table based on our internal style guide, then generate your own conclusion based on these facts.”

Research Tasks: “I want to create a research survey on current supervised fine-tuning techniques. Can you help?”

From Simple to Advanced Agents



Simple
Lower Cost
Lower Latency

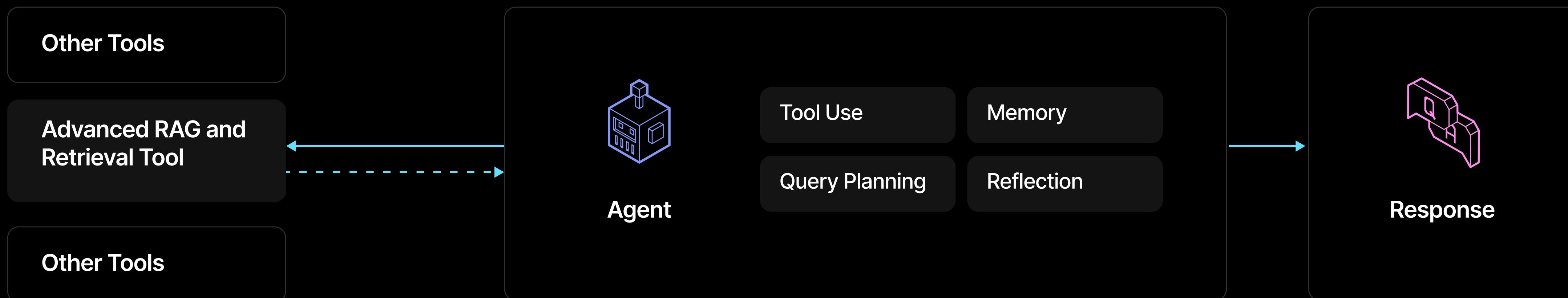
Advanced
Higher Cost
Higher Latency

Agentic RAG

Every data interface is a tool

Use agent reasoning loops (sequential, DAG, tree) to tackle complex tasks

End Result: Build personalized QA systems capable of handling complex questions!

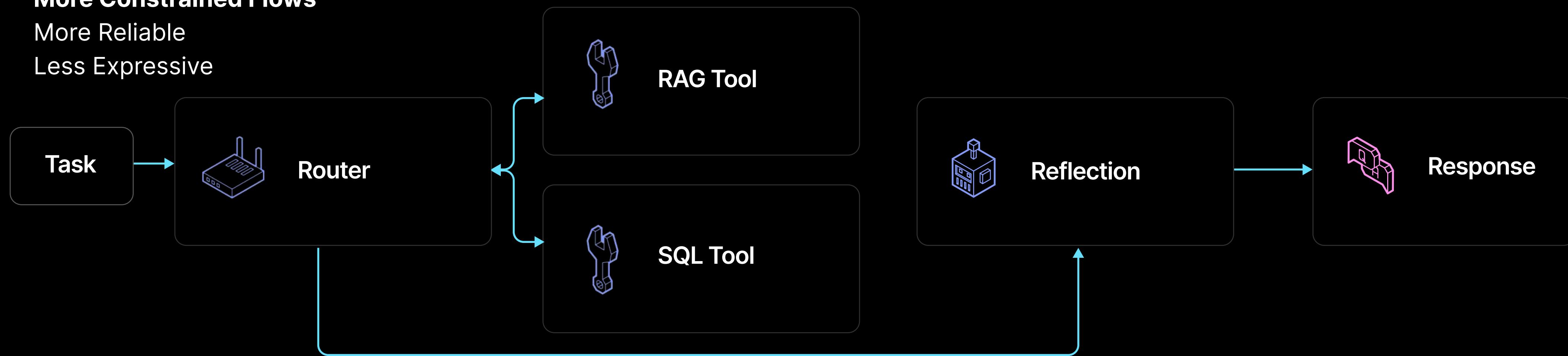


Unconstrained vs. Constrained Flows

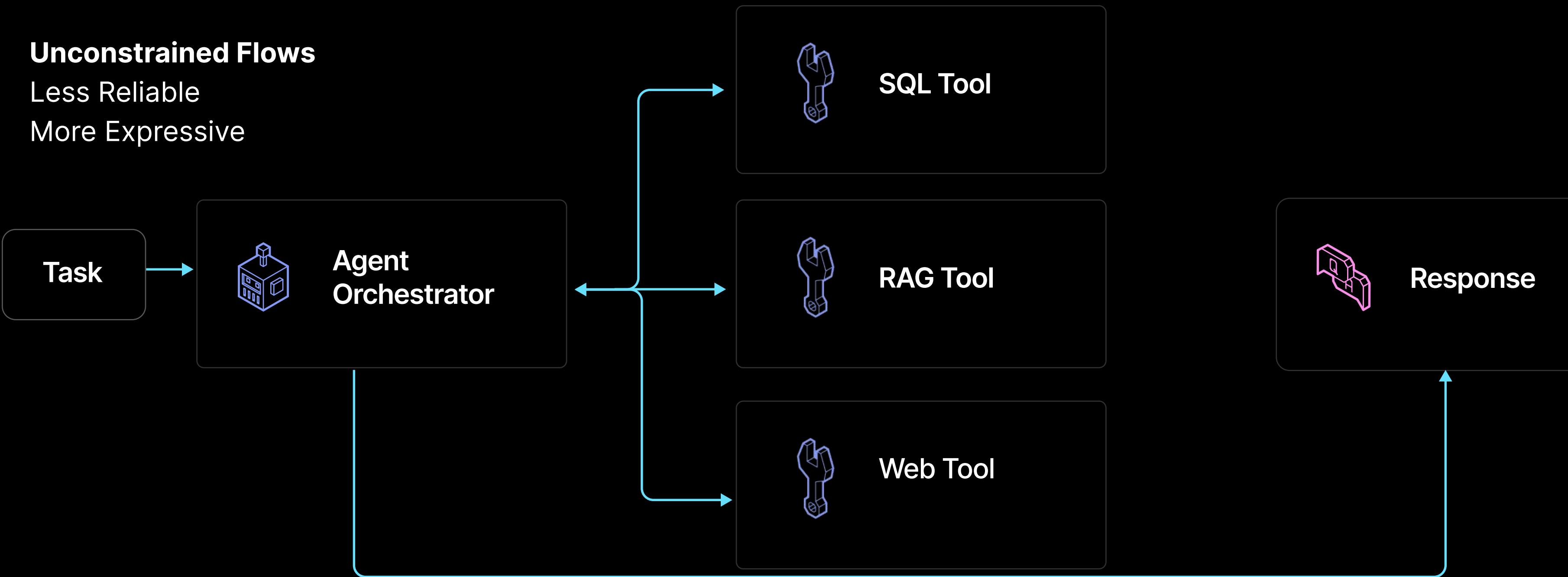
More Constrained Flows

More Reliable

Less Expressive



Unconstrained vs. Constrained Flows

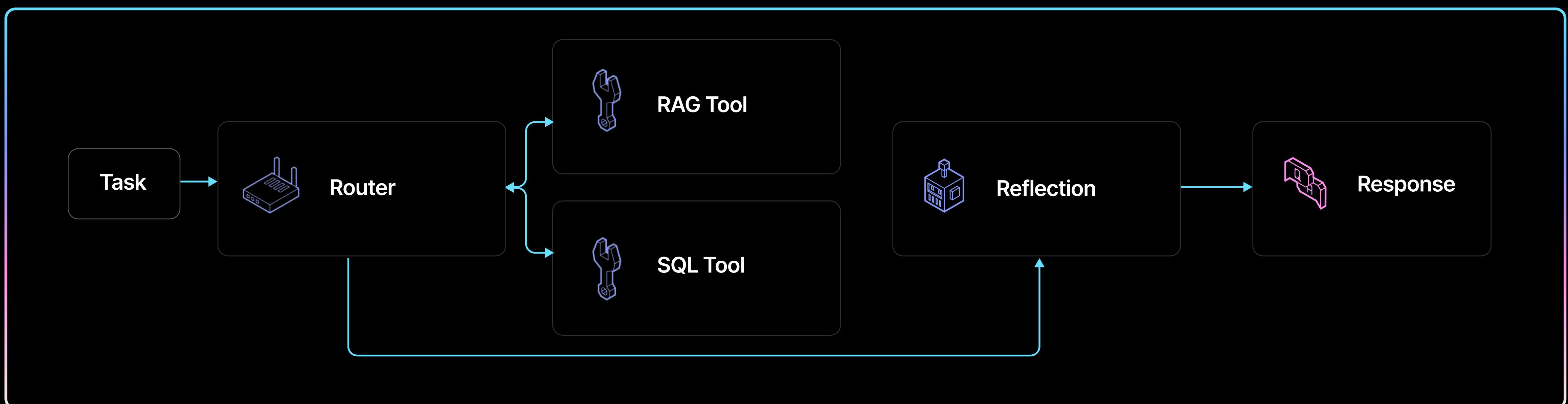


Agentic Orchestration Foundations

We believe an agent orchestration framework should have the following properties

- ✓ **Event-Driven:** Model each step as listening to input events and emitting output events
- ✓ **Composable:** Piece together granular workflows into higher-level workflows
- ✓ **Flexible:** Write logic through LLM calls or through plain Python
- ✓ **Code-first:** Express orchestration logic through code. Easy to read and easy to extend.
- ✓ **Debuggable and Observable:** Step through and observe states
- ✓ **Easily Deployable to Production:** Translate notebook code into services that run in production.

Llamaindex Workflows



Compared to Graph-based Approaches

Graph-based approaches (e.g. our deprecated Query Pipelines) can be cumbersome and non-Pythonic for complex agentic workflows.

```
def generate_response(context, query):
    prompt = f"Question: {query}\n\nContext: {context}\n\nAnswer:"
    response = llm.complete(prompt)
    return response.text

# Define the pipeline
pipeline = QueryPipeline()
pipeline.add_modules({
    "input": InputComponent(),
    "retriever": retriever,
    "reranker": reranker,
    "response_generator": FnComponent(fn=generate_response)
})

# Define the flow
pipeline.add_link("input", "retriever")
pipeline.add_link("retriever", "reranker")
pipeline.add_link("input", "response_generator", dest_key="query")
pipeline.add_link(
    "reranker", "response_generator", dest_key="context"
)

# Run the pipeline
response = pipeline.run("What is the capital of France?")
print(response)
```

- Orchestration logic baked into edges
- More lines of code, less readable
- Cumbersome to dynamically generate workflows based on runtime conditions

Compared to Graph-based Approaches

Graph-based approaches (e.g. our deprecated Query Pipelines) can be cumbersome and non-Pythonic for complex agentic workflows. Compared to query pipelines, our workflows are more readable, and easier to maintain/scale.

```
def generate_response(context, query):
    prompt = f"Question: {query}\n\nContext: {context}\n\nAnswer:"
    response = llm.complete(prompt)
    return response.text

# Define the pipeline
pipeline = QueryPipeline()
pipeline.add_modules({
    "input": InputComponent(),
    "retriever": retriever,
    "reranker": reranker,
    "response_generator": FnComponent(fn=generate_response)
})

# Define the flow
pipeline.add_link("input", "retriever")
pipeline.add_link("retriever", "reranker")
pipeline.add_link("input", "response_generator", dest_key="query")
pipeline.add_link(
    "reranker", "response_generator", dest_key="context"
)

# Run the pipeline
response = pipeline.run("What is the capital of France?")
print(response)
```

```
class RAGWorkflow(Workflow):
    def __init__(self):
        ...

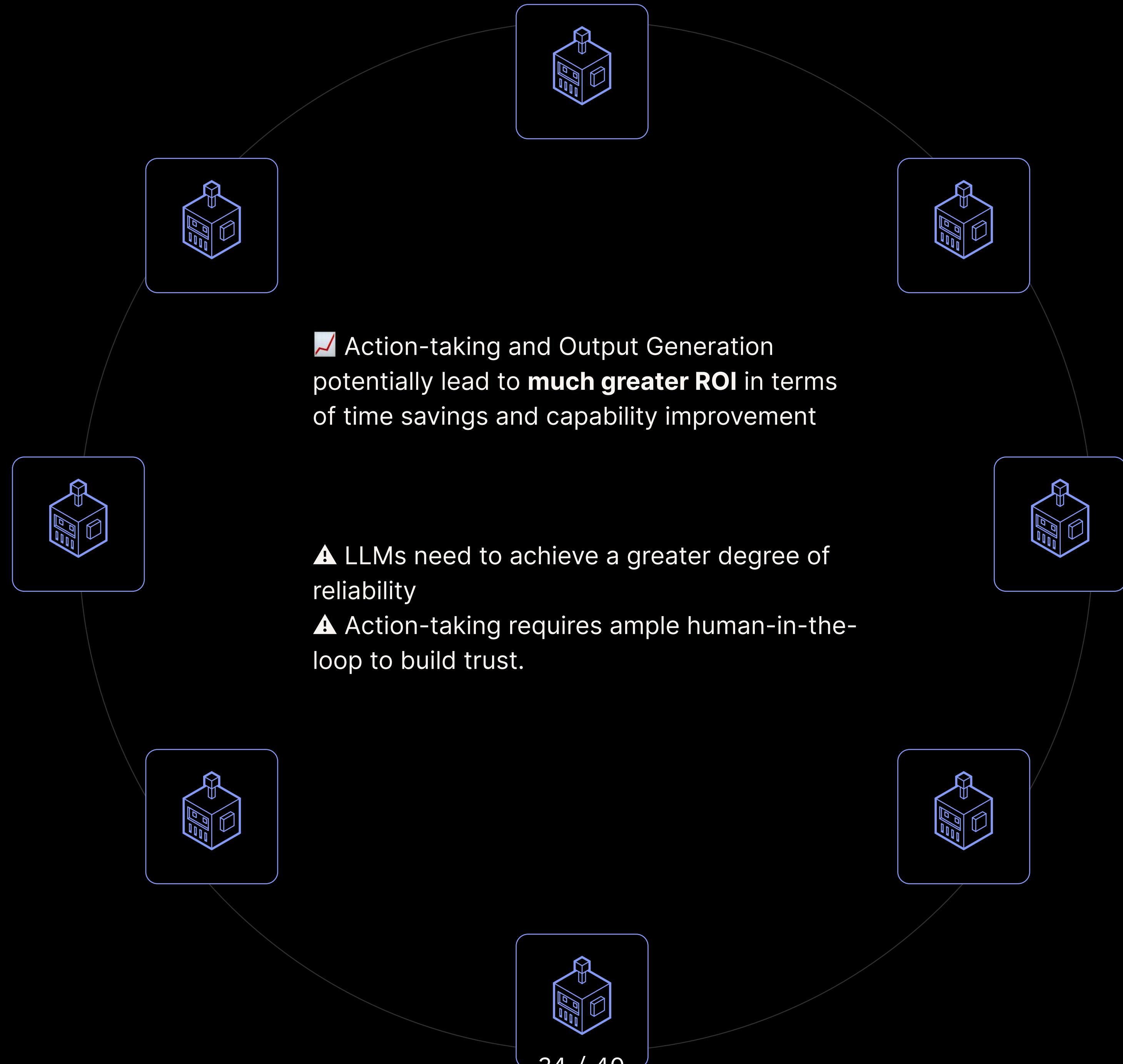
    @step
    async def retrieve(self, query: str):
        return self.retriever.retrieve(query)

    @step
    async def rerank(self, retrieved_nodes):
        return self.reranker.postprocess_nodes(retrieved_nodes)

    @step
    async def generate_response(self, query: str, context):
        prompt = f"Question: {query}\n\nContext:
{context}\n\nAnswer:"
        response = await self.llm.complete(prompt)
        return response.text

    @step
    async def run_workflow(self, query: str):
        retrieved_nodes = await self.retrieve(query)
        reranked_nodes = await self.rerank(retrieved_nodes)
        response = await self.generate_response(query,
[node.get_content() for node in reranked_nodes])
        return response
```

Benefits and Risks



Multimodal Report Generation

Generate interleaving text-and-image responses with the help of **structured outputs**.

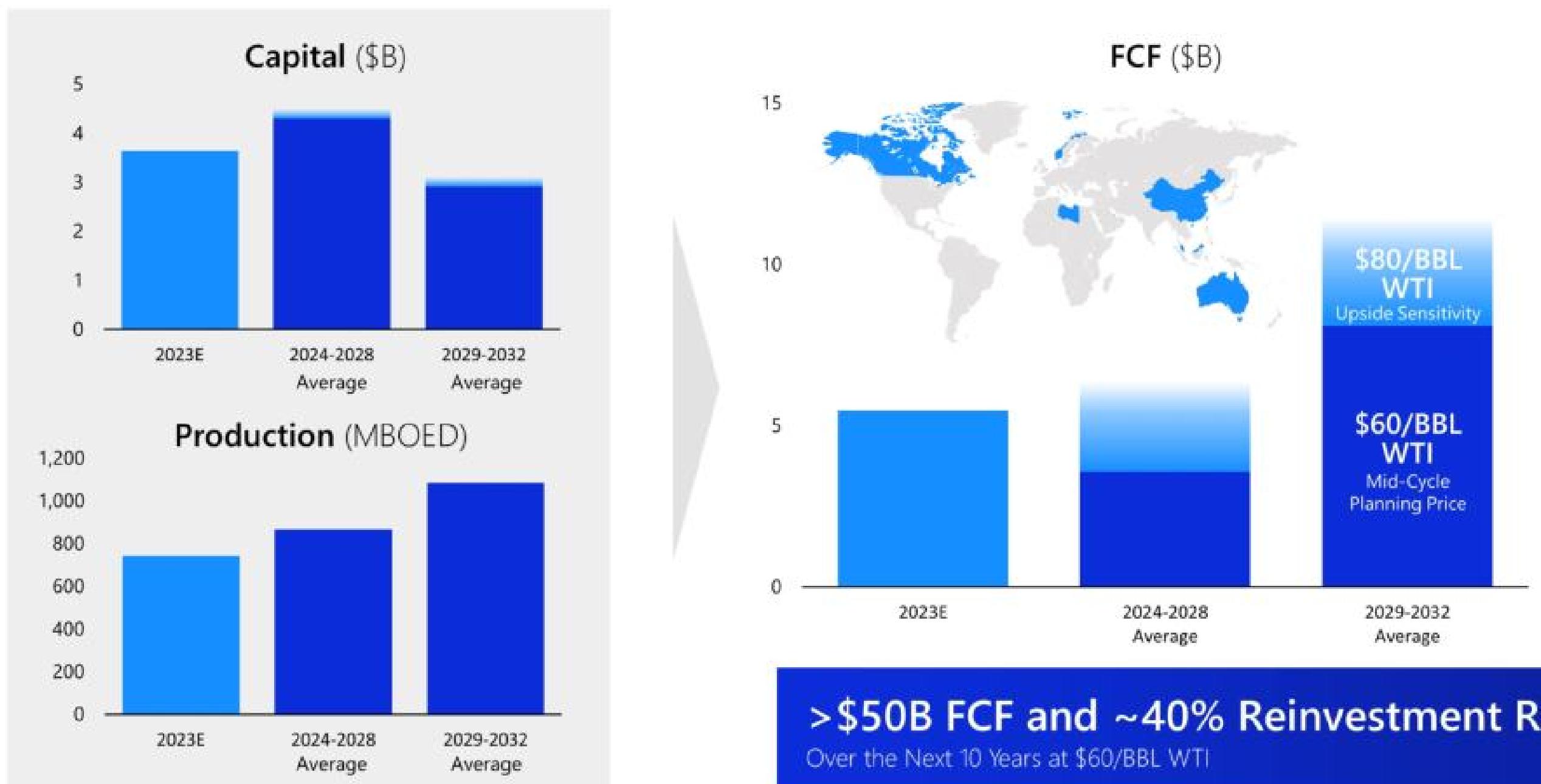
Example architecture: research and writer steps

1. The **researcher** retrieves relevant chunks and documents, and puts them into a data cache.
2. The **writer** uses the data cache to generate a structured output of interleaving text and image blocks.

Lower 48 Segment

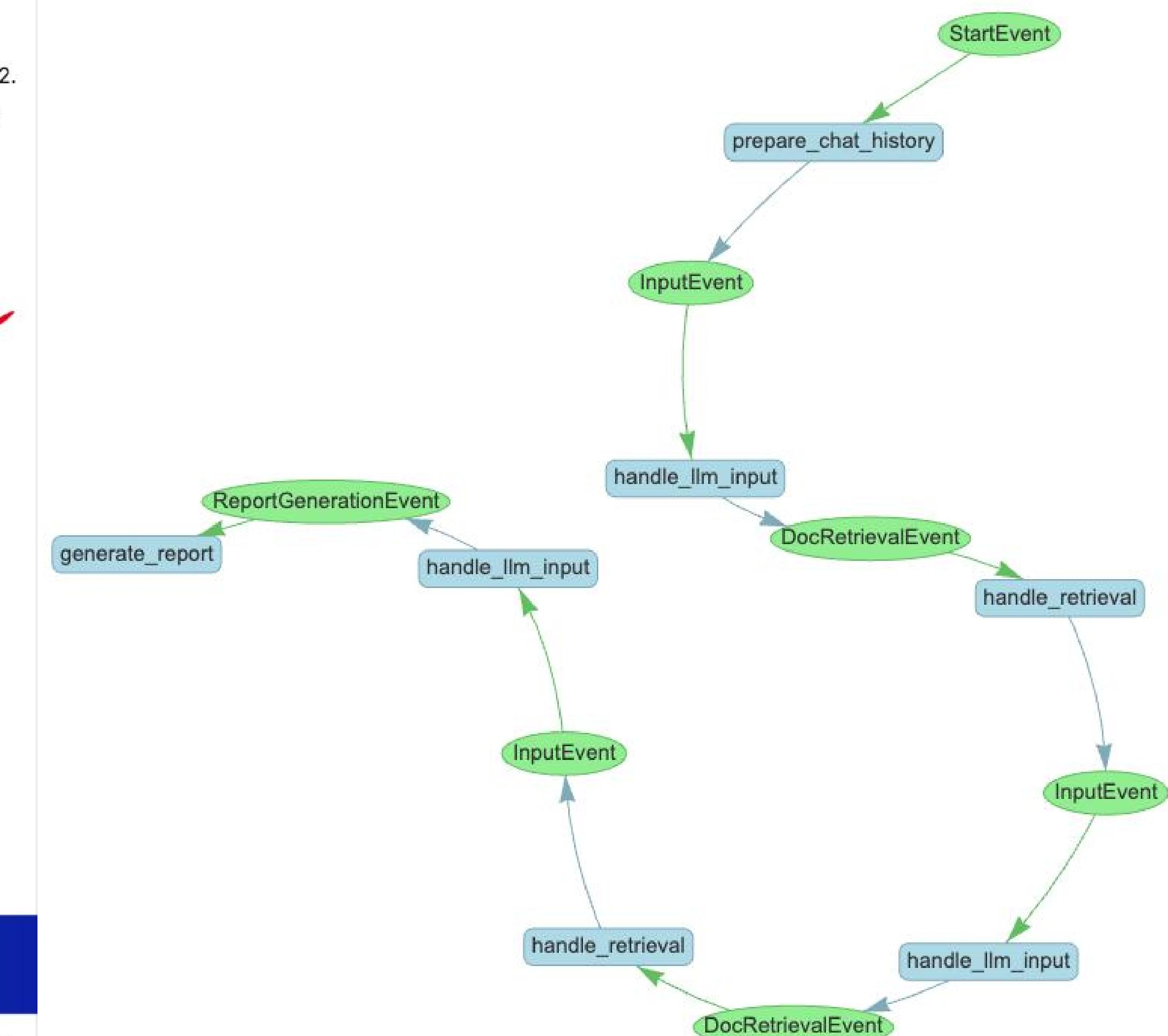
- **Capital Expenditures:** Expected to be *6.3 billion in 2023, averaging 6.5 billion from 2024-2028, and \$8.1 billion from 2029-2032.*
- **Production:** Projected to be around 1050 MBOED in 2023, increasing to 1220 MBOED on average from 2024-2028, and reaching 1530 MBOED on average from 2029-2032.
- **Free Cash Flow (FCF):** Estimated at *7 billion in 2023, averaging 5.5 billion from 2024-2028, and \$8 billion from 2029-2032.*
- **Key Projects:** Focused on the Permian Basin, Eagle Ford, and Bakken, with significant investments in technology and emissions reductions.

Alaska and International: Our Unique Diversification Advantage



Free cash flow (FCF) and reinvestment rate are non-GAAP measures defined in the Appendix.

CenocoPhilippines 29



Multimodal Report Generation

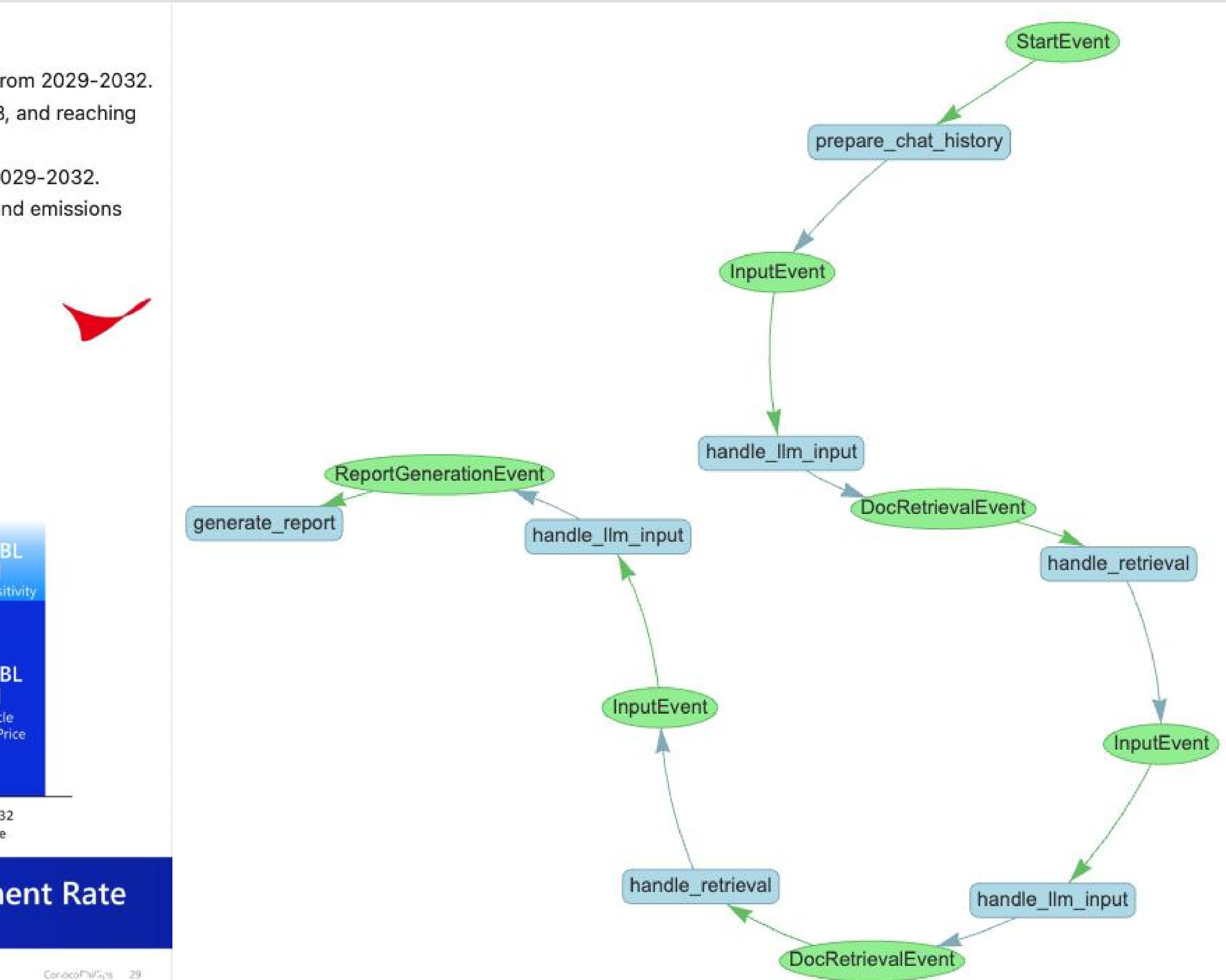
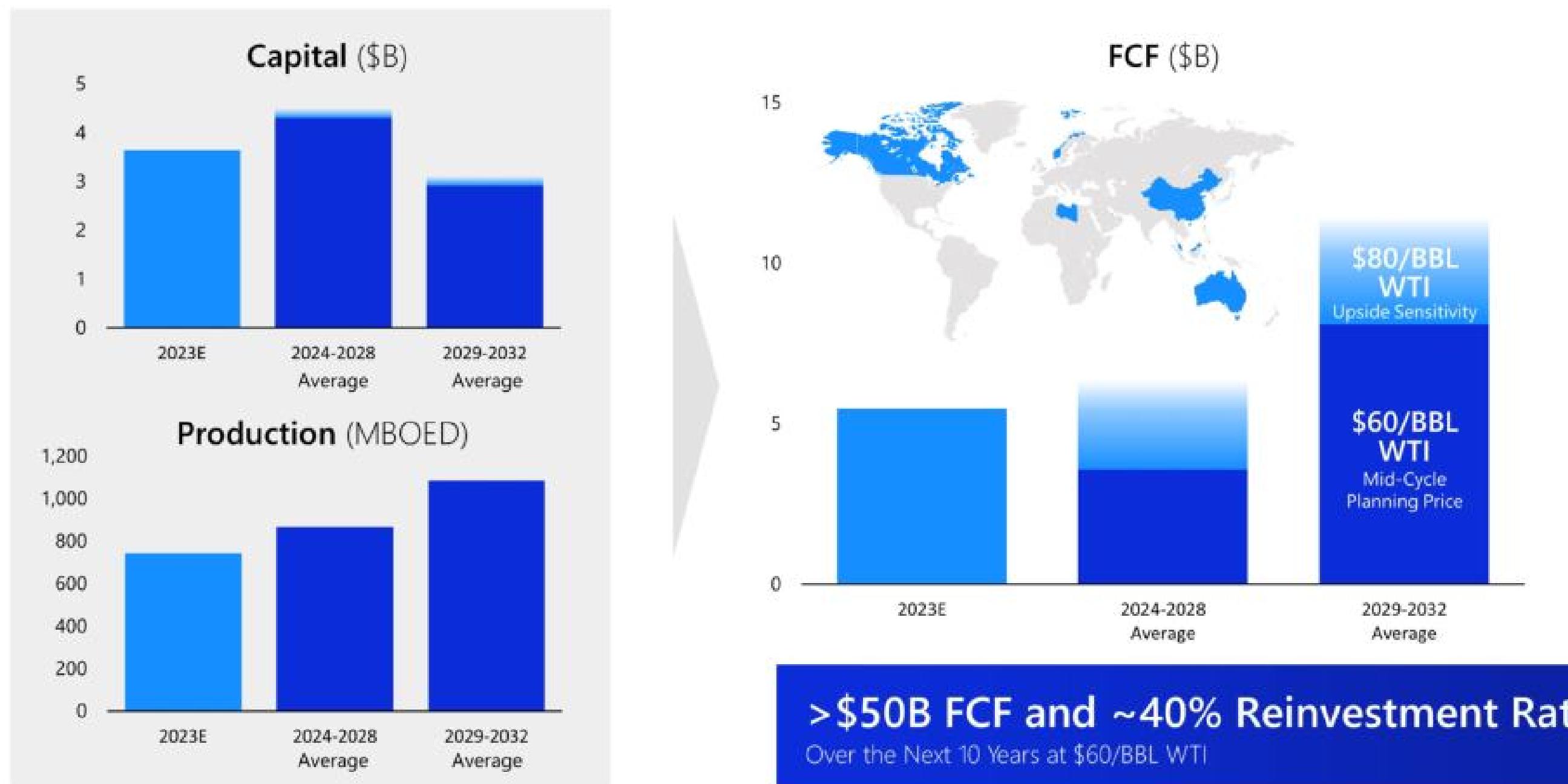
Generate interleaving text-and-image responses with the help of **structured outputs**.

https://github.com/run-llama/llama_parse/blob/main/examples/multimodal/multimodal_report_generation_agent.ipynb

Lower 48 Segment

- **Capital Expenditures:** Expected to be *6.3 billion in 2023, averaging 6.5 billion from 2024-2028, and \$8.1 billion from 2029-2032.*
- **Production:** Projected to be around 1050 MBOED in 2023, increasing to 1220 MBOED on average from 2024-2028, and reaching 1530 MBOED on average from 2029-2032.
- **Free Cash Flow (FCF):** Estimated at *7 billion in 2023, averaging 5.5 billion from 2024-2028, and \$8 billion from 2029-2032.*
- **Key Projects:** Focused on the Permian Basin, Eagle Ford, and Bakken, with significant investments in technology and emissions reductions.

Alaska and International: Our Unique Diversification Advantage



Towards a Scalable, Full-Stack Application

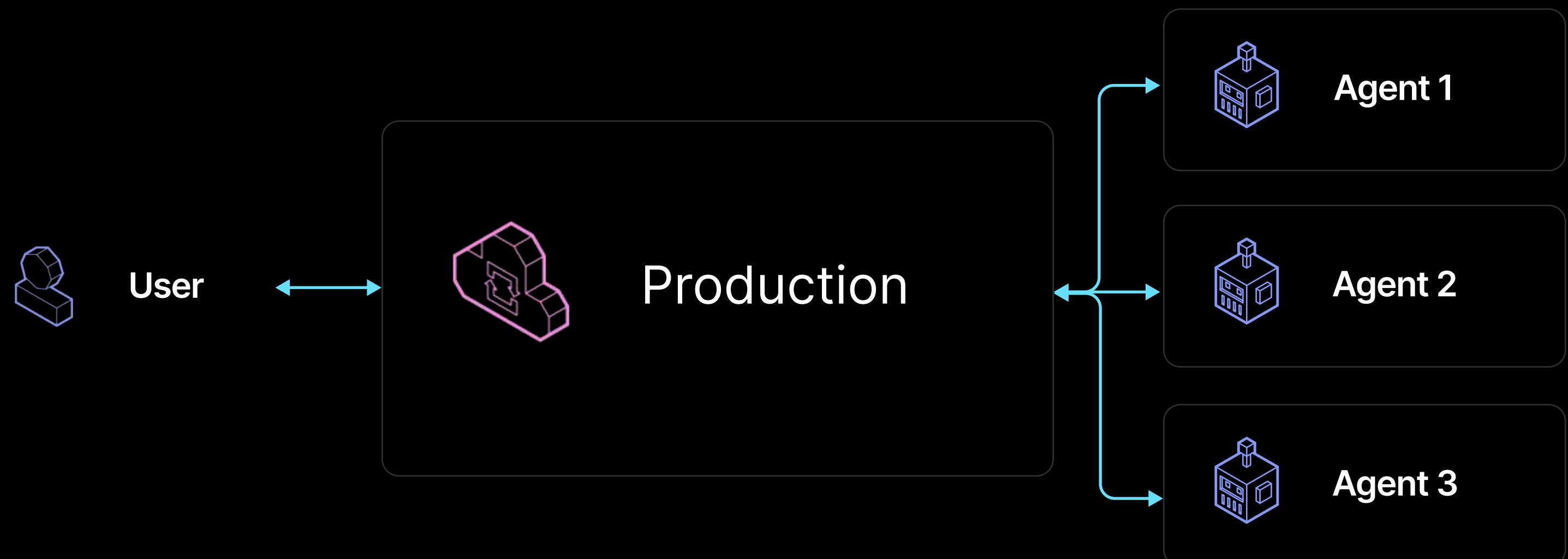


Running Agents in Production

You need the right architecture and infra components to serve complex, agentic workflows to end-users as a production application.

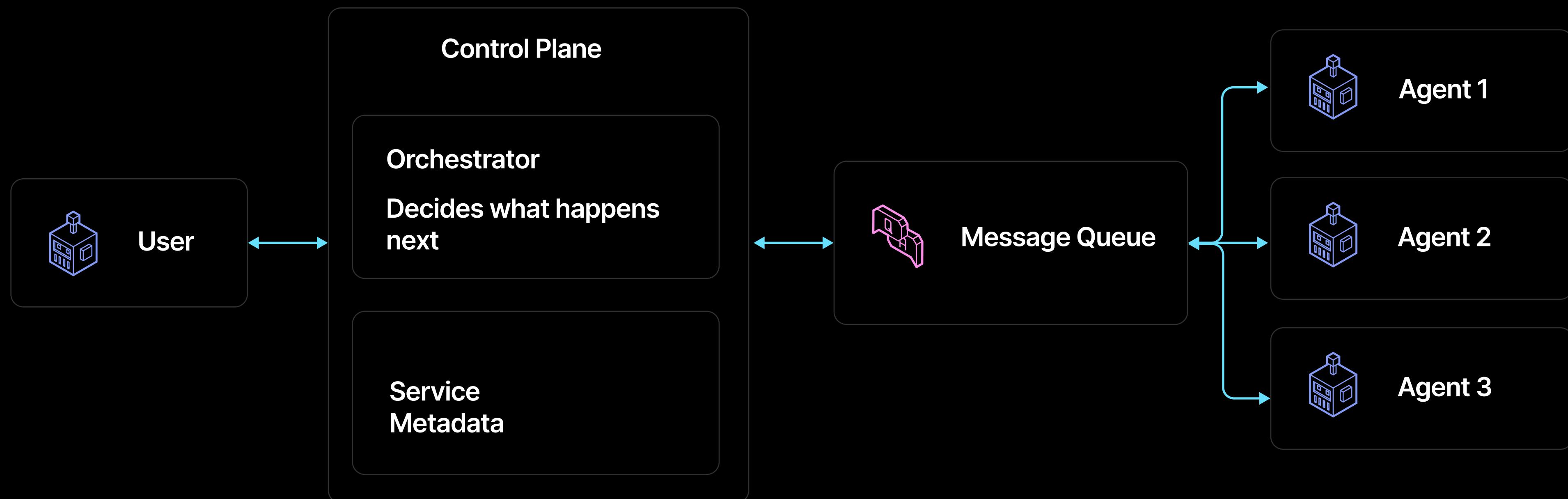
Requirements:

1. Encapsulation and re-use
2. Standardized communication interfaces between agents and with the client.
3. Scalability in number of users and number of agents
4. Human-in-the-loop for the end-user
5. Debugging and observability tools for the developer



Deploy agentic workflows as **microservices**.

- Model every agent workflow as a service API
- All agent communication occurs via a central message queue
- Distributed tool-execution
- Human-in-the-loop as a service
- Easy deployment with docker-compose and Kubernetes



- https://github.com/run-llama/llama_deploy

Thank you!

Llamalndex
September 23, 2024

