

ANTHROPIC

How to build effective AI Agents

Comprehensive breakdown of
Anthropic's report on "Building effective
Agents"



@rakeshgohe101





ANTHROPIC

Claude ▾

Research

Company

Careers

News

Try Claude

Product

Building effective agents

Dec 20, 2024

Over the past year, we've worked with dozens of teams building large language model (LLM) agents across industries. Consistently, the most successful implementations weren't using complex frameworks or specialized libraries. Instead, they were building with simple, composable patterns.

In this post, we share what we've learned from working with our customers.

Recently,

Anthropic released a full blog on how to create powerful agentic systems.

In this post, i will try cover all those details in a well defined manner so that it is easier to implement for different use-cases.

Lets get started....





Also

This report is quite different from our last breakdown about Google's AI Agent white paper.

Earlier coverage, Google's paper was more into What are AI Agents?

Where's Anthropic's blog is about Why and When you should use AI Agents





Basics of AI Agents

Anthropic shares what they think about agentic System and AI Agents in general.

They share what anthropic calls as agentic system and differences between AI Agents and Agentic workflow



Anthropic's Agentic System

Agentic Workflow



Workflows are systems where LLMs and tools are orchestrated through predefined code paths

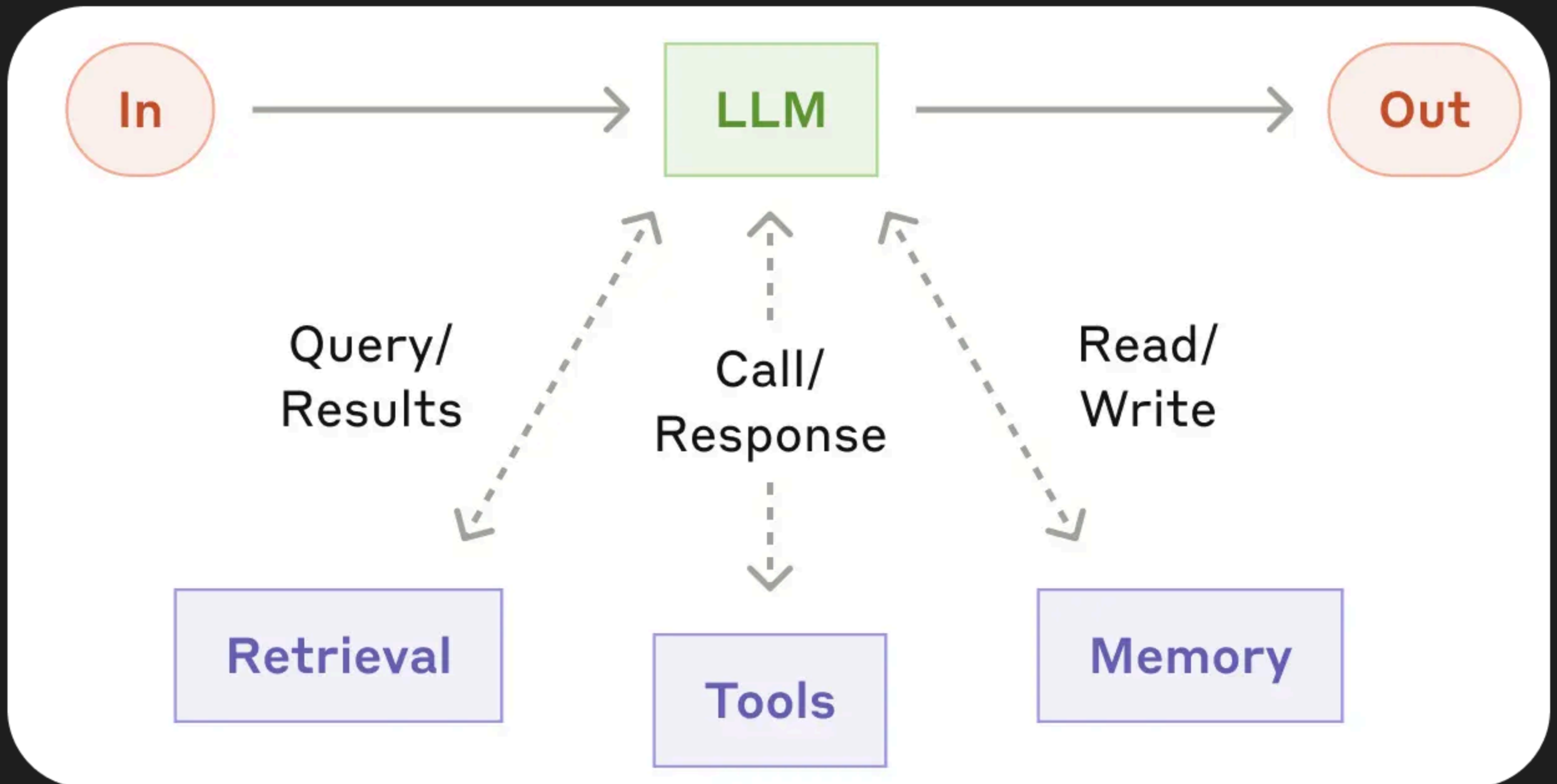
AI Agents



Agents, on the other hand, are systems where LLMs dynamically direct their own processes and tool usage, maintaining control over how they accomplish tasks



Anthropic's AI Agent Architecture



Anthropic argues that the augmented LLM is the core building block of Agentic system supported with tools, memory and a smart retrieval system.





When to choose a AI Agent framework

Anthropic shares a clever
argument to not always go for
agentic framework while building
AI Agents



When and how to choose frameworks

Anthropic gives example of popular frameworks like Langchain and Amazon bedrock agents, and share why you should not depend on frameworks

◆ Anthropic shares

1. Start with LLM API's – Most agentic pattern can be built with simple LLM API's.
2. If you need complex workflows then only use frameworks.
3. You should avoid frameworks as much as possible to make sure that your workflow remains clean.
3. They share this because frameworks come with large underlying code and sometimes this could be counter intuitive to your work



Anthropic's MCP —

Q Search...

Ctrl K

Get Started

Introduction

Get started with the Model Context Protocol (MCP)

MCP is an open protocol that standardizes how applications provide context to LLMs. Think of MCP like a USB-C port for AI applications. Just as USB-C provides a standardized way to connect your devices to various peripherals and accessories, MCP provides a standardized way to connect AI models to different data sources and tools.

Why MCP?

MCP helps you build agents and complex workflows on top of LLMs. LLMs frequently need to integrate with data and tools, and MCP provides:

- A growing list of pre-built integrations that your LLM can directly plug into
- The flexibility to switch between LLM providers and vendors
- Best practices for securing your data within your infrastructure

Anthropic suggests using their latest open-source framework protocol, MCP (Model Context Protocol) if you are building agents with a simple client workflow.

They even gave a [link](#) to their LLM cookbook which you also access from the blog linked in the comments.



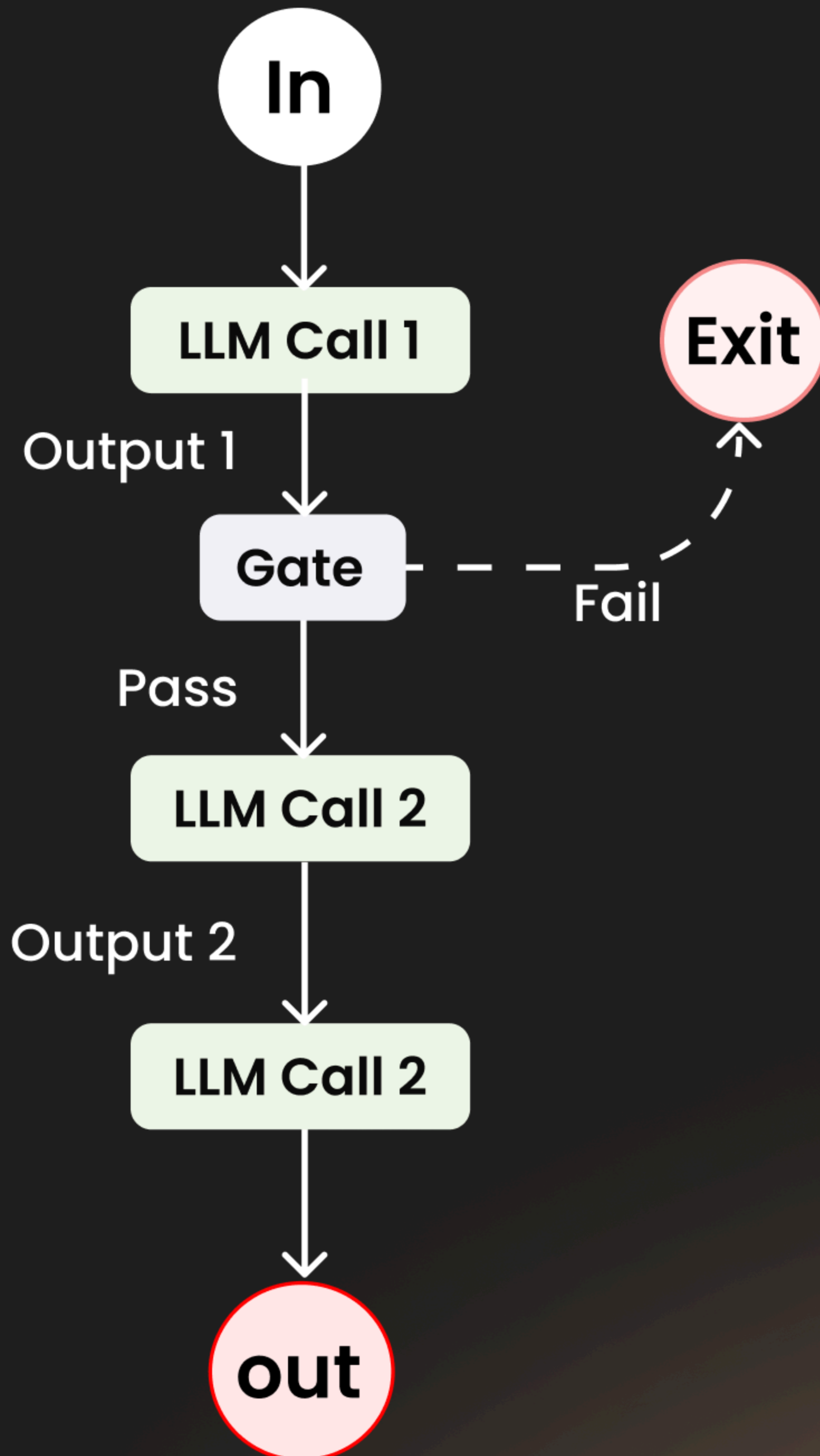


Anthropic's most common LLM-based Workflows

Anthropic shares few common LLM-based workflows for you instead of relying heavily on frameworks



Workflow: Prompt chaining



— Description

Prompt chaining decomposes a task into a sequence of steps, where each LLM call processes the output of the previous one.

You can add programmatic checks like gate in the diagram and on any intermediate steps to ensure that the process is still on track.

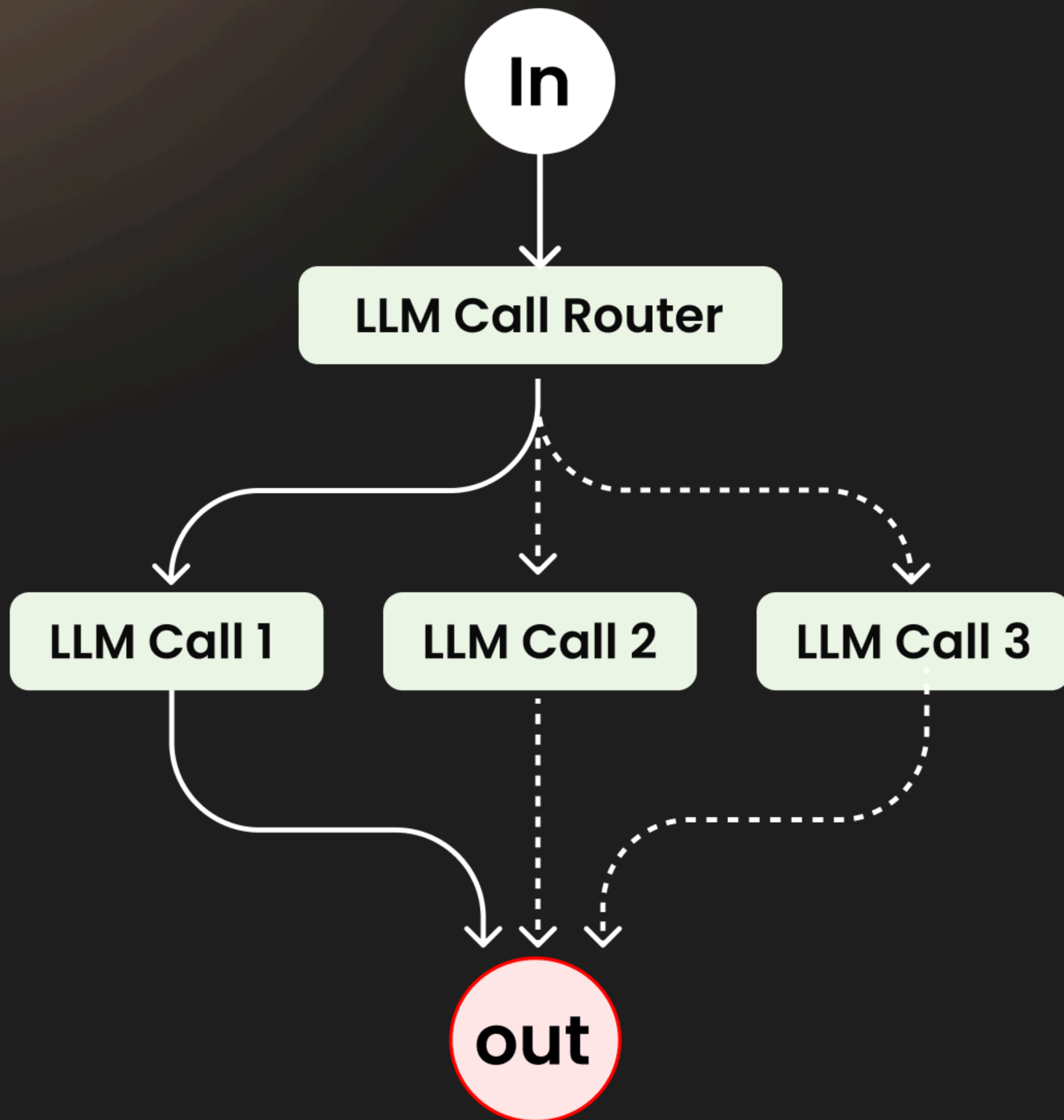
When to use ?

This workflow is best for tasks that can be broken down into fixed, simpler subtasks, aiming to improve accuracy at the cost of increased latency.

Example: Generating Marketing copy and creating outline for a document



Workflow: Routing



— Description

Routing classifies an input and directs it to a specialized followup task.

This workflow allows for separation of concerns, and building more specialized prompts.

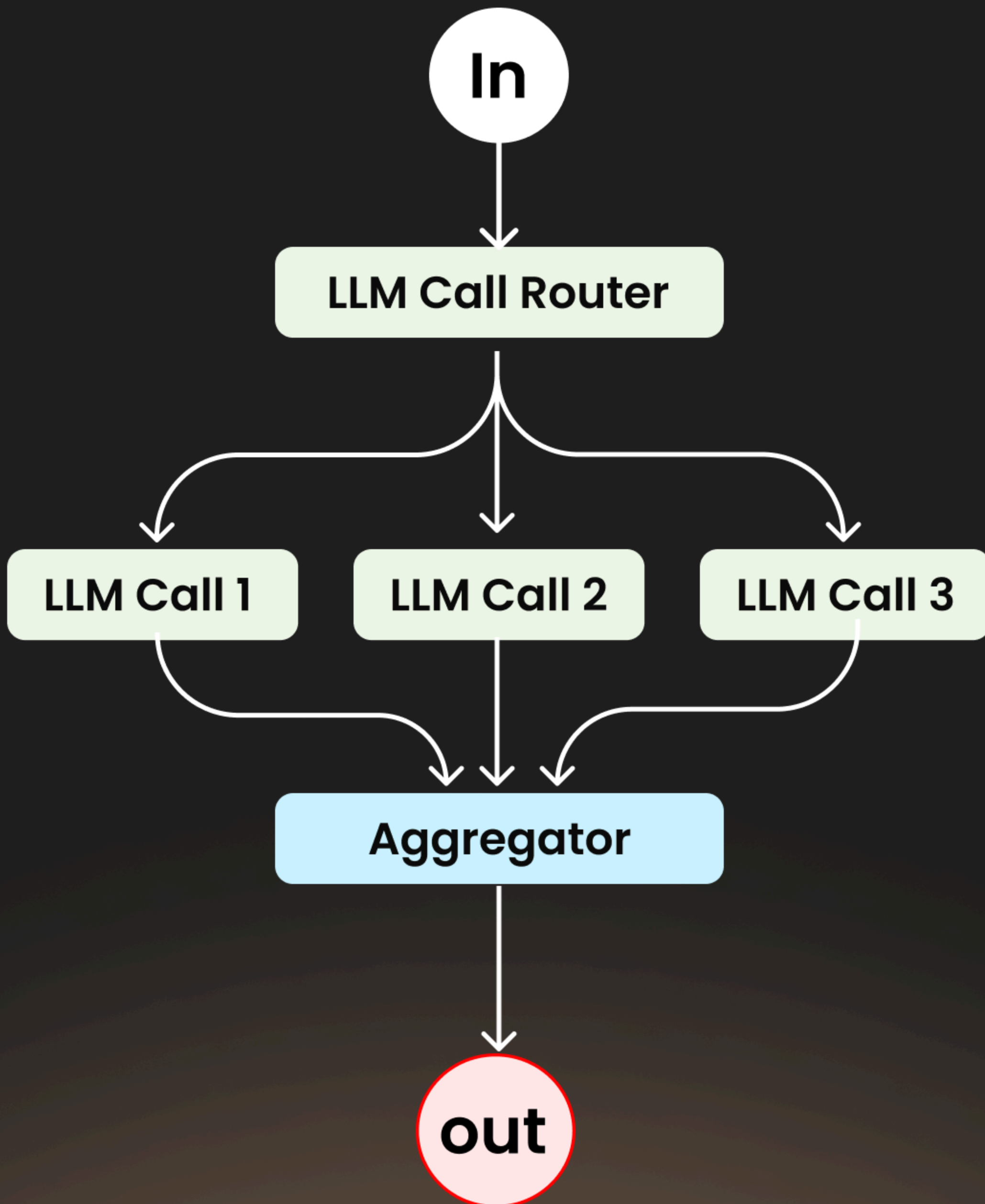
When to use ?

It is effective for complex tasks with distinct categories that are best handled separately, relying on accurate classification by an LLM or traditional model..

Example: Directing different types of customer service queries, Routing easy questions to smaller models



Workflow: Parallelization



— Description

LLMs can sometimes work simultaneously on a task and have their outputs aggregated.

This workflow, parallelization, manifests in two key variations:

- **Sectioning:** Breaking a task into independent subtasks run in parallel.
- **Voting:** Running the same task multiple times to get diverse outputs.

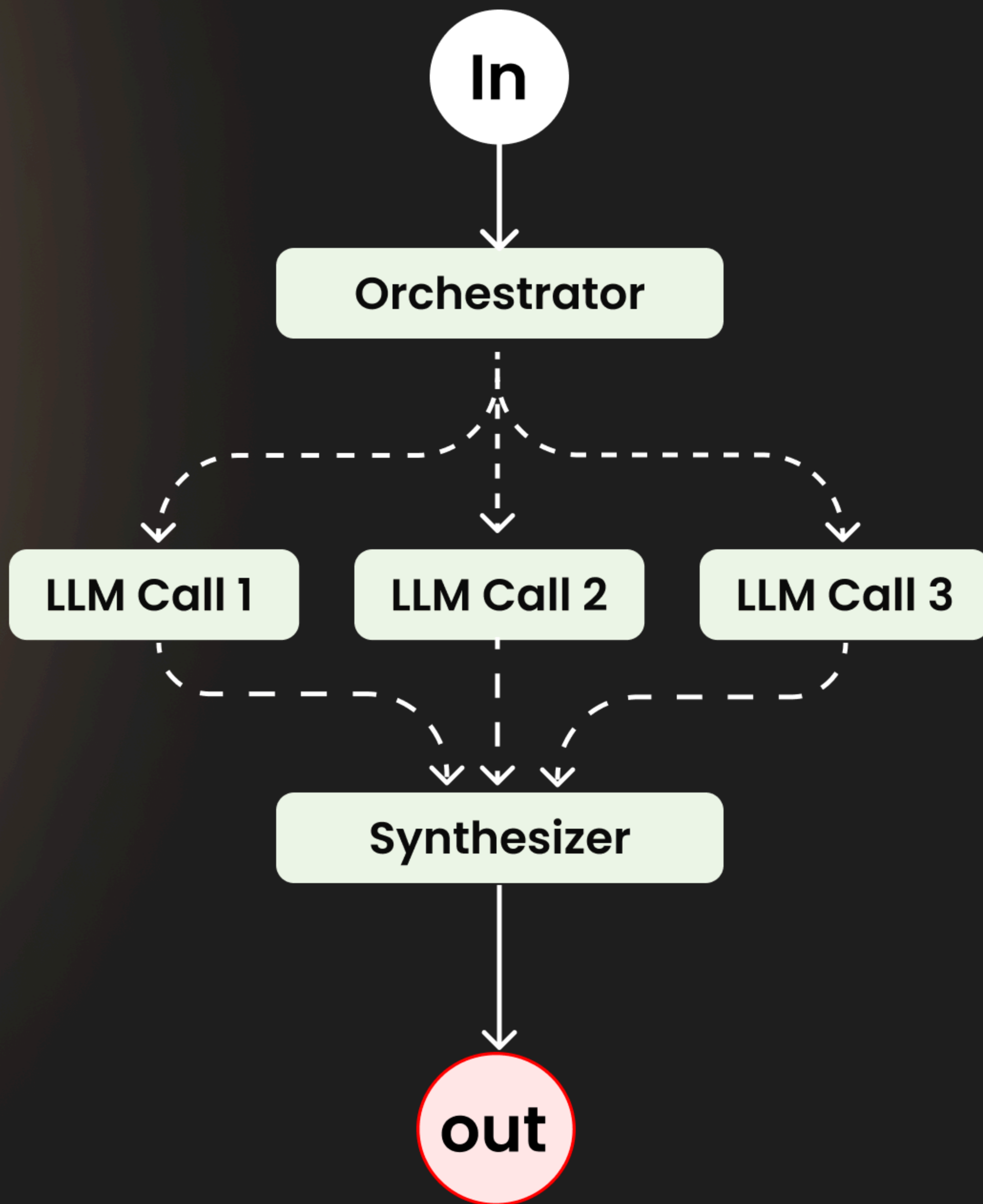
When to use ?

Parallelization is effective when subtasks can be processed simultaneously for speed or when multiple perspectives are needed for higher confidence.

Example: Reviewing a piece of code for vulnerabilities, Evaluating content



Workflow: Orchestrator based



— Description

In the orchestrator-workers workflow, a central LLM dynamically breaks down tasks, delegates them to worker LLMs, and synthesizes their results.

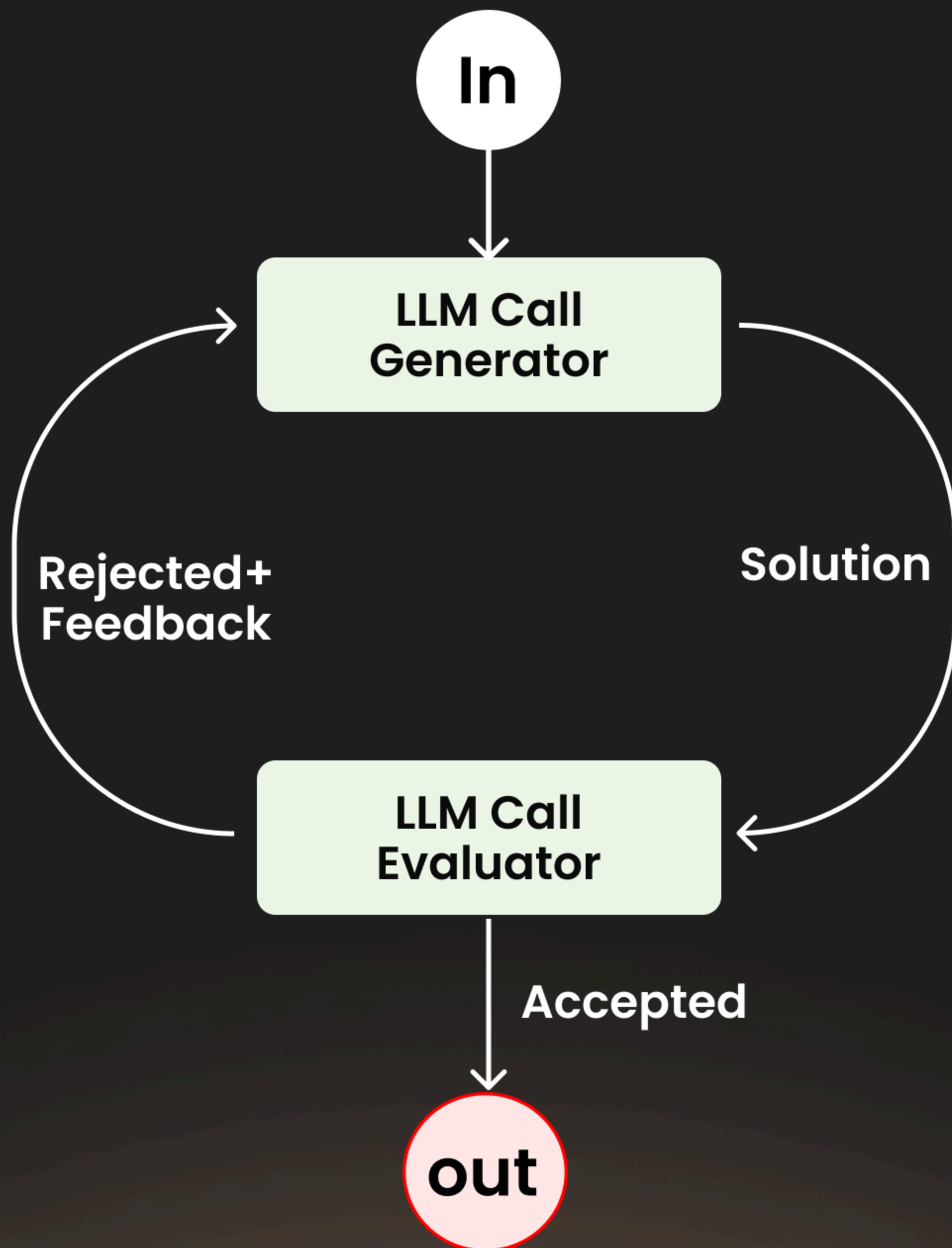
When to use ?

This workflow is ideal for complex tasks where subtasks are unpredictable, such as coding tasks with varying file changes.

Example: Coding workflows that make complex changes to multiple files, Search tasks that involve gathering information from multiple information



Workflow: Evaluator-optimizer



— Description

In the evaluator-optimizer workflow, one LLM call generates a response while another provides evaluation and feedback in a loop.

When to use ?

This workflow is effective when there are clear evaluation criteria and iterative refinement adds measurable value.

It fits well when LLM responses improve with human feedback and the LLM can provide such feedback, similar to the iterative writing process.

Example: Complex search tasks that require multiple rounds of searching, Creating conversational Agents



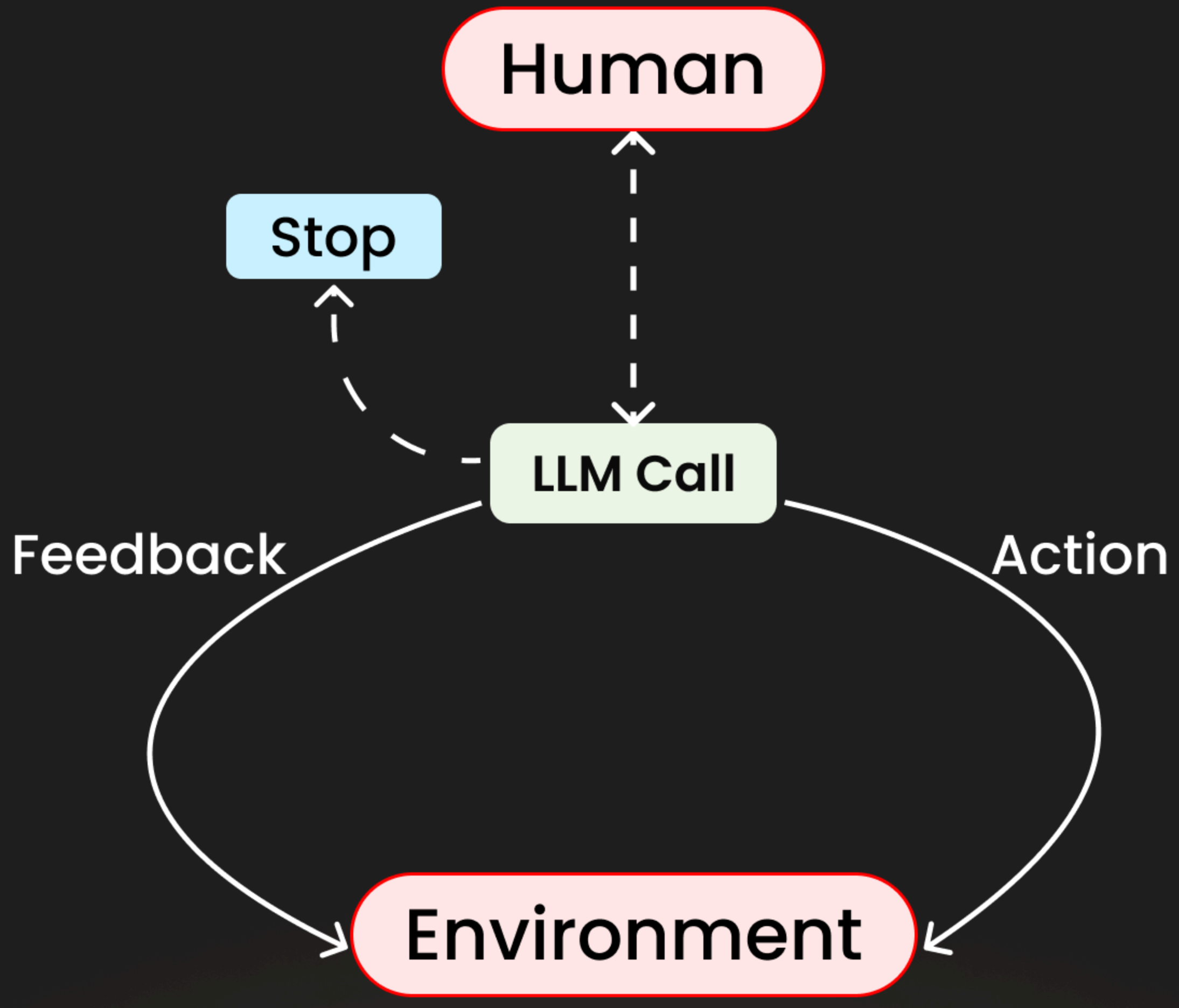
A large, stylized number '4' in a light brown color with a white outline, positioned on the left side of the slide.

Anthropic's AI Agent definition and architecture

Agents are becoming common in production as LLMs. So, Anthropic creates a common architecture about how agents works



Autonomous Agent



— Description


Agents are increasingly used in production as LLMs advance in understanding, reasoning, planning, and tool use.

They start with human input, operate independently, and verify progress, pausing for feedback as needed. Tasks end upon completion or predefined conditions.

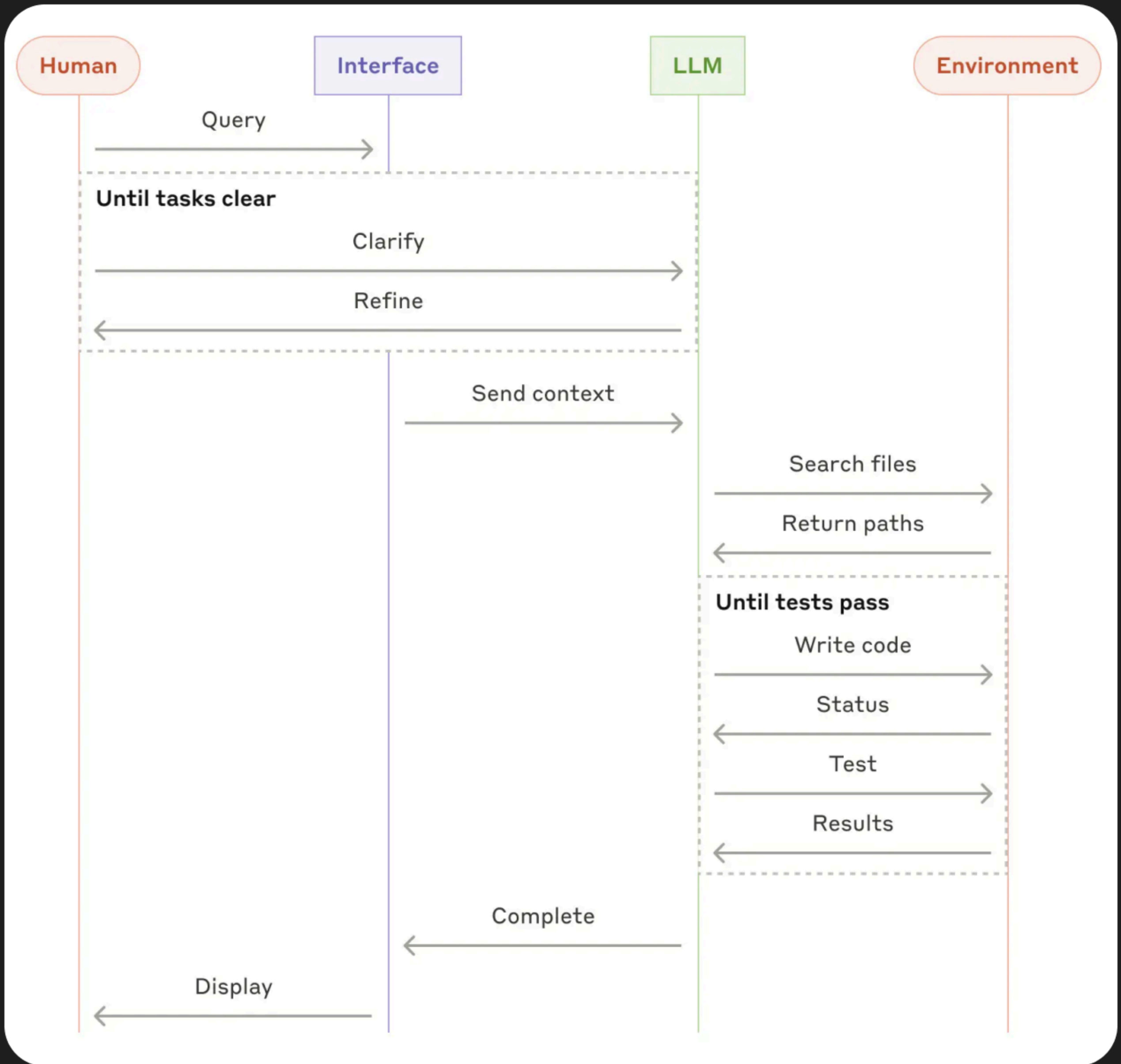
When to use ?

Agents are suited for open-ended problems where steps are unpredictable and paths can't be hardcoded..

Example: A coding Agent like cursor, GUI based Agents

They also share a high level sequence diagram for a coding agent 





Sequence diagram for high-level coding agent



Conclusion

Anthropic argue that creating an agentic solution is not just about using all the available resources.

Also the workflow shared are not just perspective, they share that this workflow was curated with the help of their customer and internal experimentation.

Hence, while building agents they want to focus on these key things:

1. Maintain simplicity in your agent's design.
2. Prioritize transparency by explicitly showing the agent's planning steps.
3. Carefully craft your agent-computer interface (ACI) through thorough tool documentation and testing.



Additional Info +

They also argue about building a proper Agent-computer interface just like that way we build your Human-based interface.

This will help AI Agent to navigate faster with user-interface

Additional Info +

They also request agent builders to properly evaluate model's parameters and computational cost before building an agent to save cost and resources accordingly.

Use smaller models for quick tasks and bigger model for reasoning tasks

Hi, I am
Rakesh Gohel



**“We help businesses 10X their growth with
Cloud and AI Agents”**

FOLLOW TO LEARN MORE ABOUT AI AGENTS

[linkedin.com/in/rakeshgohel01](https://www.linkedin.com/in/rakeshgohel01)