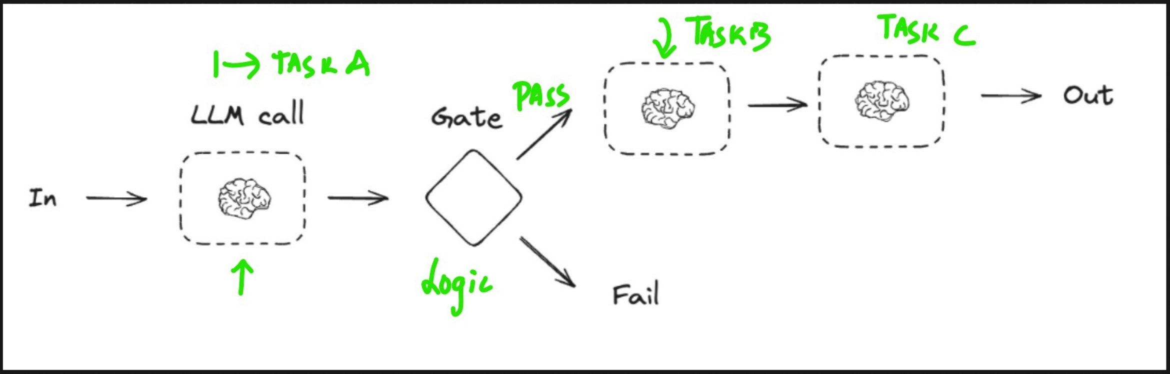
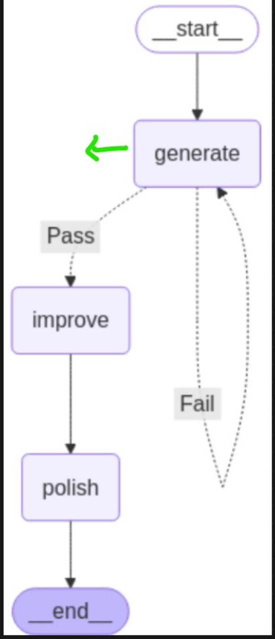
Workflows

1) Prompt Chaining

Defn:

Prompt chaining is a technique in natural language processing where multiple prompts are sequenced together to guide a model through a complex task or reasoning process. Instead of relying on a single prompt to achieve a desired outcome, prompt chaining breaks the task into smaller, manageable steps, with each step building on the previous one. This approach can improve accuracy, coherence, and control when working with large language models.





⇒ Workflow

Usecase

Generate a Story ← Logic

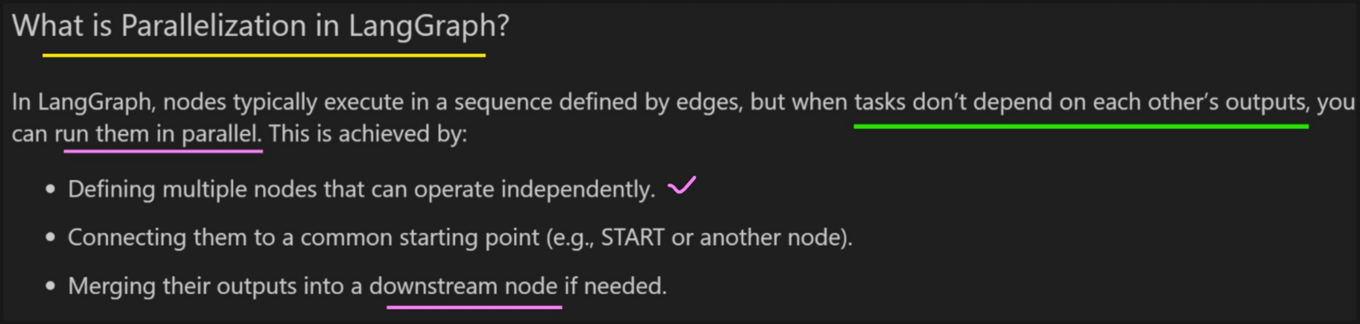
1) Generate → 1 < PASS / FAIL

2) Improve

3) Polish

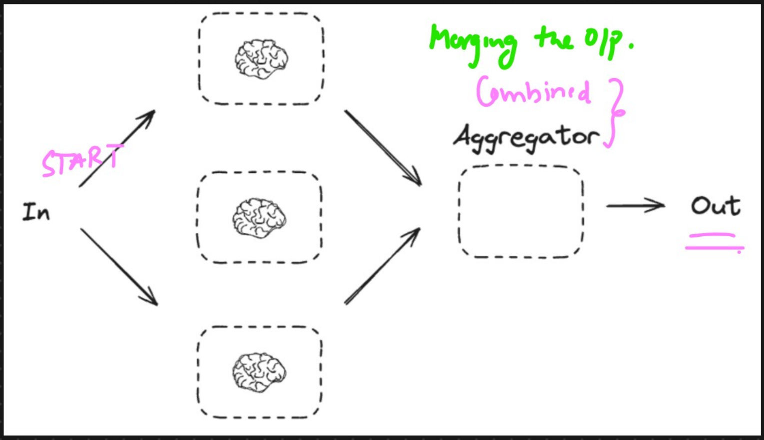
Final Result.

2) Parallelization

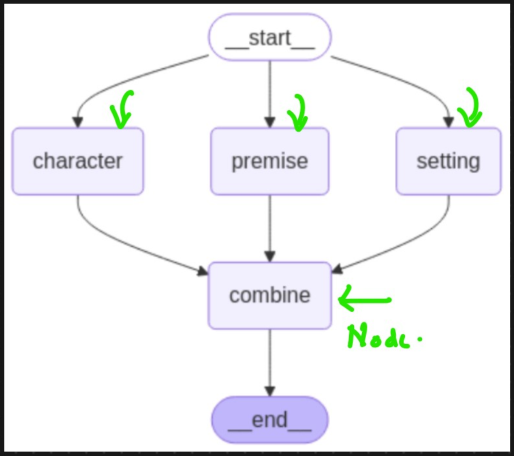

What is Parallelization in LangGraph?

In LangGraph, nodes typically execute in a sequence defined by edges, but when tasks don't depend on each other's outputs, you can run them in parallel. This is achieved by:

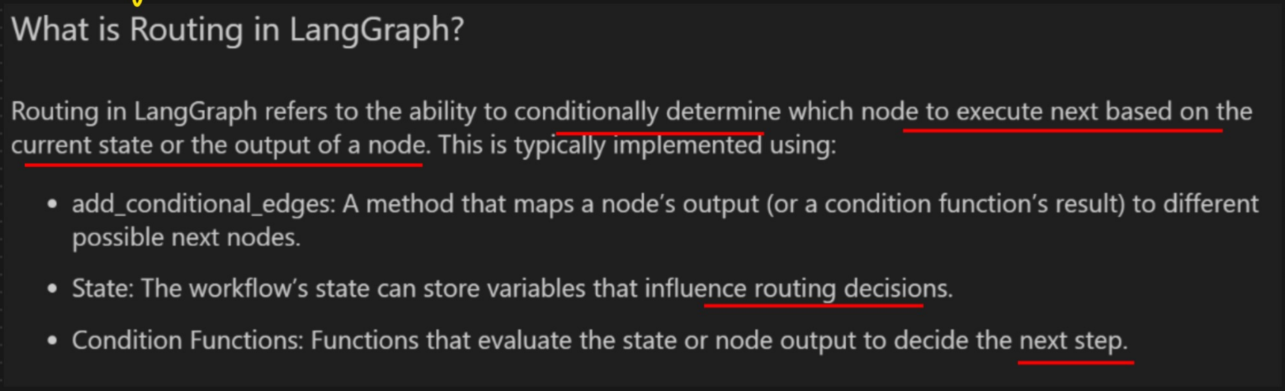
- Defining multiple nodes that can operate independently. ✓
- Connecting them to a common starting point (e.g., START or another node).
- Merging their outputs into a downstream node if needed.



Usecase



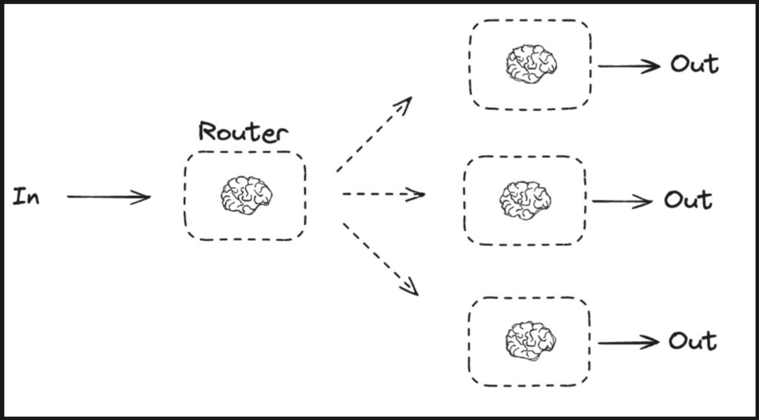
③ Routing



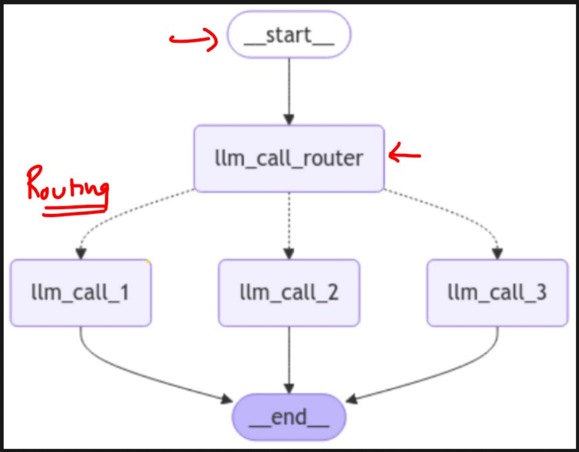
What is Routing in LangGraph?

Routing in LangGraph refers to the ability to conditionally determine which node to execute next based on the current state or the output of a node. This is typically implemented using:

- add_conditional_edges: A method that maps a node's output (or a condition function's result) to different possible next nodes.
- State: The workflow's state can store variables that influence routing decisions.
- Condition Functions: Functions that evaluate the state or node output to decide the next step.

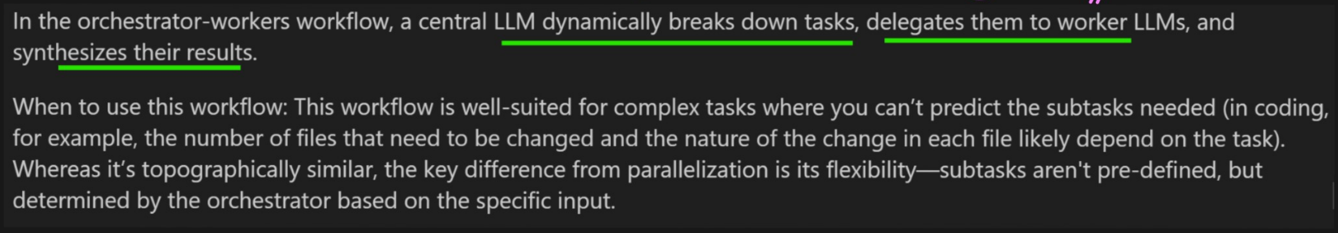


Usecase.



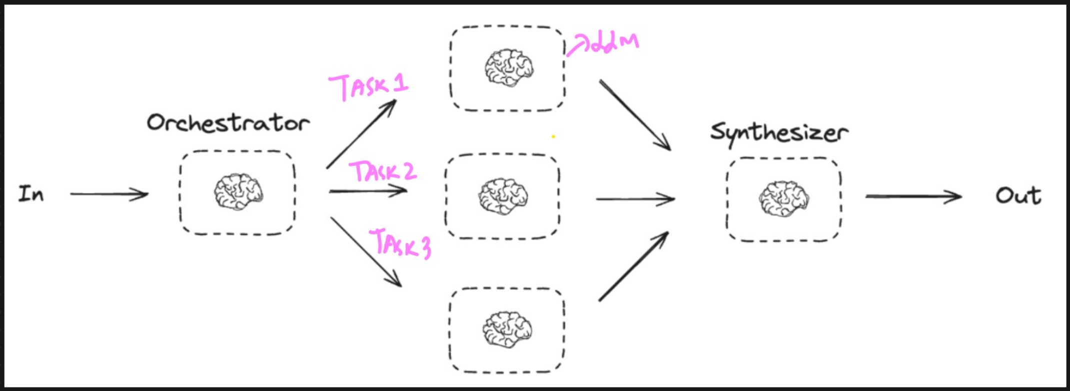
④ Orchestrator - Worker

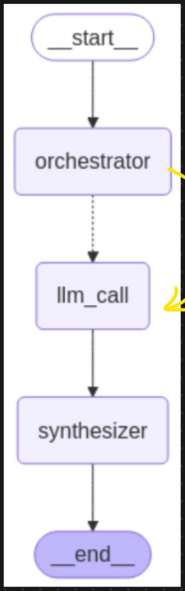
① ② ⇗ parallely



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: This workflow is well-suited for complex tasks where you can't predict the subtasks needed (in coding, for example, the number of files that need to be changed and the nature of the change in each file likely depend on the task). Whereas it's topographically similar, the key difference from parallelization is its flexibility—subtasks aren't pre-defined, but determined by the orchestrator based on the specific input.

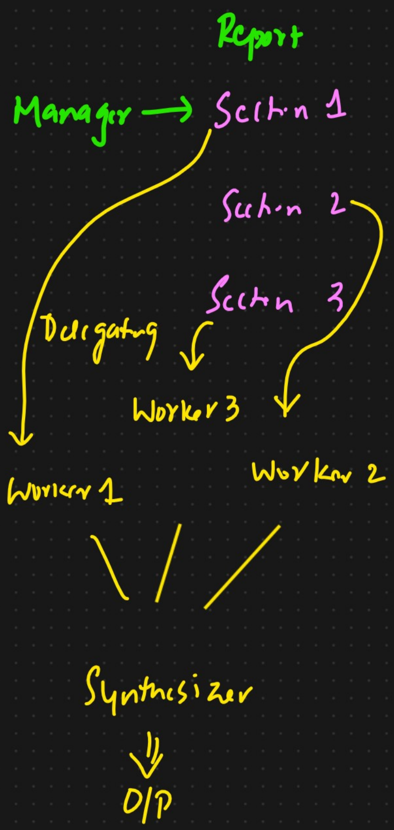




⇙ Workflow

Workers

Usecase



Generate a Detailed Report

Multiple Section → Section 1

Section 2

Agentic AI System → Introduction ←

History ←

Current Trends in 2025 ←

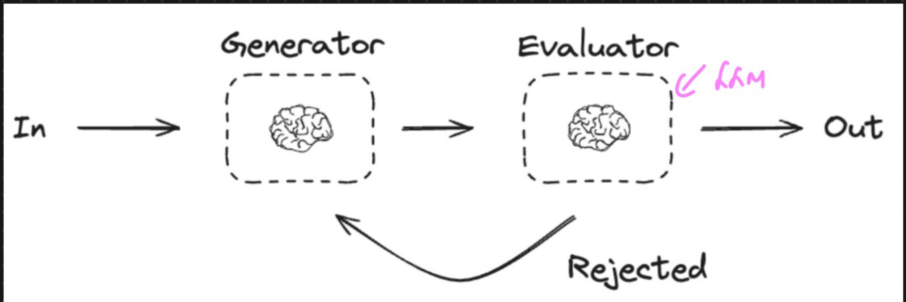
Section 1 → Name → Title

→ Description → Detailed Info.

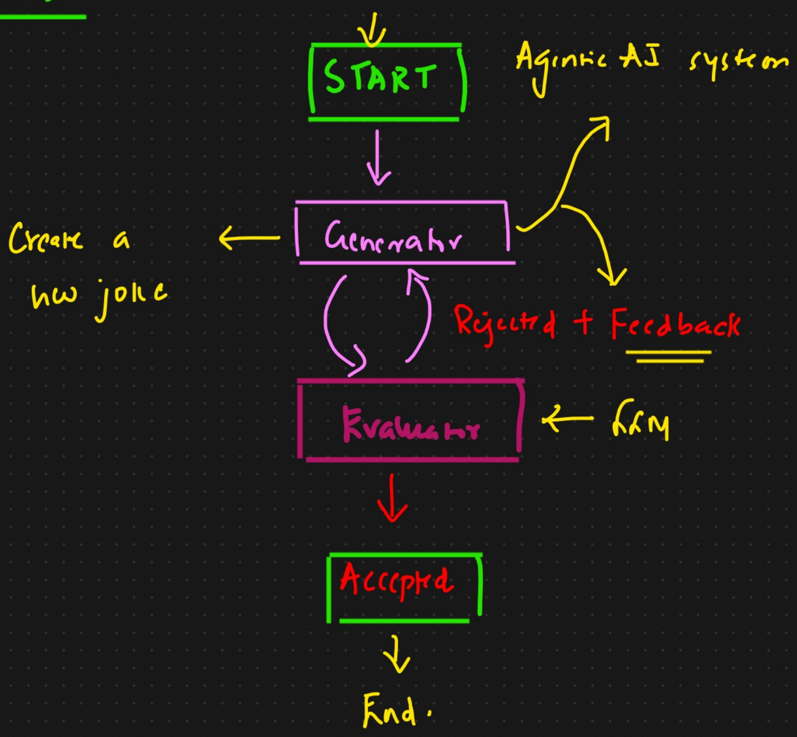
Evaluator Optimizer

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: This workflow is particularly effective when we have clear evaluation criteria, and when iterative refinement provides measurable value. The two signs of good fit are, first, that LLM responses can be demonstrably improved when a human articulates their feedback; and second, that the LLM can provide such feedback. This is analogous to the iterative writing process a human writer might go through when producing a polished document.



Usecase


START
Agentic AI system
Generator
Create a new joke
Rejected + Feedback
Evaluator
LLM
Accepted
End.