O'REILLY®

Developing with AI Agent Swarms

Agentic software development with claude-code and claude-flow

Dive Deep Into Multi-Agent Systems!

From Simple Workflows to Self-Organizing Swarms

Understand multi-agent systems

- What can you do with them?
- What are the architectures involved?
- How do you make the right decisions?

Complete GitHub repository with slides, script, and demo instructions available at:

https://github.com/jamesurquhart/swarmclass

Meet your Instructor

James Urquhart, Field CTO, KamiwazaAI

35 years of distributed systems design, development, deployment, and operations experience.

Senior technical leadership roles at AWS, VMware, Pivotal, Cisco, Dell, Enstratius, and Forte Software

Author of *Flow Architectures: The Future of Streaming and Event-Driven Integration* (O'Reilly, 2021)

Author of The Wisdom of Clouds from 2006-2016. Consistently voted one of the top three most influential blogs on cloud computing.

Currently Field CTO for KamiwazaAI, delivering AI data access, governance, and security to the enterprise

image: James Urquhart

Course schedule

Section 1:

Agentic AI and Swarm Basics (60 min)

→ Concepts, terminology, six architectures

→ Demo: Sequential Pipeline (Research & Write)

Section 2:

Creating Advanced Multi-Agent Systems (60 min)

→ Deep dives: Hierarchical, Debate, Committee, Routing

→ Demo: Multi-Agent Code Review

Section 3:

Applying Multi-Agent Patterns (60 min)

→ Decision framework, emergent coordination, best practices

→ Demo: Self-Organizing Swarm walkthrough

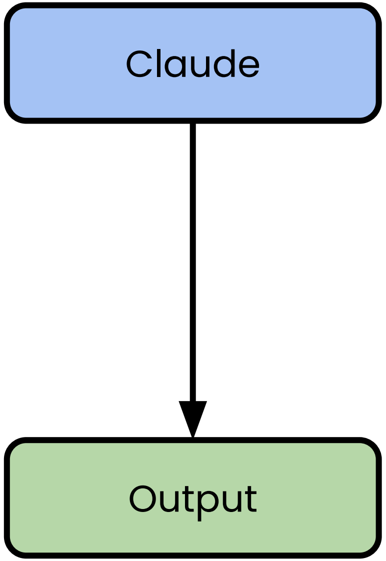
Course Objectives

By the end of this course, you will:

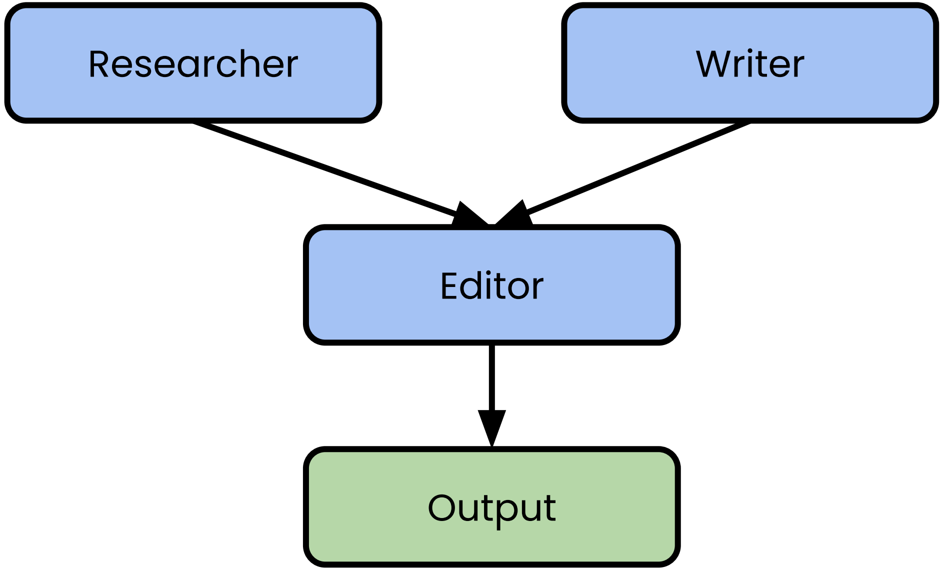
- Build multi-agent systems and see swarm coordination in action with claude-flow
- Know six architectural patterns and when to choose swarms over other approaches
- Understand how swarms differ from structured multi-agent orchestration

The Shift

Single Agent



Multi-Agent



Definition

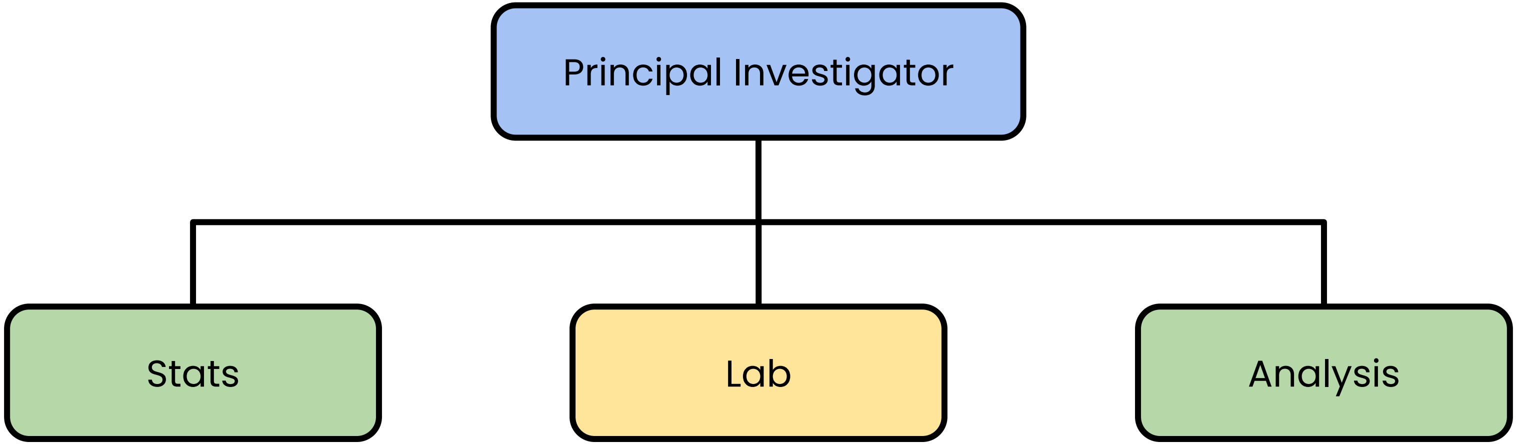
Multi-agent AI is a computational paradigm where two or more autonomous AI **agents**—each with distinct roles, capabilities, or knowledge domains—coordinate, collaborate, or compete to achieve individual or collective goals through structured **interaction protocols**.

Why Multi-Agent Systems?

Single Agent Limitations		Multi-Agent Solutions
Context window constraints	⟶	Decompose across agents
Lack of specialized depth	⟶	Each agent optimized for one task
Sequential bottlenecks	⟶	Parallel execution

The Human Analogy

How effective human teams work:



- Specialists coordinate
- They debate and verify each other
- Outcomes none could achieve alone

Real-World Deployments

Financial Services

Multi-agent fraud detections

95%+ accuracy, reduced false positives

Insurance

7-agent claims processing

80% faster processing time

Warehouses

Amazon Kiva robot swarms

Hundreds of agents coordinating in real-time

AI Products

ChatGPT (Operator + Deep Research + Code)

Claude Code (subagent spawning)

Edge/On-Prem

Local open source models (Llama, Mistral)

Privacy, performance, and air-gapped deployments

Core Definitions

Agent

Autonomous AI with specific role, tools, and decision-making capability

Subagent

An agent spawned by another agent to handle a subtask

Orchestration

Coordinating multiple agents toward a shared goal

Context Window

The information an agent can process at once; each agent has its own

Handoff

Passing work from one agent to another

Shared Memory

A common data store agents use to communicate state

Core Definitions

MCP

Model Context Protocol—open standard for connecting agents to external tools/data

Swarm

Multiple agents working in parallel with emergent coordination (not centrally directed)

Framework Examples

Framework	Philosophy	Best For
Claude Code	Terminal-based agents	Development workflows with tool access
Claude Flow	Swarm orchestration,	Parallel agent shared memory coordination
LangGraph	Graph-based stateful workflows	Complex orchestration with precise control
CrewAI	Role-based teams	Rapid prototyping ("researcher", "writer")

!

Emerging Standard:

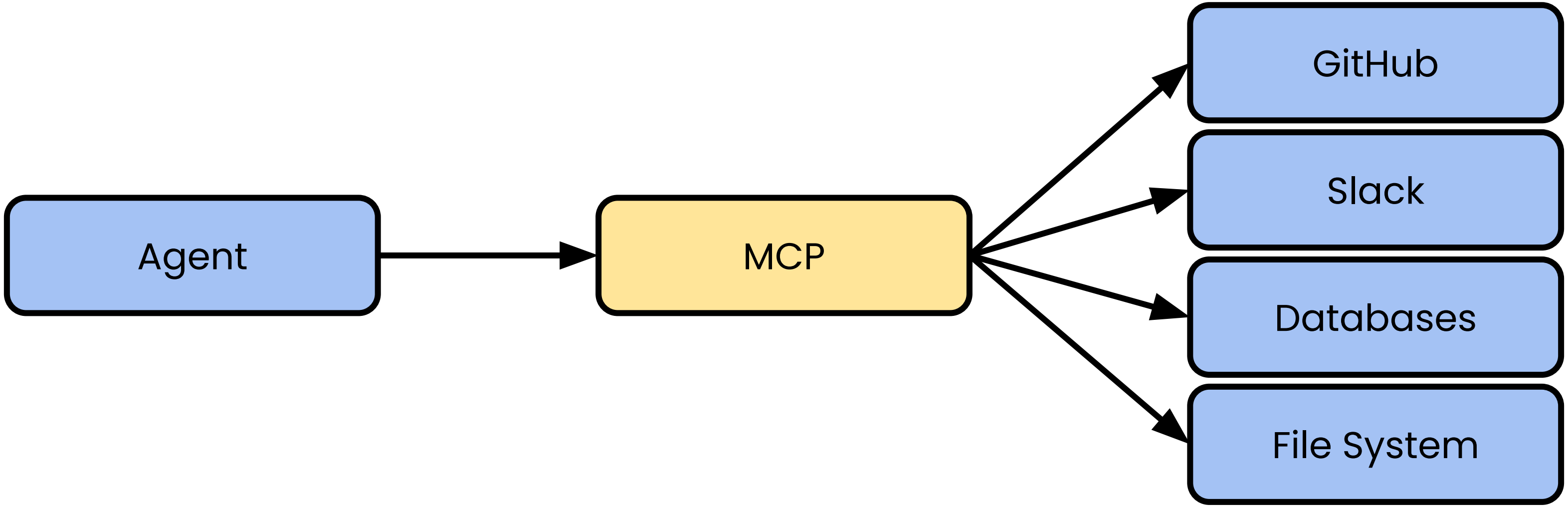
Agentic AI Foundation (Linux Foundation)

→ MCP, goose, AGENTS.md

→ Backed by Anthropic, OpenAI, Google, Microsoft, AWS

MCP – The Connector

Model Context Protocol



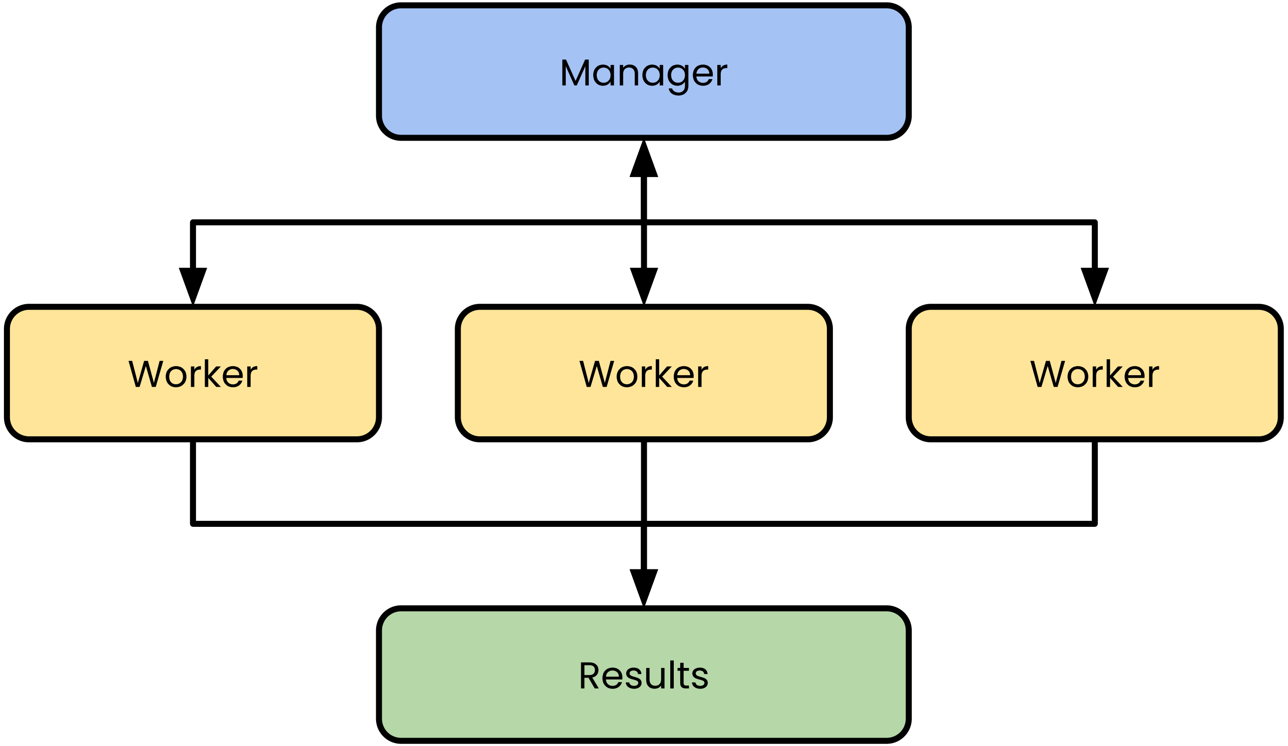
Originally Anthropic • Now AAIF/Linux Foundation

10,000+ published MCP servers

Six Multi-Agent Architectures

Pattern	Description
Hierarchical	Manager delegates to workers
Debate	Agents argue, judge decides
Committee	Parallel processing, aggregated results
Pipeline	Sequential handoffs
Routing	Dynamic dispatch to specialists
Network/Swarm	Emergent coordination

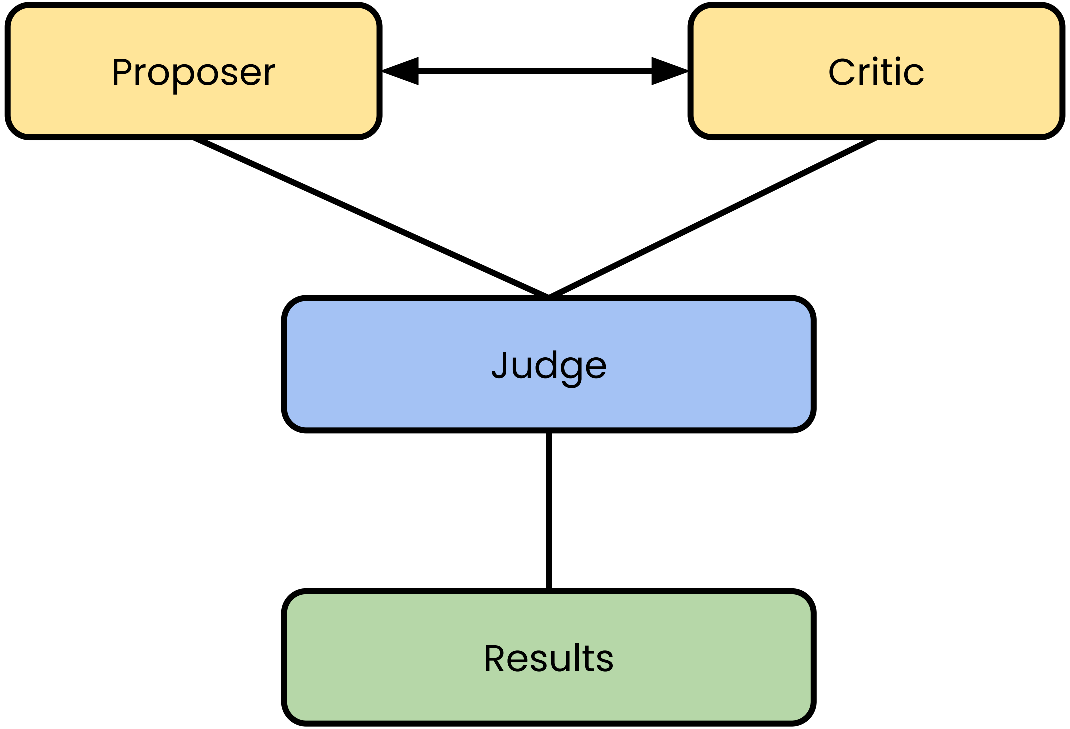
Hierarchical



Control:
Centralized

Best for:
Decomposable tasks

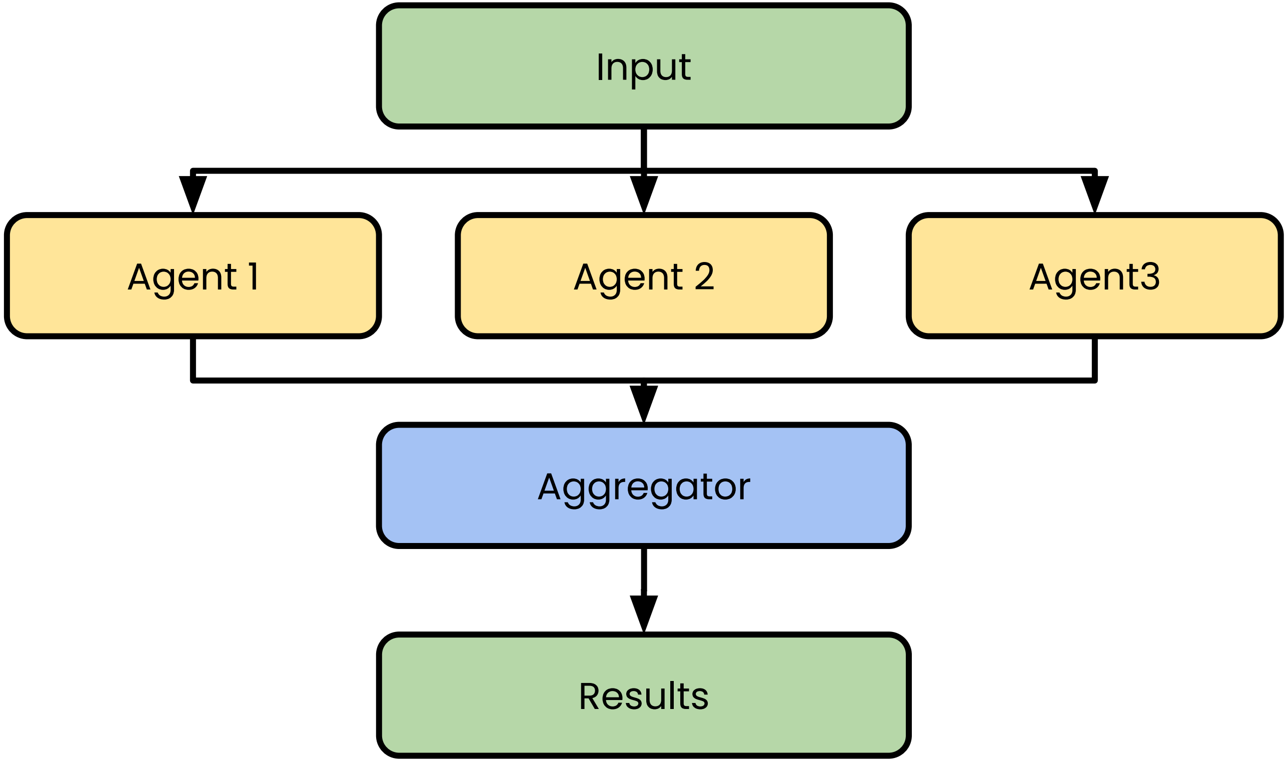
Debate (Adversarial)



Control:
Structured conflict

Best for:
High-stakes decisions

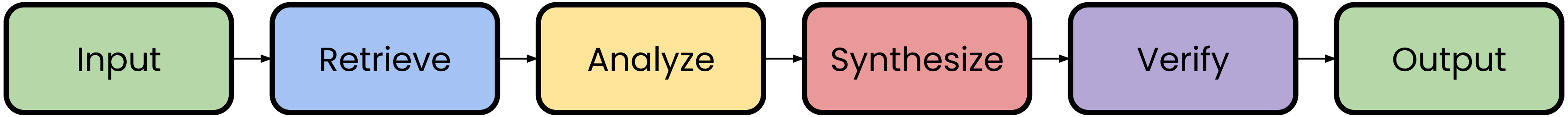
Committee



Control:
Parallel

Best for:
Reliability-critical tasks

Pipeline (Sequential)

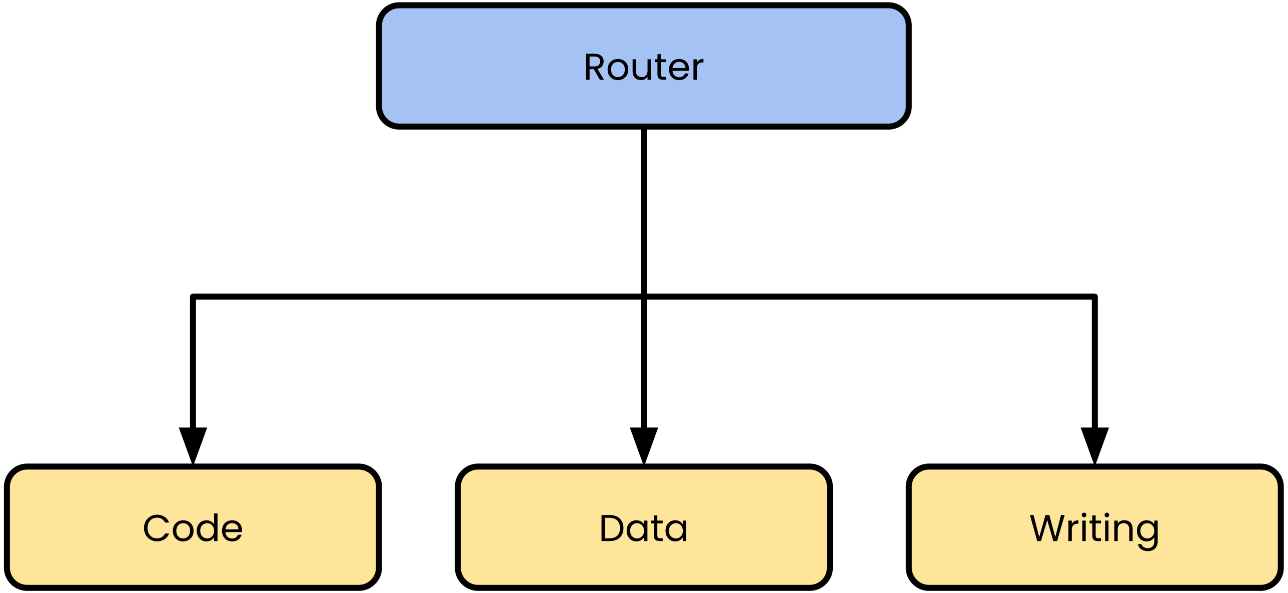


Control:
Sequential

Best for: Staged transformations

(Our first demo uses this pattern)

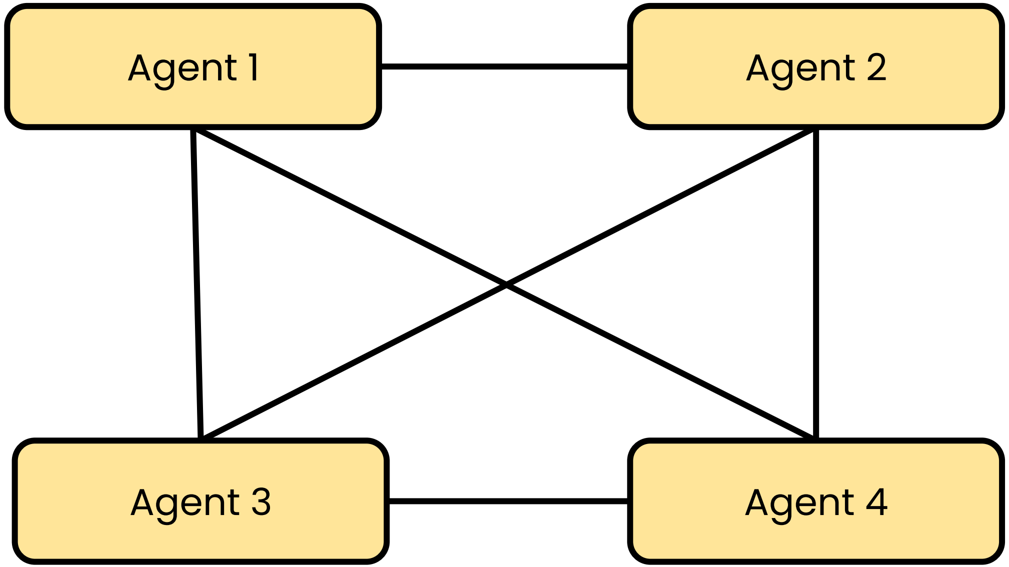
Dynamic Routing



Control:
Adaptive

Best for:
Heterogeneous queries

Network/Swarm



Control:
Emergent

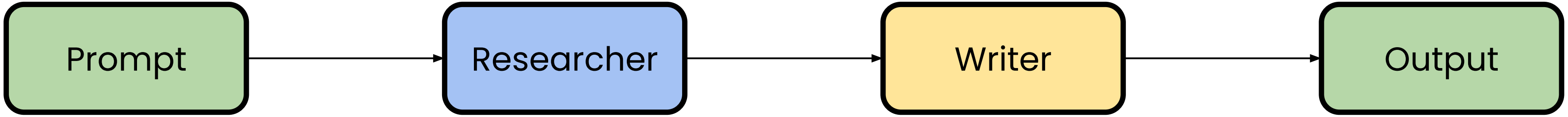
Best for:
Novel
problem-solving

Architectures at a Glance

Architecture	**Control**	**Best For**
Hierarchical	Centralized	Decomposable tasks
Debate	Structured	Nuanced decisions
Committee	Parallel	Performance and reliability
Pipeline	Sequential	Workflows
Routing	Selective	Specialization and complex activities
Swarm/Hive	Emergent	Novel problem-solving

Demo

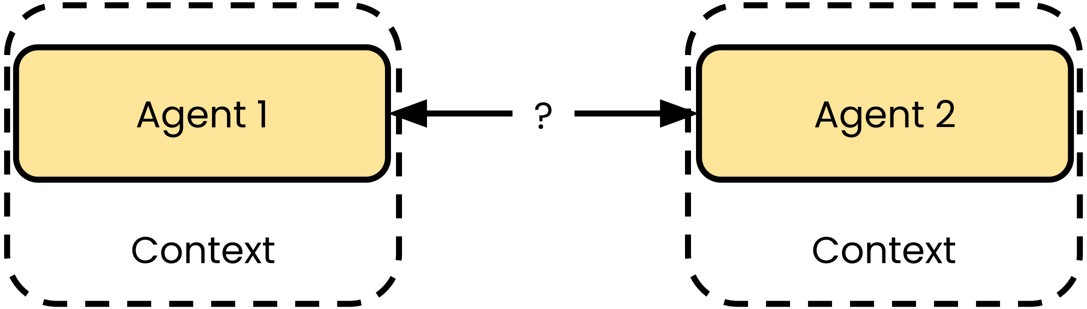
Research and Write Pipeline



Claude-code

Anthropic

Sharing Context and Signals

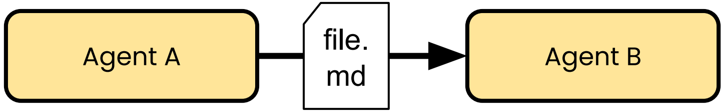


Each agent has its own context window.

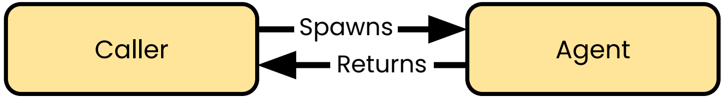
How do they share information?

Four Handoff Patterns

1. File-Based (our demo)

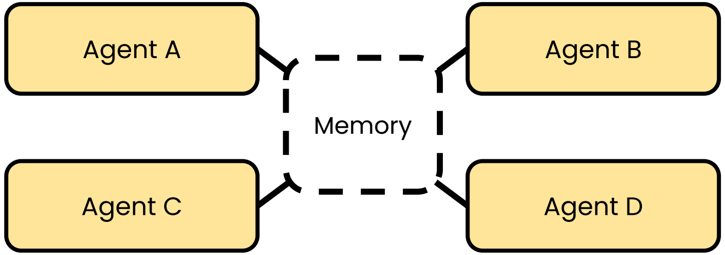


2. Return Value

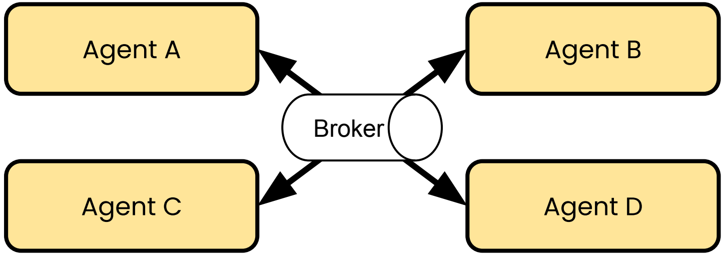


Caller could be an parent agent, an orchestrator, or the main session

3. Shared Memory



4. Message Passing

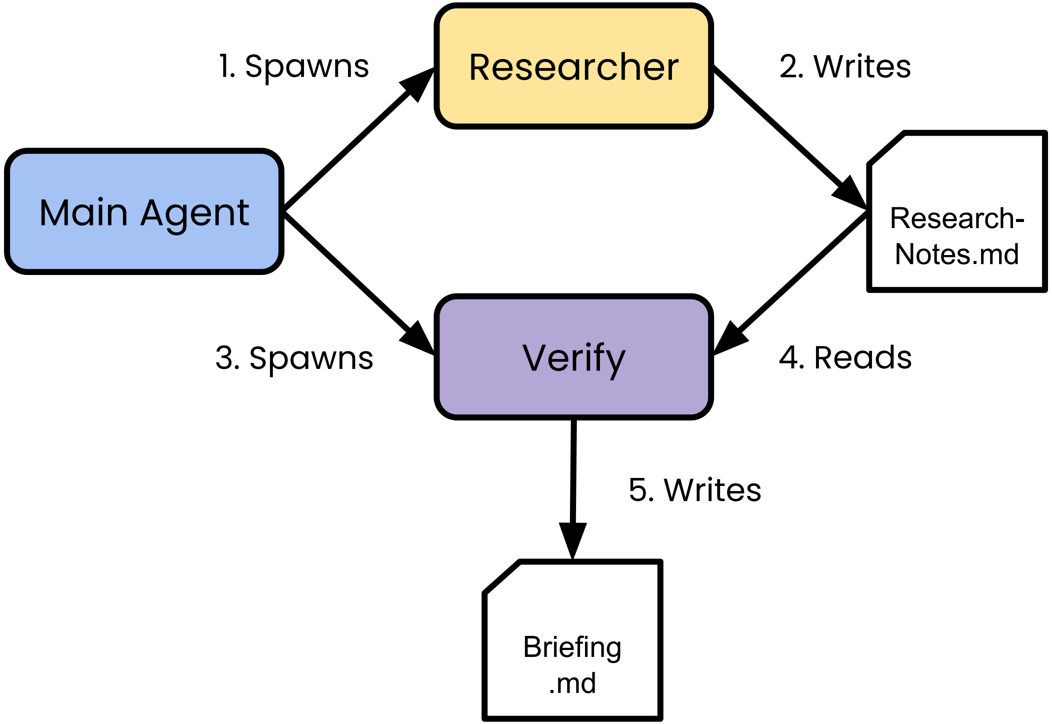


Requires external infrastructure—not native to Claude code

Pattern	Coupling	Debugability	Parallel	Use When
File-Based	Loose	High	No	Clear phases
Return Value	Medium	High	No	Call-Response
Shared Memory	Loose	Med	Yes	Swarms/Parallel
Message Passing	Tight	Low	Yes	Real-time*

* Requires external infrastructure

Our Demo’s Handoff



Main Agent

1. Spawns

Researcher

2. Writes

Research-Notes.md

3. Spawns

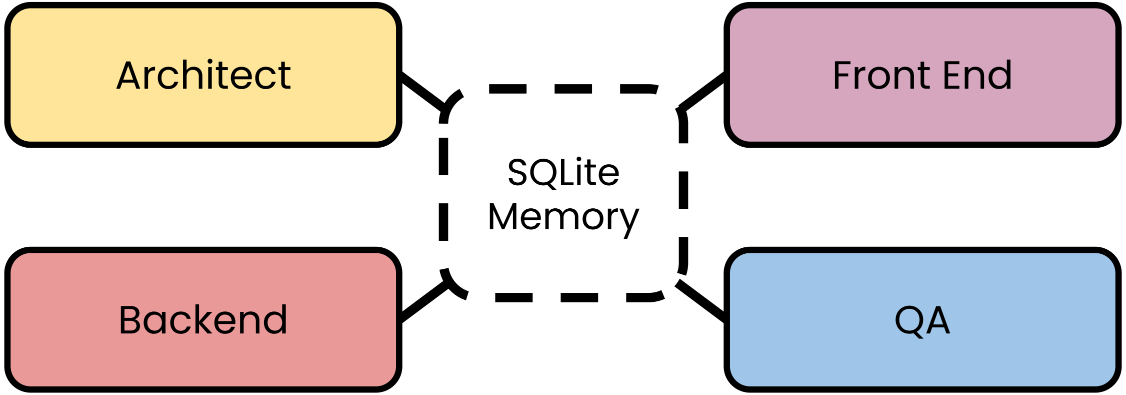
Verify

4. Reads

5. Writes

Briefing .md

Coming in Section 2: Shared Memory Swarms

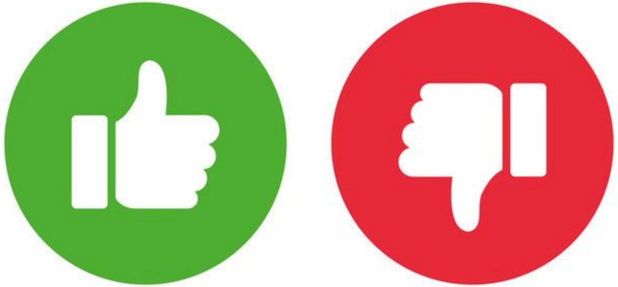


Agents discover and build on each other's work
No explicit handoffs required

Pulse Check

These are a way to quickly check in with attendees. Ask them a simple yes or no question, and the platform will prompt them to press “thumbs up” or “thumbs down”. E.g.

- Does everyone have their Colab environment ready?
- Are you clear on the key architectural differences between our three models?
- Do you feel confident about implementing SLMs in your own projects?



Q&A

Break

O'REILLY®