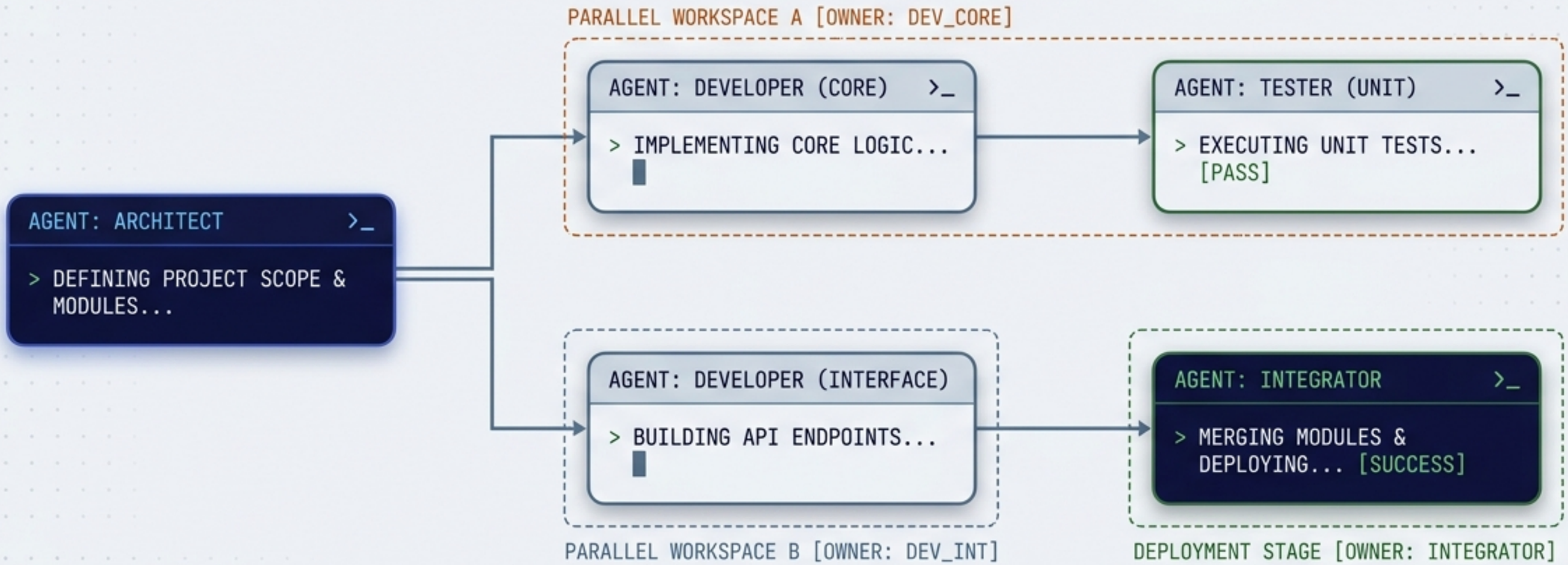
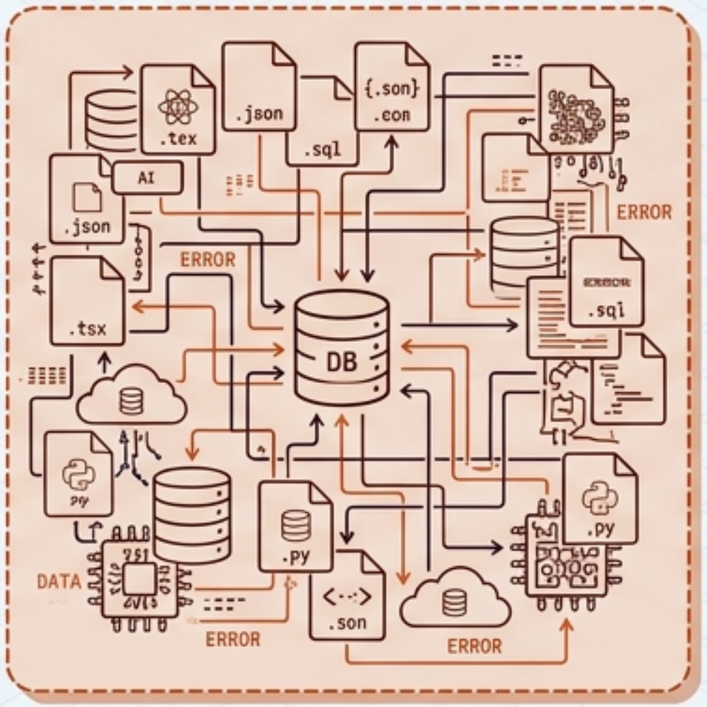
Orchestrating AI Agent Teams

A framework for moving from sequential code generation to parallel project management.



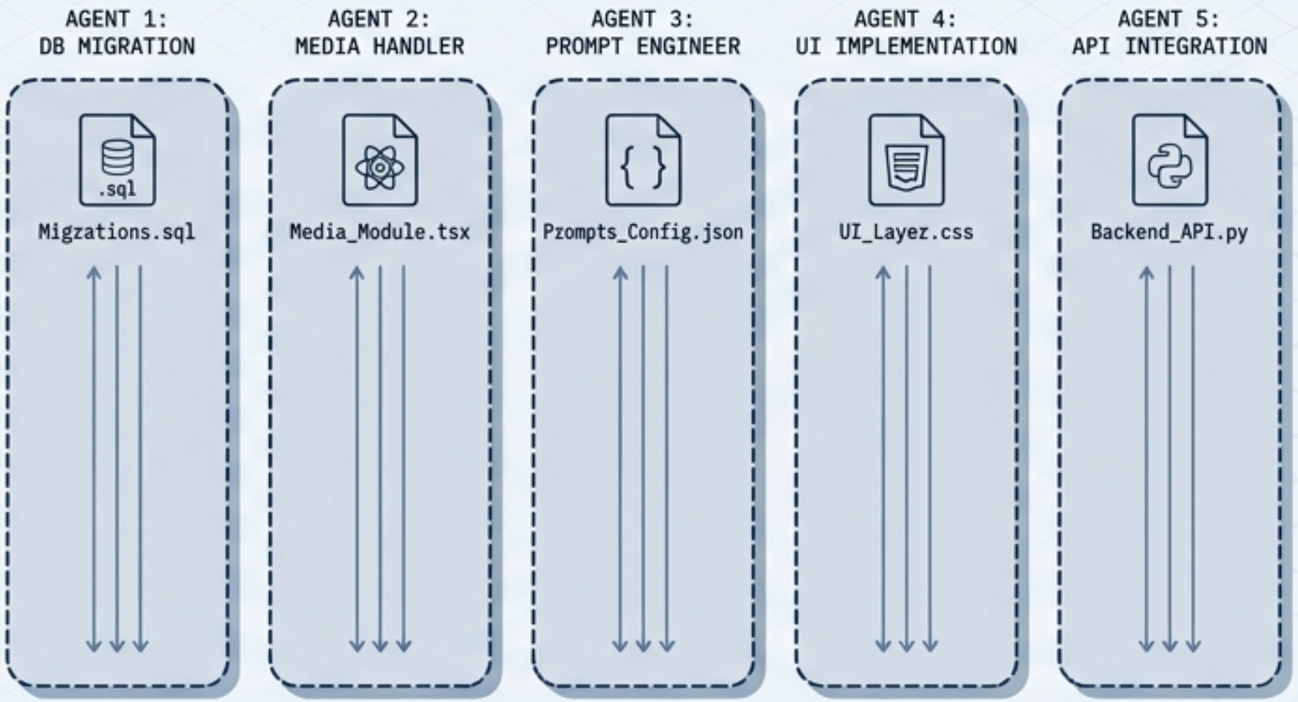
THE SEQUENTIAL PROCESSING LIMIT

SINGLE-AGENT BOTTLENECK



Processing ten complex tasks in series forces a single AI to blow through its context window, degrading logic and recall.

MULTI-AGENT DISTRIBUTION



Genuine parallelism isolates context, allowing simultaneous processing without capacity degradation.

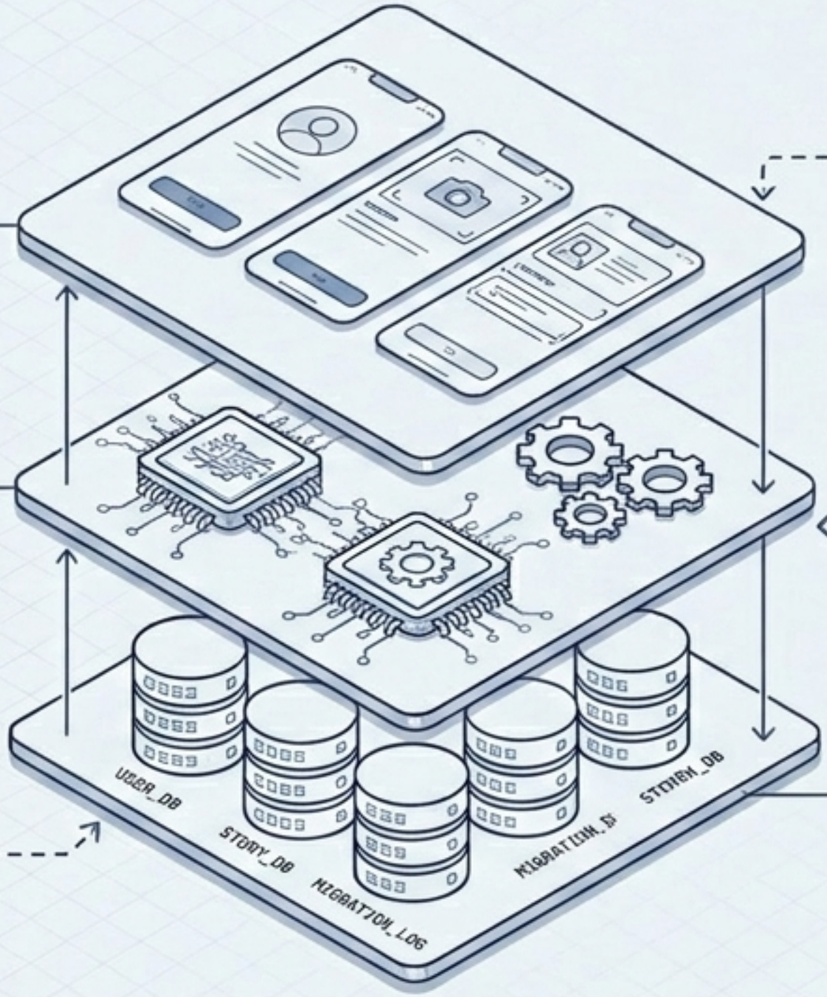
A complex feature touches database migrations, media modules, prompts, and UI layers.
True scaling requires parallel project management.

Architectural Shift: Solo vs. Teams

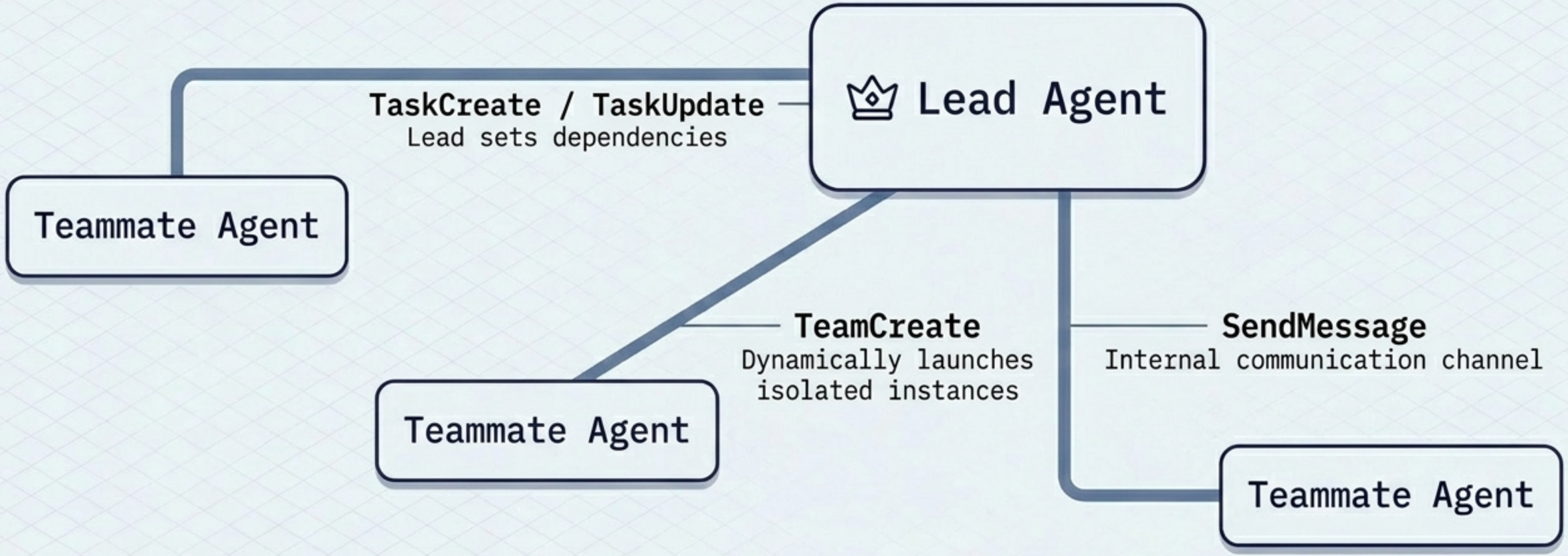
	Sequential Solo AI	Parallel Agent Teams
Context Window	Shared environment, highly prone to overflow	Strictly isolated per individual agent
Execution Speed	Linear, task-by-task grinding	Simultaneous, overlapping parallel commits
File Ownership	Global read/write access across codebase	Strictly sandboxed to assigned directories
Failure Impact	Single point of failure halts the entire chain	Isolated agent crashes with automatic state recovery

The Mio Bot Implementation Blueprint

Upgrading a multi-persona AI bot required touching three architectural layers and over a dozen files.



Core Mechanics of Team Mode

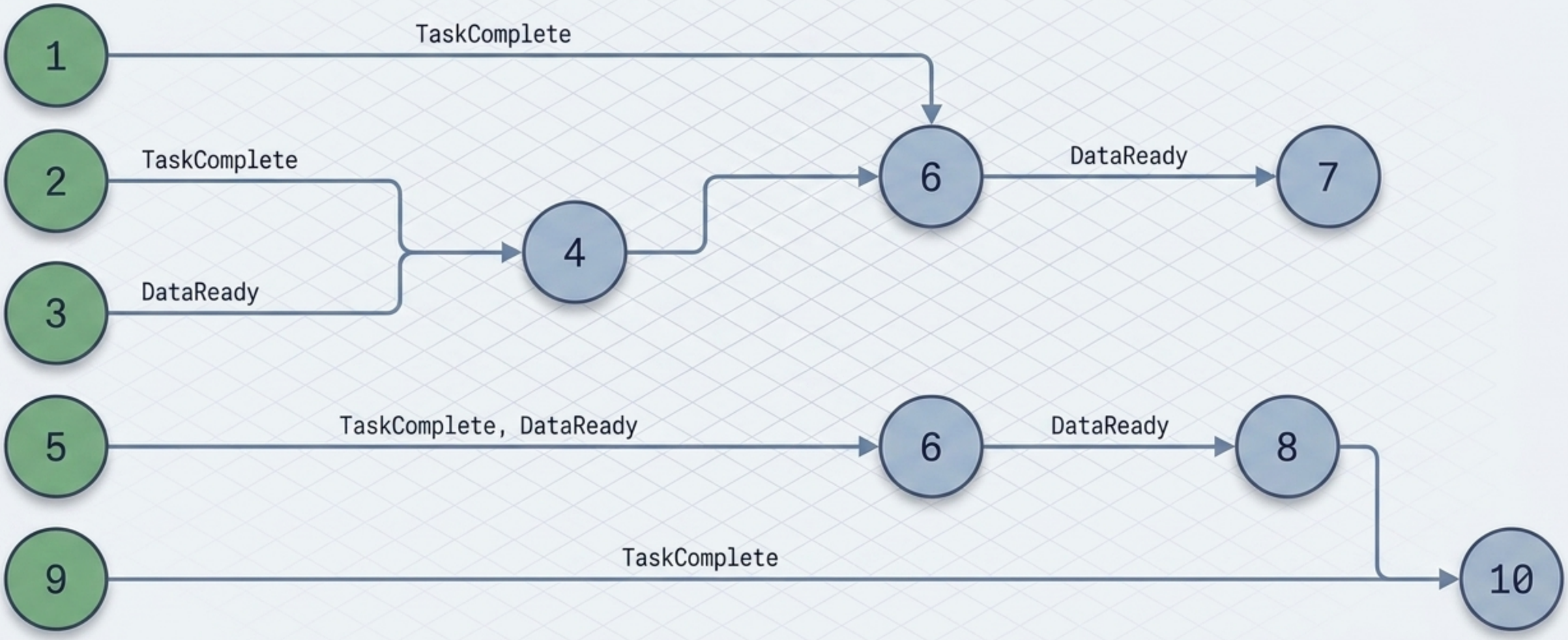


Rule Zero: Strict Sandboxing

Each Teammate is strictly sandboxed to read/write only its explicitly assigned files. Zero crossover allowed.

Decomposing the Plan into a DAG

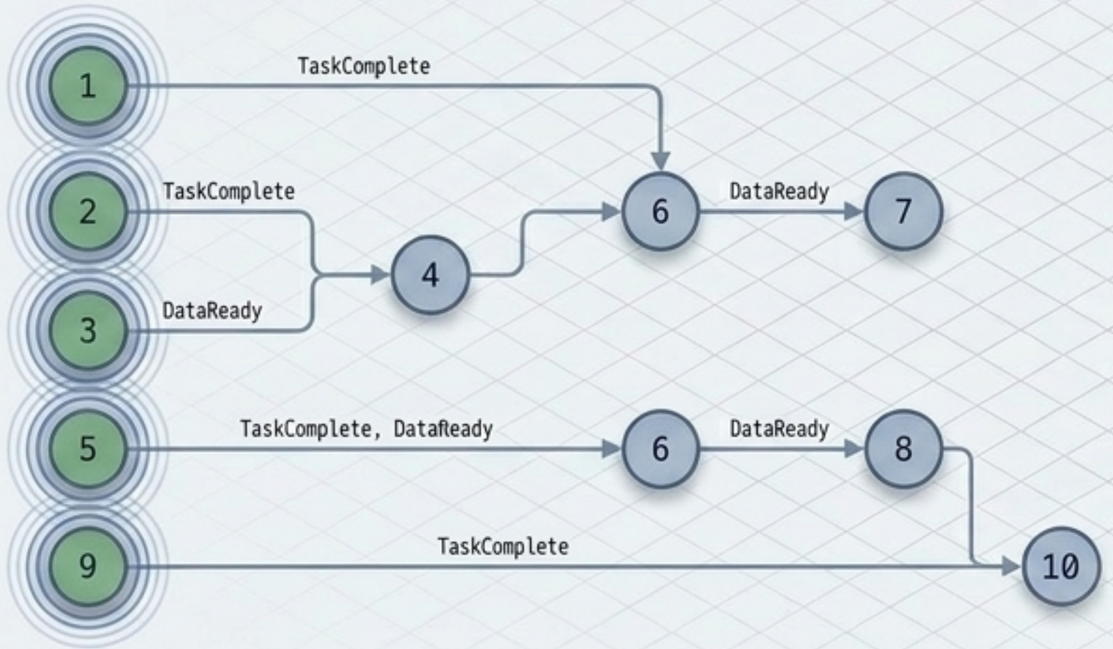
Claude Code automatically maps human intent into a strict Directed Acyclic Graph (DAG), identifying exact task dependencies.



Wave 1: Genuine Parallelism

This is not sequentially chunking a big task. It is five isolated Claude instances, writing and committing code simultaneously.

The Blueprint



The Execution

agent-schema-migrator

```
>> python migrate_schema.py --target=users_v2 ...
... MIGRATION SUCCESS: Applied 3 changes
>>
```

agent-media-builder

```
>> ffmpeg -i input.mp4 -c:v libx264 -crf 23 output.mp4 ...
... PROGRESS: 45%
>>
```

agent-file-downloader

```
>> python file-file-downloader ...
... MIGRATION SUCCESS: Applied 3 changes
>>
```

agent-persona-updater

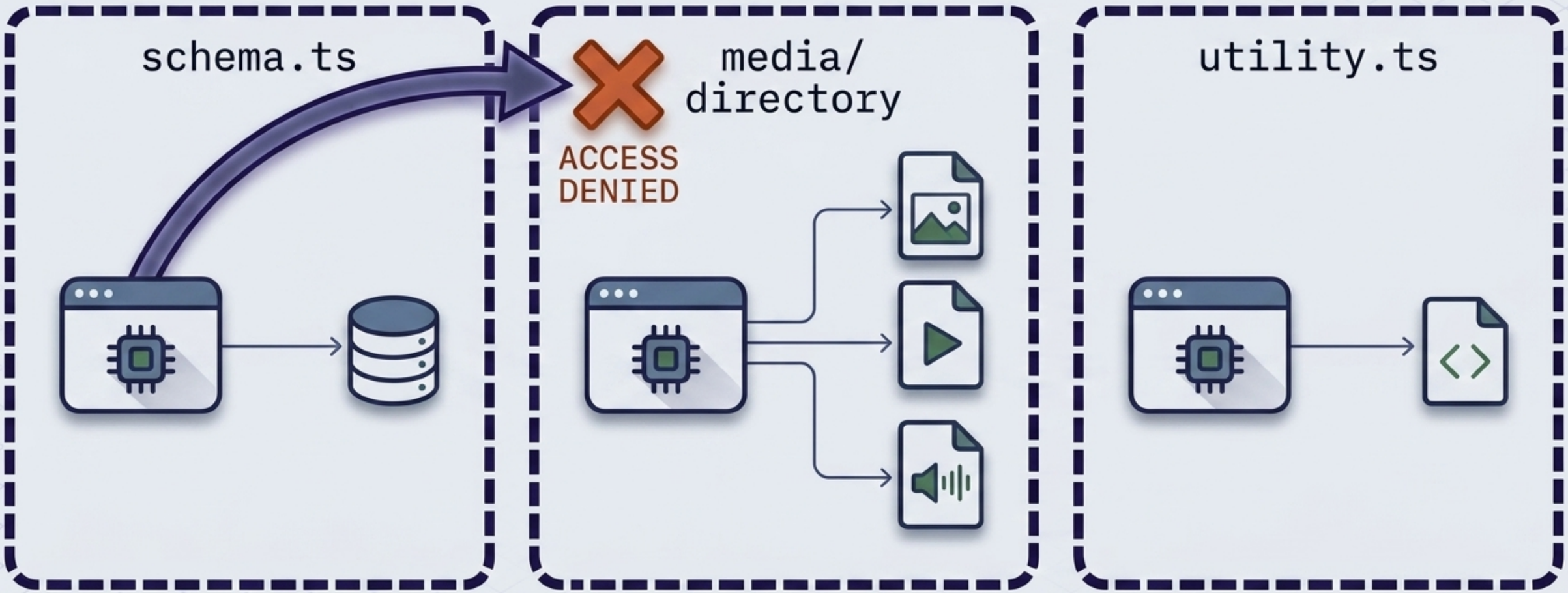
```
>> python persona-updater ...
... MIGRATION SUCCESS: Applied 3 changes
>>
```

agent-selfie-builder

```
>> python selfie-builder ...
... SELFIE SUCCESS: Applied 10 changes
>>
```

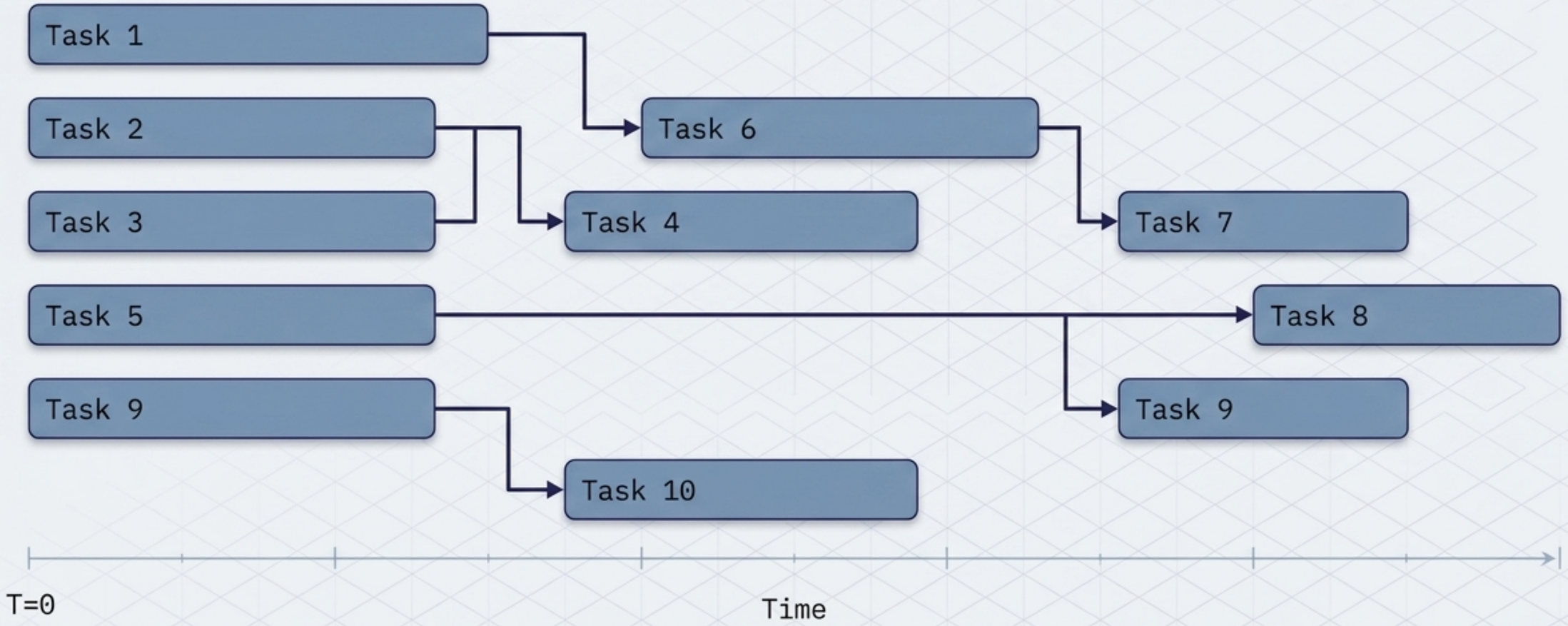
The Secret Weapon: Strict File Ownership

File ownership prevents parallel work conflicts. Each agent receives exact specs and is physically restricted from touching outside files.



Dynamic Scheduling & Dependency Resolution

The Lead Agent continuously tracks the DAG. The moment a dependency clears, it automatically launches a new Teammate. Zero human intervention.



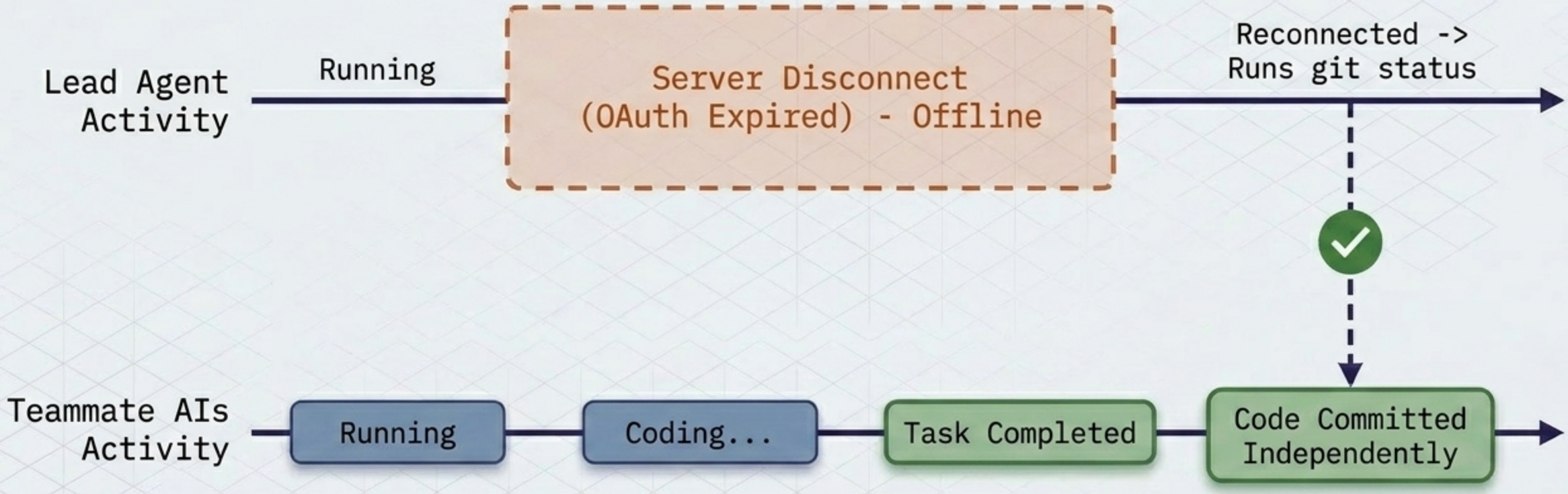
Conflict Resolution: Ignoring the Noise

Parallel work generates friction. When Git hooks fire warnings due to overlapping files, the Lead demonstrates judgment, not brittle rules.

The Lead chooses to ignore false positives rather than halting the system.

Asynchronous Resilience: Surviving Disconnects

Teammates do not rely on the Lead's polling to function. If the server drops, agents finish their tasks and commit code independently.



The Final Integration Pass

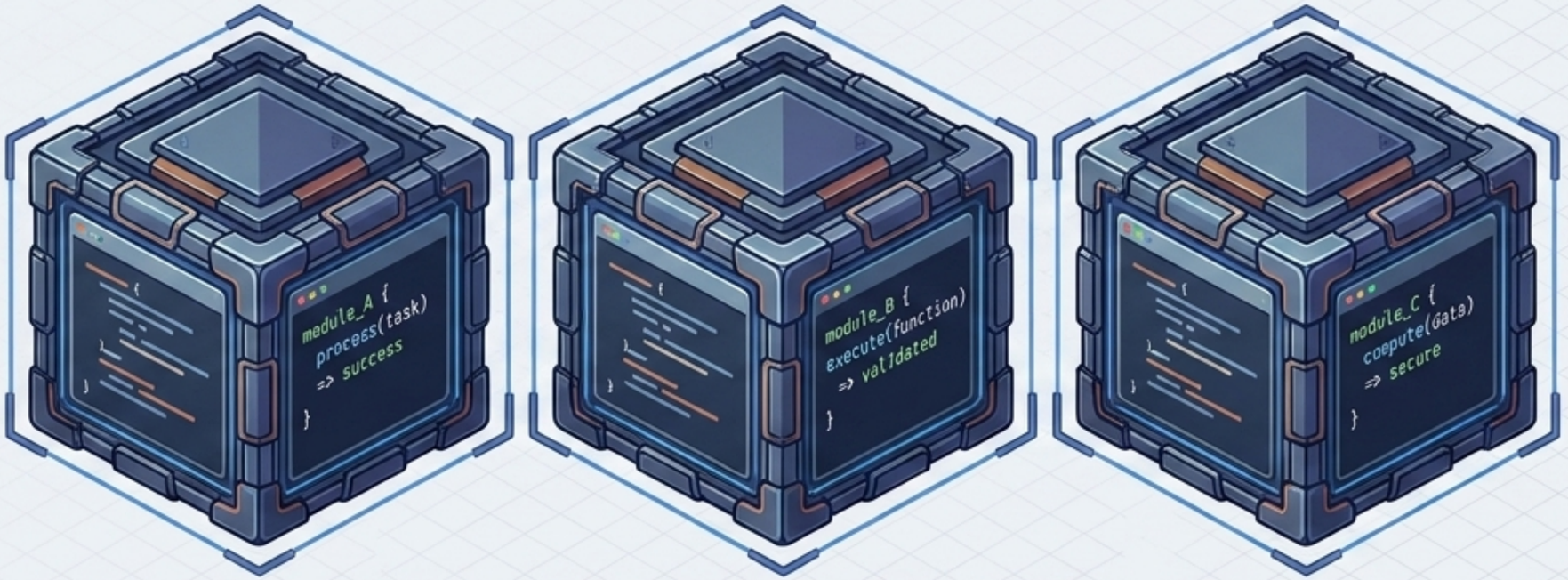
INTEGRATION_TERMINAL -- PASS: SUCCESS

1. **Typecheck Execution**
   Ran pnpm typecheck. Identified two TS18048 errors (ctx.callbackQuery undefined). Implemented direct fixes.
2. **Test Validation**
   Ran pnpm test. Confirmed zero new test failures across all integrated modules.
3. **Code Review**
   Verified all module exports and file existence across parallel branch merges.
4. **Documentation Updates**
   Spawned two final sub-agents to update markdown docs and inline TODOs in parallel.

The Core Insight: Context Isolation > Speed

Just like a human engineering team, successful division of labor means individuals don't need to know the entire codebase—they only need to understand their specific piece.

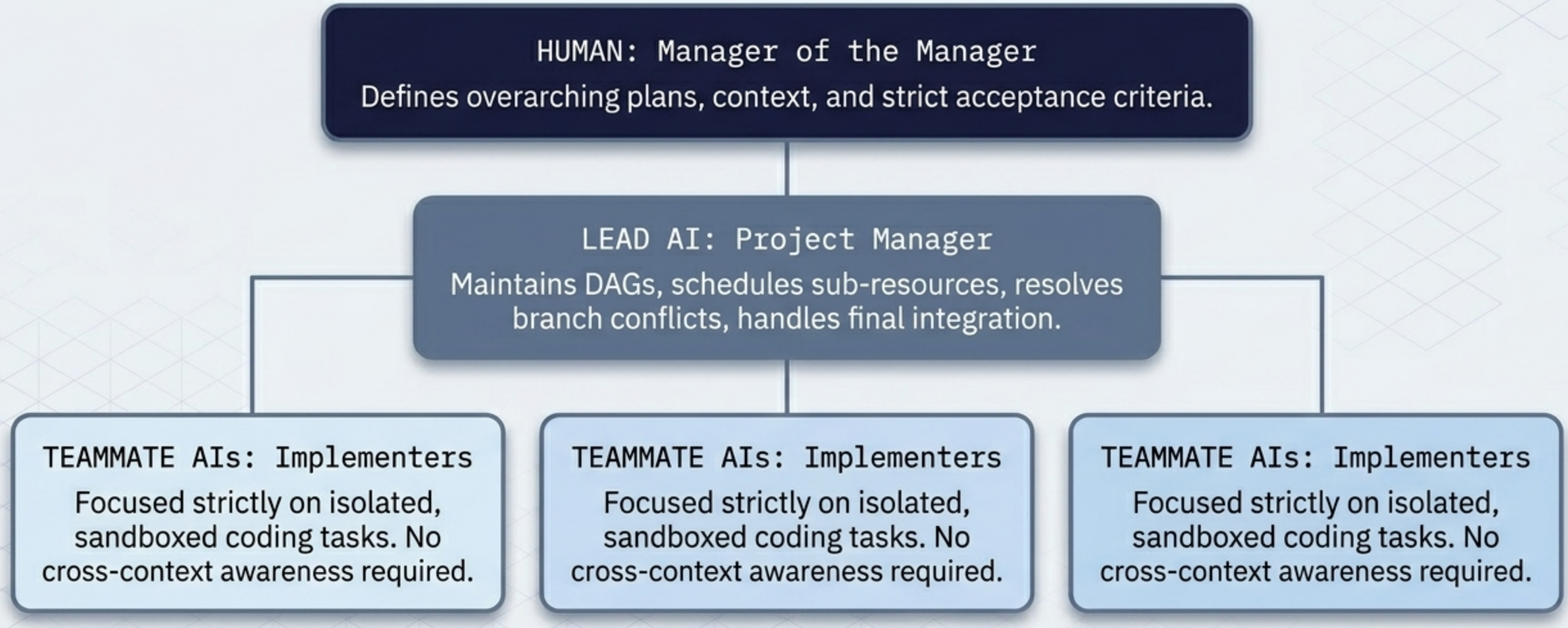
SPEED



CONTEXT ISOLATION

The New Management Hierarchy

You are no longer paired with a digital intern. You are directing a digital project manager.



Decision Matrix: When to Deploy Agent Teams

Deploy Teams When:

- Tasks form a clear **DAG** with 3+ parallelizable nodes.
- Changes span 3+ distinct directories or architectural layers.
- Total project requirements exceed a single agent's **context window**.

Rely on Solo Agents When:

- Changes are strictly isolated to 1 or 2 files.
- Tasks require mandatory, step-by-step sequential execution.
- Work requires deep, shared state or continuous interactive debugging (e.g., simulation loops).

The New **Baseline for Output**

1	10	5	3
Plan	Tasks	Autonomous AIs	Waves of Execution

Dozens of files modified across three architectural layers. Zero human code reviews during the session. That is the power of orchestrating Agent Teams.