AGENT FRAMEWORKS

LangGraph deep dive & the multi-agent framework landscape

state · reducers · supervisor routing · persistence · human-in-the-loop

FOUNDATIONS

Why agents need explicit state

Plain while-loop

- State scattered in the call stack
- Inputs, trajectory, outputs — implicit
- Nothing to inspect, checkpoint, or resume

LangGraph state schema

- Formalizes exactly that list
- A declared, typed schema per graph
- Foundation for reducers, checkpoints, replay

FOUNDATIONS

LangGraph is a strict superset of ReAct

ReAct

LLM freely sequences
any tool, any order

restriction is opt-in

LangGraph

Designer pre-wires nodes
& edges — fully connected
graph = ReAct exactly

Opting into restriction is what buys inspectability, checkpointing, and audit — not a trade against flexibility.

STATE MECHANICS

Reducers: a per-key merge policy

- Default reducer: overwrite
- Concurrent writes to the same key, no reducer declared →

InvalidUpdateError — fails loudly

Design rule

- Overwrite-keyed fields need topology guarantees (at most one writer per step)
- ...or an explicit custom reducer that declares how to merge

STATE MECHANICS

Two reducers in practice

`operator.add`

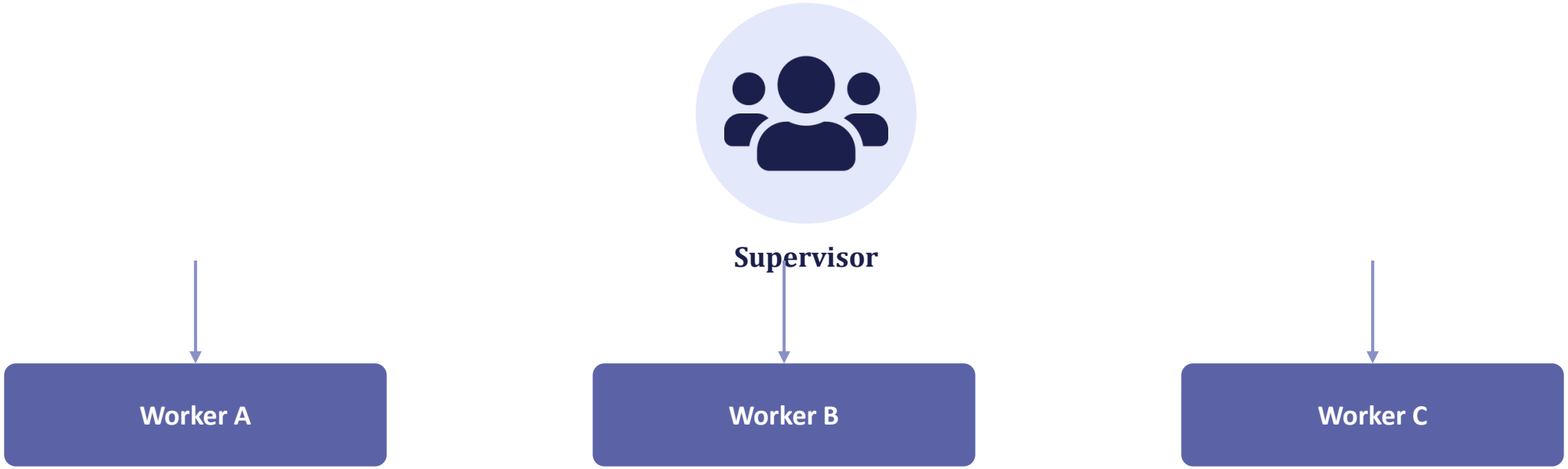
- Type-generic: sums ints, concatenates lists
- Just Python's + operator under the hood
- Fails on dicts — + isn't defined for them

`add_messages`

- Upsert-by-id, not plain append
- Same id → replaces in place; new id → appends
- Enables streaming tokens & message edits
- Gap: same-id concurrent edits in one step = last-write-wins

COORDINATION

Supervisor pattern: design-time vs. runtime



- Engineer fixes at design time: which nodes exist, which edges are reachable
- Supervisor LLM decides at runtime: which reachable path to take, per request

COORDINATION

Command(goto, update): routing in one return

```
Command(goto="supervisor", update=result)
```

- Replaces separately-registered conditional-edge functions
- One return value bundles the routing decision and the state write
- goto carries next-hop signaling — state stays purely semantic

END

Reserved graph target, not a real node. Final state becomes the caller's return value.

PERSISTENCE

Checkpointer: durable pause & resume

- Persists graph state at every step
- Process crashes or restarts → resume from the last completed step, not from scratch
- The alternative: hand-rolling serialization, crash survival, and concurrent-waiter handling yourself

Scope: single thread / conversation

PERSISTENCE

Store: memory across threads

- Checkpointer = one thread's execution state
- Store = data that must persist and be shared across threads
- Example: long-term user memory that outlives any single conversation

OVERSIGHT

Human-in-the-loop: interrupt()

Pauses execution mid-node for human approval, resumed with Command(resume=...)

The double-fire gotcha

On resume, LangGraph re-executes the entire node from the top — any code placed before the interrupt() call runs again. Non-idempotent side effects before an interrupt will fire twice.

LANDSCAPE

CrewAI vs. LangGraph vs. AutoGen

CrewAI

Fixed DAG,
role-mapped pipeline

Closer to pre-wired topology

LangGraph

Opt-in restriction,
durable, auditable

Fixed topology + runtime routing

AutoGen

Free-form
multi-agent conversation

No supervisor-worker graph

LANDSCAPE

Why LangGraph pulled ahead in 2026

vs. CrewAI

More flexibility — restriction is opt-in, not baked into a fixed DAG

vs. AutoGen

More auditability — topology fixed by the designer, every path traceable in advance

AutoGen has moved into Microsoft maintenance mode; interview signal has shifted from framework-picking to state-graph mechanics.

AGENT FRAMEWORKS — RECAP

State & reducers

explicit schema, per-key merge policy

Supervisor pattern

design-time topology, runtime routing

Checkpointer

thread-scoped, crash-resumable

Store

cross-thread memory

Human-in-the-loop

interrupt() — beware double-fire

Landscape

flexible + auditable vs. CrewAI / AutoGen