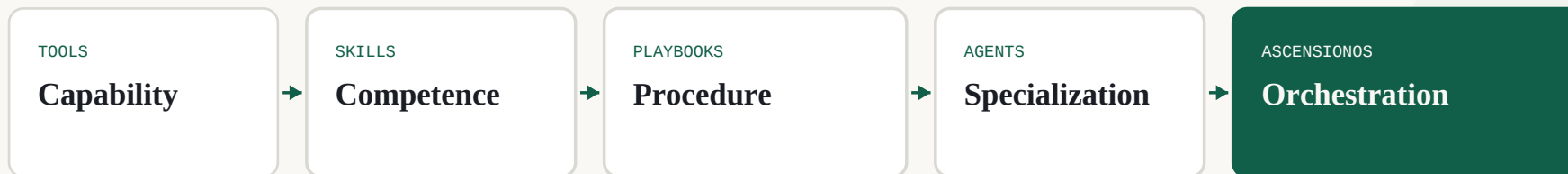


Skills, playbooks & graph orchestration.

A production model for AscensionOS: reusable competencies become playbooks; playbooks equip specialist agents; graph orchestration routes work through the smallest necessary context.



Tools → Skills → Playbooks → Agents → AscensionOS

Capability becomes reliable execution through explicit composition.

One canonical stack. No duplicated intelligence.

Playbooks reference canonical skills. Agents select them; they do not carry private copies.

01

Tools

GitHub · terminal · browser · APIs

02

Skills

Reusable procedures with verification

03

Playbooks

Sequences · branches · bounded loops

04

Agents

Specialists with selected capabilities

05

Agent OS

Policy · routing · memory · permissions

06

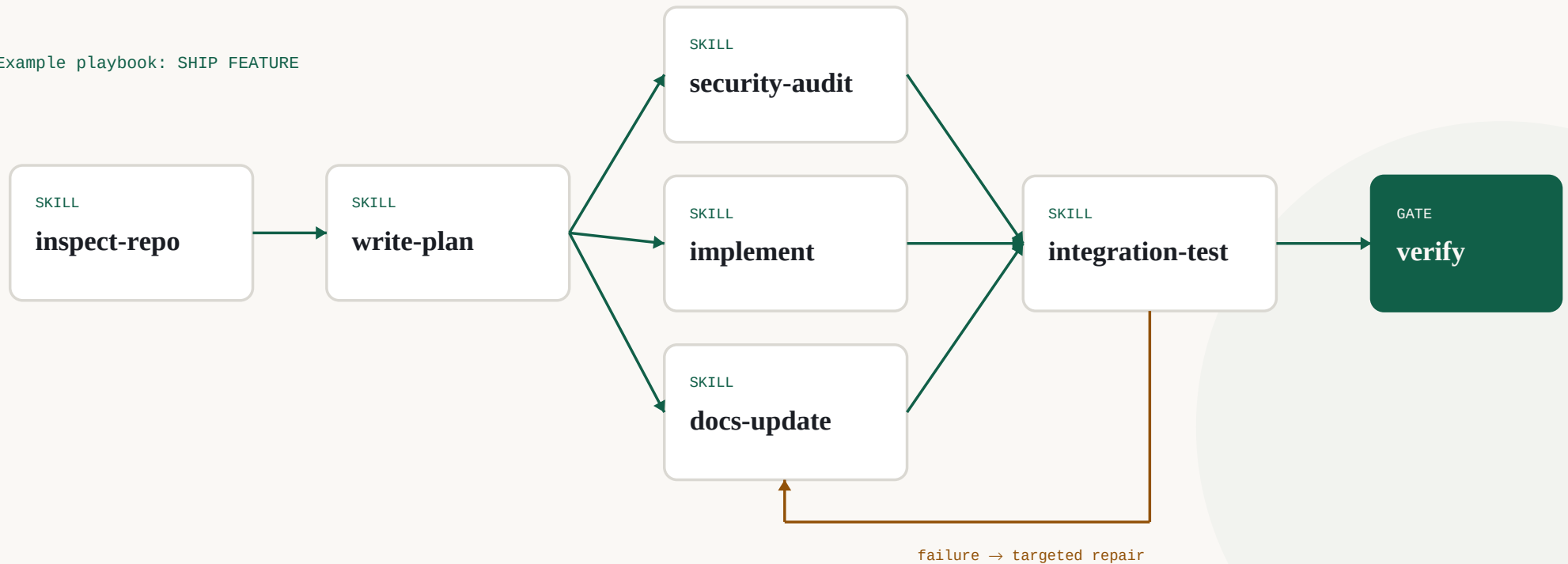
AscensionOS

Classify · route · state · reconcile

Skills are atoms. Playbooks are compositions.



Example playbook: SHIP FEATURE



An agent is a configured worker, not a giant prompt.

The agent OS determines what the specialist can do, how it works, what context it sees, and how success is proven.

SKILLS

architecture · coding · testing

TOOLS

GitHub · terminal · filesystem

SPECIALIST

Software Engineer

implementation quality

PLAYBOOKS

ship-feature · fix-bug · refactor

MODEL POLICY

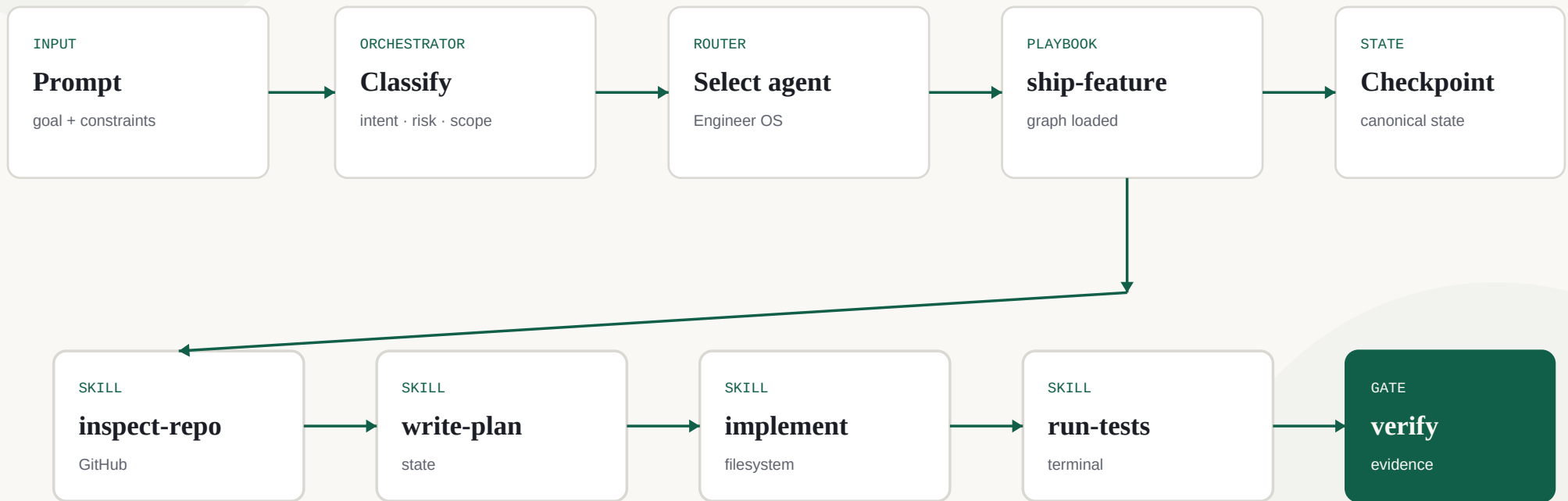
cheap explore · reason · verify

POLICIES

inspect first · smallest diff · proof

How a prompt becomes executed work.

User: “Build authentication for the dashboard, preserve the existing design system, test it, and document the change.”



CONTEXT POLICY

Load only the current node's skill instructions, relevant project state and permitted tools. Keep everything else out of working context.

Result: explicit ownership + bounded retries + mechanical verification.

Every graph node gets a small contract.

This is the layer that turns graph orchestration from prompt spaghetti into production infrastructure.

OBJECTIVE

Implement dashboard authentication.

INPUT STATE

Approved plan • architecture constraints.

SKILLS

implement-code • secure-auth-patterns.

TOOLS

filesystem • terminal • package manager.

MODEL ROUTING

Coding model; escalate on architecture ambiguity.

BUDGET

1 pass + 2 targeted repairs.

ACCEPTANCE

Auth tests + regression tests + no secrets.

OUTPUT

Patch • evidence • changed files • risks.

North star: minimum sufficient context, capability, orchestration and verification.