Superpowers: brainstorming, plans, TDD & review

Key takeaways

1. Superpowers scales its ceremony to the task; approval never scales away.
2. Specs, plans, tests, and reviews carry intent across context changes.
3. The human owns the gates; the agent produces the evidence.

One vague feature hides four policy decisions

Starting with code lets an implementation become the accidental specification

The request

```
Add retries to client.py.
Use exponential backoff.
```

- Which failures are transient enough to retry?
- Does the limit count attempts, elapsed time, or both?
- Which operations are safe to repeat after an uncertain response?
- What failing observation would prove the behavior is absent?

Superpowers makes the agent settle intent before implementation hardens it.

Install per harness; skills select themselves

After restart, the task triggers relevant skills without a command sequence to memorize

Current official installation paths

```
Claude Code
  /plugin install superpowers@claude-plugins-official

Codex App
  Plugins > Coding > Superpowers > +

Codex CLI
  /plugins > search Superpowers > Install Plugin
```

Install separately in each harness, restart, and keep evolving skills out of repositories.

source: obra/superpowers README @ v6.3.0; accessed 2026-08-19

The first skill scales ceremony to the task

A spike, bounded change, and architectural change earn different artifacts

Spike

Answer a feasibility question as cheaply as correctness allows. Any code is explicitly throwaway.

Output: recommendation

Bounded

Change an existing flow. Present a short design in chat, get approval, then implement without spec or plan files.

Output: approved chat design

Architectural

Add a subsystem or reshape interfaces. Compare approaches, write a spec, get review, then write the plan.

Output: approved spec + plan

Hidden complexity upgrades the path; it never downgrades it.

source: obra/superpowers skills/brainstorming/SKILL.md @ v6.3.0

Ceremony scales; approval does not

Every path stops at one explicit human decision before implementation

Path	Agent brings	Human decides	Then
Spike	probe question	worth investigating?	throwaway experiment
Bounded	short chat design	is this the right change?	TDD implementation
Architectural	sectioned design + spec	approve interfaces?	plan + isolated work

A shorter artifact is not permission to skip the human gate.

source: obra/superpowers skills/brainstorming/SKILL.md @ v6.3.0

A plan must survive fresh context

The next agent should not need the brainstorming transcript to act correctly

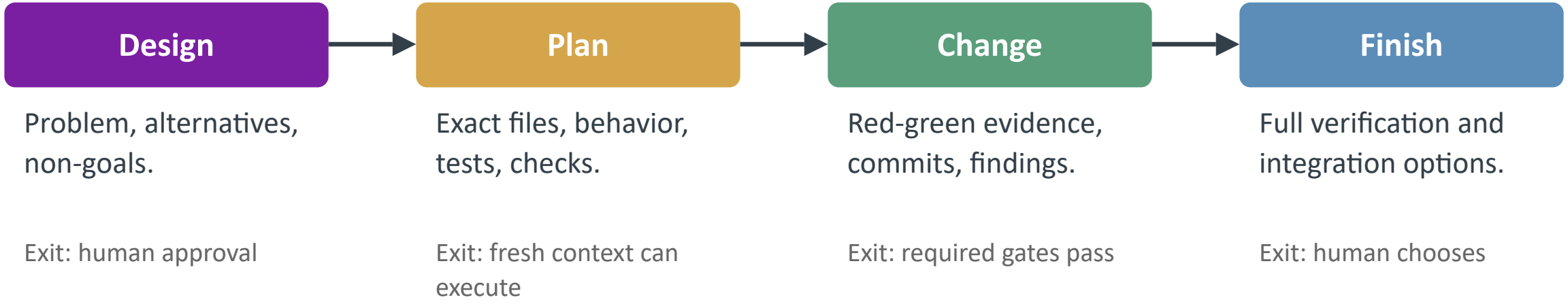
```
Spec: docs/superpowers/specs/2026-08-19-retry-policy-design.md

Task 3: Retry only idempotent requests
Files:
  - modify: src/client.py:44
  - test: tests/test_client.py

1. Add a failing POST-does-not-retry test.
2. Run the focused test; verify the expected failure.
3. Implement the smallest method guard.
4. Re-run the focused test, then the full gate.
5. Commit only these files.
```

A task names the governing spec, files, failing proof, passing proof, and stop point.

source: obra/superpowers writing-plans skill @ v6.3.0

Artifacts carry intent across context changes

Each handoff has a named input and a falsifiable exit condition



The durable product is decisions and evidence that fresh context can read.

source: obra/superpowers README — The Basic Workflow @ v6.3.0

Task coupling chooses the execution topology

Both paths run the approved plan; their coordination costs differ

Subagent-driven

Dispatch fresh implementers from task briefs. Review spec compliance first, then code quality. Small same-shape tasks may share one dispatch.

Best for disjoint or review-sensitive tasks

Executing plans

One session runs task batches with human checkpoints. Shared context reduces handoff cost when tasks depend on one another.

Best for tightly coupled tasks

Parallelism is secondary: preserve the dependencies that make the plan correct.

source: obra/superpowers README and v6.3.0 release notes — execution skills

Red-green-refactor is evidence, not theatre

Writing a test after the code cannot prove that the test detects absence

```
uv run pytest tests/test_client.py::test_post_does_not_retry -q
# FAIL: expected calls == 1, observed 3

# Make the smallest implementation change.
uv run pytest tests/test_client.py::test_post_does_not_retry -q
# PASS

uv run pytest -x && ruff check && pyright  # full gate: tests, lint, types
```

Red proves the test can detect the missing guard; green proves this change supplies it.

source: obra/superpowers skills/test-driven-development/SKILL.md @ v6.3.0

Review separates three decisions

Correctness to the spec and quality of the implementation are separate gates

1. Spec

Compare the diff with the approved design and plan. Find missing requirements and unintended behavior.

Question: does the diff match the spec?

2. Quality

Inspect correctness, tests, maintainability, and risk. Critical and important findings block progress.

Question: is the implementation sound?

3. Finish

Re-run the full gate, then present merge, PR, or keep. Integration remains a human decision.

Question: integrate or preserve?

A green suite cannot prove that the wrong feature is the right feature.

source: obra/superpowers review and finish skills @ v6.3.0

The human owns the irreversible decisions

Automation can enforce a workflow without proving that it fits the task

Your Responsibility	Agent's Responsibility
Approve the path, design, and non-goals	Invoke the relevant skills
Resolve product and interface tradeoffs	Produce durable, reviewable artifacts
Choose execution topology	Follow red-green-refactor
Choose merge, PR, or keep	Return test and review evidence

Key Takeaways

1. Superpowers classifies the task first; even its shortest path stops for human approval.
2. Specs, plans, tests, commits, and reviews let fresh context continue without reconstructing intent.
3. Test order, two-stage review, and the final integration choice make completion claims falsifiable.