Vector Society Brief

Five Builds, One Auditor

A local-only workflow for parallel multi-agent implementation

---

The premise

Spawn N agents to independently implement the same function. Spawn one more agent to audit the results and pick a winner. Keep every variant locally — nothing leaves the machine until you choose. The challenge is mechanical: how do you let five agents commit freely, in parallel, without stepping on each other or polluting a shared remote?

The pattern

Use **git worktrees**, one per agent, each on its own local branch. A worktree is a second working directory backed by the same *.git* object store — five real folders on disk, one shared history, zero remote traffic.

```
~/project/                # main checkout (you / the auditor)
~/project-agent-1/        # worktree on branch agent/1-impl
~/project-agent-2/        # worktree on branch agent/2-impl
~/project-agent-3/        # worktree on branch agent/3-impl
~/project-agent-4/        # worktree on branch agent/4-impl
~/project-agent-5/        # worktree on branch agent/5-impl
```

Setup (one-time per round)

```
cd ~/project
for i in 1 2 3 4 5; do
  git worktree add -b agent/$i-impl ../project-agent-$i main
done
```

Each agent is launched with its own working directory. They run truly in parallel — no checkout-switching, no stash juggling, no file-level collisions.

Audit step

The auditor doesn't need its own worktree. It reads diffs against *main*:

```
for i in 1 2 3 4 5; do
  echo "=== Agent $i ==="
  git diff main..agent/$i-impl -- path/to/function.py
done
```

For head-to-head comparison of two implementations, *git range-diff* shows how the same function evolved differently in each branch.

Selection & teardown

```
git checkout main
git merge agent/3-impl          # or cherry-pick specific commits
for i in 1 2 4 5; do
  git worktree remove ../project-agent-$i
  git branch -D agent/$i-impl
done
```

Still nothing pushed. Losing branches simply disappear from the local repo.

Why this beats the alternatives

Approach	Problem
Five stashes	No isolation. Agents collide on the working tree. Diffing side-by-side is awkward.
Five branches, one checkout	Agents must serialize on checkout. Defeats the parallelism.
Five patch files	Fine for handoff, but you can't run or test each variant in place.
Five forks of the repo	Heavyweight. Duplicates the object store. Network round-trips for what should be local.
Five worktrees (this brief)	One object store, five live working trees, true parallelism, local-only.

Native harness integration

Claude Code's *Agent* tool accepts **isolation: "worktree"**. Each subagent spawned with that flag gets its own auto-created worktree; worktrees that produce no changes are auto-cleaned. The orchestrator never has to script the worktree dance — pass the flag, let the harness handle the bookkeeping, and the auditor agent reads the surviving branches as siblings in the object store.

Takeaway

- **Parallelism is a filesystem problem, not a git problem.** Worktrees give every agent its own filesystem at near-zero cost.
- **The remote is a publishing step, not a collaboration step.** Five agents collaborate fine through one local *.git* directory.
- **Selection is reversible until you push.** Branches are cheap; deleting four of five costs nothing.
