Phase 1 plan

Here's a crisp, end-to-end plan for Phase 1 — Two-Agent Loop (All Mocks) that maps straight onto your repo and promotion rules.

Phase 1 — Two-Agent Loop (All Mocks)

1) Objective

Prove a typed, traceable, two-agent debate loop locally (no real model calls). Agents exchange structured messages (propose → implement → critique → verify**) via an in-memory bus; the loop converges in ≤3 turns and produces a mock diff.**

2) Scope & Non-Goals

In scope

- **JSON message schema + TS types**
- **In-memory event bus**
- **Deterministic loop exits (converged / maxTurns / failure)**
- **CLI that prints a diff and** CONVERGED|FAILED
- **Unit tests + CI**

Not in scope (Phase 2+)

- **Real provider adapters (Anthropic/OpenAI)**
- **Git PR automation**
- **RAG, review gates, policy engine**

———

3) Deliverables & Repo Changes

- **Schema**
- apps/orchestrator/src/schema/messages.ts **(TS types)**
- apps/orchestrator/src/schema/message.schema.json **(JSON Schema 2020-12)**
- **Loop & Bus**
- apps/orchestrator/src/bus/inMemoryBus.ts
- apps/orchestrator/src/kernel.ts **(loop logic & exits)**
- apps/orchestrator/src/agents/*.mock.ts **(architect/builder)**
- **CLI**
- apps/orchestrator/src/cli/pocMock.ts **(prints diff + status; proper exit code)**
- **Scripts**
- apps/orchestrator/package.json**:** "poc:mock"**,** "typecheck"
- **Root** package.json**:** "smoke:p1": "npm --workspace apps/orchestrator run poc:mock"
- **Tests**
- apps/orchestrator/tests/kernel.spec.ts
- apps/orchestrator/tests/poc.spec.ts **(fail & maxTurns)**
- **Docs**
- docs/architecture/phase-1-plan.md **(this plan)**
- **CI**
- .github/workflows/ci.yml**: add** typecheck **+** smoke:p1

———

4) Protocol (message contract)

TS Types (apps/orchestrator/src/schema/messages.ts)

```
export type MessageType = 'propose' | 'critique' | 'implement' | 'verify';
export type Role = 'architect' | 'builder';
export interface Message {
  role: Role;
  type: MessageType;
  content: string;
  turn: number;
  reasons?: string[];       // optional rationale bullets
  evidence?: string[];      // optional file/line refs
  risks?: string[];         // optional risk calls
  budgetTrace?: { inputTokens?: number; outputTokens?: number };
}
export interface DebateResult {
  status: 'CONVERGED' | 'FAILED';
  turns: number;
  diff?: string;
```

```
  log: Message[];
}
```

JSON Schema (apps/orchestrator/src/schema/message.schema.json)

- **Draft 2020-12**
- **Required:** role**,** type**,** content**,** turn
- **Disallow additional properties (keeps mocks tight)**

———

5) Loop Algorithm (deterministic)

Pseudocode

```
diff = ""
lastCritique = null
for turn in 1..maxTurns:
  if turn == 1: bus.publish(architect.propose(task))
  else:       lastCritique = architect.critique(diff); bus.publish(lastCritique)
  impl = builder.implement(task, lastCritique?.content); diff = impl.diff; bus.publish(impl.msg)
  verify = architect.verify(diff); bus.publish(verify)
  if verify says "Verified": return CONVERGED with diff + log
return FAILED with diff + log
```

Exit conditions

- **Converged:** verify **contains "Verified"**
- **Timeout:** turns >= maxTurns → FAILED
- **Budget abort (stub): if set, force** FAILED **(placeholder for Phase 2+)**

———

6) In-Memory Bus

- publish(msg) **appends to an array**
- history() **returns a copy (used for result.log)**
- **No concurrency; single-thread event order guarantees test determinism**

———

7) CLI & Scripts

CLI apps/orchestrator/src/cli/pocMock.ts

- **Args: free-form task (**"add a README section" **default)**
- **Env:** MAX_TURNS **(default** 3**)**
- **Prints** diff **then** CONVERGED|FAILED**, exits** 0/1

Scripts

- apps/orchestrator/package.json
- "poc:mock": "tsx src/cli/pocMock.ts \"${TASK:-add a README section}\""
- "typecheck": "tsc -p tsconfig.json --noEmit"
- **Root** package.json
- "smoke:p1": "npm --workspace apps/orchestrator run poc:mock"

———

8) Tests (must pass locally & in CI)

1. **Converge: default task converges ≤3 turns; diff contains** README.md
2. **Forced fail:** maxTurns=1 **returns** FAILED
3. **Log order: bus preserves chronological order (architect→builder→verify…)**
4. **Schema (optional): validate a sample** Message **against** message.schema.json
5. **Negative (optional): malformed message rejected (if you add a validator)**

———

9) CI Changes

Append to the job after npm test**:**

```
- run: npm --workspace apps/orchestrator run typecheck
- run: npm run smoke:p1
```

(Keep existing checks: build-test, CodeQL, Scorecard.)

———

10) Smoke Test (promotion gate)

From repo root:

```
npm run smoke:p1              # prints diff + CONVERGED, exit 0
npm --workspace apps/orchestrator run typecheck
npm test
```

Definition of Done (Phase 1):

- poc:mock **converges within ≤3 turns for** "add a README section"
- **Unit tests pass; CI is green including** typecheck **and** smoke:p1
- **Message schema (TS + JSON Schema) present; bus + exits implemented**
- **A 1-page** docs/architecture/phase-1-plan.md **exists**

---

11) Budget & Telemetry (stubbed)

- **Keep** budgetTrace **fields in** Message**; populate with zeros (no real calls)**
- **Ensure the CLI prints only diff + status (no spend), preserving Phase 0E guardrails for later**

---

12) Risks & Mitigations

- **Loop never converges → add stricter verification rule &** maxTurns **fail path (already in)**
- **Schema drift → lock with JSON Schema; add a sample validation test**
- **Flaky tests → pure functions; no timers; single-thread bus**
- **Scope creep → no API calls or Git touches in Phase 1**

---

13) PR Checklist (use CodeRabbit + Copilot)

- **Schema files added & referenced**
- **CLI added;** poc:mock **script wired**
- **Tests cover converge + fail + ordering**
- **CI runs** typecheck **+** smoke:p1
- docs/architecture/phase-1-plan.md **created**
- **All checks green; branch policy satisfied**

---

14) "Do now" (three commands)

```
git checkout -b phase-1/bootstrap
npm run smoke:p1 || true   # ensure script wired; fix until it prints CONVERGED
npm test
```

If anything blocks, lean on Copilot for inline fixes, Codex for scaffolds, and open a PR for CodeRabbit to review.

---

15) Promotion to Phase 2 (readiness)

Promote when:

- **Phase 1 DoD met (above)**
- **Report in** docs/architecture/phase-1-plan.md **includes:**
- **Protocol summary**
- **Exit rules**
- **Test matrix & CI proof (short excerpts)**

Then we move to Phase 2 — Real Provider Adapters (Claude + OpenAI/GPT).