Phase 1 bootstrap

You've hit the Phase 0 + 0.5 exit criteria: docs/policies/metrics are in place, all smoke scripts pass, CI is green, and your mock loop converges. The "handoff" you posted is exactly the kind of audit trail we want—useful for reviewers and future you.

Below is a tiny, surgical "Phase 1 bootstrap" patch set so you meet Phase 1's deliverables and smoke test verbatim (two-agent loop, JSON message schema, in-memory bus, clear exit rules, and a poc:mock command that prints a diff + "CONVERGED" within ≤3 turns).

———

Phase 1 — minimal patch set (apply now)

1) Add a JSON Schema for messages (deliverable)

apps/orchestrator/src/schema/message.schema.json

```
{
  "$schema": "https://json-schema.org/draft/2020-12/schema",
  "$id": "https://example.com/orchestrator/message.schema.json",
  "title": "AgentMessage",
  "type": "object",
  "required": ["role", "type", "content", "turn"],
  "properties": {
    "role": { "enum": ["architect", "builder"] },
    "type": { "enum": ["propose", "critique", "implement", "verify"] },
    "content": { "type": "string", "minLength": 1 },
    "turn": { "type": "integer", "minimum": 1 },
    "reasons": { "type": "array", "items": { "type": "string" } },
    "budgetTrace": {
      "type": "object",
      "properties": {
        "inputTokens": { "type": "integer", "minimum": 0 },
        "outputTokens": { "type": "integer", "minimum": 0 }
      },
      "additionalProperties": false
    }
  },
  "additionalProperties": false
}
```

Keep your existing src/schema/messages.ts **for TS types; this JSON file satisfies the "JSON message schema" deliverable.**

———

2) Tighten exits and expose a CLI that matches the smoke test

apps/orchestrator/src/cli/pocMock.ts

```
import { runMockLoop } from '../kernel';
async function main() {
  const args = process.argv.slice(2);
  const task = args.join(' ').trim() || 'add a README section';
  const maxTurns = Number(process.env.MAX_TURNS || 3);
  const res = await runMockLoop(task, maxTurns);
  // Print the required artifacts
  if (res.diff) {
    console.log(res.diff);
  }
  console.log(res.status); // must include "CONVERGED" or "FAILED"
  // Exit code: 0 on converged, 1 otherwise
  process.exit(res.status === 'CONVERGED' ? 0 : 1);
}
main().catch((e) => {
  console.error(e);
  process.exit(1);
});
```

apps/orchestrator/src/kernel.ts (drop-in replacement for clearer exits; keep your agent code as-is)

```
import { InMemoryBus } from './bus/inMemoryBus';
import { architectPropose, architectCritique, architectVerify } from './agents/architect.mock';
import { builderImplement } from './agents/builder.mock';
import type { DebateResult, Message } from './schema/messages';
export async function runMockLoop(task: string, maxTurns = 3): Promise<DebateResult> {
  const bus = new InMemoryBus();
  let diff = '';
  let lastCritique: Message | null = null;
  for (let turn = 1; turn <= maxTurns; turn++) {
    if (turn === 1) {
      bus.publish(architectPropose(task, turn));
    } else {
      const c = architectCritique(diff, turn);
      bus.publish(c);
      lastCritique = c;
    }
    const impl = builderImplement(task, lastCritique?.content ?? null, turn);
    diff = impl.diff;
    bus.publish(impl.msg);
    const v = architectVerify(diff, turn);
    bus.publish(v);
    if (v.content.startsWith('Verified')) {
      return { status: 'CONVERGED', turns: turn, diff, log: bus.history() };
    }
  }
  return { status: 'FAILED', turns: maxTurns, diff, log: bus.history() };
}
```

---

3) Wire scripts so the Phase 1 smoke is literal

apps/orchestrator/package.json (add scripts)

```
{
  "scripts": {
    "typecheck": "tsc -p tsconfig.json --noEmit",
    "poc:mock": "tsx src/cli/pocMock.ts \"${TASK:-add a README section}\""
  }
}
```

root package.json (add a new smoke script for Phase 1)

```
{
  "scripts": {
    "smoke:p1": "npm --workspace apps/orchestrator run poc:mock"
  }
}
```

---

4) Add tests for failure & turn-limit (optional but quick)

apps/orchestrator/tests/poc.spec.ts

```
import { describe, it, expect } from 'vitest';
import { runMockLoop } from '../src/kernel';
describe('Phase-1 exits', () => {
  it('converges within 3 turns for the README task', async () => {
    const res = await runMockLoop('add a README section', 3);
    expect(res.status).toBe('CONVERGED');
    expect(res.turns).toBeLessThanOrEqual(3);
  });
  it('fails when maxTurns=1 (forced)', async () => {
    const res = await runMockLoop('edge case: too few turns', 1);
    expect(res.status).toBe('FAILED');
  });
});
```

---

5) CI tighten (optional now, good hygiene)
In .github/workflows/ci.yml, after tests:

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

———

Phase 1 smoke (run this now)

```
# From repo root
npm run smoke:p1              # prints diff + CONVERGED and exits 0
npm --workspace apps/orchestrator run typecheck
npm test
```

If anything's sticky, lean on:

- **Copilot for quick inline TS fixes,**
- **Codex for scaffolding tests/CLI,**
- **CodeRabbitAI to review the PR once you push the Phase 1 bootstrap branch.**

———

Tiny gap notes (non-blocking for Phase 1)

- **Required checks currently: build-test, CodeQL, Scorecard. That's fine for now; we'll bring CodeRabbit + Copilot Code Review online as *enforced* in Phase 5 (Review Gates).**
- **Exact check "context" strings still need discovery on first run of those bots—your report already calls this out.**

———

If you want, I can also generate a 1-page docs/architecture/phase-1-plan.md with the schema and loop diagram—but you're clear to start Phase 1 with the steps above.