Think-Center: Coordinating Multiple Claude Code Instances

The Problem

When building full-stack applications, you often have multiple Claude Code instances working on different parts (backend, frontend) that need to share a common API contract. Without coordination, these instances drift apart, creating integration nightmares.

The Solution

Use think-center as your architectural orchestrator - not writing code but ensuring both code-writers dance together.

Workflow

1. Initial Design Session

```
Think-center:
"Maker, we're building [system description]
- Backend handles: [responsibilities]
- Frontend needs: [requirements]
- Shared concerns: [auth, errors, data formats]"

"Weaver, what's the narrative flow between frontend/backend?"
"Maker, define the API endpoints needed"
"Checker, what edge cases will break this contract?"
```

2. Create API Contract

```
"Scribe, document our API contract:
- Endpoints: [list]
- Request/Response formats: [schemas]
- Error patterns: [standardized errors]
- Auth flow: [token handling]"
```

3. Split to Claude Code Instances

Backend Instance:

```
"Implement these endpoints per our contract:
[paste contract]
Focus on [specific backend concerns]"
```

Frontend Instance:

```
"Consume these endpoints per our contract:
[paste contract]
Focus on [specific frontend concerns]"
```

4. Coordination Checkpoints

Morning Sync

```
Think-center:
"Council, here's what both sides built yesterday:
- Backend implemented: [features]
- Frontend integrated: [features]
- Conflicts found: [list]"
```

Before ANY API Change

```
Think-center:
"Maker, backend needs to change [endpoint] because [reason]
How do we version this?"

"Checker, what's the migration path?"

"E/E, is this change worth the coordination cost?"
```

Integration Mismatches

```
Think-center:
"Checker, we have a mismatch:
- Backend returns: null for missing data
- Frontend expects: empty array
How do we reconcile?"
```

5. Continuous Patterns

When Either Side Gets Stuck

```
Claude Code Backend: *hits complex problem*
→ Think-center: "Backend stuck on [problem], impacts API how?"
→ Claude Code Frontend: *adjusts expectations*
```

State Synchronization

```
Think-center:
"O/G, both sides handle user state differently
What's the source of truth?"
```

Key Principles

1. **Think-center holds the contract** - Single source of truth
2. **No unilateral API changes** - All modifications go through Council review
3. **Scribe maintains living documentation** - Contract evolves but stays synchronized
4. **Perspectives prevent problems**:
    - Weaver: Ensures narrative coherence
    - Maker: Keeps things buildable
    - Checker: Catches integration issues early
    - O/G: Spots human/team dynamics issues

Example Session

```
Morning:
You: "Council meeting on payment flow implementation"
Weaver: "The story: user selects plan → processes payment → activates features"
Maker: "Three endpoints needed: /plans, /process-payment, /activate"
Checker: "What if payment succeeds but activation fails?"
Scribe: *documents edge cases*

Split to Claude Code:
Backend: Implements with rollback capability
Frontend: Implements with retry logic

Evening reconciliation:
You: "Checker, both implemented differently - compatible?"
Checker: "Backend rollback + frontend retry could cause double-charging"
Maker: "Add idempotency key to prevent this"
```

Benefits

- **Prevents drift**: Regular sync keeps both sides aligned
- **Catches issues early**: Checker spots mismatches before they're coded
- **Documents decisions**: Scribe maintains context for "why"
- **Reduces rework**: Think before code, not after

Pro Tips

1. Start each day with think-center sync before opening Claude Code
2. End each day with integration check in think-center
3. Any "quick API change" goes through Council first
4. When in doubt, ask Checker to verify assumptions
5. Let Scribe document all contract changes with reasons

Think better together, build faster apart