🚀 BMAD V4 Execution Plan v3.0

Dependency-Based Agent Implementation Strategy

Created: 2025-10-21
Version: 3.0 - Dependency-Based Execution
Replaces: v2.0 (Timeline-Based Approach)

⚡ CRITICAL PARADIGM

Autonomous Agents = Dependency-Driven, NOT Time-Driven

WRONG APPROACH (v2.0):

```
Week 1, Day 1-2: Do X
Week 1, Day 3-4: Do Y
Week 2: Do Z
```

Problems:

- ❌ Artificial constraints
- ❌ Ignores agent autonomy
- ❌ Assumes linear progress
- ❌ Doesn't leverage parallelism

CORRECT APPROACH (v3.0):

```
Phase 1: Foundation Prerequisites
    ├─ Complete when: Infrastructure ready
    └─ Unblocks: Phase 2

Phase 2: Core Development
    ├─ Complete when: All dependencies satisfied
    └─ Unblocks: Phase 3

Phase 3: Integration & Testing
    ├─ Complete when: All systems integrated
    └─ Unblocks: Production
```

Benefits:

- ✅ **Natural Flow:** Agents work when dependencies complete
- ✅ **True Parallelism:** Multiple agents work simultaneously
- ✅ **Adaptive:** Handles blockers organically
- ✅ **Scalable:** Add agents without timeline conflicts

📋 DEPENDENCY-BASED EXECUTION PHASES

PHASE 1: Foundation Bootstrap

Prerequisites: NONE (entry point)

Completion Criteria:

- □ Claude Code SDK installed and functional
- □ Dashboard hook endpoints deployed and tested
- □ Neo4j context accessible to agents
- □ Agent boot sequence implemented
- □ First 3 agents spawned successfully
- □ Webhook authentication working
- □ Real-time monitoring operational

Agents Involved:

- Bootstrap/Orchestrator (human-initiated)
- Test Agent (validation)

Unblocks:

- Phase 2: Pilot Agent Sprint
- All future agent spawning

Success Signal:

- 3 agents complete assigned tasks autonomously
- Dashboard shows real-time status
- PRs created automatically

PHASE 2: Pilot Agent Sprint

Prerequisites:

- ☑ Phase 1 complete
- ☑ 3 agents (Alex, Sarah, David) spawned and functional

Completion Criteria:

- □ 10 foundational tasks completed
- □ Docker environment operational (Alex)
- □ Neo4j schemas deployed (Sarah)
- □ Express API foundation ready (David)
- □ Zero manual interventions required
- □ Agents collaborate autonomously via Neo4j
- □ All PRs merged successfully

Agents Active:

- Alex Martinez (DEVOPS-ALPHA)
- Sarah Chen (DATABASE-SIERRA)
- David Rodriguez (BACKEND-DELTA)

Task Dependencies:

```
TASK-AM-001 (Docker setup)
    └─ Unblocks: TASK-SC-001 (Neo4j in Docker)
    └─ Unblocks: TASK-DR-001 (Express server)

TASK-SC-002 (Schema design)
    └─ Unblocks: TASK-DR-003 (Neo4j queries)

TASK-DR-002 (API routes)
    └─ Unblocks: Wave 1 backend tasks
```

Unblocks:

- Phase 3: Scale to 10 Agents
- Frontend development
- Telnyx integration

Success Signal:

- All 10 tasks in 'COMPLETED' status
- No blocking errors
- Cost tracking shows <$50 spent

PHASE 3: Scale to 10 Agents

Prerequisites:

- ☑ Phase 2 complete
- ☑ Foundation infrastructure working
- ☑ Agent collaboration proven

Completion Criteria:

- □ 7 additional agents spawned (total 10)
- □ 40 Wave 1 tasks assigned and in progress
- □ Backend, Frontend, Database work running in parallel
- □ No critical conflicts in PRs
- □ Dashboard monitoring 10 concurrent sessions

Agents Added:

- Jennifer Kim (TELNYX-JULIET)
- Robert Wilson (CONVERSATION-ROMEO)
- Lisa Chang (VECTOR-LIMA)

- Marcus Thompson (SECURITY-MIKE)
- Michael Park (FRONTEND-MIKE)
- Emma Johnson (MONITOR-ECHO)
- James Taylor (LEADMGMT-JULIET)

Focus Areas:

```
Backend Track (David, Jennifer, Robert):
    - API endpoints
    - Telnyx integration
    - Conversation management

Database Track (Sarah, Lisa):
    - Schema optimization
    - Vector embeddings
    - Data migrations

Infrastructure Track (Alex, Marcus):
    - Security hardening
    - CI/CD pipelines
    - Monitoring setup

Frontend Track (Michael, Emma, James):
    - Dashboard foundation
    - Lead management UI
    - Real-time updates
```

Unblocks:

- Phase 4: Full 17-Agent Deployment
- Integration testing

Success Signal:

- 40 tasks completed
- All tracks progressing in parallel
- <5% error rate across agents

PHASE 4: Full 17-Agent Deployment

Prerequisites:

- ☑ Phase 3 complete
- ☑ 10 agents working smoothly
- ☑ No major blockers in dependency graph

Completion Criteria:

- □ Final 7 agents spawned (total 17)
- □ 188 Wave 1 tasks completed

- □ All systems integrated
- □ Dashboard fully operational
- □ Cost remains under $250/month
- □ No critical bugs

Agents Added:

- Priya Patel (VOICE-PAPA)
- Angela White (ANALYTICS-ALPHA)
- Rachel Green (INTEGRATION-ROMEO)
- Kevin Brown (QA-KILO)
- Nicole Davis (TESTING-NOVEMBER)
- Thomas Garcia (PERFORMANCE-TANGO)
- Daniel Lee (USERMGMT-DELTA)

Wave 1 Focus:

```
188 parallel tasks across:
    - Backend APIs (40 tasks)
    - Frontend Components (35 tasks)
    - Database Schemas (25 tasks)
    - Telnyx Integration (20 tasks)
    - Voice Agent (18 tasks)
    - Security (15 tasks)
    - Testing (20 tasks)
    - Analytics (15 tasks)
```

Unblocks:

- Wave 2 tasks (dependent on Wave 1 completion)
- Production deployment
- Live testing

Success Signal:

- 91% of total tasks complete (188/206)
- System functional end-to-end
- Ready for Wave 2

PHASE 5: Wave 2 Completion

Prerequisites:

- ☑ Phase 4 complete
- ☑ All Wave 1 dependencies satisfied
- ☑ Integration testing passed

Completion Criteria:

- □ 18 Wave 2 tasks completed
- □ Advanced features implemented
- □ Performance optimization complete
- □ Security audit passed
- □ Production ready

Wave 2 Tasks:

```
Dependent on Wave 1 completion:
    - Advanced analytics dashboards
    - Complex integrations
    - Performance tuning
    - Final security hardening
    - User acceptance testing
```

Unblocks:

- Production deployment
- Customer onboarding

Success Signal:

- All 206 tasks complete
- System production-ready
- Cost efficiency validated

💰 COST MANAGEMENT (Updated for Autonomous Model)

Cost Per Phase (Estimated):

```
Phase 1 (Bootstrap): $20-30
    - 3 agents, limited tasks
    - Setup and configuration

Phase 2 (Pilot): $30-50
    - 3 agents, 10 tasks
    - Proof of autonomous execution

Phase 3 (Scale): $50-100
    - 10 agents, 40 tasks
    - Parallel execution

Phase 4 (Full Deploy): $150-200
    - 17 agents, 188 tasks
    - Full-scale operation

Phase 5 (Wave 2): $30-50
    - 17 agents, 18 tasks
    - Optimization work

Total Project Cost: $280-430
```

ROI Comparison:

```
MANUAL APPROACH:
- 206 tasks × 2 hours/task = 412 hours
- 412 hours × $75/hour = $30,900

AUTONOMOUS APPROACH:
- 206 tasks × 0.5 hours/task = 103 hours (autonomous)
- Subscription cost: $280-430
- Savings: $30,470-$30,620 (98.6% reduction)
```

📈 SUCCESS METRICS

Phase Completion Indicators:

Phase 1:

- □ 3 agents operational
- □ 100% hook delivery success
- □ Real-time monitoring functional

Phase 2:

- □ 10 tasks completed autonomously
- □ Zero manual interventions
- □ Agent collaboration validated

Phase 3:

- □ 40 tasks completed in parallel
- □ <5% error rate
- □ All tracks progressing

Phase 4:

- □ 188 Wave 1 tasks complete
- □ 95% PR auto-merge rate
- □ Cost under $250/month

Phase 5:

- □ All 206 tasks complete
- □ System production-ready
- □ Performance targets met

🎯 DEPENDENCY GRAPH VISUALIZATION

```
Phase 1 (Bootstrap)
    │
    ├──> Phase 2 (Pilot) ──> Phase 3 (Scale) ──> Phase 4 (Full) ──> Phase 5 (Wave 2)
    │         │                    │                   │                   │
    │         │                    │                   │                   └─> PRODUCTION
    │         │                    │                   │
    │         └─> Docker Env       └─> Backend APIs    └─> Integration
    │              Neo4j DB             Frontend            Testing
    │              Express API          Telnyx              Optimization
    │                                   Security
    │
    └──> SDK Setup
         Hook Config
         Agent Boot
```

🚀 IMPLEMENTATION APPROACH

Starting Phase 1:

Human Role:

1. Install Claude Code SDK: `npm install @anthropic-ai/claude-code-sdk`
2. Deploy hook endpoints (already in `routes/hooks.js`)
3. Start dashboard: `npm run dev`
4. Spawn first test agent
5. Verify webhook delivery
6. **Then step back and let agents work**

Agent Role:

1. Agent boots with identity from Neo4j
2. Queries task graph for assignments
3. Executes tasks autonomously
4. Sends progress via webhooks
5. Notifies dependent agents when complete
6. Creates PR automatically
7. Waits for next dependency to unlock

Key Principle:

> Humans set up infrastructure and spawn agents.
> Agents coordinate and execute autonomously.
> Humans monitor progress and handle exceptions only.

🔄 DEPENDENCY RESOLUTION

How Agents Know What To Do:

1. **Agent Boots:**

```
// Agent queries Neo4j
const myTasks = await neo4j.query(`
  MATCH (a:Agent {agentId: $id})-[:ASSIGNED_TO]->(t:Task)
  WHERE t.status = 'READY'
  RETURN t
`, {id: agentId});
```

2. **Task Selection:**

```
// Check dependencies
const readyTasks = myTasks.filter(task =>
  task.dependencies.every(dep => dep.status === 'COMPLETED')
);
```

3. **Execution:**

```
// Work on first ready task
const task = readyTasks[0];
await executeTask(task);
```

4. **Completion:**

```
// Update graph
await neo4j.query(`
  MATCH (t:Task {taskId: $id})
  SET t.status = 'COMPLETED', t.completedAt = datetime()
`, {id: task.taskId});

// Notify dependent agents via webhook
await notifyDependentAgents(task);
```

📊 MONITORING & ADAPTATION

Real-Time Signals:

Green Signals (Proceed):

- ✅ Tasks completing within expected time
- ✅ PRs merging successfully
- ✅ No blocking errors
- ✅ Agents collaborating smoothly
- ✅ Cost tracking on target

Yellow Signals (Monitor):

- ⚠️ Some tasks taking longer than average
- ⚠️ Occasional PR conflicts
- ⚠️ Minor errors (but agents recovering)
- ⚠️ Cost slightly elevated

Red Signals (Intervene):

- 🚨 Agents blocked for >2 hours
- 🚨 Critical failures in multiple agents
- 🚨 Security vulnerabilities detected
- 🚨 Cost exceeding $100/phase
- 🚨 Circular dependencies discovered

Adaptation Strategies:

If Blocked:

- Review dependency graph for issues
- Check if prerequisite task failed
- Manually unblock if necessary
- Adjust task assignments

If Cost Overruns:

- Pause non-critical agents
- Batch small tasks
- Optimize agent prompts
- Review API usage patterns

If Quality Issues:

- Enable stricter PR reviews
- Add testing requirements
- Adjust agent instructions
- Implement quality gates

🎓 PHILOSOPHY: AGENT AUTONOMY

Core Principles:

1. **Trust the Graph:**

- Neo4j knows the dependencies
- Agents follow the graph
- Graph is single source of truth

2. **Parallel by Default:**
   - If no dependency, work in parallel
   - Don't serialize unnecessarily
   - Maximize throughput
3. **Fail Fast, Recover Fast:**
   - Agents detect failures early
   - Report via webhooks
   - Retry or escalate automatically
4. **Organic Scaling:**
   - Add agents when ready
   - Don't force artificial milestones
   - Let workload drive scaling
5. **Human as Exception Handler:**
   - Agents handle routine work
   - Humans resolve exceptions
   - Humans optimize process

Execution Plan Version: 3.0 - Dependency-Based Execution
Last Updated: 2025-10-21
Status: Ready for Implementation
Next Step: Begin Phase 1 Bootstrap

🚀 Autonomous Agents Work on Dependencies, Not Deadlines! 🚀