5. Use Case: Creating a New Product

Step-by-step walkthrough of how the system builds a new product from scratch

5.1 Scenario: Building "Bento" — a Meal Prep SaaS

Use Case: Building "Bento" — a Meal Prep SaaS

Let's walk through how the ALAI system would build a completely new product from CEO idea to deployed MVP.

The Idea

> Alem: "We need a meal prep subscription SaaS for the Nordic market. Next.js frontend, Kotlin/Ktor backend, PostgreSQL. Call it Bento."

Step 1: Task Creation in Mission Control

Alem (or John) creates the initial tasks:

```
node mc.js add "Bento: Project scaffold — Next.js frontend + Kotlin/Ktor API" -p H -t alem
node mc.js add "Bento: Database schema design — users, subscriptions, meals, deliveries" -p H
node mc.js add "Bento: Landing page design — Nordic minimalist, hero + pricing + CTA" -p H
node mc.js add "Bento: API — user auth + subscription management endpoints" -p H
node mc.js add "Bento: Frontend — meal selection + checkout flow" -p H
```

These 5 tasks enter MC as `[open]` status.

Step 2: Pi-Orchestrator Picks Up Tasks

Within 30 seconds, the orchestrator wakes up:

```
[INFO] Found task #4001: Bento: Project scaffold — Next.js frontend + Kotlin/Ktor API
[INFO] Task #4001 classified: complexity=3 domain=code type=scaffold
[INFO] Task #4001 → llama-tool (kimi-k2.5:tq1 on forge)
```

Classification logic:

- "Project scaffold" + "Next.js" + "Kotlin/Ktor" → domain: code, type: scaffold, complexity: 3
- Complexity 3 → Tier 3 model (Kimi K2.5 on FORGE)

Routing:

- "Next.js frontend" → keyword match → **CodeCraft** (fullstack covers both)
- CodeCraft's soul + blueprints are injected into the prompt

Step 3: CodeCraft Builds It

The agent receives a prompt containing:

1. CodeCraft's company identity and coding standards
2. The `nextjs-app.yaml` blueprint template
3. The task description
4. Relevant HiveMind context (if any previous Bento-related work exists)

Agent output:

- Project structure created
- `package.json`, `docker-compose.yml`, `README.md`
- Basic Next.js pages + Kotlin API skeleton
- GOTCHA file written as proof of work

```
[INFO] Auto QA prep completed for task #4001
[INFO] Task #4001 completed via llama-server/kimi-k2.5:tq1 (912s)
```

5.2 Pipeline Cascade: Review ? Security ? Deploy

Pipeline Cascade

Step 4: Pipeline Triggers REVIEW

After BUILD completes, the Pipeline Engine fires:

```
pipeline-engine.js advance 4001
  → Stage: BUILD complete
  → Creating REVIEW task for Proveo
```

New MC task created:

```
#4006 [M] [open] [—] [BENTO-REVIEW] Code review of project scaffold (parent: #4001)
```

Pi-Orchestrator picks it up → routes to **Proveo** (audit firm):

Proveo's agent reviews:

- ✅ Project structure follows conventions
- ⚠ Missing .env.example
- ⚠ No health check endpoint
- ✅ Docker setup correct

Result: REVIEW passes with minor findings.

Step 5: Pipeline Triggers SECURITY

```
pipeline-engine.js advance 4006
  → Stage: REVIEW complete (pass)
  → Creating SECURITY task for Securion
```

New MC task:

```
#4007 [M] [open] [—] [BENTO-SECURITY] Security audit of project scaffold (parent: #4006)
```

Securion's agent checks:

- □ No hardcoded secrets
- □ CORS configured correctly
- ⚠ CSP headers missing
- ⚠ Rate limiting not implemented

Result: SECURITY passes with findings → creates follow-up BUILD task for fixes.

Step 6: Parallel Processing

While the scaffold goes through the pipeline, the other tasks are also being processed:

```
[22:05] Task #4002 (DB schema)    → CodeCraft → Kimi K2.5
[22:08] Task #4003 (Landing page) → Vizu      → qwen2.5-coder:32b
[22:10] Task #4004 (Auth API)     → CodeCraft → Kimi K2.5
[22:13] Task #4005 (Meal UI)      → Vizu      → qwen2.5-coder:32b
```

Note: Vizu handles the frontend (landing page, meal UI), CodeCraft handles the backend (DB, API). Each gets their company-specific soul and blueprints.

Step 7: OPS Stage (Deploy)

If a task is tagged for deployment:

```
pipeline-engine.js advance 4007
  → Stage: SECURITY complete
  → Task has deploy trigger
  → Creating OPS task for FlowForge
```

FlowForge creates:

- Docker build pipeline
- GitHub Actions CI/CD
- Staging environment config
- Health check monitoring

Step 8: DOCS Stage

Finally, Lexicon creates documentation:

- Privacy Policy (GDPR-compliant for Nordic market)
- Terms of Service
- API documentation
- User guides

5.3 End-to-End Timeline

End-to-End Timeline

What the System Produces

Starting from 5 MC tasks, the full pipeline generates:

Stage	Tasks	Company	Time
BUILD — scaffold	1	CodeCraft	~15 min
BUILD — DB schema	1	CodeCraft	~10 min
BUILD — landing page	1	Vizu	~8 min
BUILD — auth API	1	CodeCraft	~15 min
BUILD — meal UI	1	Vizu	~10 min
REVIEW (all 5)	5	Proveo	~5 min each
SECURITY (all 5)	5	Securion	~5 min each
BUILD — security fixes	2-3	CodeCraft	~10 min each
OPS — deploy setup	1	FlowForge	~10 min
DOCS — legal + API docs	2	Lexicon	~8 min each

Total: ~20-25 tasks auto-generated from 5 initial tasks.

Elapsed time: ~3-4 hours (tasks run sequentially on each host, some parallel on ANVIL+FORGE).

Human involvement: Zero (unless safety patterns trigger).

What Alem Sees

```
$ node mc.js list --tag bento

#4001  [H] [done] Bento: Project scaffold
#4002  [H] [done] Bento: Database schema design
```

```
#4003  [H] [done] Bento: Landing page design
#4004  [H] [done] Bento: API — auth + subscriptions
#4005  [H] [done] Bento: Frontend — meal selection + checkout
#4006  [M] [done] [BENTO-REVIEW] Code review: scaffold
#4007  [M] [done] [BENTO-SECURITY] Security audit: scaffold
#4008  [M] [done] [BENTO-REVIEW] Code review: DB schema
...
#4020  [M] [done] [BENTO-OPS] Deploy setup
#4021  [M] [done] [BENTO-DOCS] Privacy Policy + ToS
#4022  [M] [done] [BENTO-DOCS] API documentation
```

Key Insight

The 5 tasks Alem created cascaded into 20+ tasks that were:

- **Automatically created** by the Pipeline Engine
- **Automatically classified** by the Pi-Orchestrator
- **Automatically routed** to the right virtual company
- **Automatically executed** by AI agents
- **Automatically quality-checked** by the QA gate
- **Automatically stored** in HiveMind for future reference

This is the ALAI AI Factory in action.