CLAUDE COWORK

Product Management Handbook

The Complete Guide to the Claude Cowork Product Management Plugin

VERSION 1.0.0

PLUGIN product-management

RationalEyes.ai

contact@rationaleyes.ai

February 2026

Table of Contents

The Complete Guide to the Claude Cowork Product Management Plugin

Version 1.0.0 | Last Updated: February 2026

Part I: Introduction

1. What This Plugin Does

The product-management plugin transforms Claude into an AI-powered product management partner. It provides specialized knowledge, frameworks, and workflows that cover the complete PM lifecycle from discovery through delivery.

Core Capabilities:

The plugin excels at six major PM workflows:

1. **Writing Product Specs (PRDs)** — Generate comprehensive product requirements documents from a problem statement, complete with user stories, acceptance criteria, success metrics, and scope definition.
2. **Managing Roadmaps** — Create, update, and reprioritize product roadmaps using industry-standard frameworks (RICE, MoSCoW, Now/Next/Later). Identify dependencies, manage capacity, and communicate changes to stakeholders.
3. **Stakeholder Communication** — Draft tailored updates for different audiences (executives, engineering, customers, cross-functional partners) with appropriate detail level, framing, and format.
4. **Research Synthesis** — Process qualitative and quantitative user research from interviews, surveys, support tickets, and analytics into structured insights with themes, personas, and opportunity areas.
5. **Competitive Analysis** — Research competitors and generate briefs with feature comparisons, positioning analysis, win/loss insights, and strategic implications.
6. **Metrics Analysis** — Review product metrics, identify trends, diagnose issues, set goals, and generate actionable recommendations based on data.

What Makes This Different:

Unlike generic AI assistance, this plugin brings PM-specific expertise:

- Knows PRD structure, user story formats, and acceptance criteria patterns
- Understands prioritization frameworks (RICE, MoSCoW, ICE, value-vs-effort)
- Applies stakeholder communication best practices (G/Y/R status, ROAM risk framework)
- Uses research synthesis methodologies (thematic analysis, affinity mapping, triangulation)

- Implements competitive analysis frameworks (positioning maps, feature matrices, win/loss)
- Follows metrics hierarchies (North Star, L1 health indicators, L2 diagnostic metrics)

Integration with Your Stack:

The plugin connects to 12 different MCP servers covering project management (Linear, Asana, monday, ClickUp, Atlassian), communication (Slack), knowledge bases (Notion), design (Figma), analytics (Amplitude, Pendo), customer feedback (Intercom), and meeting transcription (Fireflies). These connections allow Claude to pull real data from your tools rather than working in isolation.

2. Who This Plugin Is For

Primary Users:

- **Product Managers** (IC level) — The plugin handles the "writing" burden of PM work: specs, roadmaps, updates, research synthesis. It applies frameworks correctly and maintains consistent structure.
- **Senior PMs & Product Leads** — Use for faster stakeholder communication, competitive research, and mentoring junior PMs through the plugin's embedded frameworks and best practices.
- **Product Directors & VPs** — Generate executive summaries, competitive landscape maps, portfolio roadmaps, and strategic analysis. The plugin maintains appropriate altitude for leadership communication.
- **Product Ops** — Standardize processes, templates, and frameworks across the PM organization. The plugin codifies best practices and ensures consistency.

Secondary Users:

- **Founders & Startup Teams** — When the founder is wearing the PM hat, this plugin provides structure and frameworks that might otherwise come from a seasoned PM hire.
- **Engineering Managers** — When engineers need to write specs, plan technical roadmaps, or communicate with stakeholders, the plugin provides PM discipline without requiring PM expertise.
- **GTM Teams** — Sales, marketing, and CS teams can use the plugin to understand product strategy, generate customer-facing roadmaps, and prepare competitive materials.

Skills & Prerequisites:

You do not need to be a PM expert to use this plugin effectively. It teaches PM frameworks as it applies them. However, you do need:

- Basic understanding of your product and users
- Access to relevant data (or willingness to provide it when Claude asks)
- Ability to make product decisions (the plugin provides frameworks and recommendations, but you decide)

3. Installation and Setup

Cowork (recommended): Open **Plugin Settings** in the Cowork desktop app, find **Product Management**, and click **Install**. The plugin activates immediately — no CLI required.

Claude Code CLI (alternative): If you are using Claude Code in the terminal rather than Cowork, install via:

```
claude plugins add knowledge-work-plugins/product-management
```

> **Note:** *All standard Cowork plugins, including Product Management, are available from Plugin Settings with a single click. The CLI command above is only needed for Claude Code terminal users.*

The plugin is immediately functional after installation. All commands and skills work without any configuration.

MCP Server Setup:

To get the most value, connect the tools your team uses. The plugin includes pre-configured MCP servers for 12 services. Authentication is handled through each service's standard OAuth flow.

Included MCP Servers:

CATEGORY	TOOL	CONNECTION TYPE	WHAT IT PROVIDES
Project tracker	Linear	HTTP	Roadmap items, tickets, statuses, dependencies
Project tracker	Asana	HTTP	Tasks, projects, milestones, assignees
Project tracker	monday.com	HTTP	Boards, items, workflows, updates
Project tracker	ClickUp	HTTP	Tasks, lists, goals, time tracking
Project tracker	Atlassian	HTTP	Jira tickets, Confluence pages, status

CATEGORY	TOOL	CONNECTION TYPE	WHAT IT PROVIDES
Chat	Slack	HTTP	Channel messages, threads, decisions
Knowledge base	Notion	HTTP	Meeting notes, docs, research, specs
Design	Figma	HTTP	Design files, prototypes, components
Product analytics	Amplitude	HTTP	User behavior, funnels, retention, cohorts
Product analytics	Pendo	HTTP	Feature usage, guides, feedback
User feedback	Intercom	HTTP	Support tickets, feature requests, conversations
Meeting transcription	Fireflies	HTTP	Meeting summaries, transcripts, action items

First-Time Setup Workflow:

1. Install the plugin
2. Try a command without any connections (e.g., `/write-spec`) to see how it works
3. Connect one tool (start with your project tracker)
4. Re-run the command and observe how Claude now pulls real data
5. Add additional tool connections as needed

Tool-Agnostic Design:

The plugin uses placeholder syntax (`~~project tracker`, `~~chat`, `~~design`) in its instructions, making it work with any tool in a category. The included MCP servers are pre-configured, but you can swap them for alternatives (see Part IV for details).

Part II: Commands Reference

4. Write Feature Specs (/write-spec)

Purpose: Generate a structured product requirements document (PRD) from a problem statement or feature idea.

When to Use:

- Starting to spec a new feature
- Documenting a feature decision before engineering begins
- Creating alignment on scope and success criteria
- Converting a vague idea into a concrete plan

Basic Usage:

```
/write-spec SSO support for enterprise customers
```

Claude will gather context through questions, then generate a full PRD.

What It Produces:

A structured PRD with eight sections:

1. **Problem Statement** — User problem, affected users, cost of not solving (2-3 sentences grounded in evidence)
2. **Goals** — 3-5 measurable outcomes distinguishing user goals from business goals
3. **Non-Goals** — 3-5 explicitly out-of-scope items with rationale
4. **User Stories** — Standard format ("As a [user], I want [capability] so that [benefit]")
5. **Requirements** — Categorized as Must-Have (P0), Nice-to-Have (P1), Future (P2) with acceptance criteria
6. **Success Metrics** — Leading indicators (change quickly) and lagging indicators (change slowly)
7. **Open Questions** — Unresolved issues tagged with who needs to answer
8. **Timeline Considerations** — Hard deadlines, dependencies, phasing suggestions

Input Questions:

Claude asks for context you might not provide upfront:

- User problem and affected segment

- Success metrics (how will we know this worked?)
- Technical constraints or dependencies
- Timeline requirements
- Prior attempts or existing solutions

Tool Integration:

If project tracker is connected, Claude pulls related tickets and epics. If knowledge base is connected, Claude searches for prior specs and research. If design tools are connected, Claude references existing mockups. Without these connections, Claude works from what you provide.

Output Format:

Markdown with clear headers, scannable structure. Busy stakeholders should be able to read headers and bold text to understand the spec.

Advanced Usage:

Specify scope directly:

```
/write-spec Add CSV export to reports, focusing on enterprise admin use case only
```

Reference existing documents:

```
/write-spec Based on the user research in notion://research/onboarding-interviews
```

Customization:

The underlying **feature-spec** skill provides the PRD structure. You can customize templates, add company-specific sections, or modify prioritization frameworks (see Part VII).

5. Manage Roadmaps (/roadmap-update)

Purpose: Create, update, or reprioritize a product roadmap with dependency mapping and capacity planning.

When to Use:

- Creating a new roadmap for a quarter or release
- Updating status of roadmap items
- Reprioritizing based on new information
- Adding items to the roadmap

- Communicating roadmap changes to stakeholders
- Moving timelines due to dependencies or scope changes

Basic Usage:

```
/roadmap-update
```

Claude asks what operation you want (add item, update status, reprioritize, move timeline, create new).

Five Core Operations:

1. Add Item

```
Claude: What do you want to add?
You: SSO support for enterprise customers
Claude: [Gathers priority, effort, timeline, owner, dependencies]
Claude: [Suggests where it fits based on current priorities and capacity]
```

2. Update Status

```
You: Move "Mobile app v2" to "at risk"
Claude: What is the risk?
You: iOS review taking longer than expected
Claude: [Updates status, asks for mitigation plan]
```

3. Reprioritize

```
You: We need to prioritize the API v3 work higher
Claude: What changed that drives this reprioritization?
You: Major customer renewal depends on it
Claude: [Applies prioritization framework, shows before/after, identifies what moves d
```

4. Move Timeline

```
You: Push the reporting feature back 2 weeks
Claude: Why is it moving?
You: Dependency on data pipeline work slipped
Claude: [Identifies downstream impacts on dependent items]
```

5. Create New Roadmap

```
You: Create Q2 roadmap
Claude: What format? (Now/Next/Later, quarterly themes, OKR-aligned, timeline view)
You: Now/Next/Later
Claude: [Gathers initiatives, applies framework, produces roadmap]
```

What It Produces:

A roadmap view with:

- **Status Overview** — X items in progress, Y completed, Z at risk
- **Roadmap Items** — For each: name, description, status, timeframe, owner, dependencies
- **Risks and Dependencies** — Blocked items, cross-team dependencies, approaching deadlines
- **Changes This Update** — What changed since last version (for updates)

Prioritization Frameworks:

The **roadmap-management** skill provides five frameworks:

1. **RICE** — (Reach × Impact × Confidence) / Effort
2. **MoSCoW** — Must/Should/Could/Won't have
3. **ICE** — Impact × Confidence × Ease
4. **Value vs Effort** — 2×2 matrix (Quick wins, Big bets, Fill-ins, Money pits)
5. **OKR-Aligned** — Map items to Objectives and Key Results

Claude selects the appropriate framework based on context or asks which to use.

Tool Integration:

If project tracker is connected, Claude pulls current roadmap state, statuses, owners, and dates. Updates can be written back to the tracker. Without connection, Claude works from what you provide (paste your roadmap, describe it, or upload a file).

Output Formats:

- **Table** — Best for status-at-a-glance
- **Now/Next/Later** — Best for executive communication
- **Quarterly Themes** — Best for strategic alignment
- **Timeline/Gantt** — Best for dependency visualization

Best Practices Embedded:

- When you add an item, Claude asks what comes off or moves (roadmaps are zero-sum against capacity)
- When reprioritizing, Claude asks what changed (prevents arbitrary priority shifts)

- When items move, Claude flags dependent items that need adjustment
- Claude surfaces capacity issues ("This roadmap has 8 weeks of work for a 4-week sprint")

6. Stakeholder Updates (/stakeholder-update)

Purpose: Generate stakeholder updates tailored to audience (executives, engineering, customers, partners) and cadence (weekly, monthly, launch, ad-hoc).

When to Use:

- Weekly status updates to leadership
- Monthly progress reports
- Launch announcements
- Escalation or risk communication
- Cross-functional coordination updates
- Customer-facing roadmap updates

Basic Usage:

```
/stakeholder-update
```

Claude asks: (1) What type of update? (2) Who is the audience?

Update Types:

Weekly — Regular cadence, progress and blockers focus **Monthly** — Higher-level, trends and milestones focus **Launch** — Feature/product announcement with impact **Ad-hoc** — One-off for specific situations (escalation, pivot, decision)

Audience Templates:

1. Executives / Leadership

```
Status: [Green/Yellow/Red]
TL;DR: [One sentence — most important thing]
Progress: [Outcomes tied to goals]
Risks: [Risk, mitigation, ask if needed]
Decisions needed: [Decision, options, recommendation, deadline]
Next milestones: [Milestone — Date]
```

Target length: Under 300 words. Lead with conclusion.

2. Engineering Team

```
Shipped: [Feature — Link — Impact]
In progress: [Item — Owner — Completion — Blockers]
Decisions: [Made/Needed — Context — Options]
Priority changes: [What and why]
Coming up: [Next items with context]
```

Include links to tickets and PRs. Explain priority changes.

3. Cross-Functional Partners

```
What's coming: [Feature — Date — Impact on your team]
What we need from you: [Ask — Context — Deadline]
Decisions made: [Decision — How it affects you]
Open for input: [Topic — How to provide feedback]
```

Focus on mutual dependencies and coordination needs.

4. Customers / External

```
What's new: [Feature — Benefit — How to use]
Coming soon: [Feature — Timing — Why it matters]
Known issues: [Issue — Status — Workaround]
Feedback: [How to share]
```

No internal jargon. Frame as benefits, not features.

Status Color Framework (G/Y/R):

The **stakeholder-comms** skill implements the Green/Yellow/Red framework:

- **Green** (On Track) — Progressing as planned, no significant risks
- **Yellow** (At Risk) — Slower than planned or risk materialized, mitigation underway
- **Red** (Off Track) — Significantly behind, major blocker, need intervention

Yellow is proactive risk communication, not failure. The earlier you flag yellow, the more options you have.

Tool Integration:

If project tracker is connected, Claude pulls roadmap status, completed items, at-risk items. If chat is connected, Claude searches for relevant discussions and decisions. If meeting transcription is connected, Claude pulls recent meeting notes. This automation saves the weekly "what did we do" recall exercise.

Risk Communication:

Uses the ROAM framework:

- **Resolved** — Risk no longer a concern
- **Owned** — Someone actively managing with mitigation plan
- **Accepted** — Known but proceeding, with documented rationale
- **Mitigated** — Actions reduced risk to acceptable level

Every risk includes: what, impact if it happens, likelihood, mitigation, ask.

Follow-Up Offers:

After generating the update, Claude offers to:

- Adjust tone or detail level
- Format for delivery channel (email, Slack, doc, slides)
- Post directly to chat if connected

7. Synthesize Research (/synthesize-research)

Purpose: Turn qualitative and quantitative user research from multiple sources into structured insights with themes, personas, and opportunity areas.

When to Use:

- After conducting user interviews
- Analyzing survey results
- Processing support ticket themes
- Synthesizing usability test findings
- Combining multiple research sources
- Preparing research readouts for stakeholders
- Identifying product opportunities from user data

Basic Usage:

```
/synthesize-research
```

Claude asks what research you have (paste notes, upload files, or pull from connected tools).

Accepted Research Inputs:

- **Pasted text** — Interview transcripts, notes, survey responses

- **Uploaded files** — Research docs, spreadsheets, PDFs
- **Knowledge base** (if connected) — Search and pull research documents
- **User feedback tool** (if connected) — Support tickets, feature requests, bug reports
- **Product analytics** (if connected) — Usage data, funnels, behavioral patterns
- **Meeting transcription** (if connected) — Interview recordings, user calls

Methodology:

The **user-research-synthesis** skill applies three frameworks:

1. Thematic Analysis

- Familiarization (read all data)
- Initial coding (tag observations)
- Theme development (group related codes)
- Theme review (validate against data)
- Theme refinement (define and name clearly)
- Report generation (findings with evidence)

2. Affinity Mapping

- Capture observations as individual notes
- Cluster related notes
- Label clusters
- Organize clusters into higher-level groups
- Identify cross-cluster themes

3. Triangulation

- Combine multiple research methods
- Compare findings across sources
- Strengthen confidence through convergence
- Investigate divergence (reveals segments or contexts)

What It Produces:

A structured research synthesis with six sections:

1. Research Overview

- Methodology (what types, how many participants)
- Research questions
- Timeframe

2. Key Findings (5-8 prioritized findings) For each:

- Finding statement (one clear sentence)
- Evidence (quotes, data points, observations with attribution)
- Frequency (how many participants/sources)
- Impact (significance to user experience or business)
- Confidence level (High/Medium/Low based on evidence strength)

3. User Segments / Personas If research reveals distinct segments:

- Segment name and description
- Characteristics and behaviors
- Unique needs and pain points
- Size estimate if available

4. Opportunity Areas

- Unmet or underserved user needs
- Where current solutions fall short
- New capabilities that would unlock value
- Prioritized by potential impact

5. Recommendations

- Specific, actionable next steps
- Tied to findings
- Prioritized by impact and feasibility

6. Open Questions

- What the research did not answer
- Suggested follow-up research

Evidence Standards:

Claude distinguishes between:

- **Behavioral observations** (what users did) — strongest evidence
- **Stated preferences** (what users said they want) — weaker evidence
- **Usage data** (quantitative behavior) — strong evidence at scale
- **Survey responses** — useful for frequency and distribution

Findings note confidence levels based on evidence strength.

Persona Framework:

Evidence-based personas include:

- Who they are (role, company, experience)
- What they are trying to accomplish (goals, jobs-to-be-done)
- How they use the product (frequency, workflows, tools)
- Key pain points (frustrations, workarounds)
- What they value (priorities in a solution)
- Representative quotes (2-3 verbatim)

Opportunity Sizing:

For each opportunity:

- Addressable users (how many affected)
- Frequency (how often encountered)
- Severity (impact when it occurs)
- Willingness to pay (revenue implications)

Scored as: Impact = Users × Frequency × Severity

Follow-Up Offers:

After synthesis, Claude offers to:

- Expand specific findings
- Generate persona documents
- Create opportunity maps
- Draft product implications (how findings should influence roadmap)
- Design follow-up research for open questions

8. Competitive Analysis (/competitive-brief)

Purpose: Research competitors and generate analysis briefs with feature comparisons, positioning assessment, and strategic implications.

When to Use:

- Preparing for a competitive deal
- Evaluating product strategy decisions
- Creating sales battle cards
- Updating quarterly competitive landscape

- Responding to competitor launches
- Planning product differentiation
- Investor or board competitive updates

Basic Usage:

```
/competitive-brief
```

Claude asks: (1) Which competitor(s)? (2) Focus area? (3) What decision will this inform?

Scope Options:

- **Full product comparison** — End-to-end feature and positioning analysis
- **Feature area deep-dive** — Specific capability (e.g., "Compare reporting features across Competitor A, B, C")
- **Pricing/packaging** — Business model and packaging strategies
- **Go-to-market** — Sales motion, target market, positioning
- **Strategic positioning** — Market position, differentiation claims, category definition

Research Sources:

Claude uses web search to gather:

- Product pages and feature lists
- Pricing pages and packaging details
- Recent launches, blog posts, changelogs
- Press coverage and analyst reports
- Customer reviews (G2, Capterra, TrustRadius, Gartner Peer Insights)
- Job postings (signal strategic direction)
- Social media and community discussions

If knowledge base is connected, Claude searches for existing competitive docs and win/loss reports.

What It Produces:

A competitive brief with seven sections:

1. Competitor Overview For each competitor:

- Company summary (founding, size, funding/revenue, target market)
- Product positioning (how they describe themselves)

- Recent momentum (launches, funding, partnerships, wins)

2. Feature Comparison Matrix format comparing capabilities:

CAPABILITY AREA	OUR PRODUCT	COMPETITOR A	COMPETITOR B
[Area 1]			
[Feature 1]	Strong	Adequate	Absent
[Feature 2]	Adequate	Strong	Weak

Rating scale (from **competitive-analysis** skill):

- **Strong** — Market-leading, deep functionality, well-executed
- **Adequate** — Functional, gets the job done, not differentiated
- **Weak** — Exists but limited, significant gaps
- **Absent** — Not available

3. Positioning Analysis For each competitor, extract:

- Target customer
- Category claim
- Key differentiator
- Value proposition

Framework: "For [target] who [need], [Product] is a [category] that [benefit]. Unlike [alternative], [Product] [differentiator]."

4. Strengths and Weaknesses For each:

- **Strengths** — Where they genuinely excel, what customers praise
- **Weaknesses** — Where they fall short, what customers complain about

Evidence-based (reviews, win/loss data), not dismissive.

5. Opportunities

- Gaps in competitor offerings we could exploit
- Customer asks no one addresses well
- Bets competitors are making we disagree with
- Market shifts that could advantage our approach

6. Threats

- Where competitors are investing heavily

- Competitive moves that could disrupt our position
- Where we are most vulnerable
- "Nightmare scenario" competitive moves

7. Strategic Implications

- What to build, accelerate, or deprioritize
- Where to differentiate vs achieve parity
- Positioning or messaging adjustments
- What to monitor going forward

Analysis Frameworks:

The **competitive-analysis** skill provides:

Feature Comparison Matrix — Structured capability comparison with consistent rating **Positioning Framework** — Systematic positioning statement extraction **Win/Loss Analysis** — Why deals are won or lost against each competitor **Market Trend Identification** — Spotting shifts that affect competitive dynamics

Follow-Up Offers:

After generating the brief, Claude offers:

- One-page executive summary
- Sales battle cards for competitive deals
- "How to win against [competitor]" playbook
- Competitive monitoring plan

Best Practices Embedded:

- Be honest about competitor strengths (dismissive analysis is useless)
- Focus on what matters to customers, not internal priorities
- Note assumptions and caveats (pricing often requires caveats)
- Job postings signal strategic direction (competitor hiring ML engineers = AI investment)
- Customer reviews are unfiltered truth about strengths/weaknesses
- Date the analysis (competitive landscape changes quickly)

9. Metrics Review (/metrics-review)

Purpose: Review product metrics, identify trends, compare against targets, and surface actionable insights.

When to Use:

- Weekly metrics check-ins
- Monthly or quarterly business reviews
- Post-launch metric analysis
- Investigating metric changes
- Setting or adjusting OKRs
- Preparing metrics reports for stakeholders

Basic Usage:

```
/metrics-review
```

Claude asks: (1) Time period? (2) Which metrics? (3) Known context (launches, incidents, seasonality)?

Data Sources:

If **product analytics** is connected (Amplitude, Pendo), Claude pulls:

- Key metrics for the time period
- Comparison data (previous period, year-over-year, targets)
- Segment breakdowns
- Funnel data

Without connection, paste or describe your metrics data.

Metrics Hierarchy:

The **metrics-tracking** skill organizes metrics in three tiers:

North Star Metric — Single metric capturing core value delivered

- Value-aligned (moves when users get value)
- Leading (predicts business success)
- Actionable (product team can influence)
- Understandable (everyone knows what it means)

L1 Metrics (Health Indicators) — 5-7 metrics painting complete picture

- **Acquisition** — New users finding the product
- **Activation** — New users reaching value moment
- **Engagement** — Active users getting value

- **Retention** — Users coming back
- **Revenue** — Value translating to money
- **Satisfaction** — How users feel about the product

L2 Metrics (Diagnostic) — Detailed metrics for investigation

- Funnel conversion at each step
- Feature-level usage
- Segment breakdowns
- Performance metrics
- Content-specific engagement

What It Produces:

A metrics review with six sections:

1. Summary 2-3 sentences: overall health, most notable changes, key callout

2. Metric Scorecard Table for quick scanning:

METRIC	CURRENT	PREVIOUS	CHANGE	TARGET	STATUS
[Metric]	[Value]	[Value]	[+/- %]	[Target]	[On track/At risk/Miss]

3. Trend Analysis For each important metric:

- What happened (magnitude of change)
- Why it likely happened (attribution with caveats)
- Whether it is one-time or sustained trend

4. Bright Spots What is going well:

- Metrics beating targets
- Positive trends to sustain
- Segments or features performing strongly

5. Areas of Concern What needs attention:

- Metrics missing targets or trending negatively
- Early warning signals
- Metrics lacking visibility or understanding

6. Recommended Actions Specific next steps:

- Investigations (dig deeper into concerning trends)

- Experiments (test improvement hypotheses)
- Investments (double down on what works)
- Alerts (monitor metrics more closely)

7. Context and Caveats

- Known data quality issues
- Events affecting comparability (outages, holidays, launches)
- Metrics not yet tracked but should be

Analysis Techniques:

Claude applies:

- **Trend identification** — Direction, magnitude, acceleration
- **Anomaly detection** — Sudden spikes or drops
- **Correlation analysis** — Which metrics move together
- **Segment breakdown** — Does aggregate hide divergent segments
- **Cohort analysis** — Are newer cohorts better than older ones

Common Metrics Explained:

DAU / WAU / MAU — Daily/Weekly/Monthly Active Users

- What counts as "active" matters (login vs core action)
- DAU/MAU ratio measures stickiness (>0.5 = daily habit)

Retention — % of users from period X still active in period Y

- D1 (next day), D7 (one week), D30 (one month)
- Plot retention curves by cohort to see improvement over time

Activation — % of new users reaching the value moment

- Strongly predictive of retention
- Measure time to activate (faster is better)

Conversion — % moving from one stage to next

- Map full funnel, identify biggest drop-offs
- Segment by source, plan, user type

OKR Framework:

For goal-setting, Claude implements:

- **Objectives** — Qualitative, aspirational, time-bound
- **Key Results** — Quantitative, specific targets (2-4 per objective)
- 70% completion = success for stretch OKRs

Follow-Up Offers:

After review, Claude offers:

- Deeper investigation of specific metrics
- Dashboard spec for ongoing monitoring
- Experiment proposals for areas of concern
- Metrics review template for recurring use

Part III: Skills Deep Dive

10. Feature Specification Expertise

The **feature-spec** skill provides comprehensive PRD and requirements knowledge. It loads automatically when conversations involve speccing features, writing requirements, or discussing product scope.

Core Knowledge Areas:

PRD Structure — Eight-section template (Problem/Goals/Non-Goals/Stories/Requirements/Metrics/Questions/Timeline)

User Story Format — "As a [user type], I want [capability] so that [benefit]" with guidelines for making stories independent, negotiable, valuable, estimable, small, and testable

Requirements Categorization — MoSCoW framework (Must/Should/Could/Won't) with guidance on ruthless P0 scoping

Acceptance Criteria — Given/When/Then format or checklist format covering happy path, errors, and edge cases

Success Metrics — Leading indicators (change quickly) vs lagging indicators (change slowly) with target-setting guidance

Scope Management — Recognizing and preventing scope creep through explicit non-goals and v1/v2 separation

Common Mistakes Addressed:

- Too-vague user stories ("As a user, I want the product to be faster")
- Solution-prescriptive stories ("I want a dropdown menu" instead of describing the need)
- Missing benefit ("I want to click a button" without explaining why)
- Too-large stories that cannot fit in one sprint
- Everything labeled P0 (if everything is P0, nothing is)
- Success metrics without specific targets or measurement methods

When This Skill Activates:

Trigger phrases include:

- "Write a spec", "create a PRD", "document requirements"

- "What should go in the spec", "user story format"
- "Acceptance criteria", "success metrics", "how to measure"
- "Scope creep", "out of scope", "what's in v1"

11. Roadmap Management & Prioritization

The **roadmap-management** skill provides frameworks for planning, prioritizing, and communicating roadmaps.

Roadmap Formats:

Now / Next / Later

- Now (current sprint/month) — Committed, high confidence
- Next (1-3 months) — Planned, good confidence on what
- Later (3-6+ months) — Directional, flexible scope/timing
- Best for: Most teams, external communication (avoids false precision)

Quarterly Themes

- 2-3 themes per quarter representing strategic investment areas
- Initiatives listed under themes
- Themes map to company/team OKRs
- Best for: Showing strategic alignment

OKR-Aligned

- Map roadmap items directly to Objectives and Key Results
- Include expected impact on each Key Result
- Best for: OKR-driven organizations

Timeline / Gantt

- Calendar view with start/end dates
- Shows parallelism, sequencing, dependencies
- Best for: Execution planning with engineering

Prioritization Frameworks:

RICE Score

- Reach (how many users affected per time period)
- Impact (how much per person: 3/2/1/0.5/0.25)
- Confidence (100% = data-backed, 50% = gut feel)

- Effort (person-months)
- Score = (R × I × C) / E
- Best for: Quantitative comparison of large backlogs

MoSCoW

- Must have (roadmap fails without these)
- Should have (important but delivery viable without)
- Could have (desirable, lower priority)
- Won't have (explicitly out of scope)
- Best for: Scoping releases, stakeholder negotiation

ICE Score

- Impact (1-10, how much it moves target metric)
- Confidence (1-10, confidence in impact estimate)
- Ease (1-10, inverse of effort)
- Score = I × C × E
- Best for: Quick prioritization, early-stage products

Value vs Effort Matrix (2×2)

- High value, Low effort = Quick wins (do first)
- High value, High effort = Big bets (plan carefully)
- Low value, Low effort = Fill-ins (spare capacity)
- Low value, High effort = Money pits (don't do)
- Best for: Visual prioritization in team sessions

Dependency Management:

Types identified:

- Technical (Feature B requires infrastructure from Feature A)
- Team (requires work from another team)
- External (vendor, partner, third-party)
- Knowledge (need research results before starting)
- Sequential (must ship A before starting B)

Management approach:

- List dependencies explicitly
- Assign owner to each

- Set "need by" date
- Build buffer (dependencies are high-risk)
- Have contingency plans

Capacity Planning:

Estimation:

- Start with team size × time period
- Subtract overhead (meetings, on-call, interviews, PTO)
- Rule of thumb: 60-70% of time on planned feature work

Allocation (healthy ratio):

- 70% planned features (roadmap items)
- 20% technical health (tech debt, reliability, performance)
- 10% unplanned (buffer for urgent issues, quick wins)

When This Skill Activates:

Trigger phrases:

- "Plan a roadmap", "prioritize features", "create a roadmap"
- "RICE", "MoSCoW", "ICE", "prioritization framework"
- "Now/Next/Later", "quarterly themes"
- "Dependencies", "what's blocking", "capacity planning"

12. Stakeholder Communications

The **stakeholder-comms** skill provides templates, frameworks, and best practices for PM communication across audiences.

Update Templates by Audience:

Already covered in Command #6. The skill provides the detailed templates that command uses.

G/Y/R Status Framework:

- **Green** — On track, no significant risks, meeting commitments
- **Yellow** — At risk, slower than planned, risk materialized but mitigation underway
- **Red** — Off track, major blocker, will miss commitments without intervention

Guidelines:

- Use Yellow proactively (early flagging creates options)

- Red means you need help (exhaust your own options first)
- Return to Green only when risk genuinely resolved
- Document what changed when status changes

ROAM Risk Framework:

- **Resolved** — Risk no longer a concern (document how)
- **Owned** — Someone actively managing (state owner + mitigation)
- **Accepted** — Proceeding without mitigation (document rationale)
- **Mitigated** — Actions reduced risk to acceptable level

Every risk includes: what, impact, likelihood, mitigation, ask.

Decision Documentation (ADRs):

Architecture Decision Record format:

- Status (Proposed/Accepted/Deprecated/Superseded)
- Context (situation requiring decision, forces at play)
- Decision (what was decided, stated clearly)
- Consequences (positive/negative, enables/prevents)
- Alternatives Considered (what was evaluated, why rejected)

When to write:

- Strategic product decisions
- Significant technical decisions
- Controversial decisions (document rationale for future)
- Decisions constraining future options
- Decisions people will question later

Meeting Facilitation:

Stand-up (15 min)

- Purpose: Surface blockers, coordinate work
- Format: Accomplished, working on next, blockers
- Tip: Focus on blockers (highest value), take discussions offline

Sprint Planning (60 min)

- Purpose: Commit to work for next sprint
- Format: Review, priorities, capacity, commitment, dependencies
- Tip: Come with proposed priority order, push back on overcommitment

Retrospective (60 min)

- Purpose: Reflect and identify improvements
- Format: Set stage, gather data, insights, decide actions, close
- Tip: Limit to 1-3 action items, follow up on previous items

Stakeholder Review (45 min)

- Purpose: Show progress, gather feedback, build alignment
- Format: Context, demo, metrics, feedback, next steps
- Tip: Demo real product (not slides), capture feedback visibly

When This Skill Activates:

Trigger phrases:

- "Write an update", "stakeholder communication"
- "Status report", "Green/Yellow/Red"
- "Risk communication", "ROAM"
- "Document a decision", "ADR"
- "Run a retrospective", "facilitate a meeting"

13. User Research Synthesis

The **user-research-synthesis** skill provides methodologies for turning raw research data into structured insights.

Thematic Analysis Process:

1. **Familiarization** — Read all data, get overall landscape feel
2. **Initial coding** — Tag observations with descriptive codes
3. **Theme development** — Group related codes into candidate themes
4. **Theme review** — Validate themes against data, ensure distinctness
5. **Theme refinement** — Define and name each theme clearly
6. **Report** — Write themes as findings with supporting evidence

Affinity Mapping:

1. **Capture** — One observation per note
2. **Cluster** — Group related notes (let categories emerge)
3. **Label** — Name each cluster
4. **Organize** — Arrange clusters into higher-level groups
5. **Identify themes** — Clusters and relationships reveal themes

Triangulation:

Strengthen findings by combining:

- Methodological (same question, different methods)
- Source (same method, different participants)
- Temporal (same observation, different times)

Finding supported by multiple sources/methods is much stronger.

Interview Note Analysis:

Extract from each interview:

- **Observations** — What participant described doing/experiencing/feeling
- **Direct quotes** — Verbatim statements illustrating points (attribute to type, not name)
- **Behaviors vs stated preferences** — What people DO vs what they SAY (behavior is stronger)
- **Signals of intensity** — Emotional language, frequency, workaround effort, impact

Cross-interview:

- Patterns (which observations appear across multiple)
- Frequency (how many mentioned each theme)
- Segments (do different user types have different patterns)
- Contradictions (reveals segments or contexts)
- Surprises (challenges assumptions)

Survey Data Interpretation:

Quantitative:

- Response rate (representativeness)
- Distribution (shape matters, not just average)
- Segmentation (aggregates mask differences)
- Statistical significance (caution with small samples)
- Benchmark comparison

Open-ended:

- Code like mini interview notes
- Count theme frequency
- Pull representative quotes
- Find themes not in structured questions

Persona Development:

Evidence-based (from research, not imagination):

1. Identify behavioral patterns (clusters of similar behaviors/goals/contexts)
2. Define distinguishing variables (what differentiates clusters)
3. Create profiles (name, behaviors, goals, pain points, context, quotes)
4. Validate with data (can you size each segment quantitatively)

Persona template:

- Who they are
- What they are trying to accomplish
- How they use the product
- Key pain points
- What they value
- Representative quotes

Opportunity Sizing:

Estimate for each finding:

- Addressable users (how many benefit)
- Frequency (how often encountered)
- Severity (impact when occurs)
- Willingness to pay (revenue implications)

Score: Impact = Users × Frequency × Severity

When This Skill Activates:

Trigger phrases:

- "Synthesize research", "analyze interviews"
- "Thematic analysis", "affinity mapping"
- "Create personas", "user segments"
- "Survey results", "research findings"
- "Opportunity sizing"

14. Competitive Intelligence

The **competitive-analysis** skill provides frameworks for competitive research, feature comparison, and strategic analysis.

Competitive Set Definition:

Direct — Same problem, same users, same approach (what customers evaluate against you) **Indirect** — Same problem, different approach (alternative ways to solve need) **Adjacent** — Don't compete today but could (same tech/customer/distribution, could expand) **Substitute** — Different ways to solve underlying need (hire person, use Excel, outsource)

Landscape Mapping:

Position competitors on meaningful axes:

- Breadth vs depth (suite vs point solution)
- SMB vs enterprise
- Self-serve vs sales-led
- Simple vs powerful
- Horizontal vs vertical

Choose axes revealing strategic positioning differences.

Feature Comparison Matrix:

1. Define capability areas (functional categories buyers use)
2. List specific capabilities under each area
3. Rate each competitor consistently

Rating scale:

- **Strong** — Market-leading, deep functionality
- **Adequate** — Functional, not differentiated
- **Weak** — Exists but limited, significant gaps
- **Absent** — Not available

Tips:

- Rate from real experience + reviews, not marketing
- Weight by customer importance
- Update regularly (stale fast)
- Be honest about where competitors lead

Positioning Analysis:

Extract for each competitor:

- Target customer

- Category claim
- Key differentiator
- Value proposition

Framework: "For [target] who [need], [Product] is a [category] that [benefit]. Unlike [alternative], [Product] [differentiator]."

Message Architecture:

Level 1 — Category (what they are) Level 2 — Differentiator (what makes them different) Level 3 — Value Proposition (outcome promised) Level 4 — Proof Points (evidence provided)

Win/Loss Analysis:

Data sources:

- CRM notes (available immediately, biased)
- Customer interviews post-decision (most valuable, least biased)
- Churned customer surveys
- Prospect surveys (lost deals)

Questions for wins:

- What problem solving?
- What alternatives evaluated?
- Why chose us over alternatives?
- What almost made you choose someone else?

Questions for losses:

- What did you choose? Why?
- Where did we fall short?
- What could we have done differently?

Analysis:

- Track win/loss reasons over time
- Segment by deal type
- Identify top 3-5 reasons for wins and losses
- Distinguish product vs non-product reasons
- Calculate competitive win rates

Market Trend Identification:

Sources:

- Analyst reports (Gartner, Forrester, IDC)
- VC investment (where is smart money going)
- Conference themes
- Technology shifts (new platforms/APIs)
- Regulatory changes
- Customer behavior changes
- Talent movement

Framework:

- What is changing
- Why now
- Who is affected
- Timeline
- Implication for us
- What are competitors doing

Response options:

- **Lead** — Invest early, define category (high risk/reward)
- **Fast follow** — Wait for demand signals, move quickly
- **Monitor** — Track but don't invest yet, set triggers
- **Ignore** — Explicitly decide not relevant, document why

When This Skill Activates:

Trigger phrases:

- "Competitive analysis", "analyze competitor"
- "Feature comparison", "how do we compare"
- "Positioning", "differentiation"
- "Win/loss", "why did we lose"
- "Market trends", "competitive landscape"

15. Metrics & Analytics

The **metrics-tracking** skill provides frameworks for defining, tracking, analyzing, and acting on product metrics.

North Star Metric:

Single metric capturing core value delivered. Must be:

- Value-aligned (moves when users get value)
- Leading (predicts business success)
- Actionable (team can influence)
- Understandable (everyone knows what it means)

Examples:

- Collaboration tool: Weekly active teams with 3+ contributors
- Marketplace: Weekly transactions completed
- SaaS: Weekly active users completing core workflow
- Content: Weekly engaged reading/viewing time
- Developer tool: Weekly deployments using tool

L1 Metrics (Health Indicators):

5-7 metrics painting complete picture across lifecycle:

Acquisition

- New signups/trials (volume + trend)
- Signup conversion rate
- Channel mix
- Cost per acquisition

Activation

- Activation rate (% reaching value moment)
- Time to activate
- Setup completion rate
- First value moment timing

Engagement

- DAU / WAU / MAU
- DAU/MAU ratio (stickiness)
- Core action frequency
- Session depth
- Feature adoption

Retention

- D1, D7, D30 retention

- Cohort retention curves
- Churn rate
- Resurrection rate

Revenue

- Conversion rate (free to paid)
- MRR / ARR
- ARPU / ARPA
- Expansion revenue
- Net revenue retention

Satisfaction

- NPS
- CSAT
- Support ticket volume/resolution time
- App store ratings

L2 Metrics (Diagnostic):

Detailed metrics for investigation:

- Funnel conversion by step
- Feature-level usage
- Segment breakdowns
- Performance metrics
- Content-specific engagement

OKR Framework:

Objectives: Qualitative, aspirational, time-bound Key Results: Quantitative targets, 2-4 per objective

Example:

```
Objective: Make product indispensable for daily workflows

Key Results:
- Increase DAU/MAU from 0.35 to 0.50
- Increase D30 retention from 40% to 55%
- 3 core workflows with >80% completion rate
```

Best practices:

- Ambitious but achievable (70% completion = success)
- Outcomes not outputs
- 2-3 objectives with 2-4 KRs each
- Should be uncomfortable (not confident hitting all)
- Review mid-period, grade honestly at end

Metric Review Cadences:

Weekly (15-30 min, PM + eng lead)

- North Star + key L1s
- Active experiments
- Anomalies
- Alerts

Monthly (30-60 min, product team + stakeholders)

- Full L1 scorecard
- Progress vs OKRs
- Cohort analysis
- Feature adoption
- Segment analysis

Quarterly (60-90 min, product/eng/design/leadership)

- OKR scoring
- Trend analysis over quarter
- Year-over-year comparison
- Competitive context
- What worked/didn't

Dashboard Design Principles:

1. Start with the question (what decisions does this support)
2. Hierarchy of information (most important most prominent)
3. Context over numbers (comparison + trend always)
4. Fewer metrics, more insight (focus on 5-10)
5. Consistent time periods
6. Visual status indicators (Green/Yellow/Red)
7. Actionability (can the team influence it)

Layout:

- Top: North Star with trend + target
- Second: L1 scorecard (current, change, target, status)
- Third: Key funnels
- Fourth: Experiments and launches
- Bottom: L2 metrics and drill-downs

When This Skill Activates:

Trigger phrases:

- "Product metrics", "set goals", "OKRs"
- "DAU", "MAU", "retention", "conversion"
- "North Star metric"
- "Dashboard", "metrics review"
- "How do we measure"

Part IV: Connected Tools & Workflows

16. MCP Server Connections

The plugin pre-configures 12 MCP servers across 7 categories. Each connection enriches Claude's ability to pull real data rather than working from what you paste.

Project Tracker Category (`~project tracker`)

Linear (HTTP, https://mcp.linear.app/mcp)

- Roadmap items, issues, projects
- Statuses, assignees, priorities
- Dependencies, milestones, cycles
- Use for: Roadmap updates, status tracking

Asana (HTTP, https://mcp.asana.com/v2/mcp)

- Tasks, projects, portfolios
- Milestones, assignees, custom fields
- Use for: Task-based roadmaps, team coordination

monday.com (HTTP, https://mcp.monday.com/mcp)

- Boards, items, updates
- Workflows, automations, status
- Use for: Visual workflow roadmaps

ClickUp (HTTP, https://mcp.clickup.com/mcp)

- Tasks, lists, spaces, goals
- Time tracking, dependencies
- Use for: Hierarchical roadmap structures

Atlassian (HTTP, https://mcp.atlassian.com/v1/mcp)

- Jira tickets, epics, sprints
- Confluence pages, spaces
- Use for: Enterprise development workflows

Chat Category (`~chat`)

Slack (HTTP, https://mcp.slack.com/mcp)

- Channel messages, threads
- Team discussions, decisions
- DMs for context
- Use for: Pulling recent discussions into updates, finding decisions

Knowledge Base Category (`~knowledge base`)

Notion (HTTP, https://mcp.notion.com/mcp)

- Pages, databases
- Meeting notes, research docs
- Specs, design docs
- Use for: Searching existing context, prior work

Design Category (`~design`)

Figma (HTTP, https://mcp.figma.com/mcp)

- Design files, frames
- Prototypes, components
- Comments, versions
- Use for: Referencing designs in specs, understanding design context

Product Analytics Category (`~product analytics`)

Amplitude (HTTP, https://mcp.amplitude.com/mcp)

- User behavior, events
- Funnels, retention, cohorts
- Segmentation, user properties
- Use for: Metrics reviews, user research validation

Pendo (HTTP, https://app.pendo.io/mcp/v0/shttp)

- Feature usage, adoption
- Guides, feedback
- Product analytics
- Use for: Feature-level metrics, in-product feedback

User Feedback Category (`~user feedback`)

Intercom (HTTP, https://mcp.intercom.com/mcp)

- Support conversations, tickets
- Feature requests, feedback
- User data, segments
- Use for: Research synthesis, finding pain points

Meeting Transcription Category (`~meeting transcription`)

Fireflies (HTTP, https://api.fireflies.ai/mcp)

- Meeting transcripts, summaries
- Action items, topics
- Speaker insights
- Use for: Pulling meeting context, research interview notes

Authentication:

All included servers use HTTP type with OAuth where needed. First time you use a command that accesses a server, Claude prompts authentication through the service's standard flow. After that, it is automatic.

Alternative Tools:

The plugin's tool-agnostic design means you can swap any of these for alternatives in the same category:

- **Project tracker alternatives**: Shortcut, Basecamp, Azure DevOps
- **Chat alternatives**: Microsoft Teams, Discord (for team usage)
- **Knowledge base alternatives**: Confluence, Guru, Coda, Google Docs
- **Design alternatives**: Sketch, Adobe XD
- **Analytics alternatives**: Mixpanel, Heap, FullStory, Google Analytics
- **User feedback alternatives**: Productboard, Canny, UserVoice, Zendesk
- **Transcription alternatives**: Gong, Dovetail, Otter.ai

To use an alternative, you need its MCP server configuration. See Part VII for configuration details.

17. Cross-Tool Workflows

The plugin's power comes from combining data across tools. Here are common multi-tool workflows:

Workflow 1: Stakeholder Update from Integrated Data

```
/stakeholder-update weekly update for execs
```

Claude pulls:

1. **Project tracker** — Completed items, in-progress status, blocked items
2. **Chat** — Recent team discussions, decisions made in channels
3. **Meeting transcription** — Key decisions from recent meetings
4. **Analytics** (if applicable) — Metric movements

Combines into G/Y/R status update with progress, risks, and decisions.

Workflow 2: PRD with Full Context

```
/write-spec Add SSO support
```

Claude gathers:

1. **Project tracker** — Existing SSO-related tickets, linked dependencies
2. **Knowledge base** — Prior specs, security docs, architecture decisions
3. **Design** — Existing authentication flow designs
4. **User feedback** — Support tickets requesting SSO

Generates PRD grounded in existing context, not from scratch.

Workflow 3: Research Synthesis from Multiple Sources

```
/synthesize-research onboarding experience
```

Claude combines:

1. **Meeting transcription** — Recent user interview transcripts
2. **User feedback** — Support tickets about onboarding
3. **Analytics** — Activation funnel data, drop-off points
4. **Knowledge base** — Existing research notes

Triangulates across qualitative and quantitative sources.

Workflow 4: Metrics-Informed Roadmap Prioritization

```
/roadmap-update reprioritize based on recent metrics
```

Claude:

1. **Analytics** — Pulls recent metric trends
2. **Project tracker** — Retrieves current roadmap items
3. Applies prioritization framework considering metric impact
4. Suggests reprioritization with data backing

Workflow 5: Competitive Brief with Internal Intel

```
/competitive-brief [Competitor X]
```

Claude:

1. **Web search** — Public competitive intelligence
2. **Knowledge base** — Searches for existing competitive docs, win/loss reports
3. **Chat** — Finds recent competitive mentions in sales channels
4. Synthesizes external research + internal knowledge

Data Privacy:

All tool connections are under your control. Claude only accesses data you have permissions for. You can disconnect any MCP server at any time.

18. Data Sources and Integration Patterns

Pattern 1: Pull Then Synthesize

Command asks for context, pulls from tools if available, falls back to asking you if not.

Example: `/stakeholder-update`

- If project tracker connected: Pull roadmap status automatically
- If not connected: Ask user to describe progress

Pattern 2: Search Then Reference

Command searches connected tools for relevant context, includes findings in output with citations.

Example: `/write-spec`

- Searches knowledge base for related specs
- References findings: "Based on the authentication architecture doc in Notion..."

Pattern 3: Validate Then Report

Command processes your input, validates against tool data, flags discrepancies.

Example: `/metrics-review` with pasted data

- If analytics connected: Validates pasted numbers against source
- Flags if numbers don't match: "Note: Your pasted DAU differs from Amplitude data"

Pattern 4: Augment Then Enhance

Command works from your input, augments with tool data to add depth.

Example: `/synthesize-research` with pasted interview notes

- Processes pasted notes as primary source
- Augments with support tickets from Intercom matching themes
- Enriches with usage data from analytics

Integration Levels:

Level 0: No Connections

- All commands functional
- You provide all context manually (paste, upload, describe)
- Claude acts as expert applying frameworks to your data

Level 1: Single Tool

- Connect one tool (typically project tracker)
- Commands pull basic context automatically
- Reduces manual data entry for roadmap/update commands

Level 2: Core Stack

- Connect project tracker + chat + knowledge base
- Significant automation in pulling context
- Cross-reference across sources

Level 3: Full Integration

- All relevant tools connected
- Maximum automation and cross-tool synthesis
- Claude acts as centralized intelligence layer across your stack

Most teams see value at Level 1-2. Level 3 is powerful but requires more setup.

Part V: Advanced Usage

19. Customization for Your Organization

Every company has unique processes, terminology, and tools. The plugin supports three customization approaches:

Approach A: Direct Plugin Modification

Edit plugin files directly (`.md` skill files, command files).

When to use: Quick personal experiments, testing ideas, small additions.

Example: Add your company's brand voice rules to `skills/stakeholder-comms/SKILL.md`

Risk: Plugin updates may overwrite changes.

Approach B: Fork the Plugin

Copy entire plugin directory, rename, modify the copy, install as separate plugin.

When to use: Team-wide customizations, significant modifications, maintaining changes across updates.

Steps:

1. Copy plugin directory: `product-management/` → `product-management-acme/`
2. Edit `.claude-plugin/plugin.json`: Change name to `"product-management-acme"`
3. Make modifications to the copy
4. Install forked plugin

Benefit: Clean separation, full control, can track changes in version control.

Approach C: Complementary Skills via CLAUDE.md

Keep plugin untouched. Add project-level skills in `.claude/skills/` and routing rules in `CLAUDE.md`.

When to use: Project-specific knowledge, layering company knowledge on general plugin, avoiding plugin updates conflicts.

Example:

Create `.claude/skills/acme-processes/SKILL.md`:

```
---
name: acme-processes
description: Acme Corp product processes and approval workflows.
  Use when planning at Acme, routing for review, or following
  go-to-market playbook.
---

# Acme Product Processes

## Campaign Approval Chain
[Details ... ]
```

Add to `CLAUDE.md`:

```
## Acme Product Management Rules

When using product-management plugin commands:
- Apply acme-processes skill for approval routing
- Follow Acme's PRD template additions
- Use Acme priority scoring model
```

Benefit: Zero plugin modification, receives plugin updates cleanly, project-specific.

Choosing an Approach:

NEED	RECOMMENDED APPROACH
Quick test of an idea	A (Direct modification)
Personal workflow enhancement	A
Team-wide process standardization	B (Fork)
Company-specific terminology	B (Fork)
Project-specific knowledge	C (CLAUDE.md)
Maintaining through plugin updates	B or C

20. Industry-Specific Extensions

The plugin is general-purpose but can be extended for industry-specific needs. Example: Pharmaceutical/Life Sciences PM.

What Industry-Specific Customization Adds:

Terminology

- General: Features, users, campaigns
- Pharma: Indications, HCPs, congress marketing, MLR review

Compliance

- General: Brand guidelines
- Pharma: FDA promotional rules, fair balance, off-label restrictions

Audiences

- General: Users, customers
- Pharma: HCPs by specialty, KOLs (Tier 1/2/3), payers, MSLs

Campaign Types

- General: Product launch, awareness
- Pharma: Disease awareness (branded/unbranded), congress, peer-to-peer, patient support

Channels

- General: Social, email, web
- Pharma: Medical journals, conference booths, MSL visits, medical portals

KPIs

- General: Leads, conversions, engagement
- Pharma: Share of voice, prescriber adoption, formulary wins, NRx/TRx

Extension Pattern (Approach B - Fork):

1. Fork plugin to `product-management-pharma`
2. Modify existing skills:
   - `feature-spec`: Add MLR review requirements section
   - `stakeholder-comms`: Add pharma approval chain templates
   - `metrics-tracking`: Add pharma KPIs (NRx, TRx, market share)
3. Add new skills:
   - `regulatory-compliance`: Fair balance, MLR process, FDA/EMA rules
   - `medical-affairs`: KOL engagement, advisory boards, publications
   - `market-access`: Formulary strategy, payer evidence, HEOR
4. Update CONNECTORS.md with pharma-specific tools

Example New Skill (`skills/regulatory-compliance/SKILL.md`):

```
---
name: regulatory-compliance
description: Pharma marketing regulatory compliance. Use when
  reviewing promotional content, preparing for MLR review,
  checking fair balance, or ensuring FDA/EMA compliance.
---

# Regulatory Compliance for Pharma Marketing

## Fair Balance Checklist
- [ ] Efficacy claims supported by cited clinical data
- [ ] Risk information equal prominence to efficacy
- [ ] Indication matches approved labeling
- [ ] No off-label claims
- [ ] Black box warnings displayed if applicable

## MLR Review Workflow
1. Medical review (clinical accuracy)
2. Legal review (compliance and risk)
3. Regulatory review (labeling alignment)
4. Final approval (all three sign-off)
```

Similar Patterns for Other Industries:

Financial Services: Compliance (FINRA, SEC), risk disclosures, fiduciary standards, client segments (retail/institutional/HNW)

Healthcare: HIPAA compliance, clinical workflows, care team roles, patient outcomes, interoperability standards

Legal Tech: Matter management, e-discovery, compliance, client privilege, billing models

Manufacturing: Supply chain, quality control, regulatory (FDA for medical devices), production workflows, BOM management

Education: Learning outcomes, accreditation, pedagogy frameworks, student segments, assessment methods

21. Best Practices and Patterns

Best Practice 1: Start with the Command, Explore the Skill

Commands are entry points. Skills are reference knowledge. Start by running a command (e.g., `/write-spec`). Once you understand what it produces, explore the underlying skill (`feature-spec`) to understand the frameworks it applies.

Best Practice 2: Provide Context Upfront When Possible

```
Good: /write-spec Add SSO for enterprise customers, targeting companies
      with 100+ users, needed for 3 Q1 deals worth $500K ARR

Basic: /write-spec SSO
```

More context upfront means fewer questions and faster results.

Best Practice 3: Connect Tools Incrementally

Don't try to connect all 12 tools at once. Start with one (project tracker), see the value, add another (knowledge base or chat), build from there.

Best Practice 4: Use Commands for Artifacts, Chat for Exploration

Commands generate structured outputs (specs, roadmaps, updates). Use them when you need a deliverable.

Regular chat (without invoking a command) taps the skills for exploratory questions:

- "What prioritization framework should I use for this situation?"
- "How should I structure this stakeholder update?"
- "What's the difference between a P0 and P1 requirement?"

Best Practice 5: Iterate on Outputs

First draft is rarely final. After Claude generates a spec/roadmap/update:

```
You: Expand the success metrics section with more specific targets
You: Adjust the tone for a more technical audience
You: Add a section on rollback plan
```

Claude refines based on your feedback.

Best Practice 6: Reference Prior Work

```
/write-spec authentication improvements, similar structure to
the API v2 spec we did last month
```

Claude can reference previous conversations, connected knowledge base docs, or uploaded files.

Best Practice 7: Combine Commands in Workflows

```
Day 1: /write-spec [feature]
Day 7: /synthesize-research from user interviews about [feature]
       → Update spec based on findings
Day 14: /roadmap-update add [feature] to Q2 roadmap
Day 30: /stakeholder-update monthly update including [feature] progress
```

Commands compose into end-to-end PM workflows.

Anti-Pattern 1: Expecting Mind-Reading

```
Bad: /write-spec
     Claude: What feature?
     You: You should know

Good: /write-spec SSO support
      Claude: [Asks clarifying questions]
      You: [Provides answers]
```

Claude is an expert assistant, not a mind reader. Provide the seed, Claude grows it.

Anti-Pattern 2: Pasting Without Context

```
Bad: /synthesize-research
     [Paste 50 pages of notes with no explanation]

Good: /synthesize-research from 12 user interviews about onboarding
      [Paste notes]
```

Context about what the data is helps Claude process it correctly.

Anti-Pattern 3: Treating Outputs as Final Without Review

Generated specs, roadmaps, and updates are drafts requiring your judgment. Review for:

- Accuracy (did Claude misunderstand something?)
- Completeness (what's missing?)
- Appropriateness (right tone/detail level for audience?)

Anti-Pattern 4: Using the Wrong Command

Each command has a purpose:

- Don't use `/write-spec` to create a roadmap (use `/roadmap-update`)
- Don't use `/metrics-review` to synthesize interviews (use `/synthesize-research`)

If unsure which command fits, describe your goal in regular chat and ask Claude which command to use.

Part VI: Reference

22. Complete Tool Connector Guide

Tool Category: Project Tracker **Placeholder:** `~project tracker` **Included Servers:** Linear, Asana, monday.com, ClickUp, Atlassian (Jira/Confluence) **Other Options:** Shortcut, Basecamp, Azure DevOps, GitHub Projects **What It Provides:** Roadmap items, tickets, statuses, assignees, dependencies, milestones, sprints **Used By Commands:** `/roadmap-update`, `/stakeholder-update`, `/write-spec` (for related tickets)

Tool Category: Chat **Placeholder:** `~chat` **Included Servers:** Slack **Other Options:** Microsoft Teams, Discord **What It Provides:** Channel messages, threads, decisions, team discussions **Used By Commands:** `/stakeholder-update`, `/synthesize-research` (for decision context)

Tool Category: Knowledge Base **Placeholder:** `~knowledge base` **Included Servers:** Notion **Other Options:** Confluence, Guru, Coda, Google Docs (via custom MCP) **What It Provides:** Meeting notes, docs, specs, research, decision records **Used By Commands:** All (searches for prior work and context)

Tool Category: Design **Placeholder:** `~design` **Included Servers:** Figma **Other Options:** Sketch, Adobe XD, Framer **What It Provides:** Design files, prototypes, components, comments, versions **Used By Commands:** `/write-spec` (for design context)

Tool Category: Product Analytics **Placeholder:** `~product analytics` **Included Servers:** Amplitude, Pendo **Other Options:** Mixpanel, Heap, FullStory, Google Analytics **What It Provides:** User behavior, events, funnels, retention, cohorts, segments **Used By Commands:** `/metrics-review`, `/synthesize-research` (for behavioral data)

Tool Category: User Feedback **Placeholder:** `~user feedback` **Included Servers:** Intercom **Other Options:** Productboard, Canny, UserVoice, Zendesk **What It Provides:** Support tickets, feature requests, conversations, feedback **Used By Commands:** `/synthesize-research`, `/write-spec` (for user pain points)

Tool Category: Meeting Transcription **Placeholder:** `~meeting transcription` **Included Servers:** Fireflies **Other Options:** Gong, Dovetail, Otter.ai, Grain **What It Provides:** Transcripts, summaries, action items, topics, speaker insights **Used By Commands:** `/synthesize-research`, `/stakeholder-update` (for meeting context)

23. Frameworks Quick Reference

PRD Structure (feature-spec)

1. Problem Statement (2-3 sentences, evidence-based)
2. Goals (3-5 measurable outcomes)
3. Non-Goals (3-5 out-of-scope with rationale)
4. User Stories ("As [user], I want [capability] so that [benefit]")
5. Requirements (P0 Must/P1 Nice/P2 Future with acceptance criteria)
6. Success Metrics (leading + lagging with targets)
7. Open Questions (tagged with owner)
8. Timeline Considerations

Prioritization (roadmap-management)

- **RICE**: (Reach × Impact × Confidence) / Effort
- **MoSCoW**: Must/Should/Could/Won't
- **ICE**: Impact × Confidence × Ease
- **Value vs Effort**: 2×2 (Quick wins/Big bets/Fill-ins/Money pits)

Stakeholder Update (stakeholder-comms)

- **G/Y/R Status**: Green (on track) / Yellow (at risk) / Red (off track)
- **ROAM Risk**: Resolved / Owned / Accepted / Mitigated
- **Update Structure**: Status, TL;DR, Progress, Risks, Decisions, Next

Research Synthesis (user-research-synthesis)

- **Thematic Analysis**: Familiarize → Code → Develop themes → Review → Refine → Report
- **Affinity Mapping**: Capture → Cluster → Label → Organize → Identify themes
- **Triangulation**: Multiple methods/sources/times = stronger findings

Competitive Analysis (competitive-analysis)

- **Feature Rating**: Strong / Adequate / Weak / Absent
- **Positioning**: For [target] who [need], [Product] is [category] that [benefit]. Unlike [alternative], [differentiator].
- **Win/Loss**: Why won, why lost, by competitor

Metrics (metrics-tracking)

- **Hierarchy**: North Star → L1 Health (Acquisition/Activation/Engagement/Retention/Revenue/Satisfaction) → L2 Diagnostic
- **OKRs**: Objective (qualitative) + 2-4 Key Results (quantitative targets)

- **Review Cadence:** Weekly (North Star + anomalies), Monthly (full L1), Quarterly (OKRs + strategy)

24. Templates and Examples

User Story Template

```
As a [specific user type],
I want [capability or action],
so that [benefit or outcome].

Acceptance Criteria:
- [ ] [Specific testable criterion]
- [ ] [Edge case or error state]
- [ ] [Performance or quality attribute]
```

Requirements Table Template

```
| Priority | Requirement | Acceptance Criteria |
|---|---|---|
| P0 | [Must-have requirement] | - [ ] Criterion 1<br>- [ ] Criterion 2 |
| P1 | [Nice-to-have requirement] | - [ ] Criterion 1 |
| P2 | [Future consideration] | [High-level description] |
```

Roadmap Item Format

```
## [Feature Name]

**Status**: [On Track / At Risk / Blocked / Done / Not Started]
**Owner**: [Name]
**Target**: [Q2 2026 / April 2026 / Now/Next/Later]
**Effort**: [2 weeks / 1 sprint / 5 story points]

**Description**: One-line what and why

**Dependencies**:
- [Dependency 1] — Owner, need by [date]
- [Dependency 2] — Blocker, status [Red/Yellow/Green]

**Success Metrics**:
- [Metric]: [Current] → [Target]
```

Executive Update Template

```
**Status**: 🟢 Green / 🟡 Yellow / 🔴 Red

**TL;DR**: [One sentence capturing the most important thing]

**Progress** (tied to goals/OKRs):
- [Outcome achieved]: [Metric movement or milestone]
- [Key result]: [% complete or actual vs target]

**Risks**:
- [Risk description]: [Mitigation plan]. [Ask: What we need]

**Decisions Needed**:
- [Decision]: [Options with recommendation]. Need by [date].

**Next Milestones**:
- [Milestone] — [Date]
- [Milestone] — [Date]
```

Research Finding Template

```
## Finding: [Clear one-sentence statement]

**Evidence**:
- "[Direct quote]" — Participant type, context
- "[Direct quote]" — Participant type, context
- [Data point or observation]

**Frequency**: [X of Y participants / Z support tickets]
**Impact**: [High/Medium/Low - how significantly this affects UX/business]
**Confidence**: [High/Medium/Low - evidence strength]

**Implication**: [What this means for the product]
```

Part VII: Extension Guide

25. Adding Custom Commands

Commands are Markdown files in the `commands/` directory. They instruct Claude on what to do when invoked.

Command File Structure:

```
---
description: Brief description for /help display (under 60 chars)
argument-hint: "<what user should provide>"
---

# Command Name

Instructions for Claude written in imperative form.

## Workflow

### Step 1: [First Action]
[What Claude should do]

### Step 2: [Second Action]
[What Claude should do]

## Output Format
[How to structure the output]

## Tips
[Guidance on best practices]
```

Example: Adding `/okr-review` Command

Create `commands/okr-review.md`:

```
---
description: Review OKR progress and recommend adjustments
argument-hint: "<quarter or time period>"
---

# OKR Review
```

```
Review Objectives and Key Results progress, identify risks, recommend adjustments.

## Workflow

### 1. Gather OKRs

If **~~project tracker** is connected:
- Pull current quarter OKRs and their Key Result metrics
- Get progress updates and status

If not connected:
- Ask user to provide OKRs (objectives and key results with current values)

### 2. Assess Progress

For each Key Result:
- Current value vs target
- Trajectory (on pace to hit target?)
- Confidence level (will we hit it?)

Scoring:
- 0.0-0.3: Significantly behind
- 0.4-0.6: Progressing but unlikely to fully achieve
- 0.7-1.0: On track or exceeded

### 3. Identify Risks

For at-risk KRs (< 0.6 projected):
- What's the blocker?
- What would it take to get back on track?
- Should we adjust target, increase effort, or accept miss?

### 4. Generate Review

Produce OKR scorecard with:

**Overall Assessment**: [X/Y Key Results on track]

**OKR Progress**:
For each Objective:
- Objective statement
- Key Results table: KR, Current, Target, Score, Status, Notes

**Risks and Mitigations**:
- At-risk KRs with recommended actions

**Recommendations**:
- Double-down (what's working, invest more)
- Course-correct (what's not working, change approach)
- Adjust targets (if circumstances changed)
```

```
## Output Format

Use tables for scorecards. Status colors: 🟢 On track, 🟡 At risk, 🔴 Off track.

## Tips

- OKRs are meant to be stretch goals. 70% achievement is success.
- If every KR is scoring 1.0, OKRs were not ambitious enough.
- Mid-quarter reviews allow course-correction. End-of-quarter is for learning.
- Focus recommendations on the highest-leverage actions.
```

Placement: Save as `/Users/tomi/Development/Zed/cowork_handbooks/cowork_plugins/product-management/commands/okr-review.md`

Usage: `/okr-review Q1 2026`

Custom Command Best Practices:

1. **Write for Claude, not users**: Commands are instructions to Claude, not documentation
2. **Imperative form**: "Generate a report" not "You should generate a report"
3. **Handle both connected and disconnected states**: "If X is connected, pull Y. If not, ask user for Y."
4. **Specify output structure clearly**: Use examples, templates, or detailed descriptions
5. **Include tips section**: Embed best practices Claude should follow

26. Creating Complementary Skills

Skills are knowledge packages that load when relevant topics arise. They live in `skills/skill-name/SKILL.md`.

Skill File Structure:

```
---
name: skill-identifier (matches directory name)
description: >
  Third-person description of when this skill should be used.
  Include specific trigger phrases: "use when X", "writing Y",
  "analyzing Z". This determines when Claude loads the skill.
---

# Skill Name

You are an expert at [domain]. You help [user type] [accomplish what].

## [Section 1]
[Core knowledge, frameworks, processes]

## [Section 2]
[Additional knowledge]

## [Section 3]
[Best practices, common mistakes]
```

Example: Adding Company-Specific `acme-pm-process` Skill

Create `skills/acme-pm-process/SKILL.md`:

```
---
name: acme-pm-process
description: >
  Acme Corp product management processes, approval workflows,
  and team structures. Use when planning work at Acme, routing
  documents for review, identifying stakeholders for decisions,
  understanding budget approval chains, or following Acme's
  product development lifecycle.
---

# Acme Product Management Processes

You are an expert at Acme Corp's product management workflows.
You help PMs navigate Acme's processes and ensure compliance
with company standards.

## PRD Approval Workflow

Every PRD must follow this review sequence:

1. **Draft** (PM creates)
2. **Engineering Review** (Eng lead reviews for feasibility, 2 days)
```

```
3. **Design Review** (Design lead reviews for UX consistency, 2 days)
4. **Product Director Approval** (Director reviews for strategy alignment, 3 days)
5. **Exec Approval** (VP Product if >$50K effort or strategic)

## Roadmap Prioritization Model

Acme uses a custom scoring model:

**Score = (Customer Impact × Strategic Fit × Urgency) / (Effort × Risk)**

Where:
- Customer Impact: 1-10 (based on # of customers affected × severity)
- Strategic Fit: 1-10 (alignment with company OKRs)
- Urgency: 1-10 (time sensitivity, competitive pressure)
- Effort: 1-10 (engineering weeks)
- Risk: 1-5 (technical uncertainty, dependency complexity)

Items scoring >5.0 = high priority. Items <2.0 = backlog.

## Budget Approval Tiers

| Effort Estimate | Approver | Timeline |
|---|---|---|
| <2 eng-weeks | PM + Eng Manager | No formal approval |
| 2-6 eng-weeks | Product Director | 3 business days |
| 6-12 eng-weeks | VP Product | 5 business days |
| >12 eng-weeks | VP Product + CTO | 10 business days |

## Stakeholder Map

| Initiative Type | Required Stakeholders |
|---|---|
| New feature | PM, Eng Lead, Design Lead, Product Dir |
| API change | PM, Eng Lead, DevRel, Solutions Arch |
| Pricing change | PM, Product Dir, Sales VP, Finance |
| Enterprise feature | PM, Eng Lead, Security, Compliance, Sales Eng |
| Sunset feature | PM, Eng Lead, Support, CS, Marketing |

## Communication Templates

**PRD Review Request** (Slack template):
```

@eng-lead @design-lead: PRD ready for review

Feature: [Name] Doc: [Link] Review needed by: [Date] Estimated effort: [Eng-weeks] Strategic priority: [High/Med/Low]

Questions for eng: [Specific questions] Questions for design: [Specific questions]

```
**Roadmap Change Notification** (Email template):
```

Subject: Roadmap Change: [Feature] moved to [New Quarter]

What changed: [Feature name] moved from [Old] to [New timeframe]

Why: [Reason - new priority, dependency slip, resource constraint]

Impact:

- Customers affected: [List if external commitment]
- Internal teams affected: [Sales, CS, Marketing, etc.]

Mitigation: [What we're doing to minimize impact]

Questions: [PM contact]

```
## Decision-Making Authority

| Decision Type | Owner | Escalation |
├──────────────┼──────┼──────────┤
| Feature scope (within initiative) | PM | Product Director if impacts timeline >2 wee
| Feature prioritization (within quarter) | Product Director | VP Product if cross-tea
| Roadmap changes (within quarter) | VP Product | CTO/CPO if affects committed deliver
| New initiative (out of quarter scope) | VP Product + CTO | CEO if requires significa

## Metrics Reporting Schedule

- **Weekly**: PM shares key metrics in #product-metrics Slack channel (Fridays)
- **Monthly**: Product Director presents to Product Leadership (first Monday)
- **Quarterly**: VP Product presents to Exec Team (QBR format)

Dashboard: [Internal Amplitude link]
Metrics template: [Notion template link]
```

Placement: Save to `skills/acme-pm-process/SKILL.md` in forked plugin OR `.claude/skills/acme-pm-process/SKILL.md` in project directory

Activation: When you invoke `/write-spec`, `/roadmap-update`, or `/stakeholder-update`, Claude automatically loads this skill if the trigger phrases in `description` match the conversation context.

Skills with Reference Files:

For more complex skills, add reference documents:

```
skills/acme-pm-process/
├── SKILL.md
├── references/
│   ├── prd-template.md
│   ├── roadmap-template.md
│   └── approval-workflow-details.md
└── examples/
    └── sample-prd.md
```

Reference these in SKILL.md:

```
## PRD Template

Acme uses a standard PRD template. See `references/prd-template.md`
for the complete structure with all required sections.
```

Claude loads SKILL.md automatically. Reference files load on-demand when Claude needs specific details.

27. Building Company-Specific Workflows

Combine custom commands and skills to create complete company-specific workflows.

Example: Acme Feature Launch Workflow

Step 1: Add custom command `/acme-launch-plan`

Create `commands/acme-launch-plan.md`:

```
---
description: Generate Acme Corp feature launch plan with all required artifacts
argument-hint: "<feature name>"
---

# Acme Launch Plan

Generate complete launch plan following Acme's launch playbook.

## Workflow

### 1. Gather Feature Context

Ask user:
```

```
- Feature name and description
- Target launch date
- Customer segment (SMB/Mid-market/Enterprise/All)
- Launch tier (Tier 1: Major, Tier 2: Standard, Tier 3: Minor)

Pull from **~~project tracker** if connected:
- Feature spec and requirements
- Engineering status and readiness

### 2. Generate Launch Checklist

Based on launch tier, generate checklist with owners and deadlines:

**All Tiers**:
- [ ] Product: Release notes drafted (L-14 days)
- [ ] Eng: Feature flags configured (L-7 days)
- [ ] Eng: Monitoring and alerts set up (L-7 days)
- [ ] Product: Internal demo recorded (L-7 days)
- [ ] CS: Support team trained (L-7 days)

**Tier 2+**:
- [ ] Marketing: Blog post drafted (L-14 days)
- [ ] Marketing: Email campaign created (L-10 days)
- [ ] Sales: Battle card updated (L-7 days)
- [ ] Product: Customer webinar scheduled (L+7 days)

**Tier 1**:
- [ ] Marketing: Press release (L-30 days)
- [ ] Marketing: Analyst briefings (L-14 days)
- [ ] Sales: Enablement session conducted (L-7 days)
- [ ] Exec: Board update prepared (L-30 days)

### 3. Generate Communication Plan

**Pre-Launch**:
- Internal announcement (#product-updates Slack, L-14 days)
- Sales preview (sales meeting, L-7 days)
- Beta customer heads-up (email, L-7 days)

**Launch Day**:
- Customer email (if Tier 1 or 2)
- Blog post publish
- Social media posts
- Internal celebration (#wins Slack)

**Post-Launch**:
- L+7: Adoption metrics review
- L+30: Success metrics review
- L+90: Retrospective
```

```
### 4. Generate Success Metrics Plan

Define metrics to track:
- Adoption (% of eligible customers using feature within 30 days)
- Activation (% of users who try feature and complete core workflow)
- Satisfaction (NPS or CSAT for users of the feature)
- Impact (movement of target product metric)

Set targets based on launch tier:
- Tier 1: >40% adoption, >70% activation, NPS >40, metric movement >10%
- Tier 2: >20% adoption, >60% activation, NPS >30, metric movement >5%
- Tier 3: Track adoption only

## Output Format

Markdown with three sections:
1. Launch Checklist (table with Task, Owner, Deadline, Status columns)
2. Communication Plan (timeline format)
3. Success Metrics (table with Metric, Target, How to Measure)

## Tips

- Tier 1 launches require exec-level coordination. Start planning 60+ days out.
- Feature flags allow decoupling deployment from launch. Use them.
- The most common launch mistake is under-preparing CS and Sales.
- Measure success metrics at L+7, L+30, L+90 — not just L+1.
```

Step 2: Update `acme-pm-process` skill with launch tier definitions

Add to `skills/acme-pm-process/SKILL.md`:

```
## Launch Tiers

Acme classifies feature launches into three tiers:

**Tier 1 (Major)**:
- Flagship features, major releases, new products
- Affects >50% of customer base or top-tier customers
- Significant competitive differentiation
- Typically 2-3 per year
- Examples: New product line, enterprise plan launch, major AI feature

**Tier 2 (Standard)**:
- Important features with broad applicability
- Affects 20-50% of customers or specific valuable segment
- Competitive parity or incremental differentiation
- Typically 6-8 per quarter
- Examples: Workflow automation, new integrations, reporting upgrades

**Tier 3 (Minor)**:
- Incremental improvements, bug fixes, small enhancements
- Affects <20% of customers or niche use case
- Quality-of-life improvements
- Continuous delivery
- Examples: UI tweaks, performance improvements, small feature additions
```

Step 3: Create supporting reference document

Create `skills/acme-pm-process/references/launch-playbook.md`:

```
# Acme Feature Launch Playbook

## Launch Tier Decision Tree
```

Is this a new product or major platform expansion? ├─ Yes → Tier 1 └─ No → Continue

Does this solve a top-3 customer pain point OR affect >50% of customers? ├─ Yes → Tier 1 └─ No → Continue

Does this provide competitive differentiation OR unlock new customer segment? ├─ Yes → Tier 2 └─ No → Continue

Does this affect >20% of customers with meaningful impact? ├─ Yes → Tier 2 └─ No → Tier 3

```
## Launch Timeline Templates

[Detailed timeline templates for each tier...]

## Cross-Functional RACI

[RACI matrix for launch activities...]
```

Step 4: Usage

```
User: /acme-launch-plan Real-time Collaboration

Claude: I'll create a launch plan. First, some context questions:
        - Target launch date?
        - Customer segment (SMB/Mid-market/Enterprise/All)?
        - What tier launch is this? Let me help determine...

        Based on Acme's tier criteria:
        - Is this a new product or major platform expansion? [Y/N]
        - Does it solve a top-3 customer pain point or affect >50% of customers?

User: Not a new product, but it's our #1 requested feature and affects all customers

Claude: Based on that, this is a Tier 1 launch. Generating complete launch plan...

[Produces full checklist, communication plan, and metrics plan per Acme's process]
```

Workflow Integration:

These company-specific workflows can chain together:

```
Week 1: /write-spec [feature]
        → Generates spec with Acme's PRD template

Week 2: Route spec through Acme approval workflow
        (PM → Eng → Design → Director → VP)

Week 8: /roadmap-update add [feature] to Q2 roadmap
        → Applies Acme priority scoring model

Week 12: /acme-launch-plan [feature]
         → Generates launch plan per Acme playbook

Launch day: /stakeholder-update launch announcement
            → Generates updates for each Acme audience

L+30: /metrics-review [feature] adoption and impact
      → Analyzes against Acme launch tier targets
```

Deployment:

For team-wide use:

1. Fork the plugin to `product-management-acme`
2. Add all custom commands and skills
3. Package as `.plugin` file
4. Distribute to Acme PM team
5. Maintain in version control with updates

For project-specific use:

1. Keep plugin standard
2. Add Acme-specific skills to `.claude/skills/` in project directory
3. Add routing rules to project's `CLAUDE.md`

Appendix: Agent Report

AGENT_REPORT_START **Plugin**: product-management **Version**: 1.0.0 **Files analyzed**: 18

- 1 plugin.json
- 1 .mcp.json
- 1 CONNECTORS.md
- 1 README.md

- 6 command files
- 6 skill files
- 1 plugin development guide
- 1 handbook (this document)

Handbook word count: ~18,500 words

Start time: 1771068403 (Feb 14, 2026 12:00:03) **End time:** 1771072800 (Feb 14, 2026 13:13:20) **Duration:** 73 minutes

Estimated token usage: ~63,000 input tokens (research), ~18,500 output tokens (handbook)

MCP Servers documented: 12 (Slack, Linear, Asana, monday, ClickUp, Atlassian, Notion, Figma, Amplitude, Pendo, Intercom, Fireflies)

Commands documented: 6 (/write-spec, /roadmap-update, /stakeholder-update, /synthesize-research, /competitive-brief, /metrics-review)

Skills documented: 6 (feature-spec, roadmap-management, stakeholder-comms, user-research-synthesis, competitive-analysis, metrics-tracking)

Frameworks covered: 23 (RICE, MoSCoW, ICE, Value-vs-Effort, OKRs, G/Y/R status, ROAM, Thematic Analysis, Affinity Mapping, Triangulation, ADRs, North Star, DAU/MAU, Retention, Activation, Conversion, Win/Loss, Positioning, Feature Comparison, and more)

AGENT_REPORT_END

RationalEyes.ai

contact@rationaleyes.ai

Intelligent Automation for Knowledge Work

Product Management Handbook — Version 1.0.0