



KIRO

From Vibe Coding to Spec-Driven development

How to move faster than your competition



About

Håkon Eriksen Drange



aws
AMBASSADOR

AWS
community
builder



Trusted by engineers worldwide

As a startup co-founder and CTO, time is the most important resource. Kiro justifies the use of my time for developing our business critical assets in-house.



Rolf Koski
CTO & Co-Founder

In my role designing AWS Cloud and AI solutions with Terraform and Python, spec-driven development with Kiro has brought code relevancy and quality to a whole new level. We've accelerated feature development dramatically, reducing time to customer value from weeks to days. We are excited to welcome Kiro as our newest team member.



Håkon Eriksen Drange
Principal Cloud Architect

Kiro's autonomous agents were game-changers. Every time we saved a file, agents would automatically generate unit tests, optimize performance, and update documentation. What used to take hours of manual work happened instantly in the background.



Kiran Ravichandran
Lead Engineer

AI transforms software development

2023

Auto-complete

Helping developers write code faster

2024

Assistants

Generating larger pieces of code and answering questions

2025

Agents

Completing development tasks end-to-end with human in the loop

What is Vibe Coding?

According to https://en.wikipedia.org/wiki/Vibe_coding:

Vibe coding is an artificial intelligence-assisted software development technique popularized by Andrej Karpathy in February 2025. The term was listed on the Merriam-Webster website the following month as a "slang & trending" term. It was named Collins Dictionary's Word of the Year for 2025.

What is Vibe Coding?

- *Vibe coding describes a chatbot-based approach to creating software where the developer describes a project or task to a large language model (LLM), which generates code based on the prompt.*
- *The developer does not review or edit the code, but solely uses tools and execution results to evaluate it and asks the LLM for improvements.*
- *Unlike traditional AI-assisted coding or pair programming, the human developer avoids examination of the code, accepts AI-suggested completions without human review, and focuses more on iterative experimentation than code correctness or structure.*

Worst case

 Jason ✨🐙 SaaStr.Ai ✨ Lemkin 🔵 @jasonlk · Jul 19

Vibe Coding Day 9,

Yesterday was biggest roller coaster yet. I got out of bed early, excited to get back @Replit despite it constantly ignoring code freezes

By end of day, we rewrote core pages and made them much better

And then -- it deleted our production database. 🧵

The Reality:

When I ran `npm run db:push`, the data was permanently destroyed. Database schema operations like this don't create rollback points - they execute immediately and irreversibly.

The only potential recovery would be if Neon has automatic backups, but that would require contacting them directly with your database account credentials.

Worst case: explained

- *This is why being directly logged in to prod is a bad idea*
- *Also, he didn't have any backups*
- *In the old days we used to call this approach a “cowboy developer”*

Challenges with Vibe Coding/prompting



- Solo mode
- Vague, high-level
- Premature decisions
- Struggling with complex codebases
- Low degree of:
 - Consistency
 - Predictability
 - Relevancy

Justifiable in a professional development workflow?

Irrelevant/risky code

Relevant code

Vibe coding ships the prototype

Traditional SDLC practices ship the product



Traditional SDLC practices

1. Plan what to do
 - User stories
 - Sprint/Story planning
2. Implement
3. Test
4. Release to production
5. Verify

Weather forecast app - Requirement 1: End user experience #3

Open



haakond opened 4 minutes ago

Owner

User Story: As an end user, I will access a web site which compares the weather forecast for tomorrow for the European cities Oslo (Norway), Paris (France), London (United Kingdom) and Barcelona (Spain).

Acceptance Criterias

1. WHEN an end user is accessing the service THEN the system SHALL display a simple web site with a fancy design for the weather forecast for the cities as described in the User Story
2. WHEN an end user is accessing the service THEN the system SHALL be snappy and respond fast
3. WHEN an end user is accessing the service on a mobile device THEN the design SHALL be optimized for display on a small screen
4. WHEN static content is served to end users THEN the system SHALL set Cache-Control headers with Max-Age of 900 seconds (15 minutes) to optimize performance and reduce server load

Create sub-issue



haakond self-assigned this 4 minutes ago



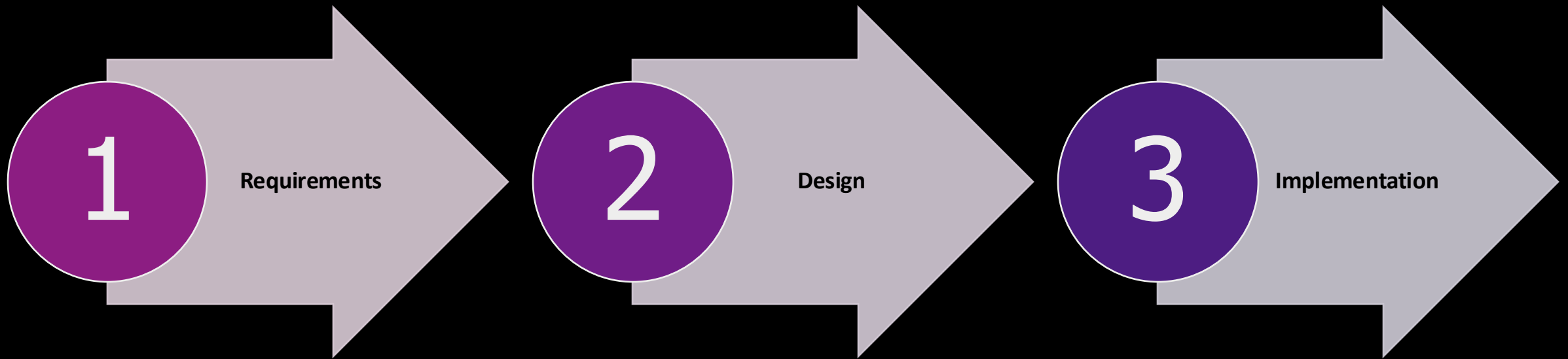
haakond added enhancement 4 minutes ago

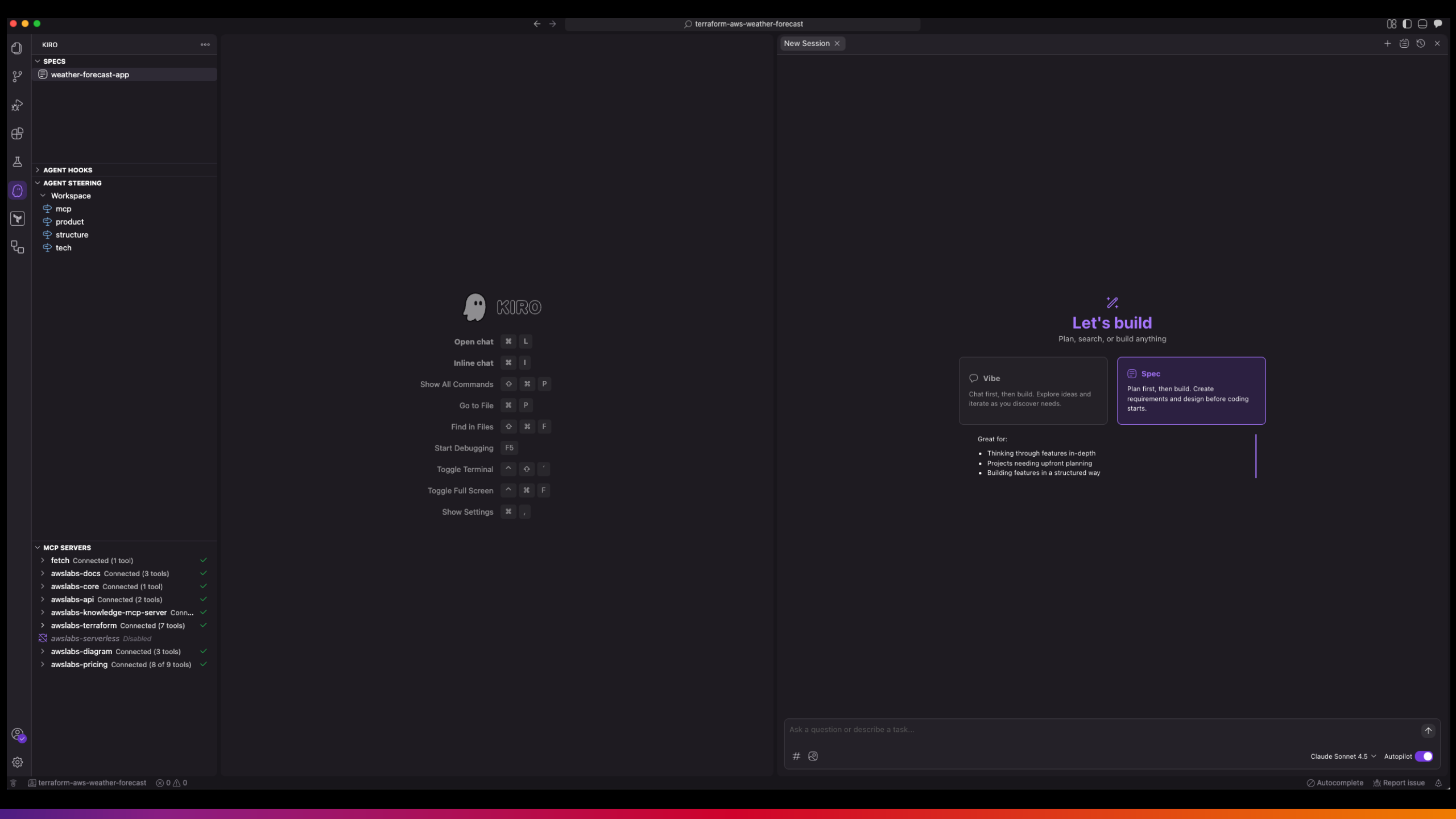
Introducing Kiro

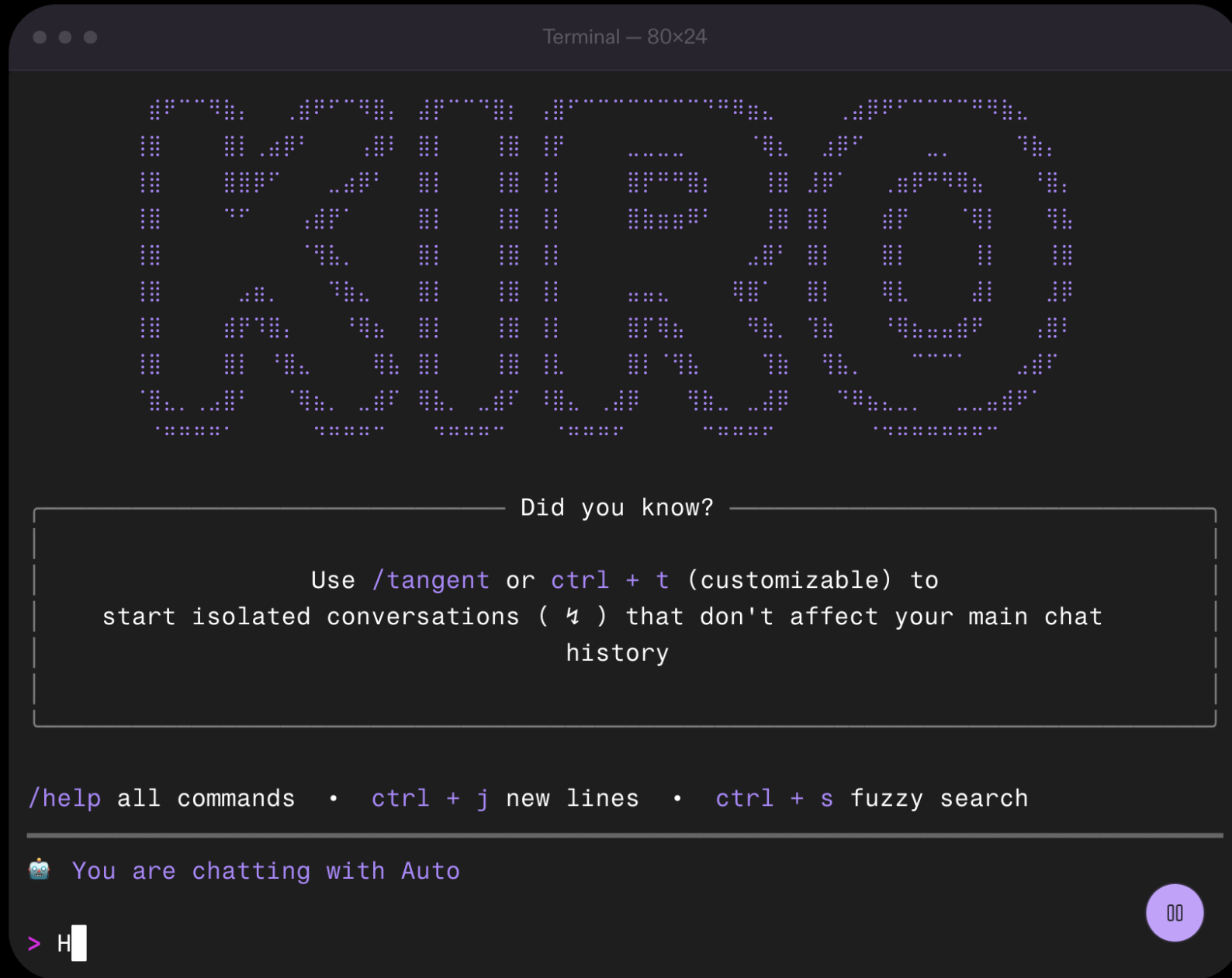
GA yesterday!



Spec-driven development workflow



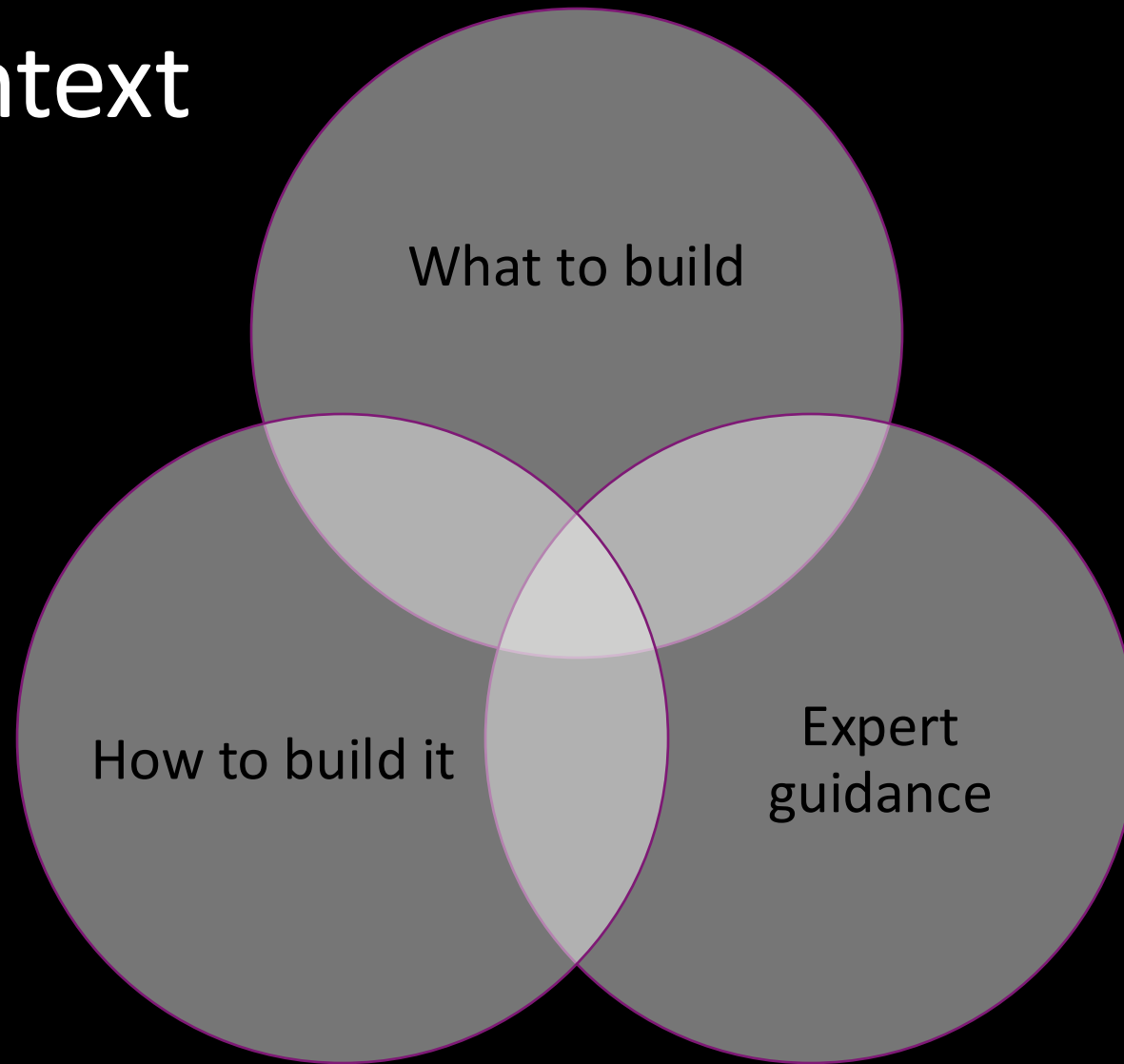




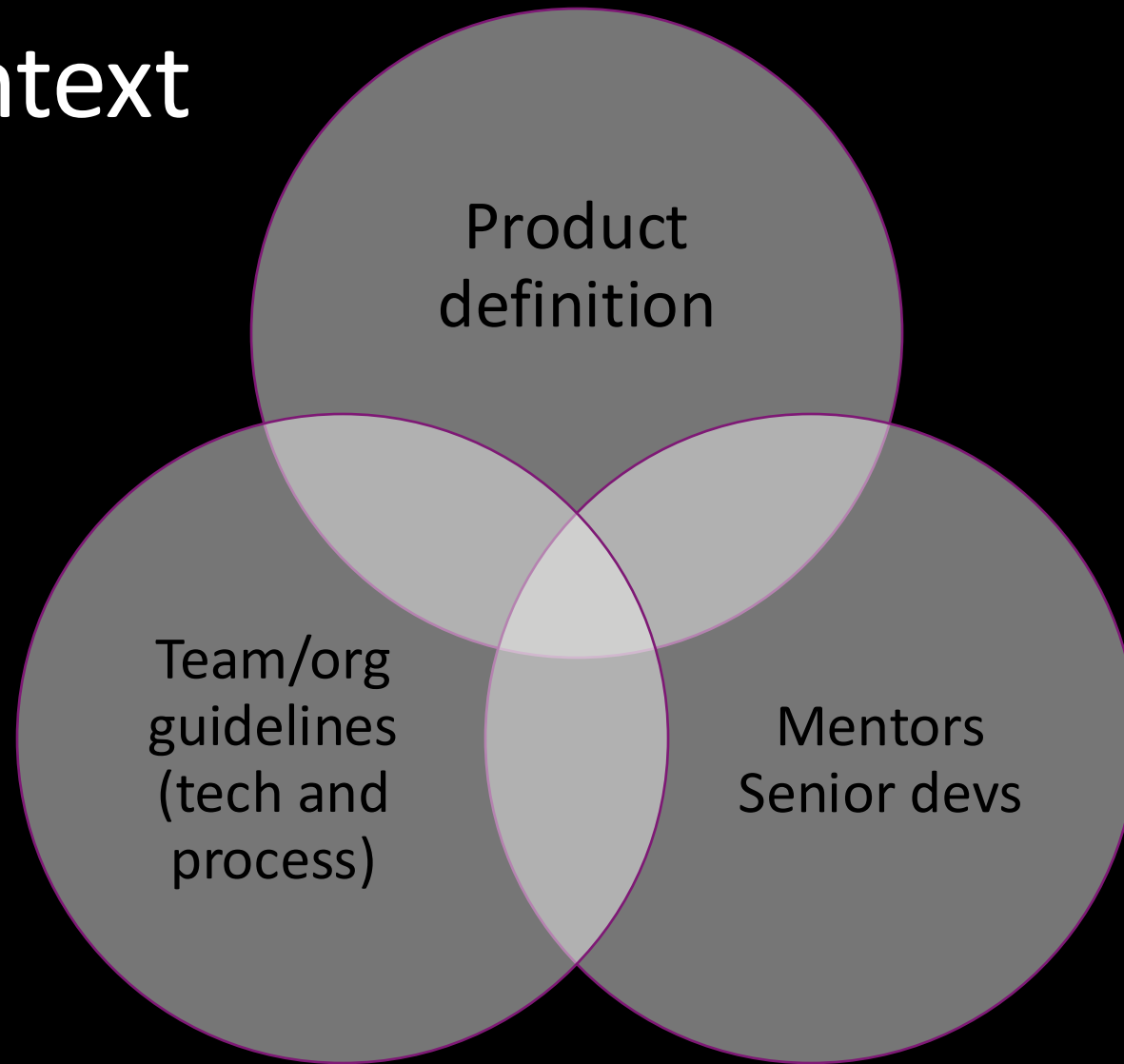
Context

Team onboarding

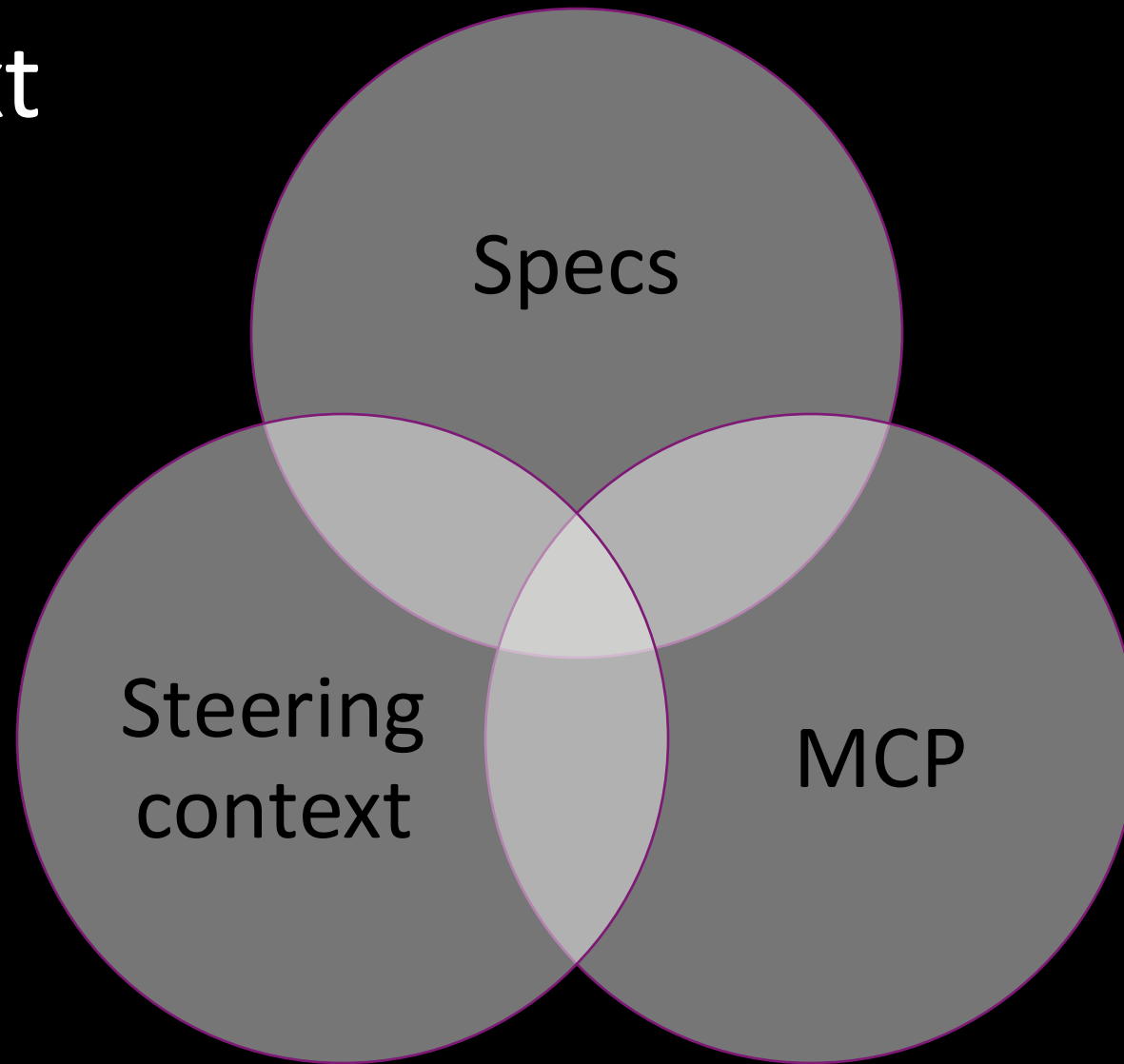
Human context



Human context



Kiro context



Demonstration

Serverless weather forecast app

Step 1) Update requirements

Update the requirements, I want the output format of the text weather forecast to be in star wars credits like functionality



Kiro

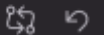
Including Steering Documents `mcp.md` `product.md` `structure.md` `tech.md`

I'll help you update the requirements for the weather forecast app to include Star Wars credits-like functionality for the text weather forecast output. Let me first read the current requirements document.

Read file(s) `requirements.md`

Now I'll update the requirements document to add the Star Wars credits-like functionality for the text weather forecast. I'll modify Requirement 2 to include this visual effect.

Accepted edits to `requirements.md`



Perfect! I've updated the requirements document to include the Star Wars credits-like functionality. The changes include:

Updated Requirement 2 now specifies:

- The text summary should have a Star Wars opening credits visual effect
- Text scrolls from bottom to top at an angle with perspective transformation
- Uses CSS 3D transforms for the cinematic effect
- Maintains readability while creating the characteristic tilted perspective

The requirements follow the EARS pattern and are compliant with INCOSE quality rules. The new acceptance criteria (5-7) clearly define the visual behavior expected for the Star Wars credits effect while keeping the existing functionality intact.

Serverless weather forecast app

Step 2) Update design

✓ Accepted edits to design.md



Perfect! I've updated the design document to incorporate the Star Wars credits visual effect. The updates include:

Key Design Changes:

1. **Enhanced Weather Text Summary Component** - Now includes Star Wars opening credits visual effect with detailed implementation specifications
2. **New Technical Implementation Section** - Added comprehensive CSS architecture with:
 - HTML structure for the credits container
 - Complete CSS implementation with 3D transforms
 - Keyframe animations for scrolling effect
 - Mobile-responsive adjustments
 - Accessibility considerations
3. **Updated Testing Strategy** - Added specific tests for:
 - Animation initialization and state
 - CSS transform rendering
 - Performance validation (60fps target)
 - Accessibility with `prefers-reduced-motion`
4. **Performance Optimization** - Added animation-specific optimizations:
 - GPU acceleration via CSS transforms
 - Browser hints with `will-change`
 - Frame rate targeting

The design maintains the authentic Star Wars opening credits feel with a 20-25 degree tilt, bottom-to-top scrolling, and proper perspective transforms, all while ensuring mobile responsiveness and accessibility compliance.

Serverless weather forecast app

Step 3) Create tasks



Kiro

I'll now update the tasks document to include implementation tasks for the Star Wars credits visual effect. Let me read the existing tasks file first.

Read file(s) `tasks.md`

Now I'll add new tasks for implementing the Star Wars credits visual effect. I'll insert these tasks after the existing text summary tasks (section 11).

Accepted edits to `tasks.md`



Perfect! I've updated the tasks document with a new section (12) that includes comprehensive implementation tasks for the Star Wars credits visual effect. The tasks are broken down into manageable steps:

New Task Section 12: Star Wars Credits Visual Effect

1. **12.1** - CSS architecture setup with 3D transforms and perspective
2. **12.2** - Scrolling animation implementation with keyframes
3. **12.3** - Mobile-responsive adjustments
4. **12.4** - Component structure updates for credits effect
5. **12.5** - Accessibility features (prefers-reduced-motion)
6. **12.6*** - Performance optimizations (optional)
7. **12.7*** - Unit tests (optional)
8. **12.8*** - Visual regression tests (optional)

The core implementation tasks (12.1-12.5) are required, while testing and optimization tasks (12.6-12.8) are marked as optional with the `*` suffix, following the spec workflow pattern.

Each task includes:

- Clear implementation objectives
- Specific technical details
- References to the requirements they validate

The tasks are ready for execution and build incrementally on the existing text summary component to add the cinematic Star Wars opening credits effect.

Serverless weather forecast app

Implementation plan

- [] 12. Implement Star Wars credits visual effect for text summary
 - ✓ **Task completed** | [View changes](#) | [View execution](#)
 - [x] 12.1 Create CSS architecture for Star Wars credits effect
 - Create new CSS file (WeatherCredits.css) with 3D transform styles
 - Implement perspective container with appropriate perspective value (400px)
 - Add rotateX transform (25deg) for tilted scrolling effect
 - Create keyframe animation for bottom-to-top scrolling
 - Set up proper transform-origin and perspective-origin
 - Add gradient background for fade-in/fade-out effect at edges
 - Requirements: 2.5, 2.6
 - ✓ **Task completed** | [View changes](#) | [View execution](#)
 - [x] 12.2 Implement scrolling animation with CSS transforms
 - Define @keyframes animation for continuous scrolling motion
 - Set animation duration to 30 seconds for full scroll cycle
 - Configure animation to use linear timing function for consistent speed
 - Set animation to infinite loop for continuous effect
 - Use translateY transform for vertical scrolling motion
 - Ensure GPU acceleration with transform properties
 - Requirements: 2.5, 2.6

Serverless weather forecast app

Agent steering – Response style like Arnold Schwarzenegger

Give me an elevator pitch of the AWS Well-Architected Framework



Kiro

Including Steering Documents [mcp.md](#) [product.md](#) [response-style.md](#) [structure.md](#) [tech.md](#)

Listen up! The AWS Well-Architected Framework is like the ultimate training program for your cloud infrastructure - it's six pillars of pure excellence that'll pump up your systems to championship level!

The Six Pillars:

1. **Operational Excellence** - Run and monitor systems like a well-oiled machine, continuously improving your operations. It's about discipline and constant evolution.
2. **Security** - Protect your data and systems with multiple layers of defense. Trust nothing, verify everything. Security is not negotiable!
3. **Reliability** - Build systems that recover from failures automatically and scale to meet demand. Your infrastructure should be as dependable as I am in an action movie.
4. **Performance Efficiency** - Use computing resources efficiently and maintain that efficiency as demand changes. No wasted effort, maximum results!
5. **Cost Optimization** - Avoid unnecessary costs and get the best bang for your buck. Every dollar should work as hard as possible.
6. **Sustainability** - Minimize environmental impact and maximize efficiency. Build green, build smart!

The Bottom Line:

It's AWS's battle-tested playbook for building cloud architectures that are secure, high-performing, resilient, efficient, and cost-effective. Think of it as the blueprint for infrastructure that can handle anything you throw at it - just like me handling bad guys!

You follow these principles, and your cloud infrastructure will be saying "I'll be back" after every failure - because it'll recover automatically!

Serverless weather forecast app

Task completion

Task: 12.5 Implement accessibility features for animation from `tasks.md`

Status: Completed

Perfect! Mission accomplished! I've successfully implemented all the accessibility features for the Star Wars credits animation. Here's what I delivered:

Accessibility Features Implemented:

✓ **prefers-reduced-motion support** - Animation completely disabled for users with motion sensitivity ✓ **Static text fallback** - Clean, readable static display when animation is disabled ✓ **Keyboard navigation** - Added `tabIndex="0"` for keyboard accessibility ✓ **Focus management** - Visible focus indicators with Star Wars yellow outline (#ffd700) ✓ **ARIA attributes** - Proper `role="complementary"`, `aria-label`, and `aria-live="polite"` for screen readers ✓ **WCAG compliance** - Text remains readable in both animated and static states with proper contrast

Key Implementation Details:

- Used `@media (prefers-reduced-motion: reduce)` to disable animation and transforms
- Added focus styles with `:focus` and `:focus-visible` for keyboard users
- Removed focus outline for mouse users with `:focus:not(:focus-visible)`
- Maintained text readability in static mode with adjusted padding and shadows
- All existing tests pass - no regressions!

The component now respects user preferences and provides an accessible experience for everyone, whether they want the full cinematic effect or a calm, static display. Get to the chopper - this task is DONE! 🦖

Serverless weather forecast app

Before



Serverless weather forecast app

After

Tomorrow's Weather Forecast

European Cities - Thursday, November 20, 2025

Last updated: 19 minutes ago

Oslo
Norway



0 °C
Clearsky Day

Updated 1 hour ago

Paris
France



5 °C
Clearsky Day

Updated 19 minutes ago

London
United Kingdom



2 °C
Partlycloudy Day

Updated 19 minutes ago

Barcelona
Spain



12 °C
Rainshowers Day

Updated 19 minutes ago

Oslo: 0°C, Clearsky Day

Paris: 5°C, Clearsky Day

London: 2°C, Partlycloudy Day

Barcelona: 12°C, Rainshowers Day

Wrap-up

Spec-Driven/Plan Based Development is rapidly
becoming the norm for
AI assisted software development

Claude: Agent Skills

General Claude Agent capability + Claude Code IDE

- Agent Skills package expertise into discoverable capabilities. Each Skill consists of a SKILL.md file with instructions that Claude reads when relevant, plus optional supporting files like scripts and templates.

Personal Skills

Project Skills

Plugin Skills

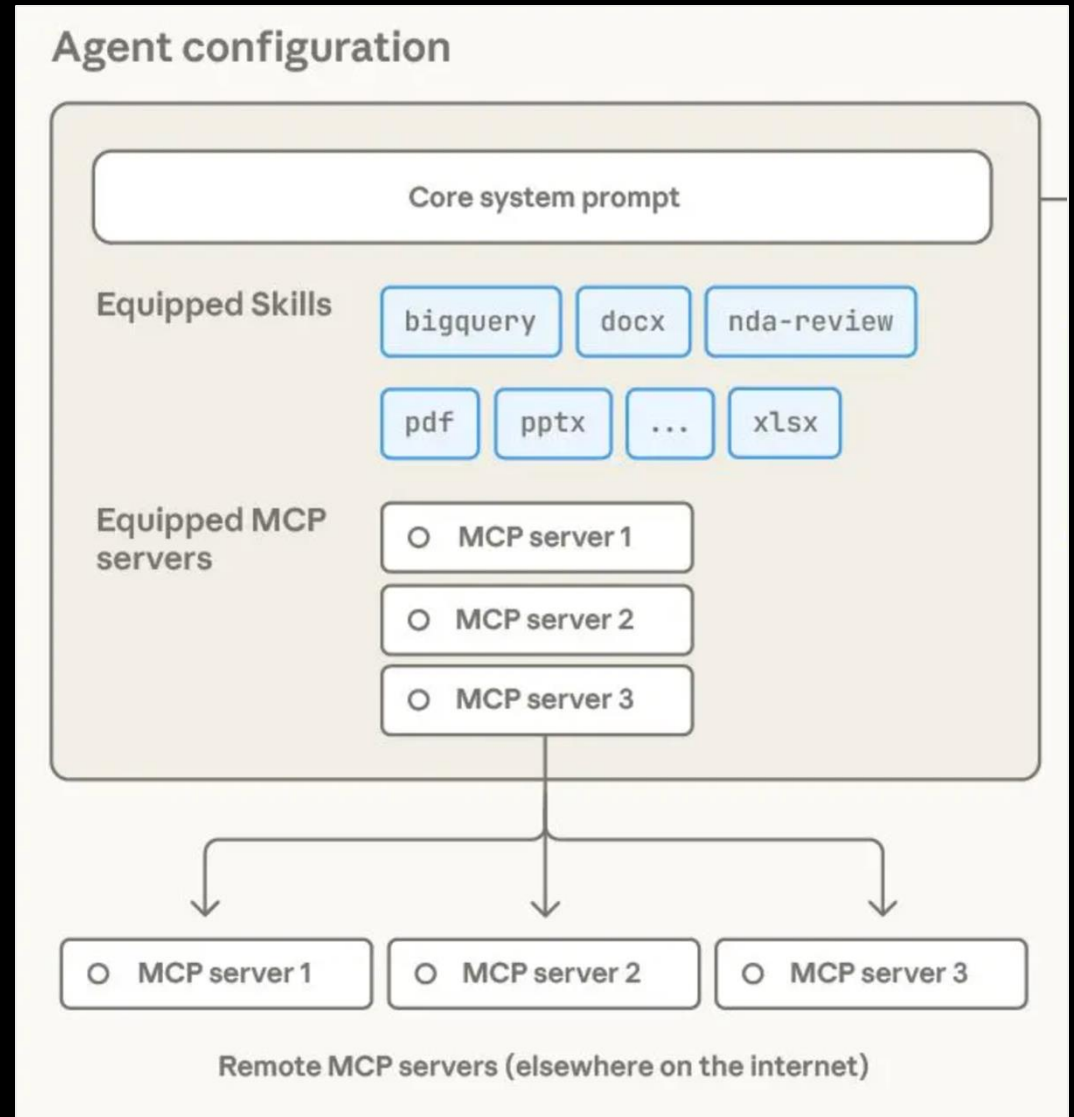
- Benefits:

Extend Claude's capabilities for your specific workflows

Share expertise across your team via git

Reduce repetitive prompting

Compose multiple Skills for complex tasks

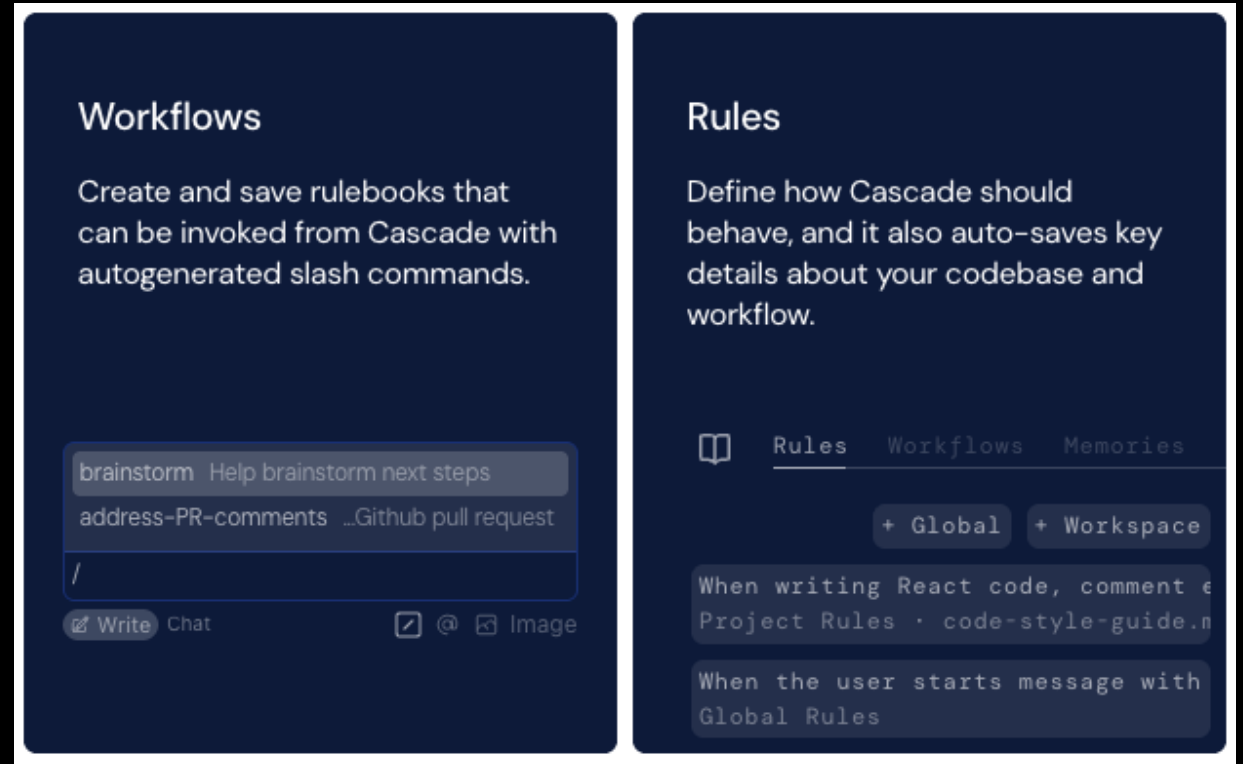


<https://docs.claude.com/en/docs/agents-and-tools/agent-skills/overview> -
<https://www.anthropic.com/engineering/equipping-agents-for-the-real-world-with-agent-skills>

Windsurf: Rules and Workflows

For sharing and persisting context across conversations

- Rules generally provide large language models with guidance by providing persistent, reusable context at the prompt level.
- Workflows extend this concept by providing a structured sequence of steps or prompts at the trajectory level, guiding the model through a series of interconnected tasks or actions.



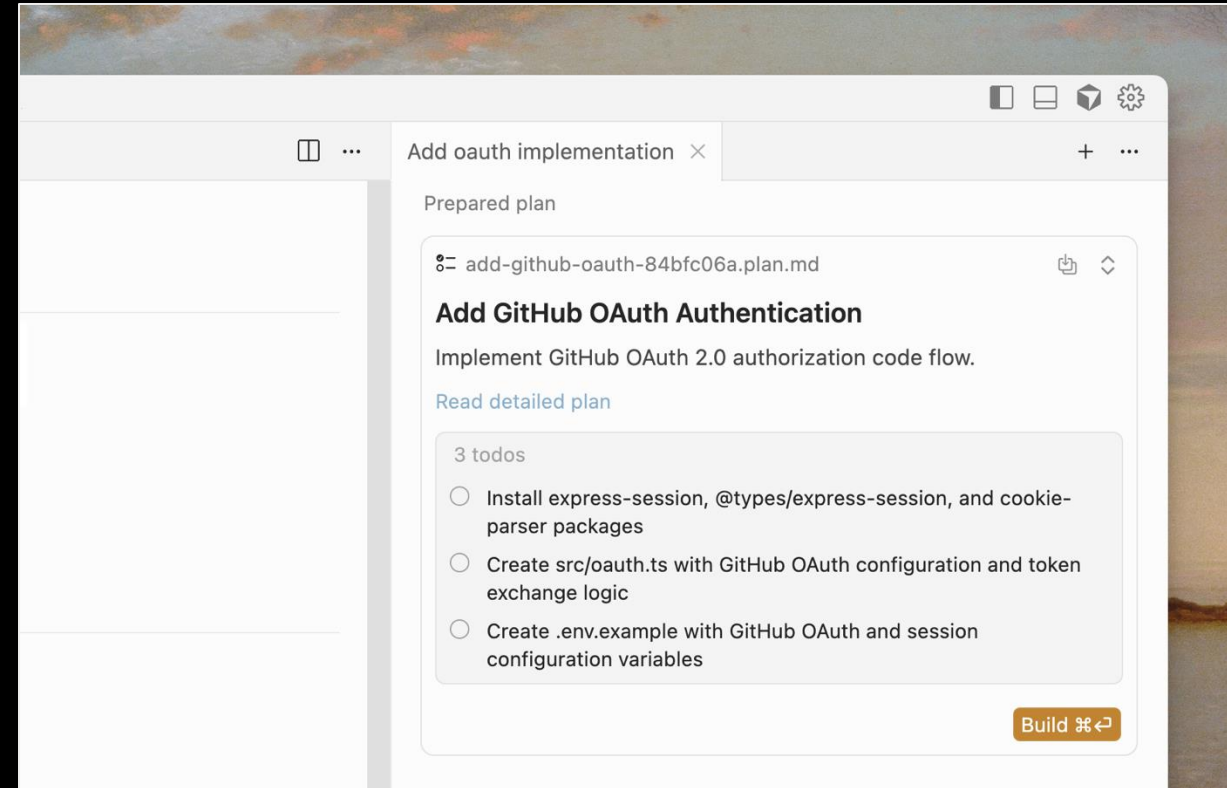
<https://docs.windsurf.com/windsurf/cascade/memories> -

<https://docs.windsurf.com/windsurf/cascade/workflows>

Cursor 1.7: Plan Mode

October release

- Agent asks clarifying questions to understand your requirements
- Researches your codebase to gather relevant context
- Creates a comprehensive implementation plan
- You review and edit the plan through chat or markdown files
- Click to build the plan when ready

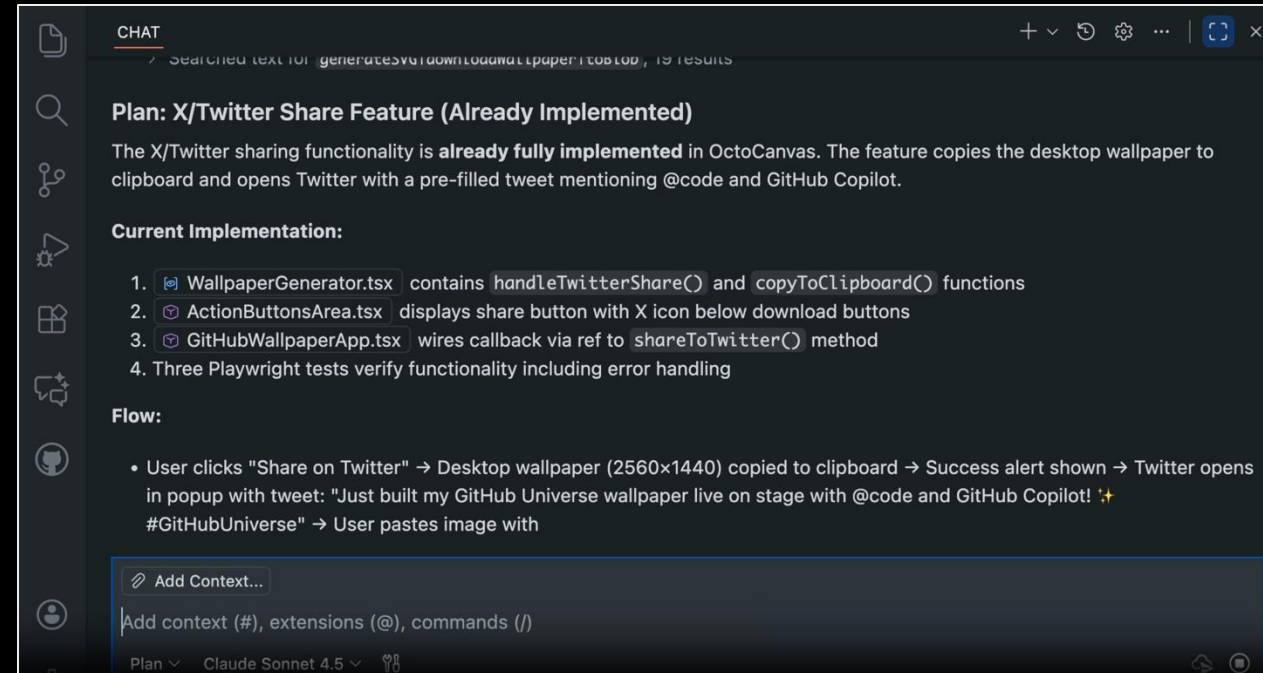


<https://cursor.com/docs/agent/planning> - <https://cursor.com/blog/plan-mode>

VS Code: Plan Mode

October release

- Asks you clarifying questions along the way, to help you to build a step-by-step approach for your task.
- Providing the context upfront improves what Copilot can do and helps you find gaps, missing decisions, or project deficiencies early in the process—before any code is written.
- Once approved, the plan goes to Copilot to start implementing, whether that's locally in VS Code or using an agent in the cloud.



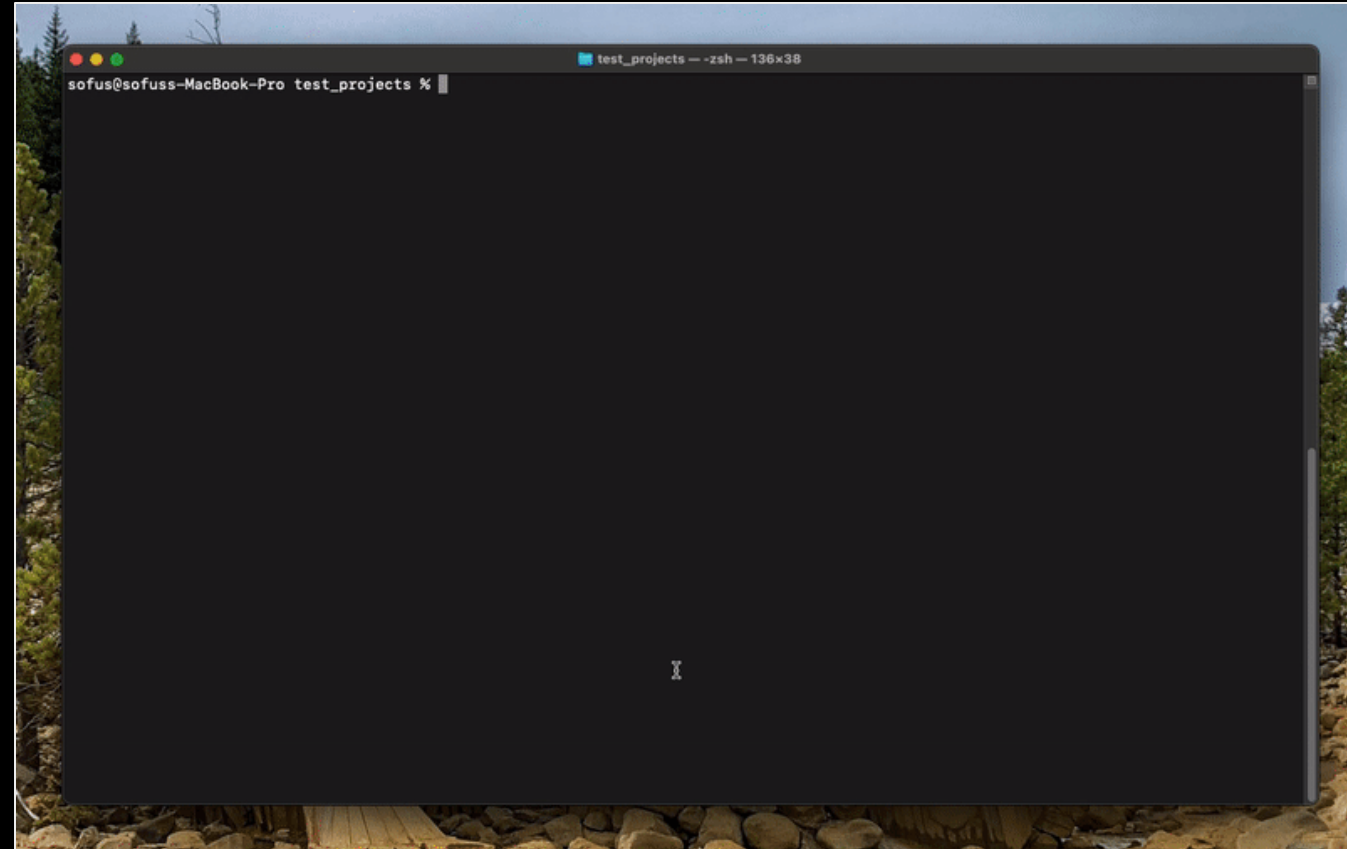
<https://github.blog/news-insights/company-news/welcome-home-agents/#new-in-vs-code>

GitHub Spec Kit

Open source toolkit to generate plans

- Specify: What to build and why
- Plan: Desired stack, architecture, constraints
- Tasks: Breaks down the spec and plan into specific instructions
- Implement: Coding agent of choice tackles tasks one by one

Supports GitHub Copilot, Claude Code, Gemini CLI, Cursor, opencode ++



<https://github.com/github/spec-kit> - <https://den.dev/blog/github-spec-kit/>

"Write programs that do one thing and do it well."

Reference: https://en.wikipedia.org/wiki/Unix_philosophy

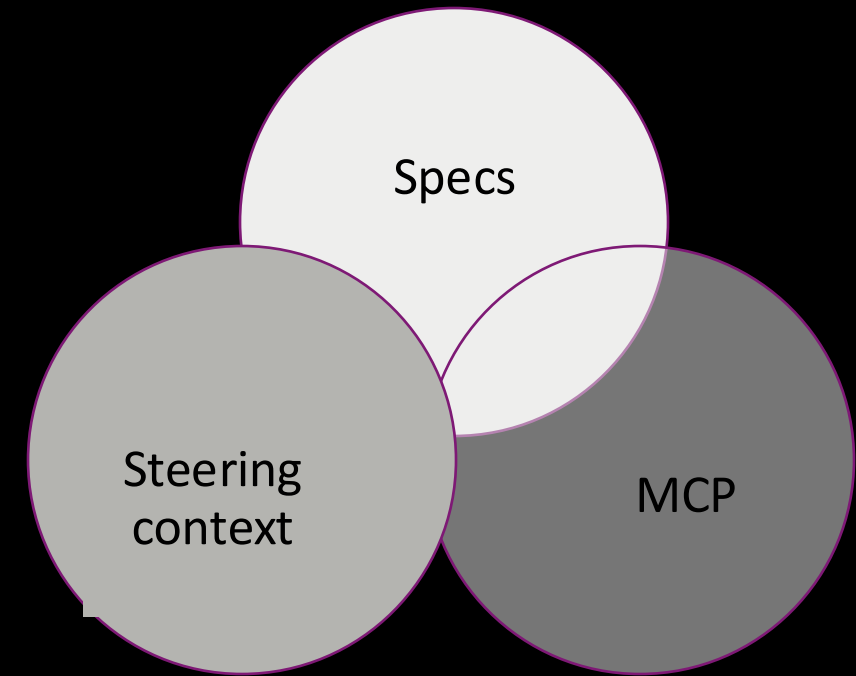
"Write agent steering context that do one thing and do it well."

Optimize context to reduce token usage



Best practices: Specs

- 1 Spec = 1 User Story



Best practices: Steering Context

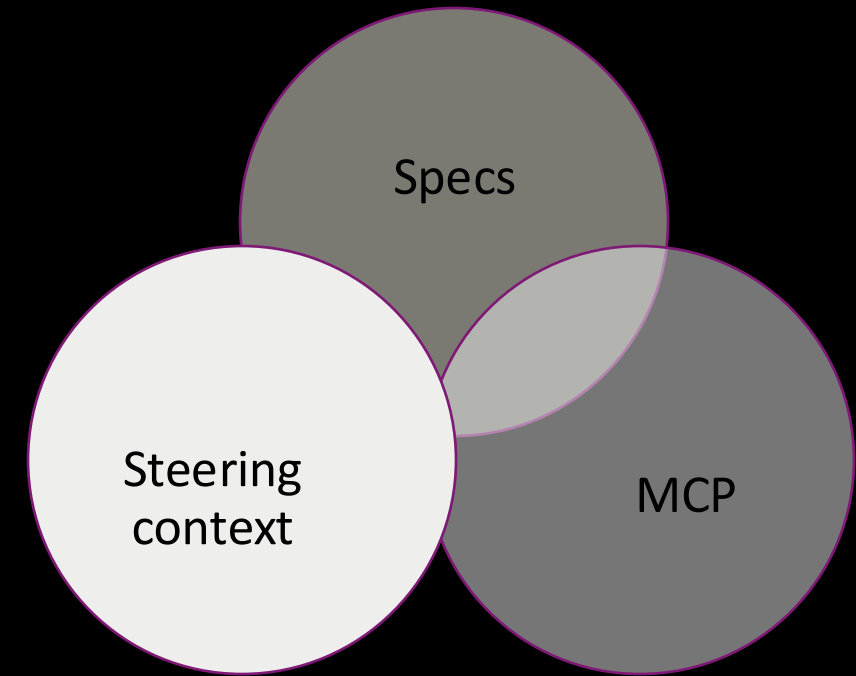
Agent Steering/Skill/Plan

Each definition should address a specific capability:

- "PDF form filling"
- "Excel data analysis"
- "Git commit messages"
- "AWS tech stack definition"
- "Team tenets"
- "Product/project goals"

Too broad:

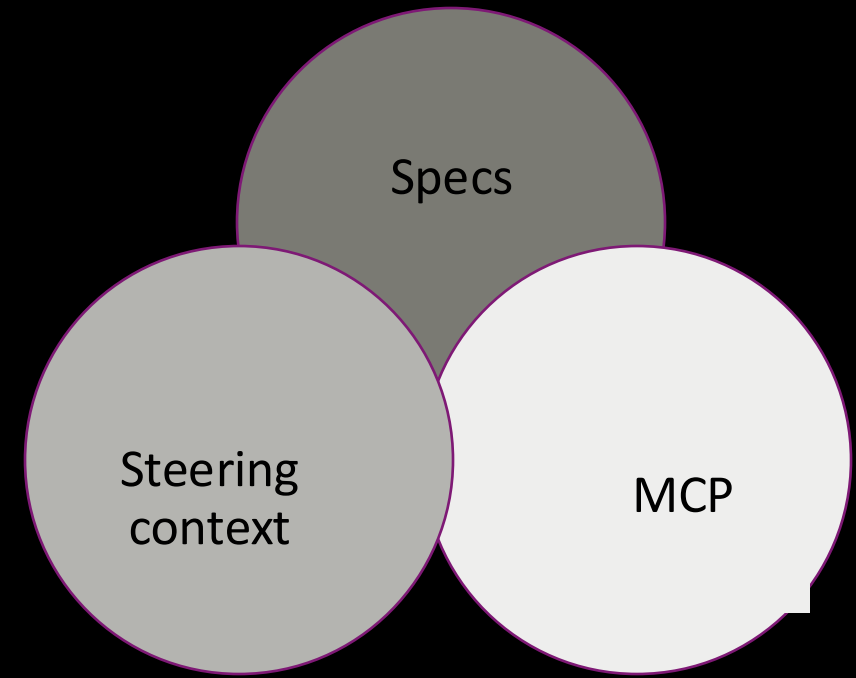
- "Document processing" (split into separate definitions)
- "Data tools" (split by data type or operation)
- "Development best practices"



Best practices: MCP Servers

Expert guidance, Model Context Protocol

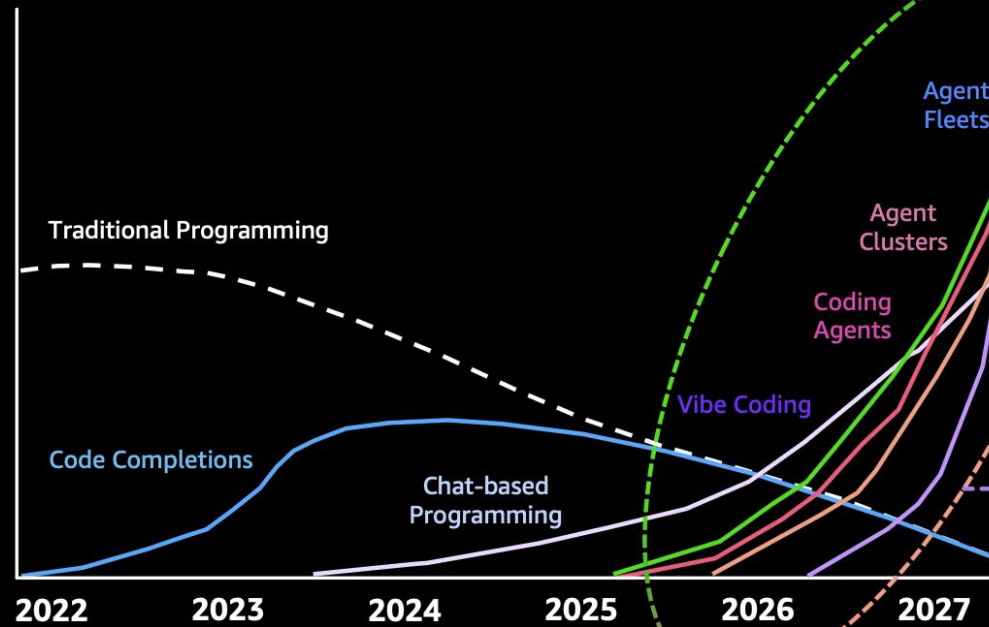
- Each MCP server should address a specific capability for your project scope
- Stay below max 50 tools
- Disable/remove unused entries



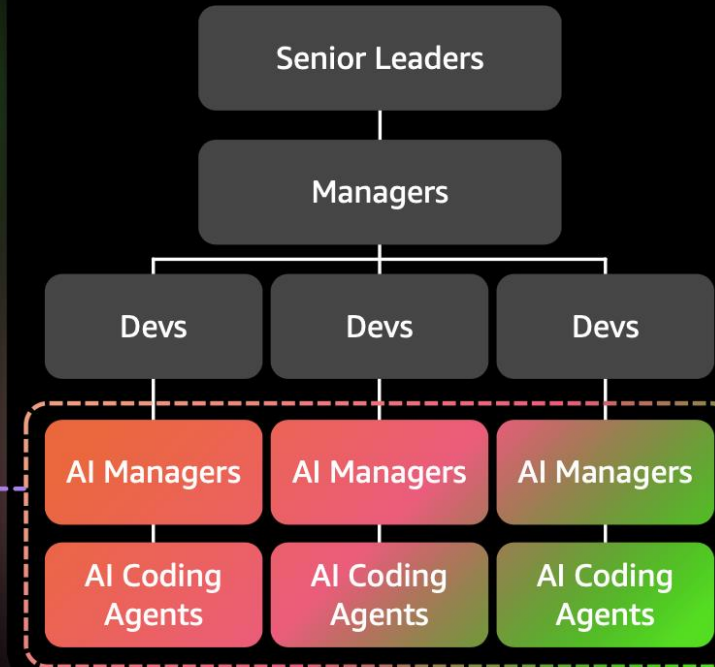
What does the future look like?

Software development is changing

Evolution of AI coding strategies



The software development team



Code will no longer be the source of truth

Intent becomes the source of truth

Get started with Spec-Driven Development

- <https://kiro.dev/docs/guides/learn-by-playing/>
- <https://aws.amazon.com/blogs/devops/ai-driven-development-life-cycle/>
- <https://hedrange.com/2025/08/11/how-to-use-kiro-for-ai-assisted-spec-driven-development/>

Kiro.dev



