Claude Code for data infrastructure

The Data Infrastructure team organizes all business data for teams across the company. They use Claude Code for automating routine data engineering tasks, troubleshooting complex infrastructure issues, and creating documented workflows for technical and non-technical team members to access and manipulate data independently.

Main Claude Code use cases

Kubernetes debugging with screenshots
When Kubernetes clusters went down and weren't scheduling new pods, the team used Claude Code to diagnose the issue. They fed screenshots of dashboards into Claude Code, which guided them through Google Cloud's UI menu by menu until they found a warning indicating pod IP address exhaustion. Claude Code then provided the exact commands to create a new IP pool and add it to the cluster, bypassing the need to involve networking specialists.

Plain text workflows for finance team
The team showed finance team members how to write plain text files describing their data workflows, then load them into Claude Code to get fully automated execution. Employees with no coding experience could describe steps like "query this dashboard, get information, run these queries, produce Excel output," and Claude Code would execute the entire workflow, including asking for required inputs like dates.

Codebase navigation for new hires
When new data scientists join the team, they're directed to use Claude Code to navigate their massive codebase. Claude Code reads their Claude.md files (documentation), identifies relevant files for specific tasks, explains data pipeline dependencies, and helps newcomers understand which upstream sources feed into dashboards. This replaces traditional data catalogs and discoverability tools.

End-of-session documentation updates
The team asks Claude Code to summarize completed work sessions and suggest improvements at the end of each task. This creates a continuous improvement loop where Claude Code helps refine the Claude.md documentation and workflow instructions based on actual usage, making subsequent iterations more effective.

Parallel task management across multiple instances
When working on long-running data tasks, they open multiple instances of Claude Code in different repositories for different projects. Each instance maintains full context, so when they switch back after hours or days, Claude Code remembers exactly what they were doing and where they left off, enabling true parallel workflow management without context loss.

Claude Code for data infrastructure

Team impact

Resolved infrastructure problems without specialized expertise
Resolved Kubernetes cluster issues that would normally require pulling in systems or networking team members, using Claude Code to diagnose problems and provide exact fixes.

Accelerated onboarding
New data analysts and team members can quickly understand complex systems and contribute meaningfully without extensive guidance.

Enhanced support workflow
Can process much larger data volumes and identify anomalies (like monitoring 200 dashboards) that would be impossible for humans to review manually.

Enabled cross-team self-service
Finance teams with no coding experience can now execute complex data workflows independently.

Top tips from the Data Infrastructure team

Write detailed Claude.md files
The better you document your workflows, tools, and expectations in Claude.md files, the better Claude Code performs. This made Claude Code excel at routine tasks like setting up new data pipelines when you have existing patterns.

Use MCP servers instead of CLI for sensitive data
They recommend using MCP servers rather than the BigQuery CLI to maintain better security control over what Claude Code can access, especially for handling sensitive data that requires logging or has potential privacy concerns.

Share team usage sessions
The team held sessions where members demonstrated their Claude Code workflows to each other. This helped spread best practices and showed different ways to use the tool they might not have discovered on their own.

Claude Code for product development

The Claude Code team uses their own product to build updates to Claude Code, expanding the product's enterprise capabilities and agentic loop functionalities.

Main Claude Code use cases

Fast prototyping with auto-accept mode
Engineers use Claude Code for rapid prototyping by enabling "auto-accept mode" (shift+tab) and setting up autonomous loops where Claude writes code, runs tests, and iterates continuously. They give Claude abstract problems they're unfamiliar with, let it work autonomously, then review the 80% complete solution before taking over for final refinements. Teams emphasize starting from a clean git state and committing checkpoints regularly so they can easily revert any incorrect changes if Claude goes off track.

Synchronous coding for core features
For more critical features touching the application's business logic, the team works synchronously with Claude Code, giving detailed prompts with specific implementation instructions. They monitor the process in real-time to ensure code quality, style guide compliance, and proper architecture while letting Claude handle the repetitive coding work.

Building Vim mode
One of their most successful async projects was implementing Vim key bindings for Claude Code. They asked Claude to build the entire feature (despite it not being a priority), and roughly 70% of the final implementation came from Claude's autonomous work, requiring only a few iterations to complete.

Test generation and bug fixes
They use Claude Code to write comprehensive tests after implementing features and handle simple bug fixes identified in pull request reviews. They also leverage GitHub Actions integration to have Claude automatically address Pull Request comments like formatting issues or function renaming.

Codebase exploration
When working with unfamiliar codebases (like the monorepo or API side), the team uses Claude Code to quickly understand how systems work. Instead of waiting for Slack responses, they ask Claude directly for explanations and code references, saving significant time in context switching.

Claude Code for product development

Team impact

Faster feature implementation
Successfully implemented complex features like Vim mode with 70% of code written autonomously by Claude.

Improved development velocity
Can rapidly prototype features and iterate on ideas without getting bogged down in implementation details.

Enhanced code quality through automated testing
Claude generates comprehensive tests and handles routine bug fixes, maintaining high standards while reducing manual effort.

Better codebase exploration
Team members can quickly understand unfamiliar parts of the monorepo without waiting for colleague responses.

Top tips from the Claude Code team

Create self-sufficient loops
Set up Claude to verify its own work by running builds, tests, and lints automatically. This allows Claude to work longer autonomously and catch its own mistakes, especially effective when you ask Claude to generate tests before writing code.

Develop task classification intuition
Learn to distinguish between tasks that work well asynchronously (peripheral features, prototyping) versus those needing synchronous supervision (core business logic, critical fixes). Abstract tasks on the product's edges can be handled with "auto-accept mode," while core functionality requires closer oversight.

Form clear, detailed prompts
When components have similar names or functions, be extremely specific in your requests. The better and more detailed your prompt, the more you can trust Claude to work independently without unexpected changes to the wrong parts of the codebase.