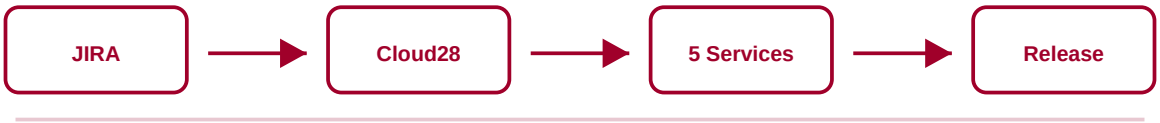
CASE STUDY

From 12-Engineer Microservice Operations to a 2-Core-Engineer Cloud28 Model

How Cloud28 turns a large engineering organization and 90+ microservices into a smaller, planned, test-driven, weekly delivery operating model.



CLIENT	
WORKFLOW	Engineering delivery, microservice operations, Jira planning, testing, documentation, and release management
SCALE	12-person engineering team, 90+ microservices, two-week agile sprint cycle, one release every two weeks
ENGAGEMENT	Cloud28 agentic engineering platform consolidates services, manages delivery flow, enforces planning, documentation, testing, and release discipline
DEPLOYED	Operating model shift from two-week sprint cycles to weekly Kanban delivery
OUTCOME	

01 - OVERVIEW

Executive Summary

The operated with a traditional engineering structure: a 12-person team responsible for a broad estate of more than 90 microservices, Jira-managed work, two-week agile sprint cycles, partial documentation, and inconsistent test coverage. The operating model could ship software, but it carried the overhead of a large team, many service boundaries, and release coordination across too many moving parts.

Cloud28 changes the economics and the control model. Instead of using headcount as the main way to manage complexity, Cloud28 acts as a full-stack agentic engineering layer that plans features before code is written, manages Jira delivery flow, supports documentation and test generation, and helps consolidate the architecture into 5 core services. The result is a smaller engineering core with stronger operational discipline.

The staffing shift is the clearest business signal. The operating model moves from 12 engineers to 2 core engineers supported by 5

The coverage model also changes. A traditional employee team operates mainly during the 9:00am-6:00pm workday, five days a

operate 24/7, so planning, documentation, test preparation, service checks, and release-support work continue without waiting for the next business day.

The delivery model moves from a two-week sprint cycle to weekly Kanban, from one release every two weeks to 2-3 releases per week, and from partial planning to full feature planning before implementation begins. Cloud28 makes the work more continuous, more testable, and easier to govern because planning, documentation, testing, and release readiness become part of the platform workflow.

12 -> 2 core engineering team reduction	**%**	**90+ -> 5** microservices consolidated into core services

02 - CURRENT OPERATION

The old model scaled complexity by adding people and services.

Before Cloud28, the Software company with a distributed team distributed R&D relied on a conventional engineering operating model. Jira tracked work, teams planned around two-week agile sprint cycles, and engineers managed a wide microservice estate with significant documentation and test gaps. This created a delivery process where much of the team capacity was spent coordinating, stabilizing, investigating, and preparing releases rather than steadily shipping planned features.

The 90+ microservice footprint created a large surface area for ownership, deployment, dependency management, and release coordination. Even small changes could require cross-service understanding, repeated regression checks, and additional communication. The architecture made delivery slower and made quality harder to guarantee because documentation and test coverage were incomplete.

The sprint model also introduced batch delay. Work was grouped into two-week cycles, and releases happened about once every two weeks. That cadence limited feedback speed and made every release more consequential. When documentation coverage sat around 40% and test coverage around 30%, every release carried extra uncertainty.

Step	Manual operating pattern today	Operational friction
1	Plan feature work in Jira for two-week sprint cycles.	Batch planning delays feedback and pushes decision-making into sprint ceremonies.
2	Coordinate changes across 90+ microservices.	Service sprawl increases dependency management, ownership overhead, and release risk.
3	Develop features with partial documentation and partial tests.	40% documentation and 30% test coverage leave implementation and release gaps.
4	Run partial testing during implementation and fuller testing near release.	Quality work arrives late, increasing rework and release pressure.
5	Release roughly once every two weeks.	Slower release cadence delays customer value and increases batch size.

12 engineers large team needed to operate complexity	**90+ services** wide microservice estate to coordinate	**1 / 2 weeks** baseline release cadence

03 - SOLUTION

Cloud28 becomes the agentic engineering operating layer.

Cloud28 does not simply replace tools. It changes the way engineering work is organized. Feature planning happens before code is written, engineering work continues in Jira but moves from two-week sprint batches into a weekly Kanban flow, and the platform manages the discipline around documentation, tests, service consolidation, and release readiness.

The architecture is simplified from 90+ microservices into 5 core services. This is not only a code refactor; it reduces ownership overhead, service-to-service coordination, deployment complexity, and release risk. A smaller service footprint makes testing, documentation, monitoring, and governance more practical for a smaller core team.

The quality model becomes more deliberate. Documentation moves from 40% coverage toward full codebase documentation. Test coverage moves from 30% toward full release coverage, with partial testing on new code implementation and full test coverage required on releases. This gives leadership a more reliable view of release readiness.

Cloud28 Platform acts as the full-stack service layer for this engineering workflow, leveraging agentic engineering to plan features, maintain code, create documentation, generate tests, manage Jira flow, consolidate services, and support release discipline. The

Capability	What it does	What changes for the startup
Agentic planning	Creates full planning across features before implementation begins.	Less ambiguity before code is written and fewer mid-stream surprises.
Jira-managed Kanban	Keeps Jira as the work system while shifting from two-week sprint batches to weekly Kanban.	Smaller batches, faster feedback, and more continuous delivery flow.
Service consolidation	Reduces 90+ microservices into 5 core services.	Lower coordination overhead and simpler release surface.
Documentation automation	Moves code documentation from 40% toward full coverage.	Knowledge becomes durable and easier for a smaller team to operate.
Test discipline	Moves test coverage from 30% toward full release coverage.	Release confidence improves while implementation testing starts earlier.
Release acceleration	Moves from one release every two weeks to 2-3 releases per week.	Customer value ships faster with smaller release batches.

04 - REALIZATION PLAN

The transformation moves from operating model to measurable engineering system.

The Cloud28 transition should be treated as an operating-model migration, not only as a tooling change. The team first stabilizes Jira work intake, defines planning rules, maps the 90+ microservice estate, and identifies which domains belong in the 5 core services. Once the service model is clear, Cloud28 can enforce planning, documentation, tests, and release readiness as part of the delivery workflow.

The team model shifts gradually into a 2-core-engineer operating structure. Those engineers focus on judgment-heavy engineering: architecture decisions, high-risk exceptions, code review, release approval, and product tradeoffs. Cloud28 absorbs the repeatable work around service consolidation, documentation creation, test generation, planning completeness, and delivery hygiene.

Phase	Focus	Customer-readable deliverable	Acceptance signal
1	Baseline and operating map	Jira flow, service inventory, documentation/test baseline, release baseline	Current delivery system and bottlenecks are documented.
2	Service consolidation plan	90+ microservices mapped into 5 core service domains	Target service ownership and migration plan approved.
3	Agentic planning and Jira Kanban	Weekly Kanban process and feature-planning rules	Features are planned before implementation and tracked continuously.
4	Documentation and tests	Full-code documentation path and release test coverage model	Documentation and release tests become required delivery gates.
5	Release acceleration	2-3 releases per week operating cadence	Smaller, more frequent releases ship with release readiness evidence.

40% -> Full documentation coverage target	**30% -> Full** release test coverage target	**4x-6x** release cadence improvement

05 - EFFECTIVENESS OVER TIME

The baseline starts with the original operating model: 12 engineers, 90+ microservices, 40% documentation coverage, 30% test coverage, two-week sprint cycles, and one release every two weeks. Those numbers become the control group for measuring whether Cloud28 is improving the engineering system, not merely changing its process language.

During the transition, the startup should track the service consolidation path, Jira flow stability, planning completeness before code, documentation coverage, implementation test coverage, release test coverage, and release cadence. The target is not to move fast by hiding risk; it is to make the risks visible earlier and reduce the number of release-blocking surprises.

After the operating model is stable, leadership should track whether the 2-core-engineer model sustains the release cadence and quality bar. The most important signal is whether the smaller team can continue shipping 2-3 releases per week while documentation, tests, planning, and release readiness remain complete.

Metric	Baseline	Target signal after Cloud28
Engineering staffing	12-person engineering team	2 core engineers with 5 Cloud28 agentic seats
Coverage	e days per week; overtime at 1.5x	24/7 agentic operation for planning, documentation, tests, checks, and release-support work
Architecture	90+ microservices	5 core services with lower coordination overhead
Documentation	40% code documentation coverage	Full documentation across the codebase
Testing	30% test coverage	Partial testing on new code and full coverage on releases
Release cadence	1 release every 2 weeks	2-3 releases per week
Planning discipline	Planning often continues during implementation	Full feature planning before any code is written

	Traditional model	Cloud28 model	Business impact
		Retained core engineers plus Cloud28 agentic-seat capacity	
Coverage window	9:00am-6:00pm, five days per week	24/7 agentic operating support	Work no longer waits for the next business day
		Continuous agent support is included in the platform operating model	Urgent planning, checks, documentation, and test work can continue without overtime escalation

Engineering capacity returned to the business

The staffing reduction from 12 engineers to 2 core engineers represents 10 engineer-equivalents of operating capacity removed from the traditional employee model. With 5 Cloud28 agentic seats, the startup keeps the delivery support needed for planning,

The practical value is that the Software company with a distributed team distributed R&D no longer needs to spend large-team effort coordinating service sprawl, catching up documentation, manually assembling tests late in the cycle, and batching releases into two-week windows. Cloud28 shifts that work into an agentic engineering operating layer so the remaining core engineers can focus on architecture, review, exceptions, and product-critical implementation.

delivery model.

06 - OPERATING CONTROLS

A smaller team works only if quality gates are visible.

The Cloud28 model must keep engineering controls visible. Service consolidation should be tracked as an explicit migration plan, not an informal cleanup effort. Documentation and testing should be treated as delivery gates, not optional after-work. Jira should remain the operational record, but weekly Kanban should make work-in-progress, blockers, and readiness visible at a higher frequency than the old sprint cadence.

Human engineers remain responsible for judgment-heavy work: architecture decisions, risky changes, code review, production exceptions, and release approval. Cloud28 handles the repeatable work around planning completeness, documentation, test creation, release evidence, and service-management discipline.

The operating control is simple: a 2-core-engineer model is only acceptable if release frequency improves while documentation, tests, planning, and service boundaries become stronger. If those controls weaken, the team is only downsizing. If those controls improve, the startup has moved to a genuinely agentic engineering operating model.