I'll answer as a world-famous developer-tools product growth strategist, PhD in Industrial Organization Economics, winner of the Dutch Marketing Award.

TL;DR: Grow CLion by (1) turning the new free non-commercial license into the biggest C/C++ adoption funnel and community loop, (2) winning two premium "willingness-to-pay" segments—**embedded/RTOS** and **large-scale C++ monorepos**—using CLion's differentiated debugging + code intelligence, (3) leading with **secure, controllable AI** (BYOK/local models/policy), and (4) tightening **enterprise GTM** via land-and-expand in existing JetBrains accounts—measured by activation speed, paid-seat growth, and net revenue retention.

---

1) What's true right now about CLion, the market, competition, and the audience

Product & positioning (what CLion can credibly win on)

- **CLion is now free for non-commercial use** (learning, hobby, open source without commercial benefit, and even content creation), while commercial use still requires a paid subscription. (The JetBrains Blog)
- The **non-commercial license is full-featured** versus paid, with a key exception called out by JetBrains: **Code With Me Community** (for most covered IDEs), and support is steered to community forums + issue tracker. (sales.jetbrains.com)
- CLion's recent differentiation arc is performance + correctness for C++:
    - **CLion Nova became the default language engine in 2025.3**, replacing the legacy "Classic" engine as the primary future investment area. (JetBrains)
    - Nova is based on the JetBrains engine originally built for **ReSharper C++ / Rider**, with Classic no longer actively developed for language improvements. (JetBrains)
    - JetBrains positions 2025.3 as a major step with **Constexpr Debugger** and **Debug Adapter Protocol support** (and more). (The JetBrains Blog)
    - JetBrains claims up to **4× faster** highlighting/completion/refactoring vs the legacy engine on 2025.3 messaging. (JetBrains)

Project / workflow coverage (important for adoption)

- CLion supports multiple **project formats**: **CMake, Makefile, JSON compilation database (compile_commands.json), Gradle, Meson, and Zephyr West**. (JetBrains)
- Remote development options include JetBrains Gateway "thin client", WSL2, Docker, and remote debug options (GDB/LLDB, gdbserver/lldb-server). (JetBrains)
- Embedded is a major wedge: CLion supports on-chip debugging through common GDB servers (OpenOCD, ST-Link, J-Link, QEMU, etc.), RTOS task views

(FreeRTOS/Zephyr/Azure RTOS), peripherals view via SVD, and PlatformIO integrations. (JetBrains)

Market reality (demand is huge; "default tool" is not you)

- C/C++ remains a large developer population. In the 2025 Stack Overflow survey: **C++ ~23.5%** and **C ~22%** of all respondents report extensive use in the past year. (Stack Overflow)
- By language-popularity proxies, C and C++ remain top-tier (e.g., TIOBE's January 2026 index shows C and C++ still near the very top). (TIOBE)
- IDE/editor usage is extremely concentrated: the same 2025 Stack Overflow survey shows **VS Code ~75.9%** and **Visual Studio ~29%** (all respondents). JetBrains tools appear strongly (e.g., IntelliJ IDEA ~27.1%), but CLion is not a top-listed default environment there. (Stack Overflow)

Competitive set (who you're really fighting)

You're fighting three overlapping competitors, each requiring a different response:

1. **Free, extensible editors**: VS Code + C++ extensions (dominant) (Stack Overflow)
2. **Full IDEs**: Visual Studio, Xcode, Eclipse/Qt Creator, vendor embedded IDEs
3. **AI-first IDEs/editors**: Cursor (~17.9%), Claude Code (~9.7%), Windsurf (~4.9%) already show meaningful adoption in 2025 survey data. (Stack Overflow)

Audience segmentation (who pays vs who fuels growth)

Use this segmentation because it maps cleanly to both *willingness-to-pay* and *distribution*:

A. Enterprise C++ platform teams (paid, high ACV)

- Monorepos, Bazel/CMake, multi-OS, remote Linux toolchains, heavy debugging and refactoring needs.
- Care about performance, stability, vendor support, security posture, admin controls.

B. Embedded/firmware teams (paid, high conversion if you "just work")

- RTOS, OpenOCD/J-Link/ST-Link, Zephyr West, cross compilers, hardware-in-loop debug.
- Care about setup friction, debug reliability, reproducibility, and vendor integrations.

C. Indie/freelance commercial devs (paid, price-sensitive, high advocacy)

- Small teams, plugin/game tooling, robotics consulting, open-source-with-commercial side gigs.

D. Learners, hobbyists, open source maintainers, content creators (free → future paid + distribution)

- Now addressable at scale via non-commercial licensing, including monetized content creation. (sales.jetbrains.com)

---

2) The growth strategy: 5 pillars that compound

Pillar 1 — Make CLion the "best-in-class" for modern C++ correctness + debugging at scale

Your durable edge against VS Code isn't "can edit C++"; it's **deep code intelligence + refactoring + debugging that stays fast on huge codebases**.

What to do

1. **Own the "fast & precise C++" narrative** around Nova being default and Classic being legacy. (JetBrains)
2. Treat **Constexpr Debugger** as a flagship differentiator (C++ teams feel pain here). (The JetBrains Blog)
3. Build "CLion for monorepos" credibility:
    - curated performance benchmarks (indexing, navigation, refactoring) on representative OSS monorepos
    - hardening work: freezing/hanging reduction, memory footprint, "project open time" as a first-class KPI

How it grows revenue

- Better enterprise evaluations → higher win rate vs "free editor + extensions".
- Lower churn driven by performance frustration (historically the #1 silent killer for IDE retention).

Key metrics

- Activation: *time to first successful build+debug* (median/p90 by segment)
- Product: *weekly freezes/hangs per active seat*, *indexing time p50/p90*
- Business: *enterprise trial-to-paid conversion*, *net revenue retention (NRR)*

Pillar 2 — Win embedded by making setup + on-chip debugging radically smoother than alternatives

You already have strong embedded claims: OpenOCD templates, STM32CubeMX auto configs, RTOS views, peripherals. (JetBrains)
Turn that into a category wedge.

What to do

1. Create **hardware-specific "starter experiences"** (Zephyr West, STM32CubeMX, ESP-IDF, Nordic nRF Connect SDK, PlatformIO) with:
    - one-click new project templates
    - auto-detected toolchains
    - prebuilt Run/Debug configs (OpenOCD/J-Link/ST-Link)
2. Double down on **Zephyr West as a growth channel** because it's listed as a supported project format—make it a hero workflow. (JetBrains)
3. Partnership play:
    - Co-market with embedded ecosystems where developers already congregate (Zephyr community, PlatformIO community, MCU vendor devrel)
    - Provide "official" CLion setup guides validated by vendors (this is how you beat vendor IDE inertia)

How it grows revenue

- Embedded teams often standardize tools; one champion win can become a whole department rollout.
- Embedded devs also churn less once workflow is stable.

Key metrics

- # of embedded activations/week (detected via use of embedded run configs, RTOS view usage, peripherals tab usage—careful with privacy constraints)
- Embedded trial→paid conversion
- Expansion: seats/account in embedded-heavy domains

Pillar 3 — Lead the "secure AI in the IDE" position (because trust is the new feature)

AI adoption is high, but fear is real:

- JetBrains reports **85%** regularly use AI tools and **privacy/security** is among top concerns. (The JetBrains Blog)
- The broader market is seeing security research alleging systemic vulnerabilities across AI-assisted dev tools ("IDEsaster"). (Tom's Hardware)

CLion can win by being the IDE that gives teams **control**.

What to do

1. Package AI in a way enterprises can say "yes" to:
    - Lean into **BYOK** and "use your own provider/account" as a trust lever. (The JetBrains Blog)
    - Promote local model support and governance controls (policy, safe defaults, clear data flows). (JetBrains)
2. Add an explicit **"Secure AI Mode"** positioning:
    - no silent terminal execution
    - explicit approvals for filesystem changes
    - workspace-level allowlists/denylists
3. Make C++-specific AI use cases productized:
    - "Explain template metaprogramming / SFINAE"
    - "Generate tests + fuzz harness"
    - "Find UB risk patterns" (as assistant to static analysis—not replacing it)

How it grows revenue

- Converts regulated industries that otherwise ban AI tools.
- Higher retention when AI is available *safely* inside the IDE rather than forcing devs to copy/paste into external tools.

Key metrics

- AI feature adoption among paid orgs (opt-in rate)
- AI-enabled retention uplift vs control cohort
- Enterprise security questionnaires passed / time-to-procurement

---

Pillar 4 — Turn free non-commercial into compounding growth loops (not just downloads)

Non-commercial licensing is your biggest structural change in years:

- It's **individual-only**, **full-featured**, **auto-renews**, and **can't opt out of anonymized stats** under that agreement. (sales.jetbrains.com)
  This is both a growth engine *and* a trust/perception risk if handled clumsily.

Design the funnel intentionally

1. **Activation-first onboarding for non-commercial**
    - "Choose your path": learning C++, contributing to OSS, embedded hobby project, etc.
    - built-in sample projects and guided setup for toolchains/build systems
2. **Graduation moments (conversion triggers)**

    - The right conversion isn't a nag screen; it's a "you're leveling up" moment:
        - when user toggles "work mode" (mon–fri daytime usage patterns aren't enough; avoid creepy inference)
        - when they explicitly indicate they're being paid to code
    - Provide a clean "Upgrade because your situation changed" workflow, matching JetBrains' own guidance that if you expect commercial benefit you should use a commercial license. (sales.jetbrains.com)
3. **Creator flywheel**
    - Non-commercial explicitly allows monetized content creation (ads/courses). (sales.jetbrains.com)
    - Build a "CLion Creator Kit":
        - OBS overlays, keymap presets, demo repositories, debugging showcases (constexpr demo!)
        - affiliate/referral program (if you have one—if not, start one) tied to commercial upgrades and team trials

How it grows revenue

- Learners → first job → "I want the tool I learned on" enterprise pull.
- OSS maintainers recommending CLion to contributors.
- Creator content becomes always-on acquisition.

Key metrics

- Non-commercial: D7/D30 activation and retention
- Free→paid conversion *later* (cohorts tracked over 12–24 months)
- Creator loop: content volume, referral installs, trial starts from creator links

---

Pillar 5 — Enterprise GTM: land-and-expand inside existing JetBrains footprint

This is the highest ROI path because you don't start from zero trust. Most enterprises that buy JetBrains already understand subscription licensing.

What to do

1. **Account-based cross-sell**: find accounts using IntelliJ/Rider/WebStorm etc. and target teams with native modules, embedded products, or performance-critical components.
2. Create an "Enterprise C++ Evaluation Kit":
    - 2-hour guided proof: import, index, run, debug, refactor, run tests
    - "Bring your build": CMake/Bazel/compile_commands + remote toolchain support
    - a checklist for security/IT (proxy, offline constraints, AI policy)
3. Procurement accelerators:

- ROI calculator framed around developer time (debugging/refactoring time saved)
- deployment playbooks for Windows/macOS/Linux fleets

How it grows revenue

- Higher enterprise win rate; larger initial deal sizes; faster expansion.
- Reduced churn due to "team standard" adoption.

Key metrics

- Sales: evaluation→close rate, sales cycle time, seats at day 0 vs day 180
- Revenue: NRR, GRR, expansion revenue per cohort

---

3) A concrete execution plan (what to ship/do, in what order)

Phase 1: Next 90 days — "Funnel + message + activation" sprint

Goal: increase trials, improve activation, reduce drop-offs, and build a conversion engine that doesn't feel predatory.

1. **Reposition the homepage + key landing pages around 3 jobs-to-be-done**
   - "Modern C++ at scale (Bazel/CMake/monorepo)"
   - "Embedded & RTOS debugging"
   - "Fast onboarding from VS Code"
     Use Nova-default + performance claims carefully and credibly. (JetBrains)
2. **Activation instrumentation**
   - Define the activation event: first successful *configure → build → run/debug*.
   - Track time-to-activation by segment and by project model (CMake vs compdb vs West etc.). (JetBrains)
3. **Non-commercial onboarding revamp**
   - Add a first-run "what are you doing?" wizard with presets:
     - CMake quickstart
     - compile_commands.json import path (JetBrains)
     - Zephyr West quickstart (JetBrains)
     - embedded GDB server wizard path (JetBrains)
4. **Enterprise trial "fast path"**
   - One page: "Evaluate CLion on your codebase in 2 hours"
   - Provide a scripted checklist + a short video

Success criteria (90 days)

- +X% trial starts from qualified pages
- –Y% early churn in trial (no activation)
- +Z% paid conversions from trial cohorts

Phase 2: 3–12 months — “Category wedges that print money”

Goal: win segments where CLion is a clear, paid-value choice.

1. **Embedded growth package**
    - Official templates: Zephyr West boards, STM32CubeMX projects, PlatformIO “golden path”
    - “Known good” debug packs (OpenOCD/J-Link/ST-Link), leveraging existing product hooks. (JetBrains)
2. **Monorepo & Bazel leadership**
    - Invest in Bazel UX and performance improvements aligned with the published roadmap themes (Starlark REPL, execlog parsing, transitions support, etc.). (The JetBrains Blog)
    - Produce 3–5 public case studies with measurable “time saved” benchmarks.
3. **Testing parity beyond CMake**
    - Roadmap mentions making unit test integration independent of CMake so Meson/compdb can get full test UX. Treat this as a growth lever, not just tech debt. (The JetBrains Blog)
4. **AI trust package**
    - Ship and market “Secure AI Mode”
    - Promote BYOK as control for cost + policy. (The JetBrains Blog)

Success criteria (12 months)

- Embedded: meaningful increase in paid adoption among embedded-heavy orgs
- Enterprise: improved win rate vs VS Code/Visual Studio in head-to-head evals
- Retention: measurable churn reduction correlated with performance/activation improvements

---

Phase 3: 12–24 months — “Build the compounding machine”

Goal: sustainable share gains despite free/AI competition.

1. **CLion learning ecosystem**
    - Deep integrations with learning content and structured “paths” (especially modern C++ and embedded)

    - A formal “CLion Certified” program for universities/bootcamps (credential = enterprise pull)
2. **Community & open-source loop**
    - A maintainer program: “CLion-friendly project setup” badges / recommended configs (CMake presets, compile_commands generation)
    - Sponsor tooling improvements upstream (CMake, clang tooling) that reduce IDE friction
3. **Agentic workflows (carefully)**
    - JetBrains is clearly moving toward agentic development concepts across products, and the market is shifting quickly. (IT Pro)
    - CLion should adopt agentic features only when they’re provably secure, controllable, and C++-aware (patch review workflows, test-first agents, refactoring agents gated by CI).

---

4) Packaging & monetization moves (without breaking trust)

Personal commercial (indie/freelance)

- Lean into “time saved” bundles: debugging + refactoring + remote toolchain convenience.
- Consider an “Indie commercial starter” offer (time-limited discount or annual promo) to capture “I just went pro” moments—without devaluing the product for teams.

B2B (orgs)

- Make CLion the default choice for:
    - cross-platform native modules inside otherwise managed-code orgs
    - embedded groups who currently tolerate vendor IDEs
- Bundle value narratives with JetBrains AI governance and BYOK for control. (The JetBrains Blog)

Non-commercial (free)

- Treat it as:
    - a training pipeline
    - a creator megaphone
    - a future enterprise pull motion
      …but stay explicit about telemetry constraints (since there’s no opt-out under that agreement) to avoid reputational damage. (sales.jetbrains.com)

---

5) The operating system: metrics, cadences, and experiments

North Star metric

Pick one that connects product value to revenue:

- **Weekly Active Paid Seats** (WA-paid) with a quality gate: activated seats only.

Weekly growth dashboard

- Activation rate (trial + non-commercial separately)
- Time-to-first-build/debug (p50/p90)
- Trial→paid conversion
- Paid churn + top churn reasons
- Expansion: seats/account trend

Experiment engine (examples that are low-risk and high learning)

- Onboarding A/B: “build config wizard” vs current
- Embedded landing page to guided template vs docs-first
- Enterprise eval kit: video-first vs checklist-first
- Upgrade journey: “graduation” message vs generic paywall

---

6) Risks to manage explicitly

1. **AI security + privacy backlash**
    - Market narratives and security research can swing sentiment quickly. (Tom's Hardware)
      Mitigation: default-safe AI, explicit controls, transparent comms.
2. **Non-commercial telemetry perception**
    - Since opt-out isn’t available under that agreement, be transparent and respectful in-product. (sales.jetbrains.com)
3. **Performance regressions**
    - Nova is now “the future”; treat performance as a brand promise with aggressive regression testing. (JetBrains)
4. **Being squeezed between free editors and AI-native tools**
    - Counter-position: CLion = “engineering-grade C++ environment” + “secure AI” + “embedded/monorepo excellence,” not “another editor”.

- Tom's Hardware
- IT Pro
- Business Insider
- wired.com
- IT Pro