Mainely Code

— AI coding with receipts

EARLY CUSTOMER BRIEF

STAGE 1 · OCI-ONLY LAUNCH

Buy seats. Submit work orders. Review verified code changes.

Mainely Code gives early software adopters a governed place to request bounded AI-assisted code changes, review diffs, inspect proof bundles, and approve work before merge.

OCI-only	No unlimited usage	Proof first	Pricing by request
Stage 1 infrastructure	Capped work orders	Diffs, checks, rollback notes	Founding-seat fit review

Public inquiry version: Pricing is handled by request during founding access. This brief is customer-facing and is not an investor offer.

For customer fit review: tahai.portal@gmail.com · mainelycode.com

WHAT EARLY ACCESS INCLUDES

Fast code help is not enough. The review path has to be clear.

Mainely Code is built for customers who want useful AI coding help without uncontrolled repo-wide edits, surprise usage burn, or vague “done” claims. Every job is scoped before execution and reviewed before merge.

Customer gets	What it means	Why it matters
Work-order intake	Submit bug fixes, small features, repo cleanup passes, and review requests.	Work is scoped before execution.
Bounded task packet	Owned files, blocked files, acceptance checks, and retry limits.	Prevents agent freelancing and repo-wide drift.
Verified diff	Changed files, unified diff, rationale, and validation status.	Customer can review before merge.
Proof bundle	Plan, checks, build/test output, failure class, and rollback/resume notes.	Receipts replace trust-me automation.
Approval-before-overage	Jobs stop before exceeding agreed budget, runtime, or retry caps.	Protects customer spend and keeps scope explicit.

Stage 1 delivery model: Early access is built around an OCI-only launch path: control plane, CPU workers, storage, queues, logs, proof bundles, and external model/API or BYOK usage where appropriate. Company-owned compute can be added later as a capacity and margin lever, not as a dependency for the first customer conversations.

EARLY ACCESS OFFERS

Seat access designed for scoped, reviewable work.

This public inquiry version keeps price values private. Founding-seat terms are quoted after repo fit, job size, support needs, and model/runtime approach are reviewed.

Offer	What it includes	Caps / controls	Best fit
Founder Seat	A lightweight entry seat for occasional verified PR attempts and small bug-fix or feature jobs.	Single-repo, low-concurrency access; small scoped tasks only.	Solo founders and small businesses.
Builder Seat	Recurring verified PR attempts, repo cleanup passes, and feature-request work orders.	Multiple repos, capped concurrent jobs, bounded retries, proof retention.	Active builders needing repeatable code help.
Team Seat	Shared review flow, approval gates, proof history, and team-oriented work-order handling.	Team users, repo caps, approval-before-merge, longer proof retention.	Small teams needing shared review and accountability.
Launch Pack	Repo onboarding plus a short list of scoped work orders completed with hands-on setup.	Statement of work or written scope before delivery; overage by approval only.	Best first sale for a customer bringing a repo and short list of jobs.

Why pricing is handled by request during founding access

Early access includes orchestration, repo handling, build/test execution, proof-bundle generation, customer support, and strict execution caps. Mainely Code sells verified coding outcomes, not unlimited token access.

Pricing principle	Customer-facing rule
No public unlimited plan	Every seat is bounded by verified PR attempts, active repos, users, runtime, retry caps, and approval-before-overage.
Quote before heavy work	Large repos, unclear requirements, or high-runtime jobs are scoped before execution.
BYOK where appropriate	Customers may be offered bring-your-own-key or dedicated/private model lanes as the product matures.
Proof before handoff	The customer receives the diff, proof bundle, failure/success report, and rollback notes before merge approval.

CUSTOMER COMMITMENTS

Clear rules before the first work order.

No silent fallback Model/provider changes require visibility and approval for critical jobs.	**No repo-wide edits** Every job has owned files and blocked files.	**No merge without approval** Mainely Code proposes; the customer approves.	**No training on repos** Customer code is not used for model training.

What counts as a verified PR attempt

A verified PR attempt includes planning, bounded task-packet creation, candidate generation, diff review, validation checks, and proof-bundle output. Some failed attempts may still consume allocated attempts when compute/API resources are used, but early access will classify failures clearly and avoid surprise billing.

Failure class	Customer sees	Customer-facing policy
model_timeout	Provider/model did not complete.	Retry or discount policy is defined in the customer quote.
non_owned_path	Candidate touched a blocked file.	Rejected before apply; handling is visible in the proof bundle.
test_failed	Build or test output failed validation.	Counts when a full attempt ran, with evidence attached.
scope_too_large	Task needs split, Launch Pack scope, or manual estimate.	No surprise overage; quote first.

Early adopter CTA: Request a founding-seat brief or Launch Pack quote. Bring a repo and a short list of jobs. Mainely Code will return scope, caps, expected proof artifacts, and private pricing before work starts.

Contact: tahai.portal@gmail.com · mainelycode.com