PUBLIC PRODUCT VISION & CAPABILITY BRIEF

The local-first AI model-development workbench

If you can dream it, we will help you FORGE it.

NATIVE WINDOWS | LOCAL-FIRST | EVIDENCE-DRIVEN

DEVELOPMENT PREVIEW

MAINELY CODE LLC | SEPTEMBER 2026

THE PRODUCT

The workshop between a model idea and a model people can actually use

Mainely Code's FORGE is a native Windows workbench for preparing datasets, fine-tuning supported AI models, evaluating results, creating qualified deployment variants, and packaging the exact artifacts needed for real use.

It is designed for builders who want more than a training script and more honesty than a green progress bar. FORGE keeps the model, data, configuration, checkpoints, evaluation, exports, and evidence together as one traceable project.

LOCAL-FIRST

Core workflows are designed to stay on the user's machine after required models and components are installed.

END TO END

Data preparation, practical fine-tuning, evaluation, quantization, packaging, and handoff live in one workbench.

WITH EVIDENCE

Every candidate carries lineage, settings, runtime facts, measured results, and export integrity.

Turn an idea for a better model into a traceable, testable, deployment-ready artifact.

What this brief covers

The planned public GA scope, customer workflow, privacy posture, hardware behavior, output packages, and the standards FORGE must meet before release.

Current status

FORGE is in active development. Capability statements in this brief describe the intended qualified product, not a claim that every feature is available today.

ONE WORKBENCH

One traceable model-development lifecycle

FORGE is designed to carry a project from raw source material to a verified package without losing the story of how the final artifact was made.

01
Create a project
Define the goal, workspace, output target, and evidence expectations.

02
Bring a model
Import or explicitly approve a supported download and record exact identity.

03
Prepare data
Map, inspect, clean, deduplicate, split, tokenize, and freeze a version.

04
Configure the run
Choose a qualified method, preset, hardware target, and resource budget.

05
Fine-tune
Run real training with live metrics, checkpoints, cancellation, and recovery.

06
Evaluate
Compare base and candidate on the same held-out tasks and runtime settings.

07
Convert
Create supported merged or quantized variants and reload the exact outputs.

08
Package and hand off
Export a model package, model card, manifest, hashes, evidence, and requirements.

Completion is not a single green light

Training completion, evaluation completion, artifact qualification, export completion, and deployment readiness remain separate states. FORGE records each one honestly.

Source stays source
Original models and raw datasets remain preserved by default. New work produces new, versioned outputs.

The exact artifact is tested
A source candidate cannot stand in for a merged or quantized export. The output that ships is the output that gets loaded and measured.

Lineage never disappears
Model revision, data version, settings, runtime, hardware, commands, and hashes travel through the entire project.

PLANNED GA SCOPE

The complete practical model workshop

The initial product is being shaped around the complete work a serious local model builder must perform - not around a collection of disconnected demos.

1

Project workspaces

Keep models, datasets, transformations, configurations, runs, checkpoints, evaluation, exports, history, and evidence together.

2

Model intake

Record exact revisions, architecture, tokenizer, chat template, special tokens, license information, integrity hashes, and supported operations.

3

Dataset workbench

Import, map, preview, clean, deduplicate, split, tokenize, inspect truncation, flag risks, and preserve source lineage.

4

Guided fine-tuning

Practical LoRA and QLoRA workflows first, with readable presets, advanced controls, resource preflight, and valid checkpoint output.

5

Experiment control

Real job states, metrics, logs, queues, cancellation, restart recovery, checkpoints, exact input binding, and safe retry behavior.

6

Evaluation and comparison

Measure base, tuned, merged, and quantized artifacts on the same held-out tasks with visible samples, failures, speed, and memory.

7

Quantization and conversion

Create supported deployment variants, expose quality-resource tradeoffs, verify output integrity, and load the exported artifact.

8

Packaging and handoff

Produce usable adapters or model packages with manifests, model cards, hashes, lineage, licenses, evaluation status, and runtime requirements.

The principle: Every visible capability must perform real work, report real progress, fail honestly, and leave something another person can inspect.

FOUNDATION

Projects that preserve the work - and the reason behind it

A FORGE project is the durable record of a model-development effort. The workspace should still make sense after a reboot, a month away, a teammate handoff, or a failed experiment.

Project operations

- Create, open, save, reopen, clone, archive, export, restore, and safely delete a project.
- Version dataset transformations, training configurations, runs, checkpoints, evaluation suites, and generated artifacts.
- Bind saved runs to frozen inputs and exact configuration snapshots - not mutable filenames.
- Use atomic persistence, safe migration, project locking, and recoverable backup behavior.
- Handle relocated projects, missing external files, read-only folders, Unicode names, and conflicting app instances explicitly.

A FORGE project is not a folder full of mystery weights. It is the record of how the artifact came to exist.

What model intake records

- Exact upstream identity and revision
- Architecture and weight format
- Tokenizer and configuration
- Chat template and special tokens
- License and source information
- Integrity hashes and shard completeness
- Trainable, adapter, merged, or inference-only state
- Qualified backends and operations

Import or approved download

Local assets can be imported. Supported external downloads require explicit approval, visible source and size, integrity checks, and resumable acquisition.

Clear compatibility boundaries

FORGE will use a qualified model and backend catalog at launch. Unsupported models and operations fail early instead of being presented as probably compatible.

Qualified catalog, not compatibility theater: One hard-coded tutorial model is not enough, but claiming universal model support would be dishonest. Support will be earned and published by exact profile.

PREPARE THE MATERIAL

Know what the model is learning from

FORGE is designed to turn raw files into a versioned, reviewable training asset without silently dropping records or hiding the transformations applied.

Import › Map fields › Inspect › Clean › Split › Freeze version

Planned input lanes

- JSONL
- CSV
- Plain text
- Instruction / response
- Multi-turn conversation

Preparation tools

- Encoding detection, field mapping, row-level preview, and source-location error reporting.
- Filter, edit, normalize, deduplicate, inspect distributions, and save reproducible transformations.
- Preview tokenization, role ordering, target masking, packing, and truncation before training begins.
- Create versioned training, validation, and held-out test splits with deterministic seeds.
- Check exact and configured near-duplicate leakage across splits.

Lineage and risk review

- Preserve the untouched source dataset and produce new derived versions.
- Record source, ownership, permission, and license notes with each dataset version.
- Flag potential secrets and personal data for review, redaction, or exclusion.
- Report excluded records and reasons instead of silently discarding them.
- Keep held-out test material from automatically returning to the training set.
- Treat dataset text as data - never as instructions or permission to run code.

Designed for datasets that outgrow memory

Processing within published limits should use bounded-memory workflows rather than requiring the entire corpus to fit in RAM.

Data truth before model training

The dataset version used for a run is frozen, hashed, and bound to the experiment. Editing the visible source later cannot quietly change an active or historical run.

Automated secret and personal-data detection can help find risk; it cannot guarantee that every sensitive item has been found. Human review remains part of responsible preparation.

PRACTICAL FIRST

Fine-tuning without pretending every machine can do everything

The planned initial focus is real adapter-based fine-tuning - LoRA and QLoRA - across a small, tested compatibility catalog. FORGE will guide the setup, but it will not hide what the settings mean.

Guided mode

Choose a qualified goal and hardware profile. FORGE proposes a readable preset, explains the expected tradeoffs, runs resource preflight, and shows every effective setting before execution.

Advanced mode

Control learning rate, training duration, sequence length, batch size, gradient accumulation, precision, adapter rank, target modules, evaluation cadence, and checkpoint retention.

Before the run

- Validate model-method-backend compatibility.
- Confirm the exact dataset, tokenizer, template, and training targets.
- Estimate host RAM, GPU memory, temporary storage, runtime, and checkpoint footprint.
- Reject conflicting settings and unsupported precision before side effects.
- Show downloads, environment requirements, and external actions before approval.

During and after the run

- Perform actual optimizer steps against the intended trainable parameters.
- Detect zero-step runs, no-op training, invalid loss, wrong target modules, and accidental base modification.
- Write valid checkpoints and adapters that can be reloaded independently.
- Preserve the full effective configuration, runtime versions, hardware, logs, and output hashes.
- Keep failed and regressed experiments as honest results.

> Training completed does not mean the model improved.

What FORGE will not pretend: The initial GA is not a promise of frontier-model training from random initialization, arbitrary architecture support, or identical results across every GPU and driver combination.

EXECUTION TRUTH

Real experiments, real progress, durable recovery

Long model jobs need more than a spinner. FORGE is designed around an authoritative job lifecycle that survives ordinary interruptions without duplicating work or inventing success.

Created › Preflighted › Queued › Running › Checkpointed › Evaluating › Completed / failed

What the operator sees

- Current stage, step, measured metrics, elapsed time, resource use, log events, and checkpoint locations.
- Queue position, exact project/model/dataset/configuration binding, and the next automatic action.
- Cancellation, pause where supported, retry, and resume controls with clear consequences.
- Estimated time only when a defensible estimate exists - otherwise, no invented ETA.
- Specific failure classes, useful diagnostics, preserved evidence, and a safe next step.

What the system protects

- Duplicate clicks, reconnects, or restarts cannot silently launch the same job twice.
- Partial checkpoints and incomplete exports never become valid artifacts.
- Material changes to model, dataset, settings, or destination create a new binding.
- Closing the UI, sleep/wake, worker failure, and app restart follow documented recovery behavior.

EXPERIMENT 024 — RUNNING

Project	Support Assistant v3
Base	Exact revision + hash
Dataset	support-clean-v7
Method	QLoRA profile A
Stage	Training step 1,842 / 4,000
Checkpoint	Valid at step 1,500
Resources	Measured RAM / VRAM / disk
Recovery	Resume supported from full state

True resume is specific. Optimizer, scheduler, random state, and data position must be restored. Restarting from weights alone is recorded as a new continued run.

Cancelled, interrupted, failed, and completed are different outcomes

FORGE does not collapse them into a generic stopped state. Recovery and evidence depend on what actually happened.

THE QUALIFICATION GATE

Evaluate the actual result - not the story around it

FORGE is designed to compare each important artifact under controlled conditions and make regressions as visible as improvements.

ARTIFACT	WHY MEASURE IT	WHAT STAYS CONTROLLED
Base model	Establish the real starting point.	Task suite, prompt/template, decoding settings, hardware, runtime, and measurement method.
Fine-tuned candidate	Determine whether the adapter changed useful behavior.	Held-out data remains held out; sample outputs and failures remain inspectable.
Merged candidate	Prove that a merge did not alter or break behavior unexpectedly.	Exact merged files, tokenizer/template, load result, scores, speed, and memory.
Quantized export	Measure the deployment tradeoff that customers will actually receive.	Exact final artifact, intended runtime, quality regression, latency, throughput, and memory.

Evaluation outputs

- Versioned suites and held-out tasks.
- Task scores and structured-output validation.
- Side-by-side prompts, outputs, references, and failure review.
- Latency, throughput, peak memory, and load behavior.
- Exact artifact, settings, runtime, and hardware identity.
- Controlled reruns and comparison notes when conditions differ.

Qualification discipline

- Promotion criteria and allowed regressions are saved before evaluation.
- A failed criterion blocks qualification while preserving the experiment.
- Generated-code tests run in a restricted test environment, not unrestricted on the host.
- Any model-based judge is clearly identified and is never the sole authority for correctness.
- Test data is not silently recycled into training.

The exported artifact is reloaded and measured. The source candidate is not allowed to stand in for it.

Honest outcome: A candidate can finish training, fail qualification, remain useful for research, and still be preserved without being mislabeled as improved.

MAKE IT DEPLOYABLE

Turn a candidate into qualified deployment variants

Quantization and conversion are treated as engineering operations with compatibility, storage, integrity, reload, and post-export evaluation - not as a file rename.

Source candidate › Merge where supported › Convert › Create profiles › Reload exact output › Re-evaluate

Planned behavior

- Preserve the source candidate and create separate versioned outputs.
- Validate architecture, adapter/base match, tokenizer/template completeness, memory, and temporary disk space.
- Support qualified deployment formats, including a GGUF path where the model and tooling are proven compatible.
- Offer multiple named quantization choices with clear expected tradeoffs.
- Write output metadata, integrity hashes, source lineage, and the exact conversion environment.
- Load and exercise the final artifact in its intended runtime before qualification.

Quality

Measure task-level regression rather than assuming a smaller file is acceptable.

Memory

Record actual load and runtime memory on named hardware.

Speed

Compare load, latency, and throughput under controlled conditions.

Compatibility

Verify the exact runtime and template path expected by the consumer.

The deployment tradeoff should be visible

Smaller can mean faster or easier to run, but it can also mean reduced quality. FORGE is designed to make that tradeoff measurable before an artifact is handed to a user or another product.

Unsupported architecture, mismatched tokenizer, invalid adapter base, malformed output, failed reload, or unacceptable regression blocks a qualified-release claim.

THE DELIVERABLE

Package a model someone else can actually use

A FORGE export is designed to be more than weights. It should carry everything required to understand, verify, load, and responsibly use the artifact.

Model files

Adapter, merged model, or qualified quantized artifact, depending on the selected supported package type.

Runtime assets

Configuration, tokenizer, chat template, special-token data, and resolvable dependencies.

Versioned manifest

Package identity, file list, hashes, source lineage, compatibility, creation tools, and qualification state.

Model card

Purpose, intended use, limitations, source model, data summary, evaluation, hardware, and operating guidance.

License notices

Recorded model, dataset, dependency, and package notices needed for review and downstream handling.

Evidence bundle

Run configuration, measurements, test results, failures, export verification, and reproducibility notes.

Standalone handoff

Export a portable package to a chosen destination, reopen it from that destination, and detect missing dependencies before the user attempts deployment.

Planned Mainely Code PROVIDER handoff

FORGE is designed toward a versioned integration contract for registering and serving approved packages while retaining standards-based standalone export.

Adapter packages remain honest: An adapter-only export names the exact required base model and cannot present itself as a standalone model.

Private datasets, credentials, developer-only paths, and unrelated project material are excluded from distributable packages by default. Unqualified research artifacts remain unmistakably labeled.

USER AUTHORITY

Local-first means the user knows where the work is happening

FORGE is being designed so that core supported workflows run on the user's computer after required assets and components are installed. External services remain an explicit choice, not a hidden fallback.

Stays local by default

Projects, raw datasets, configurations, logs, checkpoints, evaluation, and exports remain on the selected local storage path unless the user deliberately chooses otherwise.

Requires approval

Model downloads, external compute, remote storage, publishing, or any transfer of project data must disclose destination and scope before action.

Never hidden

No silent cloud fallback, background dataset upload, mystery telemetry payload, or automatic publication of models or evidence.

Privacy and security posture

- Protected credential storage and no secrets in logs, manifests, crash reports, or support bundles.
- Support bundle preview and redaction before export.
- Least-privilege local services and no blanket administrator requirement.
- Path traversal, unsafe archive extraction, command injection, malicious filenames, and executable model content treated as security boundaries.
- Untrusted model-supplied executable code disabled by default.
- Approvals bound to the operation and inputs; material changes require a new approval.

Offline and degraded behavior

- Cached local workflows continue without internet access where their qualified backend permits it.
- Offline state is visible and never masquerades as a completed external request.
- Unsupported hardware still allows safe project inspection, evidence access, and data recovery.
- Missing downloads or backends produce clear requirements and a safe next action.
- External failure cannot silently change the selected model, method, or destination.

Your hardware. Your project. Your approval.

Local-first is not a slogan if the product quietly sends the hard parts somewhere else. FORGE is designed to disclose the execution path and ask before leaving the machine.

BEFORE EXECUTION

Hardware-aware, not hardware-blind

The right question is not whether a model loads. It is whether the exact planned operation can run safely on the current machine with enough room for training, checkpoints, conversion, and recovery.

Readiness begins with the actual computer

CPU capability, available RAM, GPU identity, per-device VRAM, driver state, storage headroom, backend health, and competing jobs are measured before the work begins.

Machine profile

- CPU model and required instruction support.
- Installed and currently available system memory.
- Each GPU reported separately with dedicated VRAM.
- Driver, runtime, and backend compatibility.
- Free disk space and output/temp/checkpoint storage plan.
- Observed baseline resource use and relevant competing jobs.

Operation profile

- Training method and precision.
- Sequence length, batch size, optimizer, and memory-saving options.
- Checkpoint frequency and retention.
- Merge and conversion requirements.
- Temporary storage and final package size.
- Expected runtime range, with estimates labeled as estimates.

Loading a model is not proof that training it will fit.

No imaginary pooled memory

Separate GPUs are not treated as one large pool unless the exact backend and workload profile prove that behavior.

Estimates vs measurements

Projected fit, runtime, and storage remain separate from the resource use observed during the actual job.

Published profiles

Minimum and recommended hardware will be based on named model, method, context, runtime, and concurrency measurements.

When a job cannot fit, FORGE should preserve the project and explain the blocker. It should not silently lower settings, change models, or spend hours attempting a configuration that preflight already knew was unsafe.

THE RECEIPT

Evidence travels with every candidate

A model artifact is only as trustworthy as the record behind it. FORGE is designed to produce both human-readable evidence and machine-readable manifests for the exact experiment and export.

MODEL BUILD RECEIPT — QUALIFIED

Base model	Revision + hash
Dataset	Version + split hashes
Settings	Full effective config
Seeds	Recorded
Runtime	App / backend / libraries
Hardware	CPU / RAM / GPU / VRAM
Commands	Exact executed actions
Evaluation	Suite + scores + failures
Exports	File list + hashes + reload

Reproducibility record

- Dataset and model hashes, transformations, split assignments, tokenizer, template, and effective settings.
- Application, backend, library, and dependency versions.
- Hardware and operating environment.
- Commands, logs, checkpoints, output hashes, evaluation, and package verification.
- Rerun instructions and a clear statement of deterministic or nondeterministic limits.

Truthful comparison

FORGE distinguishes an exact replay from a statistically comparable rerun. It does not promise bit-identical outcomes across different hardware, drivers, or nondeterministic kernels unless that exact result is proven.

Verified
The required artifact and evidence passed the defined checks.

Partially verified
Some required proof exists; named gaps remain.

Not verified
The result exists but did not pass the qualification contract.

Blocked
Required hardware, permission, dependency, or input is missing.

No proof, no qualified claim

FORGE can preserve an experiment without promoting it. Failed work, inconclusive work, and useful research artifacts remain visible - and remain honestly labeled.

WHO IT IS FOR

Built for people who need control over the model-development work

Independent AI builders

Create, compare, and package practical local model variants without stitching the entire workflow together by hand.

Application developers

Adapt a supported model to a product's terminology, response format, workflow, or constrained task profile.

Researchers and evaluators

Run controlled experiments, preserve failed hypotheses, compare artifacts, and publish internal evidence without losing provenance.

Privacy-sensitive teams

Keep data preparation and qualified model work local while preserving explicit control over any external transfer.

High-end PC and homelab users

Put capable local hardware to work with exact fit checks, measured performance, and honest resource boundaries.

Organizations building model catalogs

Create repeatable packages, manifests, evaluations, and approval-ready evidence for internal deployment review.

Example projects

- Specialize a local assistant around company terminology and approved workflows.
- Compare multiple dataset-cleaning strategies before selecting a training corpus.
- Create a smaller deployment variant for a known hardware target.
- Produce an adapter package tied to the exact required base model.
- Measure whether quantization changes quality enough to block release.
- Build a repeatable evidence package for internal review or customer delivery.

What it is not

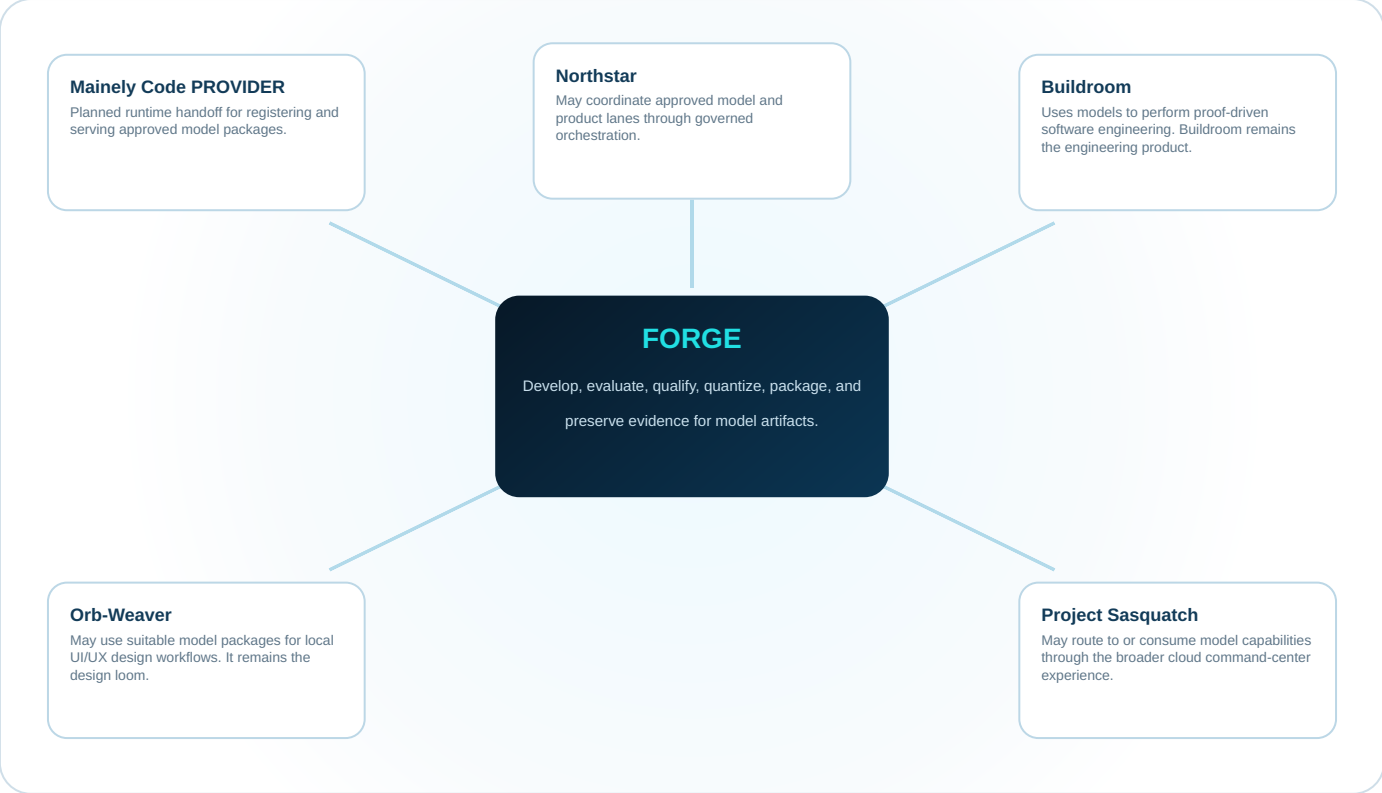
- A promise that every model can be trained on every Windows PC.
- A one-click substitute for data rights, model licensing, privacy, or security review.
- A reason to upload private data without understanding where it is going.
- A magic benchmark score that replaces task-specific human judgment.
- A front for an unbounded cloud bill or hidden compute path.

The practical goal: make serious model-development work more approachable without making it less rigorous.

DISTINCT PRODUCT, CONNECTED SYSTEM

Where FORGE fits inside Mainely Code

FORGE owns the model-development workbench. Other Mainely Code products keep their own responsibilities and connect through explicit, versioned handoffs rather than blurred product boundaries.



FORGE does not become PROVIDER

FORGE creates and qualifies model artifacts. PROVIDER is the planned runtime and serving layer for approved packages.

FORGE does not become Buildroom

FORGE develops models. Buildroom changes software systems with diffs, tests, proof, rollback, and verified engineering outcomes.

Integration labels on this page describe product direction and planned handoffs. Each connection must be implemented and qualified before it is advertised as available.

WHAT GA MUST MEAN

A complete small workshop - not a large dashboard full of promises

FORGE reaches general availability only when a customer can complete the supported lifecycle from installation through export on a qualified Windows system and recover their work when ordinary failures occur.

At GA, a customer should be able to

- Install the production application and complete first-run environment setup without developer-only tooling.
- Create and reopen a durable project.
- Import a supported model and validate a local dataset.
- Run a real qualified fine-tune with visible progress and cancellation.
- Restart and recover according to the documented checkpoint behavior.
- Evaluate the base, candidate, and exported artifact under controlled conditions.
- Create a supported deployment package and reopen it from its final destination.
- Export a complete evidence record and understand what passed, failed, or remained unverified.

Public product promises

- No fake progress and no fake success.
- No hidden cloud fallback or undisclosed upload.
- No dead primary controls or placeholder workflows.
- No training-complete-equals-improved shortcut.
- No unqualified hardware or compatibility claim.
- No exported artifact qualified without reload and evaluation.
- No support bundle that quietly includes raw datasets or secrets.
- No release claim based only on a development build or one green run.

Initial GA focus

Native Windows, local-first projects, qualified model intake, dataset preparation, LoRA/QLoRA training, durable jobs, evaluation, supported conversion and quantization, portable packaging, and reproducible evidence.

Designed for later expansion

Distillation, continued pretraining, broader model families, multi-GPU and distributed training, advanced expert workflows, and deeper Mainely Code integrations.

Not required for the first GA

Frontier-model pretraining from random initialization, arbitrary architecture support, a public model marketplace, or universal cluster orchestration.

Availability: Release timing, pricing, qualified model families, hardware profiles, and exact export formats will be announced after implementation and qualification support the claims.

QUESTIONS

FORGE FAQ

Does FORGE require the cloud?

The core supported product is designed to run locally after required models and components are installed. Downloads or optional external services require explicit approval and a visible data scope.

Can FORGE fine-tune any model?

No. GA support will be published as a qualified model, backend, operation, and hardware matrix. Unsupported combinations will fail clearly instead of being guessed into compatibility.

Will FORGE train a frontier model from scratch?

That is not the initial GA promise. The practical first release prioritizes adapter-based fine-tuning, evaluation, quantization, and packaging on supported local hardware.

Does completed training mean the model is better?

No. The base and candidate must be evaluated on saved held-out tasks. A run can complete successfully and still fail its promotion criteria.

What can I export?

Planned package types include supported adapters, merged models, and qualified quantized variants, together with configuration, tokenizer/template assets, a model card, manifest, hashes, licenses, and evidence.

Will it work with Mainely Code PROVIDER?

A versioned FORGE-to-PROVIDER handoff is part of the product direction. Standalone package export remains important so FORGE does not depend on another Mainely Code product.

What hardware will I need?

Requirements will vary by exact model and operation. FORGE will inspect the real machine, preflight the selected configuration, and publish measured profiles rather than one vague minimum.

Can my data leave the computer without approval?

It should not. External transfers, remote compute, publication, and cloud storage require an explicit action that names the destination and the material involved.

Will FORGE preserve failed experiments?

Yes. Failed, blocked, cancelled, and regressed work can remain useful. FORGE is designed to preserve the evidence and keep the result honestly labeled.

Does FORGE replace licensing review?

No. It can record source and license information and surface missing details, but model, dataset, dependency, privacy, and distribution obligations still require responsible human review.

Can I inspect a project on unsupported hardware?

The application is intended to preserve safe project access, evidence review, and export/recovery paths even when the machine cannot run the selected training or conversion job.

When will pricing and availability be announced?

After the implementation, compatibility matrix, packaged application, and qualified customer workflow are ready to support a public claim.

Public status: FORGE is in development. This brief defines the intended product and the standard it must meet before Mainely Code presents it as shipped.

MAINELY CODE'S FORGE

The model workshop is being built.

If you can dream it, we will help you FORGE it.

From raw data to a deployment-ready model - with the project history, evaluation, and evidence needed to understand what happened.

mainelycode.com | hello@mainelycode.com