Dweve
Weaving Intelligence

ENGLISH EDITION · JULY 2026

MARKETING BROCHURE · 20 PAGES

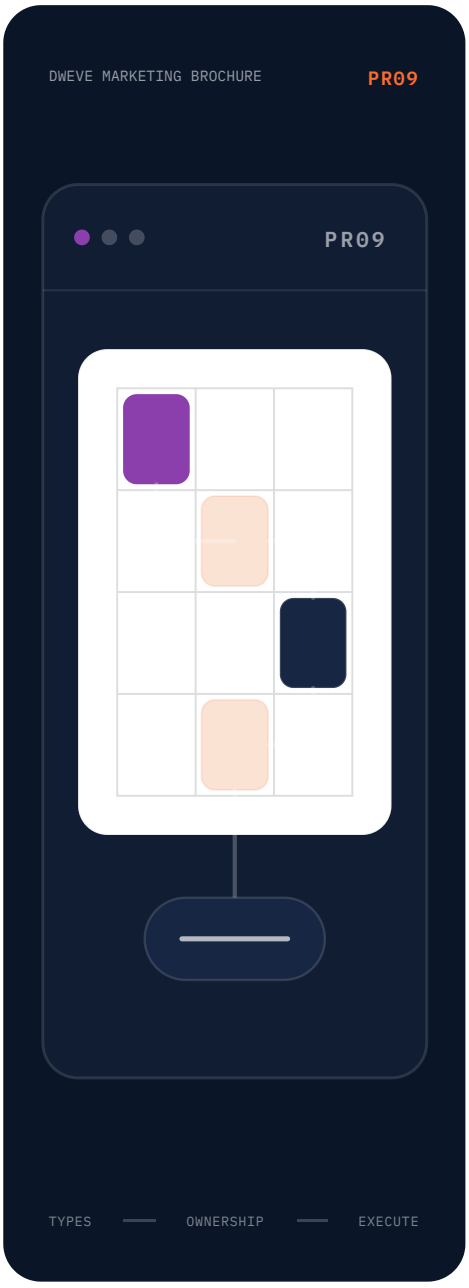
Kera: the product brochure

A standalone visual introduction to Kera, the job it owns, how it works, where it fits, and how to prove it.

VISUAL BRIEFING

PRODUCT LEADERS, BUYERS, ARCHITECTS, DELIVERY TEAMS, AND TECHNICAL EVALUATORS

ENGLISH



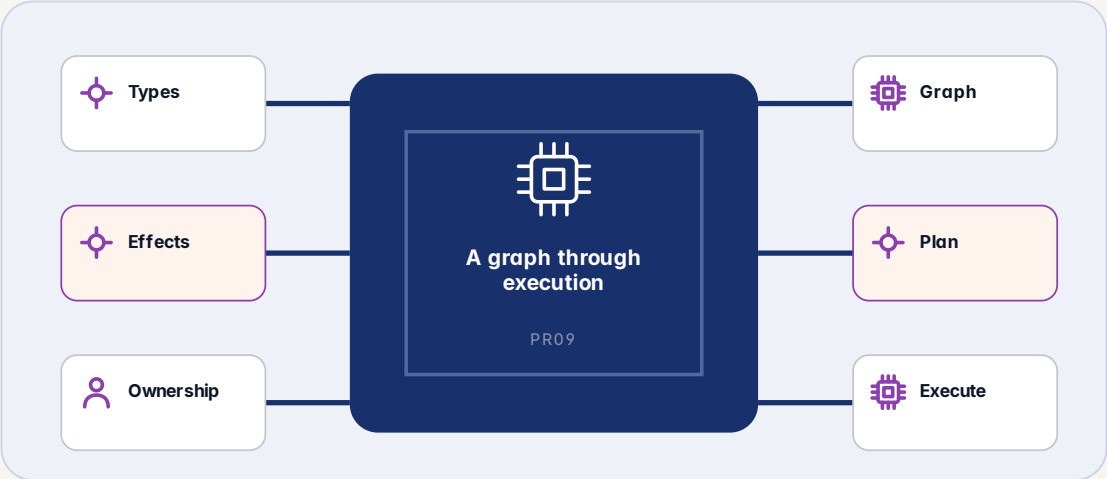
Decide whether Kera belongs in the selected workflow.

PR09

THE PROPOSITION

Graph semantics stay visible through execution

Kera is a graph-native systems language, content-addressed IR, compiler, JIT, and heterogeneous execution system. It keeps types, effects, ownership, dataflow, memory spaces, and execution policy visible while producing target-specific plans for supported CPU, GPU, FPGA, and WebAssembly paths. LLVM is not the centre of the compilation chain.



How to read this visual: follow a graph through execution through the proposition.

CONTEXT

Read the promise against the operating reality

Kera is a graph-native systems language, content-addressed IR, compiler, JIT, and heterogeneous execution system. It keeps types, effects, ownership, dataflow, memory spaces, and execution policy visible while producing target-specific plans for supported CPU, GPU, FPGA, and WebAssembly paths. LLVM is not the centre of the compilation chain.

Kera is a statically typed systems language with a graph-native, content-addressed IR. Programs compile to .keg files, directed acyclic graphs of operations, and execute across CPU, GPU, FPGA, and WASM. No LLVM: custom code generation for x86-64, ARM64, and RISC-V with register allocation and SIMD kernel selection. Effect-tracked side effects. Rust-style ownership across host, device, pinned, and unified memory. A Rust workspace with distributed execution, capability-based security, and LSP tooling.

- Types
- Effects
- Ownership
- Graph

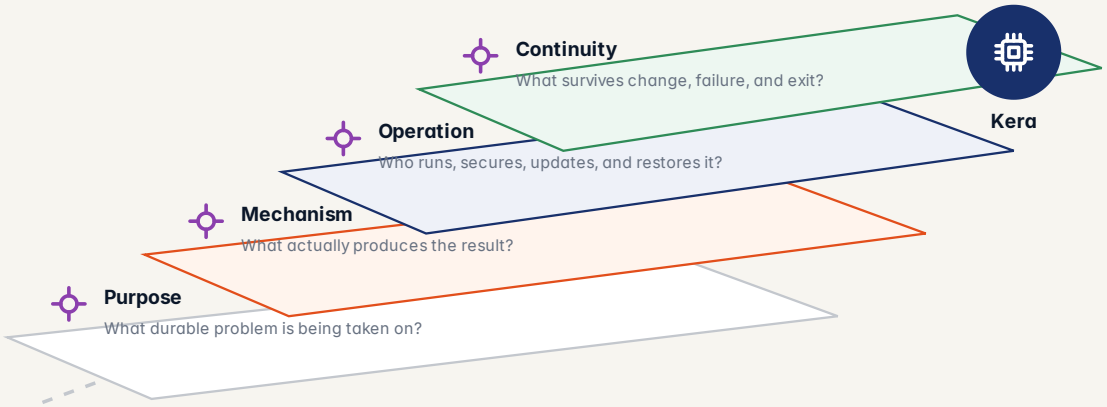
QUESTION TO CARRY

Which part of this proposition changes the outcome people receive today?

DEEPER VIEW

Read the proposition through four connected lenses

A serious decision connects purpose, mechanism, operation, and long-term responsibility.



How to read this visual: follow a graph through execution through deeper view.

BOUNDARY

Connect the mechanism to the responsibility

The emitter can shape live ranges, unrolling, vector width, instruction selection, and reduction structure according to the actual operation and target. The result is not simply source compiled without LLVM. It is a different optimisation architecture, one where the backend still knows what it is scheduling.

Those neutral and parity cases stay visible in the benchmark report rather than being removed from it. Breadth across unrelated families is the point. A gain that repeats across many operation kinds is evidence of an architecture, not of a single favourable kernel.

- Purpose
- Mechanism
- Operation
- Continuity

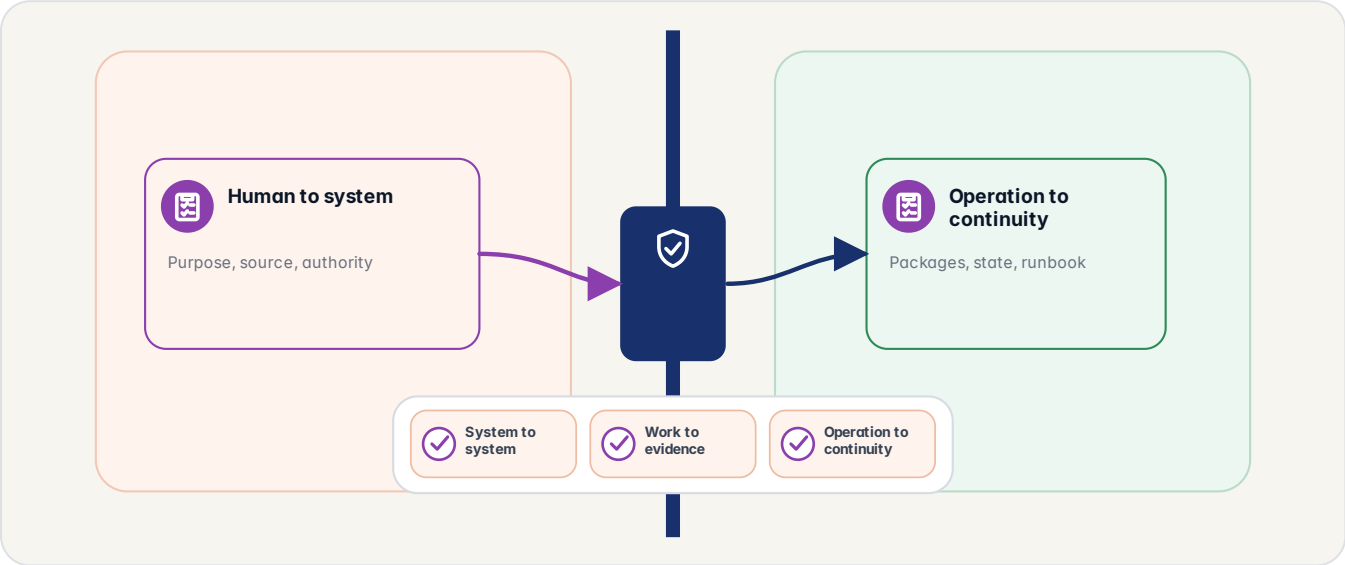
QUESTION TO CARRY

Can purpose, mechanism, operation, and continuity be reviewed together?

BOUNDARY MAP

Interfaces are promises between responsibilities

A clean boundary makes change, testing, replacement, and accountability easier to reason about.



How to read this visual: follow a graph through execution through boundary map.

CONTEXT

Inspect what must survive the hand-off

Some software should be creative. Some should make controlled variations for testing. Some should give the same supported result every time. Kera lets the product choose the right contract before the work runs.

Software can read files, use the network, change stored information, or send work to another device. Kera makes those abilities explicit, so a setup can restrict which abilities are available.

- Human to system
- System to system
- Work to evidence
- Operation to continuity

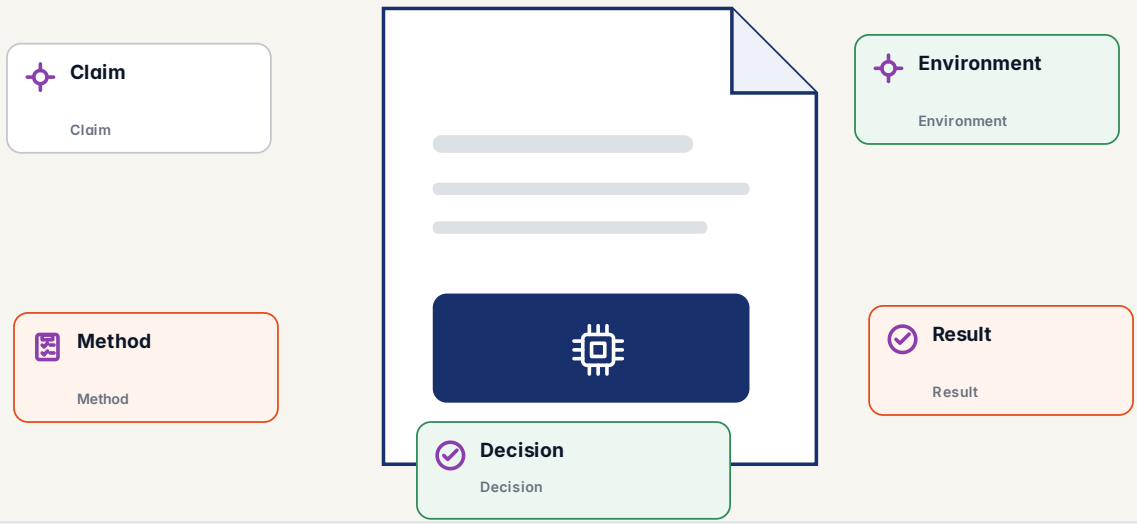
QUESTION TO CARRY

Which interface carries meaning, authority, failure, and evidence across the boundary?

EVIDENCE DESIGN

A claim becomes useful when the reviewer can test it

The brochure makes the claim visible. The proof still runs against the selected workload, environment, and acceptance criteria.



How to read this visual: follow a graph through execution through evidence design.

BOUNDARY

Turn the claim into something reviewable

External operating systems, drivers, firmware, and hardware still exist, and Kera does not pretend otherwise. But the central chain that decides what the programme means and how it becomes execution belongs to one product and one accountable engineering organisation. That ownership lets strategic customers negotiate support, escrow, and target commitments around the actual system.

Planning, fusion, placement, memory movement, lowering, JIT compilation, target emission, distributed partitioning, and replay are all derived from that graph. The programme stays semantically rich until Kera emits the execution plan for the target. That is why Kera can outperform optimised generic code where lowering has already discarded information Kera still holds.

- Claim
- Method
- Environment
- Result

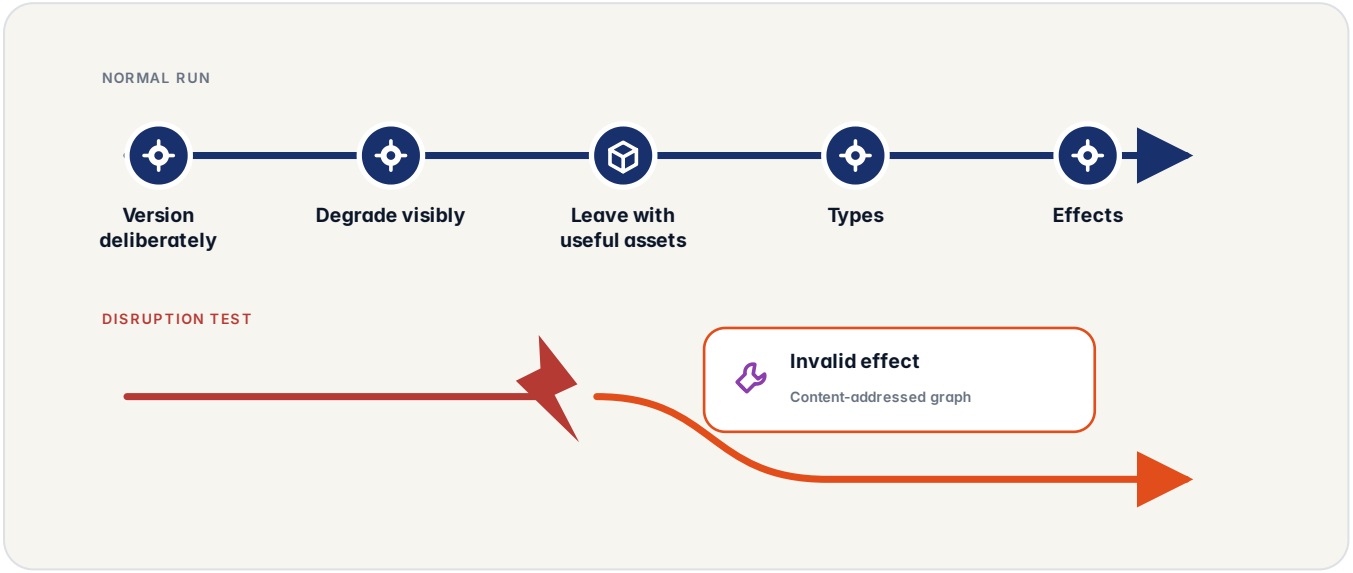
QUESTION TO CARRY

What would a reviewer need to reproduce or challenge the result?

CONTINUITY

Long-term fit includes change, failure, and exit

Evaluate the normal day and the difficult day before increasing organisational reliance.



How to read this visual: follow a graph through execution through continuity.

CONTEXT

Test the difficult day as well as the normal day

Most programmes pass through many layers before the computer does the actual work. One piece describes the task, another translates it, another turns it into more general instructions, another chooses a library, and another sends it to the processor. Each layer is useful, but each can lose information about what the task really was. Kera keeps the work as one clear plan and prepares the version that fits the machine doing the job.

Change a node and its identity changes. Keep it unchanged and the identity stays stable. An organisation can identify the exact graph that was reviewed, the graph that was deployed, the subgraph that changed, the effects added or removed, and the output associated with a controlled run. The artefact can be approved, pinned, compared, cached, reproduced, and rolled back.

- Version deliberately
- Degrade visibly
- Leave with useful assets
- Types

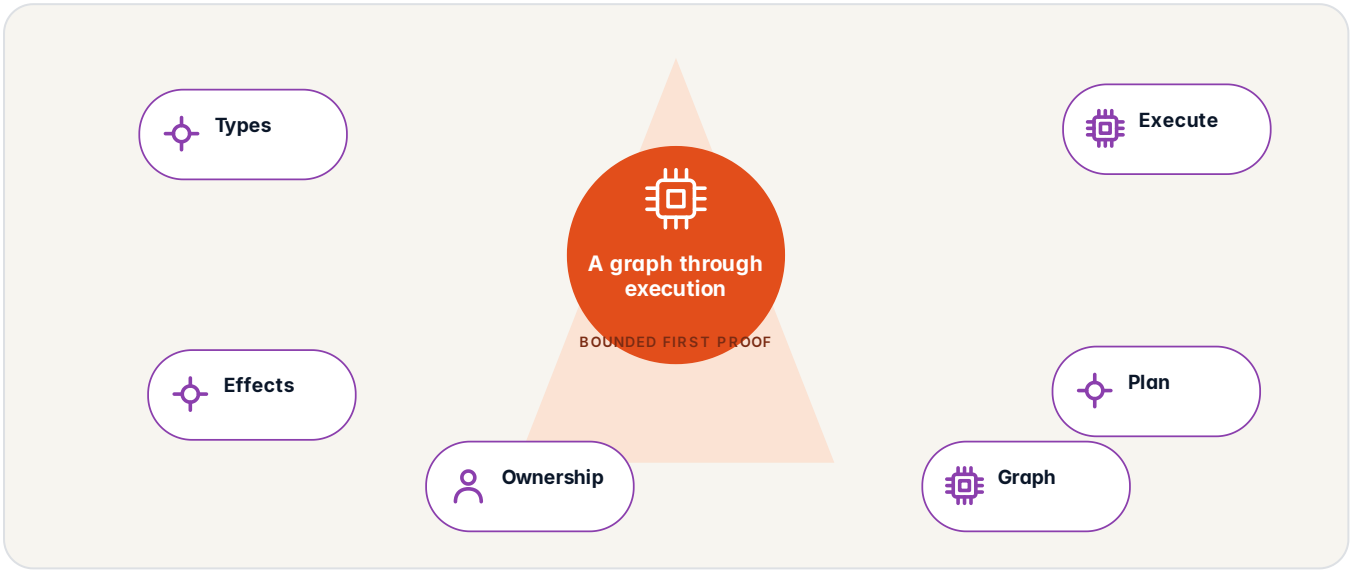
QUESTION TO CARRY

What must remain usable through change, failure, supplier loss, or exit?

USE AND FIT

Where Kera can earn its place

Start with responsibilities that have a real owner, useful sources, and a consequence worth improving.



How to read this visual: follow a graph through execution through use and fit.

BOUNDARY

Separate useful scope from attractive distraction

Software often passes through so many translation layers that the computer no longer sees the job as a whole. Kera keeps the calculation organised as one graph, then prepares an execution plan for the machine doing the work. That can remove needless steps, reduce movement, and leave a clearer record of what actually ran.

Kera can therefore plan with information a generic compiler chain often loses before final optimisation. It chooses an implementation because it knows the operation, not because it recognised a loop pattern late in the pipeline. It fuses work because it sees the graph relationship, and it plans movement because ownership belongs to the programme.

- Types
- Effects
- Ownership
- Graph

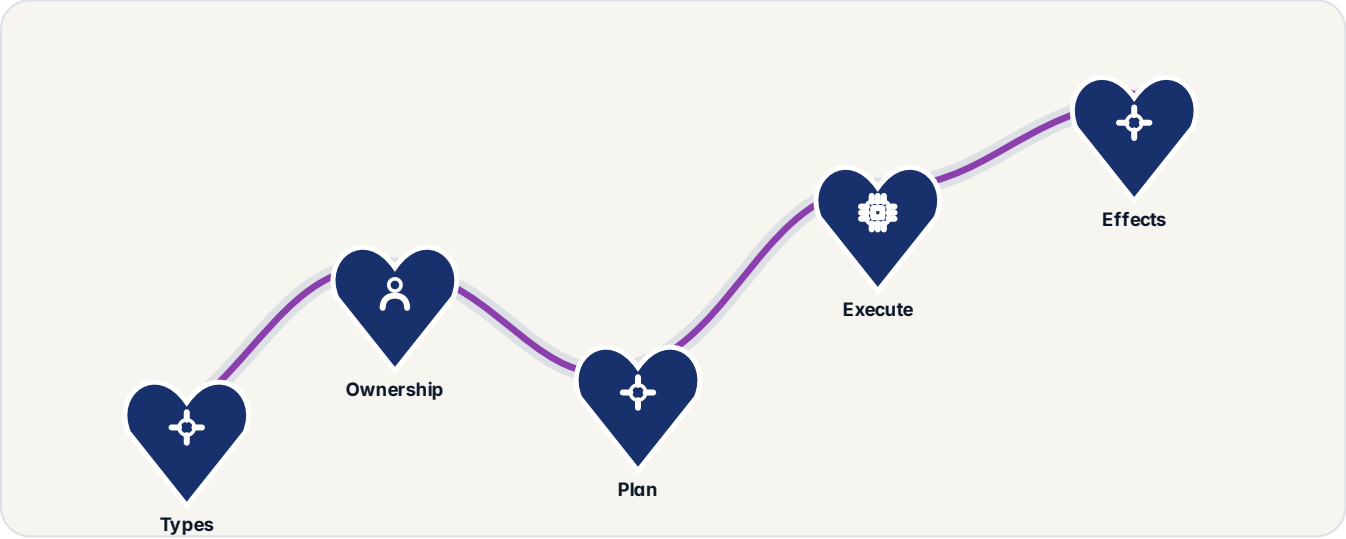
QUESTION TO CARRY

Where does this earn a place, and where should it deliberately not be used?

THE PATH

How Kera moves from question to decision

A focused engagement should make the next decision easier, not create a larger promise.



How to read this visual: follow a graph through execution through the path.

WHY THIS SHAPE WORKS

The workflow, operating boundary, evidence, and decision criteria are considered together from the start.

CONTEXT

Make every stage earn the next one

The advantages include operation-aware algorithm selection, graph-level fusion, specialised approximations under an explicit accuracy contract, reduced intermediate materialisation, direct SIMD emission, target-specific register use, packing-aware loops, branch removal, movement elimination, deterministic reduction structure, and shape-specialised kernels. The common cause is architectural: Kera retains information a generic lowering path no longer has.

Current internal benchmarks compare the Kera emit path against optimised LLVM output for the same operation family and tested shapes. The largest measured leads appear on vector logarithm, on large F32 and F64 matrix multiplication from 512 by 512 upwards, on vector exponential, on vector tanh, and on binary dot product.

- Types
- Ownership
- Plan
- Execute

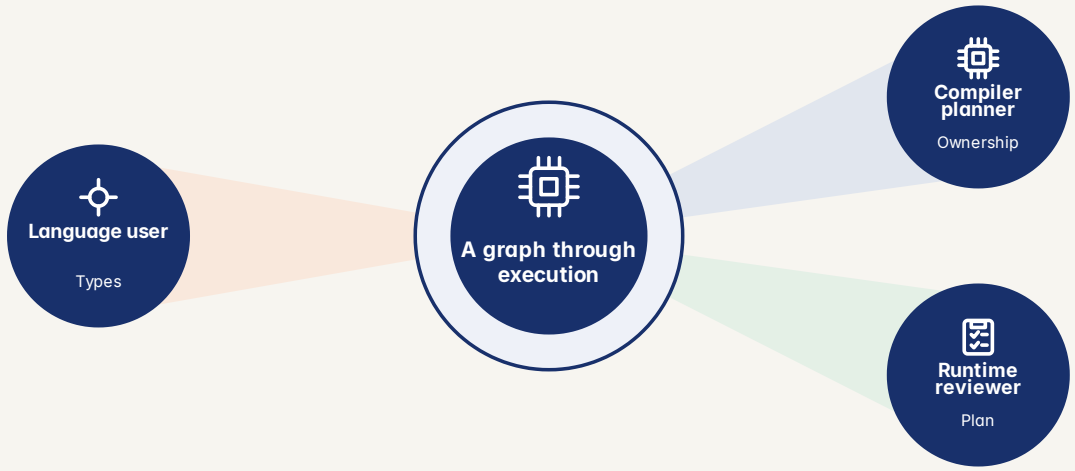
QUESTION TO CARRY

What evidence must exist before the scope is allowed to grow?

THE VALUE PICTURE

Kera seen from three responsibilities

Good AI work must make sense to the person using it, the team operating it, and the people reviewing it.



How to read this visual: follow a graph through execution through the value picture.

PRIMARY READER

Product leaders, buyers, architects, delivery teams, and technical evaluators

The first conversation.

DECISION SUPPORTED

Decide whether Kera belongs in the selected workflow.

The next useful step.

BOUNDARY

Read the same outcome from several responsibilities

Imagine asking someone to bake a cake, then passing the instruction through several people. By the time it reaches the kitchen, the final person has a list of movements but no longer knows they are meant to make a cake. They can follow the list, but they cannot make the same sensible choices as someone who still understands the job.

Exact figures are not shown here because results depend on hardware. Before public publication each figure must travel with the exact hardware, operating system, Kera revision, LLVM and frontend versions, optimisation flags, input distributions, warm-up policy, sample count, confidence interval, and source harness. The point is reproducible evidence, not decorative bars.

- Types
- Ownership
- Plan
- Effects

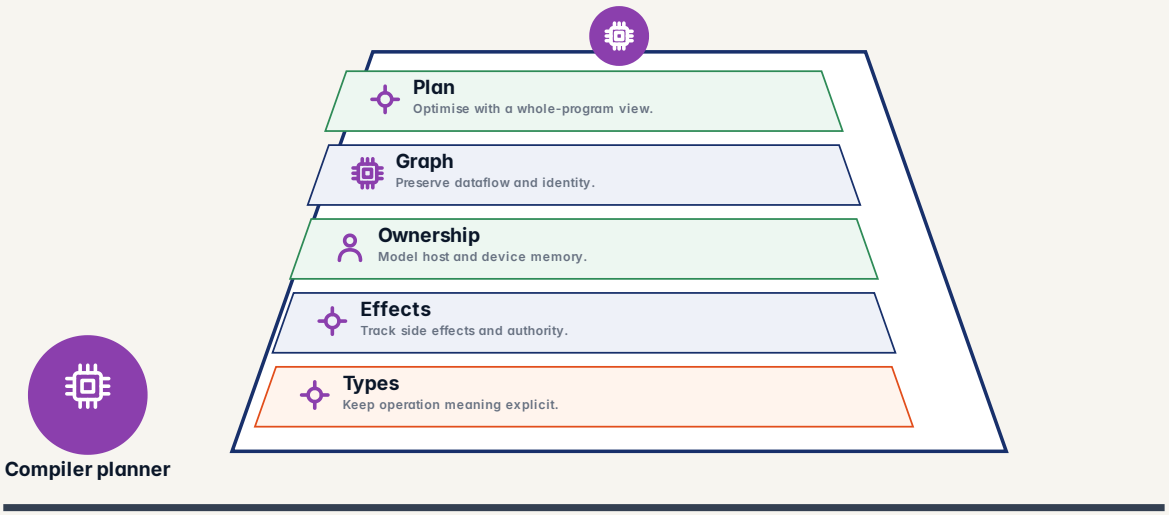
QUESTION TO CARRY

Can the same result satisfy its user, operator, and reviewer?

THE OPERATING MODEL

The control model around Kera

A credible proposition connects the visible result to the less visible responsibilities underneath it.



How to read this visual: follow a graph through execution through the operating model.

CONTEXT

Name the controls behind the simple interface

Most compilers stop understanding the computation long before the machine executes it. Kera keeps the operation, graph, type, effect, ownership, memory, and execution contract visible through planning, then emits target-specific execution itself. No generic compiler sits in the centre trying to rediscover the operation from lowered code. The graph remains the programme, and it goes straight to the machine.

Kera can see which steps belong together. It can avoid creating temporary results that are used only once. It can keep information close to the processor that needs it, and reduce unnecessary movement between parts of the machine. That can mean fewer delays, less memory use, and less energy spent moving data around.

- Types
- Effects
- Ownership
- Graph

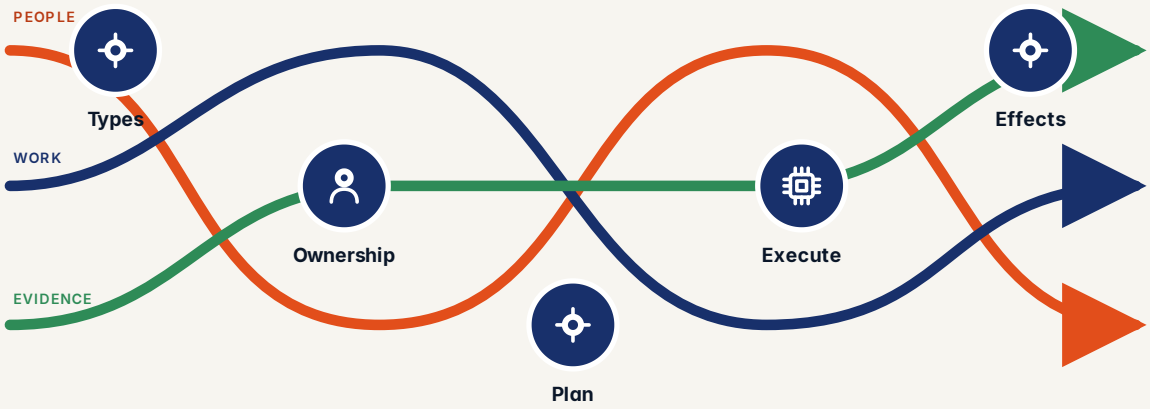
QUESTION TO CARRY

Who owns each control when the interface appears simple?

THE RESPONSIBILITY FLOW

People, Kera, and evidence move together

The work becomes easier to govern when each stage leaves a clear hand-off.



How to read this visual: follow a graph through execution through the responsibility flow.

BOUNDARY

Keep every hand-off attached to an owner

That wider field of view lets Kera fuse operations, remove intermediates that never need materialising, delete transfers, keep data in one memory space, run parallel regions together, and select target-specific kernels. Performance is therefore not limited to instruction selection. It includes preventing unnecessary work from existing at all.

In ordinary use there are no Kera controls to find. The product chooses how the work is kept together, the plan, the machine, and the behaviour it needs, and you receive faster and steadier results.

- Types
- Ownership
- Plan
- Execute

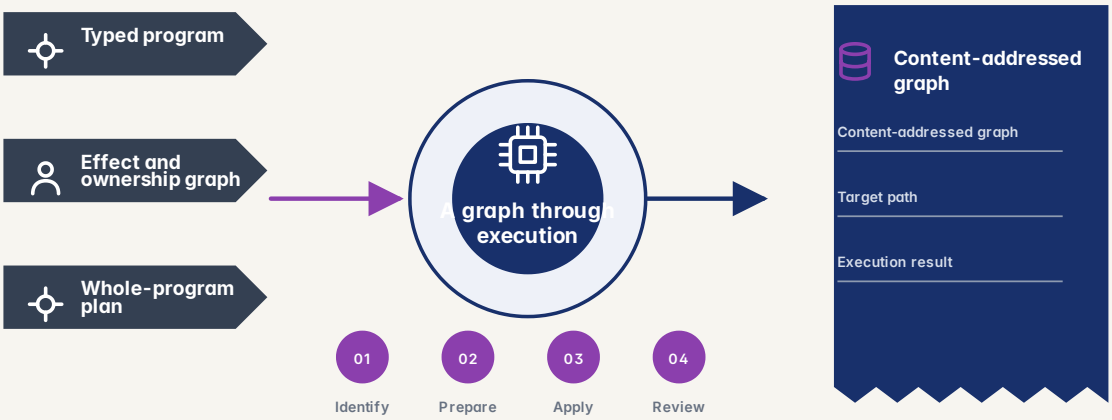
QUESTION TO CARRY

Is every hand-off visible, bounded, and assigned to an owner?

SOURCE TO RESULT

Kera: from authoritative source to result

The work stays explainable when source identity, transformation, decision, and hand-off remain visible between systems.



How to read this visual: follow a graph through execution through source to result.

CONTEXT

Keep the basis visible as the work changes form

The whole job stays intact, the machine receives work designed for it, and Dweve owns more of the route from programme to result. Kera stays out of sight. The result can arrive faster, with fewer hidden layers and a clearer record of what actually ran.

The whole job stays intact, the machine receives work designed for it, and the result can arrive with fewer hidden layers and a clearer record of what actually ran. Kera stays out of sight while it shortens the path underneath.

- Identify
- Prepare
- Apply
- Review

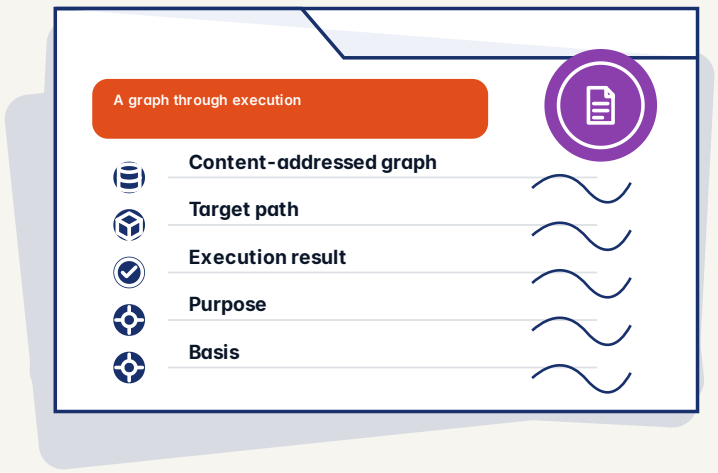
QUESTION TO CARRY

Can the result still be connected to the exact basis that shaped it?

EVIDENCE PACKET

The evidence Kera leaves with the work

Capture enough to explain the result, inspect authority, investigate failure, and make the next decision without rebuilding the story from memory.



How to read this visual: follow a graph through execution through evidence packet.

BOUNDARY

Preserve the record needed by the next question

It keeps one description of the work and prepares a suitable plan for the machine available. The plan changes, but the job does not quietly become a different job. This helps Dweve bring the same underlying products to more kinds of hardware without rebuilding their foundations each time. Support can vary by product and target.

Kera keeps the work as a sealed record, which gives it its own identity. If the work changes, the identity changes. If it stays the same, the identity stays stable.

- Purpose
- Basis
- Authority
- Action

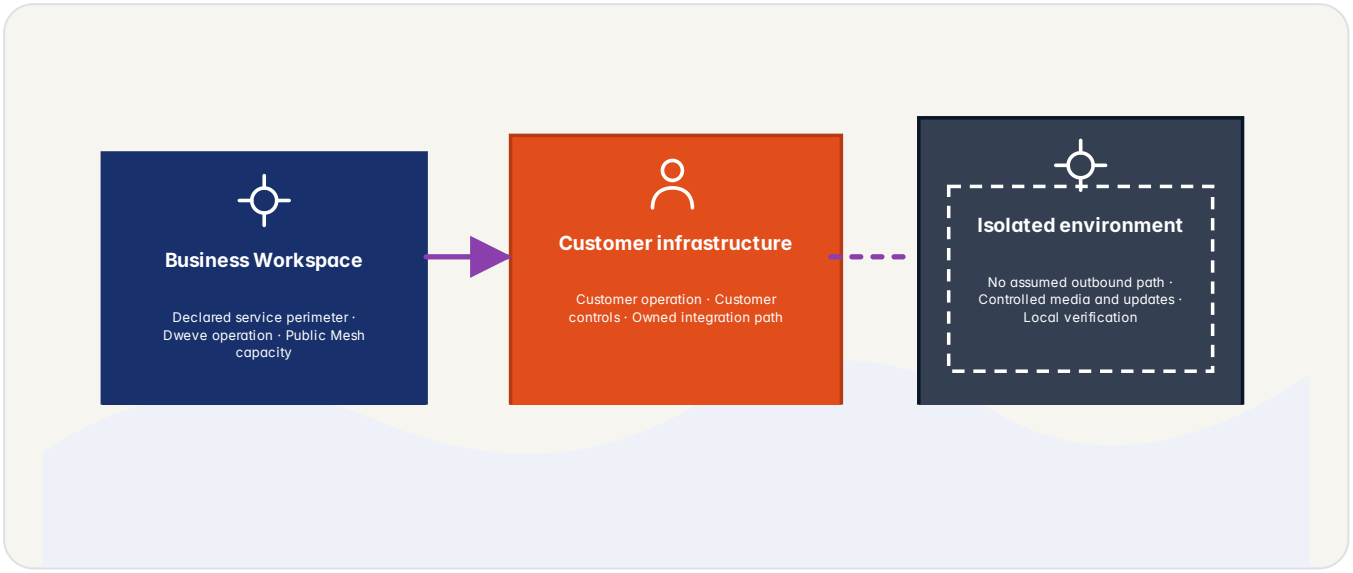
QUESTION TO CARRY

Does the retained record answer the next difficult question without reconstruction?

DEPLOYMENT CHOICE

Choose the operating posture for Kera

Location is only one dimension. Decide who operates, who holds keys, which dependencies exist, how updates arrive, and what continuity requires.



How to read this visual: follow a graph through execution through deployment choice.

CONTEXT

Compare operating responsibility, not location alone

You do not interact with those parts. Their value is that Dweve can improve the actual route from programme to processor rather than waiting for another company to change its plans. External hardware and operating systems still exist, and Kera does not pretend otherwise.

Kera tracks effects in the language and graph. Capabilities determine which effects a deployment permits. A workload cannot silently acquire a network dependency because one implementation changed. An air-gapped deployment can reject outbound capability before execution, and a regulated programme can inspect the declared effect surface of the graph it approves.

- Begin through managed Fabric
- Run on customer infrastructure
- Close the environment
- Types

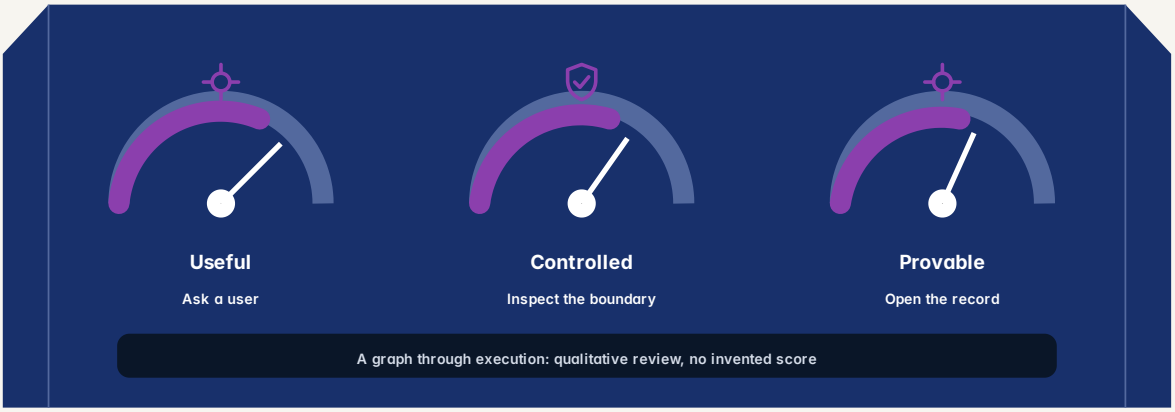
QUESTION TO CARRY

Who holds infrastructure, keys, updates, logs, recovery, and the exit path?

WHAT TO LOOK FOR

Signals that Kera is earning trust

Progress is a useful outcome under understood control, supported by reviewable evidence.



How to read this visual: follow a graph through execution through what to look for.

BOUNDARY

Use signals that can change the next decision

The strict path can keep controlled execution where the supported operation and target provide that guarantee. A seeded path can repeat the same variation. A creative path can allow different answers where that is useful. The computer does not change behaviour by accident merely because the work moved to another machine.

Kera does not claim that every programme beats every generic compiler. It shows something more useful. On current internal tests, the Kera emit path can exceed an already optimised generic path by a wide margin on several operations, while the claim stays specific to the operation and its tested shapes.

- Useful
- Controlled
- Provable
- Types

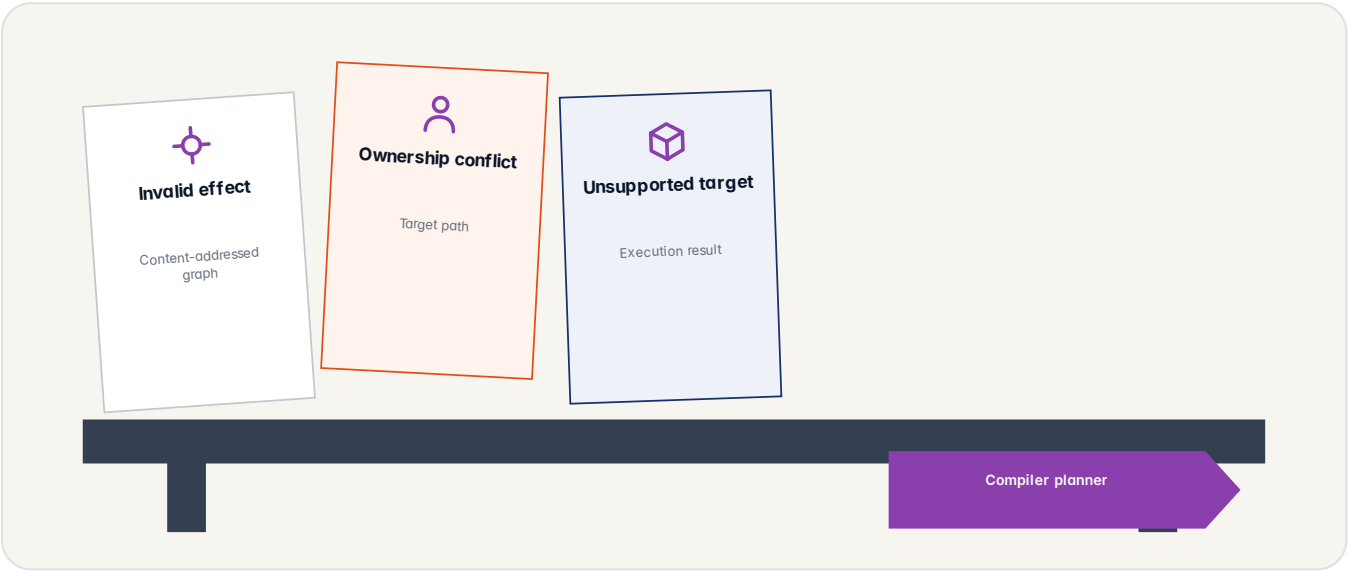
QUESTION TO CARRY

Which signals would change the next decision rather than merely impress the room?

HONEST LIMITS

Test how Kera fails before the first success demo

Visible failure is more useful than a polished result whose limits remain unknown.



How to read this visual: follow a graph through execution through honest limits.

CONTEXT

Design the failure before presenting the success

Some listed shapes are currently parity cases or do not yet show a Kera lead. They remain visible in the benchmark report rather than being removed from it. A matrix by operation family and size makes the wins, the parity cells, and the pending-analysis cells all legible at once.

CPU code generation can select among supported x86-64, ARM64, and RISC-V paths, including available SIMD instruction sets. GPU paths can emit for supported CUDA and ROCm environments and integrate with the target execution model. FPGA targets can receive supported synthesis artefacts and resource estimates. WebAssembly can carry supported graphs into browser and sandboxed environments. Per-target capability differences remain honest.

- Invalid effect
- Ownership conflict
- Unsupported target

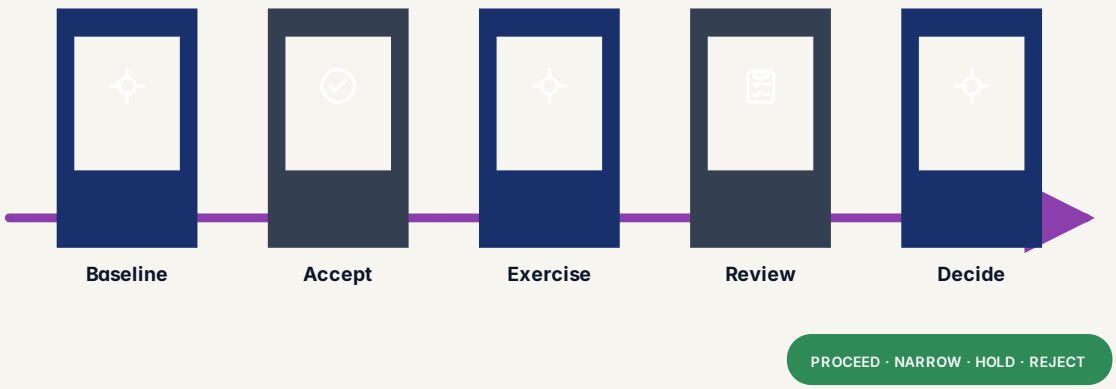
QUESTION TO CARRY

What happens when a source, permission, target, model, or supplier is unavailable?

ACCEPTANCE PATH

Close the Kera proof with a decision

Agree the decision path before the demonstration so a strong or weak result can change what happens next.



How to read this visual: follow a graph through execution through acceptance path.

BOUNDARY

Close the proof with an owned decision

Kera owns the path from the planned graph to target code. There is no LLVM IR stage at the centre. The Kera compiler and JIT perform their own lowering, register allocation, instruction selection, SIMD path selection, kernel emission, and runtime integration for the supported targets.

Because Kera still understands the work, it can sometimes choose a much better way to perform it. In current internal tests, several basic calculations run several times faster than versions already prepared by a strong general purpose tool.

- Baseline
- Accept
- Exercise
- Review

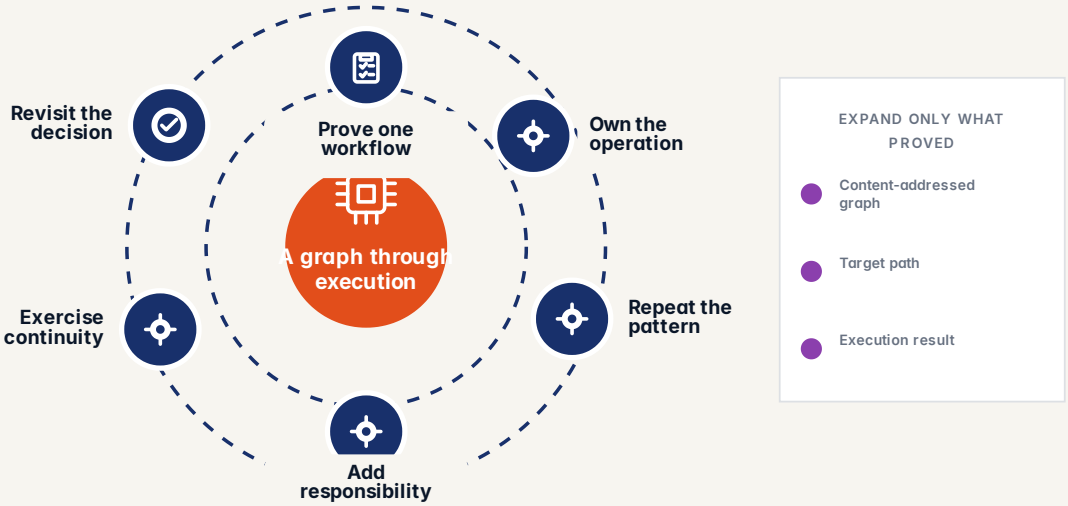
QUESTION TO CARRY

Are success, counter-metrics, failure cases, and stop conditions agreed first?

ADOPTION

Adopt Kera through owned choices

Move from one bounded result to a repeatable operating capability without making the scope vague.



How to read this visual: follow a graph through execution through adoption.

CONTEXT

Expand only what proved reusable

The Kera target-portfolio option defines supported targets and delivery scope. Support, source escrow, source review, operational source, air-gapped commissioning, operator-held keys, and technology transfer remain separate rights or services unless the agreement expressly includes them. Open companion projects remain separate.

A conventional compiler often optimises after the programme has been separated into layers, each owned by a different tool. It optimises the local representation it receives. Kera can plan the computation as a whole graph before it emits.

- Prove one workflow
- Own the operation
- Repeat the pattern
- Add responsibility

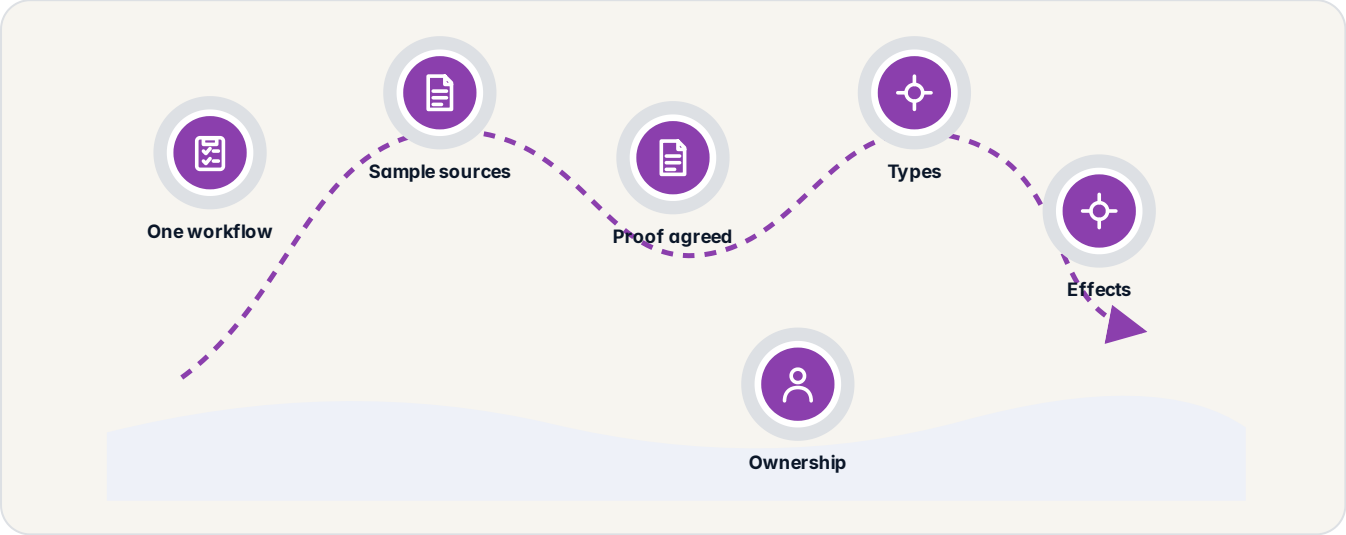
QUESTION TO CARRY

What can be reused safely, and what requires a new decision and proof?

WHAT TO PREPARE

What to bring to the Kera working session

A small amount of real context produces a better conversation than a broad catalogue of hypothetical features.



How to read this visual: follow a graph through execution through what to prepare.

KEEP IT BOUNDED

Sensitive production data is not required for the first conversation. Bring representative material and the rules that define a useful result.

BOUNDARY

Bring enough reality to make the session useful

Bring production LLVM, real flags, real shapes, the real accuracy contract, and the target that matters. Publish both harnesses and raw results. Then inspect which semantic facts produced the difference. The claim is not that custom code generation is automatically better. It is that retaining semantic information and owning the emit path creates opportunities a generic compiler cannot reconstruct.

Bring the production operation, its real shapes, the actual frontend, the strongest generic flags, the target machine, the accuracy requirements, and the deterministic contract. Compare execution, movement, emitted code, graph, and result. The question is no longer whether a custom compiler can catch a generic one. It is what becomes possible when the compiler never forgets the computation.

- Bring one workflow
- Bring representative sources
- Agree the proof
- Types

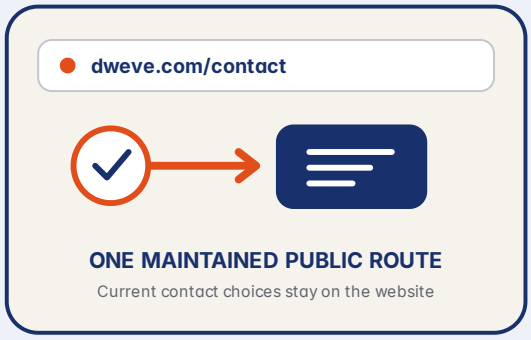
QUESTION TO CARRY

Which real sources, rules, owners, and examples will make the first session honest?

THE INVITATION

Continue the Kera conversation

Scope a briefing, demonstration, or proof around the workflow and decision that matter to your organisation.



CONTINUE ONLINE

dweve.com/contact

Current route and contact choices

START A CONVERSATION

Continue through the current contact page

Scope a workflow or ask a technical question, then use the contact page to begin.

START	**dweve.com/contact**	Use the maintained public contact route
BRING	**One bounded question**	Workflow, audience, decision, and desired evidence
CHOOSE	**Briefing · demonstration · PoV**	Select the smallest useful next step
VERIFY	**Current scope and terms**	Confirm availability, pricing, and responsibilities before commitment

CONTEXT

Turn the brochure into one bounded next step

Most individual users never choose or set up Kera directly. They experience the products built above it where it is used. Where Kera is used, where it is built and who controls it matter, but the product on top stays the same to use.

Kera does not rely on an outside tool to turn its work into something the machine runs. Dweve built the whole route itself, from the way the work is described to the way it is planned, prepared, and run on the supported machines.

- Start: dweve.com/contact
- Bring: One bounded question
- Choose: Briefing · demonstration · PoV
- Verify: Current scope and terms

QUESTION TO CARRY

What is the smallest useful next conversation Dweve should prepare?