Lateralus 1.5 Release Notes

Gradual typing, pipelines, and the first self-consistent toolchain

Lateralus Language

bad-antics · April 2026 · Lateralus Language Release

ABSTRACT Lateralus 1.5 is the first release in which every user-facing tool — compiler, interpreter, formatter, linter, LSP, DAP, and C backend — operates over a single canonical AST and a single canonical inference pass. This document records the user-visible changes, the bug fixes, the breaking changes (one, documented below), and the upgrade path from 1.4. It is intended to be readable front-to-back by library authors considering the version bump and as a reference for downstream editor-plugin authors.

1. New Language Features

1.1 Gradual type inference on by default

let bindings no longer require type annotations in the common case; the 1.5 inference pass (Hindley-Milner with bidirectional refinements) handles lists, maps, records, pipelines, and generic functions without intervention. See the companion paper Type Inference in Lateralus 1.5 for the algorithmic details.

1.2 Pipeline-assign operator

x |>= f is shorthand for x = x |> f. Especially useful for sequential ETL patterns:

```
let mut data = load_csv(path)
data |&gt;= drop_nulls
data |&gt;= normalize_columns
data |&gt;= sort_by("timestamp")
```

1.3 Where clauses

Return expressions can carry a where block of helper bindings:

```
fn distance(p, q) -&gt; float:
    return sqrt(dx*dx + dy*dy) where:
        let dx = p.x - q.x
        let dy = p.y - q.y
```

Semantically identical to a let chain above the return; aesthetically closer to maths and to SQL.

1.4 Measure and probe blocks

measure "label" { ... } times the enclosed block and emits the result to the profiler; probe "label" { ... } emits the final expression value to the trace stream. Both compile away under --release.

1.5 List comprehensions

[x * x for x in xs if x > 0] desugars to an iterator pipeline. The compiler optimizes simple cases (single source, no nested generators) to a direct loop.

1.6 Spread syntax

[...a, ...b] flattens lists at construction; {...r, x: 1} extends records, with later keys winning.

2. Standard Library

- stdlib/strings: added split_n, rsplit, strip_prefix, strip_suffix, replace_all.
- stdlib/math: added clamp, lerp, round_to (n-decimal round).

- stdlib/io: new module; io.println, io.read_line, io.open, io.walk.
- stdlib/time: time.now_ms, time.sleep_ms, time.format.
- stdlib/json: json.parse, json.stringify with pretty-print option.
- stdlib/testing: a pytest-style harness; the compiler discovers test_* functions and runs them under lateralus test.

3. Toolchain

3.1 Unified AST

Every tool now consumes the same AST that the compiler produces; no more divergence between the linter's parser and the compiler's parser. One practical consequence: a syntax error reported by the linter has the same position and message as the one reported by lateralus build.

3.2 Language Server Protocol

lateralus lsp now supports: hover (types), go-to-definition, find-references, rename-symbol, document-symbols, code-lens, and diagnostics on save. The VS Code extension is updated in the marketplace to match.

3.3 Debug Adapter Protocol

lateralus dap speaks DAP over stdio; breakpoints, stepping, stack traces, and local-variable inspection all work against the VM target. Native-code targets (C backend, bytecode) are not yet debugger-integrated.

3.4 C backend

lateralus c source.ltl -o source.c emits freestanding C99 suitable for embedding in existing projects. --freestanding targets kernel/OS use with no libc dependency. See the companion paper C Backend Transpiler Design.

3.5 Formatter

lateralus fmt is deterministic and idempotent; the file on disk round-trips through parse-format-parse bit-for-bit identical.

3.6 Linter

40 lints shipped in 1.5, covering dead code, shadowing, unused parameters, magic numbers, pipeline anti-patterns, and concurrency hazards. lateralus lint --fix applies the safe ones automatically.

4. Bug Fixes

- Parser: fix off-by-one column reporting on multi-line string literals (#412).
- Inference: the occurs check now fires at bind time rather than at apply time; eliminates a class of pathological slowdowns (#437).
- Codegen: pipeline-heavy expressions no longer allocate per-stage closures when the stage is a named function; 2-4x speedup on map/filter pipelines (#451).
- VM: integer-literal boxing eliminated on arithmetic paths; measurable on numeric benchmarks (#462).
- LSP: hover no longer returns stale types after edit-undo sequences (#478).

- Formatter: trailing-comma handling in match arms fixed (#491).

5. Breaking Changes

Exactly one:

5.1 let without annotation is no longer an error

In 1.4, let x = expr without a type annotation was rejected under --strict mode. In 1.5, --strict no longer implies mandatory annotation; inference supplies types silently. Code that relied on the error (for example, CI pipelines that grep for it) should switch to lateralus lint --rule require-annotations which still flags untyped lets under the explicit opt-in.

No other source-level breakage. Binary formats (bytecode, cached ASTs) are version-gated and automatically regenerated on the first run against 1.5.

6. Upgrade Path

- **Python package**: pip install --upgrade lateralus-lang.
- **From source**: git pull && make install.
- **VS Code extension**: Marketplace auto-update, or reinstall from .vsix in vscode-lateralus/.
- **Existing projects**: no code changes required for 1.4-compatible code. Run lateralus fmt after upgrade to pick up any minor style adjustments.
- **Bytecode caches**: will be regenerated automatically on first build.

7. Benchmarks

Measured on an M1 MacBook Pro, Python 3.11, best-of-5:

- examples/fibonacci.ltl (VM): 68 ms in 1.4, 52 ms in 1.5 (24% faster).
- examples/data_pipeline.ltl (VM): 210 ms in 1.4, 160 ms in 1.5.
- examples/crypto_challenges.ltl (VM): 1.1 s in 1.4, 0.85 s in 1.5.
- Full test suite (pytest tests/): 12.4 s in 1.4, 11.9 s in 1.5.

Most of the gain comes from the closure-elimination fix (#451) and the inference-occurs-check fix (#437); the remainder is minor allocator and lookup-path improvements.

8. What's Not in 1.5

- **Higher-rank types**: postponed to 1.6.
- **Effect system**: design in progress, target 1.7.
- **GADTs**: target 1.7.
- **Native AOT compiler**: the C backend covers the near-term need; a dedicated LLVM backend is on the long-range roadmap but not scheduled.

9. Acknowledgements

Thanks to the 22 contributors who filed issues or submitted patches against the 1.4 series. Particular thanks to early adopters who tested the gradual-inference branch for eight weeks before merge and identified three correctness bugs in the row-polymorphism code path. The release would not be shippable without their patience.

10. Next Release

Lateralus 1.6 is targeted for Q3 2026. Primary themes: higher-rank types, first-class modules, and a stable bytecode format suitable for distribution. Follow CHANGELOG.md on the main repo for in-progress notes.