

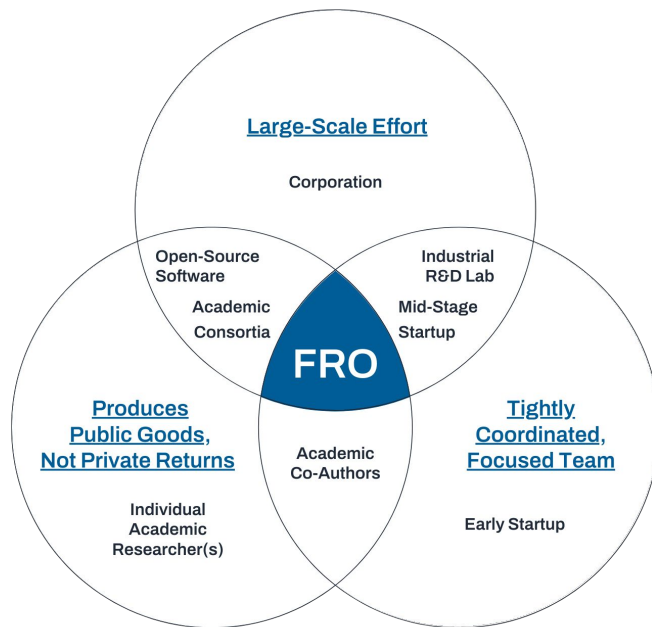
# Monthly Community Meeting

Lean FRO  
13 October 2023

# Focused Research Organization (FRO)

A new type of nonprofit startup for science developed by Convergent Research.

[convergentresearch.org](https://convergentresearch.org)



# The Lean FRO

## Missions:

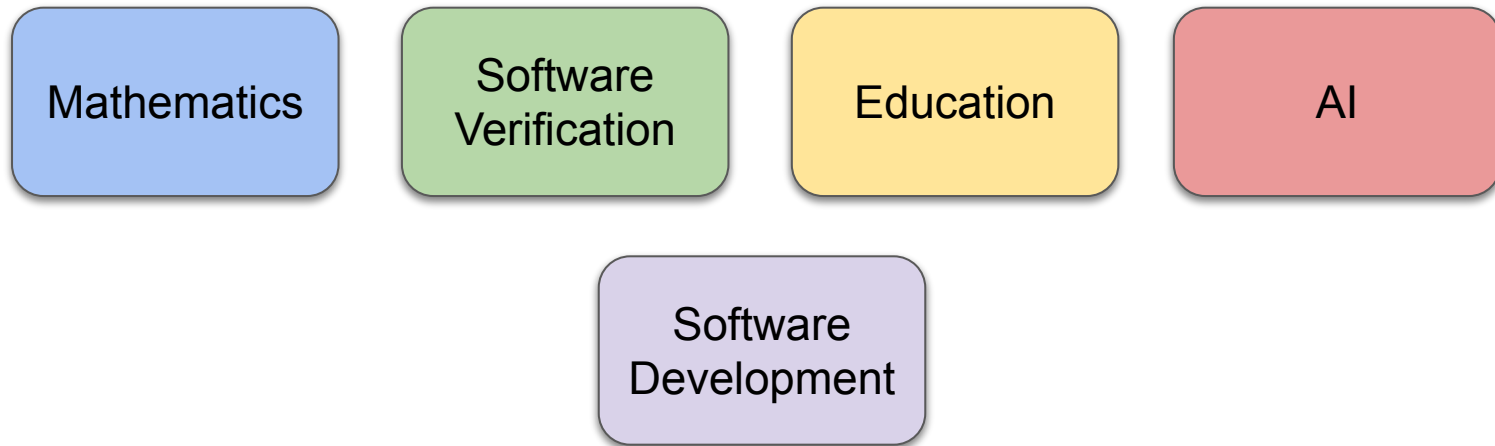
- Address **scalability**, **usability**, and **proof automation** in Lean.
- Support formal mathematics.
- Achieve self-sustainability in 5 years.

~7 FTEs by end of year

**Supported by Simons Foundation International, Alfred P. Sloan Foundation, and Richard Merkin**

[lean-fro.org](https://lean-fro.org)

# The Lean FRO serves the Lean community



# The Lean FRO: team



**Leo de Moura (AWS)**  
Chief Architect, Co-Founder



**Sebastian Ullrich**  
Head of Engineering, Co-Founder



**Joachim Breitner**  
Senior Research Software Engineer



**David Thrane Christiansen**  
Senior Research Software Engineer



**Joe Hendrix**  
Principal Research Software Engineer



**Marc Huisinga**  
Research Software Engineer



**Mac Malone**  
Research Software Engineer



**Scott Morrison**  
Senior Research Software Engineer

# The Lean FRO: team responsibilities

- **Leo** (Chief Architect): technical direction, management, backend.
- **Sebastian** (Head of Engineering): management, frontend.
- **Joachim Breitner**: backend.
- **David Christiansen**: documentation tools, documentation, outreach, frontend.
- **Joe Hendrix**: management, standard library, sledgehammer.
- **Marc Huisinga**: language server, VS Code plugin.
- **Mac Malone**: Lake, frontend.
- **Scott Morrison**: Mathlib & Lean, standard library, AI ambassador, proof automation.

**Backend** = kernel, compiler, proof automation.

**Frontend** = parser, macro system, elaborator.

# Development philosophy: ownership model

- Each file/model has a clear **owner** that is responsible for raising the bar.
  - Code is properly documented & tested.
  - Merging PRs that affect the module.
- Owner must be a Lean FRO employee.
- **Long-term goal**: transfer the ownership of many existing files.
- **Challenge**: we need more tests and benchmarks.

# RFC & Pull requests

- <https://github.com/leanprover/lean4/blob/master/doc/contributions.md>
- Before You Submit a Pull Request, Start with an Issue.
  - **User Experience:** How does this feature improve the user experience?
  - **Beneficiaries:** Which Lean users and projects do benefit most from this feature/change?
  - **Community Feedback:** Have you sought feedback or insights from other Lean users?
  - **Maintainability:** Will this change streamline code maintenance or simplify its structure?
- Quality over Quantity.
- Coding Standards.





# RFC & Pull requests


- **Reviews and Feedback:**
  - **Be Patient:** Given the limited number of full-time maintainers and the volume of PRs, reviews take some time.
  - **Engage Constructively:** Always approach feedback positively and constructively.
  - **Continuous Integration:** Ensure that all CI checks pass on your PR. Failed checks delay the review process.
- **What to Expect:**
  - **Not All PRs Get Merged:** While we appreciate every contribution, not all PRs will be merged.
  - **Feedback is a Gift:** It helps improve the project and can also help you grow as a developer.
  - **Community Involvement:** Engage with the Lean community on our communication channels. This can lead to better collaboration and understanding of the project's direction.


# From Achievements to Aspirations: A Glimpse of Our Journey and Horizon


# Issues will be addressed


 [leanprover / lean4](#) Public


 Notifications


 Fork 251


 Star 2.8k


 Code


 Issues 356


 Pull requests 66


 Discussions


 Actions

 Projects 6


 Security


 Insights

 Labels 51


 Milestones 3


New issue


 356 Open


 761 Closed


Author ▾ Label ▾ Projects ▾ Milestones ▾ Assignee ▾ Sort ▾


 **Simp regressions probably related to transparency** bug


#2670 opened 18 hours ago by PatrickMassot  2

 **simp unfolding let even with zeta := false option** bug


#2669 opened 18 hours ago by PatrickMassot  1 task done

 **simp [\*] regression** bug

#2668 opened 19 hours ago by PatrickMassot  1 task done

 **RFC: @[flat] annotation for names in the extend clause of a structure** RFC

#2666 opened yesterday by eric-wieser

 **RFC: lake update reorders the manifest** Lake RFC

#2664 opened yesterday by semorrison

# We know the Mathlib community desperately needs many issues fixed.

leanprover / lean4

<> Code Issues 355 Pull requests 68

Filters

is:issue is:open label:"Mathlib4 high prio"

Labels 51

Milestones 3

New issue

Clear current search query, filters, and sorts

☐ 7 Open

✓ 13 Closed

Author

Label

Projects

Milestones

Assignee

Sort

☐

RFC: autocompletion of imports


Lake

Mathlib4 high prio

RFC

server

#2655 opened 2 days ago by semorrison

 3


☐

RFC: change variable inclusion mechanism

Mathlib4 high prio

postponed

#2452 opened on Aug 24 by sgouezel


 2

☐

RFC: tweak structure instance elaboration to avoid un-needed eta expansion

Mathlib4 high prio

#2451 opened on Aug 24 by mattrobball


 1

☐

ridiculously long instance names

Mathlib4 high prio

#2343 opened on Jul 22 by kbuzzard 1 task done


 16

☐

elaboration of ^ shouldn't try to put the arguments in the same type

Mathlib4 high prio

#2220 opened on May 20 by fpvandoorn

 1 6


☐

Ignoring default value for field warning

bug

Mathlib4 high prio

#2178 opened on Apr 1 by gebner


 1

☐

Equation lemmas are longer generated by a simple match

Mathlib4 high prio

#2042 opened on Jan 18 by eric-wieser 1 task done

 1

# Lean releases

- Monthly stable releases with release candidates
  - Mathlib tracks release candidates (currently `v4.2.0-rc1`)
  - `nightly-testing` branch on Mathlib tracks Lean nightlies
- Lean PRs generate their own toolchain
  - `leanprover/lean4-pr-releases:pr-release-NNNN`
  - and automatically test Mathlib against this on a `lean-pr-testing-NNNN` branch
  - If your PR breaks Mathlib, please try to fix it, or ask for help!

We have our own servers now!!!

[speed.lean-lang.org](http://speed.lean-lang.org)

[loogle.lean-lang.org](http://loogle.lean-lang.org)

[live.lean-lang.org](http://live.lean-lang.org)

[reservoir.lean-lang.org](http://reservoir.lean-lang.org)

New domain name: [lean-lang.org](http://lean-lang.org)

# Loogle!

#find

## Try these

- `Real.sin`
- `Real.sin, tsum`
- `Real.sin (_ + 2*Real.pi)`
- `List.replicate (_ + _) _`
- `Real.sqrt ?a * Real.sqrt ?a`

## Documentation

This website gives access to mathlib's `#find` command:

The `#find` command finds definitions and lemmas in various ways. One can search by: the constants involved in the type; a substring of the name; a subexpression of the type; or a subexpression located in the return type or a hypothesis specifically. All of these search methods can be combined in a single query, comma-separated.

1. By constant:

```
#find Real.sin
```

finds all lemmas whose statement somehow mentions the sine function.

2. By lemma name substring:

```
#find "differ"
```



```

1 inductive Palindrome : List  $\alpha$   $\rightarrow$  Prop where
2   | nil      : Palindrome []
3   | single   : (a :  $\alpha$ )  $\rightarrow$  Palindrome [a]
4   | sandwich : (a :  $\alpha$ )  $\rightarrow$  Palindrome as  $\rightarrow$  Palindrome ([a] ++ as ++ [a])
5
6 theorem palindrome_reverse (h : Palindrome as) : Palindrome as.reverse := by
7   induction h with
8   | nil => exact Palindrome.nil
9   | single a => exact Palindrome.single a
10  | sandwich a h ih => simp; exact Palindrome.sandwich _ ih
11
12 theorem reverse_eq_of_palindrome (h : Palindrome as) : as.reverse = as := by
13   induction h with
14   | nil => rfl
15   | single a => rfl
16   | sandwich a h ih => simp [ih]
17
18 example (h : Palindrome as) : Palindrome as.reverse := by
19   simp [reverse_eq_of_palindrome h, h]
20
21 def List.last : (as : List  $\alpha$ )  $\rightarrow$  as  $\neq$  []  $\rightarrow$   $\alpha$ 
22   | [a], _ => a
23   | _::a::as, _ => (a::as).last (by simp)
24

```

▼ LeanProject.lean:16:22



▼ Tactic state



▼ **case** sandwich

```

αt : Type u_1
as ast : List αt
a : αt
h : Palindrome ast
ih : List.reverse ast = ast
⊢ List.reverse ([a] ++ ast ++ [a]) = [a] ++ ast ++ [a]

```

▼ Messages (1)

▼ LeanProject.lean:16:15



unused variable `h` [linter.unusedVariables]

► All Messages (1)





# lean4checker

Lean is a general programming language, with powerful metaprogramming.  
**You can use metaprogramming to corrupt the whole system.**

The new `lean4checker` tool allows easy reverification of files or libraries.

**In Mathlib CI now!**

**We must to port Lean 3 external checkers to Lean 4!**

# lean4checker

Lean is a general programming language, with powerful metaprogramming.  
**You can use metaprogramming to corrupt the whole system.**

The new `lean4checker` tool allows easy reverification of files or libraries.

**In Mathlib CI now!**

**We must to port Lean 3 external checkers to Lean 4!**

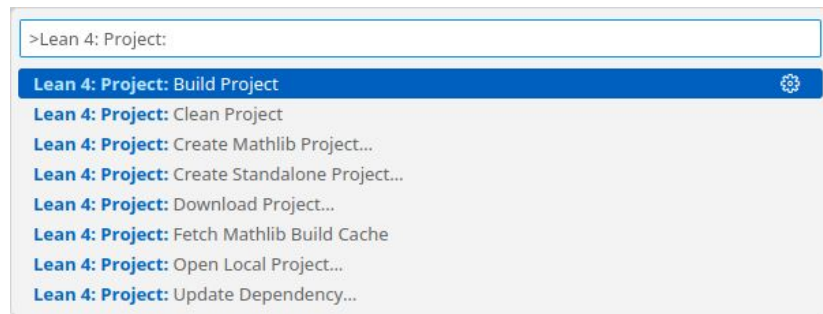
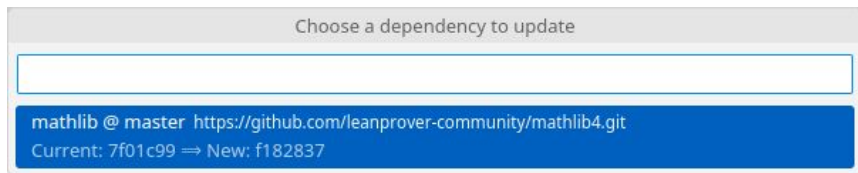
**Mario just released Lean4Lean last night!!!**

# VS Code Extension Progress - Project Commands

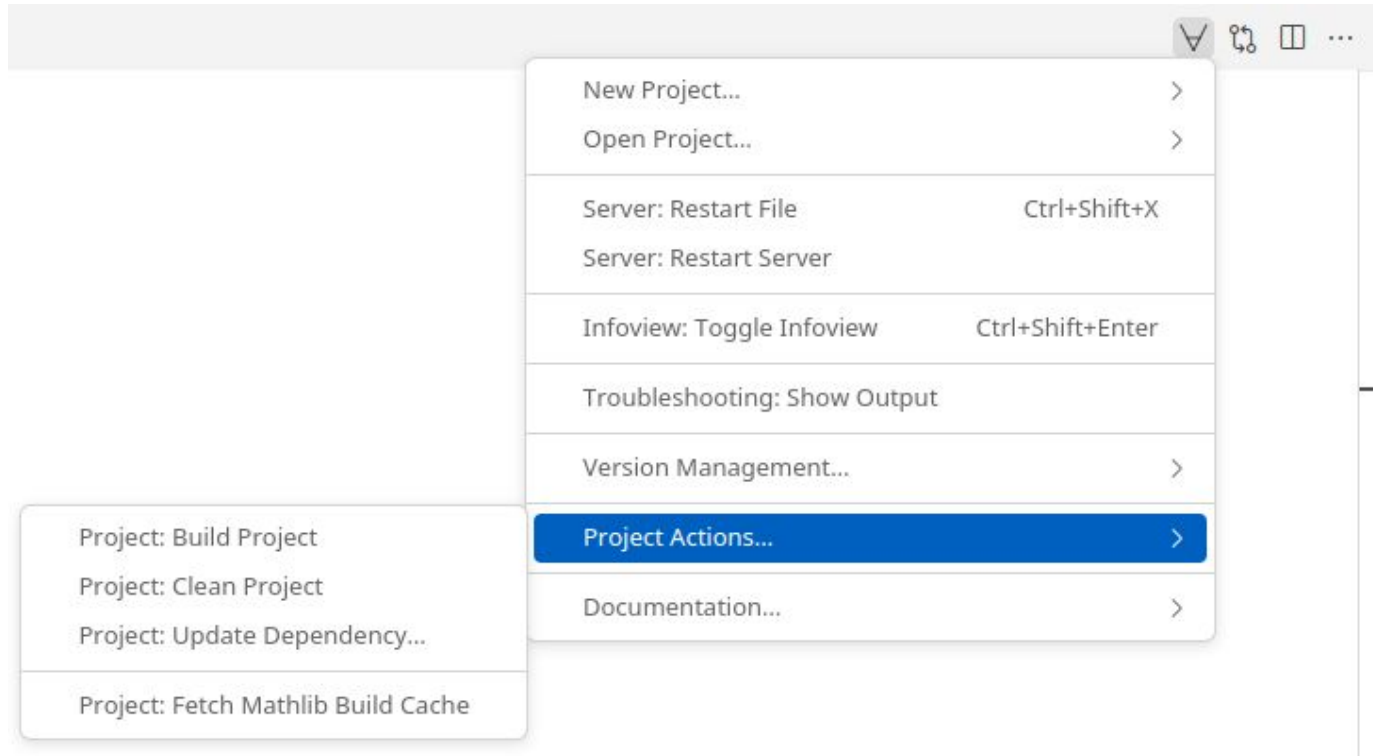
**Creating projects:** Standalone (``lake new``) & Mathlib (``lake new math``)

**Opening projects:** Local & Remote (via ``git clone``)

**Managing projects:** ``lake build``, ``lake clean``, ``lake exe cache get``  
and ``lake update <...>``



# VS Code Extension Progress - Command Menu



# VS Code Extension Progress - Walkthrough



## Lean 4 Setup

Getting started with Lean 4 on Linux

☒ Re-Open Setup Guide

☒ **Books and Documentation**

Learn using Lean 4 with the resources on the right.

☐ Install Required Dependencies

☐ Install Lean Version Manager

☐ Set Up Lean 4 Project

☐ Questions and Troubleshooting

☒ [Mark Done](#)

### Books

If you want to learn Lean 4, choose one of the following introductory books based on your background. If you are getting stuck or have any questions, click on the 'Questions and Troubleshooting' step at the bottom on the left side.

- [Functional Programming in Lean](#)  
The standard introduction for using Lean 4 as a general-purpose programming language.
- [The Mechanics of Proof](#)  
An introduction to Lean 4 as an interactive theorem prover for anyone who also wants to learn how to write rigorous mathematical proofs.
- [Mathematics in Lean](#)  
The standard introduction to Lean 4 as an interactive theorem prover for users with a mathematics background.
- [Theorem Proving in Lean 4](#)  
The standard reference for using Lean 4 as an interactive theorem prover. Suited as an introduction for users with a computer science background, advanced users and for general use as a reference manual.

Once you have completed one of these books and its exercises, you are ready to use Lean 4 for your own projects. If you want to use Lean 4 both as a general-purpose programming language and an interactive theorem prover, it is recommended to read both [Functional Programming in Lean](#) and [Theorem Proving in Lean 4](#).

### Hands-On Tutorial

If you want to dive right into using Lean 4 to prove elementary theorems about natural numbers, you can play the [Natural Number Game](#). It can be played online using your browser without a local installation.

### Additional Resources

#### Website

[Lean's website](#) links to learning resources, publications, talks and articles about Lean.

#### Lean Community

The [Lean Community website](#) links to several other helpful learning resources not listed here and provides an introduction to [mathlib](#), Lean's math library.

#### Manual

The [Lean Manual](#) documents several features of Lean 4 and can be consulted for some of the more technical details concerning Lean.

# VS Code Extension Progress - Walkthrough



## Lean 4 Setup

Getting started with Lean 4 on Linux

- ✓ Re-Open Setup Guide
- ✓ Books and Documentation
- ✓ Install Required Dependencies
- ✓ Install Lean Version Manager

### ✓ Set Up Lean 4 Project

Set up a Lean 4 project by clicking on one of the options on the right.

○ Questions and Troubleshooting

✓ Mark Done

## Project Creation

If you want to create a new project, click on one of the following:

- [Create a new standalone project](#)  
Standalone projects do not depend on any other Lean 4 projects. Dependencies can be added by modifying 'lakefile.lean' in the newly created project as described [here](#).
- [Create a new mathlib project](#)  
Mathlib projects depend on [mathlib](#), the math library of Lean 4.

If you want to open an existing project, click on one of the following:

- [Download an existing project](#)
- [Open an existing local project](#)

After creating or downloading a project, you can open it in the future by clicking the v-symbol in the top right, choosing 'Open Project...' > 'Open Local Project...' and selecting the project you created.

## Complex Project Setups

Using its build system and package manager Lake, Lean 4 supports more complex project setups than the ones described above. You can find out more about Lake in the [Lean 4 GitHub repository](#).

# VS Code Extension Progress - Other

- Lean client startup progress bar  
(no confusion while waiting for yellow bars to appear!)
- Better command output for troubleshooting purposes
- Cancellable external commands
- Opt-in for automatically building dependencies when opening files

**Full list with more details on Zulip after the community meeting.**

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
/home/mhuisi/Projects/NewMathlibProject> lake exe cache get
No files to download
Decompressing 3815 file(s)
unpacked in 14511 ms
/home/mhuisi/Projects/NewMathlibProject> lake build
[1/3] Building NewMathlibProject.Basic
[2/3] Building NewMathlibProject
```



Basic.lean 1, U x

```
1 import NewMathlibProject.Foo
```

```
2 Imports are out of date and must be rebuilt; use the "Restart File" command in your editor.
```

# Plans - VS Code Extension

## Short term plans:

- Fix race conditions in abbreviations that lead to inconsistent behavior
- Detect possibly outdated or broken setups and inform users about it

## Rough long term plans:

- Add multi-toolchain single-folder workspace support
- Re-evaluate Live Share support after Sebastian's elaboration rework
- Use updated Lean 4 extension syntax highlighting on GitHub



# Plans - Language Server

## Short term plans:

- Add **import auto-completion**
- Improve find-references & auto-completion
- Implement call hierarchy

## Rough long term plans:

- Integrate more **code actions** into core
- Make **language server more extensible**
- Improve tactic auto-completion
- Improve **trace browsing**

# Lake

- There have been a number of fixes, touch-ups, feature additions, and **performance improvements** to Lake since the launch of the FRO
- Some highlights:
  - ``lake update <pkg>`` and improved ``lake update`` error messages
  - Overhauled ``lake env`` that works without a lake configuration
  - Better ``lake new/init`` that provides a multi-file directory structure for libraries, fixes some previous bugs in naming, and uses the elan toolchain for the Lean version
  - Cache output file hashes and the elaborated configuration to improve performance
  - Lake is now part of doc-gen4's core documentation

# Lake Startup Time

A number of performance improvements landed for Lake since Lean v4.0.0. Cumulatively, these make for a **~4-6x** speedup in Lake startup time for mathlib.

macOS

```
$ hyperfine "lake-v4 build cache" "lake-nightly build cache" -w3 # no-op build
Benchmark 1: lake-v4 build cache
  Time (mean ± σ):      963.9 ms ±   3.7 ms    [User: 762.0 ms, System: 271.0 ms]
  Range (min ... max):  957.4 ms ... 967.3 ms    10 runs

Benchmark 2: lake-nightly build cache
  Time (mean ± σ):      158.7 ms ±   3.3 ms    [User: 94.1 ms, System: 183.2 ms]
  Range (min ... max):  155.6 ms ... 168.4 ms    18 runs

Summary
'lake-nightly build cache' ran
  6.07 ± 0.13 times faster than 'lake-v4 build cache'
```



# Lake Startup Time

A number of performance improvements landed for Lake since Lean v4.0.0. Cumulatively, these make for a **~4-6x** speedup in Lake startup time for mathlib.

Windows

```
$ hyperfine "lake-v4 build cache" "lake-nightly build cache" -w3 # no-op build
```

## Benchmark 1: lake-v4 build cache

```
Time (mean ± σ):      3.113 s ± 0.034 s    [User: 2.263 s, System: 0.865 s]
```

```
Range (min ... max):   3.063 s ... 3.156 s    10 runs
```

## Benchmark 2: lake-nightly build cache

```
Time (mean ± σ):      714.6 ms ± 12.3 ms   [User: 300.6 ms, System: 428.8 ms]
```

```
Range (min ... max):   697.8 ms ... 734.2 ms    10 runs
```

## Summary

```
'lake-nightly build cache' ran
```

```
4.36 ± 0.09 times faster than 'lake-v4 build cache'
```



# Cache Get Speed

For `lake exe cache get`, this translates to an overall **~2x** no-op speedup, since the `cache` program itself has not had as many performance improvements.

macOS

```
$ hyperfine "lake-v4 exe cache get" "lake-nightly exe cache get" -w3 # no-op get
```

## Benchmark 1: lake-v4 exe cache get

```
Time (mean ± σ):      1.447 s ± 0.024 s    [User: 1.121 s, System: 1.259 s]
Range (min ... max):  1.420 s ... 1.502 s    10 runs
```

## Benchmark 2: lake-nightly exe cache get

```
Time (mean ± σ):      638.5 ms ± 12.2 ms   [User: 454.8 ms, System: 1.128 s]
Range (min ... max):  622.4 ms ... 663.5 ms 10 runs
```

## Summary

```
'lake-nightly exe cache' get ran
  2.27 ± 0.06 times faster than 'lake-v4 exe cache get'
```



# Cache Get Speed

For `lake exe cache get`, this translates to an overall **~2x** no-op speedup, since the `cache` program itself has not had as many performance improvements.

Windows

```
$ hyperfine "lake-v4 exe cache get" "lake-nightly exe cache get" -w3 # no-op get
```

**Benchmark 1: lake-v4 exe cache get**

Time (mean  $\pm$   $\sigma$ ): **5.121 s**  $\pm$  0.127 s [User: 3.416 s, System: 2.401 s]

Range (min ... max): 5.008 s ... 5.402 s 10 runs

**Benchmark 2: lake-nightly exe cache get**

Time (mean  $\pm$   $\sigma$ ): **2.682 s**  $\pm$  0.016 s [User: 1.455 s, System: 1.890 s]

Range (min ... max): 2.662 s ... 2.709 s 10 runs

## Summary

'lake-nightly lake exe cache get' ran

**1.91  $\pm$  0.05** times faster than 'lake-v4 exe cache get'



# Cloud Builds for All

- Our goal is to **provide a general service for caching cloud builds** that can meet the needs of both mathlib and other packages within the ecosystem (e.g., Kevin's FLT project, LeanInfer, FFIs, etc.)
- To do this effectively, Lean & Lake need a package index that can store metadata about packages and provide a online portal from which to distribute cloud builds.
- Thus, for this reason and others, we sought to create a package repository for Lean and Lake.

# Reservoir

- [crates.io](https://crates.io) for Lean/Lake
  - for LaTeX lovers, [CTAN](https://ctan.org)
- searchable package index
- includes a ecosystem-wide testbed which will build, test, and check the compatibility of popular packages with the latest Lean toolchain(s)

The screenshot shows the Reservoir website, which is Lake's package repository. The header features the Reservoir logo and the text "Lake's package repository". A search bar with the placeholder text "Press 'S' to start searching..." is located below the header. On the right side of the header, there is a link to "All Packages" and a menu icon. The main content area is divided into three columns: "Most Popular", "Newly Created", and "Recently Updated". Each column contains a list of packages with their status (indicated by a green checkmark or a red X) and a link to view the package details. A button labeled "Get Started with Lean" is located in the top right corner of the main content area.

**Reservoir** All Packages

## Lake's package repository

Press 'S' to start searching...

**Latest Lean Toolchain:**  
leanprover/lean4:v4.2.0-rc1

[Get Started with Lean](#)

Reservoir indexes, builds, and tests packages within the Lean and Lake ecosystem.

### Most Popular

- mathlib4
- SciLean
- lean4-metaprogramming-book
- std4
- lean4-raytracer
- aesop

### Newly Created

- LeanGccBackend
- lean-codespace
- lean4checker
- mathlib4\_with\_LeanInfer
- regensburg-itp-school-2023
- ProjectiveSpace\_lean4

### Recently Updated

- CS22-Lean-2023
- NNG4
- HausdorffSchoolLean
- rinha
- mathlib4
- lean4-unicode-basic



# Package Pages

- as development progresses, individual package pages will provide more Lean/Lake-specific information than GitHub can provide
- syntax highlighting is also further tunable for Lean
- if you have any suggestions on what you would like to see here, please share them!

The screenshot shows the 'mathlib4' package page on the Reservoir platform. The header includes the Reservoir logo, a search bar with the placeholder 'Press 'S' to start searching...', and a link to 'All Packages'. The package name 'mathlib4' is prominently displayed, followed by the description 'The math library of Lean 4'. A 'Readme' tab is active. The main content area features the package name 'mathlib4' again, followed by a row of status badges: 'continuous integration passing', 'continuous integration passing', 'docs enabled', and 'zulip join chat'. The text describes it as a complete port of mathlib to Lean 4. A sidebar on the right provides additional details: 'Apache 2.0' license, '6 days ago' update, '345 stars', 'Lean' category, 'v4.2.0-rc1' version, 'Homepage' link to 'leanprover-community.git...', and 'Repository' link to 'leanprover-community/ma...'. The main content area also has sections for 'Installation', 'Experimenting', and 'Documentation'.

**Reservoir** Press 'S' to start searching... All Packages

**mathlib4**  
The math library of Lean 4

Readme

**mathlib4**

continuous integration passing continuous integration passing docs enabled zulip join chat

This is a complete port of [mathlib](#) to [Lean 4](#). Development of mathlib now takes place in this repository.

[Mathlib](#) is a user maintained library for the [Lean theorem prover](#). It contains both programming infrastructure and mathematics, as well as tactics that use the former and allow to develop the latter.

**Installation**

You can find detailed instructions to install Lean, mathlib, and supporting tools on [our website](#).

**Experimenting**

Got everything installed? Why not start with the [tutorial project](#)?

For more pointers, see [Learning Lean](#).

**Documentation**

Apache 2.0  
6 days ago  
345 stars

**Lean**

v4.2.0-rc1

**Homepage**

[leanprover-community.git...](#)

**Repository**

[leanprover-community/ma...](#)

# Some Near-Term Plans

## Reservoir

- Store testbed builds in Reservoir cloud storage and fetch them with Lake
- Add data on a package's dependency tree to Reservoir
- Test packages across multiple toolchains, run external checkers

## Lake

- Make more package configuration data available in the Lake manifest
- Intelligent toolchain versioning on ``lake update`` ([lake#180](#))
- Proper support for C FFIs which depend on the C++ stdlib
- Support for shared external libraries

# Elaboration Performance Plans

**Done:** significant improvements to reduction, defeq, server performance

**Short-term:** incremental execution of tactic blocks

**Mid-term:** file-level parallelism, of theorems and much more

**Long-term:** module system, to define abstract interfaces of files and cut down on rebuilds

# Parser Plans

```
#lang Lean.Language.Programming
```

```
#lang Lean.Language.Math
```

```
#lang Lean.Doc.Manual
```

- **Extensibility:** new dialects.
- **Education:** use simplified subset.
- **DSLs:** domain specific languages.
- **Modularity:** allows you explicitly state the subset/dialect.

Reserved tokens per syntactic category.

# Documentation Tooling - Plans and Dreams

- No documentation tool works well for everyone - bloggers, textbook authors, paper authors, and package doc writers have different but related needs
- **Plan: documentation as DSL in Lean**
  - Library of re-usable documentation components (code samples, embedded maths, section headers, figures, ...)
  - Export to single HTML page, multi-page online book, epub, various journals' LaTeX, etc
  - Doc components in Lean packages like any other
  - Convenient syntax, mostly-Markdown-compatible

```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

There are two recommended ways to get Lean:

- \* Install the [\[VS Code extension\]\(https://...\)](https://...) and then open a Lean file
- \* [\[Install 'elan'\]\(https://...\)](https://...), the commandline Lean toolchain management tool

```
# Using Lean
```

```
## Defining Functions
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
  def even : Nat → Bool
  | 0 => true
  | n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
  def even (n : Nat) : Bool :=
    match n with
    | 0 => true
    | n' + 1 => not (even n')
```

- Use `#lang` to select the right syntax for the file on the first line and import the right libraries
- Lean provides highlighting to the editor, and all the usual features still work
- This code example library allows provisional examples that aren't in scope in subsequent examples

```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

```
  Lean.Doc.Header.Section
```

```
Th
* A section header. Section headers
* are rendered relative to the
  current document's placement in
  the final document, and will
# accordingly end up as chapters,
  sections, or subsections.
```

```
## Defining Functions
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
  def even : Nat → Bool
  | 0 => true
  | n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
  def even (n : Nat) : Bool :=
    match n with
    | 0 => true
    | n' + 1 => not (even n')
```

get Lean:  
(<https://...>) and then open a Lean file  
the commandline Lean toolchain management tool

Hovers will work as usual for custom syntax

```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

There are two recommended ways to get Lean:

 [Install the \[VS Code extension\]\(https://...\)](https://...) and then open a Lean file  
 ..., the commandline Lean toolchain management tool

```
# Use
## [
  Go to Definition
  Go to Declaration
  Go to Type Declaration
  Go to References
```

```
## [
  Change to Numbered List
  Change to Description List
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
def even : Nat → Bool
| 0 => true
| n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
def even (n : Nat) : Bool :=
  match n with
  | 0 => true
  | n' + 1 => not (even n')
```

Right-click menu

Code actions for changing between list styles



```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

There are two recommended ways to get Lean:

- \* Install the [\[VS Code extension\]\(https://...\)](https://...) and then open a Lean file
- \* [\[Install 'elan'\]\(https://...\)](https://...), the commandline Lean toolchain management tool

```
# Using Lean
```

```
## Defining Functions
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
```

```
def even : Nat → Bool
| 0 => true
| n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
```

```
def even (n : Nat) : Bool :=
  match n with
  | 0 => true
  | n' + 1 => not (even n')
```

```
▼LeanManual.lean:28:22
▼Expected type
  n : Nat
  ⊢ Bool
```

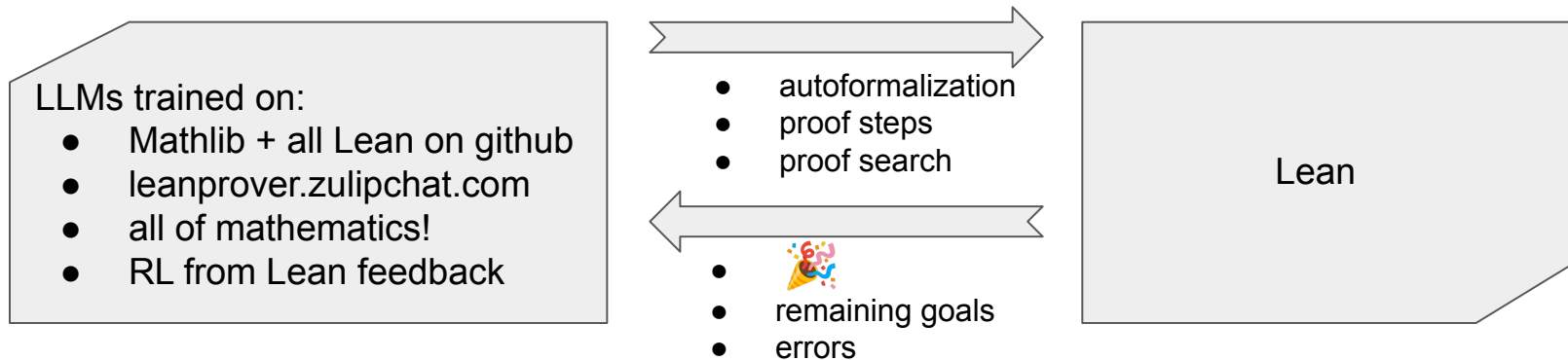
This code examples library invokes the elaborator in the usual way, so all interactive editing features can work as usual in the embedded examples

# Documentation Tooling - Plans and Dreams

- Other goals:
  - Integrate nicely with docstrings, doc-gen4, etc
  - Support all existing documentation - port FPiL and the manual ASAP
  - Integrate with playgrounds, editors, and other tools
- Current status: design sketches, feasibility studies
- Please send your feedback and interesting use cases to [david@lean-fro.org](mailto:david@lean-fro.org) or say "hi" on Zulip

# ML and Lean

**Lean is an ideal framework for developing ML reasoning**



# Machine Learning and Lean

- We want tactic suggestion tools (**LeanInfer**, **LLMStep**, ...) for all users.
- Smoothing the path for outside ML groups to “solve math” using Lean.
- We’d like to see **reusable ML components** for
  - Premise selection
  - Tactic suggestion
  - Proof search
  - Auto-formalization
- New features in the **Lean REPL targeting ML users**:
  - experimental tactic mode
  - pickling REPL states for distributed use
- **Data extraction tools at [lean-training-data](#).**

# What about?

- **Compiler:** more people joining soon.
- **Sledgehammer for Lean:** Yes, it is on our roadmap.
- **Efficient kernel reduction:** Joachim is starting next month ;)

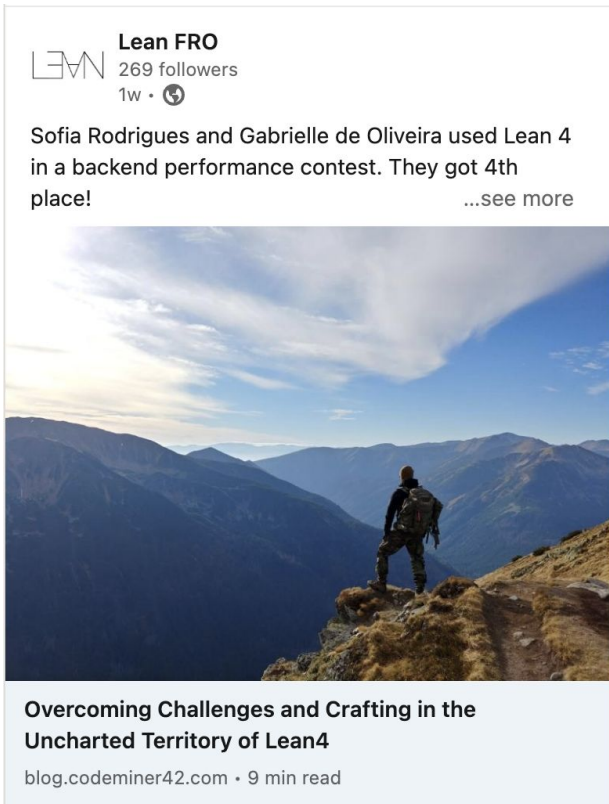
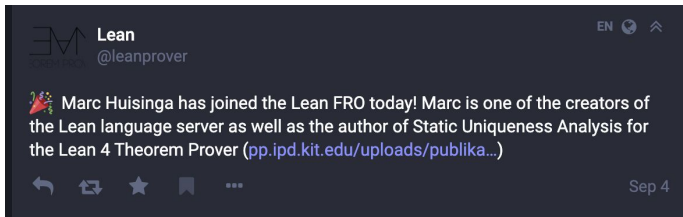
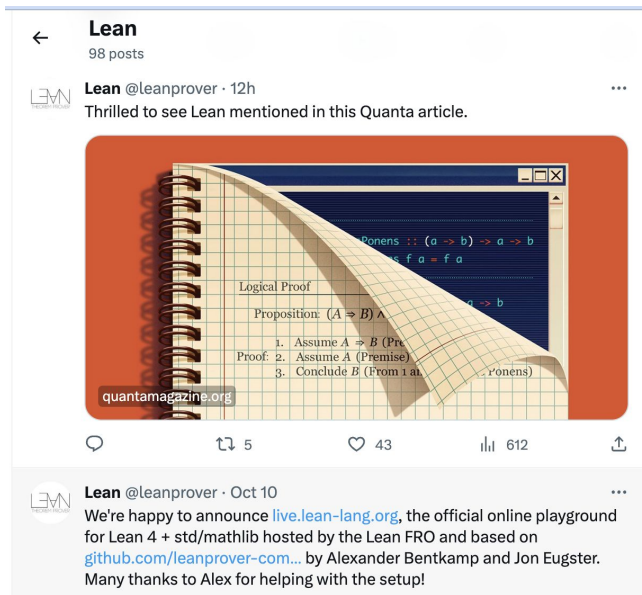
# Social media

Follow us @

Twitter

LinkedIn

Mastodon



Q & A

## Extra slides - begin here

- The following slides can be used in future dive deeps & demos



# Next steps...

## TODO:

- Cloud build.
- Reservoir.
- Incremental tactics.
- Extensible LSP server & code actions.
- ...

```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

There are two recommended ways to get Lean:

- \* Install the [\[VS Code extension\]\(https://...\)](https://...) and then open a Lean file
- \* [\[Install 'elan'\]\(https://...\)](https://...), the commandline Lean toolchain management tool

```
# Using Lean
```

```
## Defining Functions
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
  def even : Nat → Bool
  | 0 => true
  | n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
  def even (n : Nat) : Bool :=
    match n with
    | 0 => true
    | n' + 1 => not (even n')
```

Breadcrumbs know about the document's structure

```
#lang Lean.Doc.Manual
```

```
@import Lean.Doc.ExampleCode
```

```
@about
```

```
  title: The Lean Manual
```

```
  author:
```

- One person
- You too?

```
# Downloading and Installing Lean
```

There are two recommended ways to get Lean:

- \* Install the [VS Code extension](https://...) and then open a Lean file
- \* [Install 'elan'](https://...), the commandline Lean toolchain management tool

```
# Using Lean
```

```
## Defining Functions
```

```
@open Lean.Doc.Examples
```

```
@example_context defuns
```

A function definition looks like this:

```
@example defuns (keep := false)
  def even : Nat → Bool
  | 0 => true
  | n + 1 => not (even n)
```

It could have also been written like this:

```
@example defuns (keep := true)
  def even (n : Nat) : Bool :=
    match n with
    | 0 => true
    | n' + 1 => not (even n')
```

Highlighting all the bullet points for the current list

# VS Code Extension Progress - New Project Commands

**Creating projects:** Standalone (``lake new``) & Mathlib (``lake new math``)

- Mathlib projects use Mathlib's toolchain and are initialized with ``lake exe cache get``

**Opening projects:** Local & Remote (via ``git clone``)

- Opening local Lean projects or using “Open Folder” performs sanity checks
- Opening and downloading a remote project uses ``lake exe cache get`` if available

# VS Code Extension Progress - New Lake Commands

**Managing projects:** ``lake build``, ``lake clean``, ``lake exe cache get`` and ``lake update <...>``

- ``lake exe cache get`` is used whenever it makes sense to do so
- ``lake update <...>`` command updates toolchain version if desired

