

# Master Your Local Development Environment for AI adoption

YEONG SHENG



STANLY



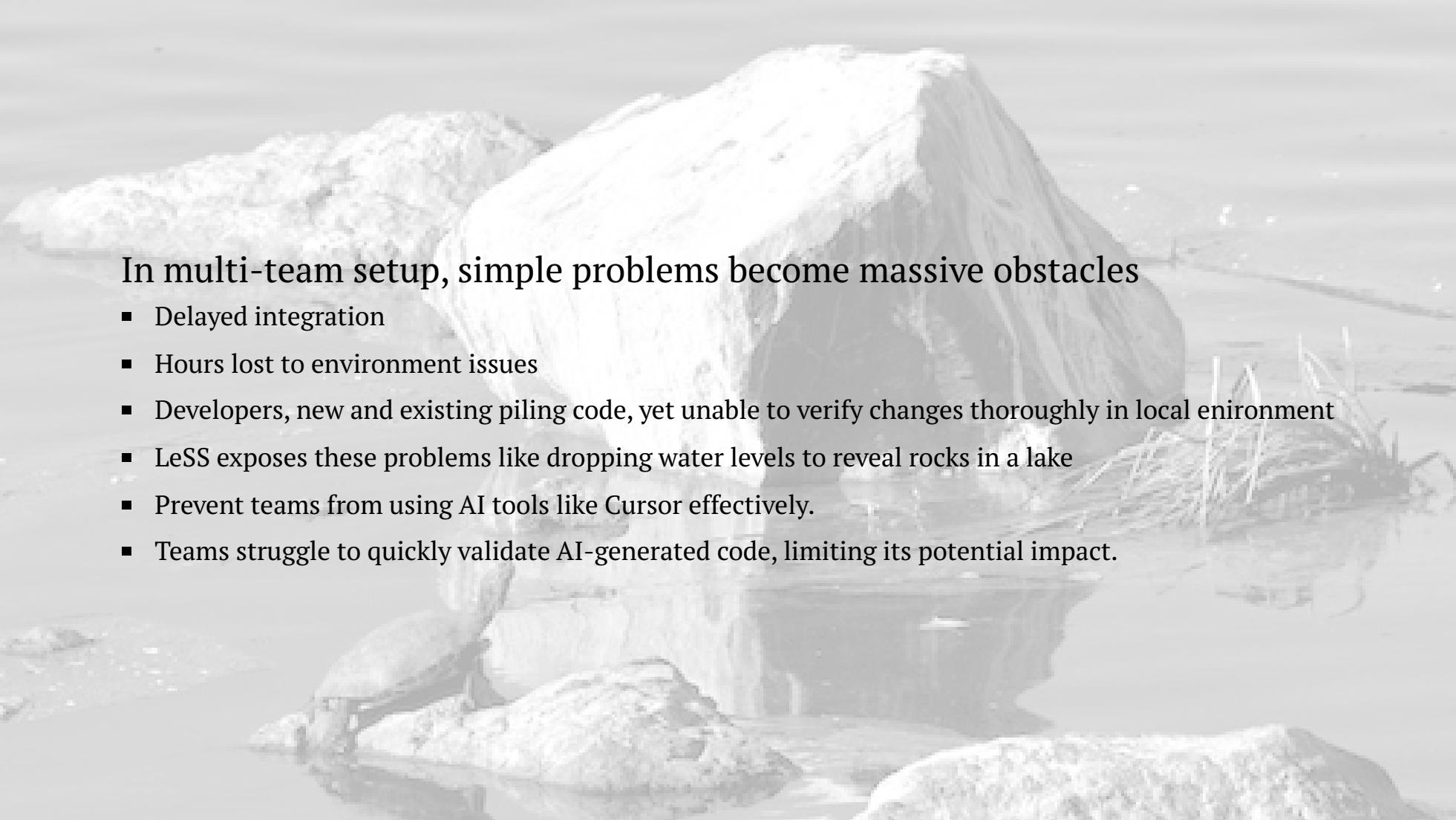
LeSS

The Lake and Rocks Metaphor (LeSS - Queueing Theory)

Press Space for next page →

"Works on my machine"

Isn't funny when it blocks the entire team/s



## In multi-team setup, simple problems become massive obstacles

- Delayed integration
- Hours lost to environment issues
- Developers, new and existing piling code, yet unable to verify changes thoroughly in local environment
- LeSS exposes these problems like dropping water levels to reveal rocks in a lake
- Prevent teams from using AI tools like Cursor effectively.
- Teams struggle to quickly validate AI-generated code, limiting its potential impact.

Unlock the full potential of AI integration in your development workflow.  
Achieve a tight, full-cycle feedback loop, with rapid iteration.

# Demo Repository



[lunchbox\\_api](#)

lunchbox\_api

EXPLOR... ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

food\_controller\_test.exs

```
test > lunchbox_api_web > controllers > food_controller_test.exs
  1 defmodule LunchboxApiWeb.FoodControllerTest do
  2   import Plug.Conn
  3   import Phoenix.ConnTest
  4   import Plug.BasicAuth
  5   use LunchboxApiWeb.ConnCase
  6
  7   alias LunchboxApi.Lunchbox
  8   alias LunchboxApi.Lunchbox.Food
  9
 10  @create attrs %{
 11    name: "some name",
 12    status: "some status"
 13  }
 14  @update attrs %{
 15    name: "some updated name",
 16    status: "Some updated status"
 17  }
 18  @invalid attrs %{name: nil, status: nil}
```

run full unit test to verify the current state of my dev env with  
`BASIC\_AUTH\_USERNAME=specialUserName BASIC\_AUTH\_PASSWORD=superSecretPassword mix test`. Do not proceed any further if it fails to run at first.

Kiro

I'll run the full unit test suite with the specified environment variables to verify your dev environment state.

⚠ Command

```
BASIC_AUTH_USERNAME=specialUserName BASIC_AUTH_PASSWORD=superSecretPassword mix test
```

zsh: command not found: mix

The `mix` command isn't found in your environment. This suggests Elixir isn't properly set up in your current shell session. Let me check if devbox is being used to manage the environment:

run full unit test t... x run full unit test t... x New Session x

EXPLOR... ⌂ ⌂ ⌂ ⌂ ⌂ ⌂

food\_controller\_test.exs

```
test > lunchbox_api_web > controllers > food_controller_test.exs
  1 defmodule LunchboxApiWeb.FoodControllerTest do
  2   import Plug.Conn
  3   import Phoenix.ConnTest
  4   import Plug.BasicAuth
  5   use LunchboxApiWeb.ConnCase
  6
  7   alias LunchboxApi.Lunchbox
  8   alias LunchboxApi.Lunchbox.Food
  9
 10  @create attrs %{
 11    | name: "some name",
 12    | status: "some status"
 13  }
 14  @update attrs %{
 15    | name: "some updated name",
 16    | status: "Some updated status"
 17  }
 18  @invalid attrs %{name: nil, status: nil}
 19
 20  # get auth username
 21  @username System.get_env("BASIC_AUTH_USERNAME")
 22  @password System.get_env("BASIC_AUTH_PASSWORD")
 23
 24  # setup auth on conn
 25  setup %{conn: _conn} do
 26    | conn =
```

run full unit test to verify the current state of my dev env with  
`BASIC\_AUTH\_USERNAME=specialUserName BASIC\_AUTH\_PASSWORD=superSecretPassword mix test`. Do not proceed any further if it fails to run at first.

Kiro

I'll run the full unit test suite with the specified environment variables to verify your dev environment state.

Command

```
BASIC_AUTH_USERNAME=specialUserName BASIC_AUTH_PASSWORD=superSecretPassword mix test
```

Running ExUnit with seed: 187454, max\_cases: 24

.....

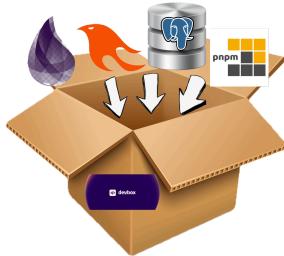
Finished in 0.1 seconds (0.05s async, 0.09s sync)  
16 tests, 0 failures

Perfect! Your test suite is running successfully. All 16 tests passed with no failures, which means your development environment is in a good state. The tests completed in 0.1 seconds, indicating everything is properly configured and working as expected.

Is it even possible to take complete control of your local environment?

- databases, messaging, and dependencies?
- repeatable, composable, clean-slate DB migrations?
- time-travel testing, and bending your software product system to your will for effective testing & verification?

# { DEMO }



```
{
  "packages": ["postgresql@17.5", "beam27Packages.elixir@1.18.4", "nodejs@22.14.0"],
  "shell": {
    "init_hook": ["cd assets && pnpm --frozen-lockfile recursive install"],
    "scripts": {
      "test": ["BASIC_AUTH_USERNAME=specialUserName BASIC_AUTH_PASSWORD=superSecretPassword mix test"],
      "init-db": [
        "export PGDATA=$PWD/.devbox/virtenv/postgresql/data",
        "rm -rf $PWD/.devbox/virtenv/postgresql",
        "mkdir -p $PGDATA",
        "initdb --username=postgres -D $PGDATA"
      ]
    }
  }
}
```

# Key Outcomes & Takeaways

## The Problem

- Integration delays ripple across teams, creating massive obstacles in multi-team setups
- Hours lost to "works on my machine" issues that block entire development workflows
- AI tools become unreliable without proper local verification capabilities
- Blocked productivity affects both human developers and AI-assisted development
- AI-generated code is only as valuable as your ability to test it locally
- Without a clean, reproducible environment, you can't trust AI suggestions

# Outcome

- **Recognize the true cost** of broken local environments - it's expensive and blocks both human and AI productivity
- **Master your local development environment** - service dependencies, & runtime for instant code verification
- **Unlock AI tool potential** - use tools like Cursor productively with reliable local verification to validate generated code
- **Build improvement roadmaps** - guide your teams toward better integration practices and effective AI adoption

## The Bottom Line

- AI-generated code is only as valuable as your ability to test it locally
- Without a clean, reproducible environment, you can't trust AI suggestions
- Fast feedback loops are essential for both human and AI-assisted development success

# Learn More

Lunchbox API Demo Code · Nix · Jetify Devbox

Powered by  Sliddev