Apex Is Slower Than It Should Be

Salesforce makes fast developers wait. Every deployment interrupts your flow. Every test suite adds friction. Every debug cycle pulls you away from the problem you're trying to solve.

Deploys interrupt flow

Push to org, wait, refresh, repeat. The mental context switch kills momentum.

Test suites take hours instead of seconds

What should be instant becomes a waiting game. Iteration slows to a crawl.

Debugging means reading logs

No breakpoints. No inspection. Just text files and educated guesses.

The language isn't the problem. The environment is. Modern developers deserve modern feedback loops, not architectural compromises from a different era.

NEW PARADIGM

Local Changes Everything

What if Apex ran on your machine?

Not a simulator. Not a sandbox. Your code, your machine, your speed. The same language you know, executing where you work—no network latency, no deployment ceremony, no artificial barriers between thought and validation.

Local execution transforms the developer experience. Write a test, run it instantly. Change a line, verify immediately. Debug with real tools, not log archaeology. It's the workflow every modern language takes for granted.

01

No org dependency

Work offline. Own your environment.

02

No deployment delays

Zero wait between code and execution.

03

Same language, radically faster feedback

Apex as it should be.

Meet aer

Apex Execution Runtime

aer is a local Apex runtime built for speed, clarity, and modern development workflows. It runs your Apex code on your machine—tests, classes, triggers—without touching an org. No deployment. No waiting. No compromise on fidelity.

Run Apex locally

Execute production code on your development machine with zero network overhead.

Execute tests without deployment

Validation in milliseconds, not minutes. Run your entire test suite as fast as you can type.

Faithful to Salesforce behavior

Schema-aware, trigger-respecting, governor-limit-enforcing. Real Apex, just faster.

This isn't a learning tool or a toy. It's production-grade infrastructure for teams who take Apex seriously.

Speed That Changes Habits

Milliseconds change how developers work. When tests run instantly, you run them constantly. When feedback is immediate, you write smaller, better tests. When iteration is frictionless, quality becomes reflexive.

1

Run tests constantly

Not before commit. Not in CI. During development. Every change, every save, every thought.

2

Write smaller, better tests

Fast feedback encourages focused tests. Focused tests catch bugs earlier and document intent better.

3

Shorten feedback loops everywhere

From idea to validation in under a second. The cognitive difference is profound.

> When your test suite completes before you can context-switch, testing stops being a chore and becomes part of thinking.

DEVELOPER EXPERIENCE

Debug Apex Like a Real Language

Breakpoints. Variables. Call stacks.

Set a breakpoint in VS Code. Run your test. Step through execution line by line. Inspect variables as they change. Watch the call stack build and collapse. See exactly what's happening, exactly when it happens.

No log statements. No deploy-and-pray. No guessing what the value was three lines ago. Just actual, interactive debugging with the tools modern IDEs have perfected.

Step through Apex in VS Code

Line-by-line execution control

Inspect live state

Watch variables, not logs

Fix bugs without guesswork

See the problem, understand it, solve it

First-Class Language Tooling

Apex deserves the same IDE support as TypeScript, Go, or Rust. Go-to-definition that actually works. Find-all-references across your codebase. Autocomplete that understands your schema. Refactoring that's safe because the tools understand the language.

Autocomplete and navigation

Intelligent suggestions based on actual type information and context.

Go-to-definition and references

Jump to any symbol, find every usage, understand connections instantly.

Consistent analysis across projects

Same tooling, same results, whether you're in a scratch org or production metadata.

aer brings language-server-protocol capabilities to Apex. Not as a nice-to-have, but as the foundation of how you work.

Real Salesforce Semantics

Local doesn't mean fake.

aer understands your org's metadata. It parses your schema, respects field-level security, honors validation rules, and executes triggers in the correct order. SOQL queries return actual results based on your data model. DML operations follow governor limits. Managed packages work as expected.

Schema-aware execution

Custom objects, fields, relationships—aer knows your data model and enforces it.

Managed package support

Dependencies, namespaces, and package visibility all work correctly.

SOQL, DML, triggers, callouts

The full Apex surface area, executed with production fidelity.

This isn't an approximation. It's a faithful implementation of Salesforce semantics, running locally for speed without sacrificing correctness.

When you deploy to an org, the behavior matches. When you write tests in aer, they pass in production. Trust is earned through precision.

ENTERPRISE READY

Built for Teams and CI

From laptop to pipeline. aer works the same way everywhere—your machine, your teammate's machine, your CI runner. No special configuration. No org provisioning. No flaky tests because someone else is using the sandbox.

Same commands everywhere

One interface for local dev, CI, and automation. No environment-specific scripts.

Deterministic test runs

No shared state. No race conditions. Tests pass or fail for the right reasons.

Faster CI without org contention

Run parallel test suites without fighting over sandbox capacity or rate limits.

Teams ship faster when infrastructure stops being the bottleneck. aer removes the dependency on remote execution, giving you back control over your development pipeline.

More Than a Tool

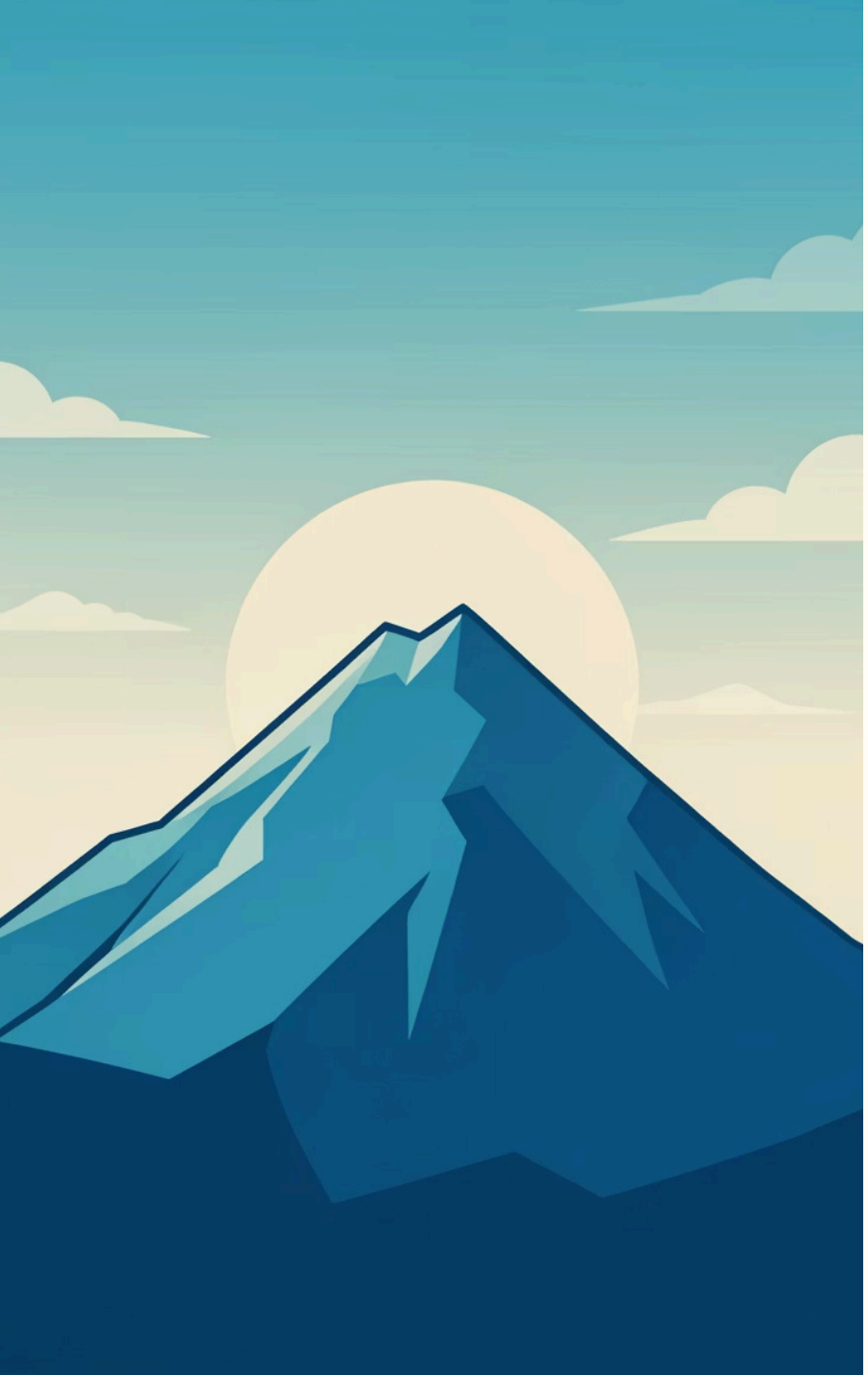
A platform for forward-thinking Apex developers.

aer is opinionated where Salesforce isn't. It encourages testability, modularity, and clarity. It's designed for real codebases with hundreds of classes, complex dependencies, and high standards. It's built by people who write Apex every day and felt every limitation.

This isn't about making Apex easier. It's about making it better—faster feedback, stronger tooling, clearer debugging, and a development experience that matches the sophistication of the problems you're solving.

- Opinionated where Salesforce isn't
- Designed for real codebases
- Built by people who live in Apex

If you care about developer experience, aer is for you. If you're tired of waiting, aer is for you. If you believe Apex can be more, aer is for you.



Join Early

If you write Apex, you should try aer.

01

Start locally

Install aer. Run your first test in milliseconds. Feel the difference immediately.

02

Bring it to your team

Share the speed. Standardize on better tooling. Watch productivity compound.

03

Help shape what Apex becomes next

Early adopters define the future. Your feedback makes aer better for everyone.

This is the beginning of local-first Apex development. Join the developers who refuse to wait, who demand better tools, and who believe Apex deserves a modern runtime.

Get Started with aer

Read the Documentation