# Perstack: Expert Stack for Agent-first Development


📚 Documentation • 🚀 Getting Started • 𝕏 Twitter

## Overview

Perstack is a package manager and runtime for agent-first development. Define modular micro-agents as Experts in TOML, publish them to a registry, and compose them like npm packages.

Perstack isn't another agent framework — it's npm/npx for agents.

## Quick Start

Prerequisites:

- Node.js 22+
- Provider API Credentials
  - See LLM Providers

**Run a demo Expert:**

Set ANTHROPIC_API_KEY (or any provider key you use):

```
$ ANTHROPIC_API_KEY=<YOUR_API_KEY> npx perstack start tic-tac-toe "Ga
```

**Run in headless mode (no TUI):**

```
$ ANTHROPIC_API_KEY=<YOUR_API_KEY> npx perstack run tic-tac-toe "Game
```

**What's next?**

- Rapid Prototyping — build your own Expert
- Taming Prompt Sprawl — fix bloated prompts with modular Experts
- Extending with Tools — add MCP skills to your Experts

## Examples

| Example | Description |
|---|---|
| github-issue-bot | Automated GitHub issue responder with real-time activity logging |
| gmail-assistant | AI-powered email assistant with Gmail integration |

## Key Features

- **Agent-first development toolkit**
  - Declarative Expert definitions as modular micro-agents
  - Dependency management for composing Experts
  - Public registry for reusing Experts instead of rebuilding them
- **Sandbox-ready runtime**
  - Secure execution designed for sandbox integration
  - Observable, event-driven architecture
  - Reproducible, checkpoint-based history

### Safety by Design

Perstack runtime is built for production-grade safety:

- Designed to run under sandboxed infrastructure
- Executes inside Docker containers with no shared global state
- Emits JSON events for every execution change
- Can be embedded in your app to add stricter policies and isolation

## Why Perstack?

AI agent developers struggle with:

- Complex, monolithic agent apps
- Little to no reusability
- No dependency management
- Context windows that explode at scale
- Poor observability and debugging

Perstack fixes these with proven software engineering principles:

- **Isolation** — clear separation of concerns per Expert
- **Observability** — every step is visible and traceable
- **Reusability** — Experts compose declaratively through the runtime

Before/After:

| | Traditional Agent Dev | With Perstack |
|---|---|---|
| **Architecture** | Monolithic & fragile | **Modular & composable** |
| **State** | Global context window | **Isolated per Expert** |

| | Traditional Agent Dev | With Perstack |
|---|---|---|
| **Reuse** | Copy-paste prompts | **npm-style dependencies** |
| **Observability** | Hard to trace | **Full execution history** |
| **Safe execution** | On you | **Sandbox-ready by design** |

## Example: Fitness Assistant

Here is a small example showing how two Experts can collaborate: a fitness assistant and a professional trainer.

```
# ./perstack.toml

[experts."fitness-assistant"]
description = """
Assists users with their fitness journey by managing records and sugg
"""

instruction = """
As a personal fitness assistant, conduct interview sessions and manag
Manage records in a file called `./fitness-log.md` and update it regu
Collaborate with the `pro-trainer` expert to suggest professional tra
"""

delegates = ["pro-trainer"]

[experts."pro-trainer"]

description = """
Suggests training menus by considering the given user information and
"""

instruction = """
Provide training menu suggestions with scientifically verified effect
tailored to the user's condition and mood, and the user's training hi
"""
```

To run this example, execute the following command:

```
$ npx perstack start fitness-assistant "Start today's session"
```

This example shows:

- **Componentization** — each Expert owns one role
- **Isolation** — contexts are separate; shared data lives in the workspace
- **Observability** — full, replayable execution history

- **Reusability** — Experts collaborate declaratively via the runtime

## Next Steps

- Working with Perstack
- Understanding Perstack
- Making Experts
- Using Experts
- Operating Experts
- References

## Motivation

Perstack is built to tackle the core problems of agent development using software engineering best practices.

It centers on three ideas: **Isolation**, **Observability**, and **Reusability**.

### Isolation

Isolation means separating an agent from everything except its role — that's what makes it a true Expert.

Specifically:

- **Model isolation:** the runtime mediates access to LLMs
- **Role isolation:** each Expert focuses on one job
- **Control isolation:** all controls live in tools; Experts only decide how to use them
- **Dependency isolation:** collaboration is resolved by the runtime
- **Context isolation:** context windows are never shared; data flows through runtime/workspace
- **Sandbox support:** designed to align with infra-level isolation

### Observability

Observability means agent behavior is fully transparent and inspectable.

Specifically:

- **Prompt visibility:** no hidden instructions or context injection
- **Definition visibility:** only perstack.toml or registry definitions execute
- **Registry visibility:** write-once per version; text-only, fully auditable
- **Tool visibility:** tools run through MCP; usage is explicit

- **Internal state visibility:** state machines emit visible events
- **Deterministic history:** checkpoints make runs reproducible

### Reusability

Reusability enables agents to collaborate as components — the path to more capable agentic apps.

An Expert is a modular micro-agent:

- Built for a specific purpose
- Reliable at one thing
- Modular and composable

An agent represents a user. An Expert is a specialist component that helps an application achieve its goals.

## Perstack Components

Perstack provides **Expert Stack**:

- **Experts** — modular micro-agents
- **Runtime** — executes Experts
- **Registry** — shares Experts
- **Sandbox Integration** — safe production execution

> ⓘ Note
>
> The name "Perstack" is a combination of the Latin word "perītus" and the English word "stack". "perītus" means "expert", so Perstack means "expert stack".

📚 Read the full documentation →

## FAQ

### Is this an AI agent framework?

No. The relationship is like Express vs npm, or Rails vs RubyGems.

Agent frameworks help you build agent apps. Perstack helps you package, share, and compose the Experts that power them.

### Can Experts in the Registry be used with other AI agent frameworks?

Yes. Registry entries are plain text definitions, so other frameworks can consume them too. See the API Reference.

### Can Experts created with other AI agent frameworks be used with Perstack?

Not directly — but you can re-express their roles in `perstack.toml` to make them Perstack-compatible.

## Contributing

See CONTRIBUTING.md.

## License

Perstack Runtime is open-source under the Apache License 2.0.