State of Env Files Report

2025

.ENV

Table of contents

Introduction

The rise of the env file

In 2013, env files began as a simple way to manage configuration for Heroku-style apps. Over the next decade, they quietly became a universal standard. Today, nearly every modern codebase uses them — even if just during development.

The env pattern emerged from the growing need to separate configuration from code. Inspired by Heroku's developer experience and tools like Foreman, early adopters began managing app secrets and environment-specific settings in a plaintext file. In 2011, the Twelve-Factor App cemented this approach by advocating for environment variables as the canonical method for runtime configuration.

The first dotenv library appeared in Ruby in 2012, followed quickly by ports to Node.js, Python, PHP, Go, and more. By the mid-2010s, env support was widespread—not only as a de facto convention but increasingly bundled directly into frameworks. What started as a developer convenience had become a foundational layer of modern software infrastructure.

Today that foundational layer is maturing with the introduction of encrypted env files via dotenvx. These encrypted versions maintain backwards-compatibility and familiar developer workflows while adding security. And just as env files once supported the rise of the cloud, they're now quietly powering the next wave of software: AI agents, agentic tools, and autonomous coding workflows—all of which need a way to securely configure themselves.

2008 – Secrets lived in code

2011 – Twelve-Factor App

2012 – First dotenv

2013 – Node, Python, PHP support

2014 – Express.js adopts dotenv

2015 – Create React App adopts dotenv

2016 – Next.js adopts dotenv

2020 – Vite adopts dotenv

2023 – Node.js adds native env support

2024 – Dotenvx introduces encrypted env files

2025 – AI agents make use of env files

Ecosystem overview

Env files have become a cornerstone of modern software—embraced by languages, frameworks, and teams worldwide.

Install rates are in the billions per year.

Language	*Library*	***Yearly Installs***
Node.js	npm@dotenv	**2.5 Billion**
Python	python-dotenv	**1.8 Billion**
PHP	phpdotenv	**125 Million***
Ruby	dotenv	**100 Million***
Cross-platform	dotenvx	**24 Million**

* Estimated based on lifetime downloads

Install rates are growing.

Dotenvx burst onto the scene with over 900% growth in its first year—driven by early adopters of the encrypted env file format. Security-conscious teams are embracing encryption of env files by default.

Meanwhile, python-dotenv saw over 100% growth, and npm@dotenv grew over 40%. These surges align with the rise of AI agents and Agentic coding workflows—especially in Python and Node.js ecosystems, where env files remain the simplest way to inject secrets into runtime environments.

We can’t yet say whether security and agent-driven development are directly linked—but both are reshaping how configuration is handled in modern software.

Developer experience

Env files are widely adopted across development teams due to their simplicity and alignment with common software workflows. They offer a lightweight way to manage configuration across environments—supporting portability, ease of setup, and broad compatibility without requiring external services or runtime dependencies.

What developers say.

Honestly? I love .env files because they just work.

I don’t have to spin up some secrets manager, register a new app, or figure out how to inject config into five different environments. I drop a .env file next to my code, and my app runs. That’s it.

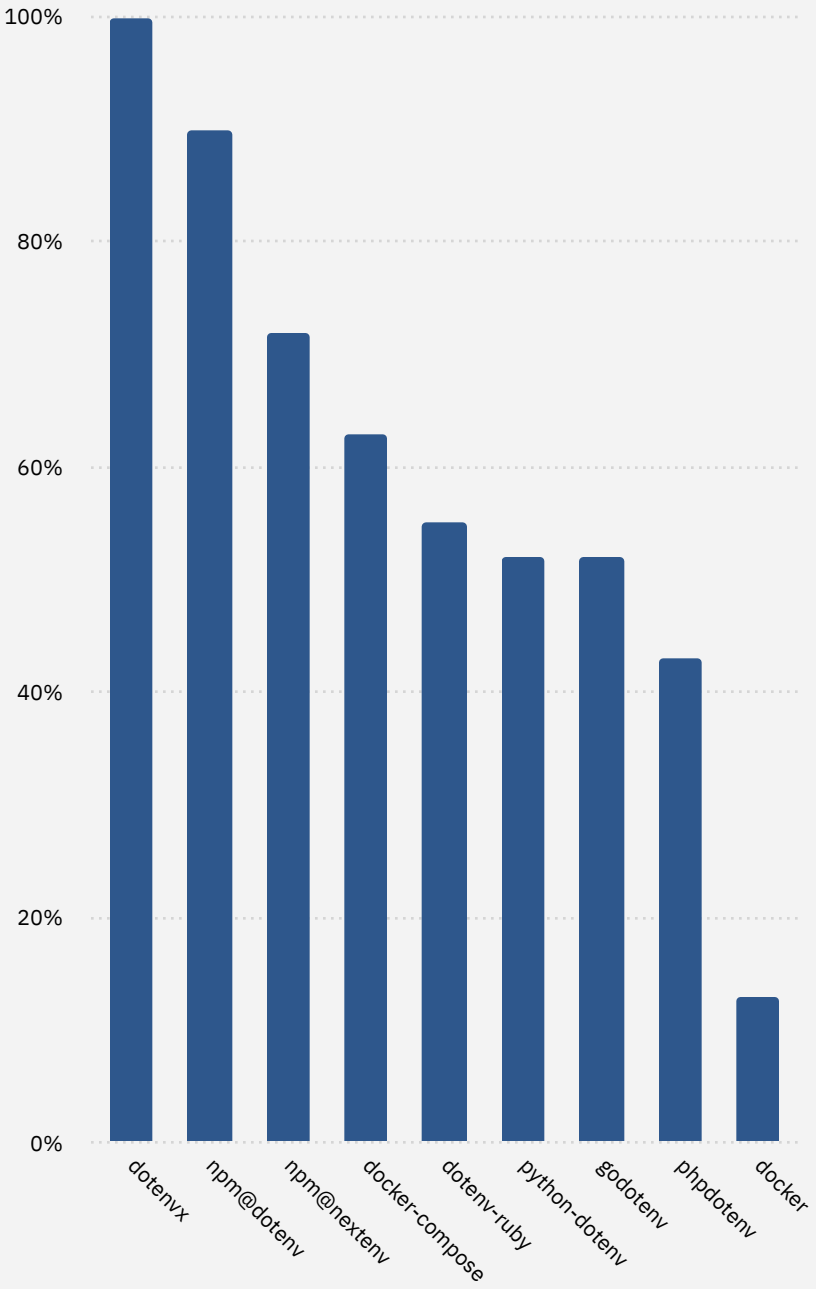
They’re readable. They're versioned (kind of). They don’t care what language I’m using — I can use them in Node, Python, Go, Docker, whatever. I don’t need to learn new syntax or install anything weird. And when I hand off my repo to someone else, I don’t have to explain much:
“Copy .env.example to .env, fill in your keys, and go.”

There’s something powerful about that level of shared muscle memory. It’s invisible infrastructure. Every dev knows what to do with it.

Could it be more secure? Sure. Could it be better structured? Probably. But until there’s something just as portable, just as language-agnostic, just as developer-friendly — I’m sticking with .env.

Feature comparison

Parsing behavior is inconsistent



Dotenvx is the only library in our test suite to achieve 100% pass rate across 80+ real-world env cases.

Most of the libraries fail on progressive variable expansion and nuances between single and double quotes leading to missing variables and inconsistent behavior across environments.

Docker particularly falls short on its env file parsing–unable to even support quotes or inline comments.

source: www.dotenvx.com/spec

These inconsistencies are manageable in small apps—but become liabilities in security-sensitive, agent-driven, or multi-platform environments.

Security gaps

Plaintext by default

Most teams treat env files as "safe enough"—just add them to .gitignore and move on. But in practice, they're often shared over Slack, copied into CI/CD systems, and left behind on laptops. This makes them one of the most widely used yet weakly protected containers of secrets in modern software.

🚫 NO ENCRYPTION

Env files are plaintext by default—if you can read the file, you can read the secrets.

🚫 NO AUDIT TRAIL

There's no way to know who viewed or changed the file. This matters for compliance and incident response.

🚫 NO DISTRIBUTION MODEL

Most teams sync env files manually via email, chat, or shared folders—leaving them exposed and out of sync.

👨‍💻🤖 DEVELOPERS AND AGENTS

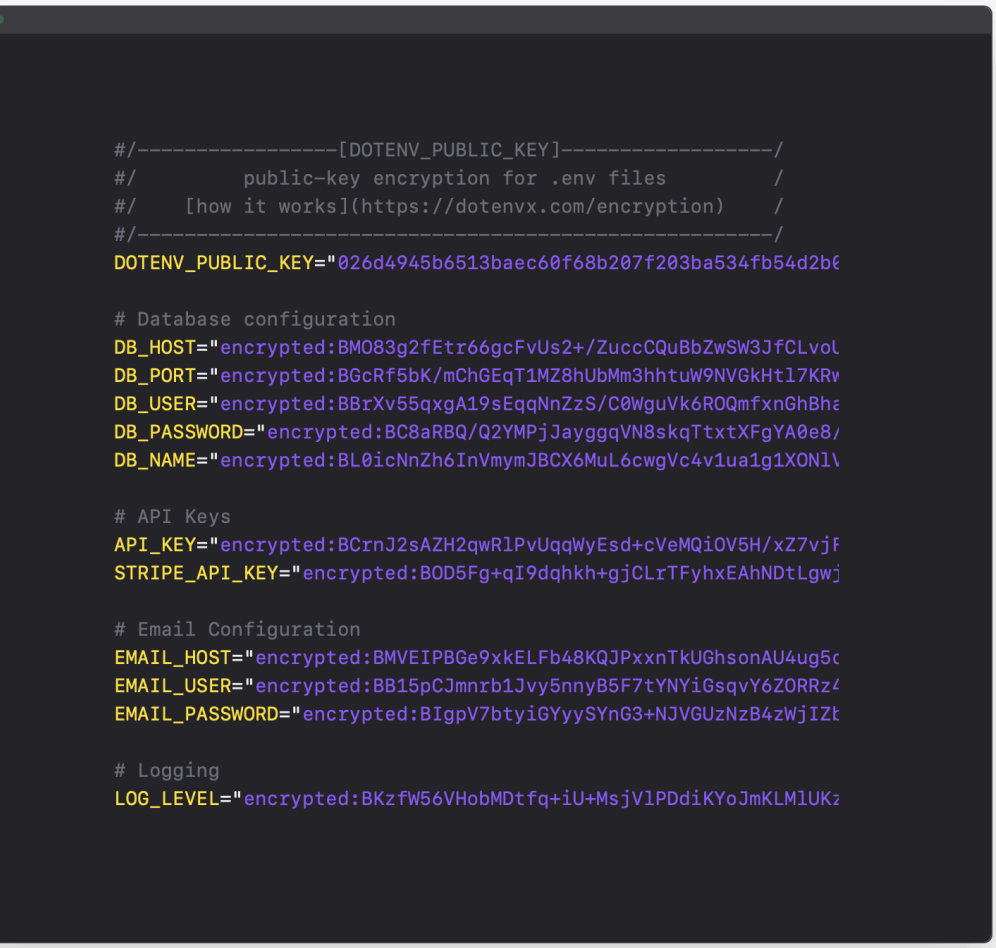
Secrets aren't just stored in production anymore—they live on developer laptops, local env files, and now inside AI agents acting on behalf of users. These are surfaces with minimal visibility, no access controls, and often no encryption.

As agents gain more capabilities, they also inherit the same trust flaws developers have faced for years—becoming a new vector for secret leakage.

"We've hardened production, but secrets still leak from the edges—developer machines, CI jobs, rogue scripts. That's where env files live."

Encrypted env files

Making env files secure at rest



In 2025, dotenvx introduced encrypted env files—a response to the long-standing security gaps in plaintext secrets. The format retains everything developers value: portability, readability, and workflow simplicity. But now, secrets are protected at rest, by default.

Dotenvx encrypts only the values, leaving variable names (API_KEY=...) intact. This means teams can still version and share .env.encrypted files just like before—but without exposing sensitive data.

source: Dotenvx: Reducing Secrets Risk with Cryptographic Separation

✔ *Encrypted*
✔ *Auditable*
✔ *Built-in distribution model*
✔ *Backwards-compatible*
✔ *Unchanged developer workflows*

AI implications

The future is autonomous, and it needs secrets

As AI agents evolve from copilots to coworkers, they're beginning to act independently—deploying infrastructure, calling APIs, writing code, and chaining tools together. These agents don't just need instructions—they need credentials. And increasingly, those credentials are delivered the same way developers receive them: via env files.

Env files are showing up in AI workflows

- **Code generation tools** often scaffold env files automatically (e.g. for OpenAI keys)
- **Agents run in local/dev environments**, where env files are the simplest injection method
- **Multi-agent frameworks** (like CrewAI, LangChain) rely on env files for secret injection
- **Developers guiding LLMs** default to existing patterns—including copying env files

The risk: uncontrolled access

These environments lack centralized controls. AI tools can run on a developer's laptop or a throwaway VM. Once an env file is accessible to an agent, it can be read, logged, or reused—potentially outside its intended scope.

The opportunity: encrypted, agent-safe config

Encrypted env files offer a path forward:

- Environment-variable semantics stay the same
- Agents can be granted access to a key, not the secret itself
- Secrets remain secure, auditable, and scoped to runtime

Just like the env file helped developers configure cloud apps in 2013, it may now help autonomous agents operate securely in 2025 and beyond.

Conclusion

For over a decade, env files have served as a practical, language-agnostic solution for managing environment-specific configuration. Their widespread adoption across ecosystems and tooling reflects not only their technical simplicity, but also their alignment with how developers build, share, and deploy software.

However, as modern applications become more distributed, automated, and security-sensitive—particularly in the context of agentic systems and AI-driven infrastructure—the limitations of plaintext env files become more apparent.

Dotenvx introduces an evolutionary step forward: encrypted env files that maintain backward compatibility while enabling strong encryption, environment-specific access, and secure distribution across developer workflows. This approach preserves the developer experience while addressing the increasingly urgent need for secret management that is both secure and seamless.

As the role of autonomous software agents grows, and as configuration becomes increasingly dynamic and decentralized, encrypted env files offer a model for trustless, portable, and audit-ready configuration—at every layer of the stack.