

The Dark Ages of App Configuration

presented by **Scott Motte – Mot**

www.dotenvx.com



Secrets lived in code
(2008)

cloud application platform
deploy and scale powerful apps



Forget Servers

Run Anything

See Everything

Trust & Manage

Then came Heroku (2009)

Deploy Ruby, Node.js, Clojure, and Java apps.

Get up and running in minutes, and deploy instantly with git.

Focus 100% on your code, and never think about servers,
instances, or VMs again.

Agile Deployment on Heroku ▶

```
$ heroku create
Creating sushi.herokuapp.com | git@heroku.com:sushi.git

$ git push heroku master
----> Heroku receiving push
----> Rails app detected
----> Compiled slug size is 8.0MB
----> Launching... done, v1
http://sushi.herokuapp.com deployed to Heroku
```

[How it Works](#)

It's free to get started and sign up is instant.

[Sign Up](#)

Procfile (2009)

```
1 Procfile
1 web: RAILS_ENV=production rails server
2 worker: RAILS_ENV=production script/delayed_job start
3
```

2 2 2 2 2 2 2 2 2 2 2 2

NORMAL

master

Procfile

33%

1:1

Foreman (2010)

ddollar update readme ce5c8b4 · 15 years ago History

Preview Code Blame 96 lines (61 loc) · 2.58 KB

foreman(1) -- manage Procfile-based applications

SYNOPSIS

```
foreman start [process]
foreman export <var>format</var> [location]
```

DESCRIPTION

Foreman is a manager for Procfile-based applications. Its aim is to abstract away the details of the Procfile format, and allow you to either run your application directly or export it to some other process management format.

RUNNING

`foreman start` is used to run your application directly from the command line.

If no additional parameters are passed, foreman will run one instance of each type of process defined in your Procfile.

The .env Renaissance



Twelve-Factor App (2011)



The screenshot shows a dark-themed web browser window. The address bar displays the URL "12factor.net/config" with a lock icon indicating it's secure. The main content area features a large white diamond logo in the center. To the right of the logo are three links: "Blog", "Community", and a blue "GitHub" button with a white octocat icon. Below the logo, the text "THE TWELVE-FACTOR APP" is displayed in a large, bold, white sans-serif font. The overall theme is minimalist and modern.

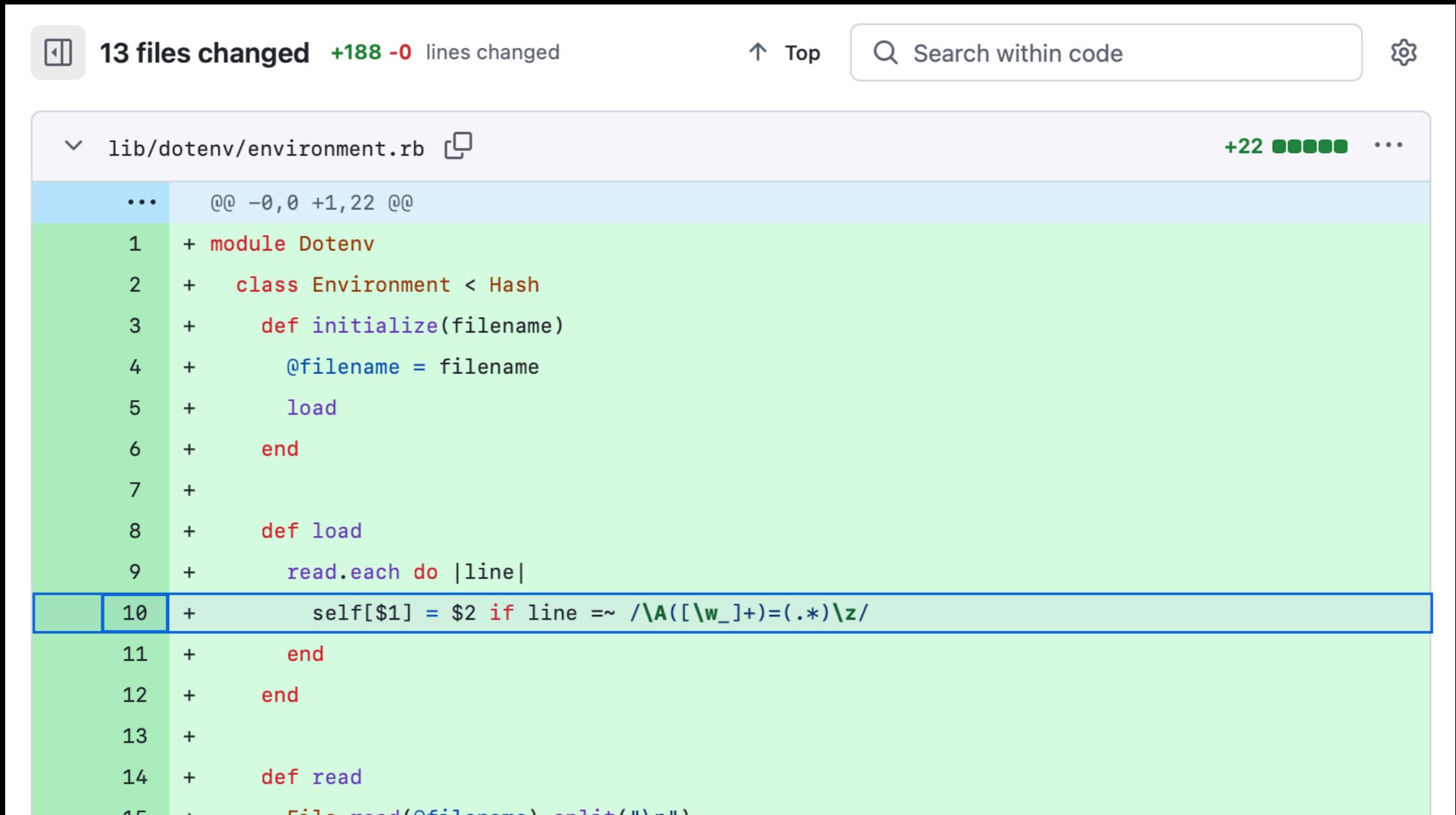
III. Config

Store config in the environment

An app's *config* is everything that is likely to vary between deploys (staging, production, developer environments, etc). This includes:

- Resource handles to the database, Memcached, and other backing services
- Credentials to external services such as Amazon S3 or Twitter
- Per-deploy values such as the canonical hostname for the deploy

Dotenv is Born (2012)



A screenshot of a GitHub commit interface. At the top, it shows "13 files changed +188 -0 lines changed". Below this is a list of changes for the file "lib/dotenv/environment.rb". The changes are color-coded: green for additions and red for deletions. The code shows the initial implementation of the dotenv module and its Environment class.

```
diff --git a/lib/dotenv/environment.rb b/lib/dotenv/environment.rb
@@ -0,0 +1,22 @@
+module Dotenv
+  class Environment < Hash
+    def initialize(filename)
+      @filename = filename
+      load
+    end
+
+    def load
+      read.each do |line|
+        self[$1] = $2 if line =~ /\A([\w_]+)=(.*)\z/
+      end
+    end
+
+    def read
+      File.read(@filename).split("\n")
+    end
+  end
+
```

Ruby to Node (2013)

2013: phpdotenv

2013: python-dotenv

2013: dotenv (nodejs)

2013: godotenv

2013: dotenv_elixir

2014: dotenv-rs

2014: docker-compose

2017: dotenv.net

2017: dotenv-kotlin

2020: dotenv-java

2021: swift-dotenv

Small Code, Big Reach

dotenv v17.2.2

Loads environment variables from .env file

github.com/motdotla/dotenv#readme

9,229,989,255 total npm downloads

month week 📸 ⌂ </>

dotenv 3.1.8

Loads environment variables from `.env`.

GEMFILE:

```
gem 'dotenv', '~> 3.1', '>= 3.1.8'
```

INSTALL:

```
gem install dotenv
```

Star 6,704

TOTAL DOWNLOADS
454,781,786

FOR THIS VERSION
12,399,414

VERSION RELEASED:
ON APR 10

Jun 2023



Before .env files

- Configuration scattered across code, hardcoded credentials, no order.
- Every app reinvented the wheel. Chaos.



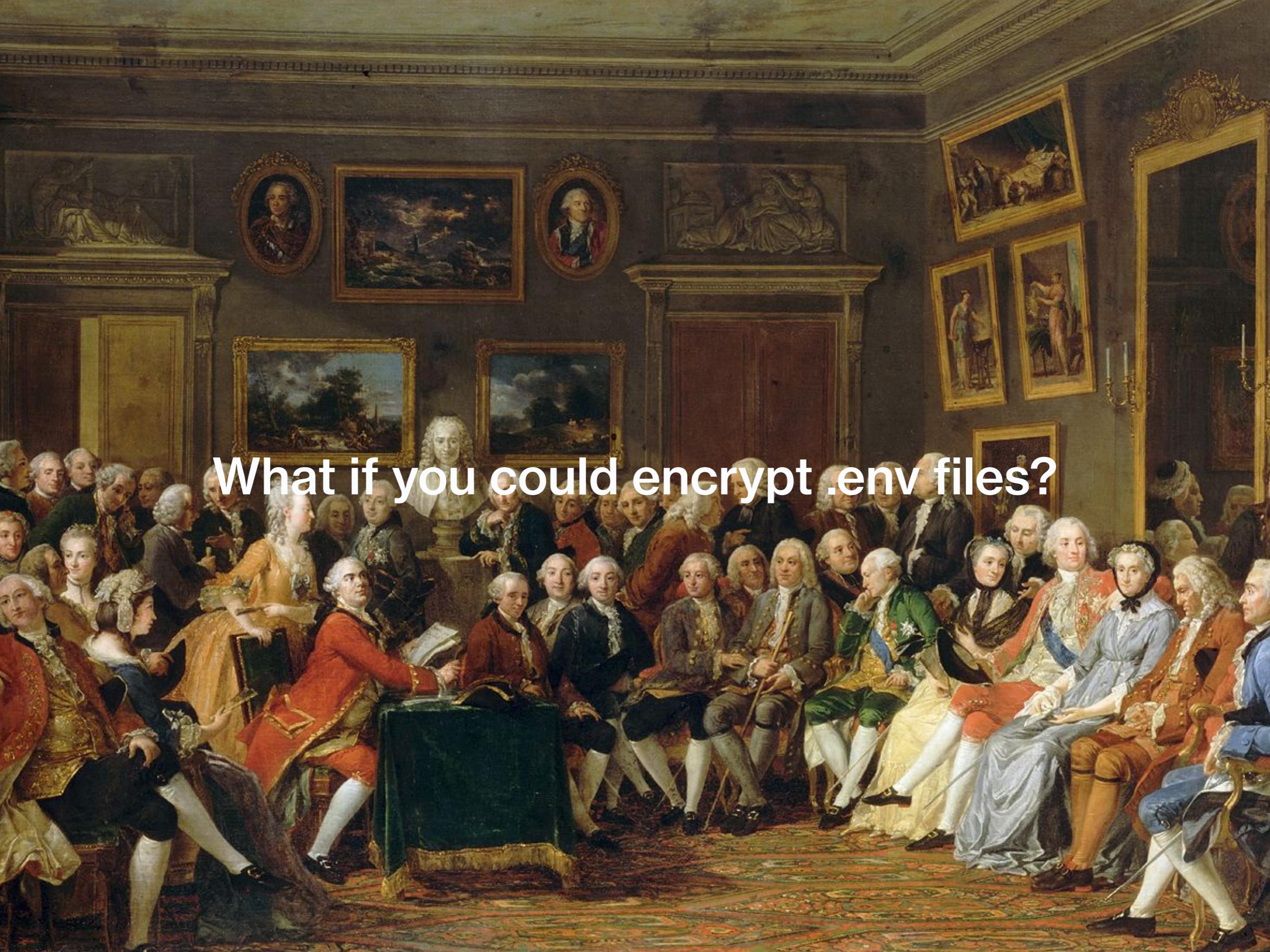
The .env Renaissance

- Simplicity, clarity, and portability return.
- Developers rediscover “human-scale” configuration.
- Tools like dotenv spread literacy and discipline – configuration becomes readable again.



Present day

- What comes next?



What if you could encrypt .env files?

Introducing dotenvx

The screenshot shows a web browser window with the URL `dotenvx.com` in the address bar. The page content is as follows:

.ENV Dotenvx Docs Ops Account

A secure dotenv – from the creator of dotenv

Plaintext .env files have been a major attack vector, but they've also been undeniably useful – even AWS uses them.¹ **What if you could encrypt them?** Now you can with **dotenvx**. Dotenvx encrypts your .env files – limiting their attack vector while retaining their benefits. It's free, open-source, and built and maintained by the creator of the original **dotenv**. – Scott (Mot) Motte

[Quickstart →](#) [Documentation](#)

```
# 🔒 Encrypted with dotenvx
DOTENV_PUBLIC_KEY="026d4945b6513baec60f68b207f203ba534fb54d2b0c995255

# 🗃 Database
DB_HOST="encrypted:BM083g2fEtr66gcFvUs2+/ZuccCQuBbZwSW3JfCLvoUiACmusx
DB_PORT="encrypted:BGcRf5bK/mChGEqT1MZ8hUbMm3hhtuw9NVGkHt17KRwqbSKnVc
```

live demo: `dotenvx encrypt`

How ECIES works

Elliptical Curve Integrated Encryption Scheme

Let's start with the inputs

[DOTENV_PRIVATE_KEY]

330e791f55fd59ae377eb4328e2296629244b9e37372dd5a41322a5133ed0a19

[CIPHERTEXT]

BPWRbDxAUjjLpVJ+w1ytS1FjMPUvnT6Jv9w+I8c+eA3V9rl1JwADfIM9hbmd3ewtP
fYpJd0IBxmuHkFY8+AjfMSoNNKkNmAIZm8qjnFmex9xwbo39987fDOAjwQ3cNkQYF
tqPP

```
4
5 # -----
6 # How ECIES works
7 # -----
8 DOTENV_PRIVATE_KEY =
  "330e791f55fd59ae377eb4328e2296629244b9e37372dd5a41322a5133ed0a19"
9 CIPHERTEXT          = "BPWRbDxAUjjLpVJ+w1ytS1FjMPUvnT6Jv9w+I8c+
                        eA3V9rl1JwADfIM9hbmd3ewtPfYpJd0IBxmuHkFY8+
                        AjffMSoNNKkNmAIZm8qjnFmex9xwbo39987fDOAjwQ3cNkQYFtqPP"
```

Grab their binary form

```
[ pk_bin ]  
{ [DOTENV_PRIVATE_KEY].pack("H*") }
```

```
[ ciphertext_bin ]  
{ Base64.strict_decode64(CIPHERTEXT) }
```

```
10  
-- 11 pk_bin = [DOTENV_PRIVATE_KEY].pack("H*")  
12 ciphertext_bin = Base64.strict_decode64(CIPHERTEXT)  
13
```

Expand ciphertext_bin

```
[ ciphertext_bin ]  
| ephemeral_pk (65) | encrypted_blob |
```

```
13  
-- 14 puts "[ciphertext_bin]"  
15 ephemeral_pk = ciphertext_bin[0, 65]  
16 encrypted_blob = ciphertext_bin[65..]  
-- 17 puts "[ ephemeral_pk (65) | encrypted_blob ]"  
18
```

Expand encrypted_blob

```
[ ciphertext_bin ]  
| ephemeral_pk (65) | encrypted_blob  
| ephemeral_pk (65) | nonce (16) | auth_tag (16) | decryptable (rest) |
```

18

```
19 nonce = encrypted_blob[0, 16]  
20 tag = encrypted_blob[16, 16]  
21 decryptable = encrypted_blob[16 + 16..]  
-- 22 puts "[ ephemeral_pk (65) | nonce (16) | auth_tag (16) | decryptable (rest) ]"  
-- 23 puts [ephemeral_pk.unpack1("H*"), nonce.unpack1("H*"), tag.unpack1("H*"),  
        decryptable.unpack1("H*)].join(" | ")
```

24

Derive shared point (ECDH on secp256k1)

```
[ ciphertext_bin ]  
| ephemeral_pk (65) | encrypted_blob  
| ephemeral_pk (65) | nonce (16) | auth_tag (16) | decryptable (rest) |  
[ ephemeral_pk ]  
{ ephemeral_pk + pk_bin }(secp256k1) = shared_point
```

```
24  
25 # Derive shared point (ECDH on secp256k1)  
26 curve = Secp256k1::PublicKey.new(pubkey: ephemeral_pk, raw: true)  
27 shared_point = curve.tweak_mul(pk_bin)  
28
```

Using ECDH on secp256k1, we combine the ephemeral public key (from the ciphertext) with our private key. That produces a shared point unique to this pair of keys.

Derive AES key from shared point

```
[ ciphertext_bin ]  
| ephemeral_pk (65) | encrypted_blob  
| ephemeral_pk (65) | nonce (16) | auth_tag (16) | decryptable (rest) |  
[ ephemeral_pk ]  
{ ephemeral_pk + pk_bin }(secp256k1) = shared_point  
{ shared_point }(hkdf) = aes_key
```

```
29 # Derive AES key from shared point  
30 combined = curve.serialize(compressed: false) + shared_point.  
    serialize(compressed: false)  
31 aes_key = OpenSSL::KDF.hkdf(combined, salt: "", info: "", length: 32, hash:  
    "SHA256")  
32 puts "[aes_key] #{aes_key.unpack1("H*")}"  
33
```

We pass the combined values through HKDF, a key derivation function, to get our AES-256 key. This is the same key the sender used to encrypt the data.

Decrypt with AES key

```
[ ciphertext_bin ]  
| ephemeral_pk (65) | encrypted_blob  
| ephemeral_pk (65) | nonce (16) | auth_tag (16) | decryptable (rest) |  
[ ephemeral_pk ]  
{ ephemeral_pk + pk_bin }(secp256k1) = shared_point  
{ shared_point }(hkdf) = aes_key  
{ aes_key, decryptable }(aes-256-gcm.decrypt) = plaintext
```

Decrypt [decryptable]
using AES-GCM.

```
33  
34 # Decrypt  
35 decipher = OpenSSL::Cipher.new("aes-256-gcm").decrypt  
36 decipher.key = aes_key  
37 decipher.iv_len = 16  
38 decipher.iv = nonce  
39 decipher.auth_tag = tag  
40 decipher.auth_data = ""  
41 plaintext = decipher.update(decryptable) + decipher.final  
42 puts "[plaintext] #{plaintext}"  
43
```

phew...but

There's now a gem!

gem "eciesrb", require: "ecies"

Quick Start

```
# examples/quickstart.rb
require "ecies"

# Generate a secret key
sk = Ecies.generate_key
raw_sk = Ecies.decode_hex(sk.send(:serialize))
raw_pk = sk.pubkey.serialize(compressed: false)

# Encrypt data with the receiver's public key
plaintext = "Hello, World!"
encrypted = Ecies.encrypt(raw_pk, plaintext)

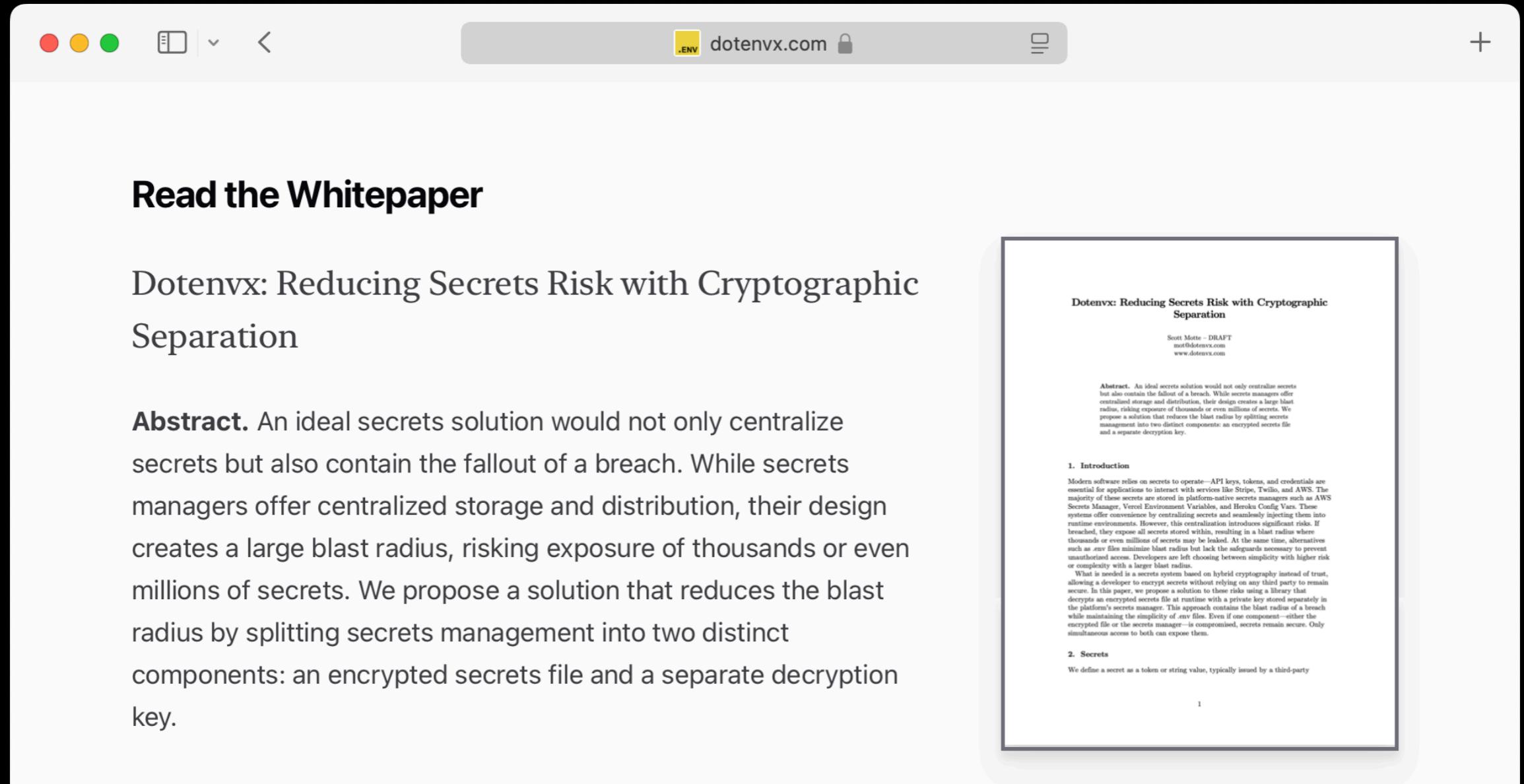
# Decrypt data with the receiver's secret key
decrypted = Ecies.decrypt(raw_sk, encrypted)
puts decrypted # => "Hello, World!"
```

The screenshot shows the RubyGems.org website interface. At the top, there's a navigation bar with links for RELEASES, BLOG, GEMS, GUIDES, and SIGN IN. On the right side of the header, there are buttons for 'Star' (with a count of 1) and 'TOTAL DOWNLOADS' (with a count of 141). Below the header, the main content area displays the 'eciesrb' gem page. The title 'eciesrb' is shown in large text with a version '0.0.1' in red. A brief description follows: 'Elliptic Curve Integrated Encryption Scheme for secp256k1 in Ruby, based on libsecp256k1 and OpenSSL.' Under the 'GEMFILE:' section, there's a code snippet: `gem 'eciesrb', '~> 0.0.1'`. In the 'INSTALL:' section, another code snippet is shown: `gem install eciesrb`. At the bottom, sections for 'VERSIONS:' (listing '0.0.1') and 'RUNTIME DEPENDENCIES (2):' (listing 'libsecp256k1 ~> 0.6.1' and 'openssl ~> 3.3') are present.

That's it. Thank you.

dotenvx.com

Go Deeper



The screenshot shows a web browser window with the address bar containing ".ENV dotenvx.com". The main content area displays a whitepaper titled "Dotenvx: Reducing Secrets Risk with Cryptographic Separation". The page includes an abstract, an introduction, and a section on secrets. A sidebar on the right contains a summary of the paper's purpose and its benefits.

Read the Whitepaper

Dotenvx: Reducing Secrets Risk with Cryptographic Separation

Abstract. An ideal secrets solution would not only centralize secrets but also contain the fallout of a breach. While secrets managers offer centralized storage and distribution, their design creates a large blast radius, risking exposure of thousands or even millions of secrets. We propose a solution that reduces the blast radius by splitting secrets management into two distinct components: an encrypted secrets file and a separate decryption key.

Dotenvx: Reducing Secrets Risk with Cryptographic Separation

Scott Motte – DRAFT
mott@dotenvx.com
www.dotenvx.com

Abstract. An ideal secrets solution would not only centralize secrets but also contain the fallout of a breach. While secrets managers offer centralized storage and distribution, their design creates a large blast radius, risking exposure of thousands or even millions of secrets. We propose a solution that reduces the blast radius by splitting secrets management into two distinct components: an encrypted secrets file and a separate decryption key.

1. Introduction

Modern software relies on secrets to operate—API keys, tokens, and credentials are essential for applications to interact with services like Stripe, Twilio, and AWS. The majority of these secrets are stored in platform-native secrets managers such as AWS Secrets Manager, Vercel Environment Variables, and Heroku Config Vars. These systems offer convenience by centralizing secrets and seamlessly injecting them into runtime environments. However, this centralization introduces significant risks. If breached, they expose all secrets stored within, resulting in a blast radius where thousands or even millions of secrets may be leaked. At the same time, alternatives such as .env files minimize blast radius but lack the safeguards necessary to prevent unauthorized access. Developers are left choosing between simplicity with higher risk or complexity with a larger blast radius.

What is needed is a new system based on hybrid cryptography instead of trust, allowing a developer to encrypt secrets without relying on any third party to remain secure. In this paper, we propose a solution to these risks using a library that decrypts an encrypted secrets file at runtime with a private key stored separately in the platform's secrets manager. This approach contains the blast radius of a breach while maintaining the simplicity of .env files. Even if one component—either the encrypted file or the secrets manager—is compromised, secrets remain secure. Only simultaneous access to both can expose them.

2. Secrets

We define a secret as a token or string value, typically issued by a third-party

Sources

- Apr 24, 2019. Heroku Launches. https://blog.heroku.com/commercial_launch
- May 16, 2010. First Foreman commit. <https://github.com/ddollar/foreman/commit/2dbd3e6be5f5a45ee3b236735130beaedc436bb2>
- May 13, 2011. .env file introduced to Foreman. <https://github.com/ddollar/foreman/issues/17>, <https://github.com/ddollar/foreman/commit/9193a675a3e53739f412d4e493ab74594d1e826c#diff-d3b801ad9a2c2e0606e87eadd12d18b740274ba85bd0354a6ff749245e4d1deeR204>
- Jun 03, 2011. Twelve-Factor App is written. <https://github.com/adamwiggins/12factor/commit/2b06e7deabb64bb759f9fc6f4d9b6fcc546921bb>
- Jul 23, 2012. dotenv ruby is born. <https://github.com/bkeepers/dotenv/commit/c3568a06b341f1182bd4e8b0d6e58a594cac7966#diff-b335630551682c19a781afebcf4d07bf978fb1f8ac04c6bf87428ed5106870f5R5>
- Jan 22, 2013. phpdotenv is born. <https://github.com/vlucas/phpdotenv/commits/master/?after=20d6a1bdfd62910da28b447b2ccb39ac542e96b2+570>
- Jun 13, 2013. python-dotenv is born. <https://github.com/theskumar/python-dotenv/commit/5fc02b7303e8854243970e12564f2433da7a1f7f>
- Jul 5, 2013. dotenv node is born. <https://github.com/motdotla/dotenv/commit/71dabbf27b699fcb7a04714709cefc6e78892b9>

Sources continued

- Jul, 2013. godotenv is born. <https://github.com/joho/godotenv/commit/973cf53008332ef58d44121143d3ef29758c3352>
- Dec 03, 2013. dotenv_elixir is born. https://github.com/avdi/dotenv_elixir/commit/9a9d61ae4e449dc5f4fb414bd189183e339d7210
- Oct 23, 2014. dotenv-rs is born. <https://github.com/dotenv-rs/dotenv/commit/47570b116e1b9653be66861191152e7ad0a9078a>
- Dec 09, 2014. docker-compose. <https://github.com/docker/compose/pull/665>
- Nov 21, 2017. dotenv.net is born. <https://github.com/bolorundurowb/dotenv.net/commit/67714bc46294008aa6726323bf0319ad9e03c6d9>
- Nov 25, 2017. dotenv-kotlin is born. <https://github.com/cdimascio/dotenv-kotlin/commit/429a5172a8bc113818cf8687c6a15bd4244d9d35>
- Sep 18, 2020. dotenv-java is born. <https://github.com/cdimascio/dotenv-java/commit/4057ea1c1d930d74b5282f48d68332d517eca718>
- Oct 17, 2021. swift-dotenv is born. <https://github.com/thebarndog/swift-dotenv/commits/develop/?after=bed53315bc88e8e9d3a7eabb2b9440c6e9c7aa51+100>