Advanced Lambda Tips, Tricks, and Usage

Configuration

- Startup is a lambda cost
- Instance memory also determines network, disk, and CPU performance.
- A lambda living inside a VPC has a much longer lifespan
- Encrypt your ENV values with KMS

Configuration

- Startup is a lambda cost, understand yours
- Startup times vary from language to language, and with instance size.

	A	B	C	D	E	F	G
1	language	memory size	std dev	mean	median	95%-tile	99%-tile
2	csharp	128	405.17	4387.87	4372.80	4980.11	5352.78
3	csharp	256	355.80	2234.23	2132.14	2848.27	3040.10
4	csharp	512	349.12	1223.42	1538.10	1636.65	1948.62
5	csharp	1024	192.28	524.29	489.16	712.55	1320.18
6	csharp	1536	270.25	407.99	338.04	694.43	1049.45
7	java	128	547.83	3562.12	3464.38	4493.39	5100.89
8	java	256	210.62	1979.44	1958.26	2358.66	2593.52
9	java	512	132.99	999.78	826.86	1212.71	1311.84
10	java	1024	86.31	530.99	534.37	677.64	744.75
11	java	1536	73.97	339.65	329.34	425.31	479.87
12	nodejs6	128	12.43	12.67	2.06	38.23	55.57
13	nodejs6	256	8.41	8.94	2.08	25.44	35.89
14	nodejs6	512	4.08	3.73	2.08	15.12	20.04
15	nodejs6	1024	0.45	2.11	2.06	2.28	3.21
16	nodejs6	1536	0.13	2.09	2.07	2.29	2.42
17	python	128	12.10	1.28	0.19	0.41	32.43
18	python	256	2.94	0.83	0.20	3.88	16.08
19	python	512	1.45	0.39	0.20	0.45	6.76
20	python	1024	1.60	0.45	0.20	0.47	9.00
21	python	1536	1.01	0.34	0.19	0.38	6.60

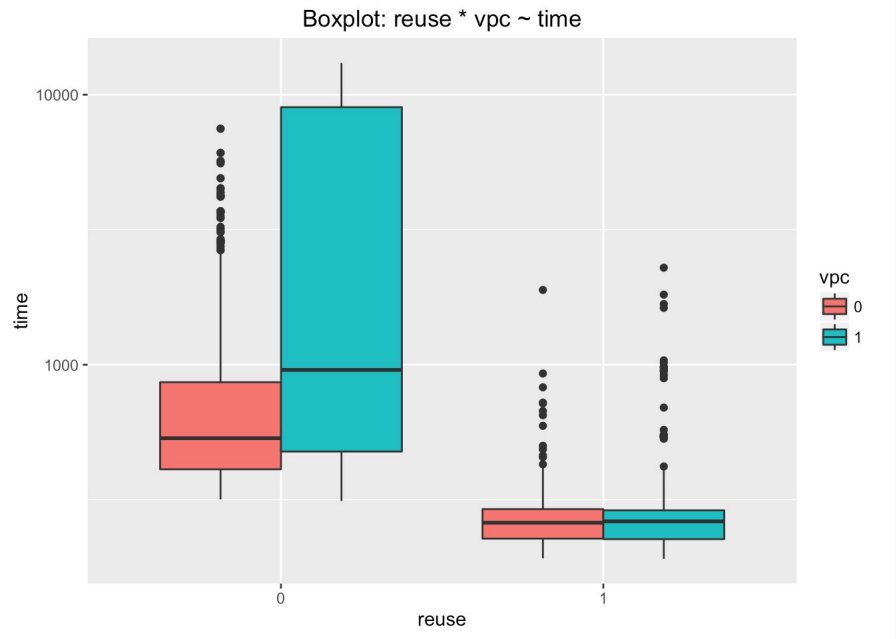
* Courtesy TheBurningMonk.com
https://github.com/theburningmonk/lambda-coldstart-comparison

Configuration

- Your lambda can live in VPC, or not...

Configuration

- Startup times vary from VPC contained versus not.



*Courtesy robertvojta.com
https://github.com/purposefly/lambda-performance

Configuration

- Your Lambda will consume an ENI when networked:
  - Default limit per account is 350

- VPC contained Lambdas:
  - CANNOT contact the internet without a NAT
    - CANNOT use an Internet Gateway for this
    - Managed NAT provides 10GBPS bursts
  - NAT can be managed or instance based.
  - Care about DNS resolutions, and caching them

Packaging

- Your Lambda code size has significant effect on warm up time.

- All your binaries are ~~belong to us~~ x64
- Every runtime comes with:
  - AWS SDK – AWS SDK for JavaScript version 2.54.0
  - AWS SDK for Python 2.7 (Boto 3) version 3-1.4.4 botocore-1.5.52
  - AWS SDK for Python 3.6 (Boto 3) version 3-1.4.4 botocore-1.5.52
  - Amazon Linux build of java-1.8.0-openjdk for Java.

Tricks

- Scratch space on /tmp
  - Store DNS, counters, etc.
- Code required outside the scope of your handle function is not wiped between invocations!
- console.log handy but not guaranteed! Also, implicitly asynch.

...

- Or just run any JVM based language (Scala, Clojure, JRuby)
- Or transpile into JavaScript (ClojureScript, TypeScript)
- Or run any C-friendly language via pyc (golang is a popular choice!)
- Or bring native runtimes with you (traveling-ruby)

Dev & Deploy

- SAM and SAM Local from AWS
  - CloudFront mounted automated deployment, including a local test env
  - Swagger.yml based API Gateway integration
  - Lots of other easy integrations (tables, IoT, S3 and even Alexa!)
  - Highly recommended.

- Or try Serverless.js (javascript)
- Or Chalice (python), or Sparta (Golang)
- Or Apex (Rust, Clojure, Python, Node, Golang, Java)

Or just use CloudFront!

Thank You!