

KubeCon + CloudNativeCon Seattle

Distributed Tracing in Serverless Systems

Nitzan Shapira, Epsagon

> whoami



Nitzan Shapira (@nitzanshapira)

Software engineer > 12 years

Co-Founder, CEO at Epsagon





2

Things to discuss

What is serverless? How is it different?

What is observability for serverless?

How can distributed tracing help?

How will it help my job?



What is serverless?



AWS

AUC

Coracle

Cloub

OPENFARS
Google Cloud
IBM Cloud
BINARIS
Elines



[Compute-as-a-Service] **FaaS**: Function-as-a-Service **CaaS**: Container-as-a-Service

Managed services (APIs)

=

+

Don't manage infrastructure Focus on business logic



Δ

Why serverless?

Pay-per-use: reduces cloud compute cost by 90%

Out-of-the-box auto-scaling

DevOps \rightarrow LowOps

++Developer velocity

Focus on business logic – iterate faster





5

The limitations of FaaS



Limited memory



Limited running time

+ concurrency limit+ some others...



Stateless



Cold starts



The properties of serverless applications

Serverless is micro-services

Serverless applications are

- Highly distributed
- Highly event-driven

Utilizing managed services via **APIs** is key





A real example – HSBC



Source: re:Invent 2018



8

The challenge in serverless



COMPLEX



Yan Cui





What the community thinks

Which part of development poses the biggest challenge when it comes to serverless?



2018 Serverless Community Survey, serverless.com, July 2018

Which of the following are serious pain points for you in developing serverless architectures?

(175 responses)





Observability – why do we need it?



Track system health



Troubleshoot and fix



Optimize performance and cost



Observability in serverless

Let's go one by one



Track system health

System == Functions ?



Functions are important



- Errors
- Timeout
- Out-of-memory
- Cold start



Track system health



Serverless != Functions



15 epsagon

Track system health

System > Functions !



Functions APIs **Transactions**



Troubleshoot and fix

	Filter events	all 30s 5m 1h 6h 1d 1w cust	- mc	
	Time (UTC +00:00)	Message		
	2018-05-16			
	• 07:23:06	[INFO] 2018-05-16107:23:06.5552 18014219-5809-1168-0268-058020686245 93.25.170.119 [16/May/	2018:07:	
	07:23:06	END Requestid: f8cf4219-58d9-11e8-b2e8-b58d206a6245		
Servo CLCX approvingent buller designated	• 07:23:06	REPORT Requestid: 18c14219-58d9-11e8-b2e8-b58d206a6245 Duration: 2505.00 ms Billed Duration: 260	0 ms M	
	07:23:08	START Requestid: tb4c7/a8-58d9-11e8-a11e-85e5b5t027cc Version: \$LATEST		
approval apr	07:23:08	[DEBUG] 2018-05-16107:23:08.2292 tb4c77a8-58d9-11e8-a11e-85e5b5t027cc Zappa Event: ('resource'	'/{prox}	
PCRT Impuni	• 07:23:08	[INFO] 2018-05-16107:23:08.230Z fb4c7/a8-58d9-11e8-a11e-85e5b5f027cc 93:25.170.119 [16/May/2	018:07:	
	• 07:23:08	END Requestid: tb4c/7a8-58d9-11e8-a11e-85e5b5t027cc		
CloudWatch > Log Gro	▶ 07:23:08	REPORT Requestid: tb4c77a8-58d9-11e8-a11e-85e5b5f027cc Duration: 1.94 ms Billed Duration: 100 ms Memory		
	• 07:23:08	START RequestId: fb75837b-58d9-11e8-9cc1-dda6c3fcedc2 Version: \$LATEST		
Search Les Cours	► 07:23:08	[DEBUG] 2018-05-16107:23:08.4922 fb/583/b-58d9-11e8-9cc1-dda6c3fcedc2 Zappa Event: ('resource'	/{prox	
Search Log Group	Creat ► 07:23:09	[INFO] 2018-05-16107:23:09.943Z tb75837b-58d9-11e8-9cc1-dda6c3tcedc2 93.25.170.119 [16/May/	2018:07:	
Filter: Log Stream N	Ime Pref 07:23:09	END Requestid: fb/583/b-58d9-11e8-9cc1-dda6c3fcedc2		
Los Streame	• 07:23:09	REPORT Requestid: tb75837b-58d9-11e8-9cc1-dda6c3fcedc2 Duration: 1451.67 ms Billed Duration: 150	0 ms M	
2019/01/1/12/21 ATE	► 07:23:32	START RequestId: 09c09179-58da-11e8-a16b-1389cef3ed2a Version: \$LATEST		
	► 07:23:32	[DEBUG] 2018-05-16T07:23:32.486Z 09c09179-58da-11e8-a16b-1389cef3ed2a Zappa Event: {'resource	: '/{prox	
2018/01/13/[SLATE	STJC665 > 07:23:32	[INFO] 2018-05-16T07:23:32.488Z 09c09179-58da-11e8-a16b-1389cef3ed2a 93.25.170.119 [16/May/	2018:07	
2018/01/13/[SLATE	STJ064ft > 07:23:32	END Requestid: 09c09179-58da-11e8-a16b-1389cef3ed2a		
2018/01/07/[\$LATE	STJa60a 07:23:32	REPORT RequestId: 09c09179-58da-11e8-a16b-1389cef3ed2a Duration: 4.55 ms Billed Duration: 100 m	Memo	
2018/01/07/[\$LATE	ST]1c951 07:23:32	START RequestId: 09e753t0-58da-11e8-b9a0-7d5a560e9tad Version: \$LATEST		
2018/01/07/[\$LATE	STjbd9al 07:23:32	[DEBUG] 2018-05-16T07:23:32.724Z 09e753f0-58da-11e8-b9a0-7d5a560e9fad Zappa Event: ('resource'	'/{prox	
2018/01/07/[\$LATE	STJ0077: 07:23:32	[INFO] 2018-05-16T07:23:32.734Z 09e753f0-58da-11e8-b9a0-7d5a560e9fad 93.25.170.119 [16/May/2	:018:07:	
2018/01/06/ISLATE	► 07:23:32	END Requestid: 09e753t0-58da-11e8-b9a0-7d5a560e9tad		
Lambda > Exections > ping librato www 2018/01/06//\$LATE	► 07:23:32	REPORT Requestid: 09e75310-58da-11e8-b9a0-7d5a560e9fad Duration: 11.37 ms Billed Duration: 100 m	s Memo	
2018/01/06/\$LATE	07:24:52	START Requestid: 39b1d27b-58da-11e8-9f84-b9f40d4e8837 Version: \$LATEST		
Actions 2010/00/30/12	07:24:52	[DEBUG] 2018-05-16T07:24:52.959Z 39b1d27b-58da-11e8-9f84-b9f40d4e8837 Zappa Event: {'time': '20	18-05-1	
Code Configuration Event sources	STJ03mck ► 07:24:52	[DEBUG] 2018-05-16107:24:52.979Z 39b1d27b-58da-11e8-9f84-b9f40d4e8837 Zappa Event: {}		
CloudWatch metrics at a plance (last 24	ST12641 > 07:24:52	END Requestid: 39b1d27b-58da-11e8-9f84-b9f40d4e8837		
2018/01/02/[\$LATE	STJ0730 07:24:52	REPORT RequestId: 39b1d27b-58da-11e8-9f84-b9f40d4e8837 Duration: 20.26 ms Billed Duration: 100 n	is Memo	
Invocation count C 2017/12/24/[\$LATE	ST]c3e00904081b4477ab254d823734c28c	2017-12-24 03:39 UTC		
15 2017/12/24/[\$LATE	ST]bdfbfe1fbf9d45a7a77eae39742d4165	2017-12-24 03:38 UTC		
2017/12/24/[\$LATE	STJ24249247d9864f8b954eea0554bd625d	2017-12-24 03:35 UTC		
10 2017/12/24/[\$LATE	ST]63851da58bcb4b078ceee2098e287c96	2017-12-24 03:31 UTC		
5	ST118-00174005944440906006883106799	9017.19.94 09-94 HTC		
100				
0 0 15:00 23:00 7:00 15:00 23:00 7:00	0 15.00 22.00 7.00 0	15.00 22.00 7.00		
Other monitoring information				
Last modified date 2015-11-30T16:09:54.764+0000				
		produce production provide product pro		
Ikibana	·	als als als als als als		
a h		Law Pate API		

Functions are not enough

Need: track asynchronous events



Transactions



18 epsagon

Tracing asynchronous invocations





Tracing asynchronous invocations



20 epsagon

Tracing asynchronous invocations





Distributed tracing



...a **trace** tells the story of a transaction or workflow as it propagates through a (potentially distributed) system. Distributed tracing is a method used to profile and monitor applications.



Distributed tracing



Jaeger

Implementing distributed tracing

Manual tracing/instrumentation

Before/after calls

At the end of each micro-service

High maintenance High potential of errors





Inbound request

Somewhere in your server's request handler code:

def handle request(request): span = before_request(request, opentracing.global_tracer()) # store span in some request-local storage using Tracer.scope manager. # using the returned `Scope` as Context Manager to ensure # `Span` will be cleared and (in this case) `Span.finish()` be called. with tracer.scope_manager.activate(span, True) as scope: # actual business logic handle_request_for_real(request) def before_request(request, tracer): span_context = tracer.extract(format=Format.HTTP_HEADERS, carrier=request.headers, span = tracer.start_span(operation_name=request.operation, child_of(span_context)) span.set_tag('http.url', request.full_url) remote_ip = request.remote_ip if remote_ip: span.set_tag(tags.PEER_HOST_IPV4, remote_ip) caller_name = request.caller_name if caller_name: span.set_tag(tags.PEER_SERVICE, caller_name) remote_port = request.remote_port if remote_port: span.set_tag(tags.PEER_PORT, remote_port) return span



Serverless apps are very distributed

Complex systems have thousands of functions

What about the **developer velocity**?



Can it be done differently in serverless?





Automation can help to keep up with the

development speed of serverless





Example







Monitoring serverless



Limited memory



Limited running time



Stateless



Cold starts



Time **is** \$\$\$



Where do we spend the most time?

Our own code

API calls

Serverless cost crisis A real-life example

\$\$\$\$\$\$\$\$\$

33 epsagon

Scanning functions

Scanning CloudWatch using AWS Lambda

Every 5 minutes, save to RDS

A new Lambda is spawned for every customer's function

CloudWatch became highly throttled

Requests took too much time

5K concurrent Lambdas, for 5 minutes, timing out, every 5 minutes

Why you should care about external APIs

36 epsagon

Track service health

37 epsagor

Business flows

What should I optimize first?

D	Dataflows 24 hours 7 days Q Search by Dataflow name							
	Dataflow Name	Invocation Count	p50 Duration (ms)	p99 Duration (ms)	\$			
	Posts Analysis	6,464	2229.96	3240.58	⊡ Transactions			
	Posts Upload	7,171	515.70	67722.40	☐ Transactions			
	Read Blog Posts	5,252	403.39	462.81	⊡ Transactions			

Remember...

Serverless + Distributed Tracing

 \equiv

Perfect marriage (but only if you automate)

nitzan@epsagon.com

@nitzanshapira

www.epsagon.com

