



COMMUNITY DAY

AOTEAROA



Video Streaming

Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%

The move from a distributed microservices architecture to a monolith application helped achieve higher scale, resilience, and reduce costs.



COMMUNITY DAY

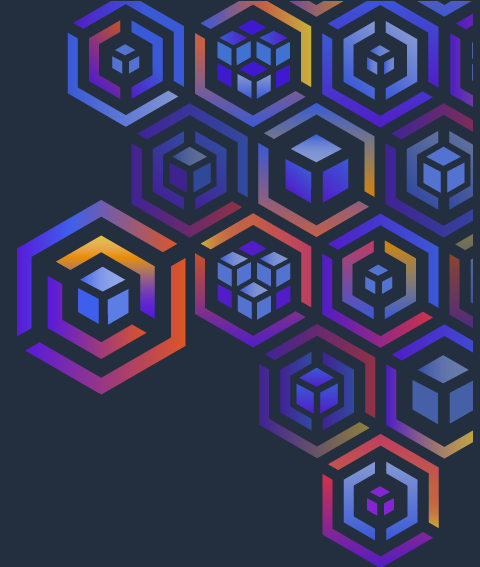
AOTEAROA

Convert video streams to frames/decrypt audio

Monitoring the quality of every video stream

Analyse frames or audio buffers for defects

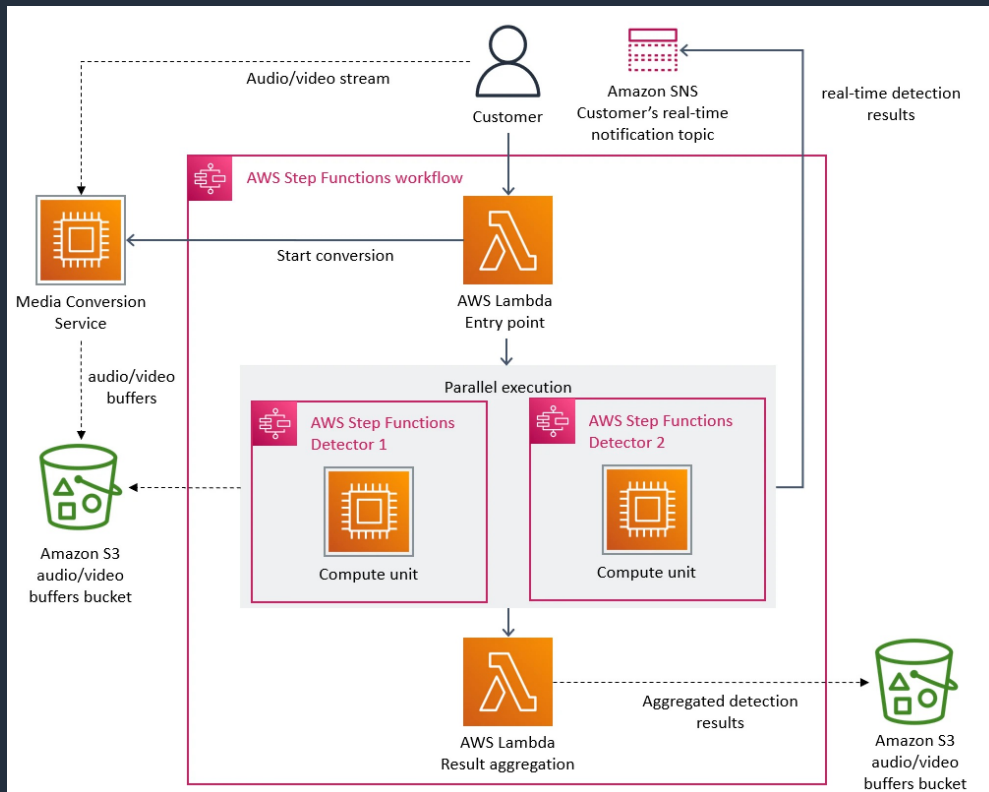
Send real-time notifications



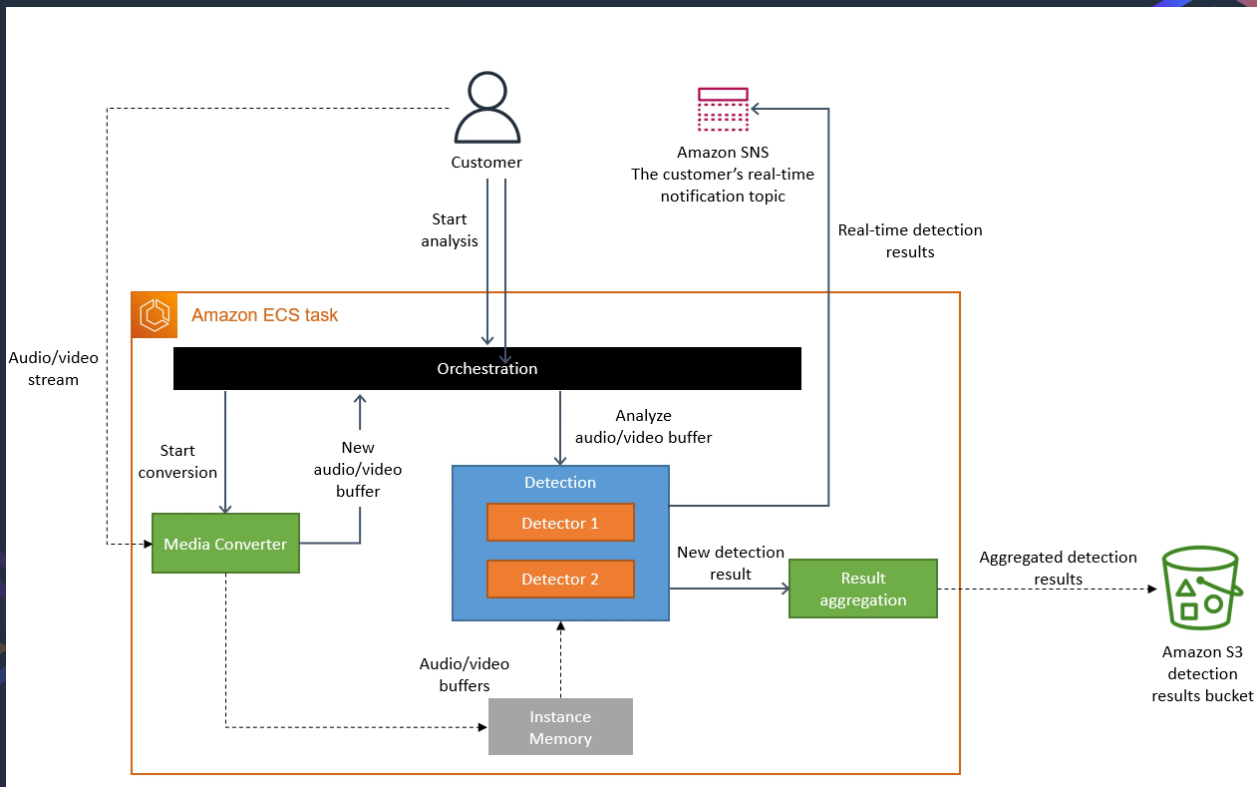


COMMUNITY DAY

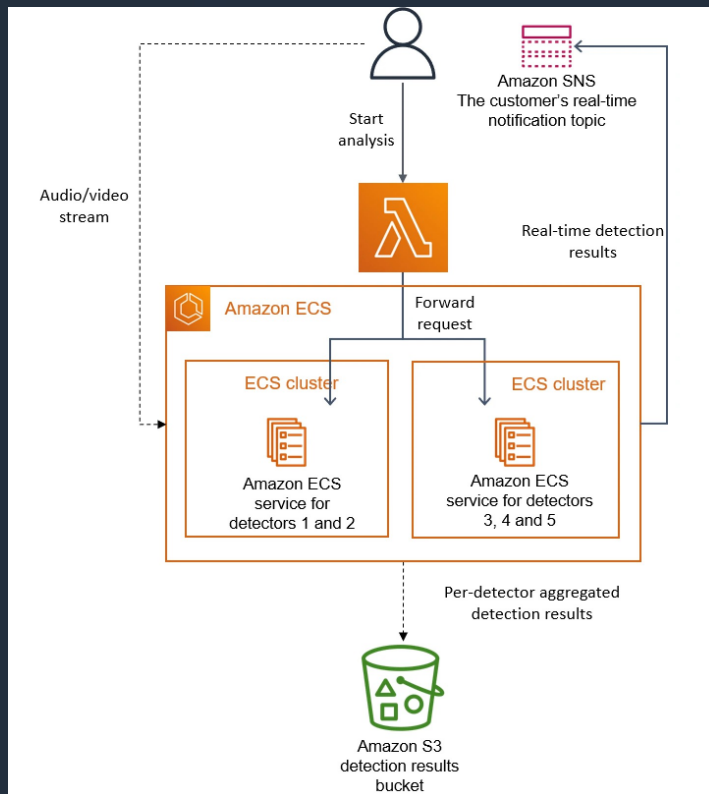
AOTEAROA



aws
COMMUNITY DAY
AOTEAROA



aws
COMMUNITY DAY
AOTEAROA







COMMUNITY DAY

AOTEAROA

So many bad takes - what is there to learn from the Prime Video microservices to monolith story

Containers

Custom code and services

Lots of choices of frameworks and API mechanisms

Where needed, optimize serverless applications by also building services using containers to solve for

- Lower startup latency
- Long running compute jobs
- Predictable high traffic

Serverless

Serverless events and functions

Standardized choices

Combine these building blocks

- λ AWS Lambda
- API Gateway
- Amazon SNS, SQS
- Amazon DynamoDB
- AWS Step Functions

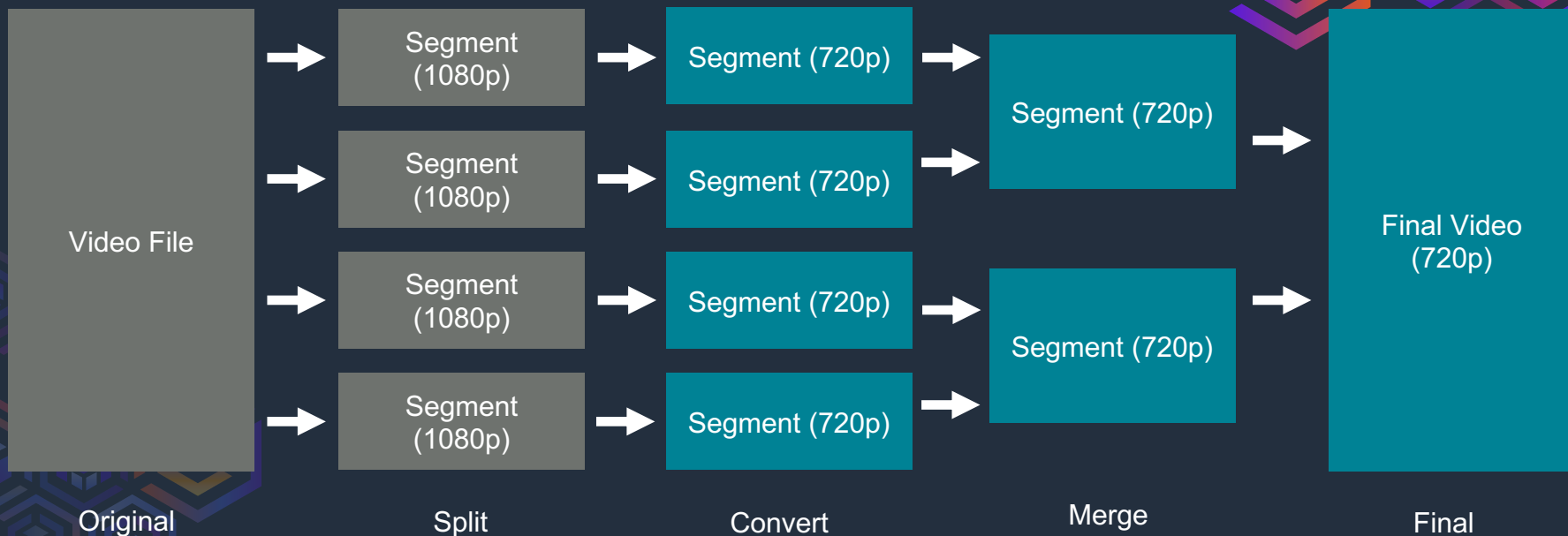




COMMUNITY DAY

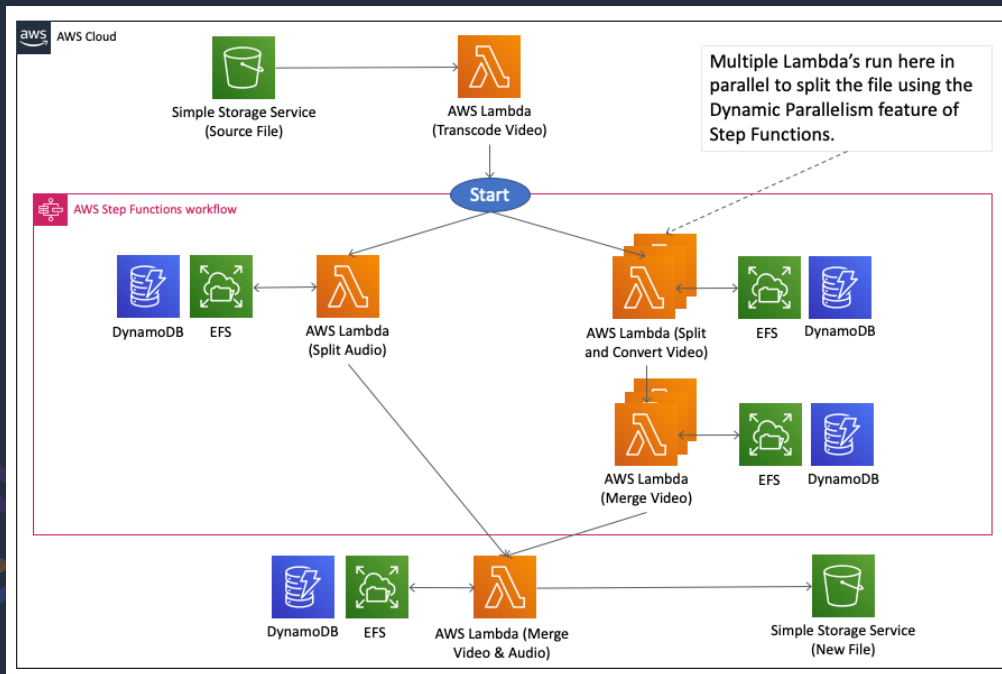
AOTEAROA

Divide and conquer



Read more: <https://bit.ly/3wJOdvQ>

Parallel Computing with Lambda & Step Functions



Serverless Video Transcoder

	Serverless Lambda	Traditional EC2 (t2.large)	MacBook Pro 16GB 3.5GHz i7
34MB MP4 (00:43, 1920x1080)	11 seconds	32 seconds	18 seconds
77MB MP4 (6:49, 2048x1152)	26 seconds	144 seconds	78 seconds
100MB MP4 (59:56, 1280x720)	86 seconds	1073 seconds	592 seconds
350MB MP4 (07:45, 2560x1440)	35 seconds	432 seconds	224 seconds
420MB MKV (01:02, 3840 x 1606)	112 seconds	157 seconds	101 seconds
1GB MKV (57:57, 1280 x 718)	185 seconds	4320 seconds	2367 seconds



Encoding, Fast and Slow: Low-Latency Video Processing Using Thousands of Tiny Threads

Sadjad Fouladi¹, Riad S. Wahby¹, Brennan Shacklett¹,
Karthikeyan Vasuki Balasubramaniam², William Zeng¹,
Rahul Bhalerao², Anirudh Sivaraman³, George Porter², Keith Winstein¹

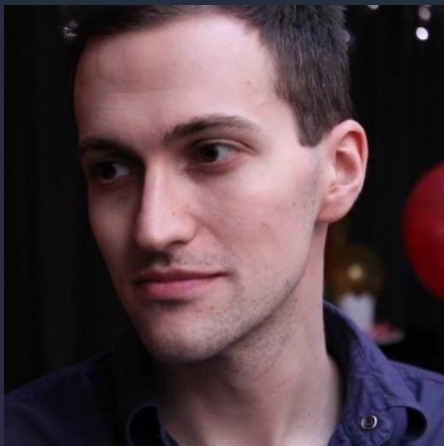
¹Stanford University, ²UC San Diego, ³MIT



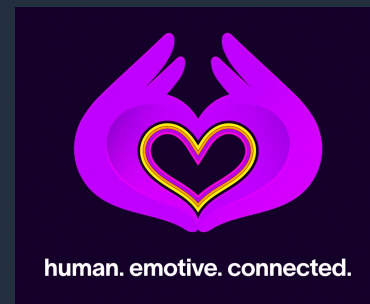


COMMUNITY DAY

AOTEAROA



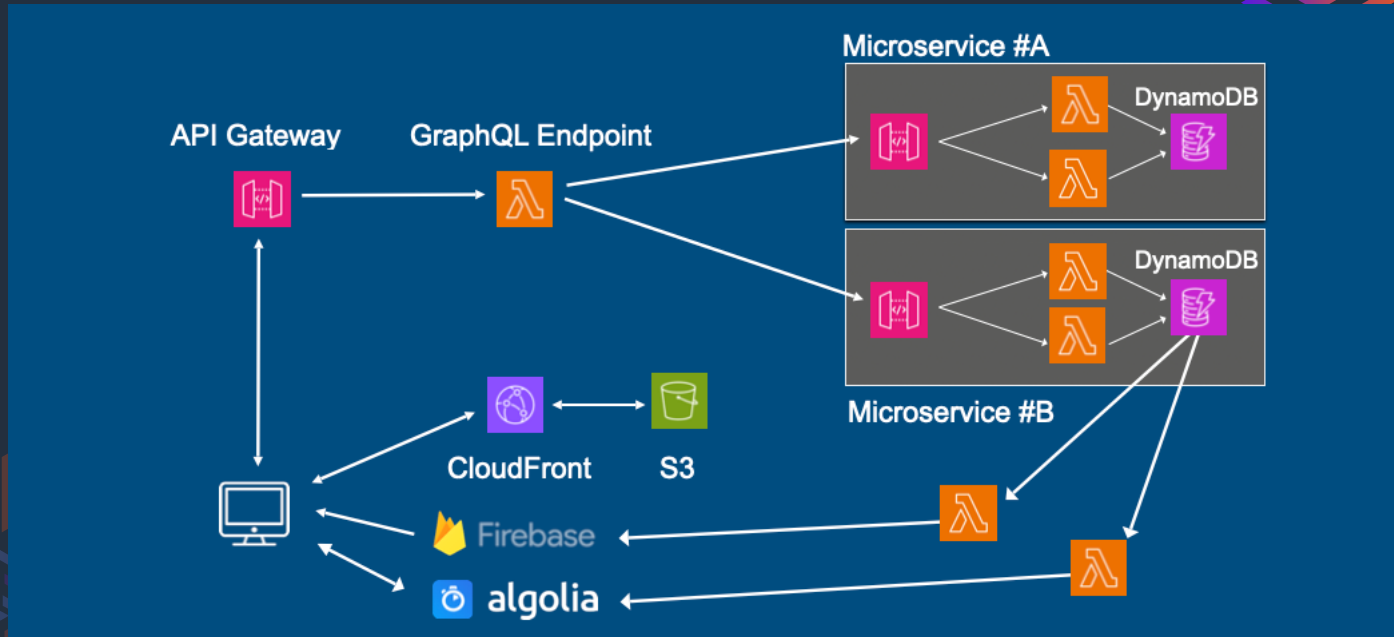
- AWS Serverless Hero
- Author of Serverless Architectures on AWS
- Co-organiser of the Melbourne Serverless Meetup
- Co-organiser ServerlessDays ANZ
- Former VP Education & Research at A Cloud Guru
- Former head of Serverlessconf
- Co-founder of heart hands





aws
COMMUNITY DAY
AOTEAROA

Teams of developers working in parallel





COMMUNITY DAY

AOTEAROA

And the costs weren't too bad either (2018)

289 Lambda Functions
19 Microservices
3.68TB of data in S3

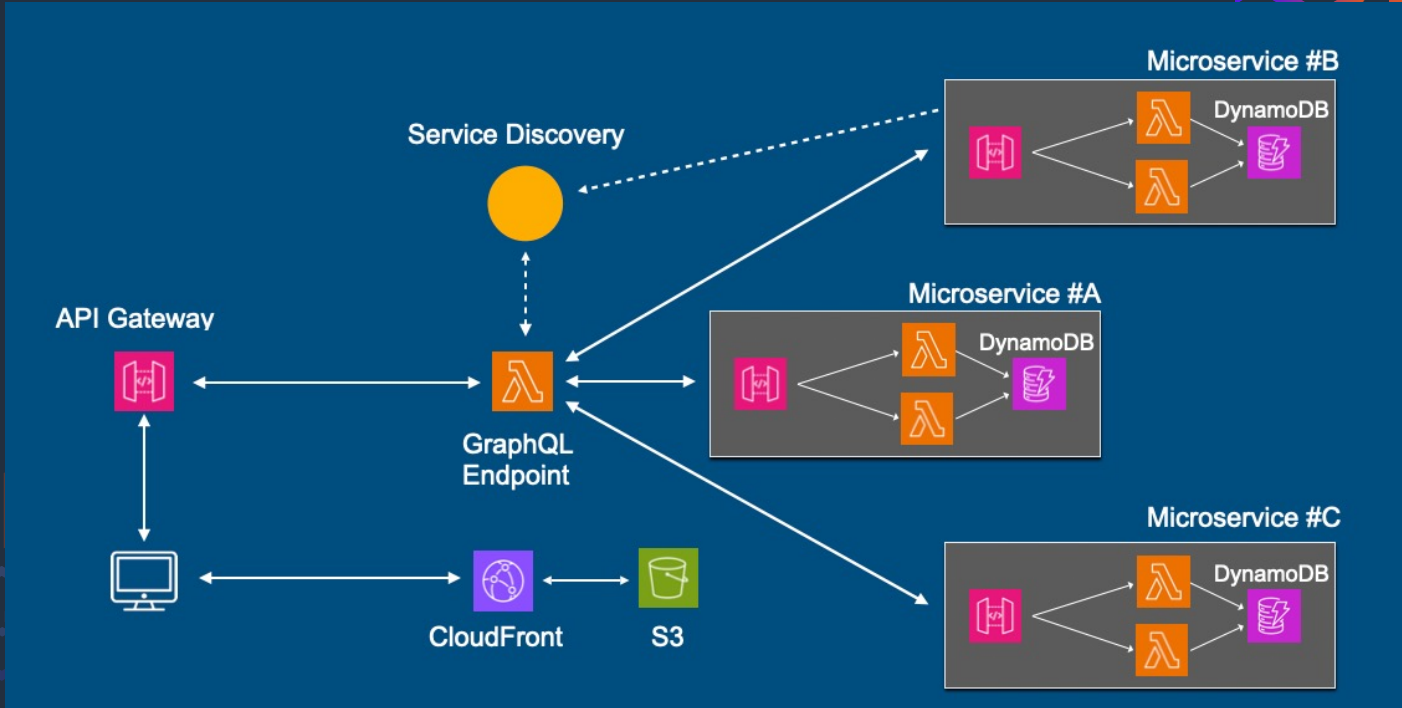
107m Lambda Invocations / month
45m API Requests / month
3.8+ TB of data via CloudFront per day

650K+ users

Service	Cost
Key Management Service	\$25.26
Simple Storage Service (S3)	\$108.23
Config	\$109.84
Elastic Transcoder	\$154.17
API Gateway	\$192.14
Developer Support	\$314.59
Redshift	\$371.50
DynamoDB	\$373.54
Lambda	\$706.49
CloudWatch	\$3,142.73
CloudFront	\$5,099.85

aws
COMMUNITY DAY
AOTEAROA

Gaining team efficiency and speed



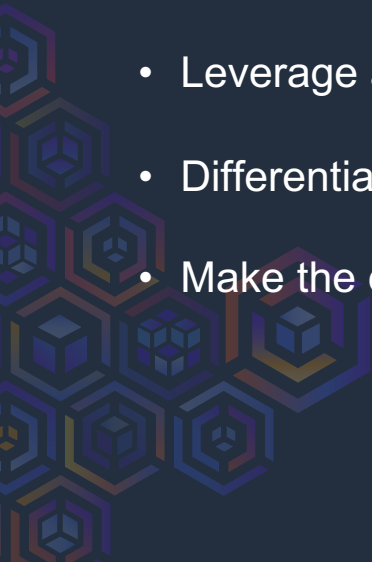
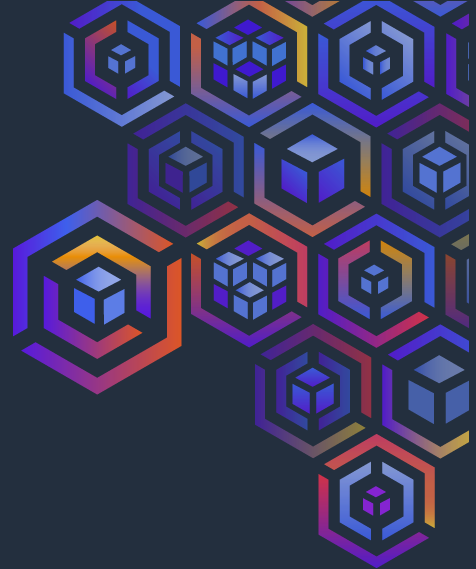




COMMUNITY DAY
AOTEAROA

How we build today: from a scaleup back to a startup

- Serverless first approach makes technical decisions easier
- Leverage as many AWS and third-party services as possible
- Differentiate with a magical user experience
- Make the development experience easy





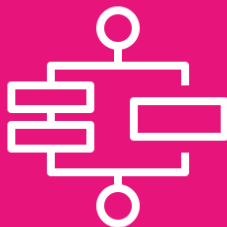
COMMUNITY DAY

AOTEAROA

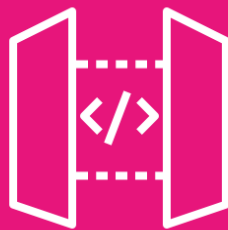
Core Services



Lambda



Step Functions



API Gateway



DynamoDB



EventBridge



S3

aws COMMUNITY DAY

AOTEAROA

Serverless Stack (SST)

Test your apps live

Set breakpoints in your Lambda functions and test your apps live. [Learn more >](#)

```
src > TS lambda.ts > ha
1 export function h
2   return {
3     statusCode: 200,
4     body: "Hello World"
5   };
6
7
```

Set breakpoints in your functions

Local

- Stacks
- Functions
- API
- DynamoDB
- RDS
- Buckets
- GraphQL
- Cognito

Invocations

Success	GET /integration/transactions
06:57:37.368	▶ "Request" : { ... } 8 items
06:57:37.738	Missing transactions for [] getting cached response false
06:57:38.216	▶ "Response" : { ... } 3 items
Success	GET /integration/transactions
06:57:37.306	▶ "Request" : { ... } 8 items
06:57:37.736	Missing transactions for [] getting cached response false
06:57:38.200	▶ "Response" : { ... } 3 items
Success	GET /integration/transactions
06:57:37.303	▶ "Request" : { ... } 8 items
06:57:37.738	Missing transactions for [] getting cached response false
06:57:38.215	▶ "Response" : { ... } 3 items
Success	GET /integration/transactions
06:57:37.300	▶ "Request" : { ... } 8 items
06:57:37.737	Missing transactions for [] getting cached response false
06:57:38.216	▶ "Response" : { ... } 3 items

pete-dev ←

aws COMMUNITY DAY

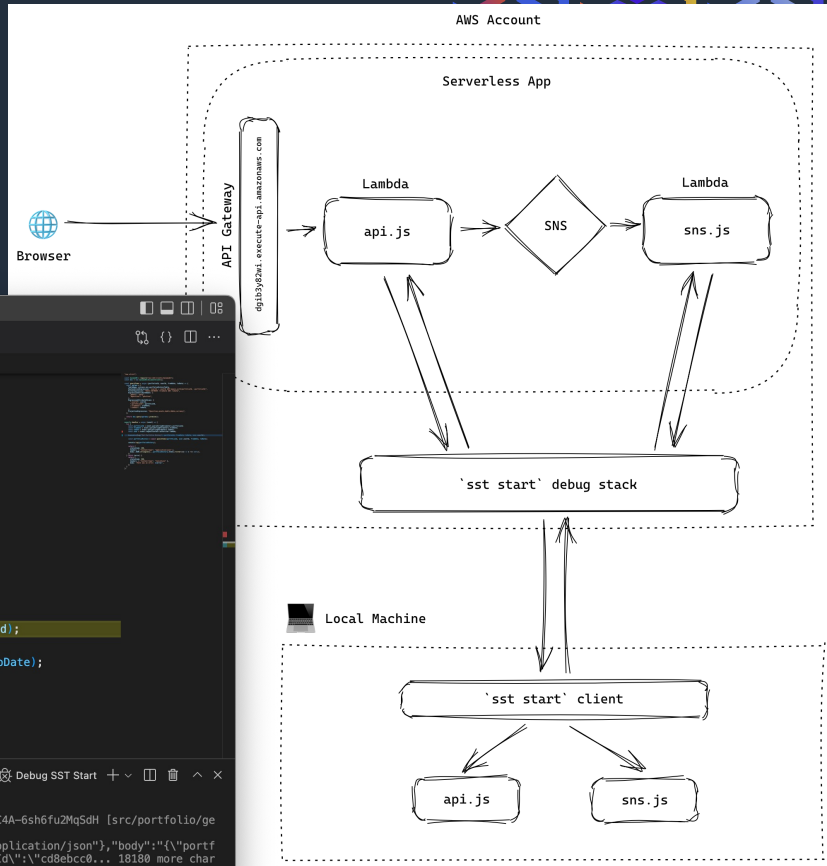
AOTEAROA

Serverless Stack (SST)

```
src > analytics > portfolio > JS lambda.js > handler > handler
19   "toDate": toDate,
20   },
21   ProjectionExpression: "#position,assets,debts,#date,currency",
22   });
23 };
24 return doc.query(params).promise();
25 };
26 };
27 exports.handler = async (event) => {
28   try {
29     const portfolioId = event.queryStringParameters.portfolioId;
30     const fromDate = event.queryStringParameters.fromDate;
31     const toDate = event.queryStringParameters.toDate;
32     const user = event.requestContext.authorizer.lambda;
33   } catch (err) {
34     console.log("Get Portfolio History", portfolioId, fromDate, toDate, user.userId);
35   }
36   const portfolioHistory = await queryItems(portfolioId, user.userId, fromDate, toDate);
37   console.log(portfolioHistory);
38 };
39 };
40 return {
41   statusCode: 200,
```

DEBUG CONSOLE OUTPUT:

```
"tenantId":"rbv68z4b"}
0749100b-d4c4-45b6-bbf8-0cbd9b4b21d7 REQUEST pete-dev-fat-fire-app-cor--ApiLambdaGETportfolioC4A-6sh6fu2Mq5dH [src/portfolio/get/lambda.handler] invoked by API GET /portfolio
0749100b-d4c4-45b6-bbf8-0cbd9b4b21d7 RESPONSE {"statusCode":200,"headers":{"Content-Type":"application/json"},"body":{"portfolioId":"0a05f3a0","\currency":"AUD","\updatedAt":"2022-08-31T06:56:31.223Z","\userId":"cd8ebcc0... 18180 more characters"}
1e0299bf-0120-40da-9d80-c7c622bac147 REQUEST pete-dev-fat-fire-app-cor--UserAuthorizerFn8FD7EFB-509EBH8tkPZZ [src/authorizer/user/lambda.handler] invoked by API GET /analytics/portfolio
1e0299bf-0120-40da-9d80-c7c622bac147 RESPONSE {"isAuthorized":true,"context":{"userId":"cd8ebcc0-153b-4912-8829-0a3b4a39fa68"},"tenantId":"rbv68z4b"}
595f6843-abd6-4b81-8df2-cae42cf71b8c REQUEST pete-dev-fat-fire-app-ana-LambdaGETanalyticsportfo-9oX4Pugh8Idf [src/analytics/portfolio/lambda.handler] invoked by API GET /analytics/portfolio
Debugger attached.
```

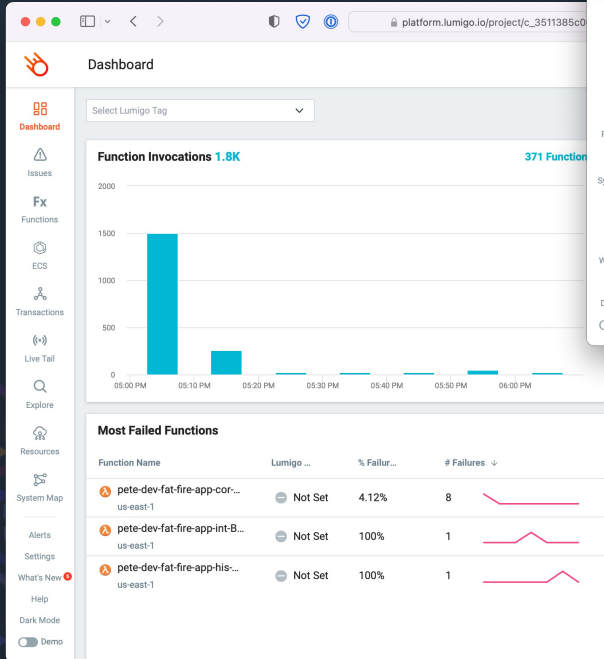




COMMUNITY DAY

AOTEAROA

Observability: Lumigo



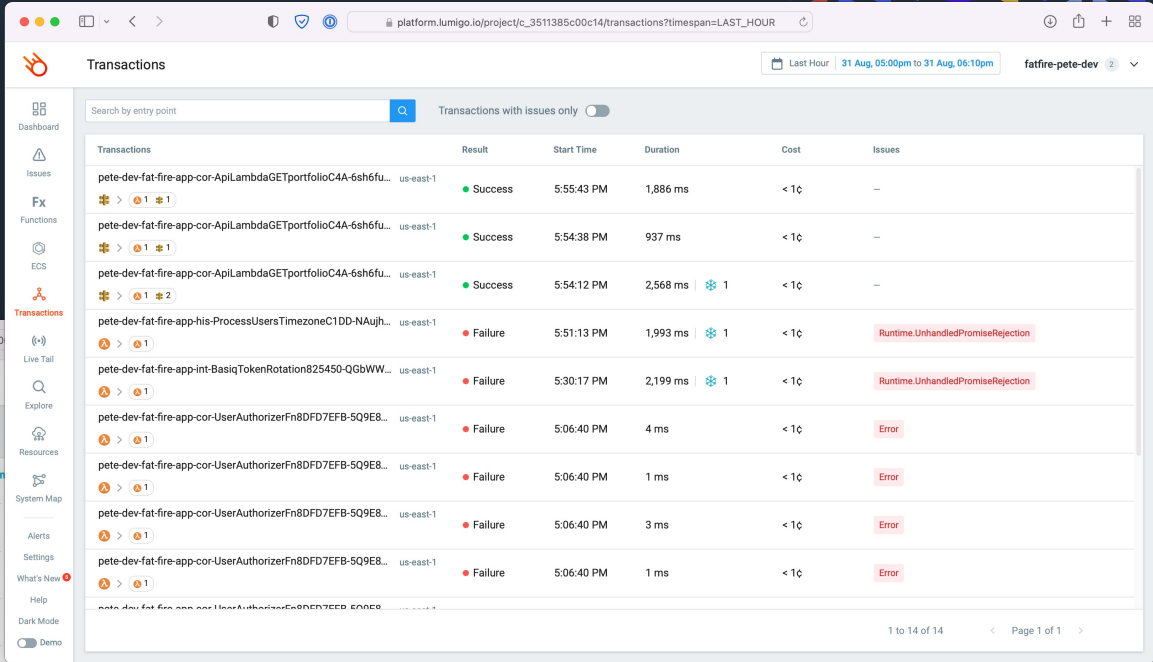
Dashboard

Select Lumigo Tag

Function Invocations 1.8K 371 Function

Most Failed Functions

Function Name	Lumigo ...	% Failur...	# Failures ↓
pete-dev-fat-fire-app-cor-... us-east-1	Not Set	4.12%	8
pete-dev-fat-fire-app-int-B... us-east-1	Not Set	100%	1
pete-dev-fat-fire-app-his... us-east-1	Not Set	100%	1



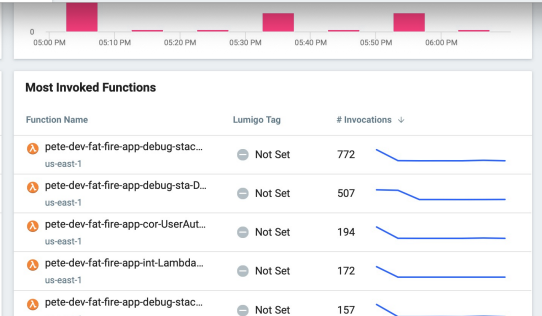
Transactions

Last Hour 31 Aug, 05:00pm to 31 Aug, 06:10pm fatfire-pete-dev

Search by entry point Transactions with issues only

Transactions	Result	Start Time	Duration	Cost	Issues
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:55:43 PM	1,886 ms	< 1c	—
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:54:38 PM	937 ms	< 1c	—
pete-dev-fat-fire-app-cor-ApiLambdaGETportfolioC4A-6sh6fu... us-east-1	Success	5:54:12 PM	2,568 ms	< 1c	—
pete-dev-fat-fire-app-his-ProcessUsersTimezoneC1DD-NAujh... us-east-1	Failure	5:51:13 PM	1,993 ms	< 1c	Runtime.UnhandledPromiseRejection
pete-dev-fat-fire-app-int-BasiqTokenRotation825450-QG6WW... us-east-1	Failure	5:30:17 PM	2,199 ms	< 1c	Runtime.UnhandledPromiseRejection
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	4 ms	< 1c	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	1 ms	< 1c	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	3 ms	< 1c	Error
pete-dev-fat-fire-app-cor-UserAuthorizerFn8DFD7EFB-5Q9E8... us-east-1	Failure	5:06:40 PM	1 ms	< 1c	Error

1 to 14 of 14 Page 1 of 1



Most Invoked Functions

Function Name	Lumigo Tag	# Invocations ↓
pete-dev-fat-fire-app-debug-stac... us-east-1	Not Set	772
pete-dev-fat-fire-app-debug-sta-D... us-east-1	Not Set	507
pete-dev-fat-fire-app-cor-UserAut... us-east-1	Not Set	194
pete-dev-fat-fire-app-int-Lambda... us-east-1	Not Set	172
pete-dev-fat-fire-app-debug-stac... us-east-1	Not Set	157





COMMUNITY DAY

AOTEAROA

Microsoft Guidance vs LangChain

```
# we use LLaMA here, but any GPT-style model will do
llama = guidance.llms.Transformers("your_path/llama-7b", device=0)

# we can pre-define valid option sets
valid_weapons = ["sword", "axe", "mace", "spear", "bow", "crossbow"]

# define the prompt
character_maker = guidance("""The following is a character profile for an RPG game
```json
{
 "id": "{{id}}",
 "description": "{{description}}",
 "name": "{{gen 'name'}}",
 "age": {{gen 'age' pattern='[0-9]+' stop=','}},
 "armor": "{{#select 'armor'}}leather{or}chainmail{or}plate{/select}}",
 "weapon": "{{select 'weapon' options=valid_weapons}}",
 "class": "{{gen 'class'}}",
 "mantra": "{{gen 'mantra' temperature=0.7}}",
 "strength": {{gen 'strength' pattern='[0-9]+' stop=','}},
 "items": [{{#geneach 'items' num_iterations=5 join=','}}"{{gen 'this' temperature=0.7}}"/{{/g
}}"""])

generate a character
character_maker(
 id="e1f491f7-7ab8-4dac-8c20-c92b5e7d883d",
 description="A quick and nimble fighter.",
 valid_weapons=valid_weapons, llama=llama
)
```

```
from langchain.prompts import ChatPromptTemplate
from langchain.prompts.chat import SystemMessage, HumanMessagePromptTemplate

template = ChatPromptTemplate.from_messages(
 [
 SystemMessage(
 content=(
 "You are a helpful assistant that re-writes the user's text to "
 "sound more upbeat."
)
),
 HumanMessagePromptTemplate.from_template("{text}"),
]
)

from langchain.chat_models import ChatOpenAI

llm = ChatOpenAI()
llm(template.format_messages(text='i dont like eating tasty things.'))
```

# aws

# COMMUNITY DAY

AOTEAROA

The screenshot shows the AWS Bedrock console interface. The browser address bar displays the URL: `us-west-2.console.aws.amazon.com/bedrock/home?region=us-west-2#/providers?model=`. The console header includes the AWS logo, a search bar, and the user's name, Peter Sbarski. The left-hand navigation pane is titled "Amazon Bedrock" and lists various sections: Overview, Examples, Foundation models, Models, Providers (highlighted), Embeddings, Playgrounds, Text, Image, and Terms of service. The main content area is titled "Model providers" and includes an "Info" link. Below the title, there are tabs for different providers: AI21 Labs, Amazon, Anthropic (selected), and Stability AI. The "Provider overview" section for Anthropic states: "Anthropic is an AI safety and research company that's working to build reliable, interpretable, and steerable AI systems." The "Models" section lists "Claude v1.3", "Claude v2" (selected), and "Claude Instant v1.1". A description for Claude v2 notes: "Anthropic offers the Claude family of large language models purpose built for conversations, summarization, Q&A, workflow automation, coding and more. Early customers report that Claude is much less likely to produce harmful outputs, easier to converse with, and more steerable - so you can get your desired output with less effort. Claude can also take direction on personality, tone, and behavior." A button labeled "Open in playground" is visible. At the bottom, there are links for "Model version v2", "Supported use cases" (Question answering, information extraction), and "Resource Prompt Design".



aws  
**COMMUNITY DAY**  
AOTEAROA

- Security/compliance first
- Use microservices
- Serverless where possible
- CI/CD
- Monitor, monitor, monitor!
- <https://youtu.be/IPOvrK3S3gQ>
- Serverless monoliths can be OK!
- Automation is a must
- Think through your testing strategy
- Go Serverless to enable experimentation and evolution
- Have a technical strategy

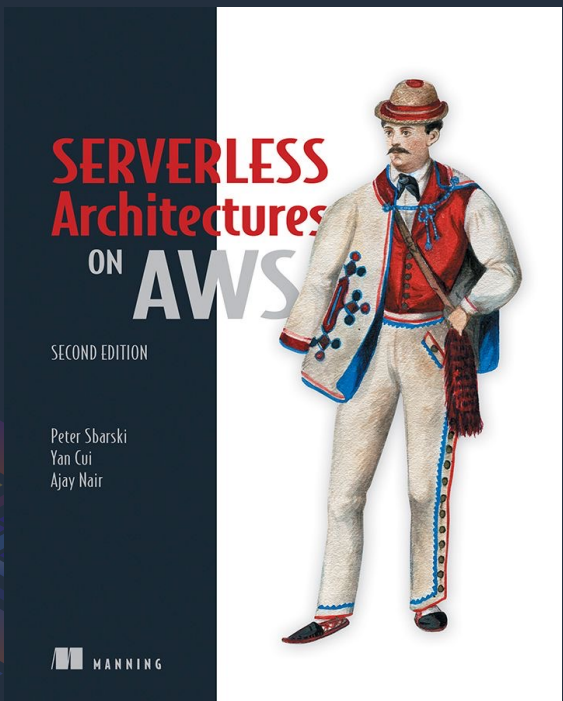




# COMMUNITY DAY

AOTEAROA

Thank you!



**Serverless Architectures on AWS:**  
<https://www.manning.com/books/serverless-architectures-on-aws-second-edition>

**The Value Flywheel Effect:**  
<https://itrevolution.com/product/the-value-flywheel-effect/>





# COMMUNITY DAY

AOTEAROA