
aws
COMMUNITY DAY
AOTEAROA

aws
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.

https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

aws
COMMUNITY DAY
AOTEAROA

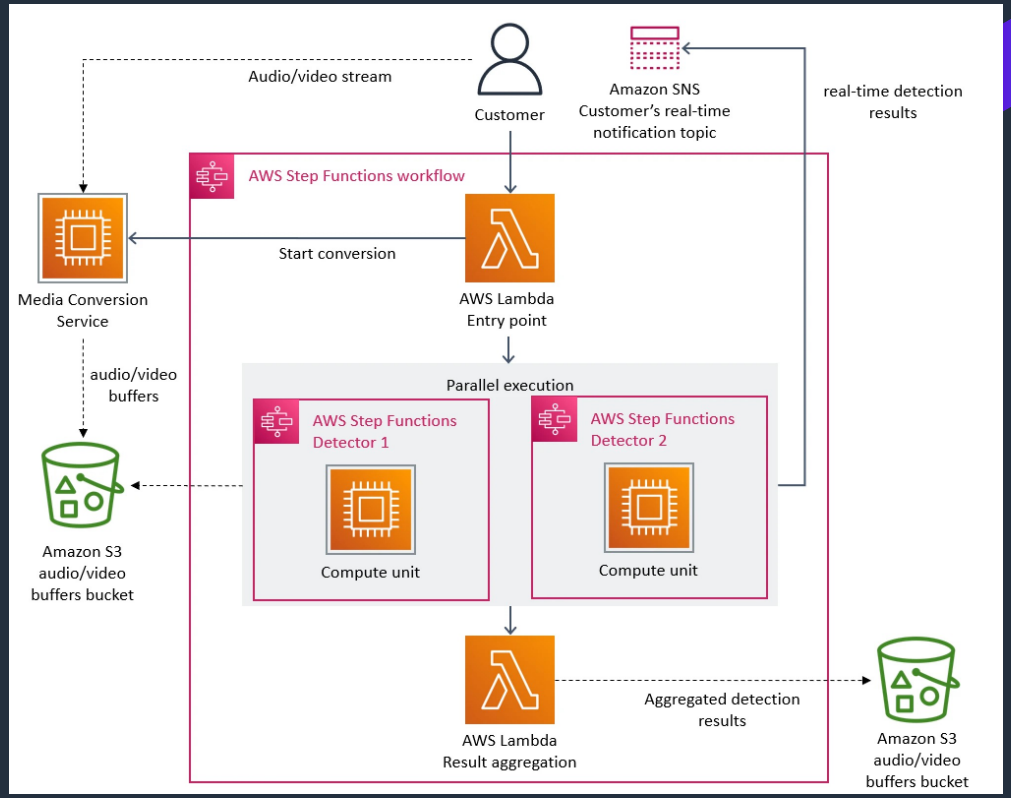
Convert video streams to frames/decrypt audio

Monitoring the quality of every video stream

Analyse frames or audio buffers for defects

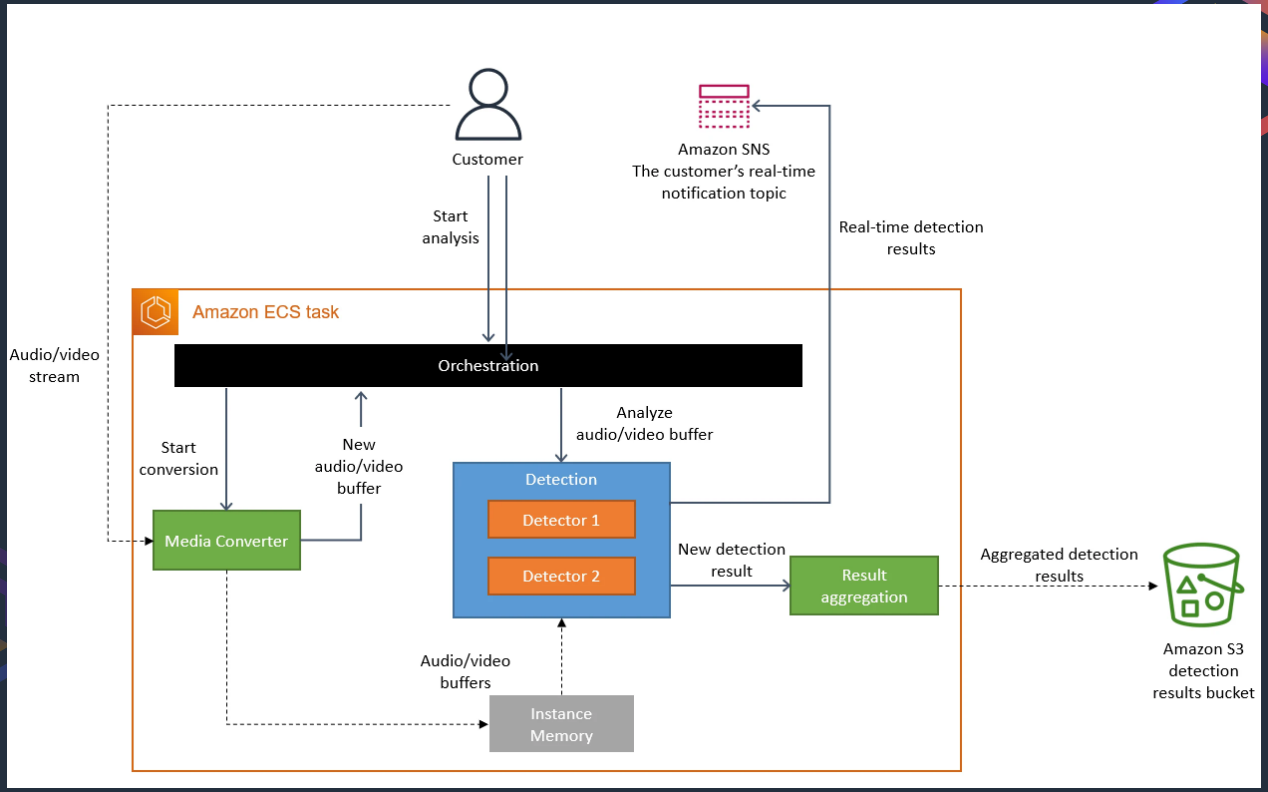
Send real-time notifications

aws
COMMUNITY DAY
AOTEAROA



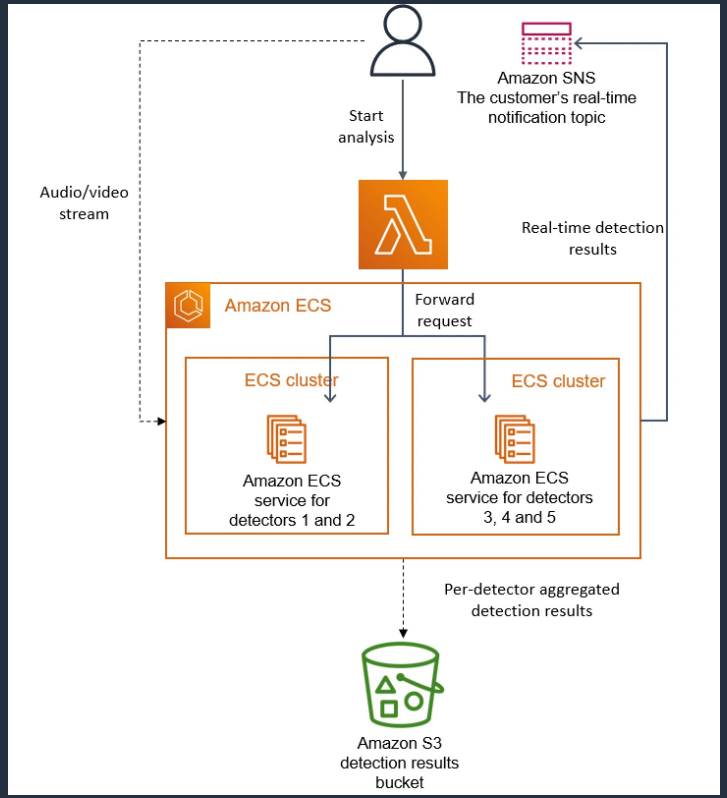

https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

aws
COMMUNITY DAY
AOTEAROA




https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

aws
COMMUNITY DAY
AOTEAROA




https://www.primevideotech.com/video-streaming/scaling-up-the-prime-video-audio-video-monitoring-service-and-reducing-costs-by-90

aws
COMMUNITY DAY
AOTEAROA

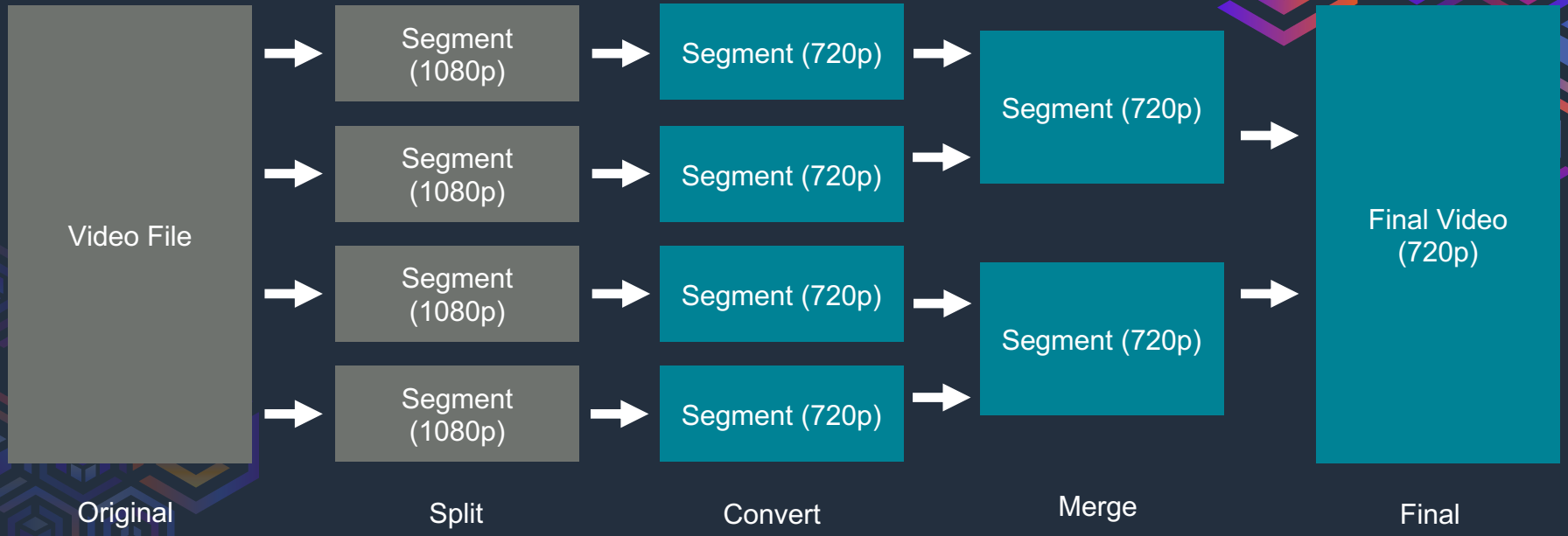
# So many bad takes - what is there to learn from the Prime Video microservices to monolith story

| Containers | Serverless |
| --- | --- |
| Custom code and services | Serverless events and functions |
| Lots of choices of frameworks and API mechanisms | Standardized choices |
| Where needed, optimize serverless applications by also building services using containers to solve for<br>• Lower startup latency<br>• Long running compute jobs<br>• Predictable high traffic | Combine these building blocks<br>λ AWS Lambda<br>API Gateway<br>Amazon SNS, SQS<br>Amazon DynamoDB<br>AWS Step Functions |

https://adrianco.medium.com/so-many-bad-takes-what-is-there-to-learn-from-the-prime-video-microservices-to-monolith-story-4bd0970423d4

aws
COMMUNITY DAY
AOTEAROA

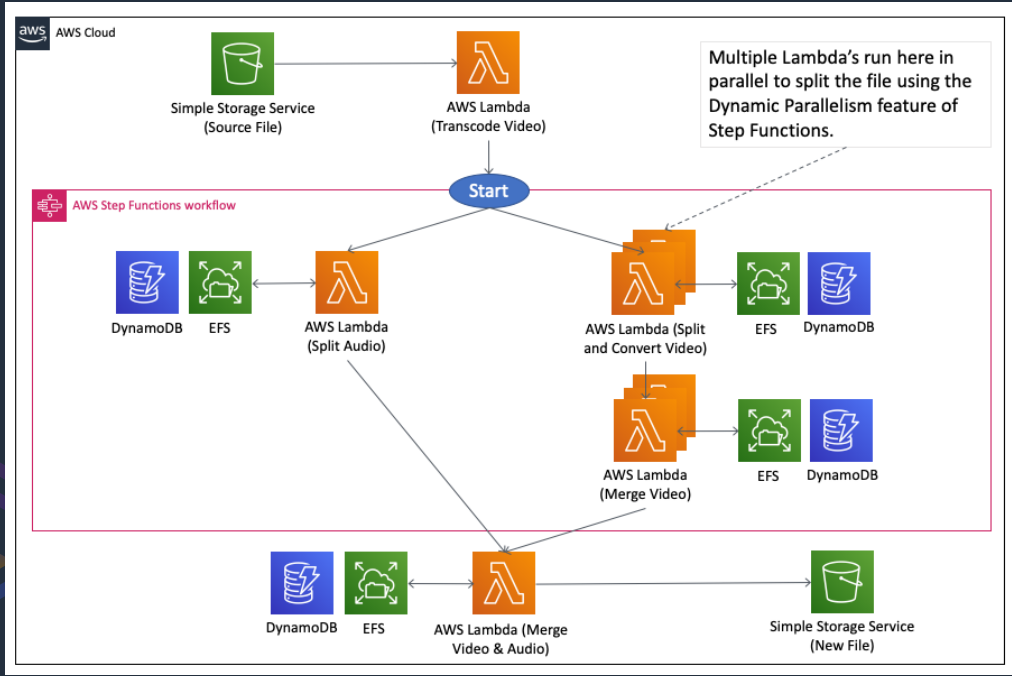
# Divide and conquer




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

aws
COMMUNITY DAY
AOTEAROA

# Parallel Computing with Lambda & Step Functions




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

aws
COMMUNITY DAY
AOTEAROA

## Serverless Video Transcoder

| | Serverless Lambda | Traditional EC2 (t2.large) | MacBook Pro 16GB 3.5GHz i7 |
|---|---|---|---|
| 34MB MP4 (00:43, 1920×1080) | 11 seconds | 32 seconds | 18 seconds |
| 77MB MP4 (6:49, 2048×1152) | 26 seconds | 144 seconds | 78 seconds |
| 100MB MP4 (59:56, 1280×720) | 86 seconds | 1073 seconds | 592 seconds |
| 350MB MP4 (07:45, 2560×1440) | 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 |

Read more: https://bit.ly/3wJOdvO

aws
COMMUNITY DAY
AOTEAROA

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

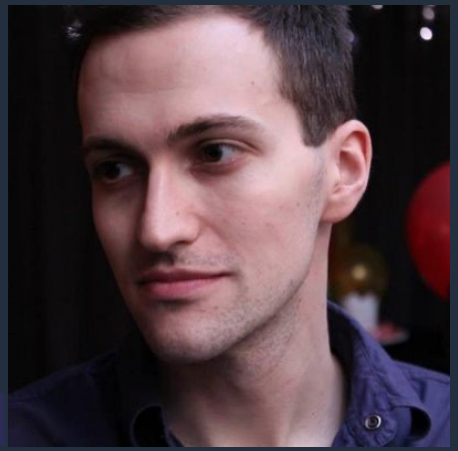
Sadjad Fouladi[1], Riad S. Wahby[1], Brennan Shacklett[1],
Karthikeyan Vasuki Balasubramaniam[2], William Zeng[1],
Rahul Bhalerao[2], Anirudh Sivaraman[3], George Porter[2], Keith Winstein[1]

[1]*Stanford University,* [2]*UC San Diego,* [3]*MIT*

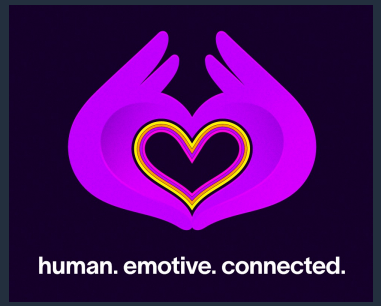
aws

# COMMUNITY DAY

AOTEAROA

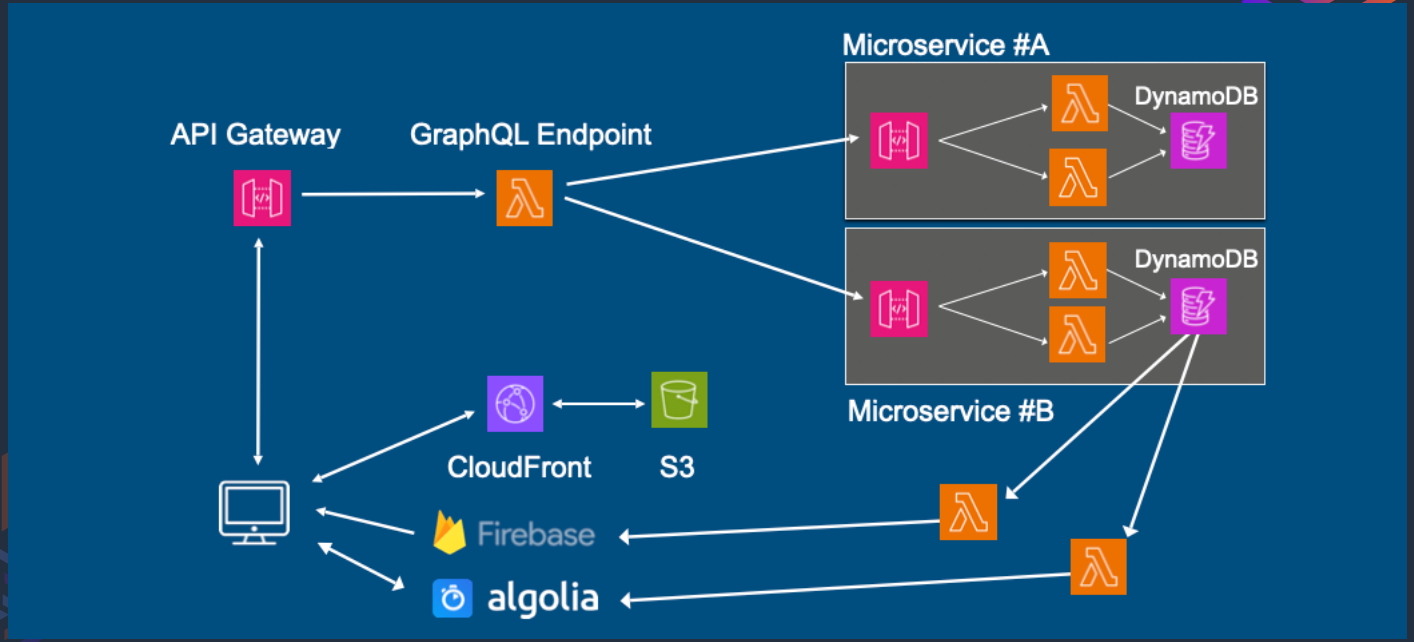



- AWS Serverless Hero
- Author of Serverless Architectures on AWS
- Co-organiser of the Melbourne Serverless Meetup
- Co-organiser ServerlessDays ANZ
- Former VP Education & Research at A Cloud Guru
- Former head of Serverlessconf
- Co-founder of heart hands



aws
COMMUNITY DAY
AOTEAROA

Teams of developers working in parallel



aws
COMMUNITY DAY
AOTEAROA

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

289 Lambda Functions
19 Microservices
3.68TB of data in S3

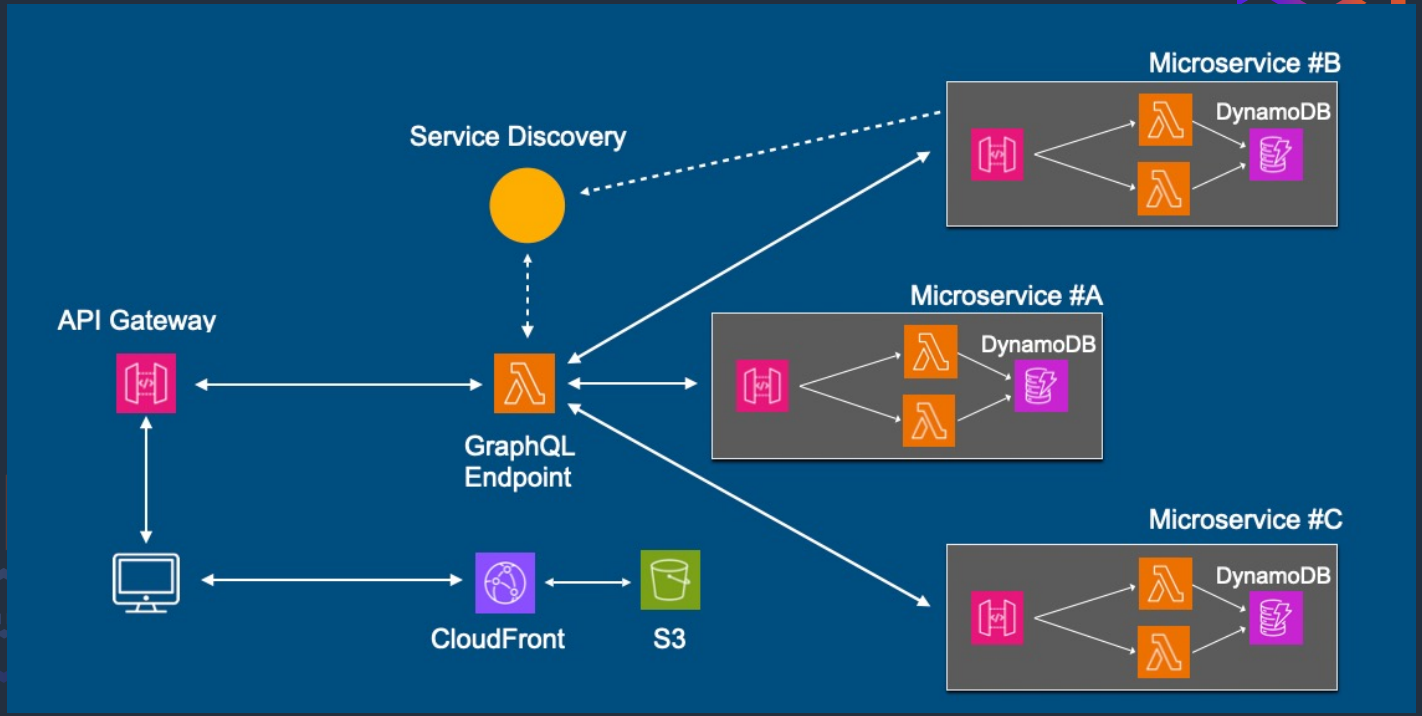
107m Lambda Invocations / month
45m API Requests / month
3.8+ TB of data via CloudFront per day

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



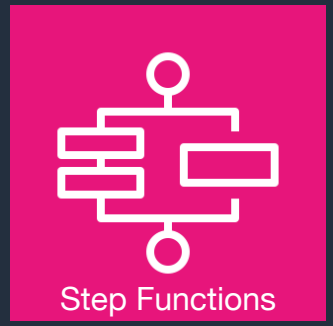
aws
COMMUNITY DAY
AOTEAROA

# 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

aws
COMMUNITY DAY
AOTEAROA

# Core Services



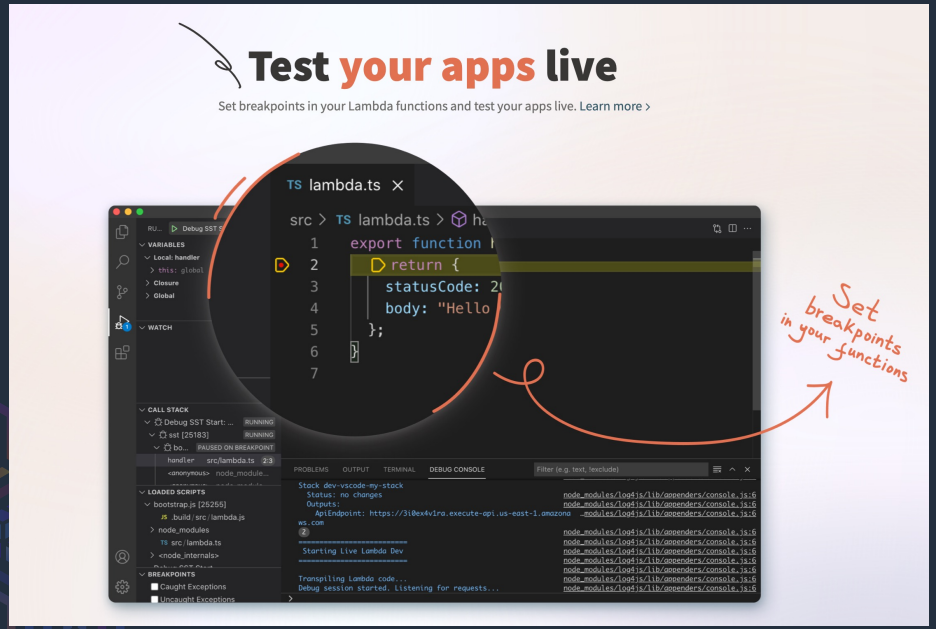


aws

# COMMUNITY DAY

AOTEAROA

## Serverless Stack (SST)

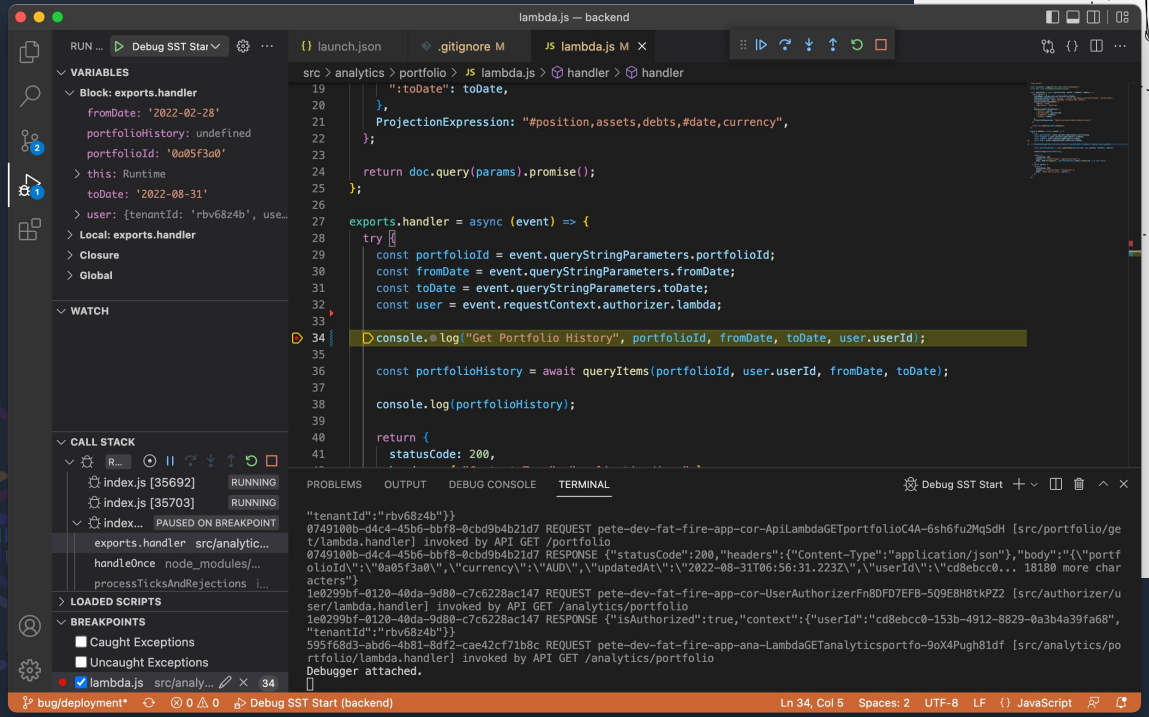


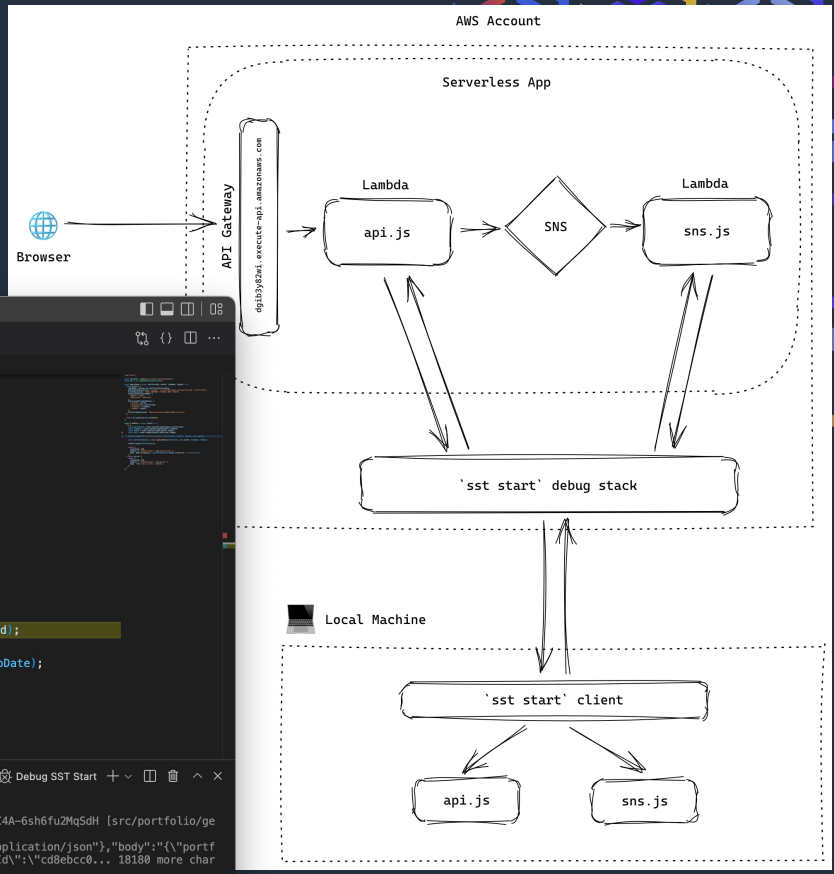

aws

# COMMUNITY DAY

AOTEAROA

## Serverless Stack (SST)

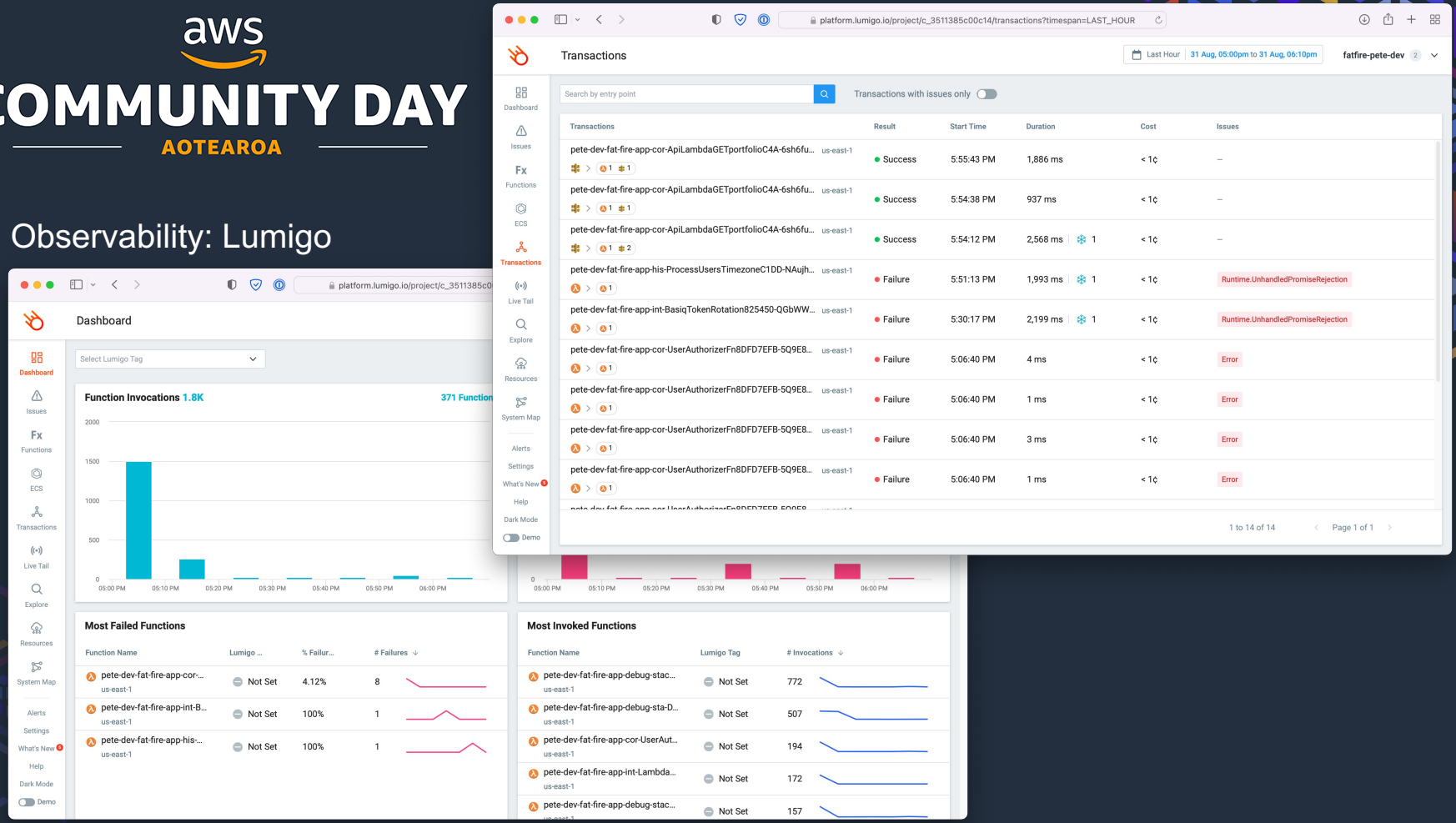


aws

# COMMUNITY DAY AOTEAROA

## Observability: Lumigo



aws

# 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 gam
```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, llm=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}"),
    ]
)

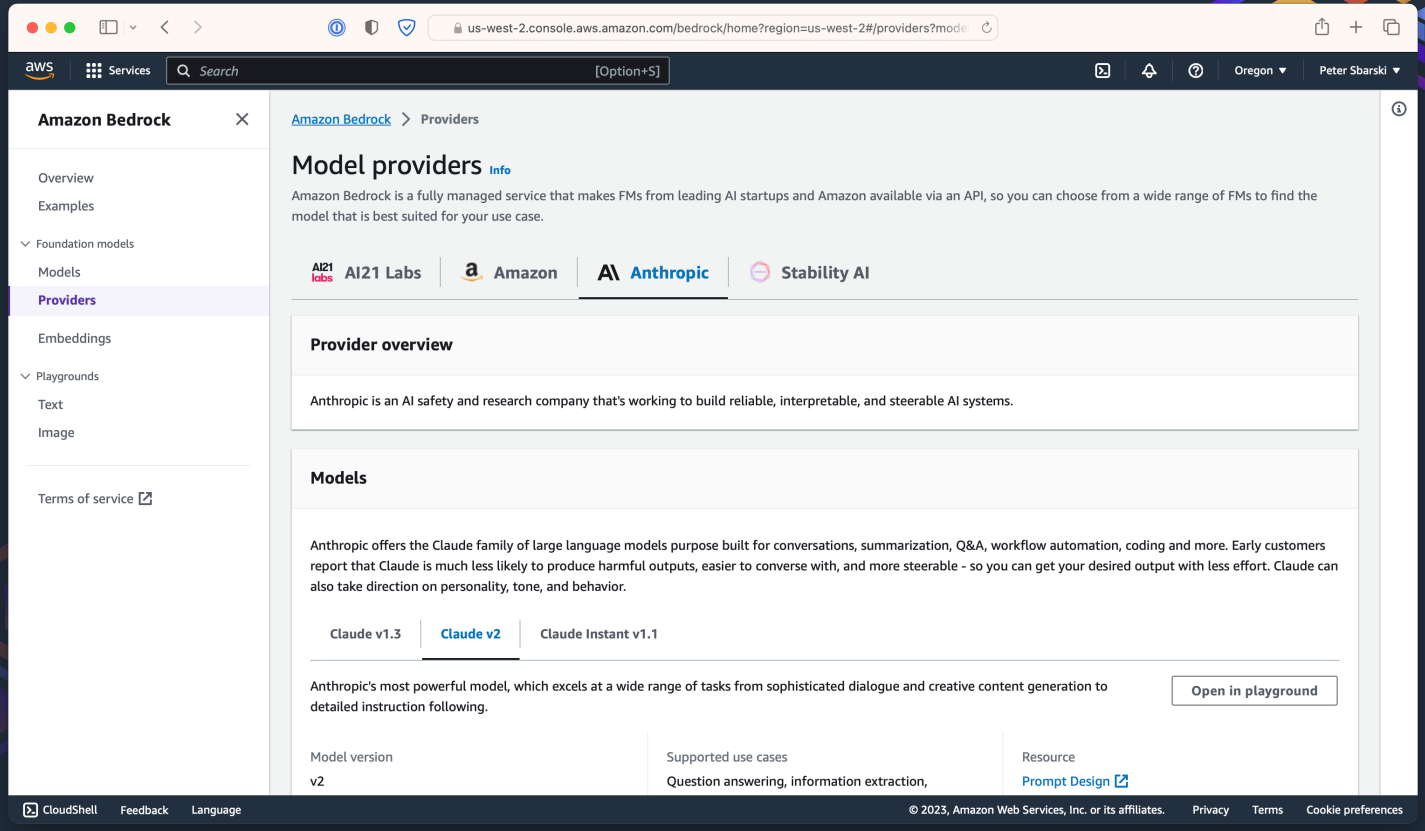
from langchain.chat_models import ChatOpenAI

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

aws

# COMMUNITY DAY

AOTEAROA



aws
COMMUNITY DAY
AOTEAROA

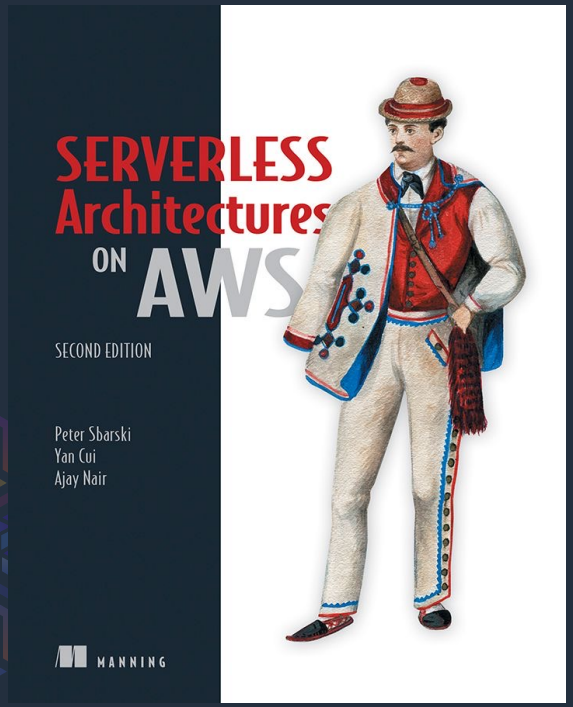
- Security/compliance first
- Use microservices
- Serverless where possible
- CI/CD
- Monitor, monitor, monitor!
- https://youtu.be/IPOvrK3S3gQ
- Serverless monoliths can be OK!
- Automation is a must
- Think through your testing strategy
- Go Serverless to enable experimentation and evolution
- Have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

aws

# 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/


aws
COMMUNITY DAY
AOTEAROA