Lambda Stories from the Trenches

What worked and what didn't work for us using AWS Lambda

@rantav
Yodas
(now AppsFlyer)

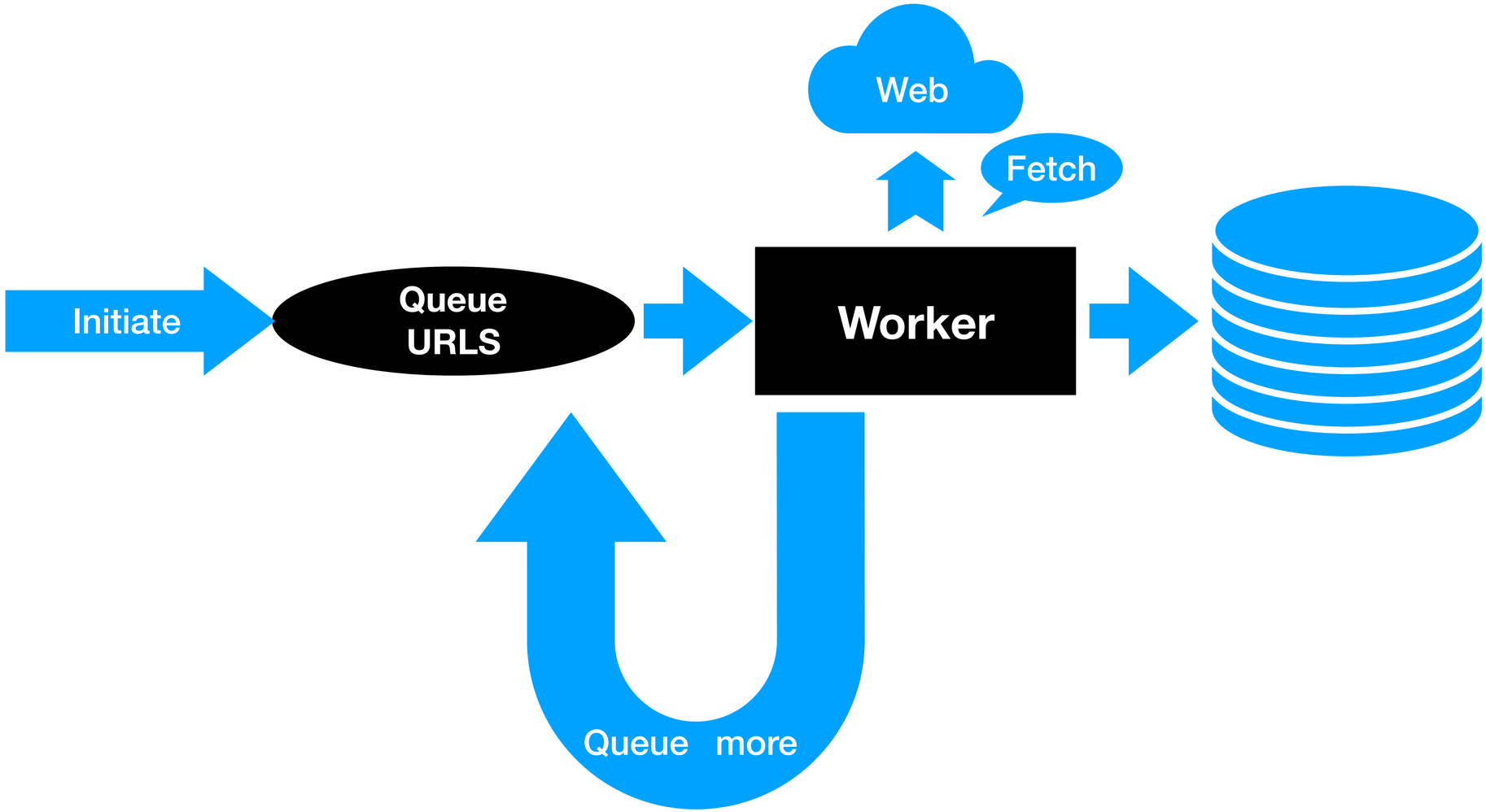
whoami

- Yodas cofounder and Janitor
- Now at AppsFlyer
- Reversim Podcast
- Reversim Summit (Oct 8-9)
- … been coding Lambda since early 2016

A tale of two Lambdas

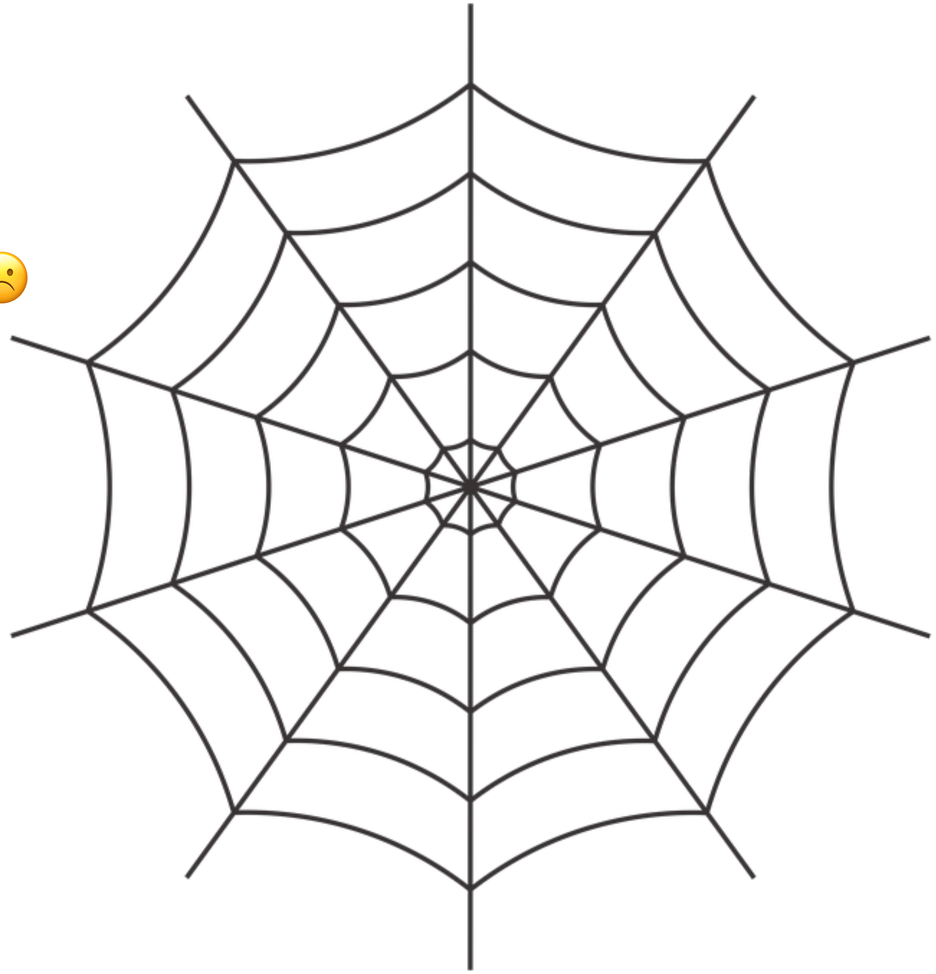
- The Happy Lambda
- The Sad Lambda

How a web crawler works (simplified)



Web Crawler Properties

- Typical page: seconds → sometimes minutes ☹️
- Failures / Retries ↺
- Easily distributed 👍
- Throttle 🤫



Web Crawler on AWS Lambda

- Easy 👍
- Function: Fetch
  - Input: URL
- But wait, no SQS support!
- Go + Apex

Shortcomings

- Async invocation 👎
- Throttling 👎
- Concurrency 👎

Crawler on Lambda Take 2

- Concurrency:
  - Micro Batching
  - But rough edges

Crawler on Lambda: Challenges

- Cost
- Throttling
- SQS

Crawler on Lambda: Conclusion

Solution: Elastic Beanstalk workers

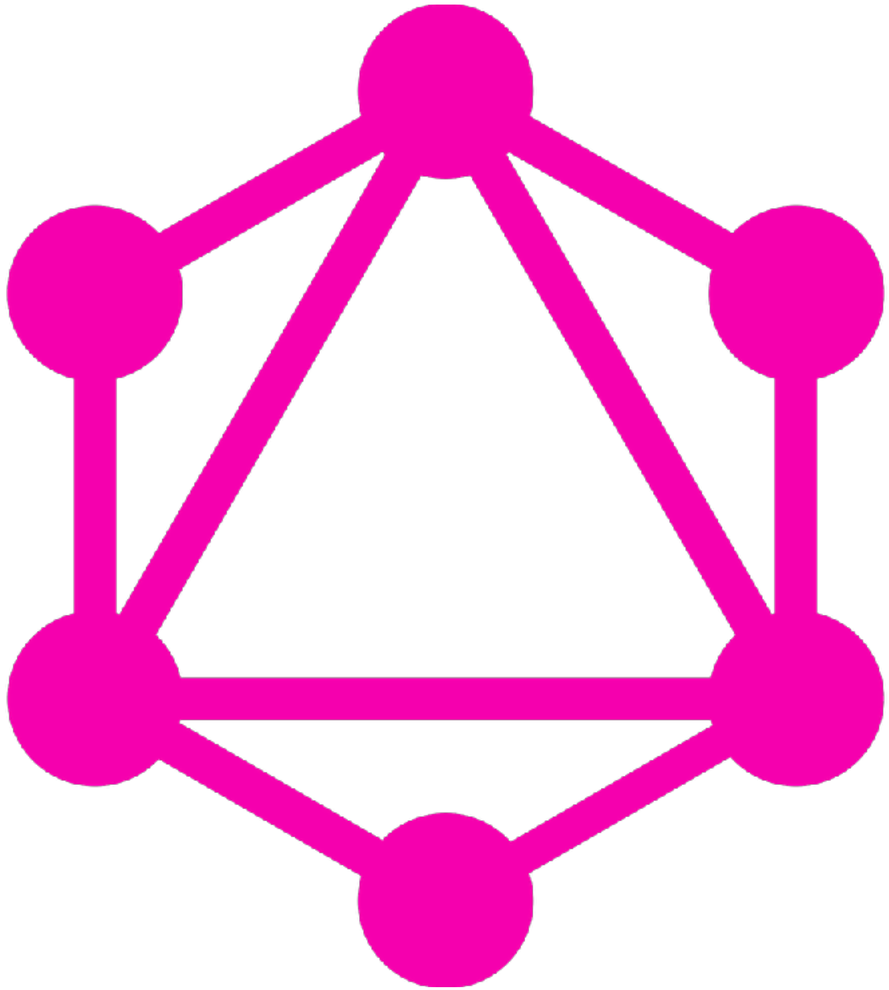
- Worker mode works with SQS
- Spots
- Scale and Concurrent
- Capacity and Throttling support.

API on Lambda

- The stack:
  - API GW
  - Go + Apex
  - Terraform
  - GraphQL

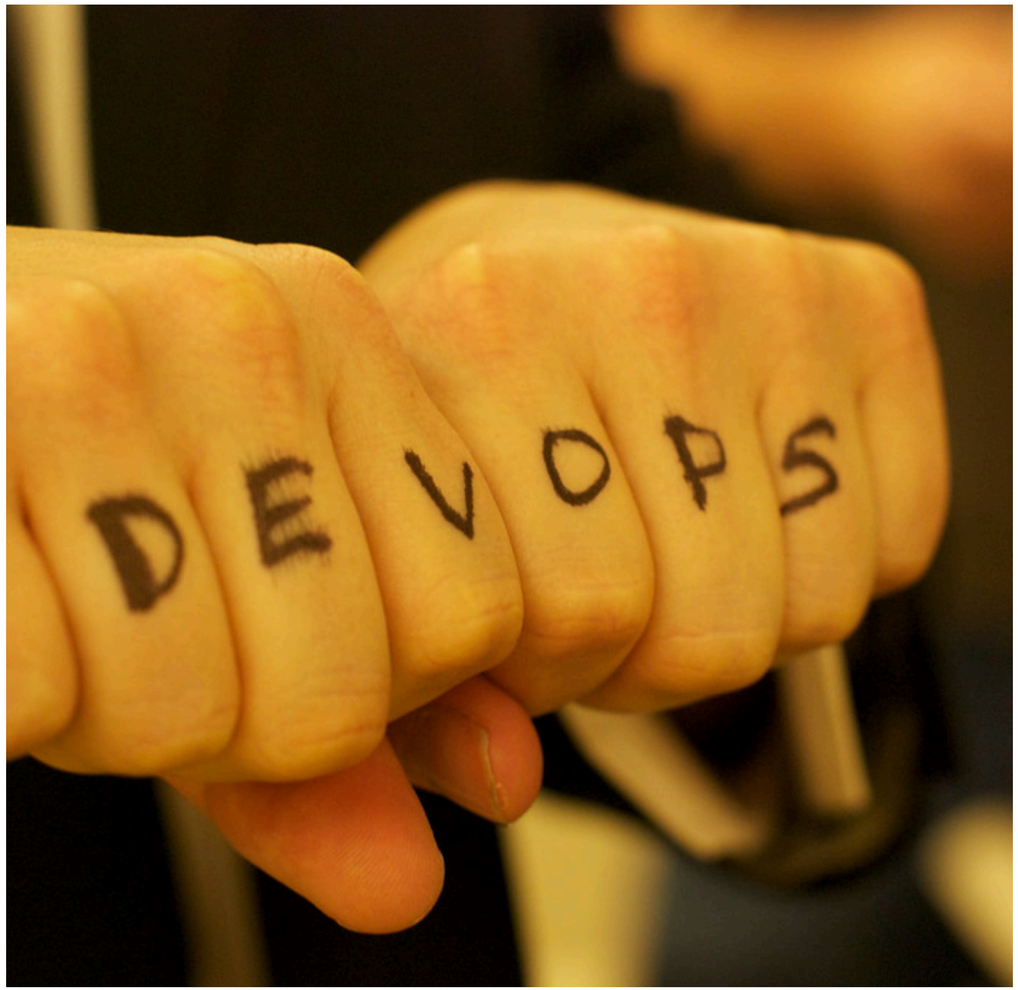
Why GraphQL?

- Clients love it
- Easy to maintain a monolith
  - Wait, what?



Observation: Ops @ Lambda

- NoOps?
  - No, Ops



API on Lambda: Challenges

- Latency
  - Cold start
  - API GW
  - Conn pooling
- Visibility
  - Logging
  - Monitoring

Latency Mitigation

- Delayed/background jobs
- Conn pooling when possible
- No premature initialization

Conclusion

- Lambda sounds good, in theory
  - Reality: practical limitations
- GraphQL is a nice fit for API atop FaaS
- API: If you can afford the latency, it’s worth it