

# **Towards a serverless event-sourced Nordstrom**

Rob Gruhl, Senior Manager Nordstrom Technology

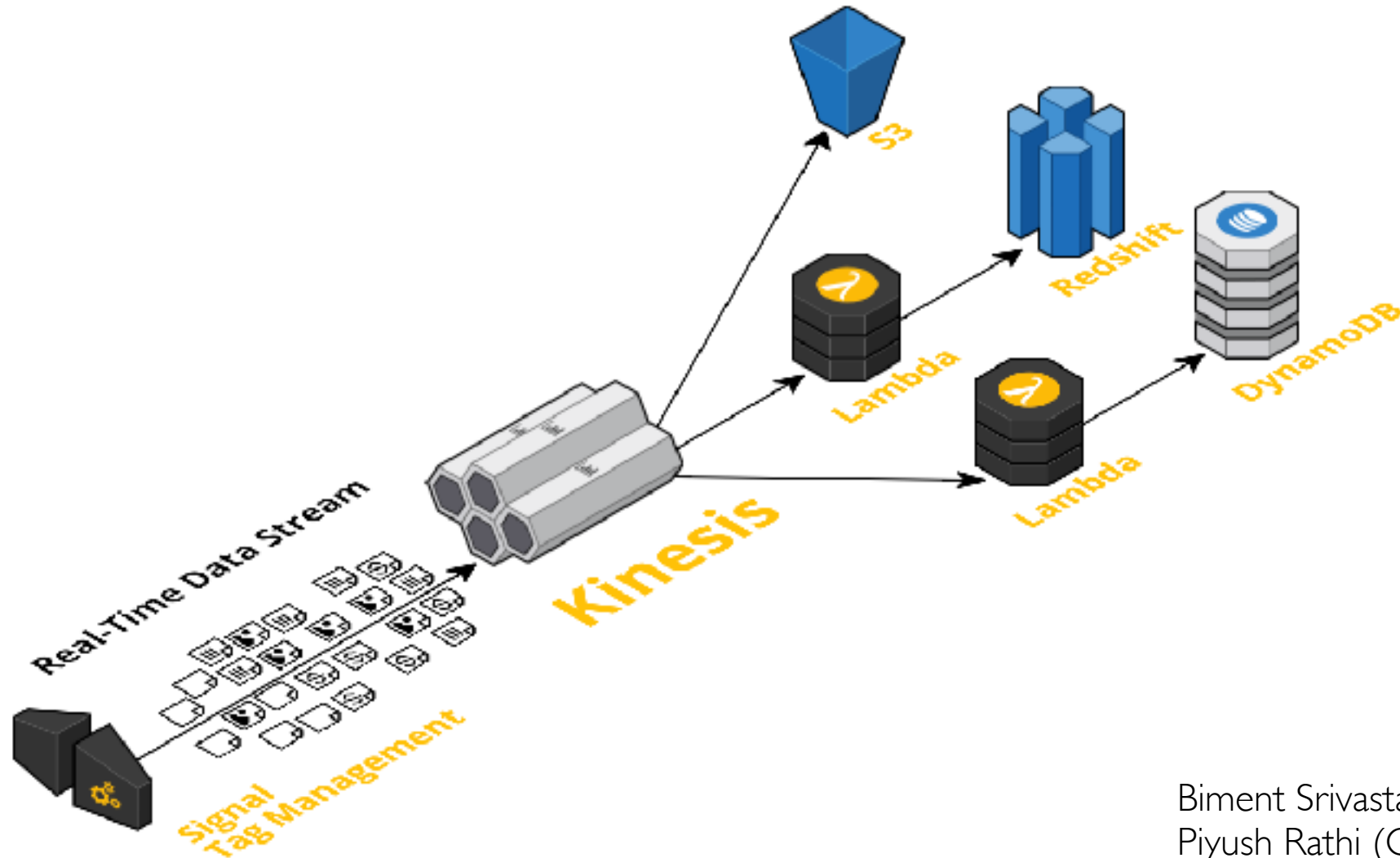
Ashmi Bhanushali  
Erik Erikson  
Greg Smith  
Lauren Wang  
Sayna Parsi





# **A brief history of serverless event-sourcing at Nordstrom**

# March 2015: Real-time recently & frequently viewed



Biment Srivastav  
Piyush Rathi (QA)  
Inspired by Nordstrom Datalabs

# Serverless.com framework open source contributions



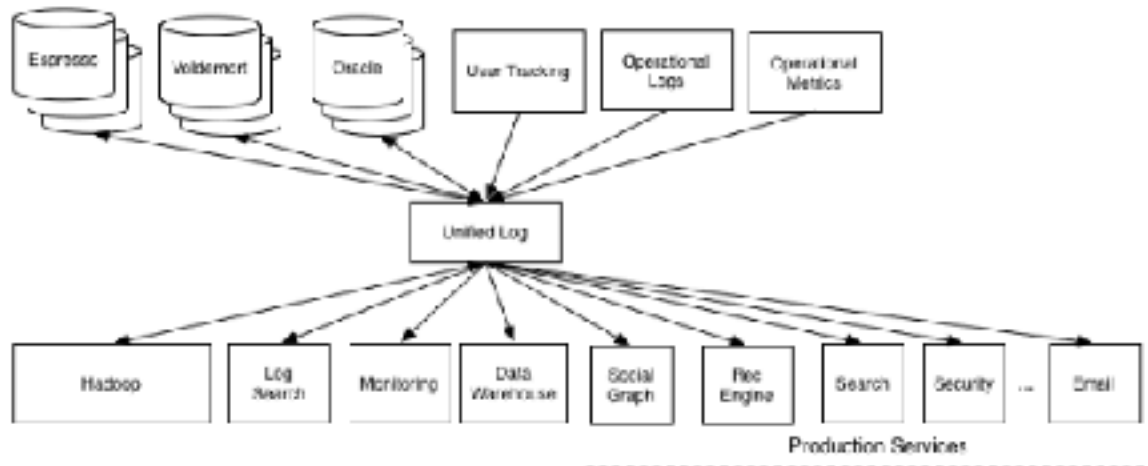
Erik Erikson  
Greg Smith

**Lots of serverless functions, and functions love events.**

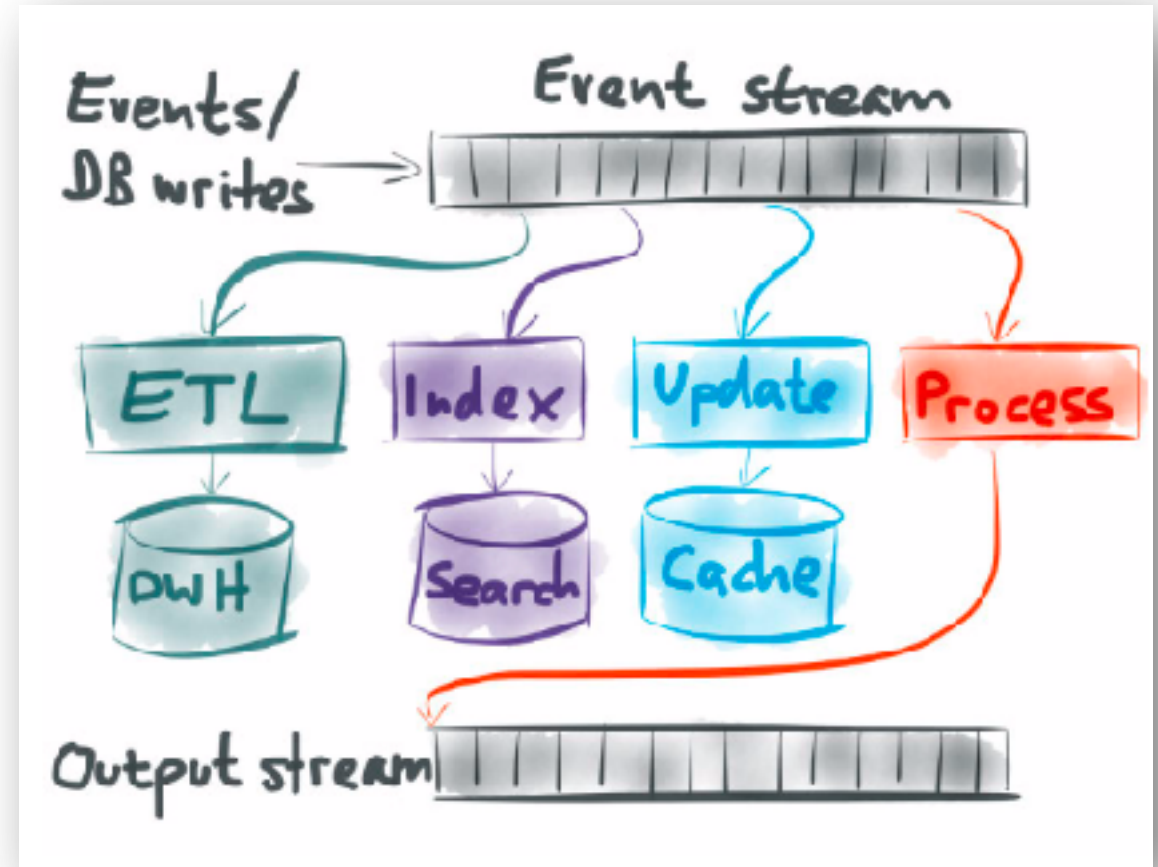
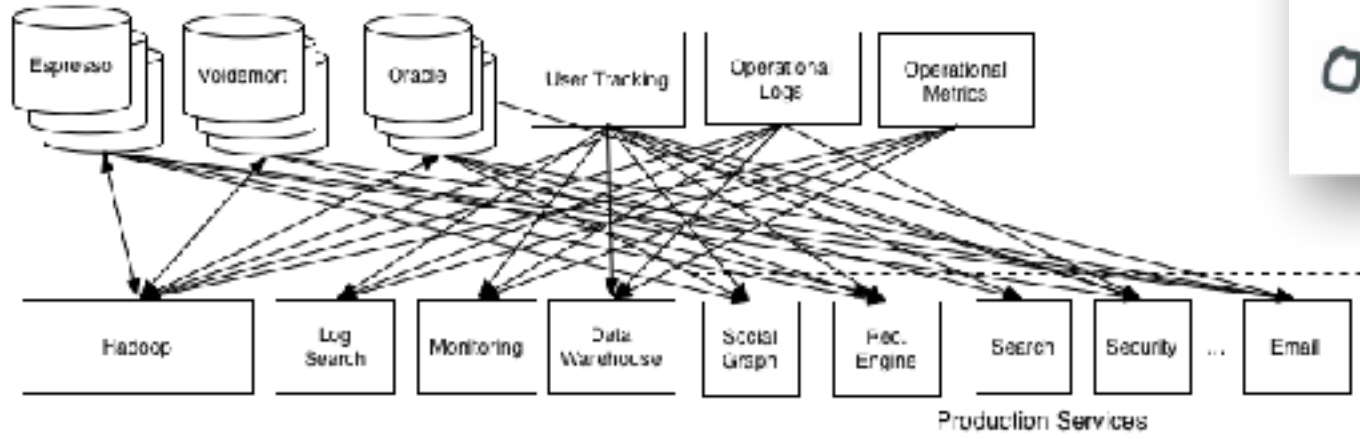


# What is an event-sourced architecture?

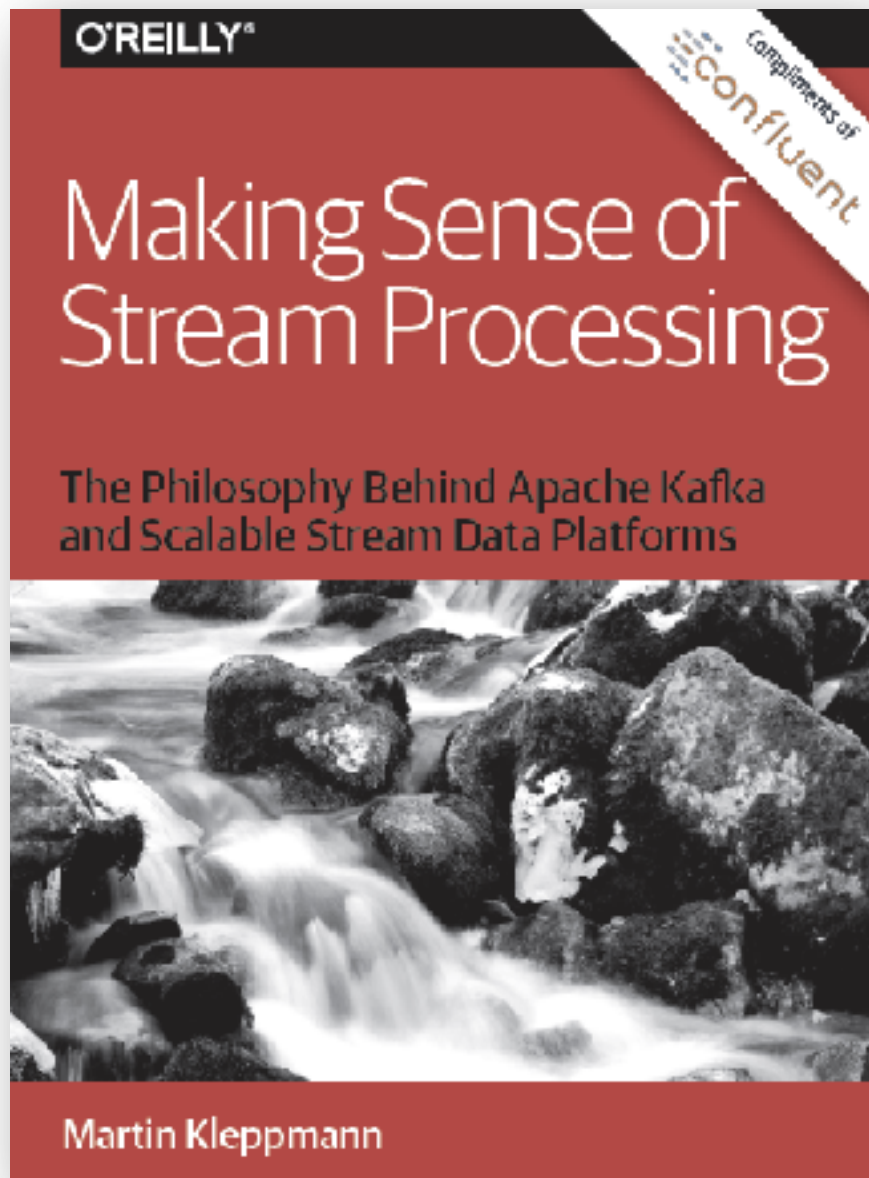
Basically, this:



Instead of this:







Linked in Engineering Home Blog Data Open Source Jobs Woman In Tech

## The Log: What every software engineer should know about real-time data's unifying abstraction

 Jay Kreps December 16, 2013

[in Share](#) 1,887 [Tweet](#) [Like](#) 421 [G+1](#) 424

## The world beyond batch: Streaming 101

A high-level tour of modern data-processing concepts.

By Tyler Akidau, August 5, 2015

## Versioning in an Event Sourced System

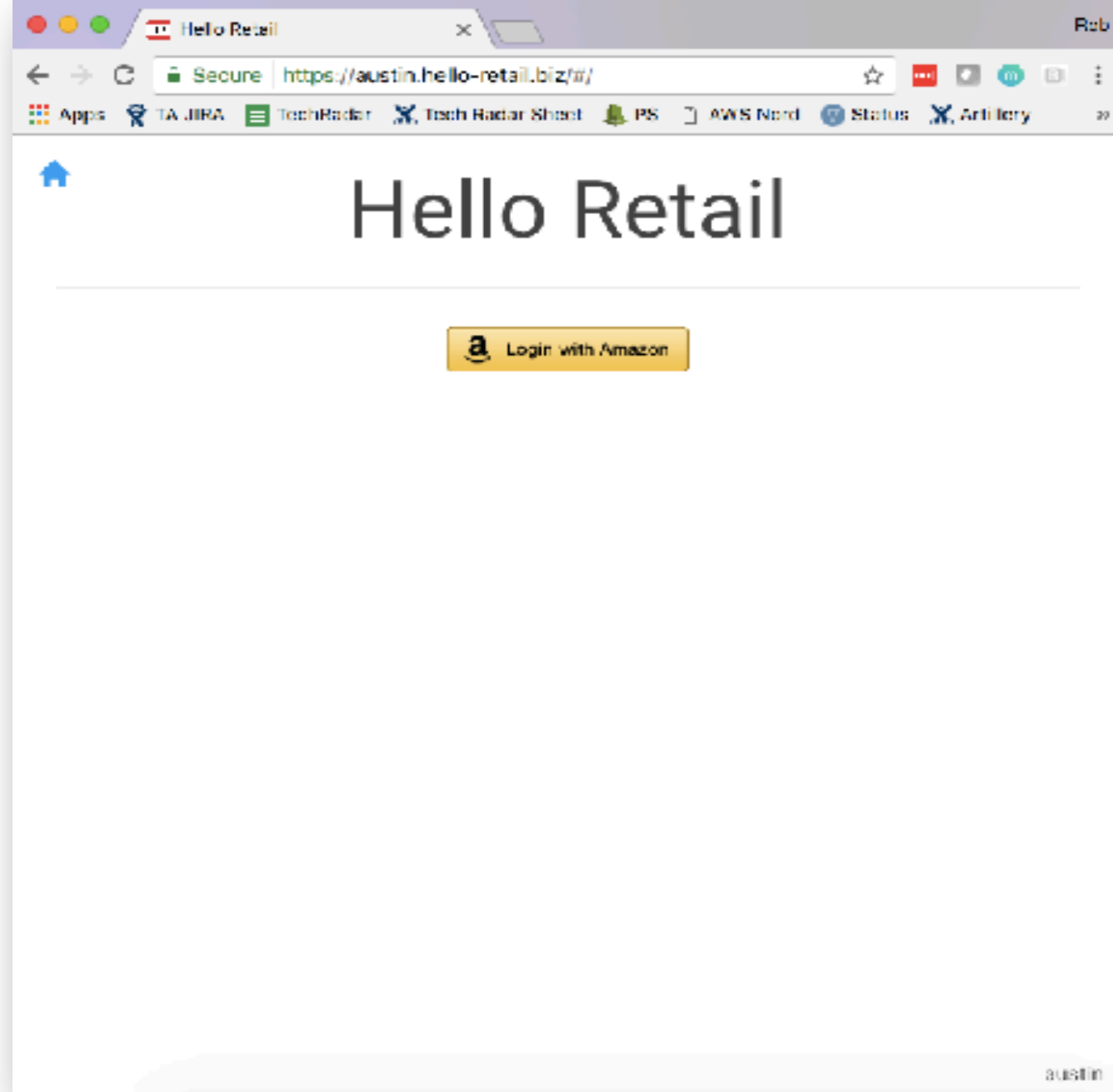
Gregory Young

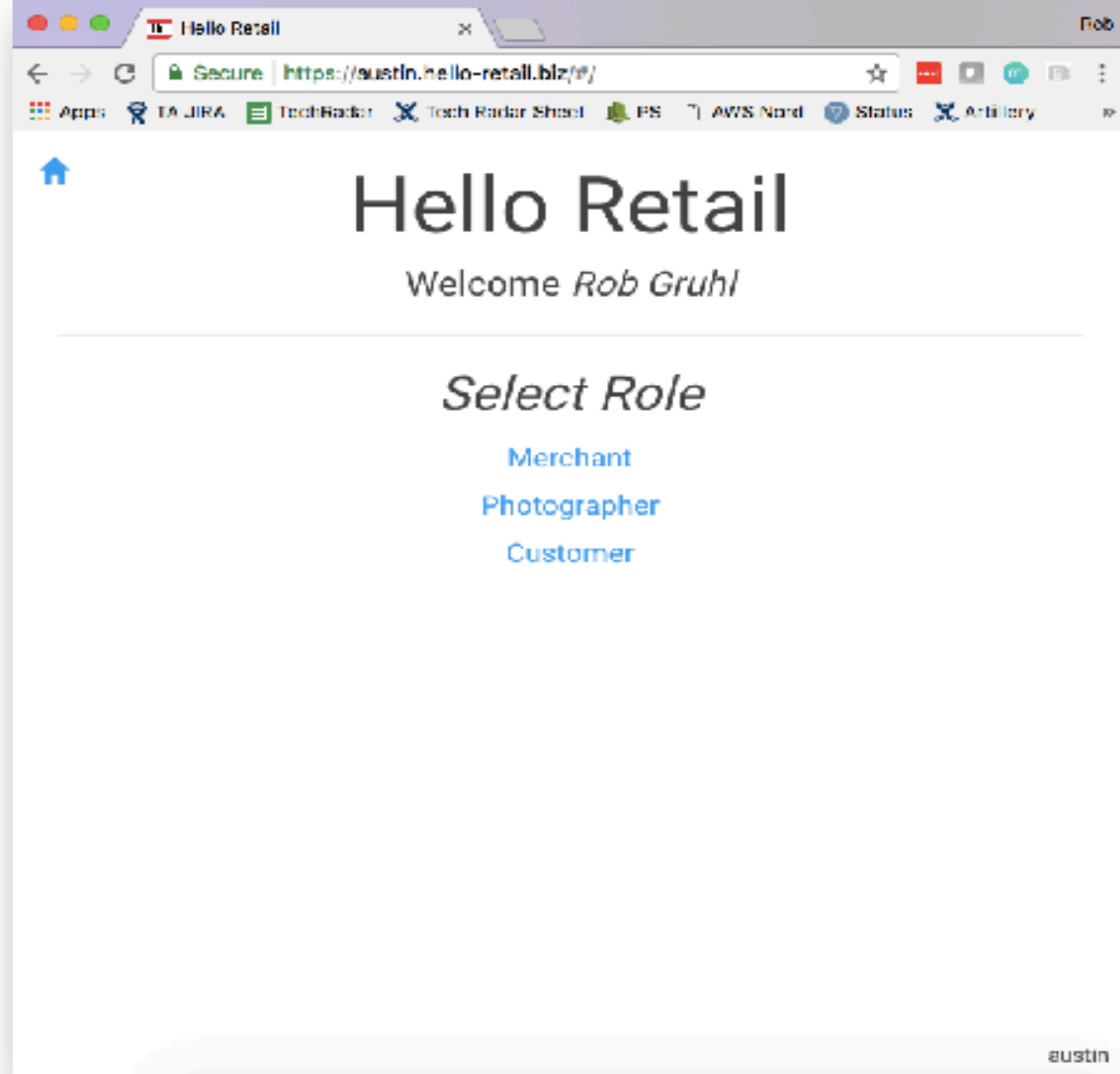
Jay Kreps, Tyler Akidau  
Martin Kleppmann, Greg Young

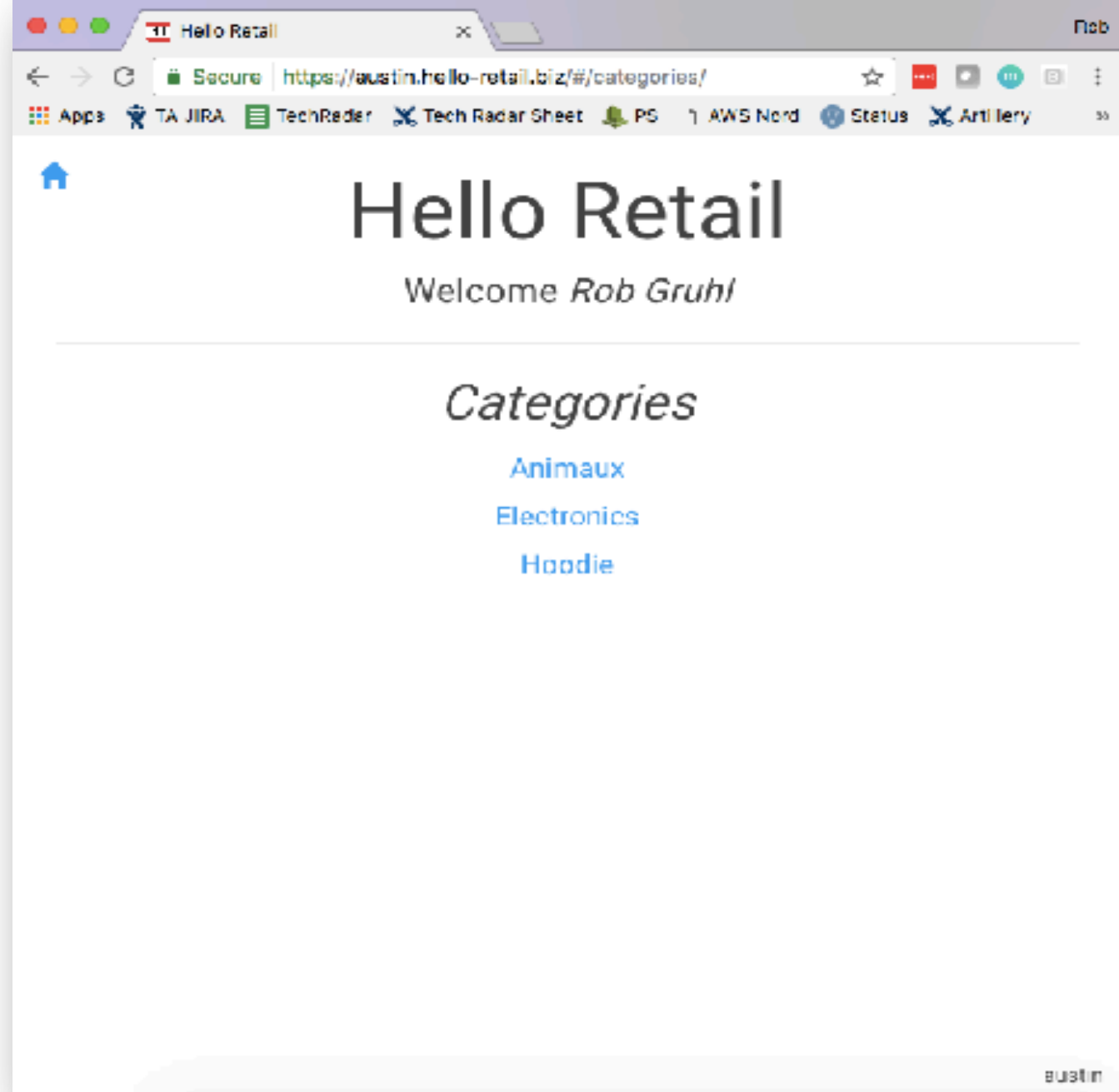


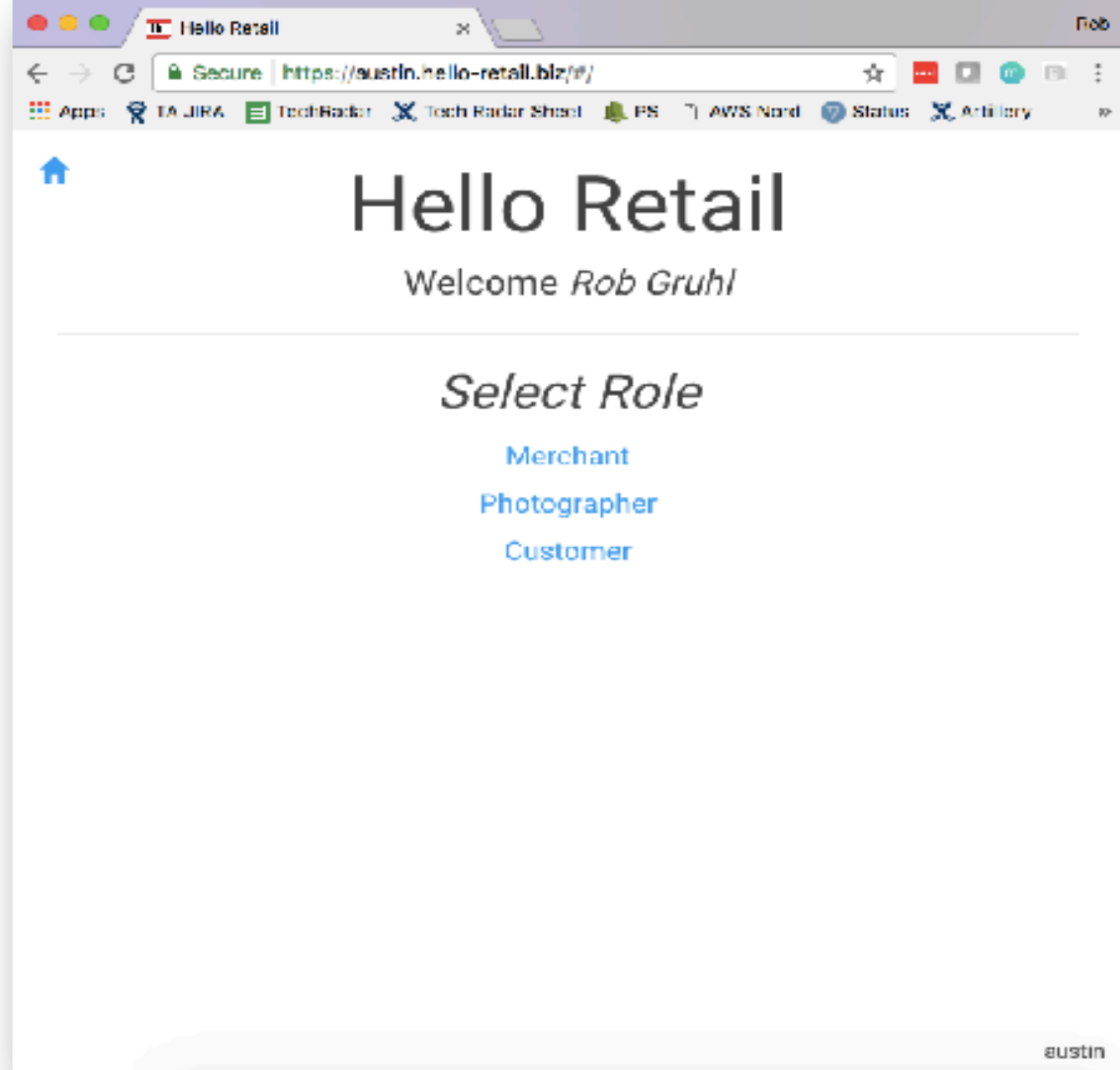
# **June 2017:** **[github.com/Nordstrom/hello-retail](https://github.com/Nordstrom/hello-retail)**

“Hello, Retail!” is an open-source, 100% serverless, functional proof-of-concept showcasing an event-sourced approach as applied to the retail platform space.










Hello Retail

Secure <https://austin.hello-retail.biz/#/photographer/>

Apps TA JIRA TechRadar Tech Radar Sheet PS AWS Nord Status Artillery

 **Hello Retail**

Welcome *Rob Gruhl*

---

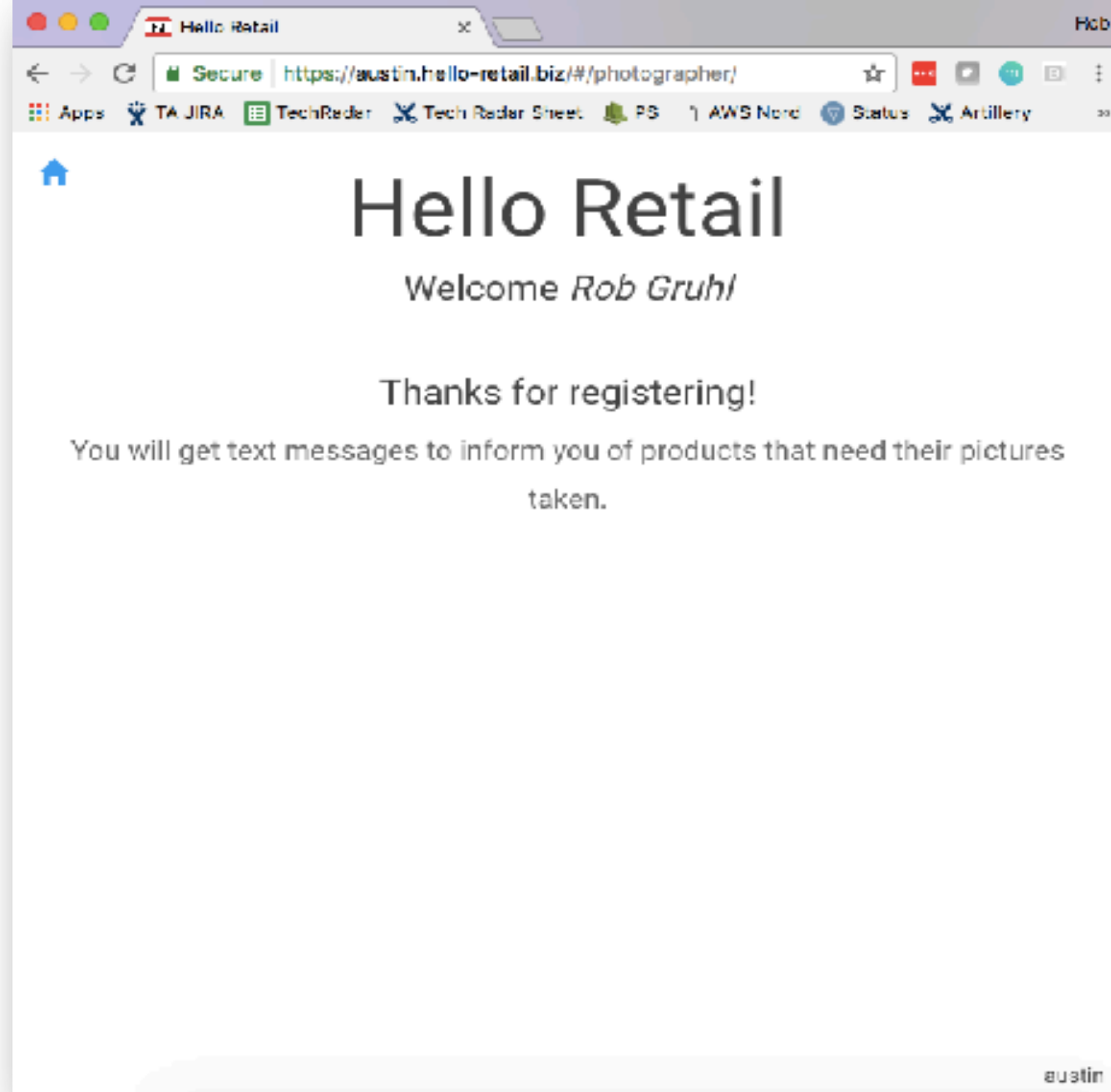
*Photographer Registration*

Phone Number:

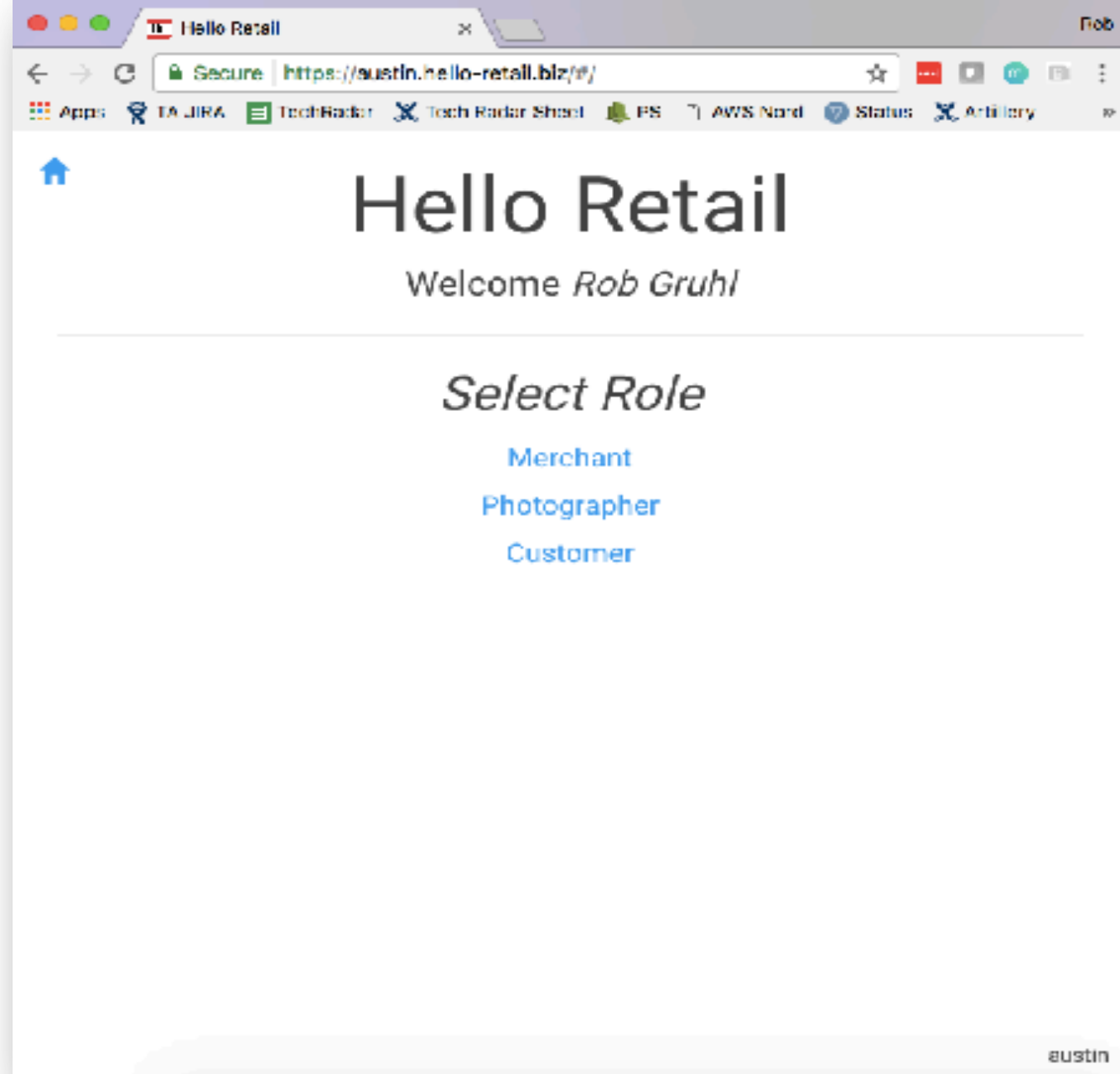
(Additional charges may apply.)

**Register**

austin








Hello Retail

Secure <https://austin.hello-retail.biz/#/merchant/>

Apps TA JIRA TechRadar Tech Radar Sheet PS AWS Nord Status Artillery

 **Hello Retail**  
Welcome *Rob Gruhl*

---

*Create New Product*

Category:

Animals

Name:

Cats

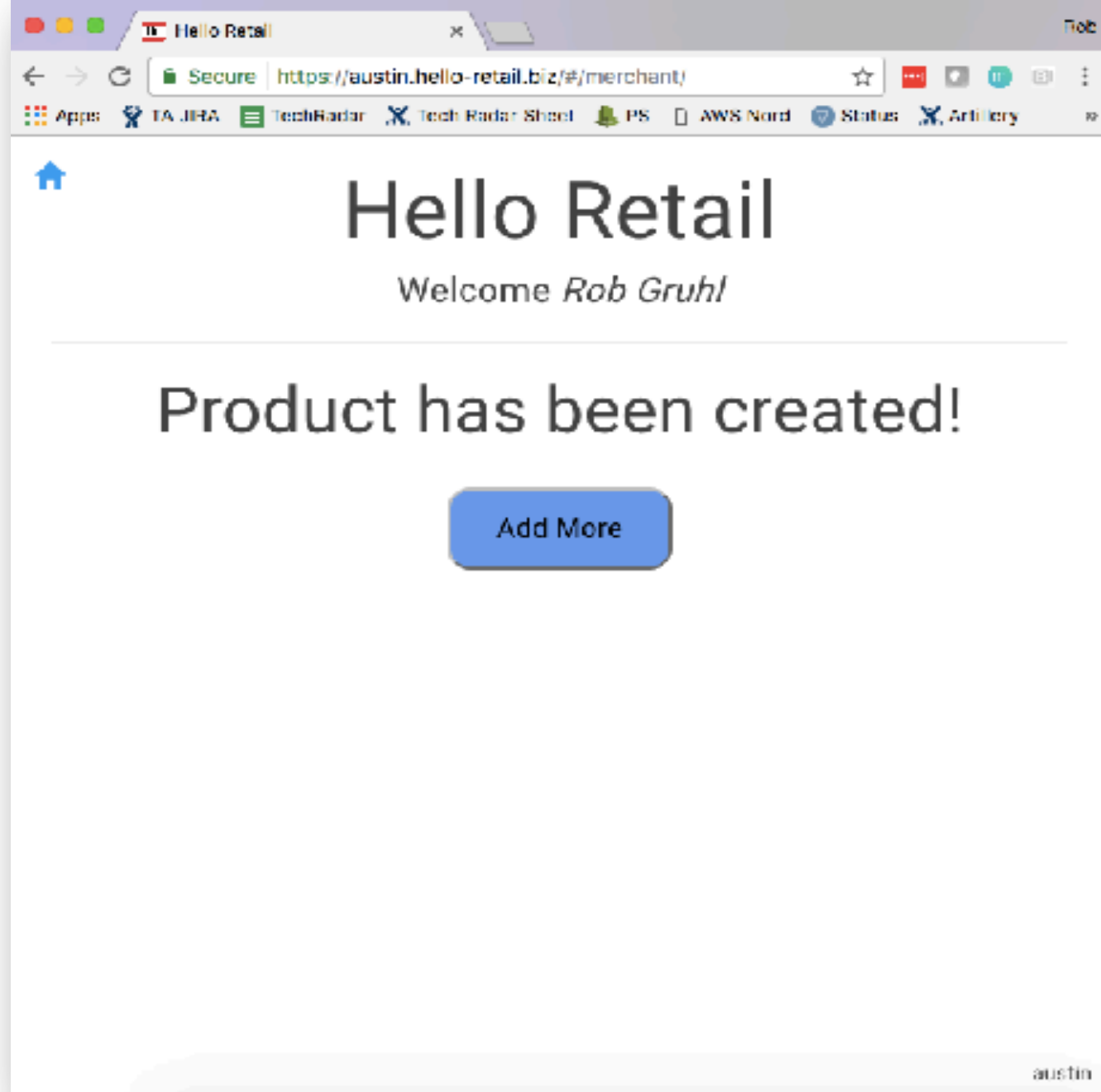
Brand:

AirBnB

Description:

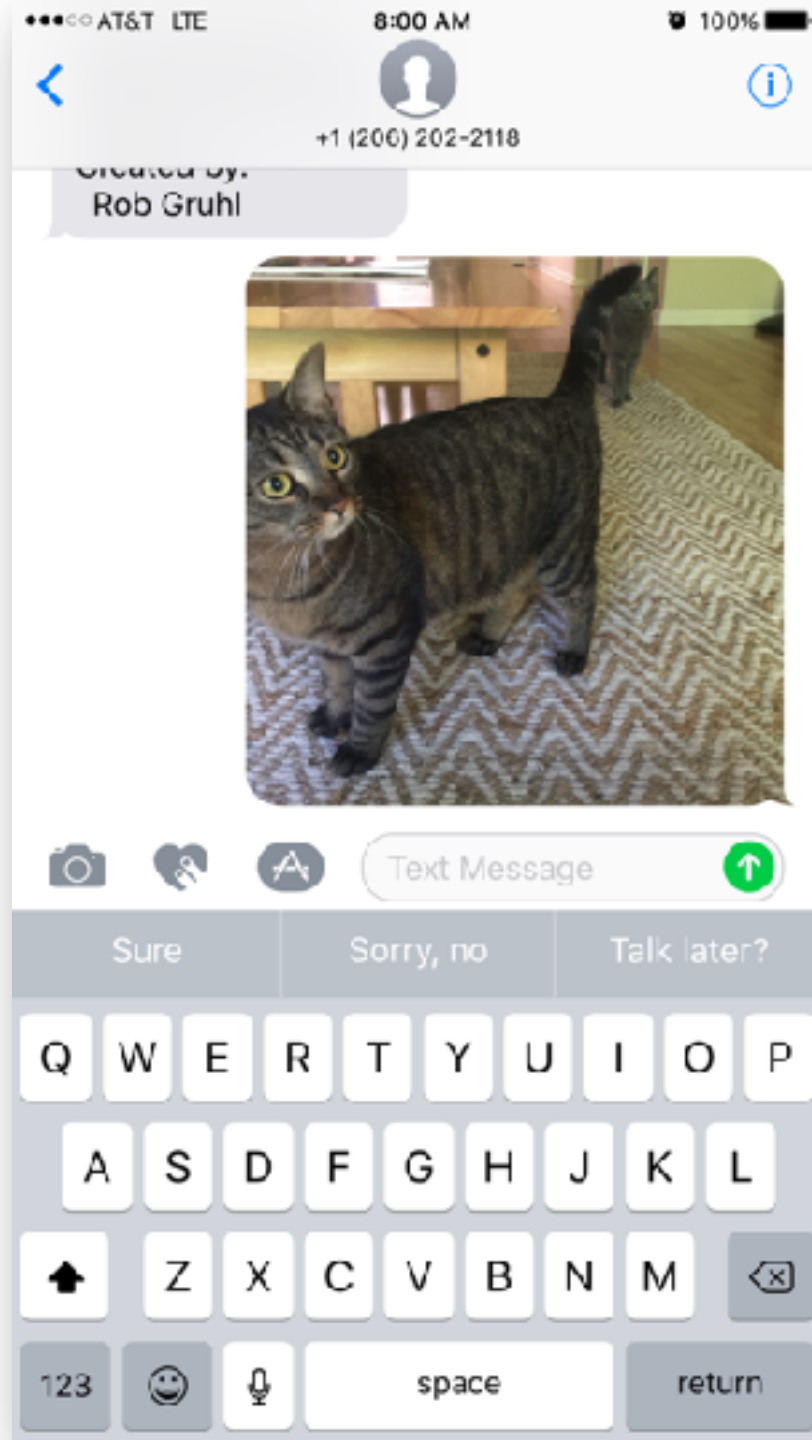
This AirBnB comes complete with two delightful cats!

austin



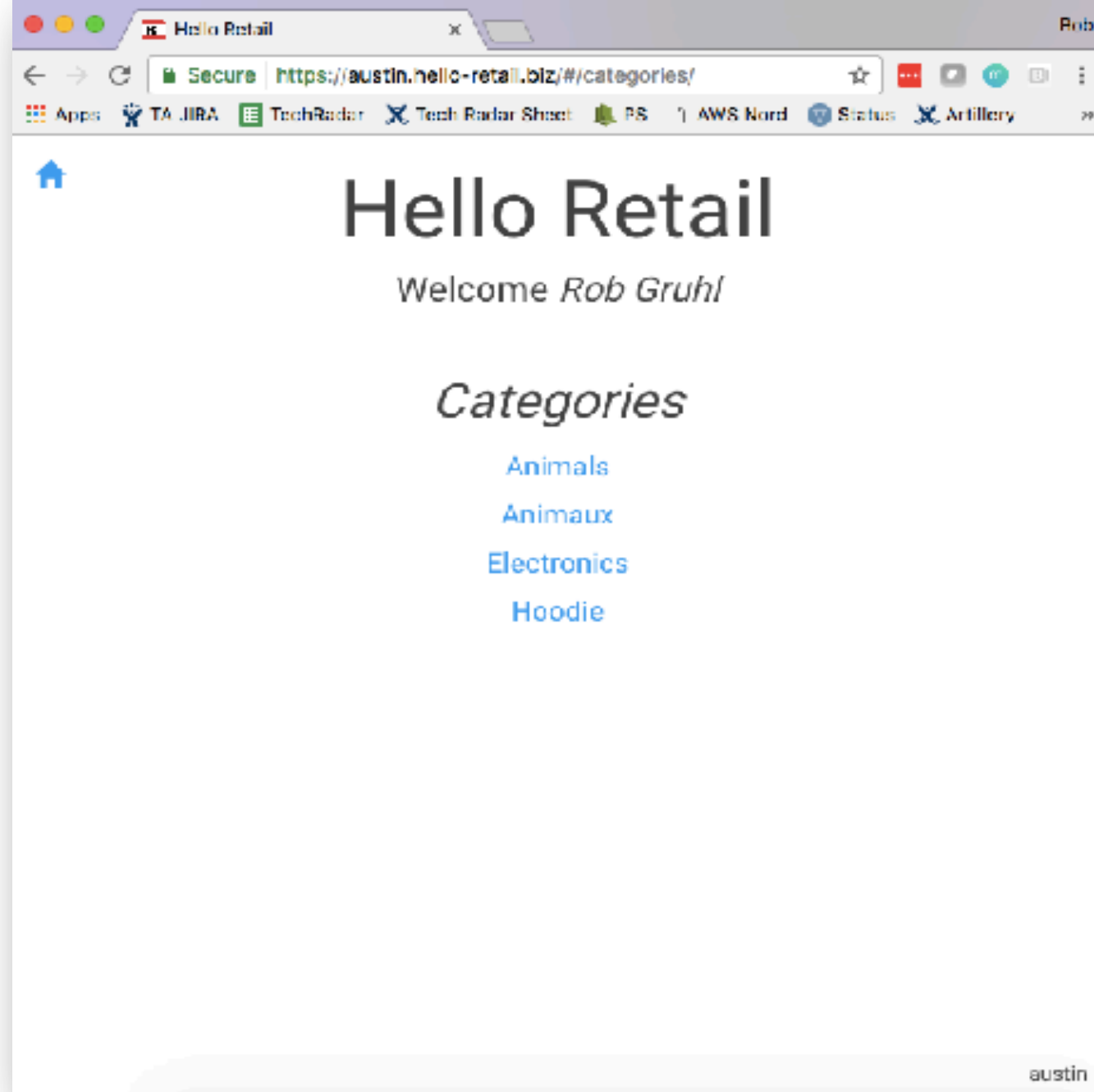


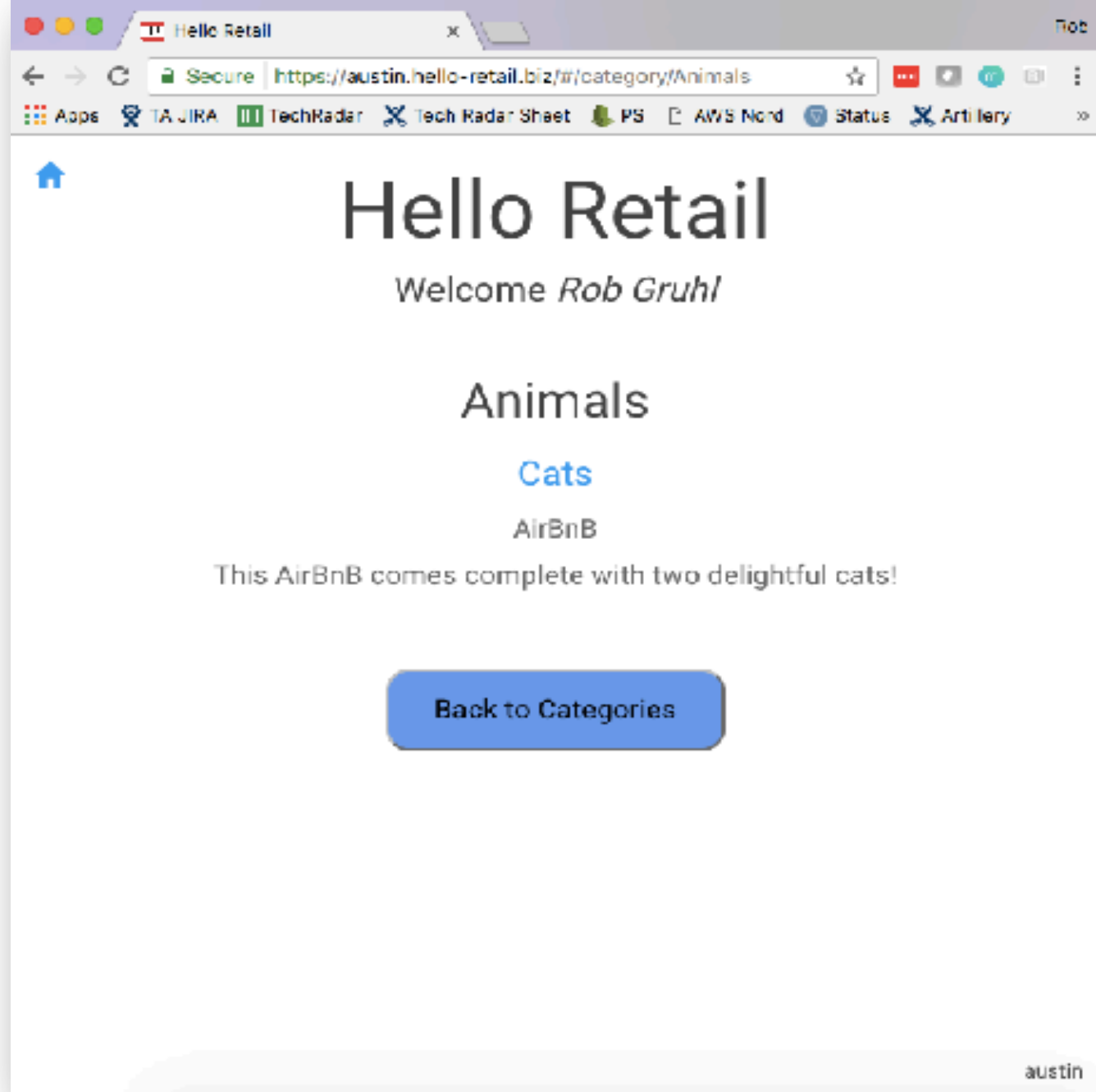


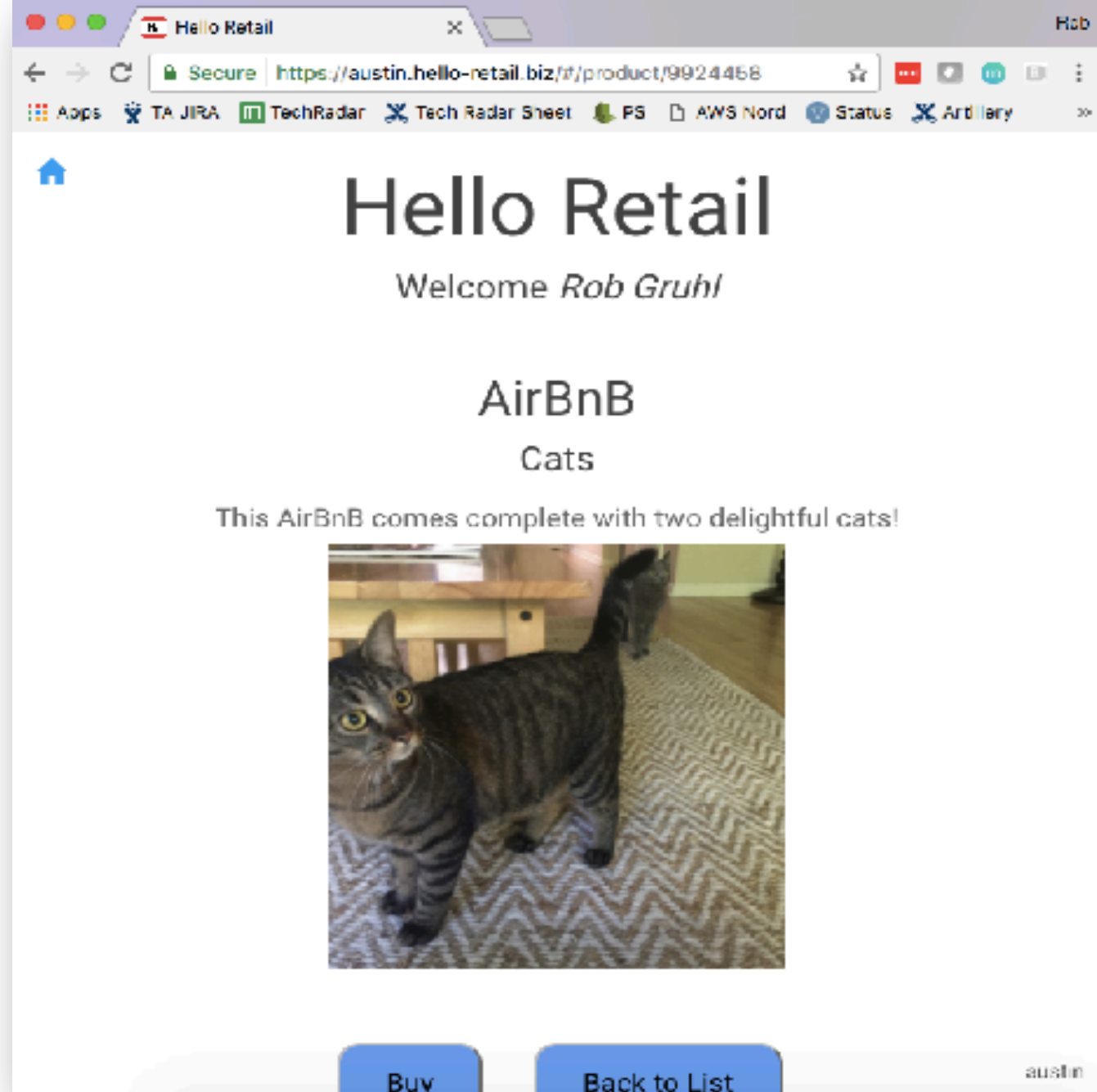


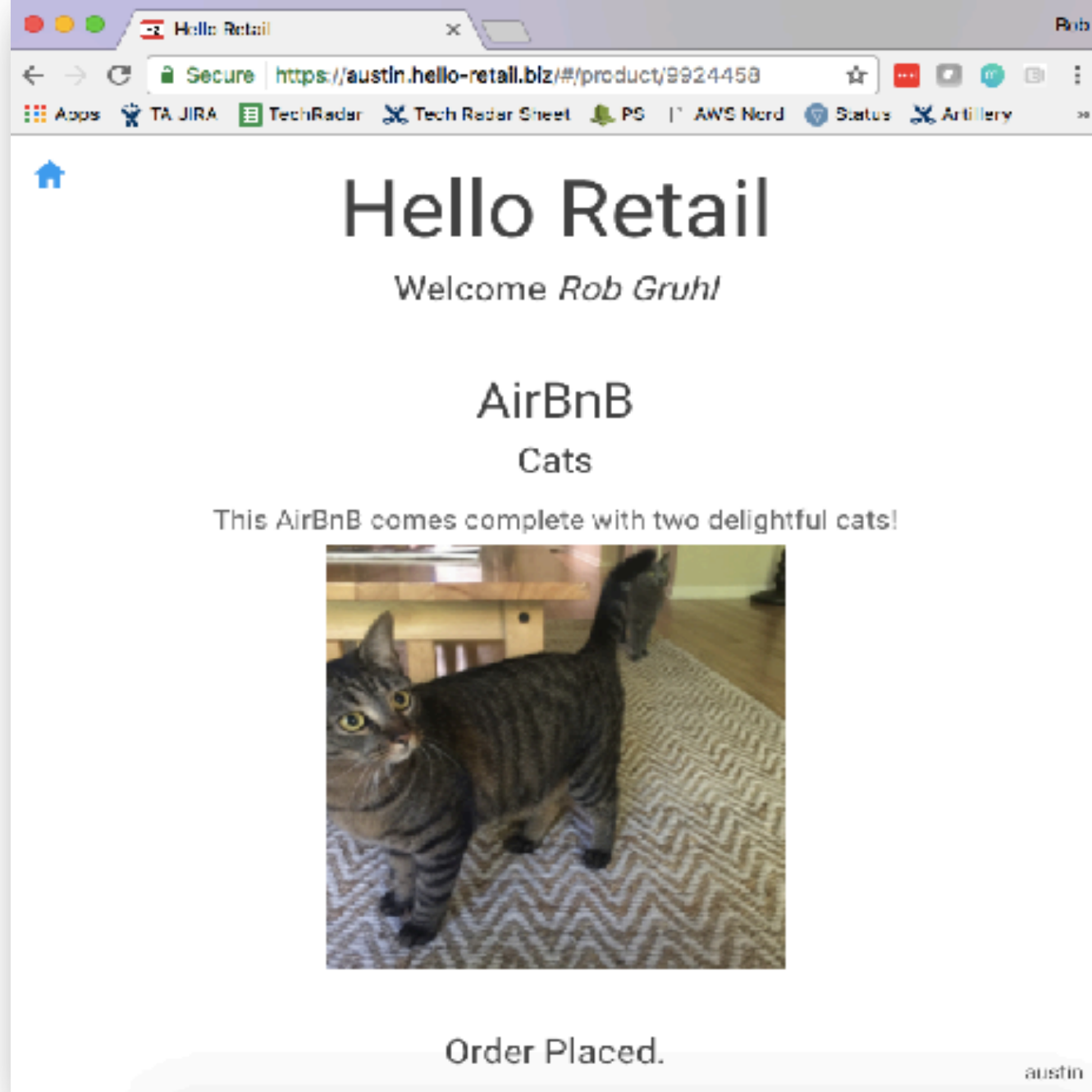




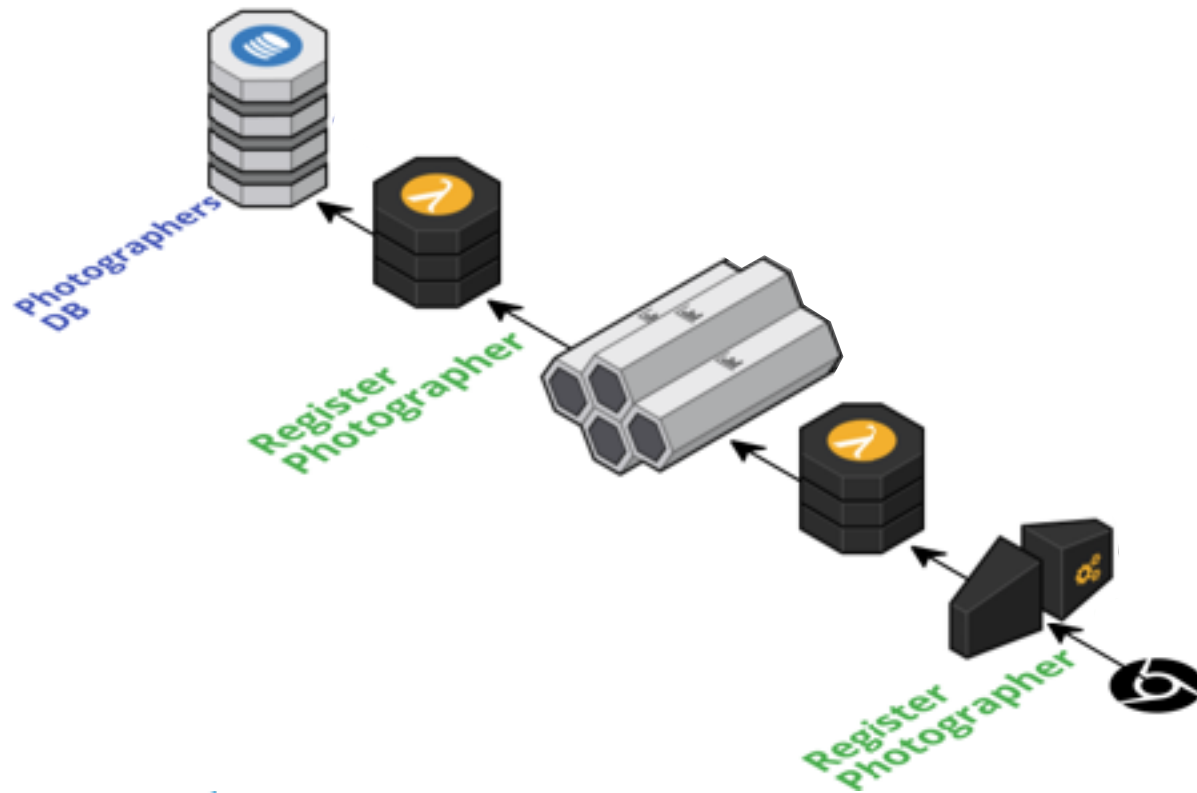




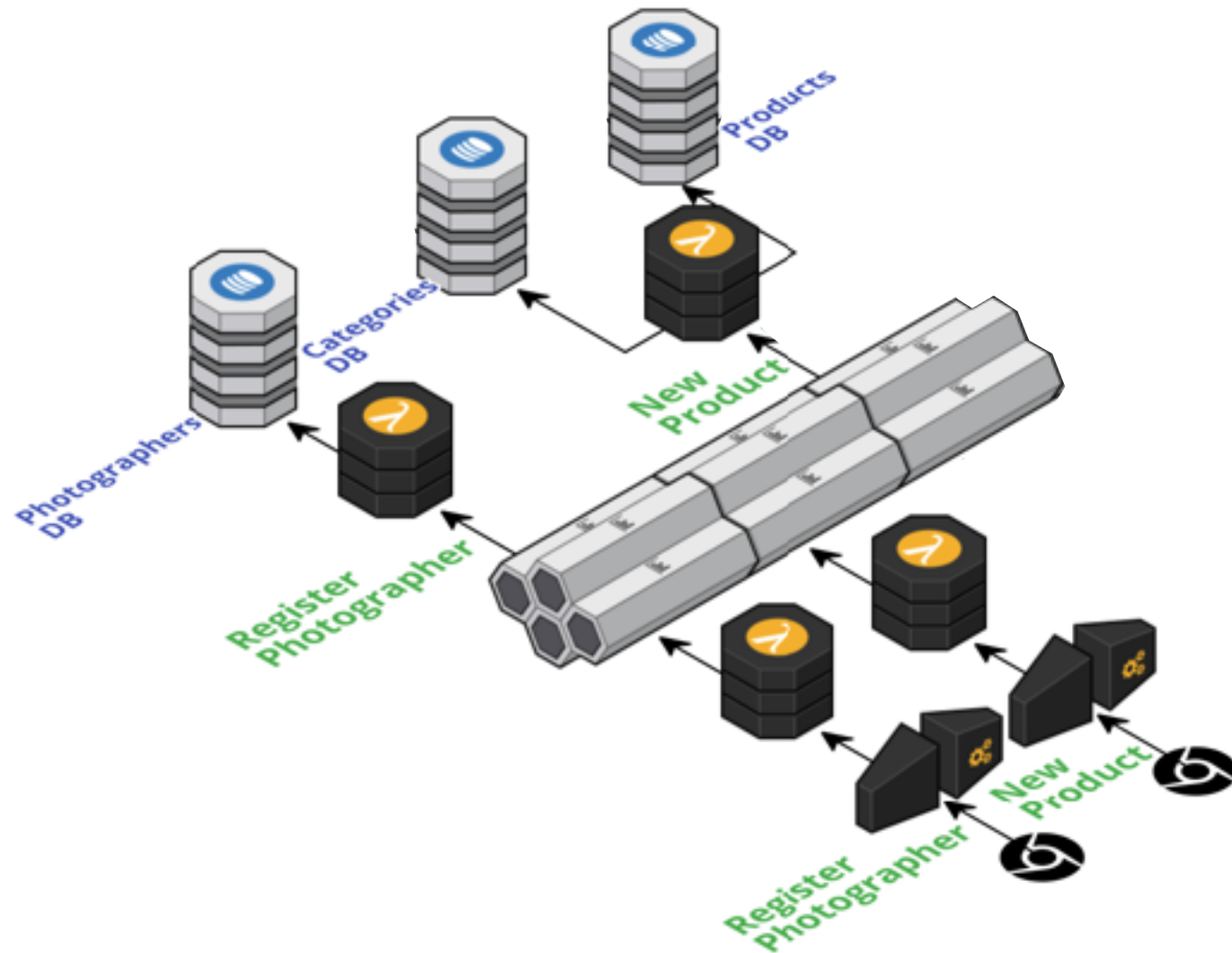




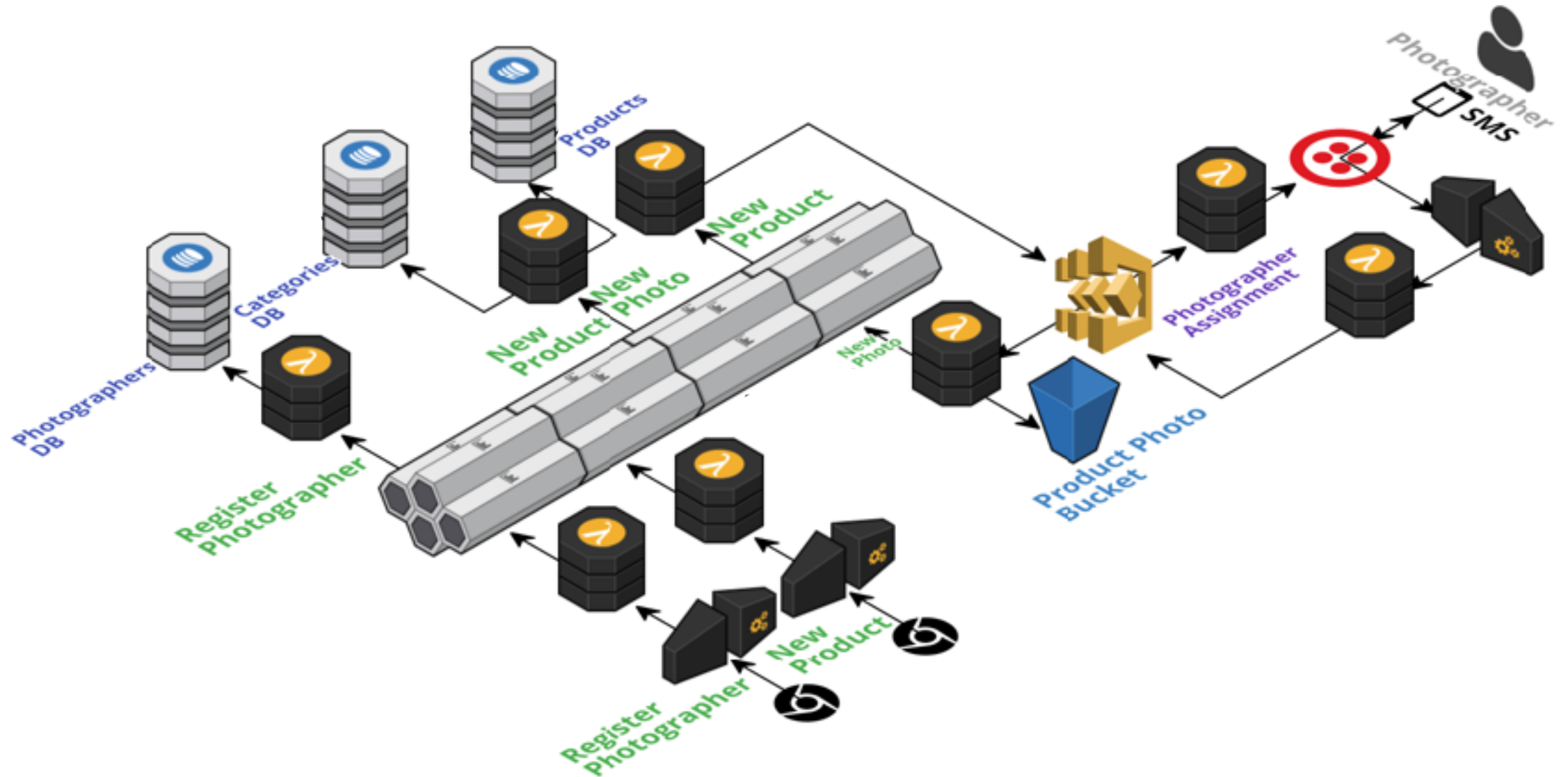
# Hello, Retail!



# Hello, Retail!

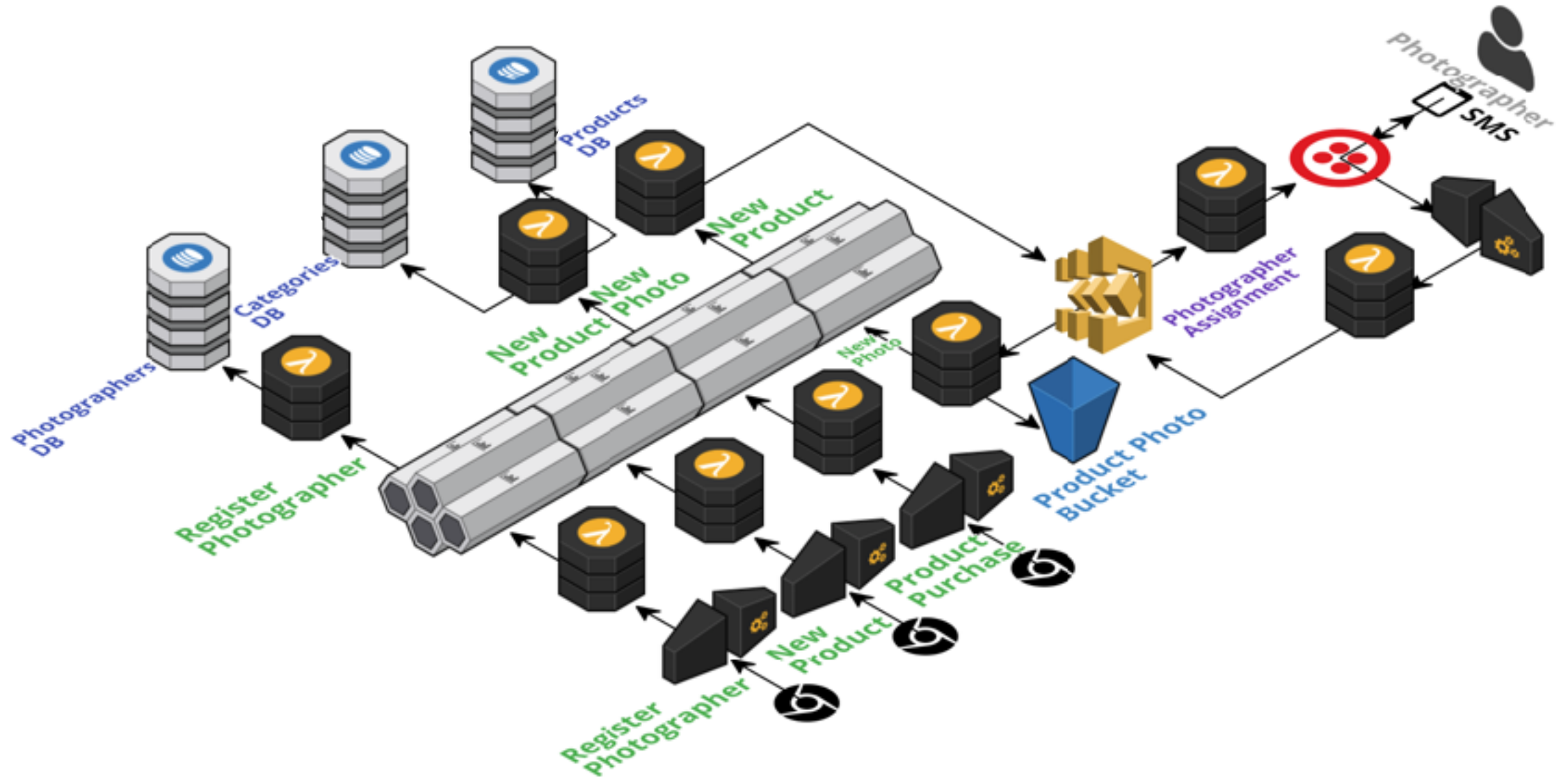


# Hello, Retail!

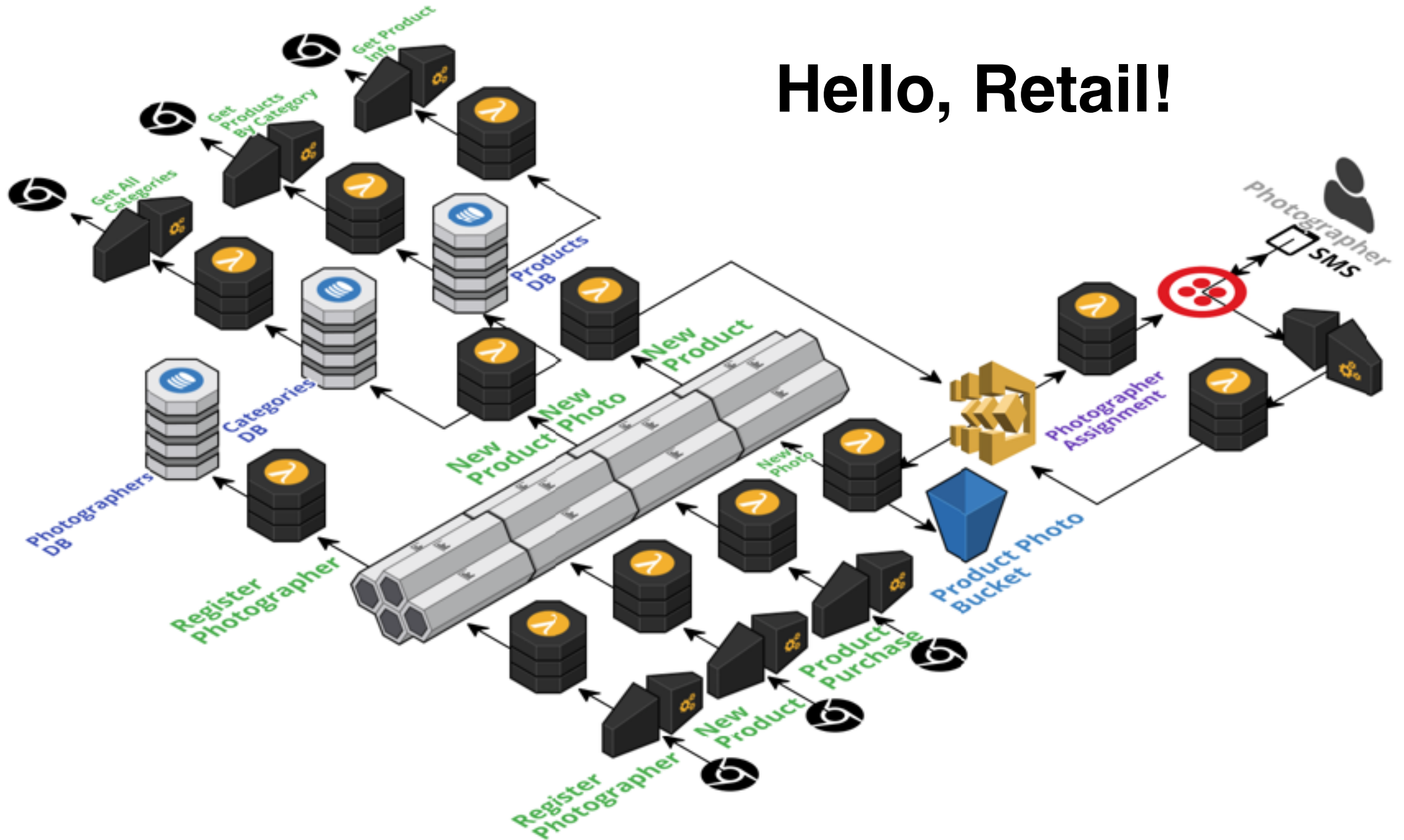




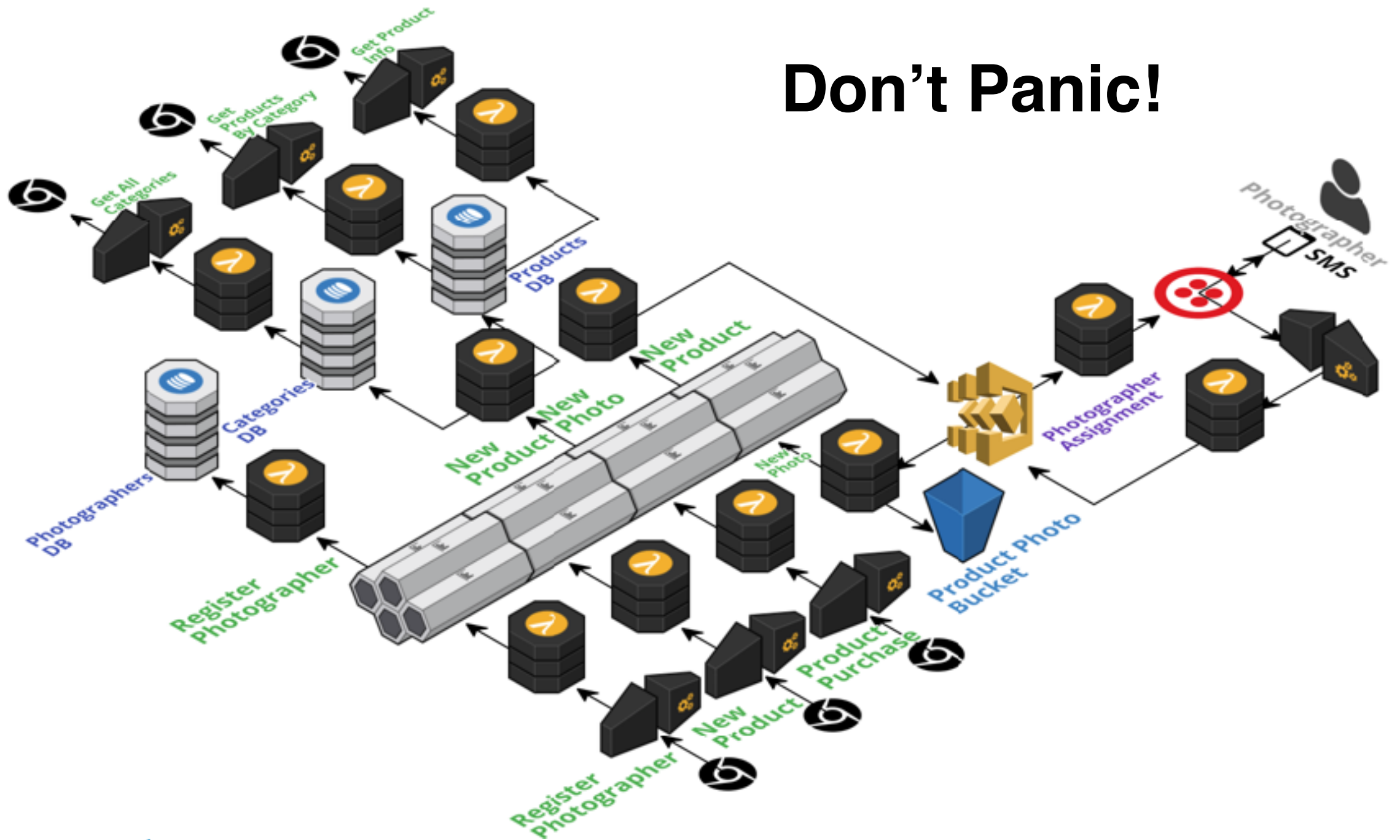
# Hello, Retail!

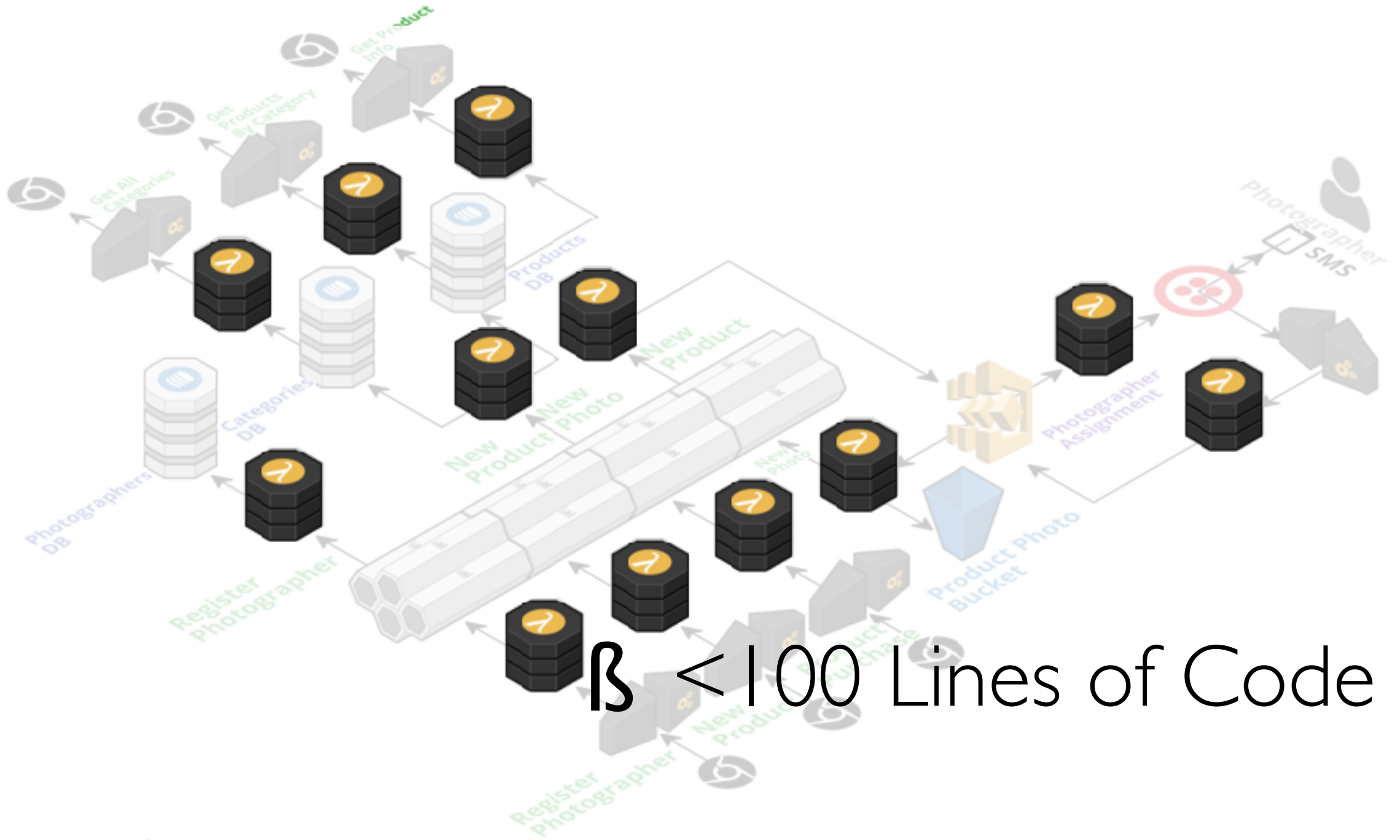


# Hello, Retail!



# Don't Panic!





# What we *think* we've learned about event-sourcing



- [serverless@nordstrom.com](mailto:serverless@nordstrom.com) with feedback



# Produce simple, high-quality events

- Written in the past tense
- An observation of something that happened in the world
- The record of a decision – a “royal decree”



# Swim upstream to find your source of truth



- Find the person making the original decision
- Find the sensor or system observing the fact



# Everyone's first job: fix the stream

- “I can’t use the stream because...”
- Add attributes, clean up data, fix things
- Improve alerting/monitoring/availability
- Everyone benefits



# Tech Debt Streams vs Published Streams

- Tech debt streams are filled with complexity
- Published streams contain high quality events
- Don't confuse the two, both are useful



# Maintain ordering



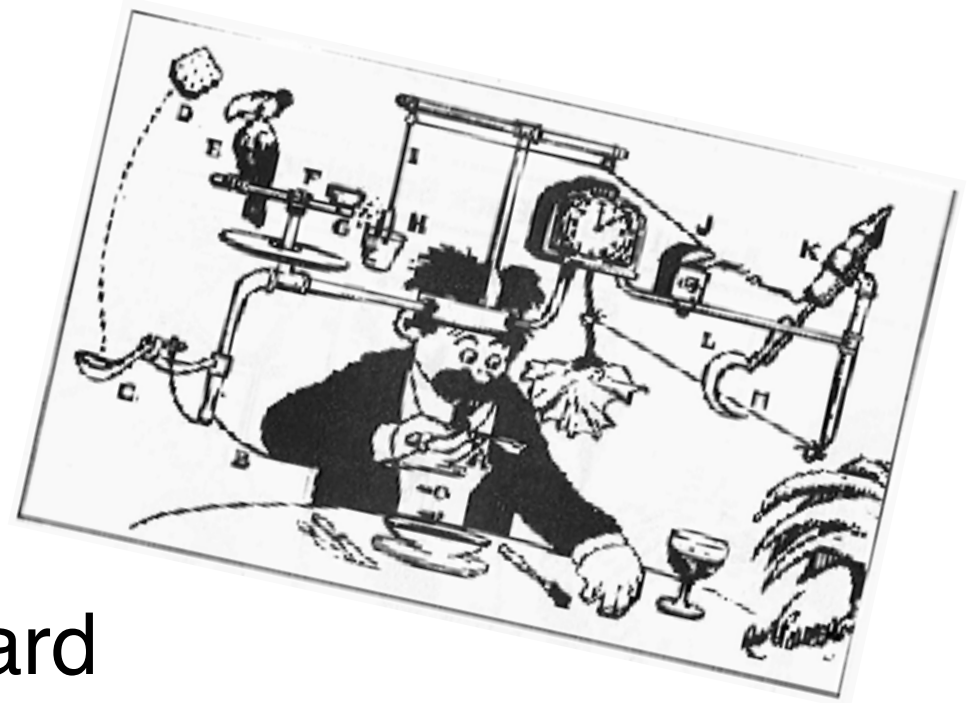
- Many systems, libraries, tools, and developers want to process events asynchronously.
- Synchronous, single-threaded execution is required to guarantee ordering.

# Partition key selection is (really) important



- Partition keys determine your ordering guarantee
- Will any system behave differently if the data is re-ordered?
- You can only guarantee ordering for each specific partition key

# Do (much) more with your event consumers



- Don't just transform and forward
- What is that downstream system doing?
- What is the system after that doing?
- Can you just do that?



# Distributed systems are hard, eventual consistency is weird

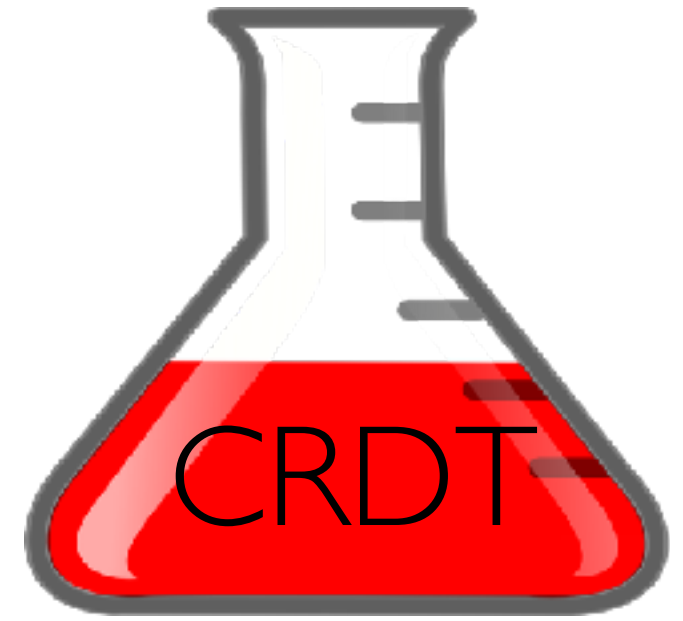
- Acceptance test: Add to Cart, View Cart



# What we're figuring out.

- [serverless@nordstrom.com](mailto:serverless@nordstrom.com) with feedback

# Joins and aggregates across partition keys



- Ex: riders in taxis.
- Join streams into a canonical joined stream
- Use a convergent data type (CRDT) Woo!

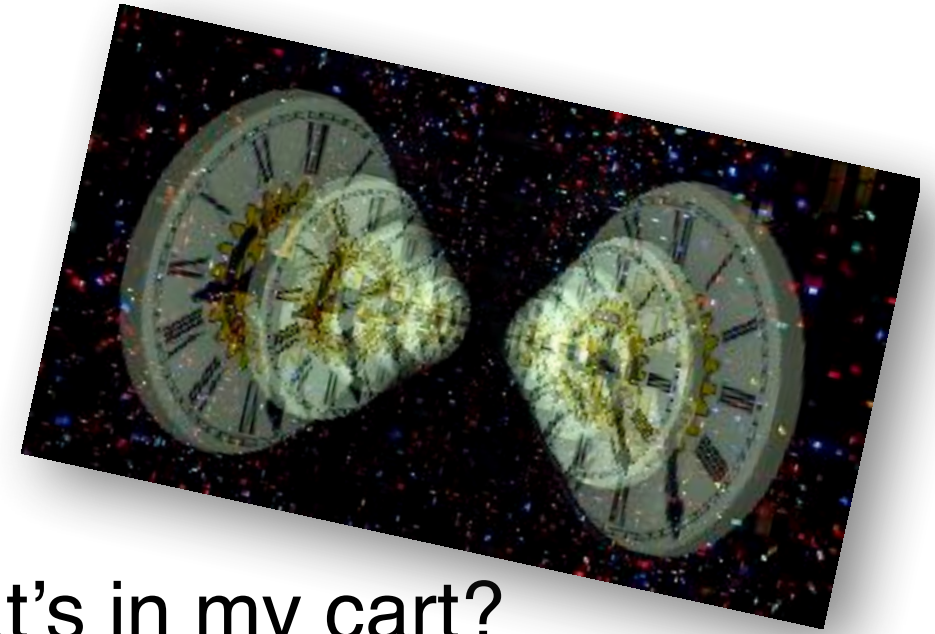


## An (a)synchronous approach

*A1, B, A2, C, D, E*

- Strictly speaking, you only need to maintain order on events with identical partition keys, so optimize by handling different keys asynchronously.

# Effect-after-cause



- Given my add to cart event, what's in my cart?
- UI holds open a connection to the cart contents service?
- Write-through the cart DB?
- Use ledger write receipts as causal preconditions to read?

# My serverless event-sourcing wish list



# Better ways to manage serverless fan-out



- Hooray! Your published stream is super-popular!
- How can 50+ consumers read from your stream?
- Single biggest issue today with AWS Kinesis

# Replay from event archive at 10,000x speed



- Cold start a new feature using historical data.
- Three years of events in one hour.
- Identical code to my real-time stream processor function.
- Once caught-up, attach to the live stream.

# Get started!

- Serverless?
  - Try [github.com/Nordstrom/serverless-artillery](https://github.com/Nordstrom/serverless-artillery)
- Event-sourcing?
  - Choose a visible but non-critical system.

**Thanks!**  
**serverless@nordstrom.com**

- <http://github.com/nordstrom/hello-retail>
- <http://github.com/nordstrom/hello-retail-workshop>
- <http://github.com/nordstrom/serverless-artillery>
- <http://github.com/nordstrom/serverless-artillery-workshop>