
AWS
re:Invent

DAT334

Advanced design patterns for Amazon DynamoDB

Sean Shriver
Sr. DynamoDB SA
Amazon Web Services

Daniel Yoder
Sr. NoSQL Solution Architect
Amazon Web Services

Rick Houlihan
Principal Technologist, NoSQL
Amazon Web Services

AWS re:Invent

aws

Agenda

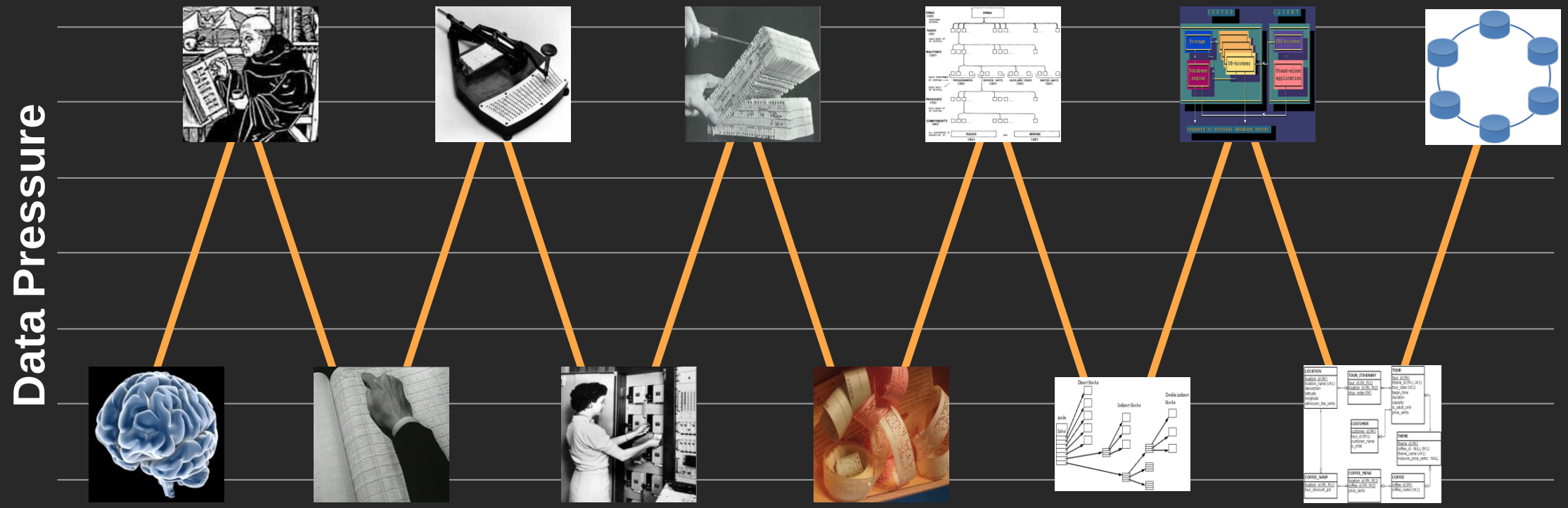
- NoSQL history
- Amazon DynamoDB end-to-end
- Scaling NoSQL
- NoSQL modeling
- DynamoDB features
- Lab preparation
- `<lab>Advanced design patterns for Amazon DynamoDB</lab>`

History of data processing

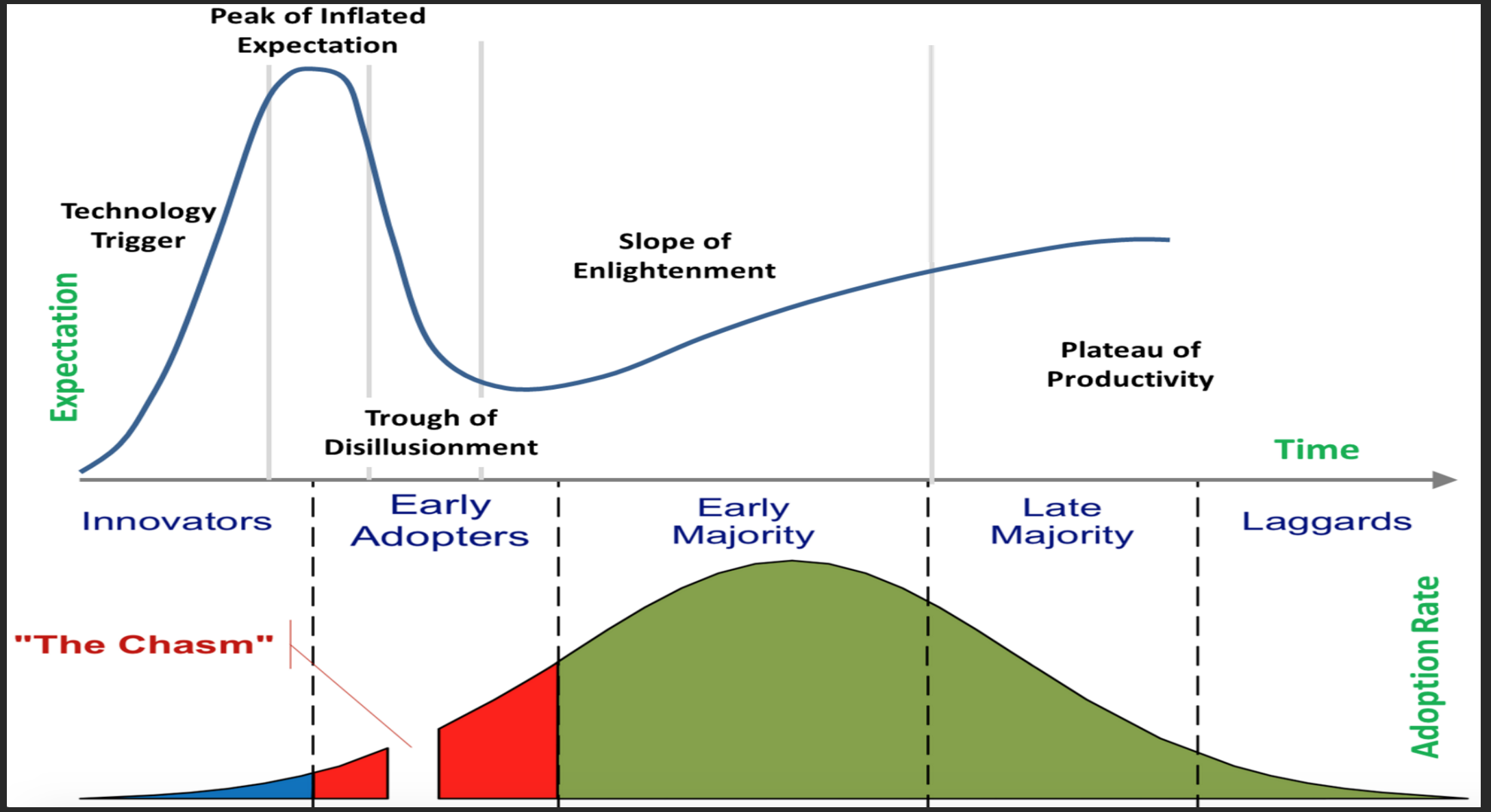
"History repeats itself because nobody was listening the first time."

- Anonymous

Timeline of database technology



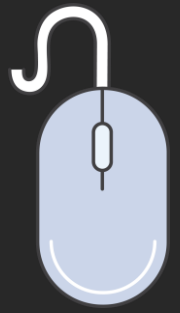
Technology adoption and the hype curve



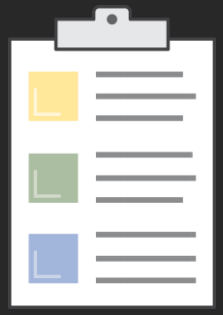
Why NoSQL?

SQL	NoSQL
Optimized for storage	**Optimized for compute**
Normalized/relational	Denormalized/hierarchical
Ad hoc queries	Instantiated views
Scale vertically	Scale horizontally
Good for OLAP	Built for OLTP at scale

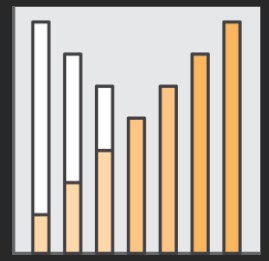
Amazon DynamoDB



Fully managed NoSQL



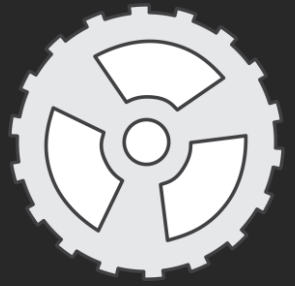
Document or key-value



Scales to any workload

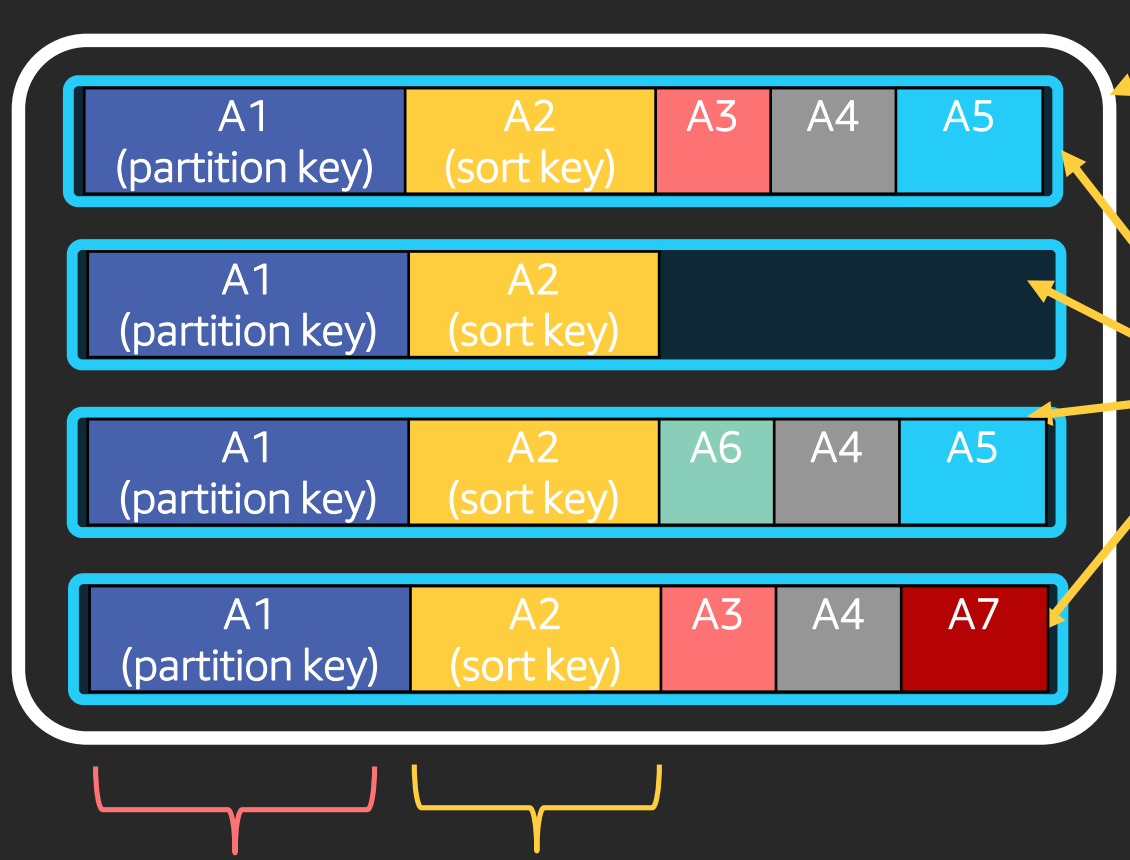
Fast and consistent

Access control



Event-driven programming

DynamoDB Table



All items for a partition key
==, <, >, >=, <=
“begins with”
“between”
sorted results
counts
top/bottom N values
paged responses

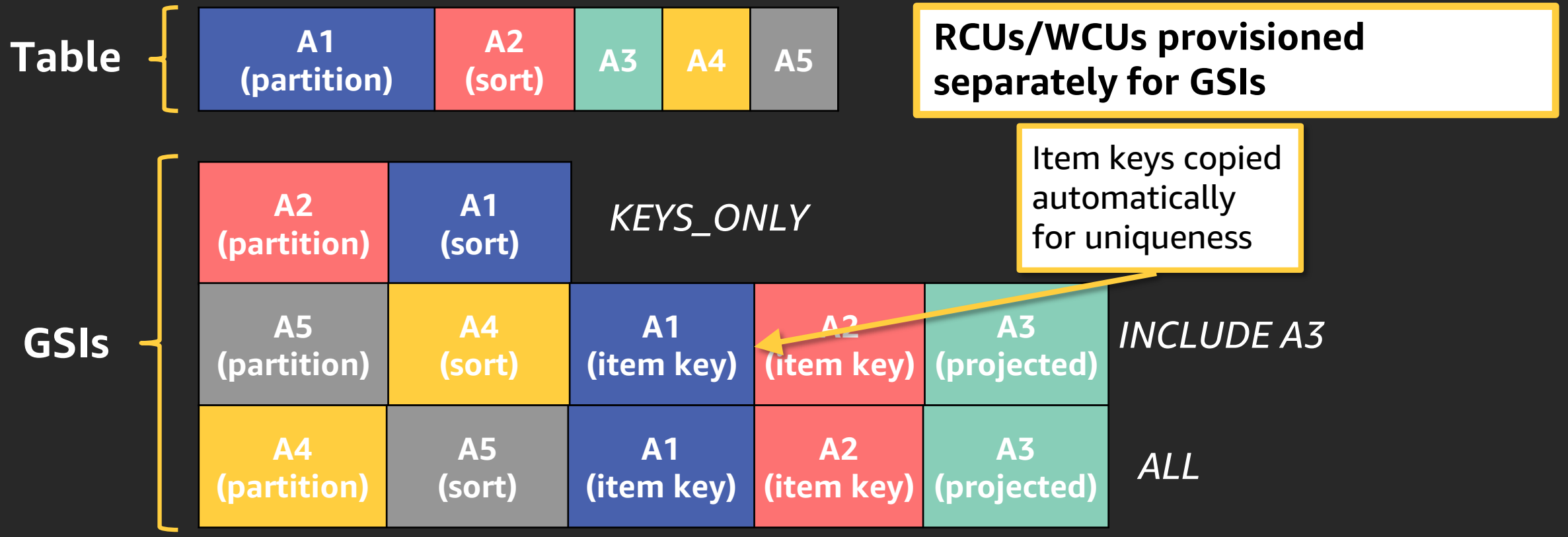
Mandatory
Key-value access pattern
Determines data distribution

Optional
Model 1:N relationships
Enables rich query capabilities

Global secondary index (GSI)

Up to 20 GSIs per table

Alternate partition and/or sort key
Index is across all partition keys
Use composite sort keys for compound indexes



Partition overloading

Terminology
PK is short for **partition key** ***SK*** is short for **sort key**

Use generic keys to facilitate heterogeneous partitions

Primary Key		Attributes	
PK	SK		
role:1	root	roleName	
		Default role	
	policy:1	policyName	GSI_1_PK
		Admin	role:policy:1
	policy:2	policyName	GSI_1_PK
		RejectWrites	role:policy:2
	policy:3	policyName	GSI_1_PK
		RevokeAssumeRole	role:policy:3

```
SELECT * WHERE PK=role:1 AND SK starts_with policy:2
```

Adjacency lists

Generic partition and sort key names

Often used with GSI overloading

Primary Key		Attributes		
PK	SK			
processorder:1	part:1	porosity	mass	
			122g	
	processstep:materials:1	materialIds		
		[materialId:1, materialId:2]		
	processstep:mixing:2	mass	volume	density
		130g	26cm^3	5g/cm^3
	processstep:print:3	bedTemp	materiallTemp	adhesionSurface
		120	400	tape

GSI_1_PK
part:1

Soft references for n:m relationship

Primary Key		Attributes			
PK	SK				
materialid:1	processorder:1	mass	volume	density	timestamp
		45g	9cm^3	5g/cm^3	1575423000
	processorder:2	mass	volume	density	timestamp
		66g	13.2cm^3	5g/cm^3	1575477000
materialid:2	processorder:1	mass	volume	density	timestamp
		85g	17cm^3	5g/cm^3	1575423000
	processorder:2	mass	volume	density	timestamp
		64g	12.8cm^3	5g/cm^3	1575477000

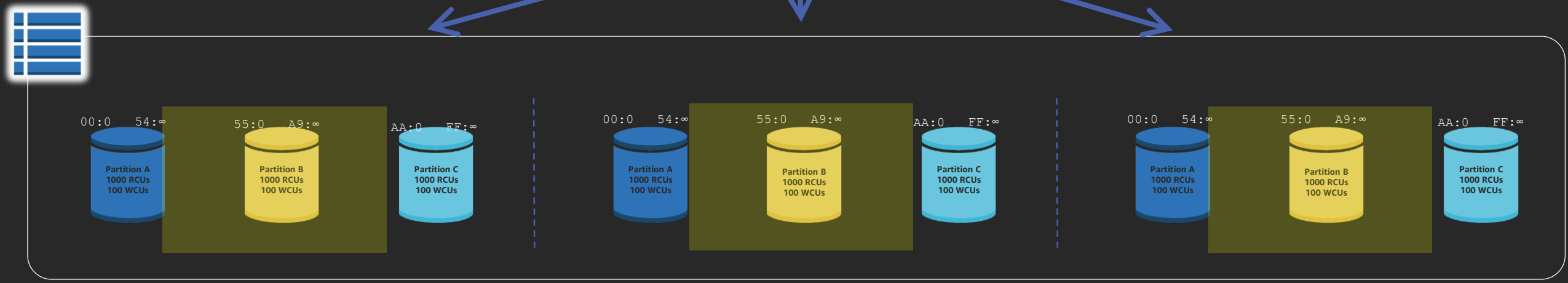
Partition keys in NoSQL

A partition/shard key is used for building an unordered hash index
Allows table to be partitioned for scale

3-way replication

Data is replicated to three Availability Zones by design

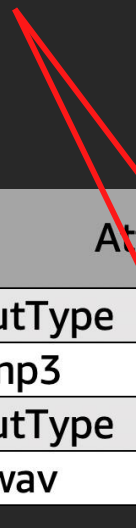
OrderId: 1
CustomerId: 1
ASIN: [B00X4WHP5E]



CustomerOrdersTable

Write sharding

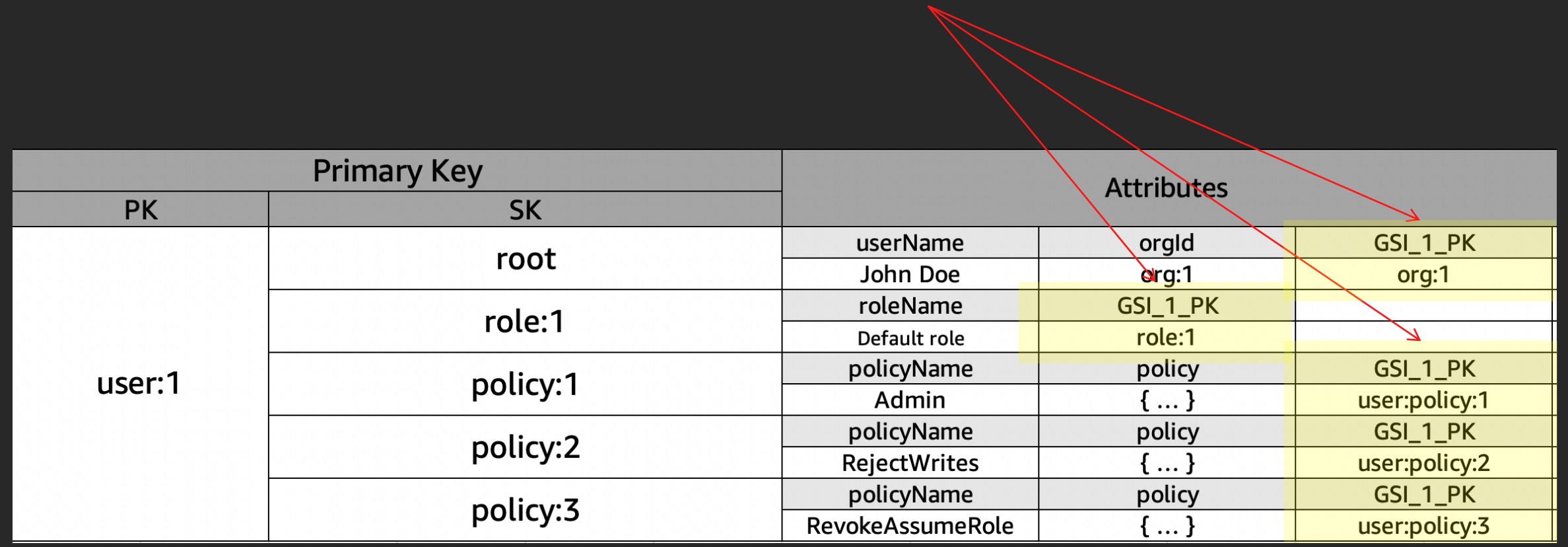
Salt partition keys to support high-density aggregations on GSIs



Primary Key		Attributes			
PK	SK				
job:1	root	JobName	InputType	GSI_1_PK	GSI_1_SK
		"Transcribe audio"	mp3	Shard#(0-N)	2019-12-31
job:2	root	JobName	InputType	GSI_1_PK	GSI_1_SK
		"Verbatim"	wav	Shard#(0-N)	2019-12-29

GSI overloading

Use generic keys once more to use indexes for multiple access patterns



Primary Key		Attributes		
PK	SK			
user:1	root	userName	orgId	GSI_1_PK
		John Doe	org:1	org:1
	role:1	roleName	GSI_1_PK	
		Default role	role:1	
	policy:1	policyName	policy	GSI_1_PK
		Admin	{ … }	user:policy:1
	policy:2	policyName	policy	GSI_1_PK
		RejectWrites	{ … }	user:policy:2
	policy:3	policyName	policy	GSI_1_PK
		RevokeAssumeRole	{ … }	user:policy:3

GSI overloading (continued)

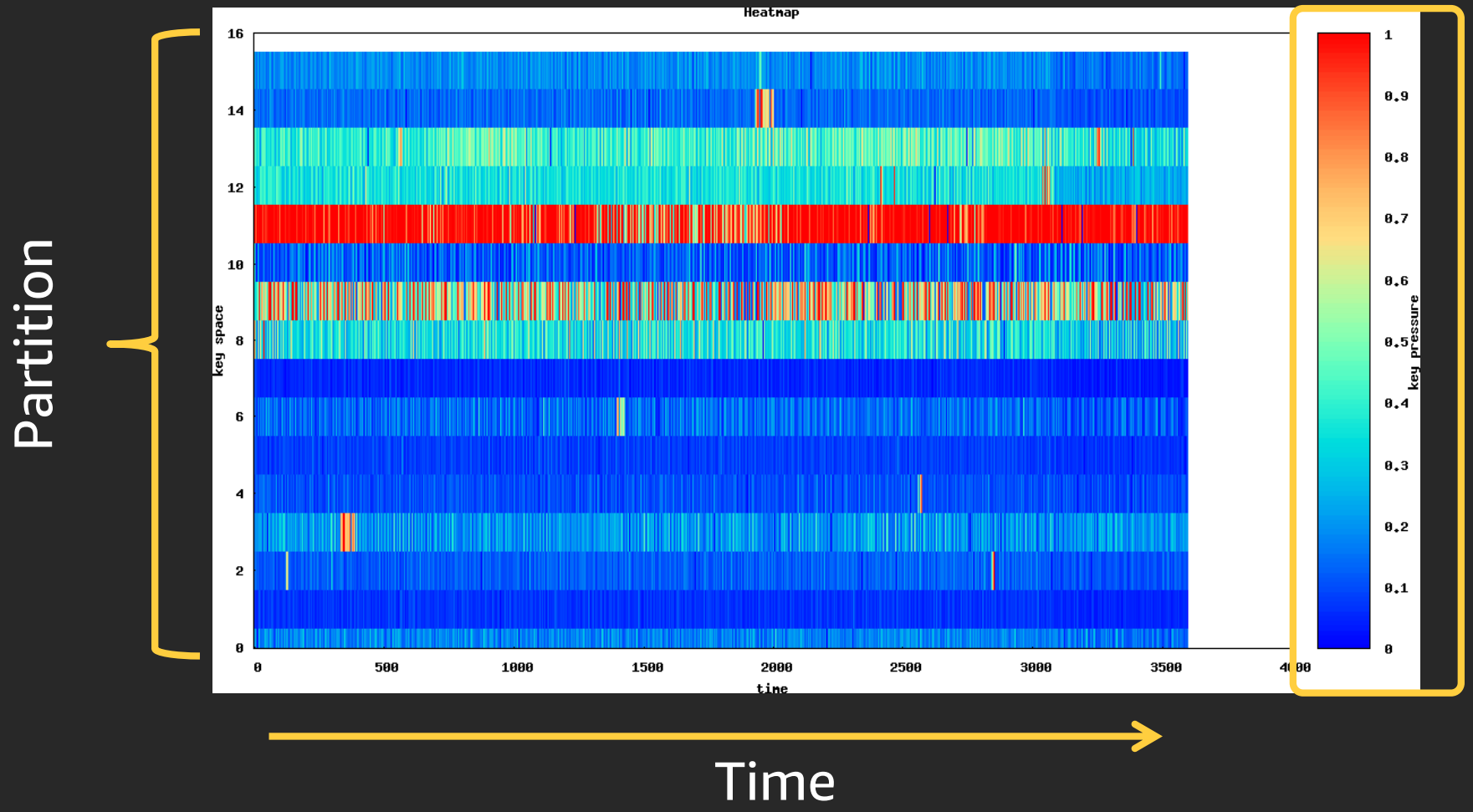
```
SELECT * WHERE PK=ONLINE#0 AND SK=US
…
SELECT * WHERE PK=ONLINE#N AND SK=US
```

Primary Key		Attributes			
GSI1PK	GSI1SK				
Online#(0-N)	US	PK	SK	Store	CustomerType
		Customer_1	2019-11-29T08:31:28Z#O1	www.amazon.com	Regular
B07G6CQQYG#(0-N)	PROCESSING	PK	SK	Product	FCCID
		Customer_1	2019-11-29T08:31:28Z#O1#I1	BOOM 3	JNZS00170

```
SELECT * WHERE PK=B07G6CQQYG#0 AND SK=PROCESSING
…
SELECT * WHERE PK=B07G6CQQYG#N AND SK=PROCESSING
```

Scaling NoSQL

What bad NoSQL looks like…



Getting the most out of DynamoDB throughput

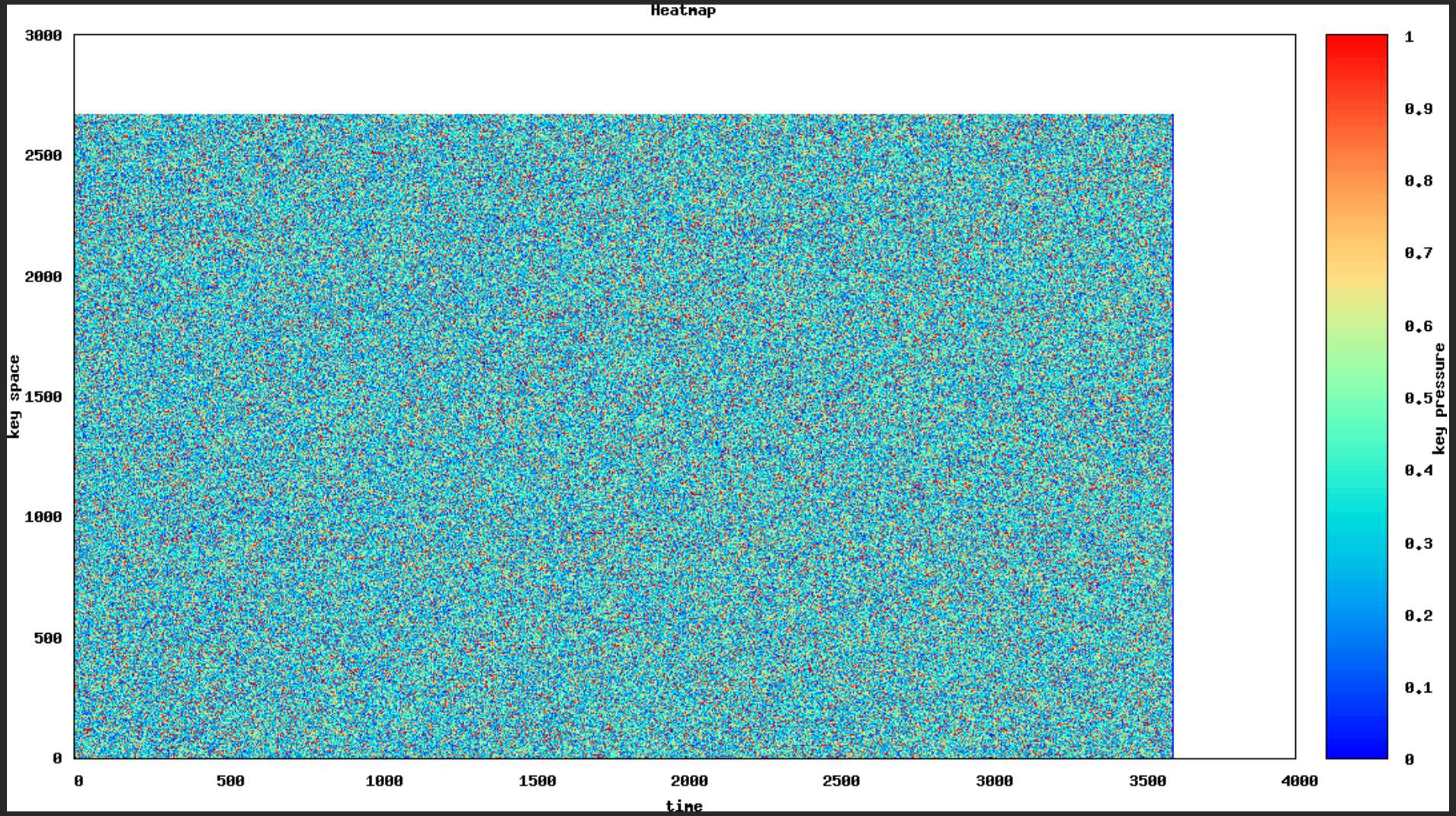
"To get the most out of DynamoDB throughput, create tables where the partition key element has a large number of distinct values, and values are requested fairly uniformly, as randomly as possible."

—*DynamoDB Developer Guide*

Space: access is evenly spread over the keyspace

Time: requests arrive evenly spaced in time

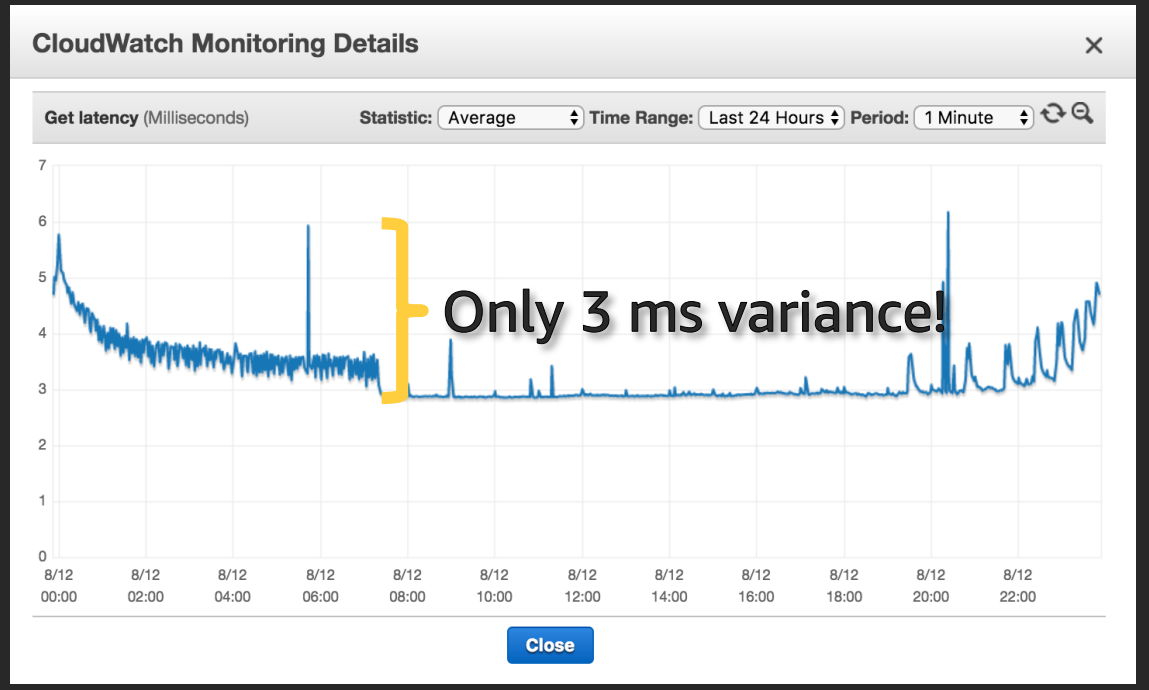
Much better picture…



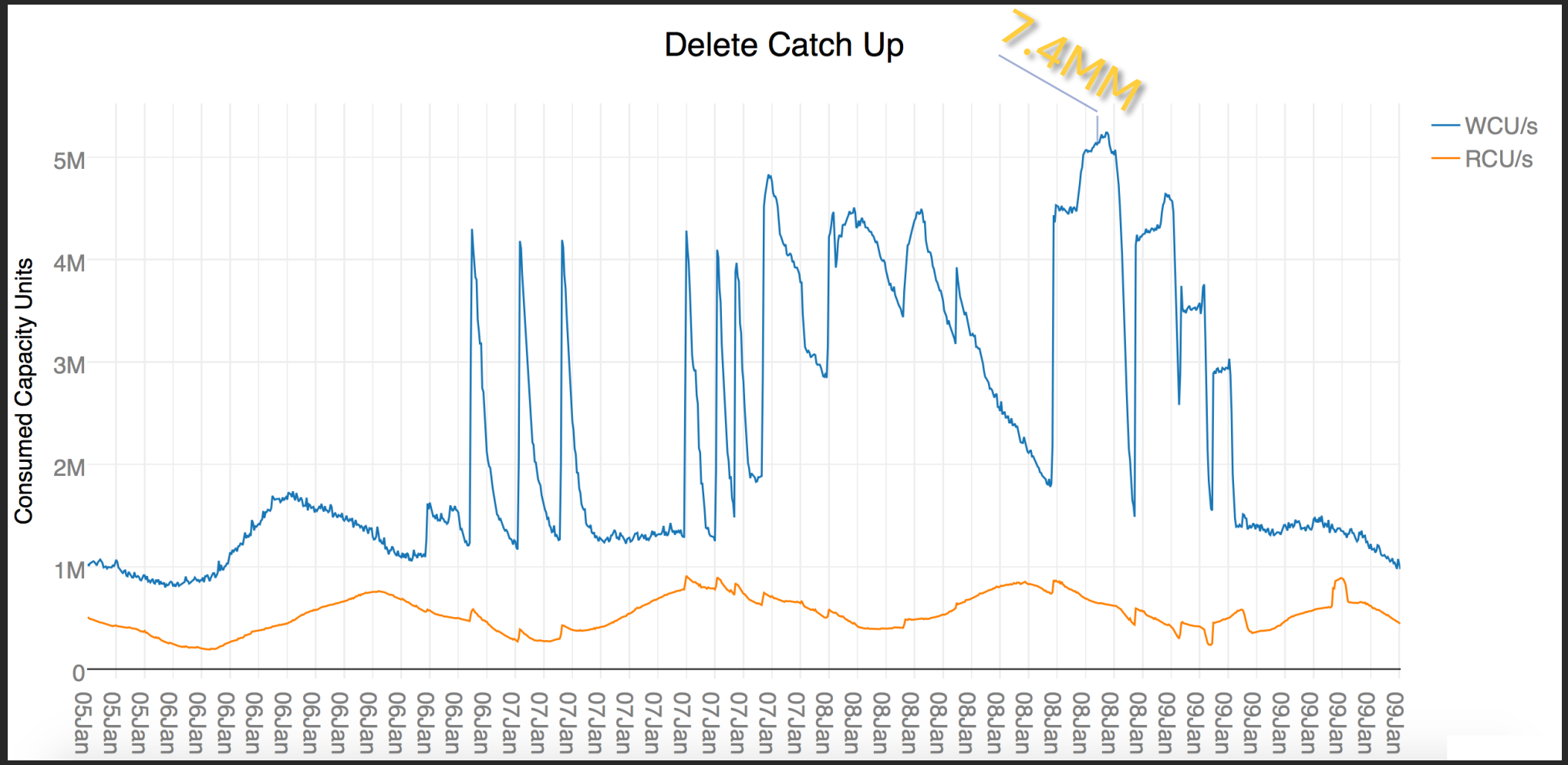
Performance at any scale

Request Volume

Latency



DynamoDB elasticity



NoSQL modeling

All data is relational

IT monitoring

Social graph

Documents

Auth services

Tenets of DynamoDB data modeling

- Understand the use case
- Identify the access patterns
  - Read/write workloads
  - Query dimensions and aggregations
- Data modeling
  - Using NoSQL design patterns
- Review -> Repeat -> Review

- Nature of the data
  - OLTP / OLAP / full-text search
- Relationships between the entities
- What does concurrent access look like?
- Time series data
- Archiving needs, etc.

Tenets of DynamoDB data modeling

- Understand the use case
- Identify the access patterns
  - Read/write workloads
  - Query dimensions and aggregations
- Data modeling
  - Using NoSQL design patterns
- Review -> Repeat -> Review

- Source data analysis (write workload)
- Reading one item versus multiple items (read workload)
- Query aggregations and KPIs

Tenets of DynamoDB data modeling

- Understand the use case
- Identify the access patterns
  - Read/write workloads
  - Query dimensions and aggregations
- Data modeling
  - Using NoSQL design patterns
- Review -> Repeat -> Review

- 1:1, 1:n, m:n relationships
- **1 application = 1 table**
  - Avoid unnecessary fetches
  - Simplify access patterns
- Identify primary key
  - Partition key and Sort key
- Query dimensions using LSIs and GSIs

Tenets of DynamoDB data modeling

- Understand the use case
- Identify the access patterns
  - Read/write workloads
  - Query dimensions and aggregations
- Data modeling
  - Using NoSQL design patterns
- Review -> Repeat -> Review

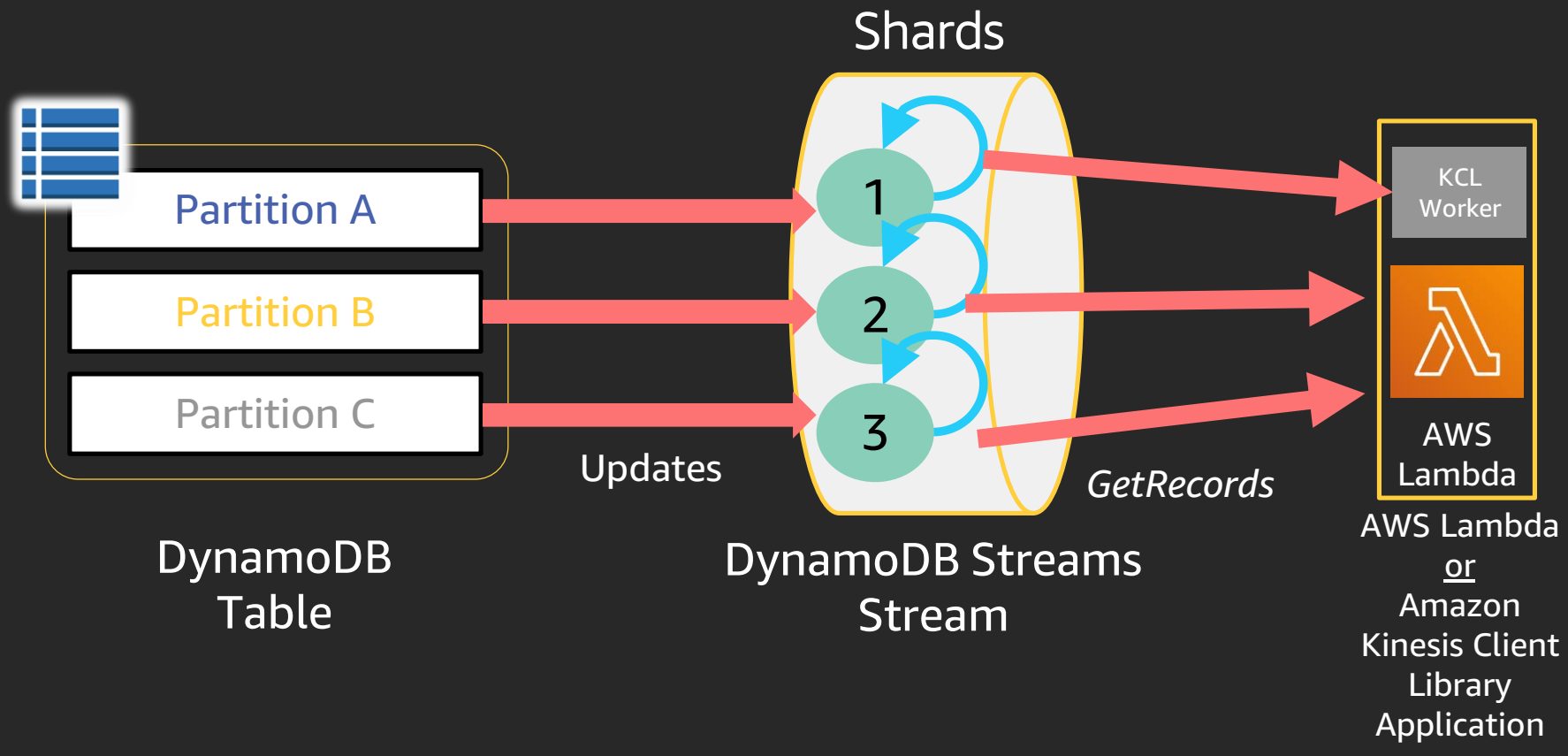
DynamoDB features

DynamoDB Streams

Shards have a lineage and automatically close after time or when the associated DynamoDB partition splits

DynamoDB Streams

- ✓Ordered stream of item changes
- ✓Exactly once, strictly ordered by key
- ✓Highly durable, scalable
- ✓24 hour retention
- ✓Sub-second latency
- ✓Fully integrated with AWS Lambda

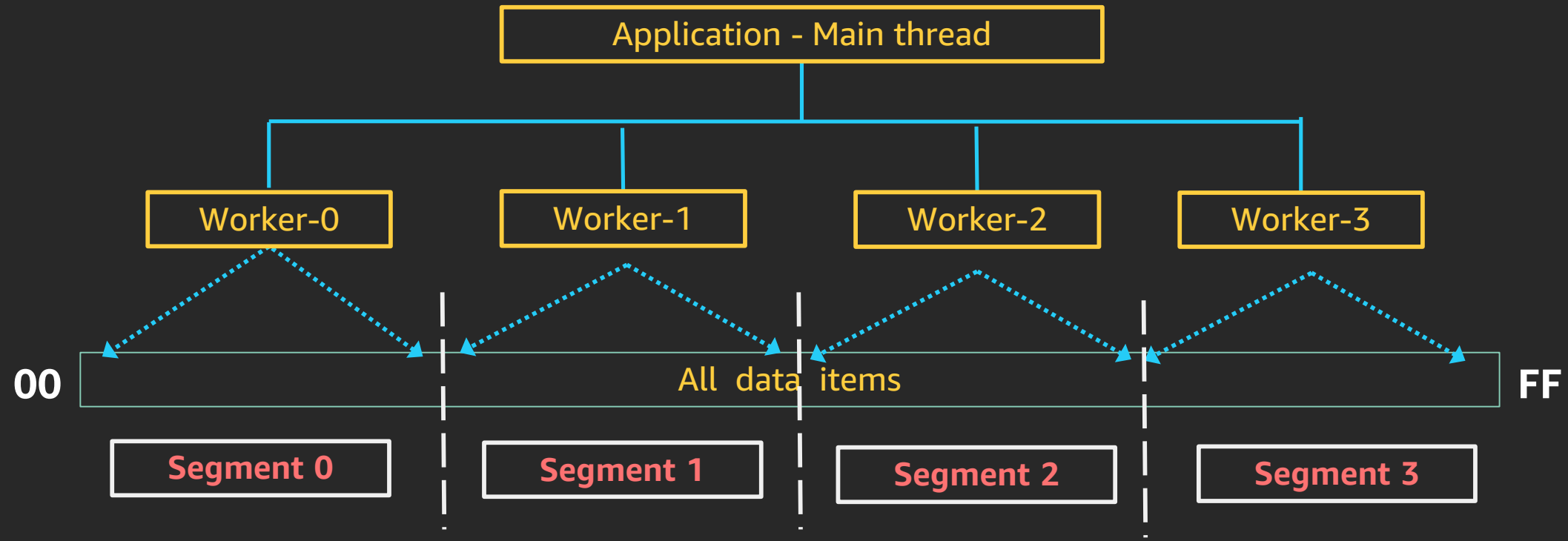


DynamoDB Scan operations

- Access every item in a table on an index
- Read 1-MB data in each operation
  - Use `LastEvaluatedKey` to continue
- Reads up to the maximum throughput of a single partition
- Parallel scans versus sequential scans

Parallel Scan

- Read all the items from a table faster
- Take advantage of the table's provisioned capacity
- Set `TotalSegments` = number of application workers; each worker scans a different *segment*



Sequential versus parallel Scan

Scenario:

Scan server logs data for response code = OK

Sequential Scan

```
fe = "responsecode <> :f"
 eav = {":f": 200}
 response = table.scan(
   FilterExpression=fe,
ExpressionAttributeValues=eav,
   Limit=pageSize
 )
```

Parallel Scan

```
fe = "responsecode <> :f"
  eav = {":f": 200}
  response = table.scan(
    FilterExpression=fe,
 ExpressionAttributeValues=eav,
    Limit=pageSize,
          TotalSegments=totalsegments,
             Segment=threadsegment
           )
```

Lab preparation

During the lab…

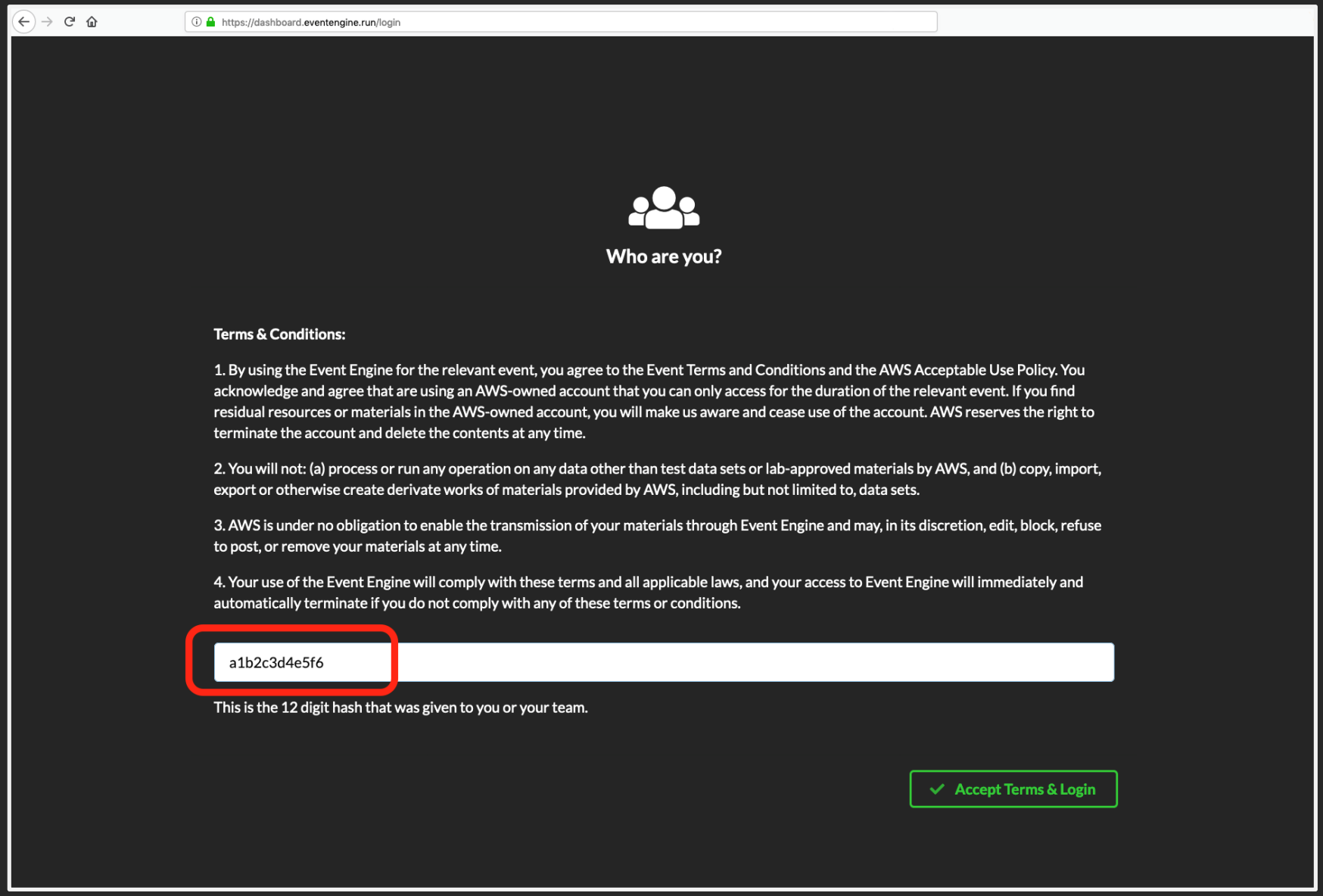
1. Questions that are asked will be sent to the main presenter to answer
2. During the lab, the presenter will be running through it in front of you. Watch along if you don’t have a laptop, or want to see the process
3. Please fill out your surveys at the end

AWS Event Engine

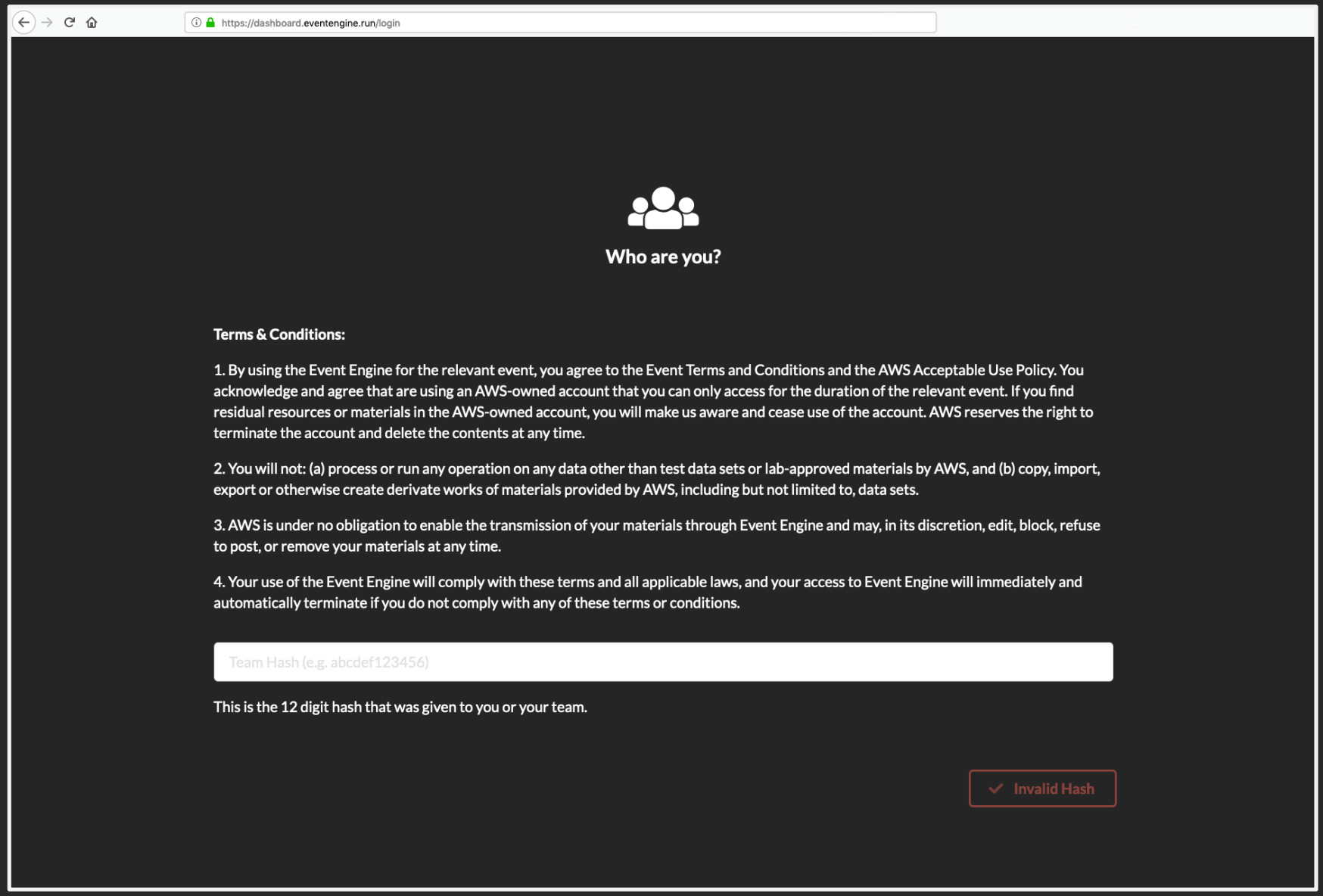
https://dashboard.eventengine.run

The AWS Event Engine was created to help AWS field teams run Workshops, GameDays, Bootcamps, Immersion Days, and other events that require hands-on access to AWS accounts.

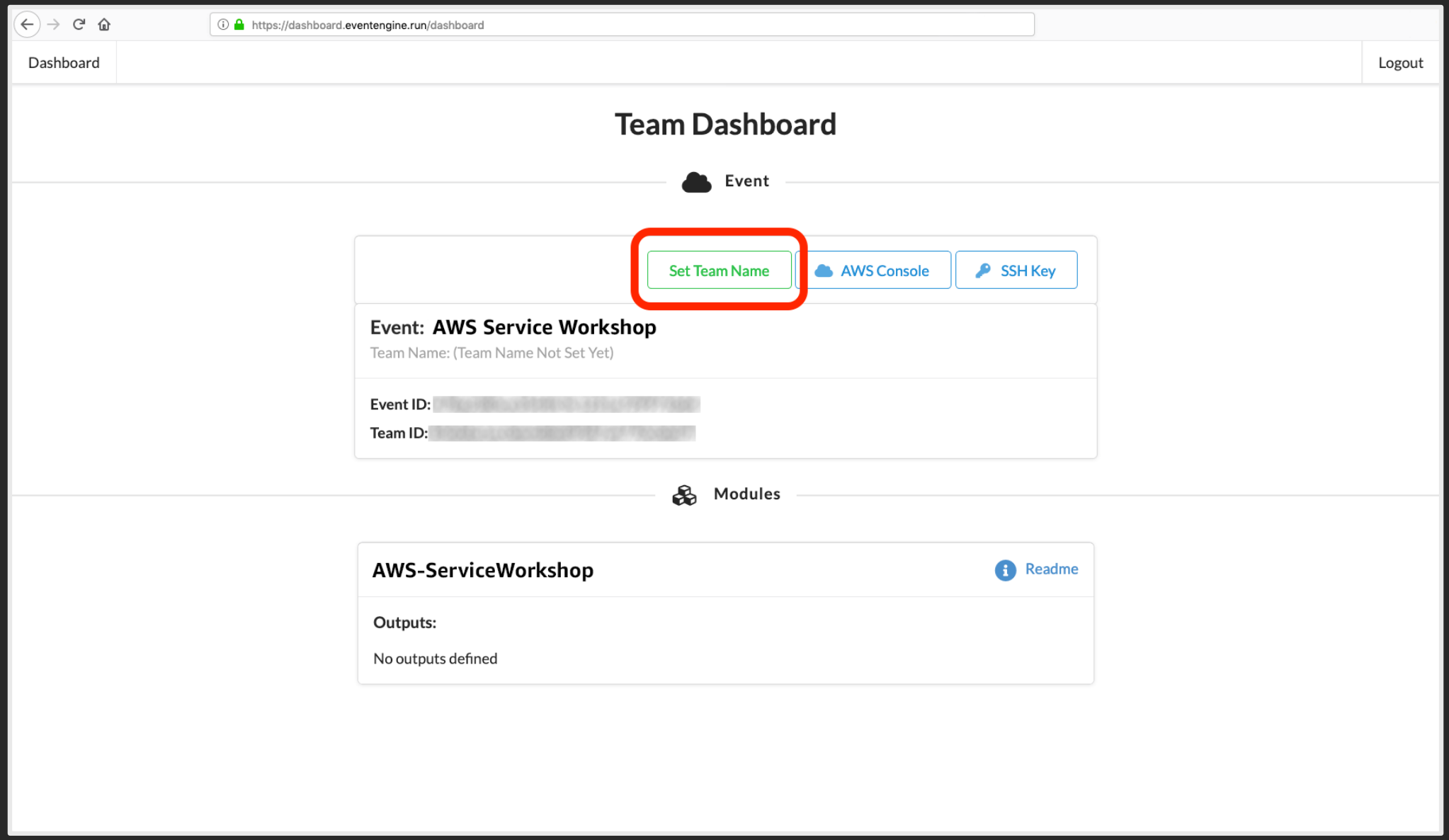
https://dashboard.eventengine.run



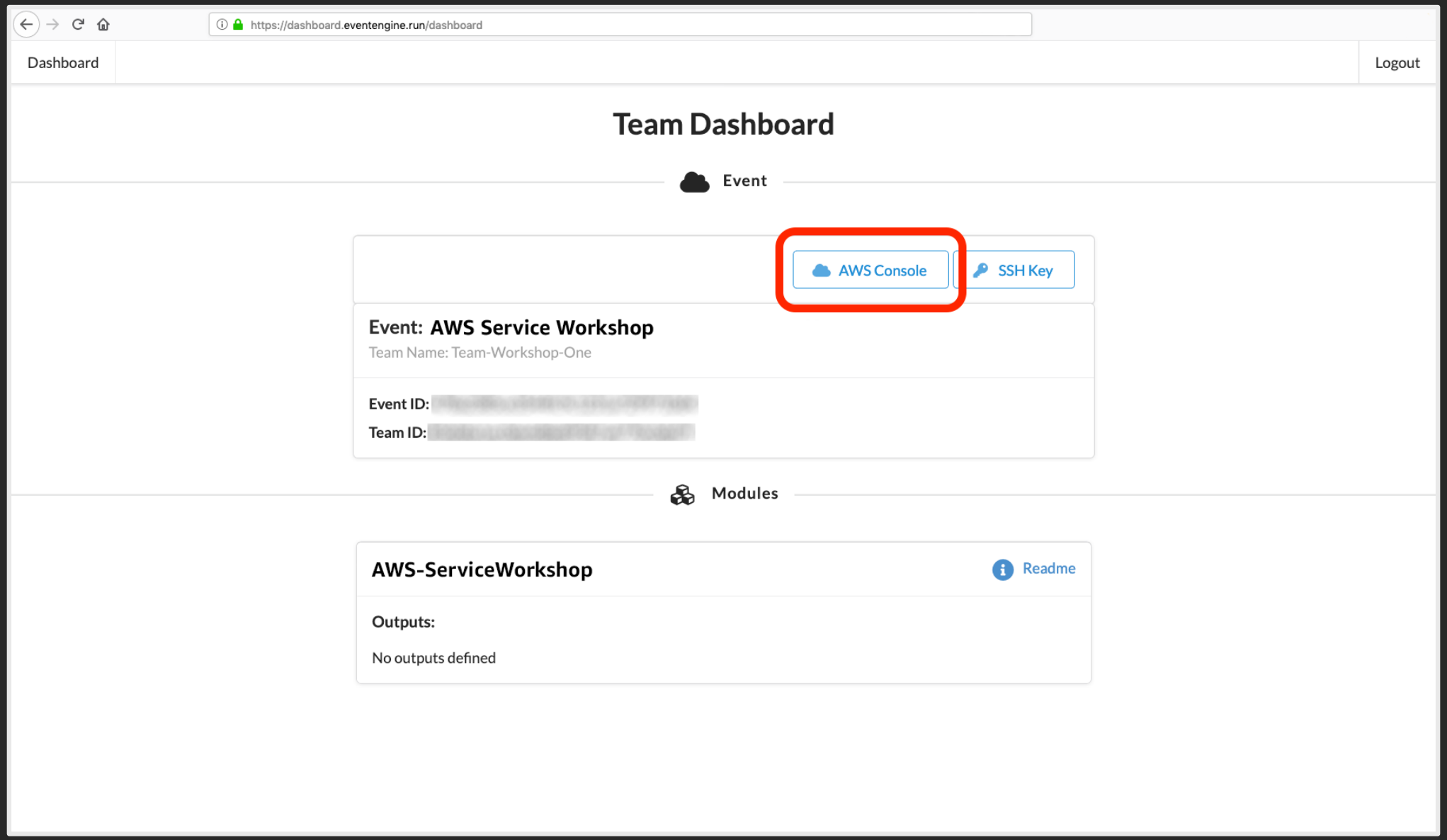
https://dashboard.eventengine.run



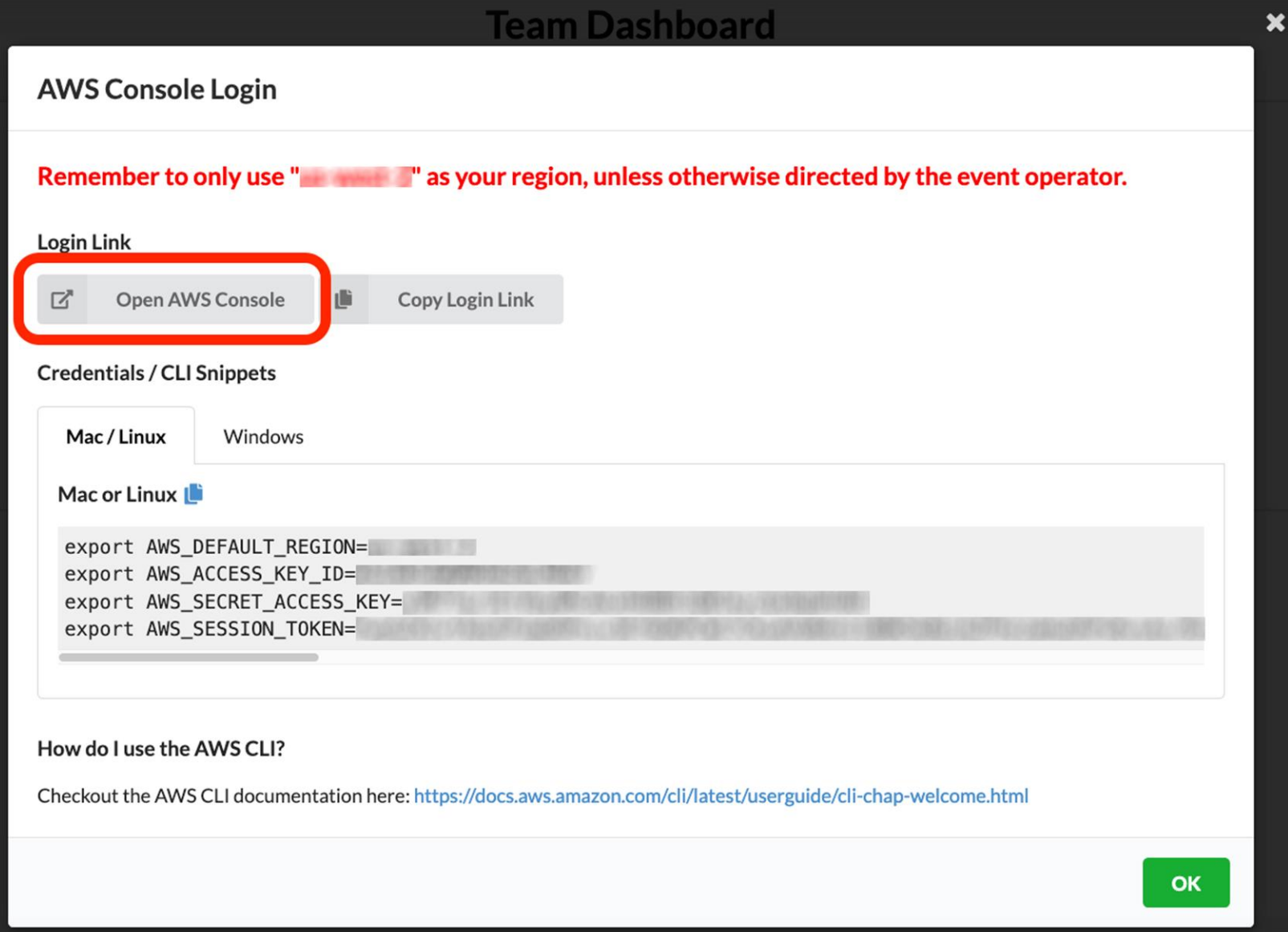
https://dashboard.eventengine.run/dashboard



https://dashboard.eventengine.run/dashboard

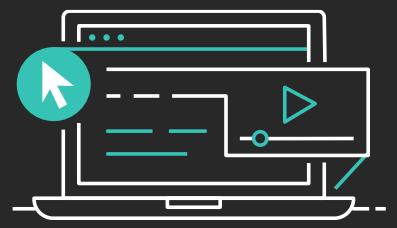


https://dashboard.eventengine.run/dashboard



Learn databases with AWS Training and Certification

Resources created by the experts at AWS to help you build and validate database skills



25+ free digital training courses cover topics and services related to databases, including:

- Amazon Aurora
- Amazon Neptune
- Amazon DocumentDB
- Amazon DynamoDB
- Amazon ElastiCache
- Amazon Redshift
- Amazon RDS

Validate expertise with the new **AWS Certified Database - Specialty** beta exam

Visit aws.training

aws training and certification

Thank you!

Sean Shriver, Daniel Yoder, and Rick Houlihan

@sean_shriver, @daniel_yod,
@houlihan_rick, and @DynamoDB

Please complete the session survey in the mobile app.