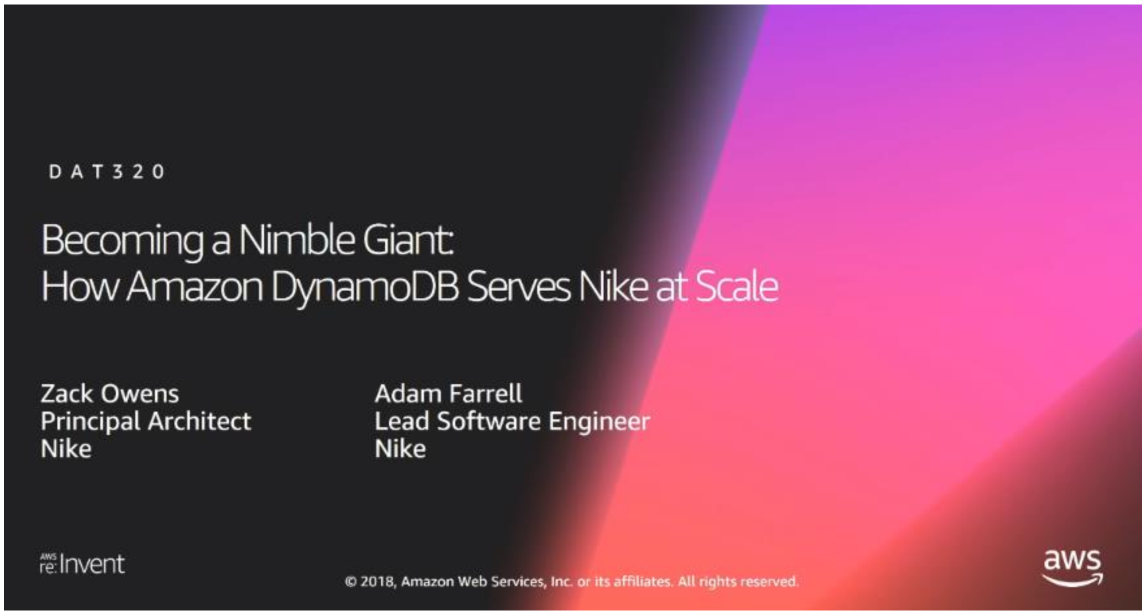


DAT320

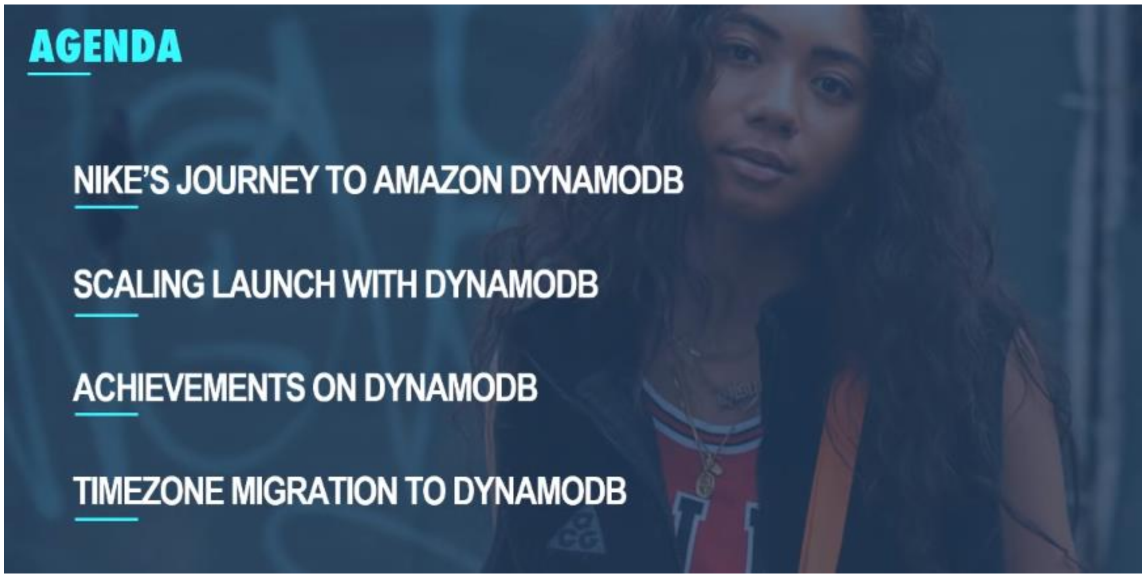
Becoming a Nimble Giant: How Amazon DynamoDB Serves Nike at Scale

Zack Owens
Principal Architect
Nike

Adam Farrell
Lead Software Engineer
Nike

re:Invent

aws



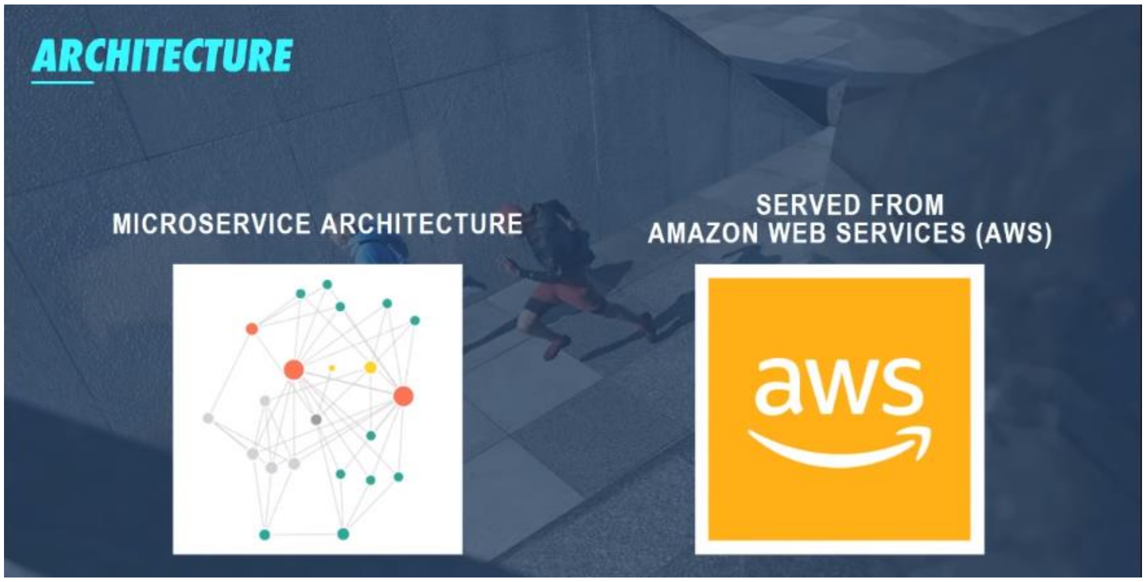
AGENDA

- NIKE'S JOURNEY TO AMAZON DYNAMODB
- SCALING LAUNCH WITH DYNAMODB
- ACHIEVEMENTS ON DYNAMODB
- TIMEZONE MIGRATION TO DYNAMODB

SERVING ATHLETES* THROUGH DIGITAL EXPERIENCES

* IF YOU HAVE A BODY, YOU ARE AN ATHLETE.

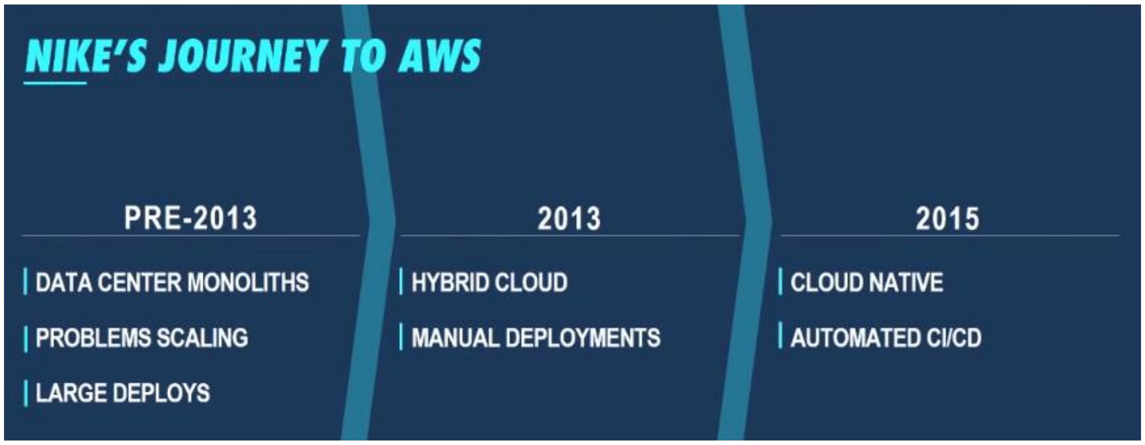
Nike Digital Engineering



ARCHITECTURE

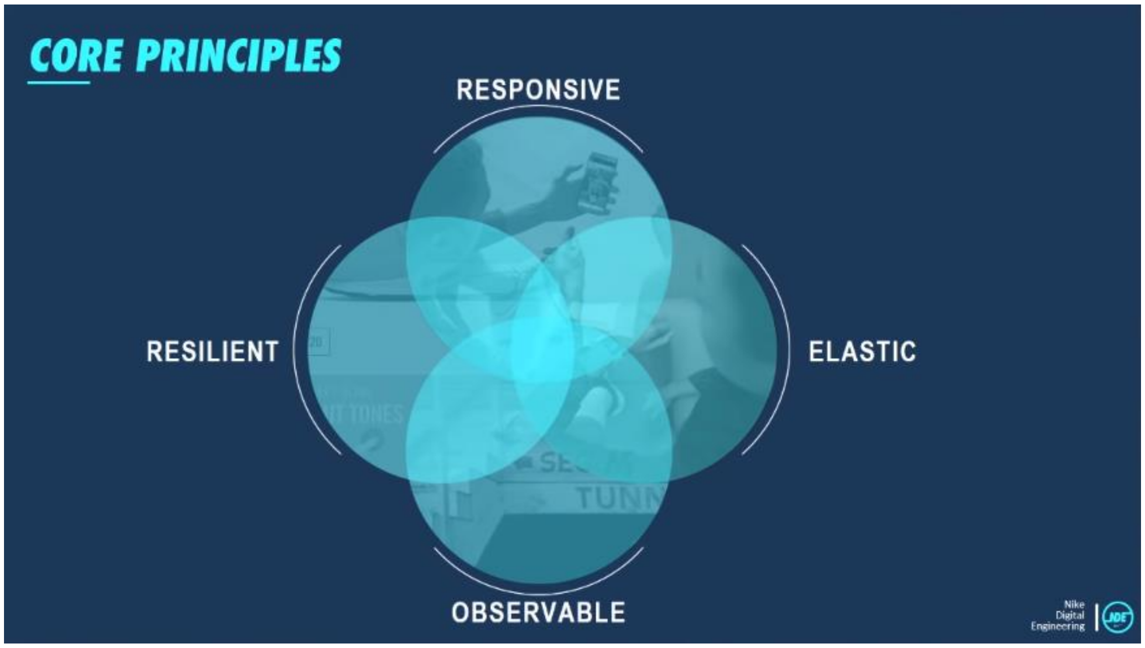
MICROSERVICE ARCHITECTURE

SERVED FROM
AMAZON WEB SERVICES (AWS)

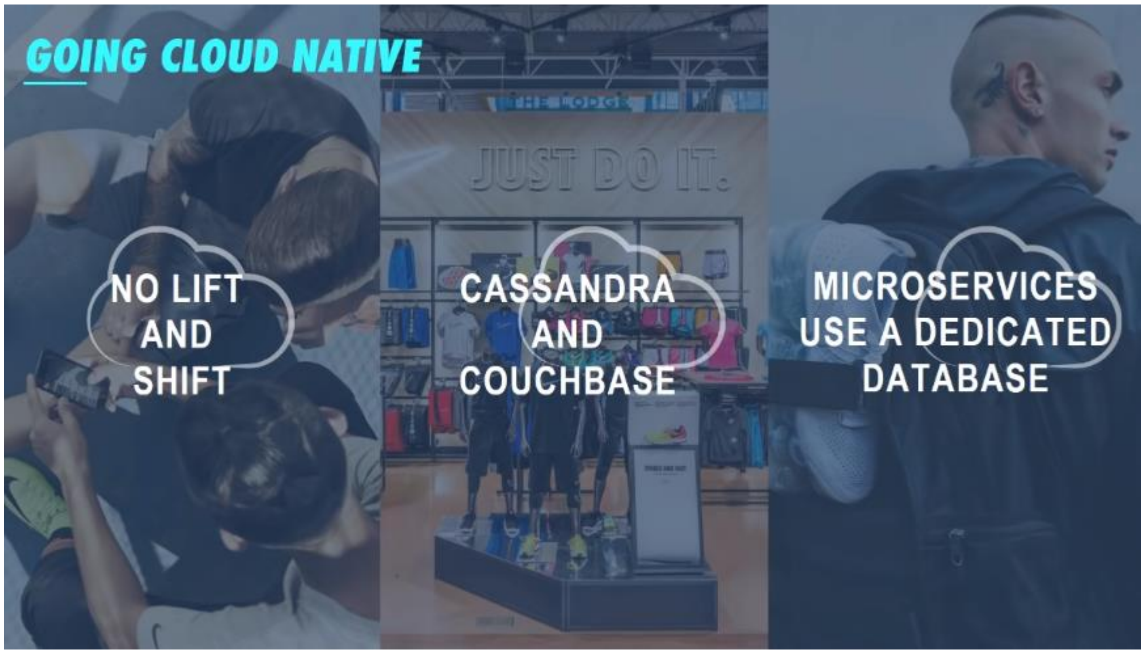


NIKE'S JOURNEY TO AWS

PRE-2013	2013	2015
DATA CENTER MONOLITHS	HYBRID CLOUD	CLOUD NATIVE
PROBLEMS SCALING	MANUAL DEPLOYMENTS	AUTOMATED CI/CD
LARGE DEPLOYS		



CORE PRINCIPLES

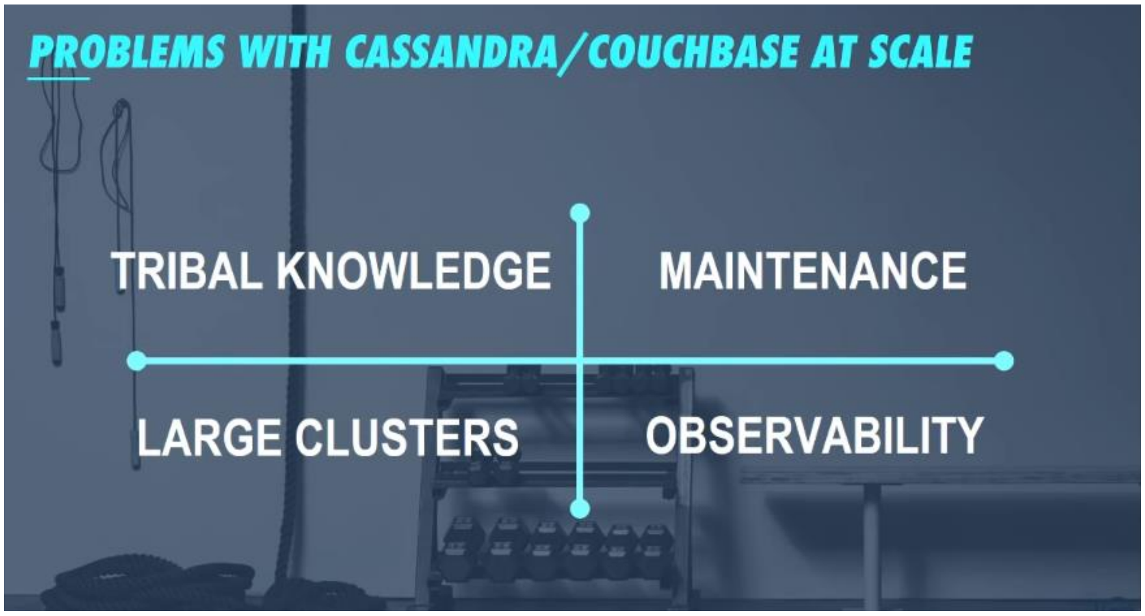


GOING CLOUD NATIVE

NO LIFT AND SHIFT

CASSANDRA AND COUCHBASE

MICROSERVICES USE A DEDICATED DATABASE



PROBLEMS WITH CASSANDRA/COUCHBASE AT SCALE

TRIBAL KNOWLEDGE

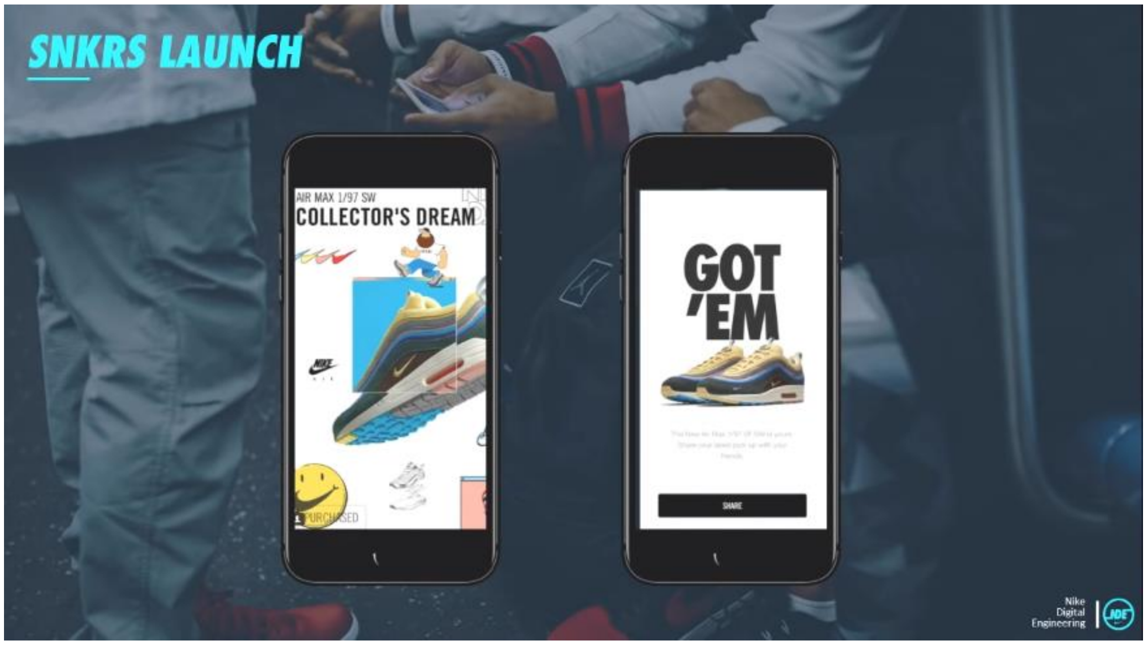
MAINTENANCE

LARGE CLUSTERS

OBSERVABILITY

SCALING LAUNCH

WITH DYNAMODB

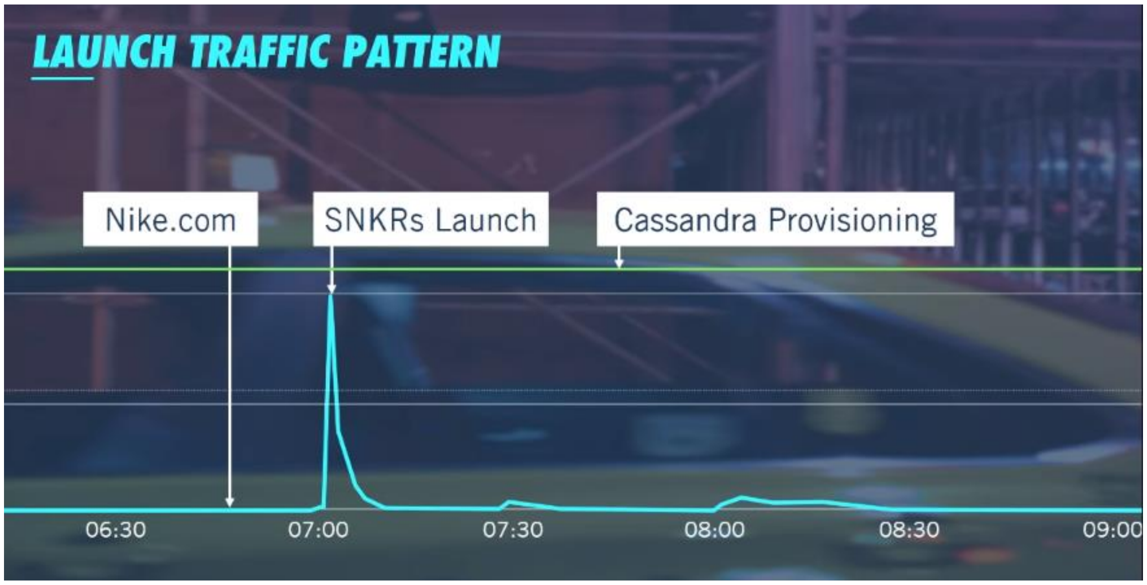


SNKRS LAUNCH

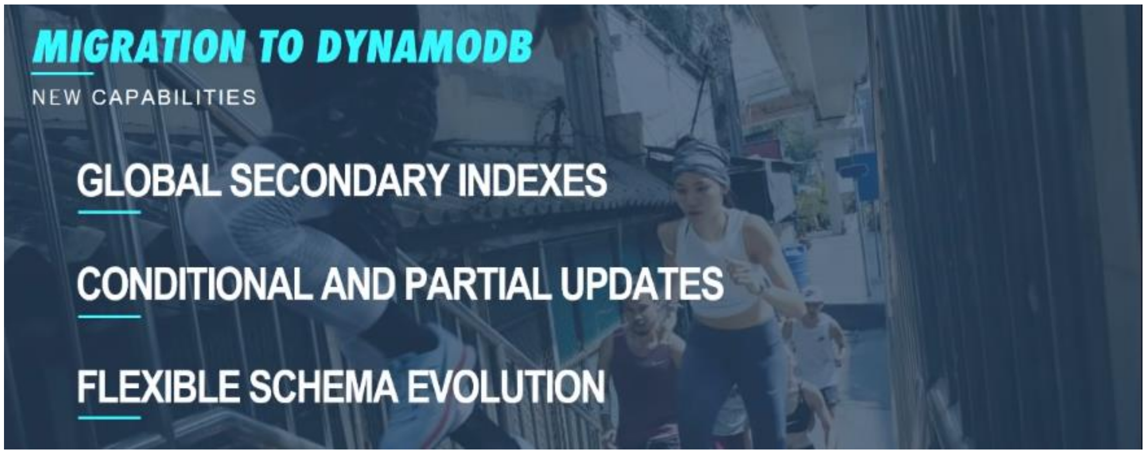
LAUNCH IN THE CLOUD

HIGH-DEMAND PRODUCT

HIGH-THROUGHPUT TRAFFIC



LAUNCH TRAFFIC PATTERN

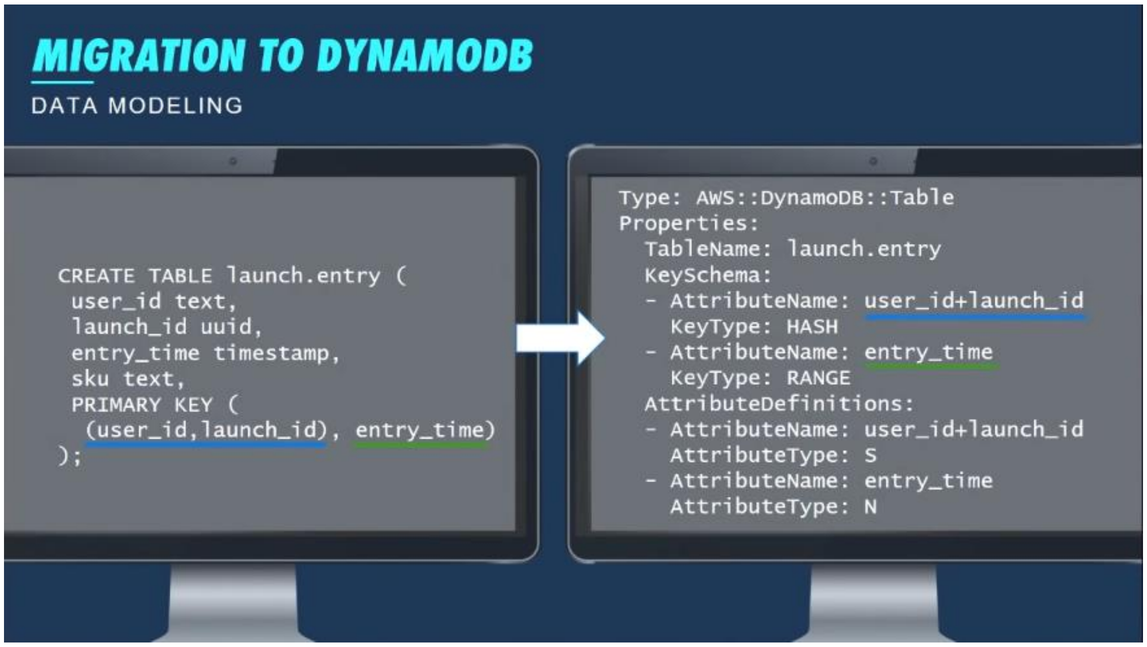


MIGRATION TO DYNAMODB

NEW CAPABILITIES

- GLOBAL SECONDARY INDEXES
- CONDITIONAL AND PARTIAL UPDATES
- FLEXIBLE SCHEMA EVOLUTION

The DynamoDB feature set is extensive and solves a lot of our problems. **GSI's allows you to query data using a hash and range key that is different from your table**, this allows multiple different, efficient query patterns to be possible. **GSI's can also be applied after the table has been created and has data inside it**, you can **add new indexes with a single API call**. This allows us to update our data models as needed.



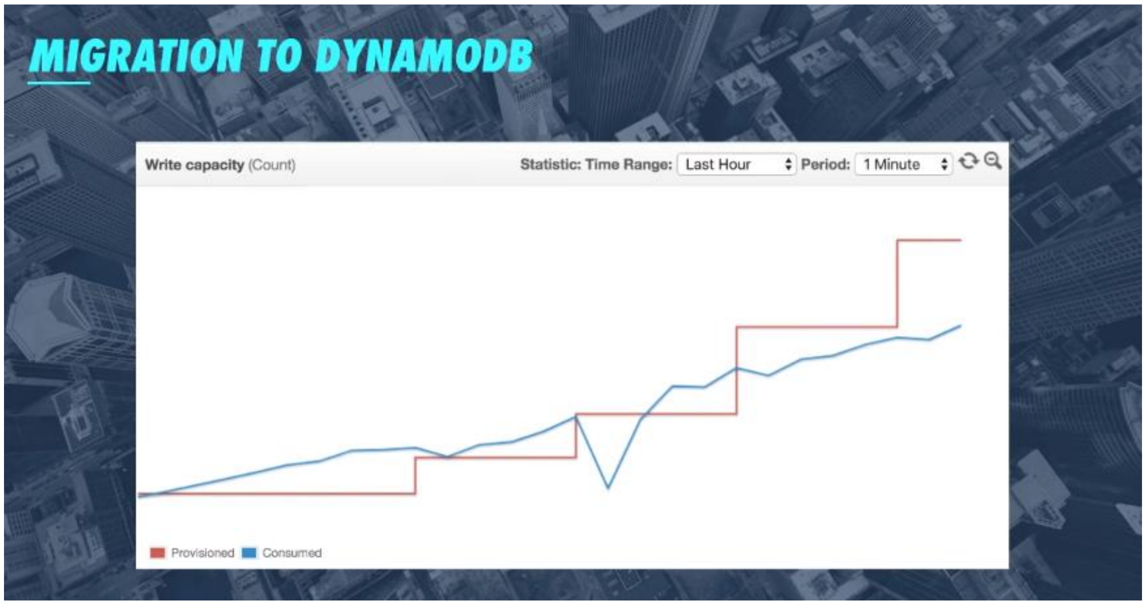
MIGRATION TO DYNAMODB

DATA MODELING

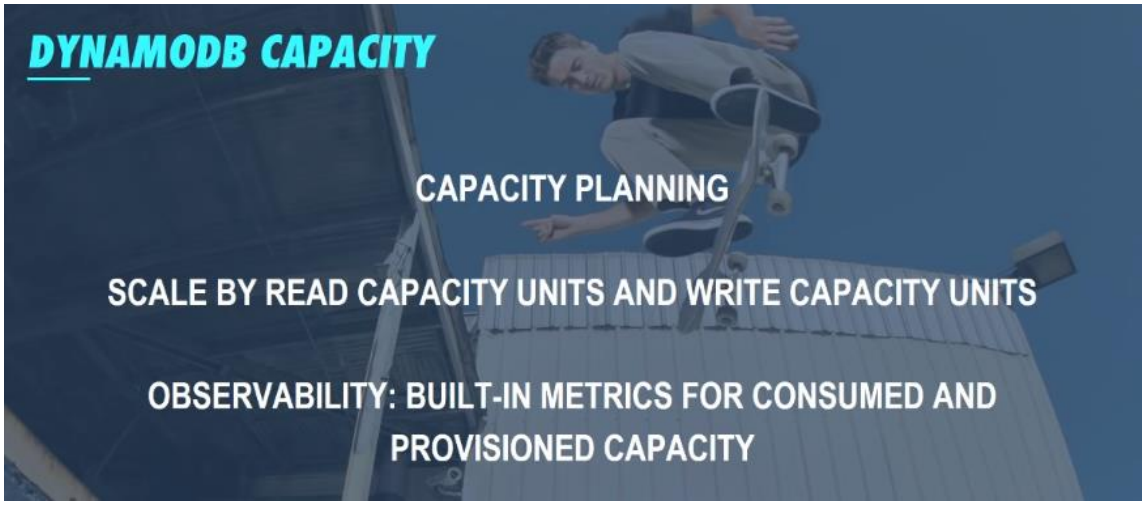
```
CREATE TABLE launch.entry (
 user_id text,
 launch_id uuid,
 entry_time timestamp,
 sku text,
 PRIMARY KEY (
  (user_id,launch_id), entry_time)
);
```

```
Type: AWS::DynamoDB::Table
Properties:
  TableName: launch.entry
  KeySchema:
  - AttributeName: user_id+launch_id
    KeyType: HASH
  - AttributeName: entry_time
    KeyType: RANGE
  AttributeDefinitions:
  - AttributeName: user_id+launch_id
    AttributeType: S
  - AttributeName: entry_time
    AttributeType: N
```

In DynamoDB on the right, we can use 2 keys in our query by using string concatenation to create a unique key like **user_id+launch_id**.



MIGRATION TO DYNAMODB

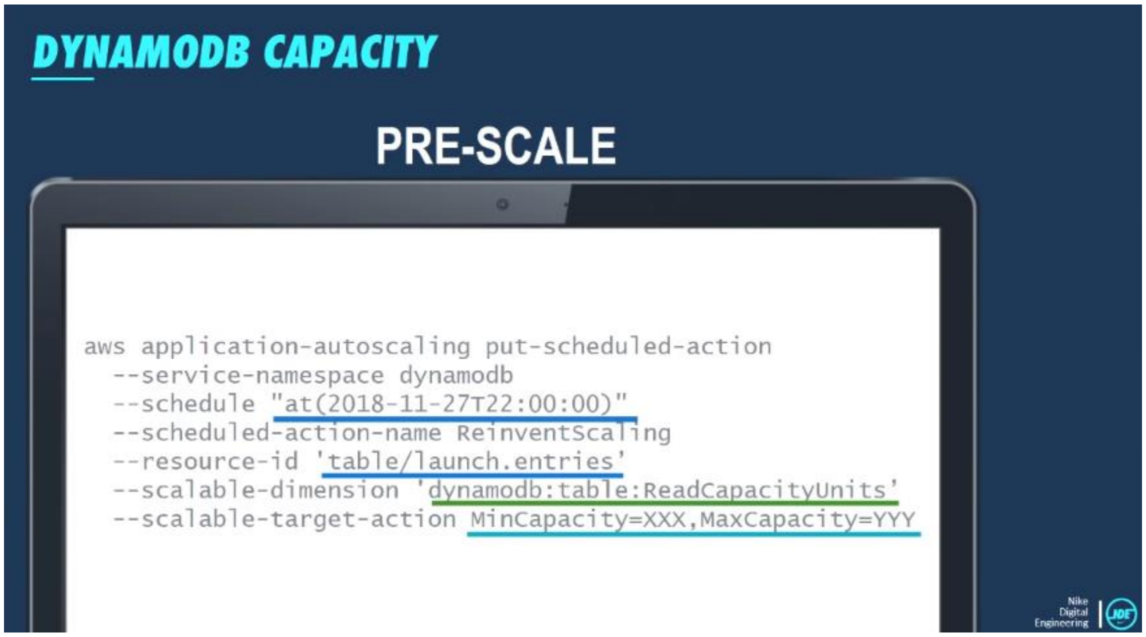


DYNAMODB CAPACITY

CAPACITY PLANNING

SCALE BY READ CAPACITY UNITS AND WRITE CAPACITY UNITS

OBSERVABILITY: BUILT-IN METRICS FOR CONSUMED AND PROVISIONED CAPACITY



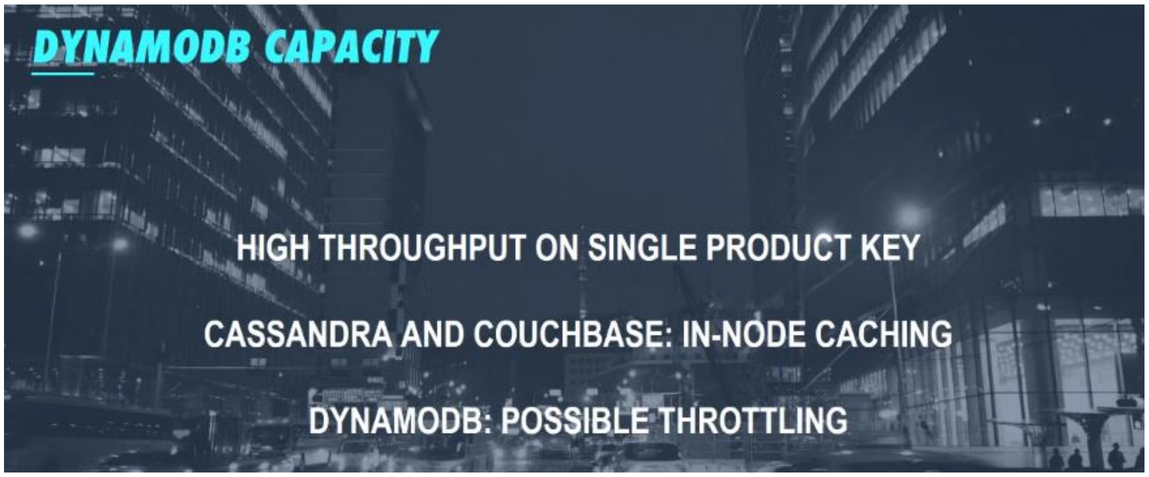
DYNAMODB CAPACITY

PRE-SCALE

```
aws application-autoscaling put-scheduled-action
  --service-namespace dynamodb
  --schedule "at(2018-11-27T22:00:00)"
  --scheduled-action-name ReinventScaling
  --resource-id 'table/launch.entries'
  --scalable-dimension 'dynamodb:table:ReadCapacityUnits'
  --scalable-target-action MinCapacity=XXX,MaxCapacity=YYY
```

Nike Digital Engineering

We also use pre-scaling because auto-scaling is not fast enough to respond to our user traffic spike

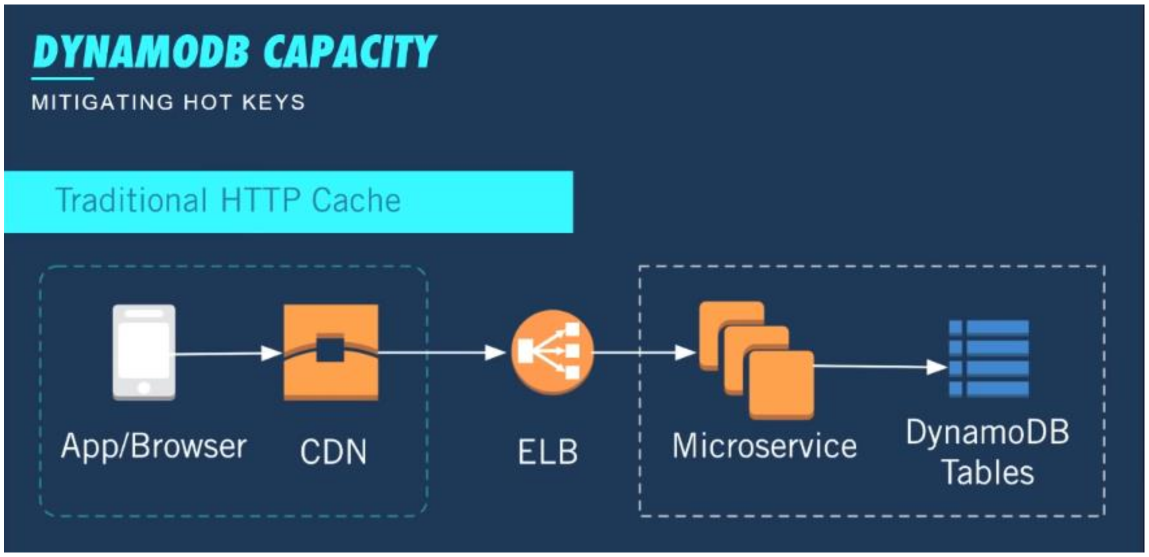


DYNAMODB CAPACITY

HIGH THROUGHPUT ON SINGLE PRODUCT KEY

CASSANDRA AND COUCHBASE: IN-NODE CACHING

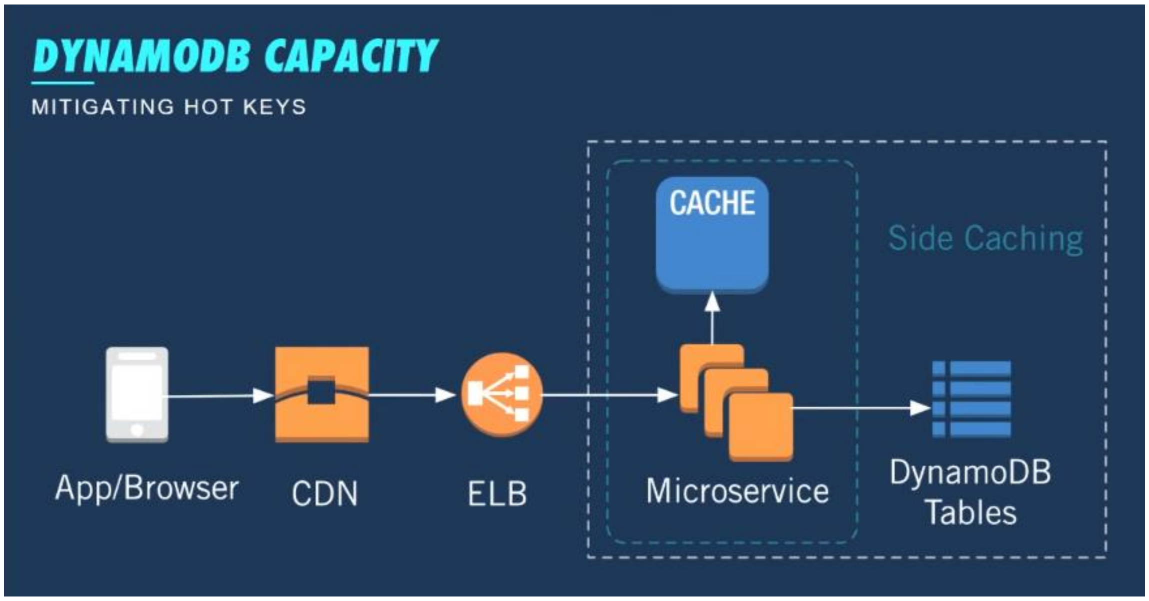
DYNAMODB: POSSIBLE THROTTLING



DYNAMODB CAPACITY

MITIGATING HOT KEYS

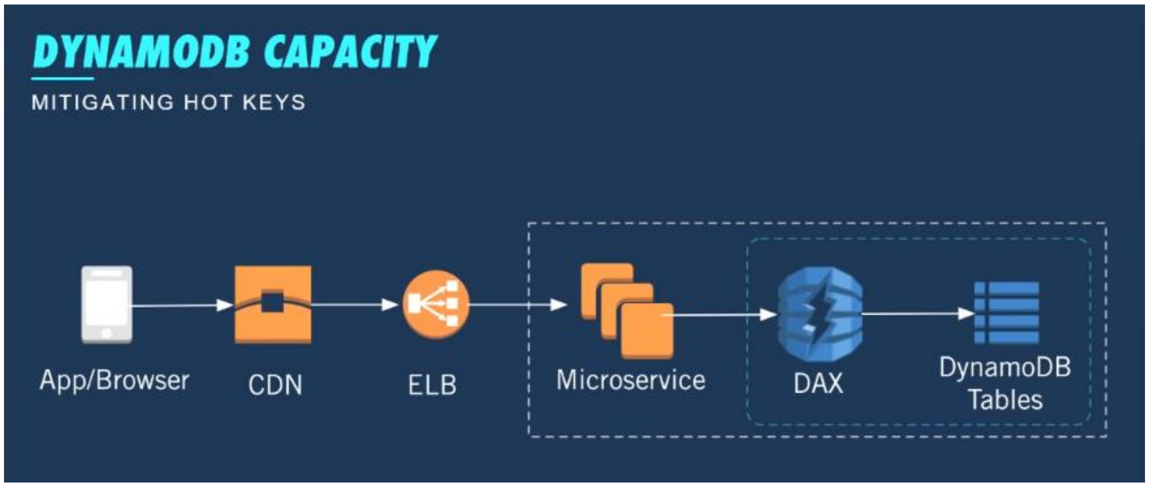
The first place we cache is for full HTTP responses at the CDN layer on different data centers around the world



DYNAMODB CAPACITY

MITIGATING HOT KEYS

We then use a side-cache using memcache to keep some data in-memory for each microservice

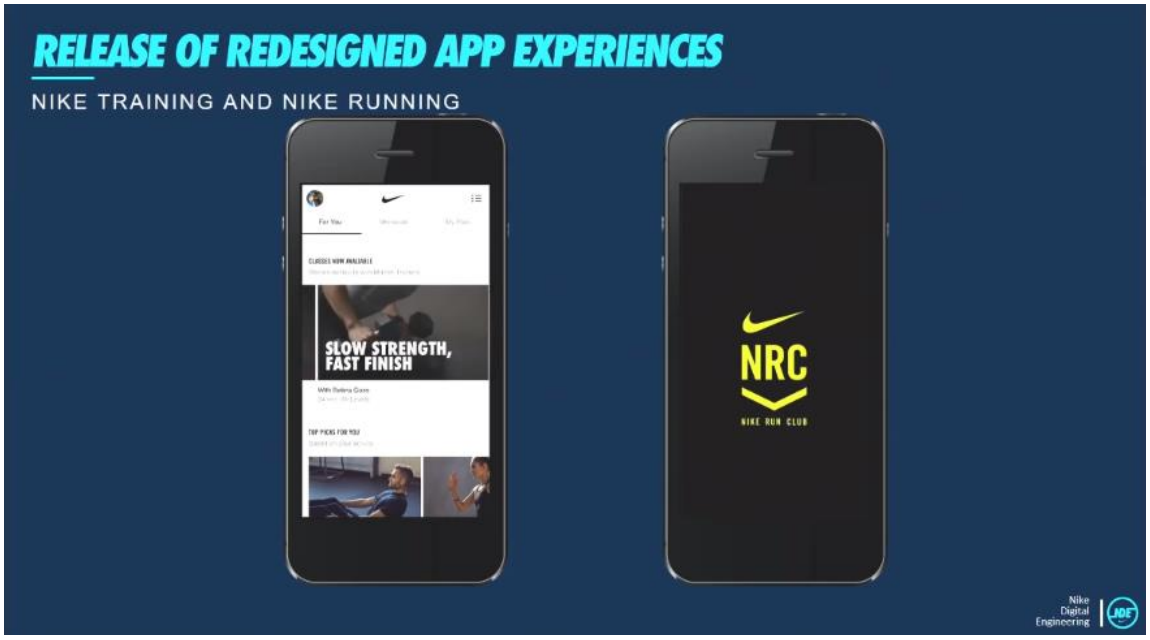


DYNAMODB CAPACITY

MITIGATING HOT KEYS

DAX sits in front of your DynamoDB and can do the caching for you effectively

ACHIEVEMENTS

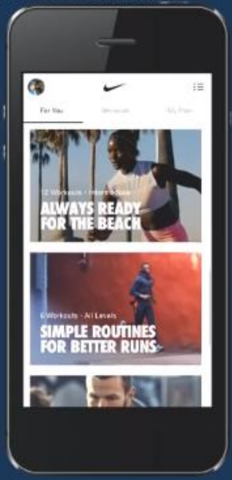


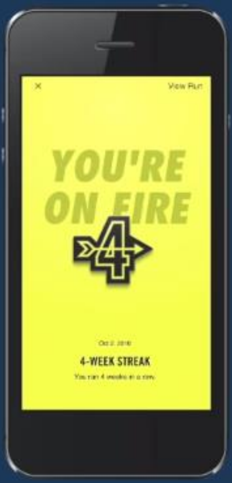
RELEASE OF REDESIGNED APP EXPERIENCES

NIKE TRAINING AND NIKE RUNNING

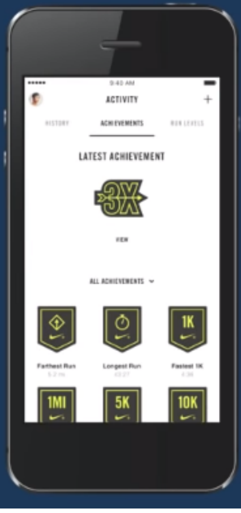
RELEASE OF REDESIGNED APP EXPERIENCES

NIKE TRAINING AND NIKE RUNNING



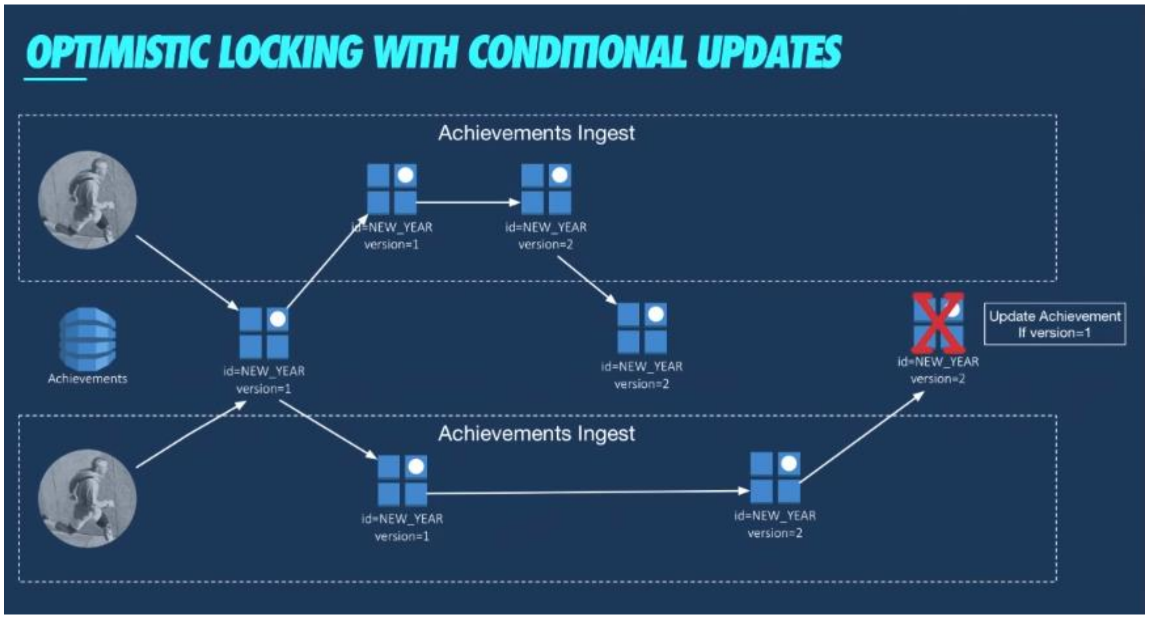


ACHIEVEMENTS WAS NOT IN THE CLOUD IN FIRST RELEASE

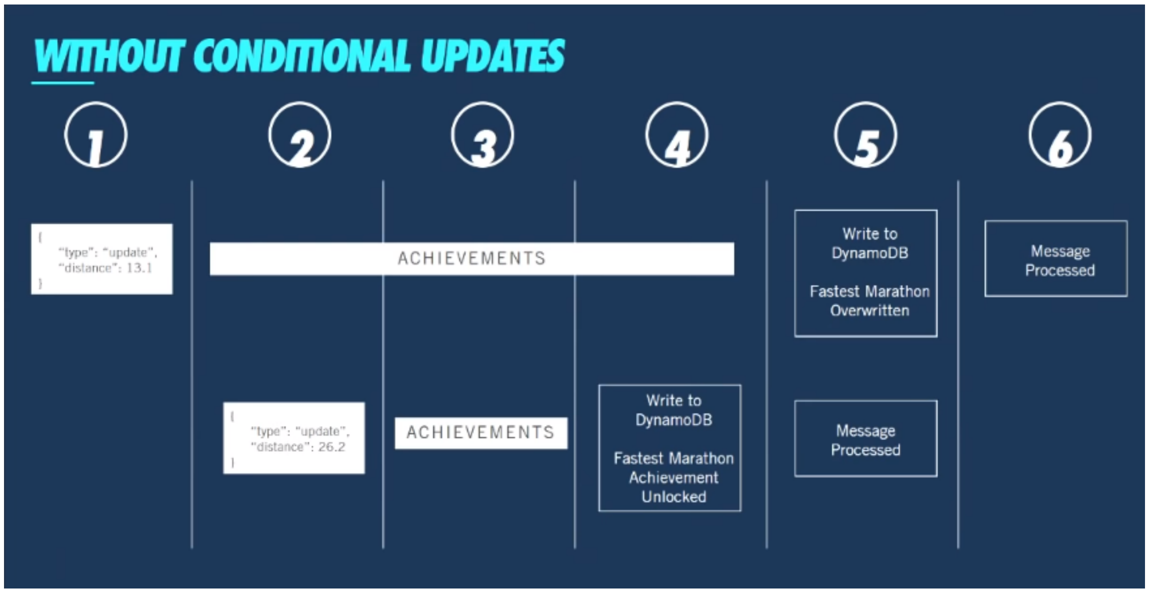


ACHIEVEMENTS RUNS WITH DYNAMODB

SCALABLE	COST EFFICIENT
DURABLE	OBSERVABLE
ELASTIC	CONDITIONAL UPDATES

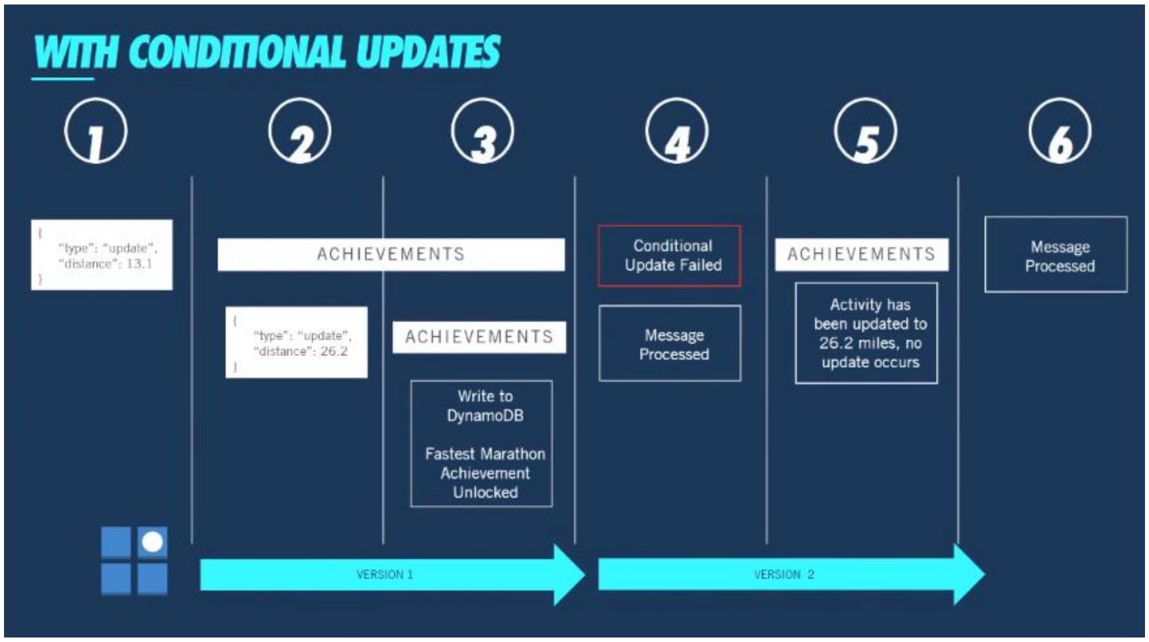


OPTIMISTIC LOCKING WITH CONDITIONAL UPDATES



WITHOUT CONDITIONAL UPDATES

1
2
3
4
5
6
{ "type": "update", "distance": 13.1 }
ACHIEVEMENTS
Write to DynamoDB Fastest Marathon Overwritten
Message Processed
{ "type": "update", "distance": 26.2 }
ACHIEVEMENTS
Write to DynamoDB Fastest Marathon Achievement Unlocked
Message Processed



WITH CONDITIONAL UPDATES

1
2
3
4
5
6
{ "type": "update", "distance": 13.1 }
ACHIEVEMENTS
Conditional Update Failed
ACHIEVEMENTS
Message Processed
{ "type": "update", "distance": 26.2 }
ACHIEVEMENTS
Message Processed
Activity has been updated to 26.2 miles, no update occurs
Write to DynamoDB Fastest Marathon Achievement Unlocked
VERSION 1
VERSION 2

CHALLENGES

ACHIEVEMENTS

IS IN PRODUCTION

BUT ACHIEVEMENTS STARTED WITH

DYNAMODB

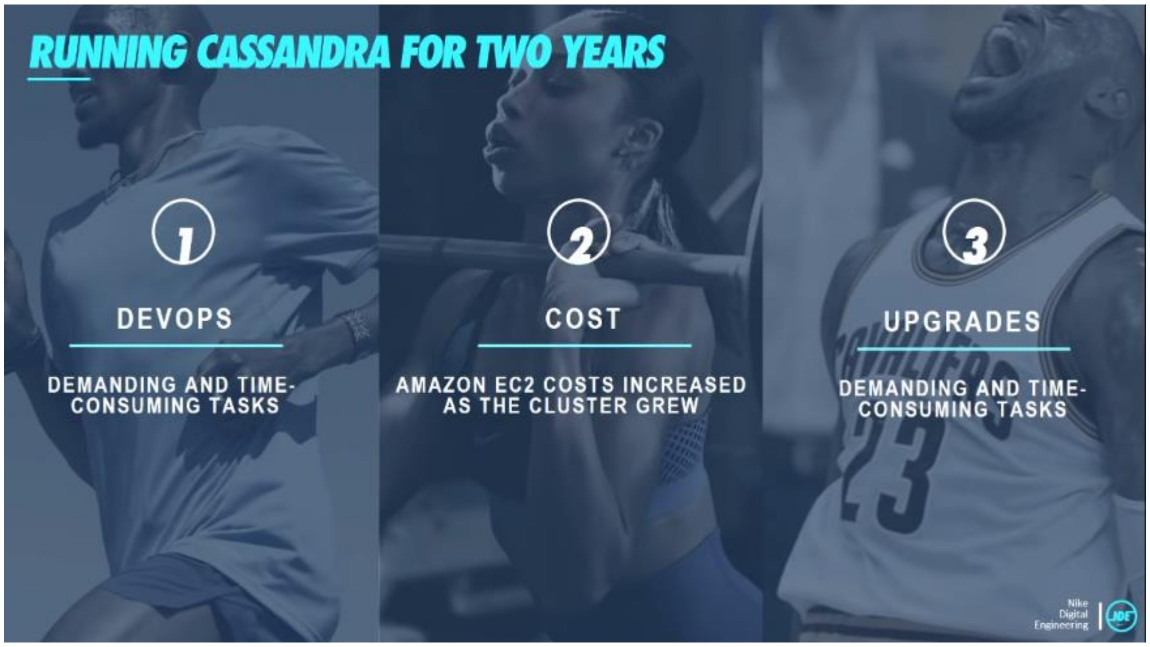
Nike Digital Engineering

TIMEZONE SERVICE

BUILT WITH CASSANDRA

Responsible for keeping track of an athlete's time zones as they move through the world.

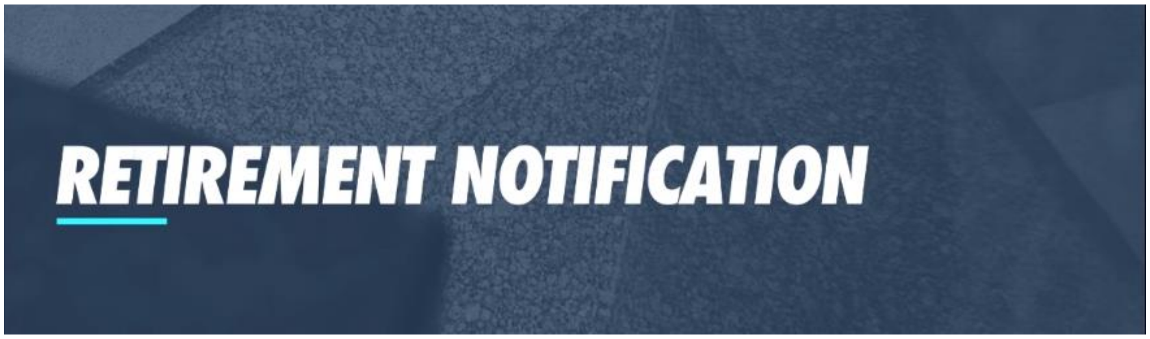
All our activity data is stored in UTC time. We use time zones to adjust the user's activity data when needed.



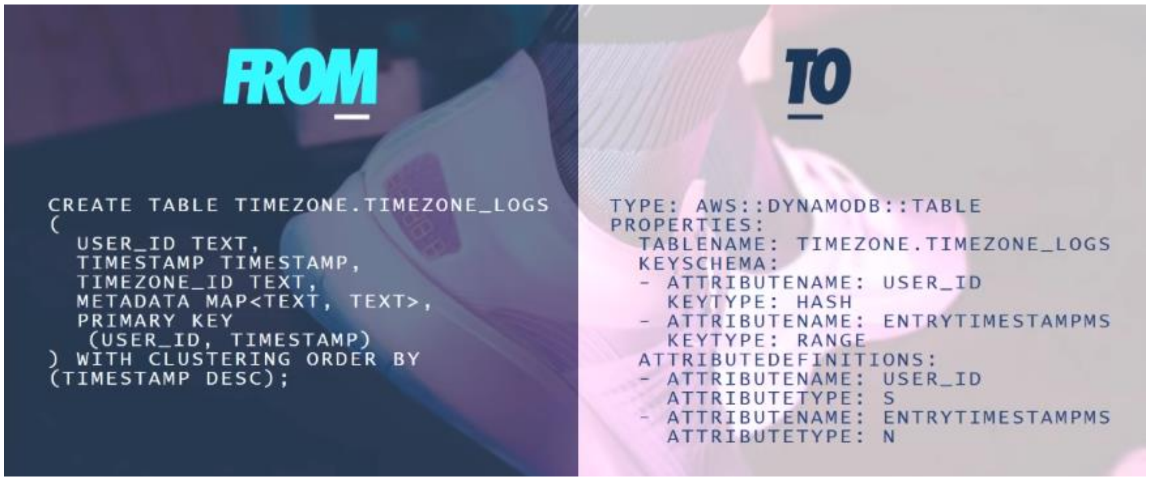
RUNNING CASSANDRA FOR TWO YEARS

1. **DEVOPS** — DEMANDING AND TIME-CONSUMING TASKS
2. **COST** — AMAZON EC2 COSTS INCREASED AS THE CLUSTER GREW
3. **UPGRADES** — DEMANDING AND TIME-CONSUMING TASKS

Nike Digital Engineering



RETIREMENT NOTIFICATION



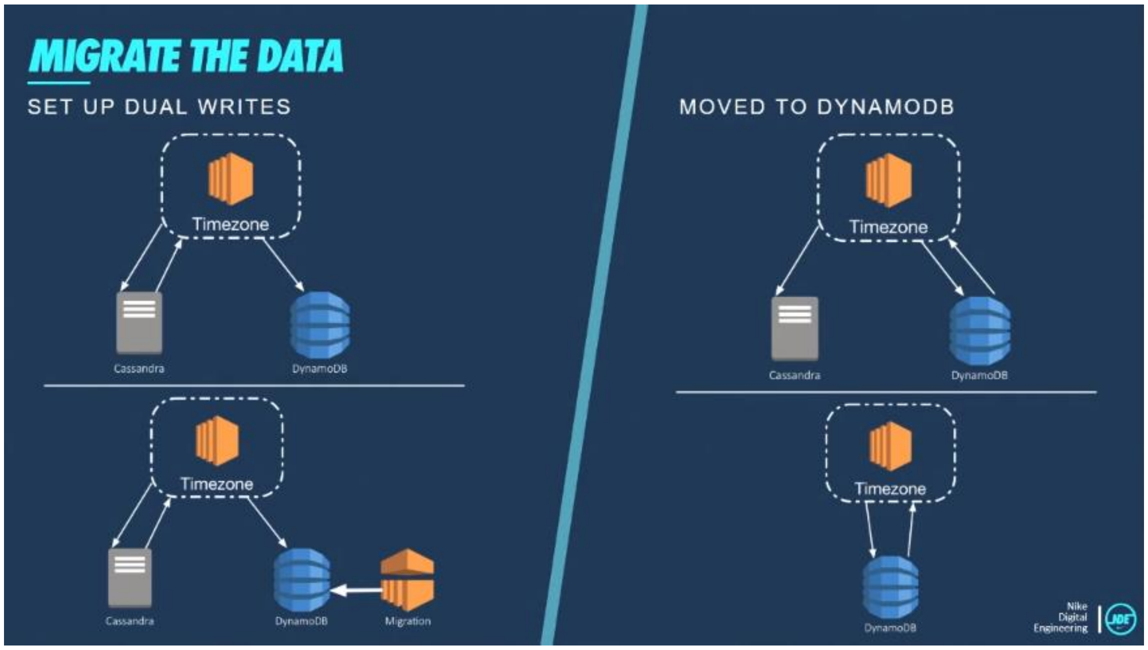
FROM

```
CREATE TABLE TIMEZONE.TIMEZONE_LOGS
(
  USER_ID TEXT,
  TIMESTAMP TIMESTAMP,
  TIMEZONE_ID TEXT,
  METADATA MAP<TEXT, TEXT>,
  PRIMARY KEY
   (USER_ID, TIMESTAMP)
) WITH CLUSTERING ORDER BY
(TIMESTAMP DESC);
```

TO

```
TYPE: AWS::DYNAMODB::TABLE
PROPERTIES:
  TABLENAME: TIMEZONE.TIMEZONE_LOGS
  KEYSCHEMA:
  - ATTRIBUTENAME: USER_ID
    KEYTYPE: HASH
  - ATTRIBUTENAME: ENTRYTIMESTAMPMS
    KEYTYPE: RANGE
  ATTRIBUTEDEFINITIONS:
  - ATTRIBUTENAME: USER_ID
    ATTRIBUTETYPE: S
  - ATTRIBUTENAME: ENTRYTIMESTAMPMS
    ATTRIBUTETYPE: N
```

The left contains a Cassandra schema and the right is the DynamoDB table schema where we just need to specify the key columns only.



MIGRATE THE DATA

SET UP DUAL WRITES

MOVED TO DYNAMODB

98% COST SAVINGS

Nike Digital Engineering

DYNAMODB IS THE NEW DEFAULT

SCALE

CONTINUOUS INNOVATION

FIND US ON MEDIUM

MEDIUM.COM/NIKEENGINEERING

Nike Digital Engineering