
AWS
re:Invent

DAT314

Running Apache Cassandra Workloads with Amazon Keyspaces

Arturo Hinojosa
Principal Product Manager
AWS

Mihir Desai
Sr. Solutions Architect
AWS

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda

Amazon Keyspaces (for Apache Cassandra) overview

Service architecture

Benefits

Recap of 2020 launches

Customer example: PricewaterhouseCoopers



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Apache Cassandra

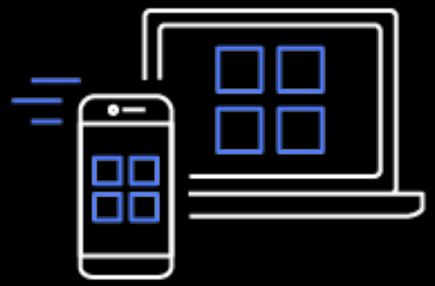
- Wide-column data store
- Designed for large-scale applications that need fast read and write performance



Apache Cassandra



User profiles



Device metadata



Event logging

Cassandra Query Language (CQL)

```
SELECT name, title FROM myapp.users
    WHERE user_id = ‘x’
```



“Thirty-six percent of practitioners currently using Cassandra for mission critical applications say that a lack of Cassandra-skilled team members may deter adoption.”

Cassandra Usage Report 2020

Apache Software Foundation

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software
- **Scaling clusters down to optimize resources is complex**

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software
- Scaling clusters down to optimize resources is complex
- **Version upgrades are difficult due to clunky rollback features**

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software
- Scaling clusters down to optimize resources is complex
- Version upgrades are difficult due to clunky rollback features
- **Restoring data is error prone**

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software
- Scaling clusters down to optimize resources is complex
- Version upgrades are difficult due to clunky rollback features
- Restoring data is error prone
- **No built-in encryption support to protect data**

Challenges of managing Cassandra

- Specialized expertise to deploy, configure, and manage software
- Scaling clusters down to optimize resources is complex
- Version upgrades are difficult due to clunky rollback features
- Restoring data is error prone
- No built-in encryption support to protect data
- **Patching, updating, and maintaining cluster infrastructure**

“Can you help us run our Cassandra workloads more easily?”



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Keyspaces (for Apache Cassandra)

A ***scalable***, ***highly available***, and ***fully managed*** Apache Cassandra–compatible database service



Production ready for mission-critical, operational workloads

Generally available in 20 AWS Regions

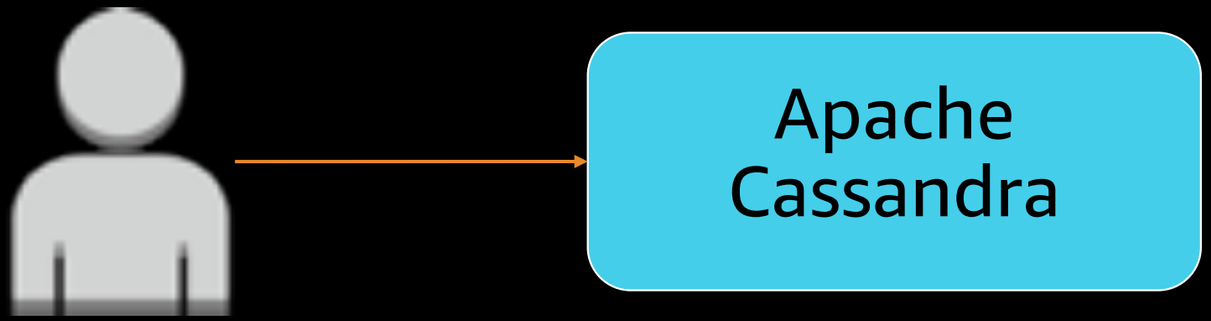
Built-in **data encryption**, **high availability**, and **continuous backup and restore**



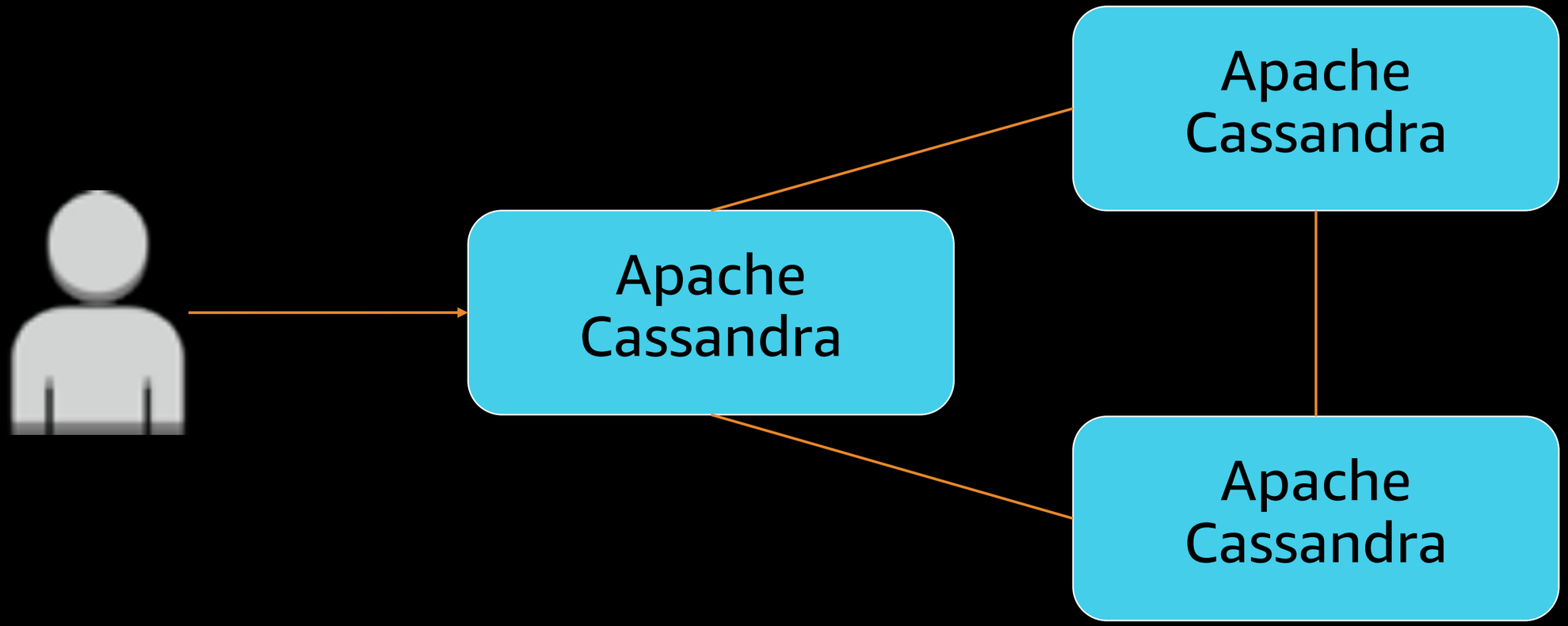
99.99% availability **SLA**

Integrated with AWS **logging and monitoring** services

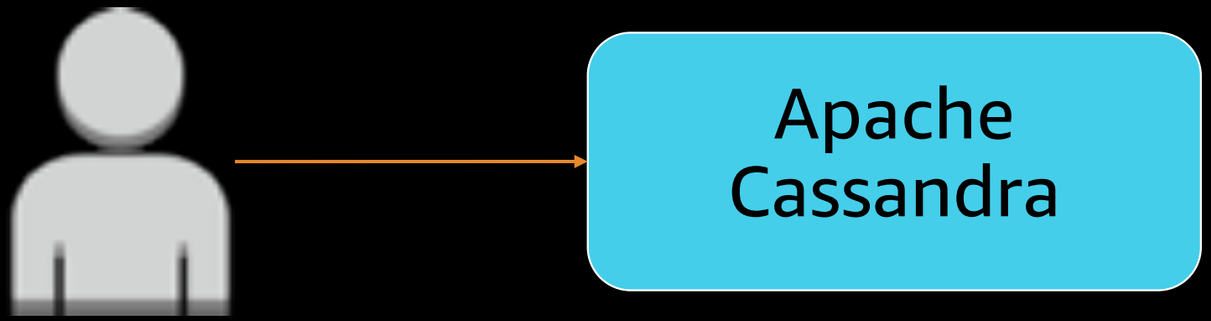
Typical Cassandra cluster



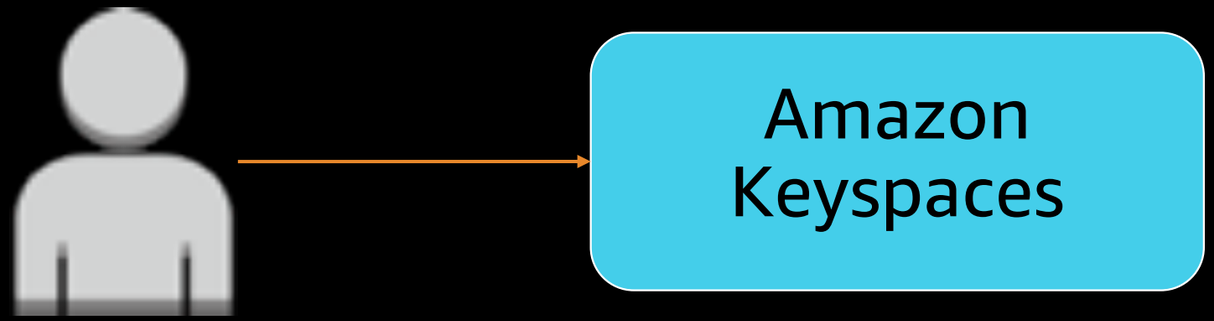
Typical Cassandra cluster



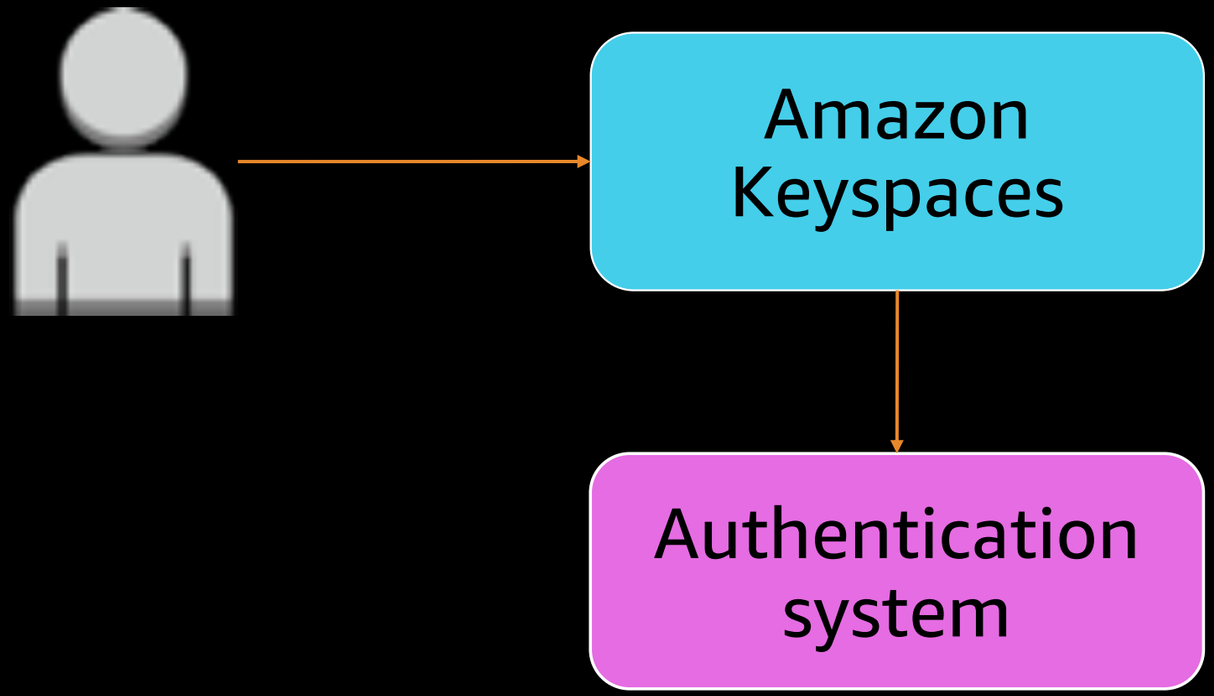
Amazon Keyspaces service architecture



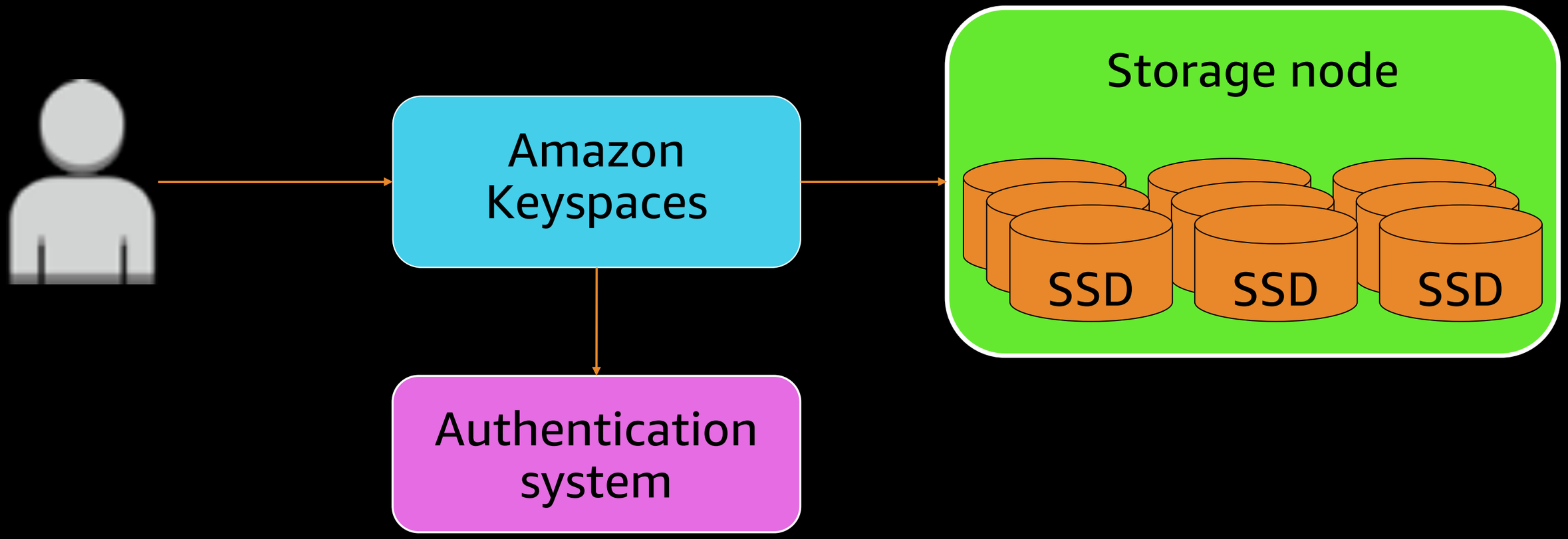
Amazon Keyspaces service architecture



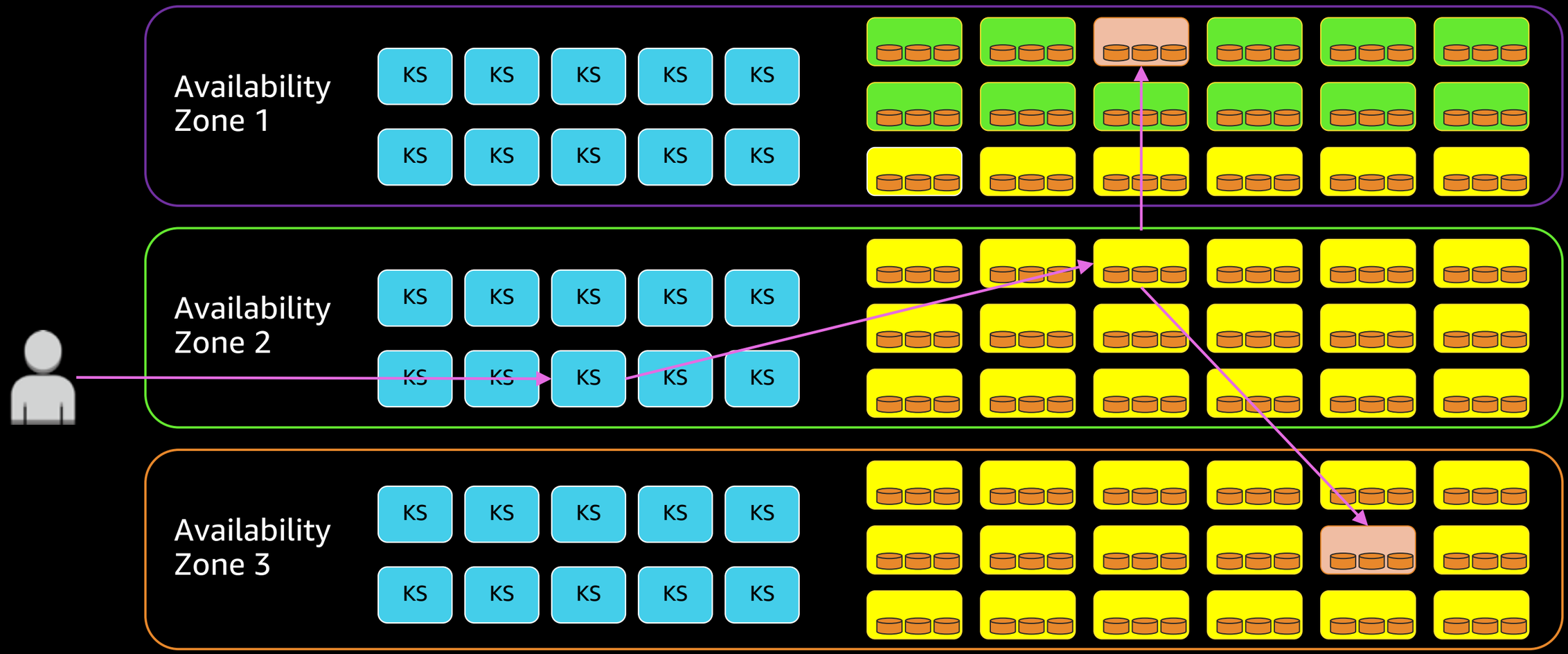
Amazon Keyspaces service architecture



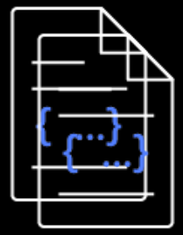
Amazon Keyspaces service architecture



Highly available, massively scalable architecture



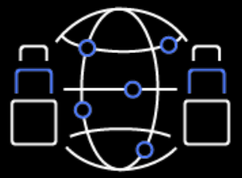
Benefits



Compatible with Apache Cassandra



No servers to manage



Performance at scale



Highly available and secure

Compatible with Apache Cassandra

- **Use your existing Cassandra Query Language (CQL) code**
  - Create and manage keyspaces and tables
  - Read and write data



Compatible with Apache Cassandra

- Use your existing Cassandra Query Language (CQL) code
  - Create and manage keyspaces and tables
  - Read and write data

- **Works with most open-source Cassandra drivers and tools**
  - Step-by-step instructions for Java, Node.js, Python, Go, .NET, and Perl Cassandra drivers
  - Read, write, and copy data to / from existing clusters by using cqlsh

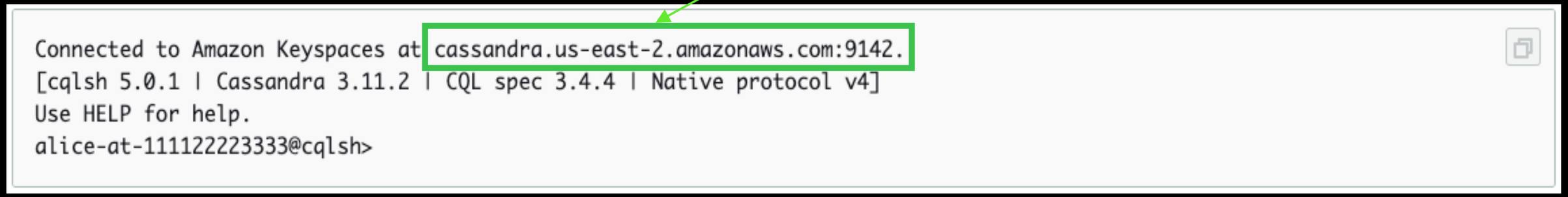


Compatible with Apache Cassandra

- Use your existing Cassandra Query Language (CQL) code
  - Create and manage keyspaces and tables
  - Read and write data
- Works with most open-source Cassandra drivers and tools
  - Step-by-step instructions for Java, Node.js, Python, Go, .NET, and Perl Cassandra drivers
  - Read, write, and copy data to / from existing clusters by using cqlsh
- **Amazon Keyspaces appears to be 9-node, 3.11.2 cluster**
  - Drivers and clients must be compatible with 3.11.2
  - Amazon Keyspaces offers same backwards compatibility as 3.11.2

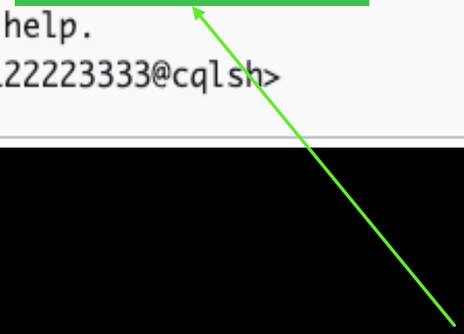


Compatible with Apache Cassandra



Compatible with Apache Cassandra

```
Connected to Amazon Keyspaces at cassandra.us-east-2.amazonaws.com:9142.
[cqlsh 5.0.1 | Cassandra 3.11.2 | CQL spec 3.4.4 | Native protocol v4]
Use HELP for help.
alice-at-111122223333@cqlsh>
```



Cassandra version

Differences with Apache Cassandra

- Low-level control plane APIs for managing clusters
- Deprecated Apache Thrift API
- Experimental features
- Features with known scalability or stability issues

No servers to manage

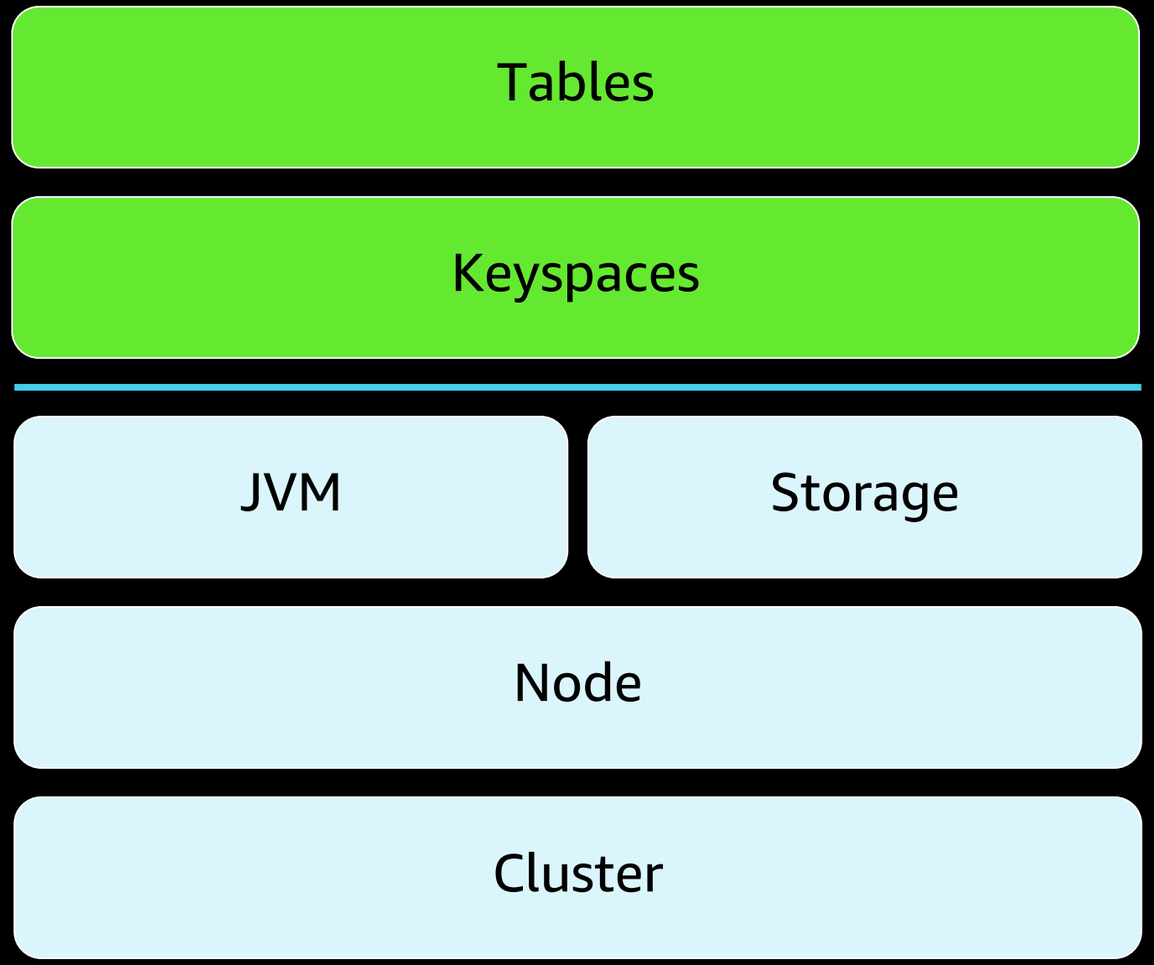
- **Fully managed**
  - No servers to provision, patch, or manage
  - No software to install, maintain, or operate





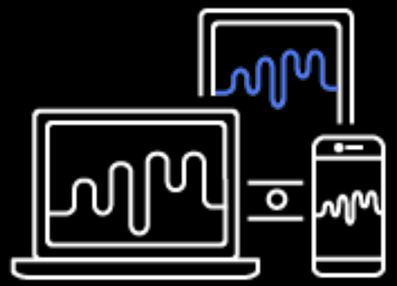
No servers to manage

- **Serverless**
  - No cluster or nodes
  - No compaction strategies or tombstones
  - No Java virtual machine (JVM) or garbage collection
  - Keyspaces and tables are fundamental building blocks



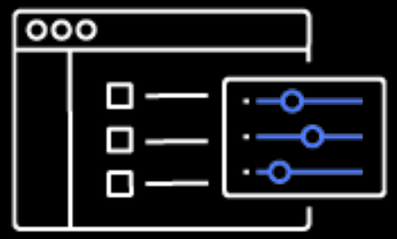
Flexible read / write capacity options

On-demand capacity



Pay based on actual reads and writes

Provisioned capacity with auto scaling



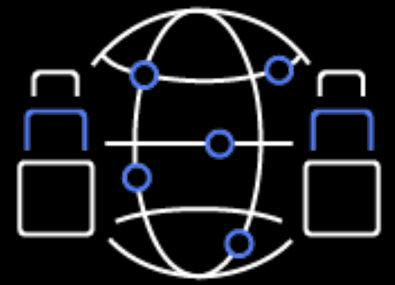
Set throughput capacity settings in advance to optimize price

Choosing a capacity mode

On demand	Provisioned
• New or very spiky workloads • Capacity fully managed by Amazon Keyspaces • Capacity can scale up to previous peak load almost instantly • Peak capacity can double every 30 minutes	• Workloads with predictable traffic • Define min, max, and target utilization rate for throughput • Amazon Keyspaces adjusts capacity up and down automatically to maintain target utilization • Short traffic bursts accommodated by using burst capacity

Performance at scale

- Built for massive-scale, performance-sensitive workloads
- Provides consistent, single-digit millisecond read and write performance at any scale
- Tables scale up and down automatically based on actual application traffic
- Virtually no limit on the size of tables and partitions



Read / write consistency options

- All writes performed with LOCAL_QUORUM consistency and a replication factor of 3

- Support for ONE, LOCAL_ONE, and LOCAL_QUORUM consistency for reads
  - Use ONE and LOCAL_ONE consistency, when possible, for higher availability, faster performance, and lower price
  - ONE and LOCAL_ONE consistency reads are billed in half increments

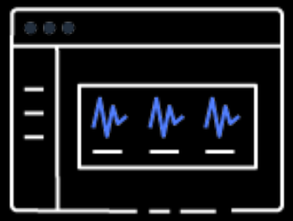
Highly available



Data is replicated automatically across multiple AWS Availability Zones using a replication factor of 3



99.99% availability SLA within an AWS Region



Monitor performance using Amazon CloudWatch

Monitoring performance and availability

Keyspaces CloudWatch metrics

- Provisioned capacity
- Consumed capacity
- Capacity utilization rate
- Request latency
- User errors
- Unsuccessful lightweight transactions
- System errors

Amazon Keyspaces Metrics and Dimensions

The metrics and dimensions that Amazon Keyspaces sends to Amazon CloudWatch are listed here.

Amazon Keyspaces Metrics

Amazon CloudWatch aggregates Amazon Keyspaces metrics at one-minute intervals.

Not all statistics, such as `Average` or `Sum`, are applicable for every metric. However, all of these values are available through the Amazon Keyspaces console, or by using the CloudWatch console, AWS CLI, or AWS SDKs for all metrics. In the following table, each metric has a list of valid statistics that are applicable to that metric.

Metric	Description
`AccountProvisionedReadCapacityUtilization`	The percentage of provisioned read capacity units utilized by an account. Units: `Percent` Valid Statistics: • `Maximum` – The maximum percentage of provisioned read capacity units utilized by the account. • `Minimum` – The minimum percentage of provisioned read capacity units utilized by the account. • `Average` – The average percentage of provisioned read capacity units utilized by the account. The metric is published for five-minute intervals. Therefore, if you rapidly adjust the provisioned read capacity units, this statistic might not reflect the true average.

Built-in security



- Data is encrypted at rest using encryption keys stored in AWS Key Management Service (AWS KMS)
- Secure connectivity from clients using TLS
- Manage access to individual keyspaces and tables by using AWS Identity and Access Management (IAM)
- Connect Keyspaces to resource in Amazon Virtual Private Cloud (Amazon VPC) through AWS PrivateLink
- Administrative DDL actions are logged in CloudTrail

Point-in-time recovery (PITR)

Helps protect against accidental deletions and updates

Table data is backed up continuously

Table data can be restored to any second in time in the previous 35 days

No impact on table performance or availability



PITR

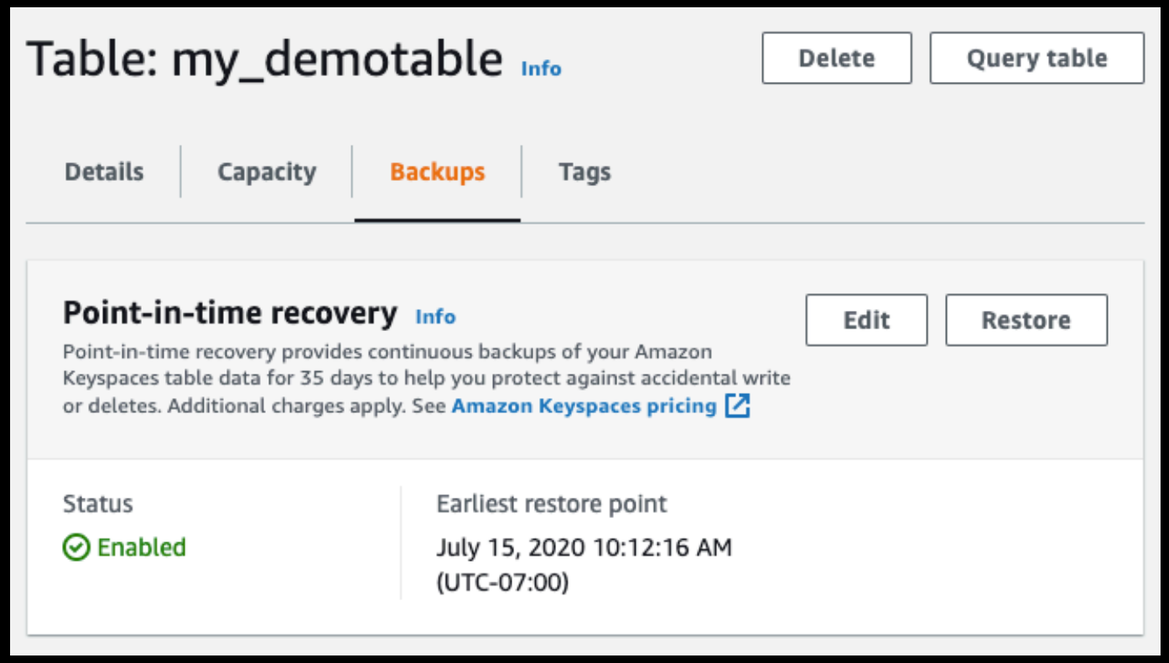
PITR can be enabled selectively on the tables you choose

Enabling PITR on production tables is highly recommended

Pricing is based on the size of the table

Restore tables programmatically by using CQL or from the AWS console

```
RESTORE TABLE mykeyspace.mytable_restored
FROM TABLE mykeyspace.mytable
WITH restore_timestamp = '2020-06-30T19:19:21.175Z';
```



Recent launches



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Pace of innovation: 2020 launches

18
Launches YTD

20
AWS Regions

3+
Open-source contributions

Ordering clauses
Provisioned capacity with auto scaling
Altering tables
Service Quotas support
Authentication plugins for Java, Go, Python, and Node.js drivers
Availability in 20 AWS Regions
AWS CloudFormation support
Tagging and tag-based access management
Counters
AWS PrivateLink support
Point-in-time recovery (PITR)
NoSQL Workbench support

User example: PricewaterhouseCoopers



© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

About PricewaterhouseCoopers



- PricewaterhouseCoopers (PwC) is a multinational professional services network of firms operating as partnerships under the PwC brand globally
- Provides audit and assurance, advisory, and tax services to companies
- Second-largest professional services firm in the world and one of the “Big Four” consulting companies

Check-In: A PwC product

Manage risk and support your return to workplace plans holistically
Take preventive action to safeguard your people, mitigate infection risks, manage capacity, and give people workplace alternatives

Prevent overcrowding and keep potentially ill individuals away
A digital health status connect helps you prevent too many people and potentially ill people from coming to a workplace

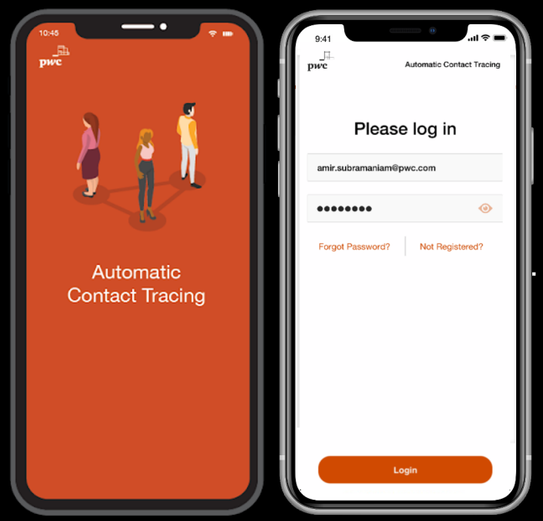
Help your employees feel more confident in the workplace
An automatic contact tracing ecosystem with 4 device options – including an app geofenced to operate only at your workplace, for privacy purposes – allows you to act swiftly and help reduce the impact on others from an infected employee

Forecast, identify trends, and connect to other systems with accuracy
Use Check-In to tap into your own tools, policies, and systems for case management, alerts, and workforce planning as you work to help build confidence and protect your people

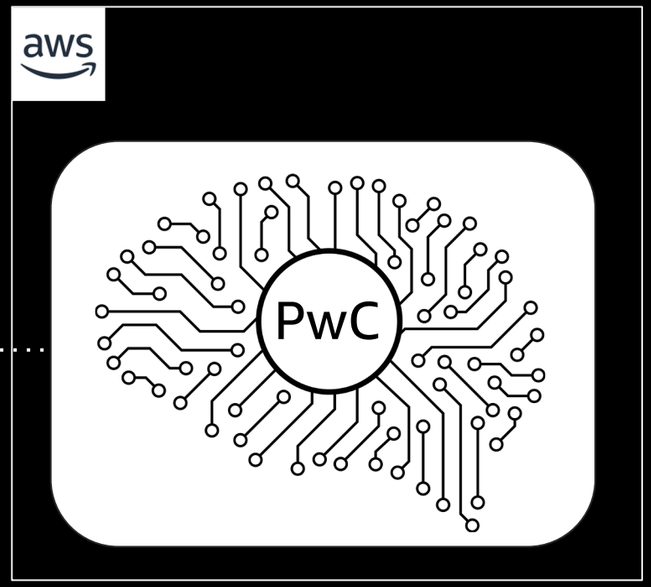
Identify exposure risk so you can act fast and target communications
Close the information gap with data-driven dashboards and recommended actions to manage and mitigate illness-related risks to your workforce and business

Source: PwC, https://www.pwc.com/us/en/products/check-in.html

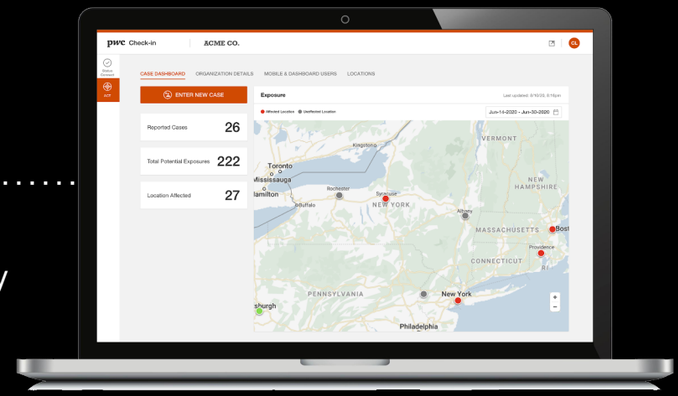
How Check-In Automatic Contact Tracing works



Pseudonymized contact tracing



Links to employees only to determine potential exposure



Mobile application

Users either install a mobile app on their phones or carry a hardware device such as a beacon

Backend tech

Associations between signals observed by anonymized phones enables proximity tracing between employees

Dashboard

Authorized users input name of impacted employee; ACT instantly provides a list of potentially at-risk colleagues

Why Amazon Keyspaces?



The ACT use case requires the rapid, flexible scalability provided by Keyspaces for scaling both up and down, reducing operating costs



Easy CloudWatch integration for building dashboards and alarms



Quick time to market:

- Amazon Keyspaces does not require specific Cassandra operation knowledge
- No ramp-up time building a Cassandra cluster
- Easier alternative to self-managing a Cassandra fleet



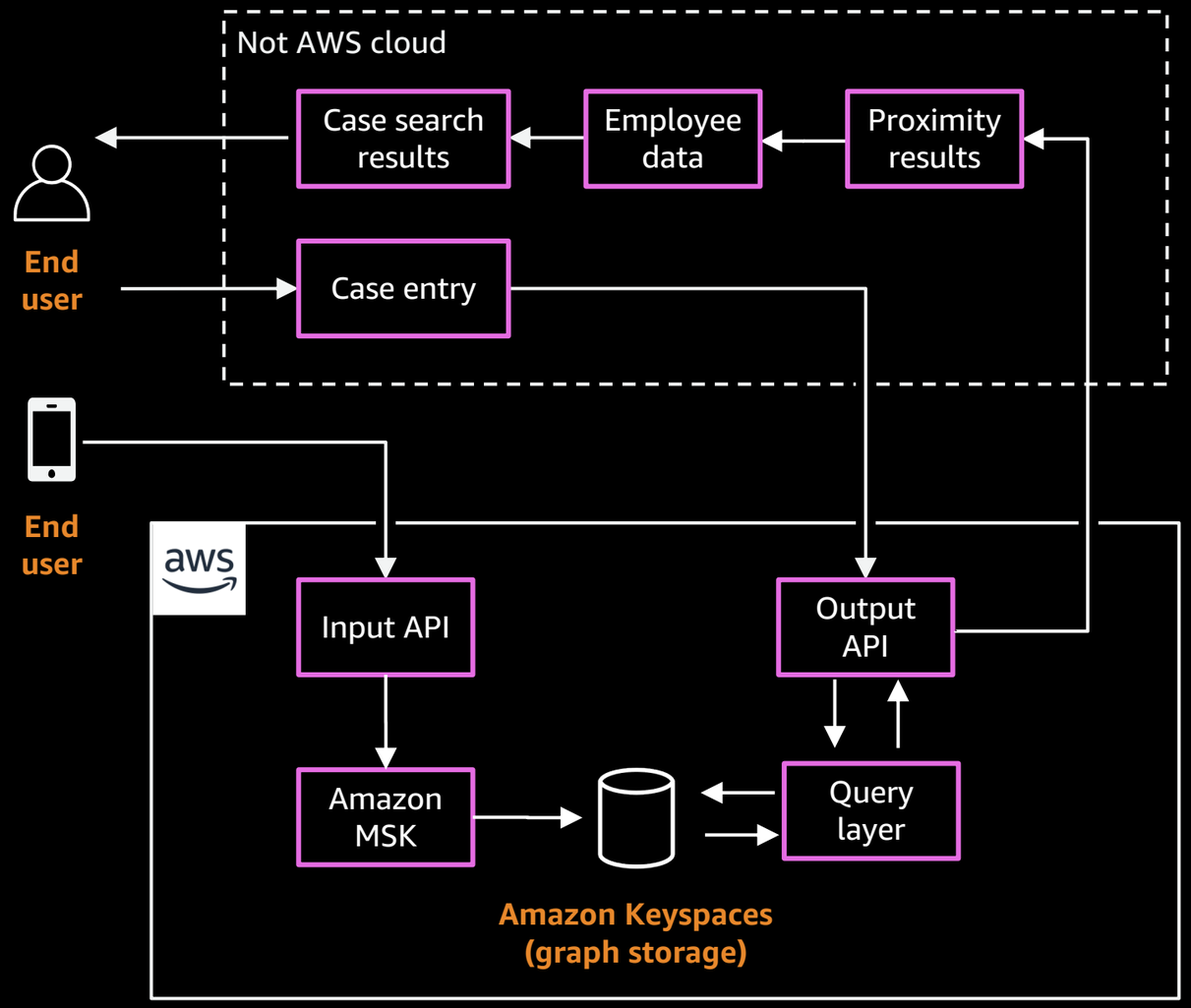
Pay-as-you go, serverless infrastructure offered PwC greater flexibility and enabled it to be more agile



Amazon Keyspaces was selected for its ability to process a very high number of writes (up to 200 billion per day)

Proximity architecture

HIGH-LEVEL DIAGRAM



Keyspaces stores the time-sharded graph-data model that ACT relies on for proximity data

Upon ingest via an Input API, data is directed via a Kafka queue into Amazon Keyspaces

Data is stored in Amazon Keyspaces in a time-sharded graph model

A distributed query layer powers the solution's API queries, delivering near-real-time results from high-volume streaming data

Data stored across 3 Keyspaces with an identical data schema, for retention purposes; this will be replaced with the new TTL feature when it becomes available

Lessons learned

Adapting to compatibility differences

- **Truncating tables** – Modified cron-job rollover to delete and re-create tables with all settings; additional steps required for reuse of timed shards (Keyspaces)
- **Batchsize limit (30)** – Requires application-side tuning; no performance impact
- **Capacity errors** – When using the driver, insufficient read / write capacity errors are reported as immediate timeouts and require application-side exception handling adjustments
- **Retry mechanisms** (e.g., exponential back off) – Creates a circle, requiring a custom algorithm with fixed wait time for upper retry limit

Feature requests

- **Throttle events are unclassified** (by reason, i.e., capacity or hotkey) – Requires an approximation via arithmetic; more specific error messages would help
- **The lack of an AWS user accessible error log** – Makes it difficult to determine the nature of system and/or user errors

Thank you!



AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Please complete the session survey

AWS re:Invent

© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.