BIG DATA CASE STUDY NIGHT #3

STREAM AND ANALYZE TIME SERIES IN NEAR-REAL-TIME WITH AWS

John Mousa

Sr. Solutions Architect
Amazon Web Services

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

John Mousa

Engineer at heart, enabling enterprise customers in DACH and world wide bridge the worlds of microservices and analytics as a Sr. Solutions Architect at Amazon Web Services.

/in/johnmousa

/johnmousa

@JohnMousa_

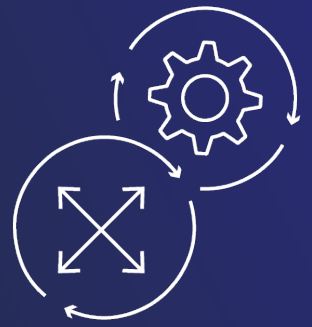
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Agenda for Today

Our use case for Power and Utilities

Overview on Architecture

Solutions

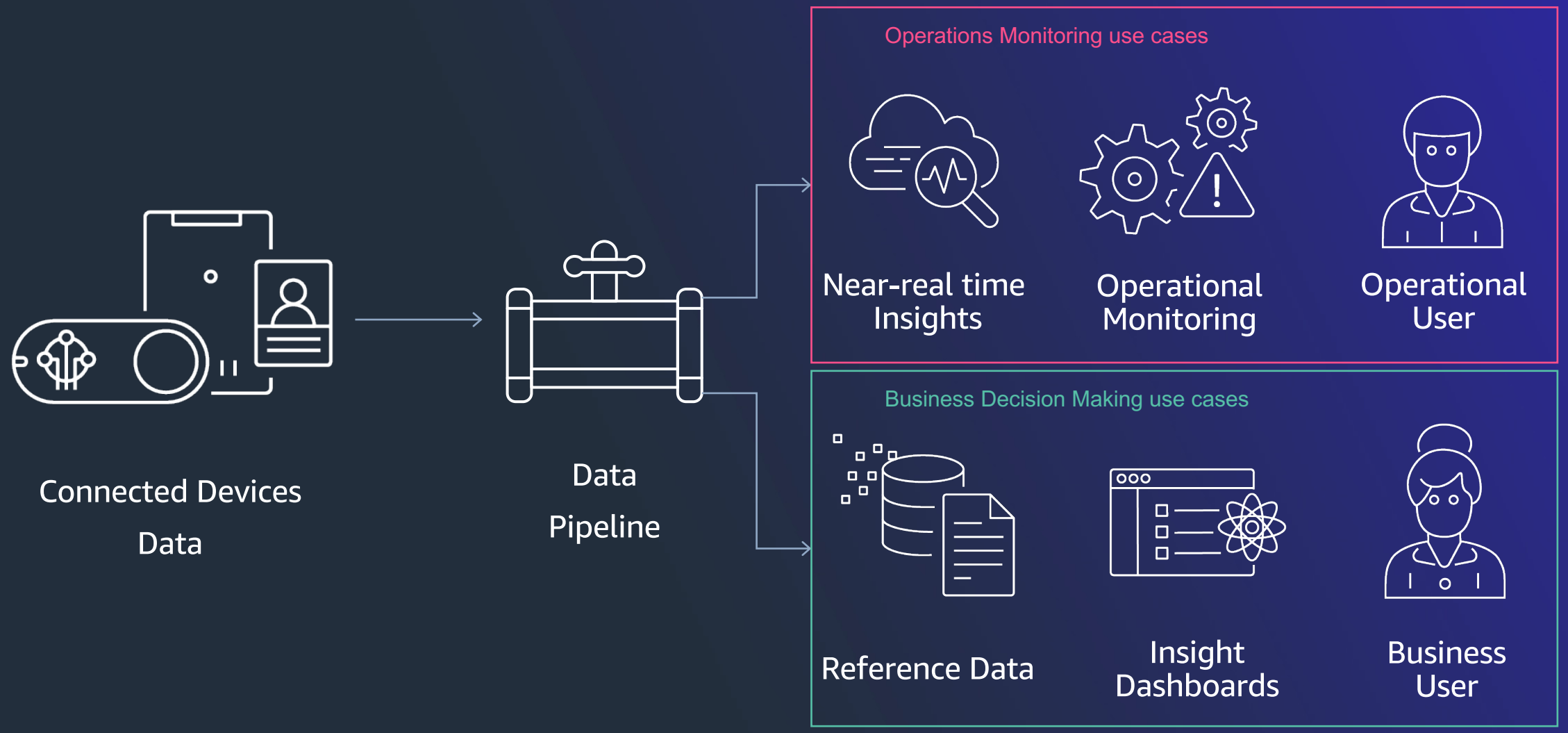


Operational View

Strategic View

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Collecting Telemetry Data from Connected Devices



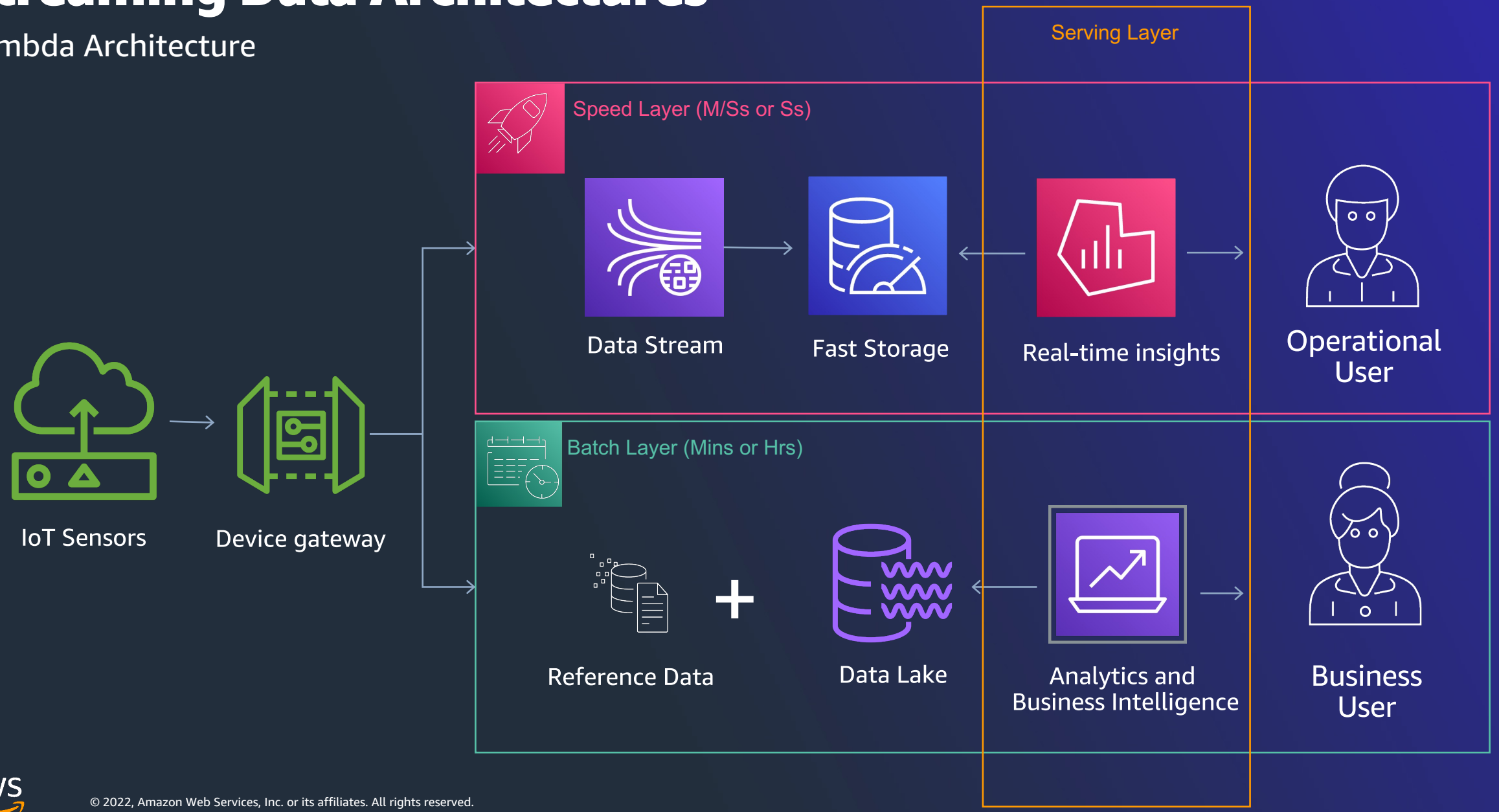
© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Architectures

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Data Architectures

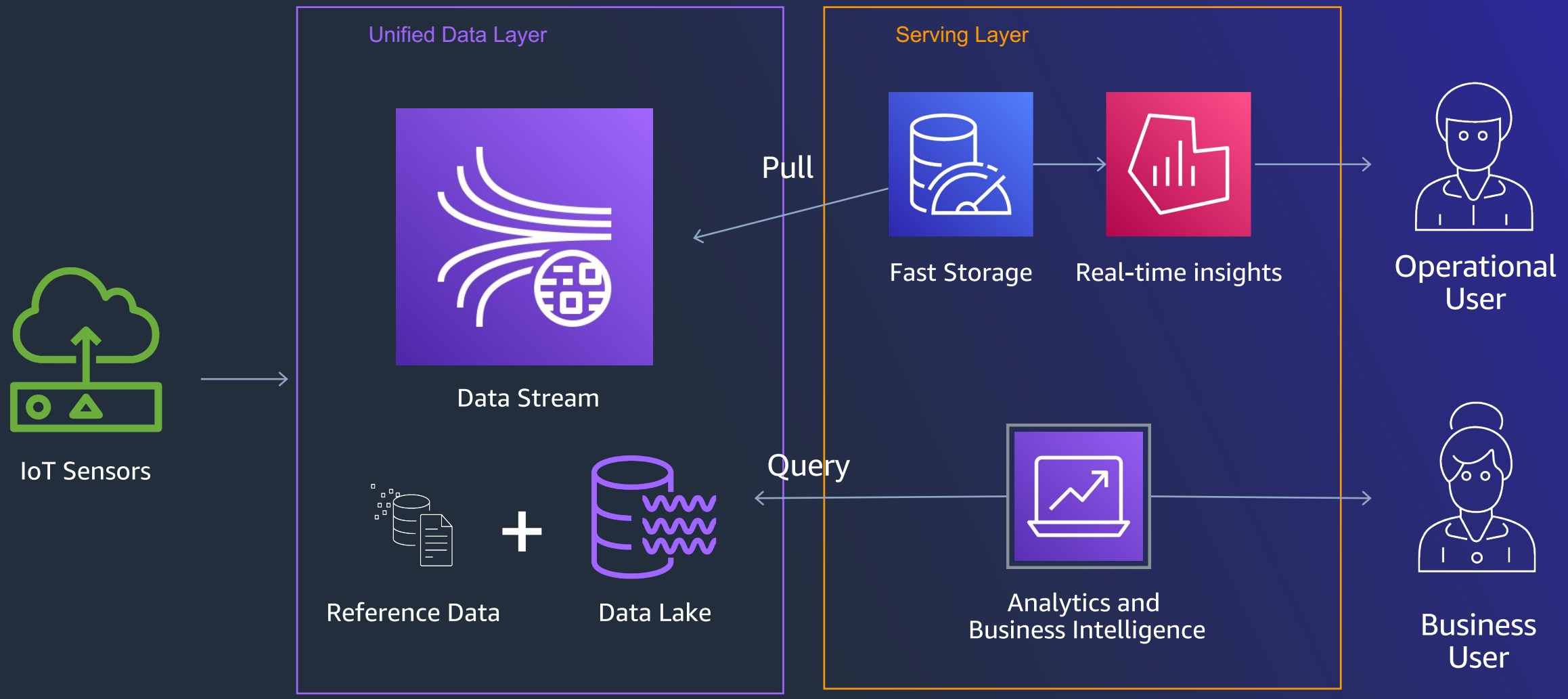
Lambda Architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Data Architectures

Kappa Architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Data Architectures

Lambda VS Kappa

λ

- Batch layer is built for big data
- Result in overhead due to code and resource duplication

K

- No redundancies
- Batch workloads needs to be in a sink

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Solutions

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

But Why Did We Consider Such Solutions

Streaming is Expensive

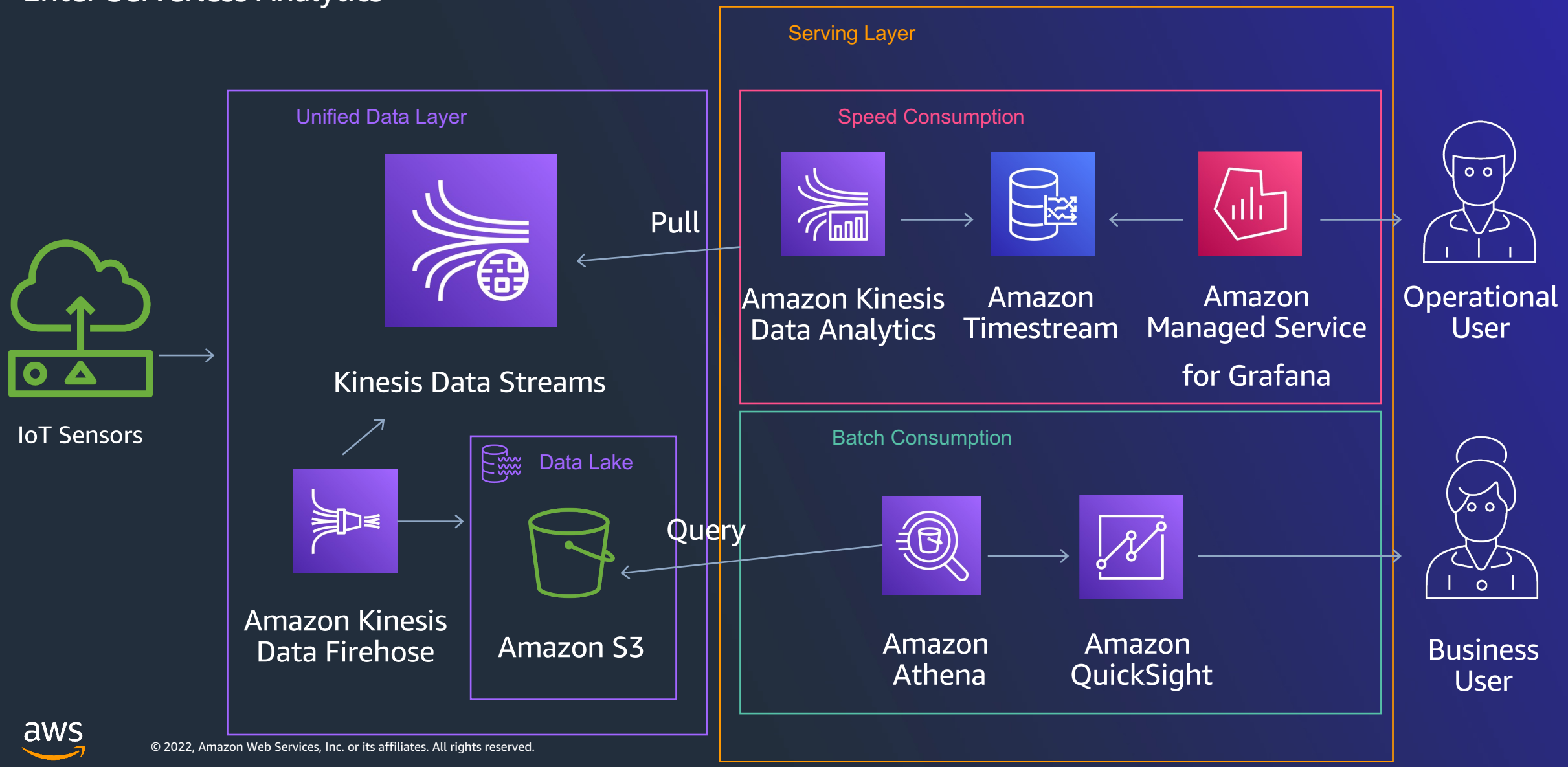
Streaming is Lossy

Hard to Operate,
Scale and Maintain

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Current Data Architecture

Enter Serverless Analytics

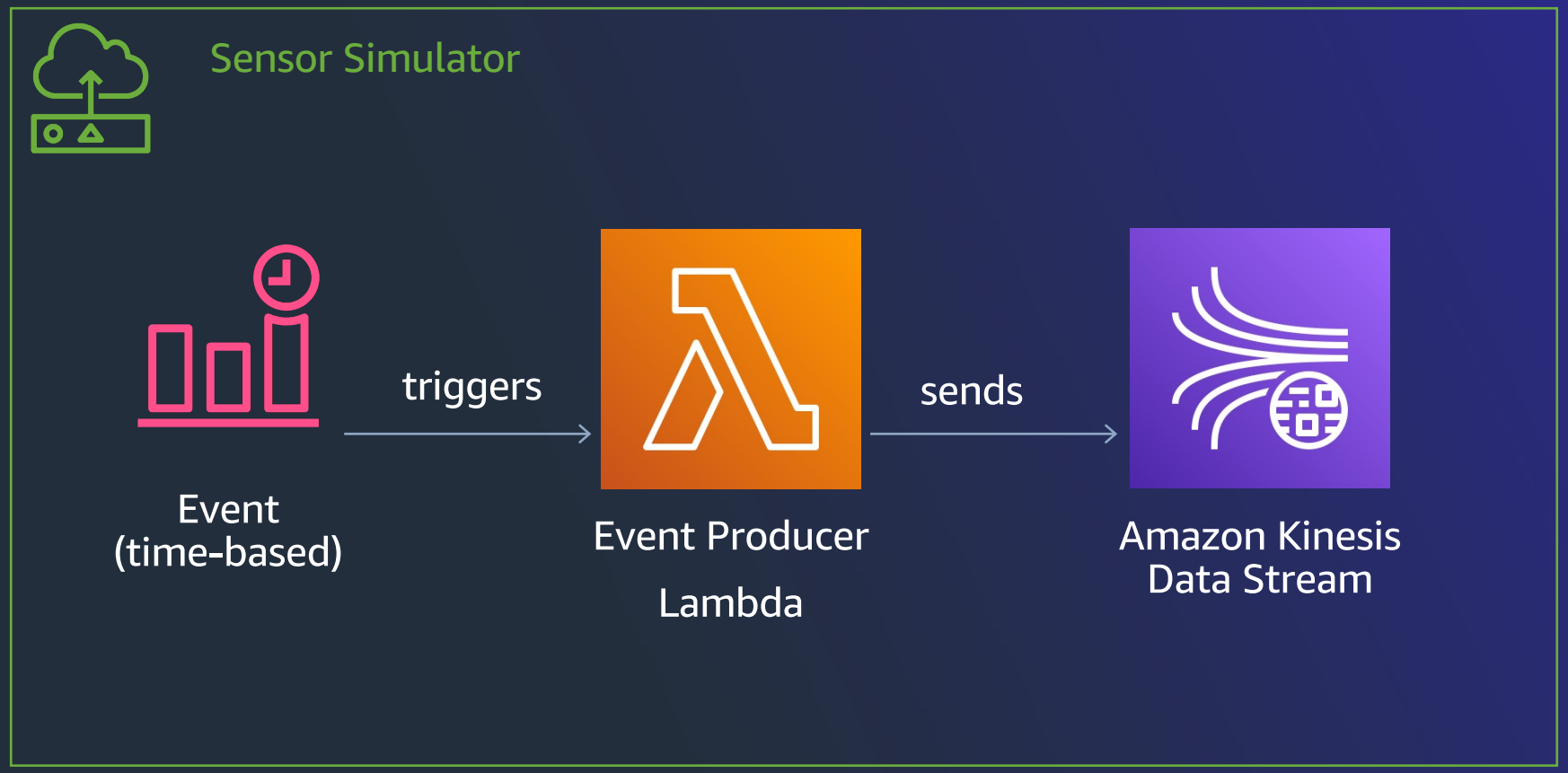


© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Ingestion

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Ingestion from Devices



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Ingestion from Devices

CDK Infrastructure

```
sample_device_producer = aws_lambda_python.PythonFunction(self, 'SampleDeviceProducer',
                                                          entry='stacks/sample_kinesis_stream_producer/producer_lambda',
                                                          index='app.py',
                                                          runtime=aws_lambda.Runtime.PYTHON_3_8,
                                                          timeout=core.Duration.seconds(30))

stream.grant_write(sample_device_producer)

lambda_input = {"Stream": stream.stream_name}
Rule(self, 'ProducerTriggerEventRule',
     enabled=True,
     schedule=Schedule.rate(Duration.minutes(1)),
     targets=[aws_events_targets.LambdaFunction(handler=sample_device_producer,
                                                event=RuleTargetInput.from_object(lambda_input))])
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Ingestion from Devices

Lambda Writing to Kinesis Data Streams

```
def write_records(stream_name, records):
    kinesis_records = []
    for record in records:
        kinesis_records.append({
            'Data': json.dumps(record),
            'PartitionKey': record["DeviceID"]
        })

    result = None
    try:
        result = kinesis.put_records(
            StreamName=stream_name,
            Records=kinesis_records, )

        status = result['ResponseMetadata']['HTTPStatusCode']
        print("Processed %d records. WriteRecords Status: %s" %
              (len(records), status))
    except Exception as err:
        print("Error:", err)
        if result is not None:
            print("Result:{}".format(result))
```

```
def handler(event, context):
    sent = 0
    for x in range(iterations):
        records = []
        for device_id in device_ids:
            records.append(prepare_record(measures, device_id,
                                          (iterations - x) / iterations * 60))

        print("records {}".format(len(records)))
        write_records(event["Stream"], records)
        sent = sent + len(records)
    return {"records": sent}
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Streaming

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

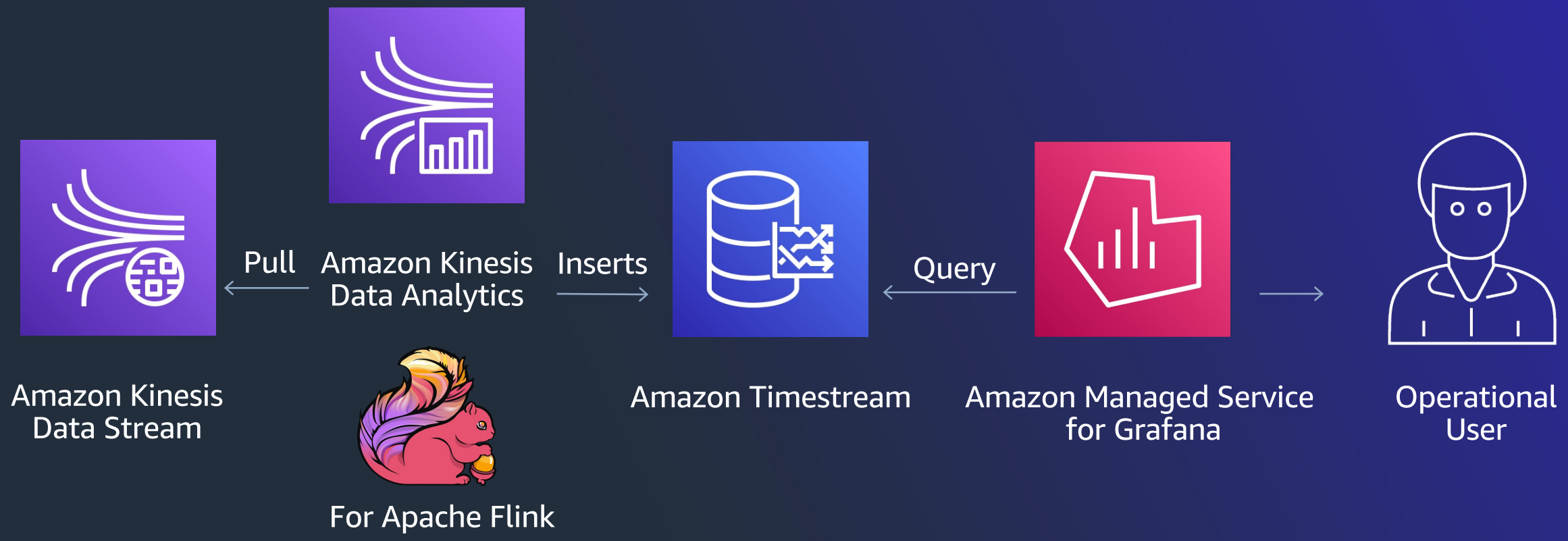
Kinesis Data Streams: Granular Provisioning and Scaling

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Operational View

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

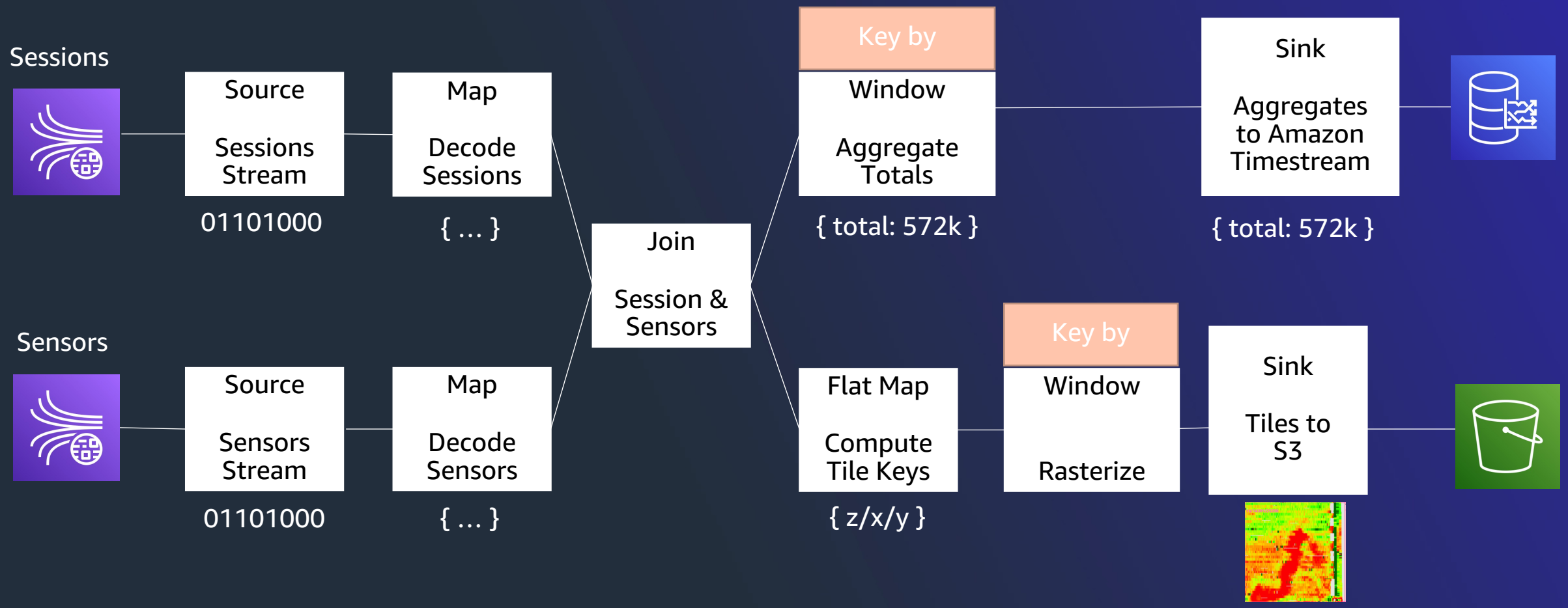
Near-Real Time Operational Insights Architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Analytics

Apache Flink - Example



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Streaming Analytics

Apache Flink – Declarative Streams

```
createKinesisSource(env, parameter)
    .map(JsonToTimestreamPayloadFn()).name("MaptoTimestreamPayload")
    .process(OffsetFutureTimestreamPoints()).name("UpdateFutureOffsetedTimestreamPoints")
    .addSink(TimestreamSink(region, databaseName, tableName, batchSize))
    .name("TimestreamSink<$databaseName, $tableName>")
```

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Timestream Architecture

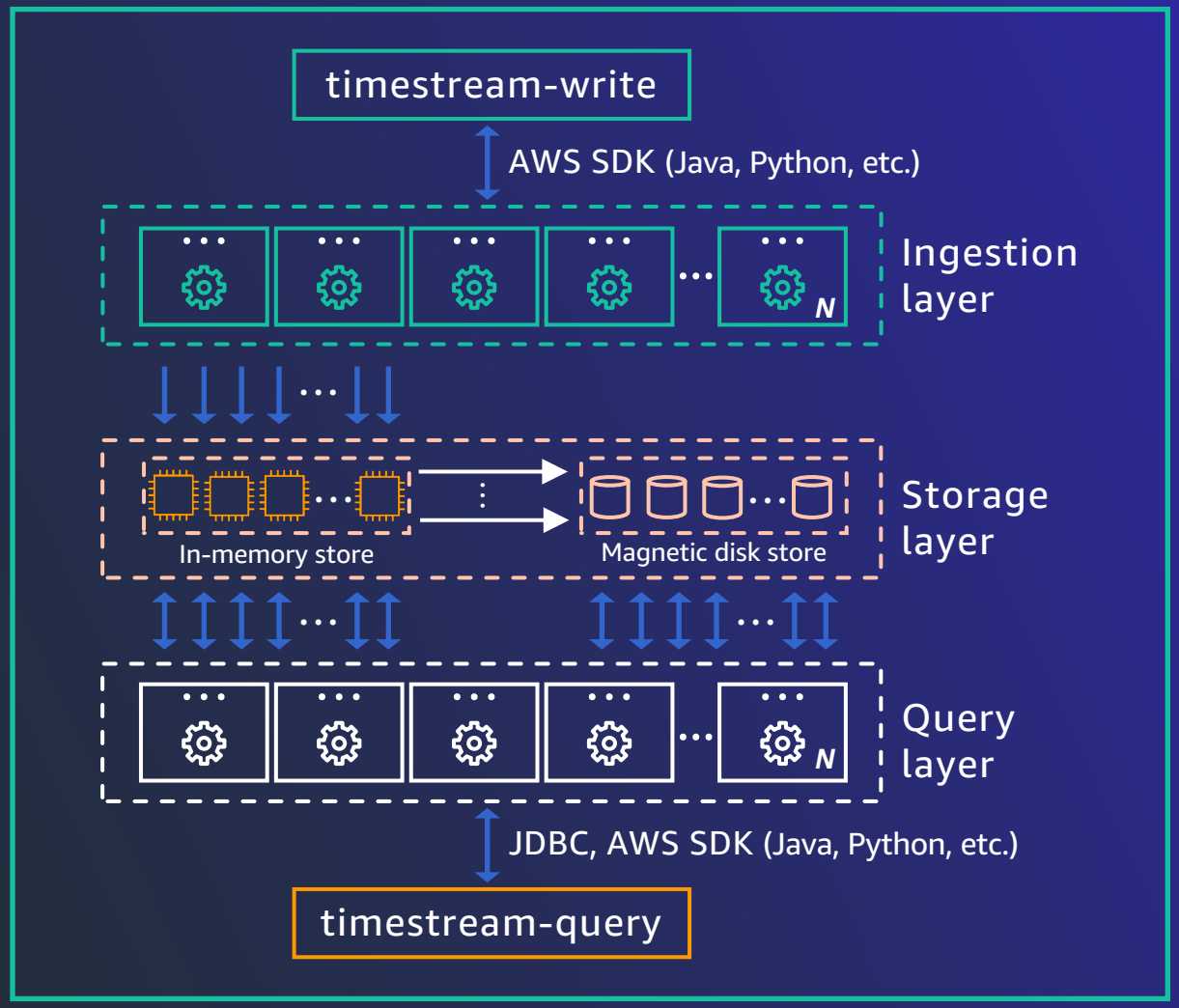
Purpose-built for time series workloads

Decoupled architecture

- Highly available
- ***Independently*** scalable ingestion, storage, and query processing

High throughput auto scaling ingestion

- ***Durable***: data is ***replicated*** across multiple Availability Zones
- Automatic data deduplication handling
- No need to provision or configure resources



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Amazon Timestream Architecture

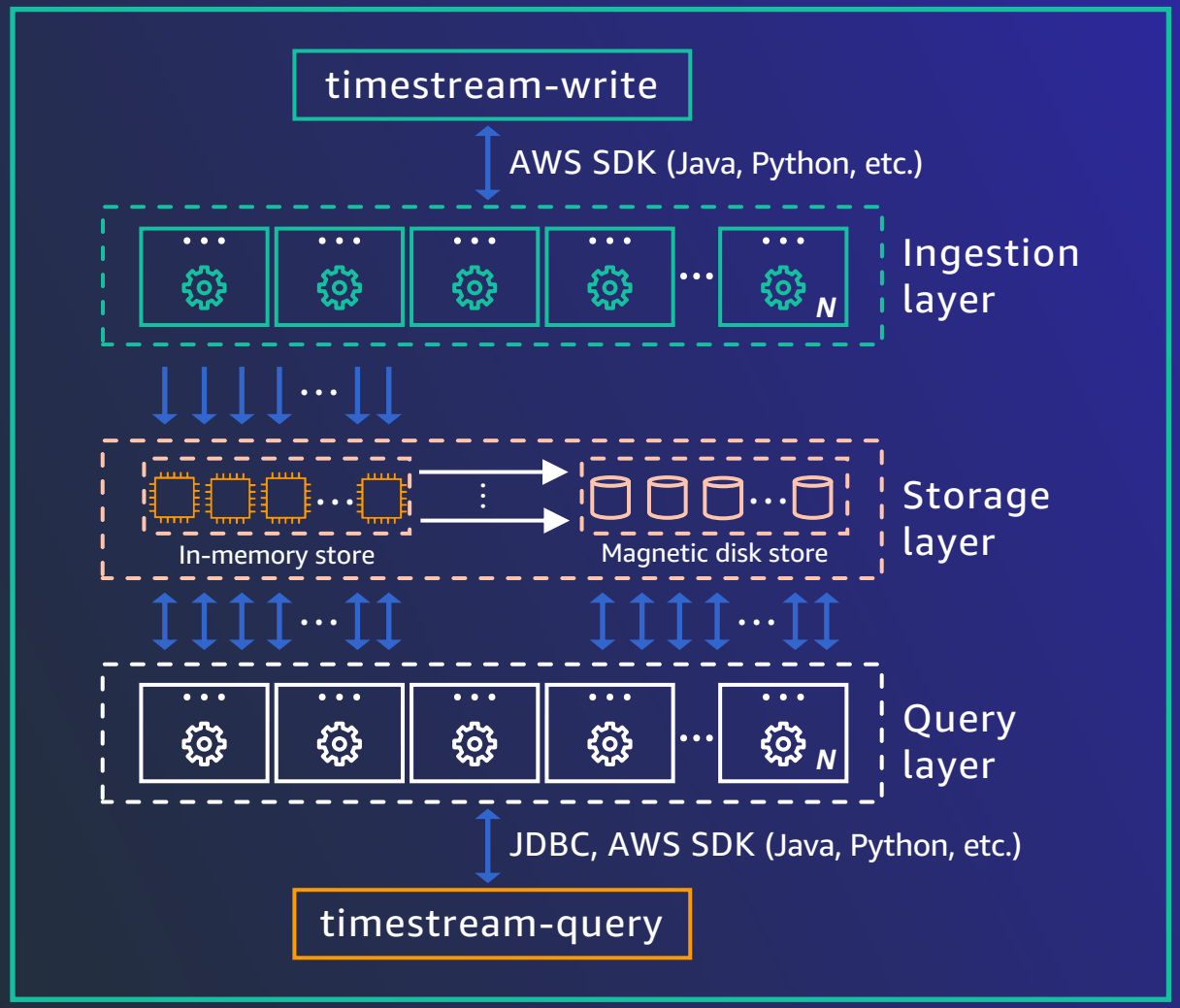
Serverless, cloud-native AWS built technology

Multiple tiers of storage

- Scalable to petabytes and beyond
- ***Memory store***: fast point-in-time queries, high-throughput durable ingestion
- ***Magnetic store***: high-performance analytical queries and low-cost long-term storage

Scalable SQL query engine

- ***Adaptive query engine*** is capable of querying data across multiple data tiers
- ***No indexes*** to configure and ***no provisioning*** required



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Operational Views

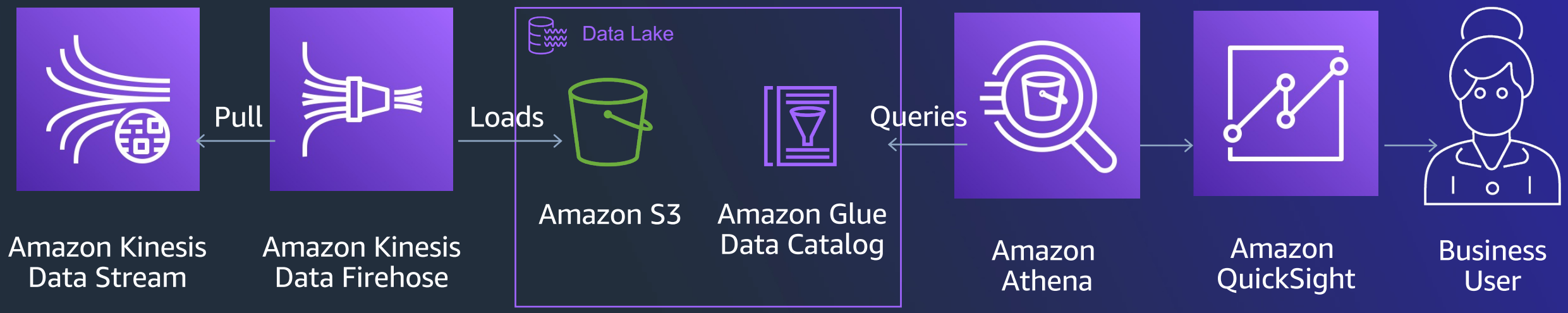
Amazon Managed Service for Grafana

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Strategic Big Data Insights

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

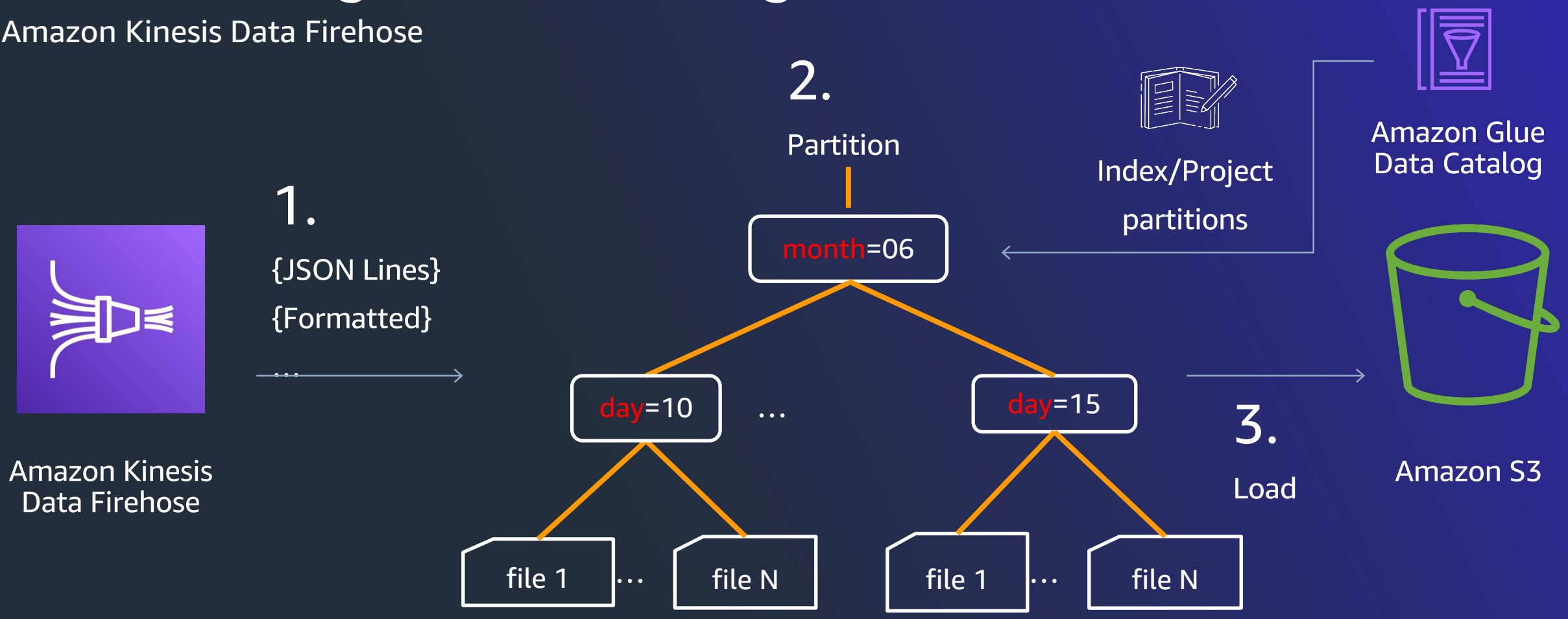
Strategic Big Data Insights Architecture



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Encoding and Partitioning

Amazon Kinesis Data Firehose



© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Data Encoding and Partitioning

Amazon Kinesis Data Firehose

```
RawEventsFirehose:
    DependsOn: DeliveryPolicy
    Type: AWS::KinesisFirehose::DeliveryStream
    Properties:
      DeliveryStreamName: RawRecordsDeliveryStream
      DeliveryStreamType: KinesisStreamAsSource
      KinesisStreamSourceConfiguration:
        KinesisStreamARN: !GetAtt RawRecordsStream.Arn
        RoleARN: !GetAtt DeliveryRole.Arn

      ExtendedS3DestinationConfiguration:
        BucketARN: !Sub "arn:aws:s3:::${RawDataBucket}"
        BufferingHints:
          IntervalInSeconds: '60'
          SizeInMBs: '100'
        CompressionFormat: UNCOMPRESSED
        ErrorOutputPrefix: "errors/json/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!
{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type}"
        Prefix: "readings/json/year=!{partitionKeyFromQuery:year}/month=!{partitionKeyFromQuery:month}
/day=!{partitionKeyFromQuery:day}/hour=!{partitionKeyFromQuery:hour}/"
        DynamicPartitioningConfiguration:
          Enabled: true
          RetryOptions:
            DurationInSeconds: 300
        RoleARN: !GetAtt DeliveryRole.Arn
        ProcessingConfiguration:
          Enabled: true
          Processors:
            - Type: MetadataExtraction
              Parameters:
                - ParameterName: MetadataExtractionQuery
                  #.reading_date_time with type string and format "yyyy-MM-dd HH:mm:ss.SSS"
                  ParameterValue:
"{year:.reading_date_time[:4],month:.reading_date_time[5:7],day:.reading_date_time[8:10],hour:.reading_
date_time[11:13]}"
                - ParameterName: JsonParsingEngine
                  ParameterValue: JQ-1.6
            - Type: AppendDelimiterToRecord
              Parameters:
                - ParameterName: Delimiter
                  ParameterValue: "\\n"
```

Move data from Stream to Raw S3 Bucket

Late arriving data compensation

JSON lines

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Discover Dynamic Partitions

Glue Data Catalog Partition Projection

```
        Prefix: "readings/json/year=!{partitionKeyFromQuery:year}/month=!{partitionKeyFromQuery:month}
/day=!{partitionKeyFromQuery:day}/hour=!{partitionKeyFromQuery:hour}/"
```

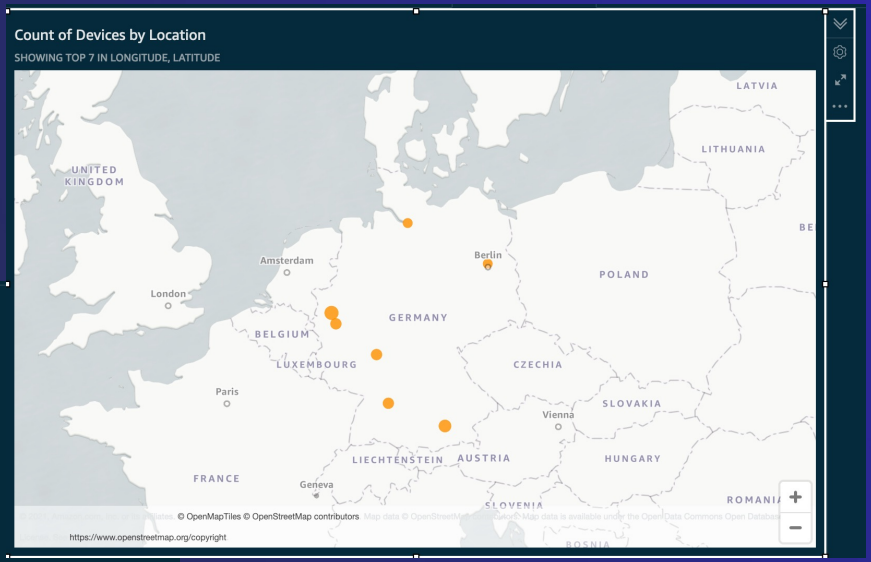
bucket_name > measures/ > json/ > year=2023/ > month=01/ > day=02/ > hour=01/

```
CREATE EXTERNAL TABLE raw_device_data (
 ...
)
...
PARTITIONED BY (
 year  INT,
 month INT,
 day   INT,
 hour  INT,
)
LOCATION "s3://DOC-EXAMPLE-BUCKET/prefix/"
TBLPROPERTIES (
 "projection.enabled" = "true",
 "projection.year.type" = "integer",
 "projection.year.range" = "2011,2099",
 "projection.year.digits" = "4",
 "projection.month.type" = "integer",
 "projection.month.range" = "1,12",
 "projection.month.digits" = "2",
 "projection.day.type" = "integer",
 "projection.day.range" = "1,31",
 "projection.day.digits" = "2",
 "projection.hour.type" = "integer",
 "projection.hour.range" = "0,23",
 "projection.hour.digits" = "2",
 "storage.location.template" = "s3://DOC-EXAMPLE-BUCKET/prefix/year=${year}/month=${month}/day=${day}
/hour=${hour}/"
)
```

```
"projection.enabled" = "true",
"projection.year.type" = "integer",
"projection.year.range" = "2011,2099",
"projection.year.digits" = "4",
```

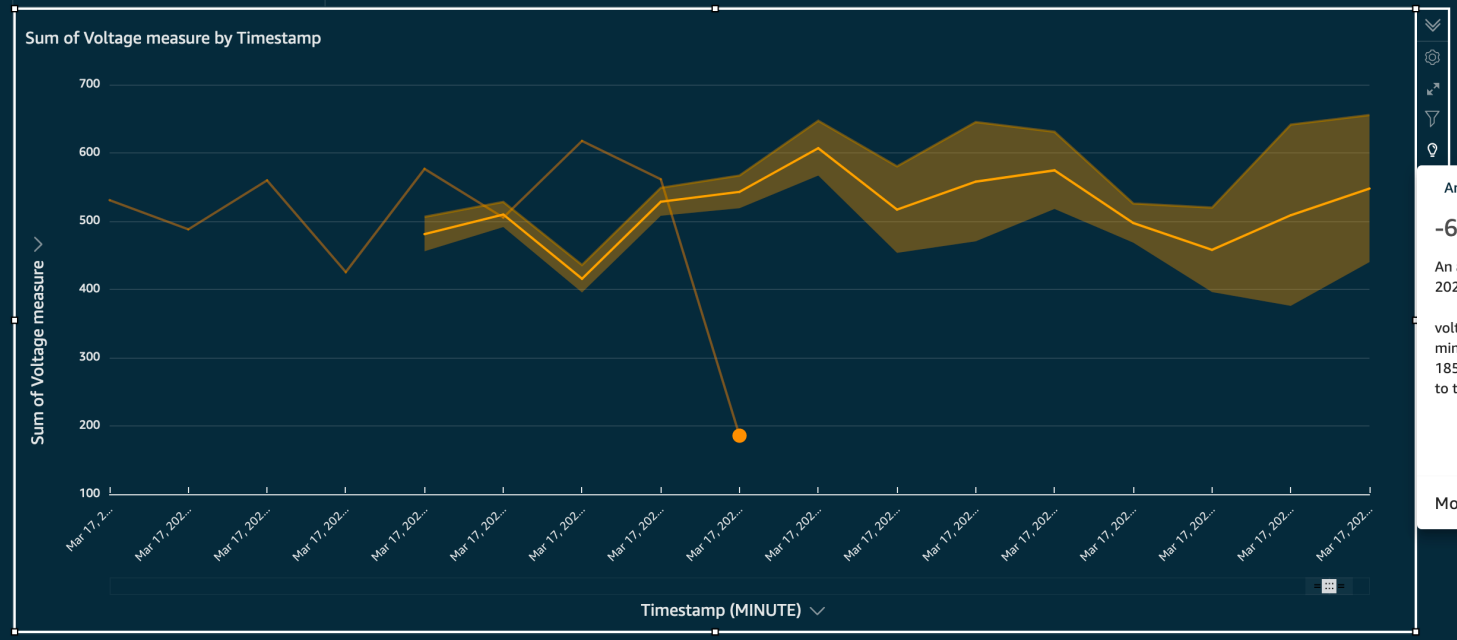
Insights Views

Amazon QuickSight



deviceid equals

02a6a13d-0c5c-47f1-b70c-e...



Anomaly Insight

-66.91% ↓

An anomaly was detected on Mar 17, 2021 7:36am primarily driven by than expected total voltage_measure of 507.25, where per minute total voltage_measure was 185.72, (375.58) compared to the previous minute.

More options

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

AWS Expansive Serverless Data Analytics in the Cloud

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Cost Drivers

Dynamic (per load)

- Athena when queried $5.00 per TB of data scanned
- Kinesis
  - Data ingested, per GB (Includes 24-hour retention) $0.08
  - Data retrievals, per GB $0.04
- Kinesis Firehose
  - Per GB Delivered $0.029 and JQ processing $0.07 per GB
- Amazon Timestream
  - 1 million write of 1KB size $0.50

Static (even when no events come)

- Amazon S3 Storage (has tiered storage from $0.023 per GB and as low as $0.004 per GB
- Amazon Timestream
  - Price per GB stored per hour $0.036
- Amazon Kinesis
  - Per stream, per hour $0.04
- Kinesis Data Analytics
  - Kinesis Processing Unit, Per Hour $0.11 per hour
  - Running Application Storage, Per GB-month $0.10 per GB-month

* Cost as of 28th of September 2022, in US-EAST-1 region, you pay for what you actually consume, checkout https://calculator.aws

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Purpose-Built AWS Data Analytics Services for Your Modern Data Architecture

- Work backwards from context and use cases, don't build technical platforms
- Use the right tool for your use case
- AWS has multiple serverless analytics services for different needs
- You're one Experiment away

© 2022, Amazon Web Services, Inc. or its aff

More Resources

- Near real-time processing with Amazon Kinesis, Amazon Timestream, and Grafana Blog Post
- Companion samples github repository

- Patterns for IoT time series ingestion Blog Post
- Smart meter analytics QuickStart

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Please complete the session survey

Direct link to slides right after 5-click survey feedback

https://eventbox.dev/survey/SQ1X96F

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

John Mousa

Amazon Web Services
Sr. Solutions Architect | AWS DACH Enterprise

/in/johnmousa

/johnmousa

@JohnMousa_

© 2022, Amazon Web Services, Inc. or its affiliates. All rights reserved.