
AWS
re:Invent

COM203

Building out a serverless application with Amazon QLDB

Matt Lewis

Chief Architect, UK Driver and Vehicle Licensing Agency (DVLA)

AWS Data Hero

AWS re:Invent

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

Agenda

- Position QLDB
- Introduce bicycle license serverless application
- Build out backend
- Build out using QLDB streams
- Resources



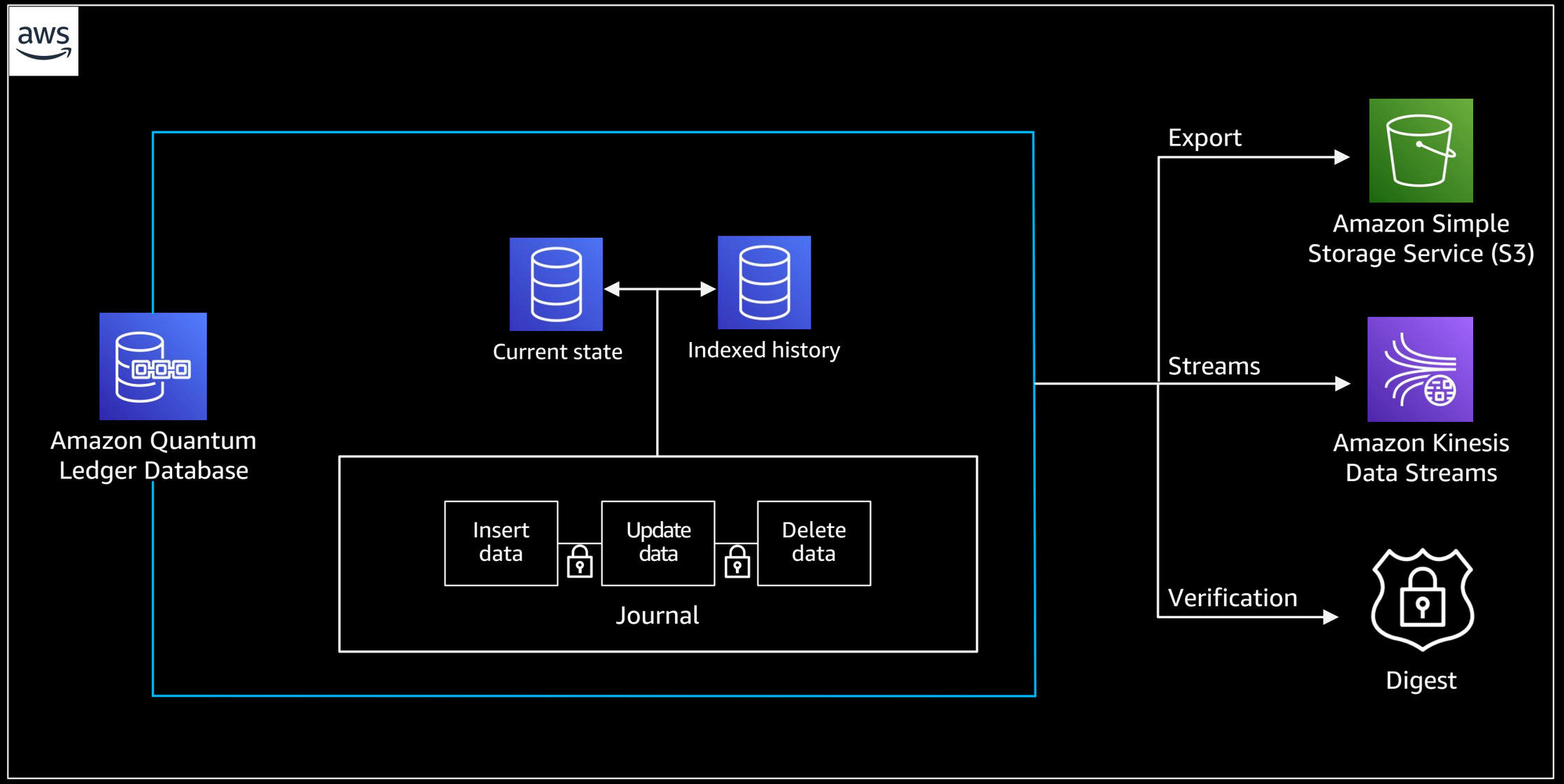
© 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Purpose-built databases at AWS

Choose from 15 purpose-built database engines

 Relational	 Key value	 Document	 In-memory	 Graph	 Time series	 Wide column	 Ledger
Referential integrity, ACID transactions, schema-on-write	High throughput, low-latency reads and writes, endless scale	Store documents and quickly access querying on any attribute	Query by key with microsecond latency	Quickly and easily create and navigate relationships between data	Collect, store, and process data sequenced by time	Flexible, low-latency reads and writes and endless scale	Complete, immutable, and verifiable history of all changes to application data

What is a ledger database?



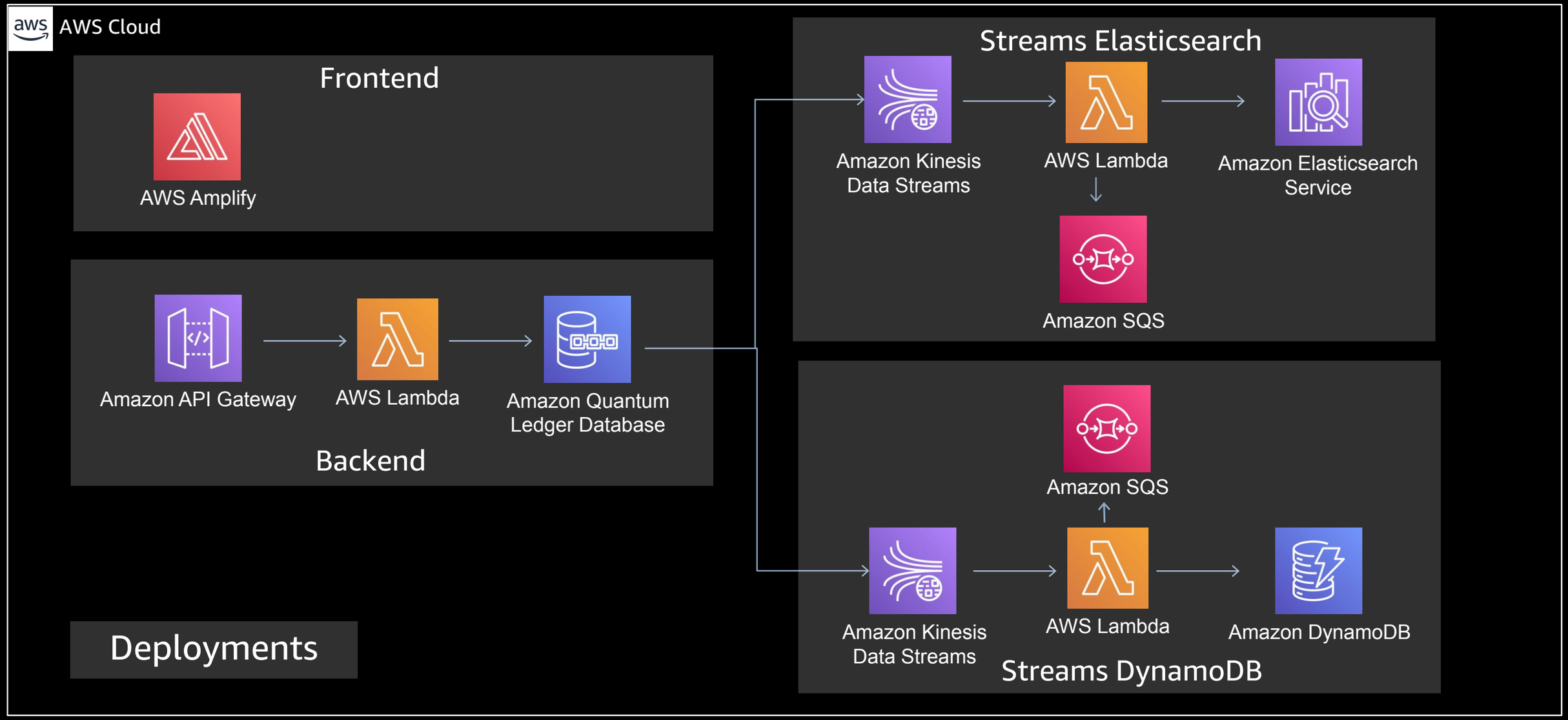
Building out bicycle license application



AWS re:Invent

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

Bicycle license serverless application



Infrastructure as code







Creating an Amazon QLDB ledger

```
Resources:
    QldbLedger:
        Type: AWS::QLDB::Ledger
        Properties:
            Name: qldb-bicycle-licence
            DeletionProtection: false
            PermissionsMode: ALLOW_ALL
            Tags:
                -
                    Key: name
                    Value: qldb-bicycle-licence
```

Using custom resource for table / index

```
QldbTable:
    Type: Custom::CreateQldbTable
    DependsOn: QldbLedger
    Properties:
        ServiceToken: !Sub arn:aws:lambda:...:function:${CreateQldbTable}
        Version: 1.0
```

Adding permissions

```
CreateLicence:
  Type: AWS::Serverless::Function
  Properties:
    ..
    Policies:
      - AWSLambdaBasicExecutionRole
      - Version: 2012-10-17
        Statement:
        - Effect: Allow
          Action:
            - qldb:SendCommand
          Resource: !Sub arn:aws:qldb…:ledger/${QldbLedger}
```

Creating a QLDB driver

```
const { QldbDriver, RetryConfig } = require('amazon-qldb-driver-nodejs');
const qldbDriver = createQldbDriver();

function createQldbDriver(
      ledgerName = process.env.LEDGER_NAME,
      serviceConfigurationOptions = {},
) {
      const retryConfig = new RetryConfig(4);
      const qldbDriver = new QldbDriver(ledgerName, serviceConfigurationOptions, 10,
retryConfig);
      return qldbDriver;
}
function getQldbDriver() {
  return qldbDriver;
}
```

Interacting with QLDB

```
const createLicence = async (param1, param2, ...) => {

  const qldbDriver = await getQldbDriver();
  await qldbDriver.executeLambda(async (txn) => {

    // all functionality committed as one atomic transaction
    ...
  });
};
```

Executing statements against QLDB

```
const doc = { name: 'Matt', points: 0, ...};
const statement = 'INSERT INTO Licence ?';
txn.execute(statement, doc);

const statement = ‘SELECT name, points FROM Licence where ID =  ?’;
const result = txn.execute(statement, ID);
const resultList = result.getResultList();
const name = resultList[0].get('name').stringValue();
const points = resultList[0].get('points').numberValue();

JSON.stringify(resultList[0]);
```

Conditional check

```
await qldbDriver.executeLambda(async (txn) => {
    // Check if the record already exist
    const recordsReturned = await checkIdUnique(txn, id);
    if (recordsReturned === 0) {
        // record unique
    } else {
        // record not unique
    }
});
```

Using metadata ID

```
await qldbDriver.executeLambda(async (txn) => {
  // Creating record returns unique document Id
  const result = await insertDocument(txn, docToInsert);
  const docIdArray = result.getResultList();
  const docId = docIdArray[0].get('documentId').stringValue();
  // Update document with documentId
  await addDocId(txn, docId, otherId);
  ...
});
```

Testing

- Testing AWS Cloud resources from local development environment

```
sls invoke local –function functionName
Sam local invoke [OPTIONS] [FUNCTION_IDENTIFIER]
```

- Load testing against API Gateway endpoints

```
artillery run …
config:
  target: "https://{API}.execute-api.{region}.amazonaws.com/{stage}/"
  phases:
    - duration: 300
      arrivalRate: 10
```

Optimization top tips

1. Enable AWS X-Ray tracing
2. Gather metrics using Amazon CloudWatch log insights
3. Enable HTTP keep-alive
4. Reduce cold start times
5. Optimize Lambda configuration

Enable X-Ray tracing

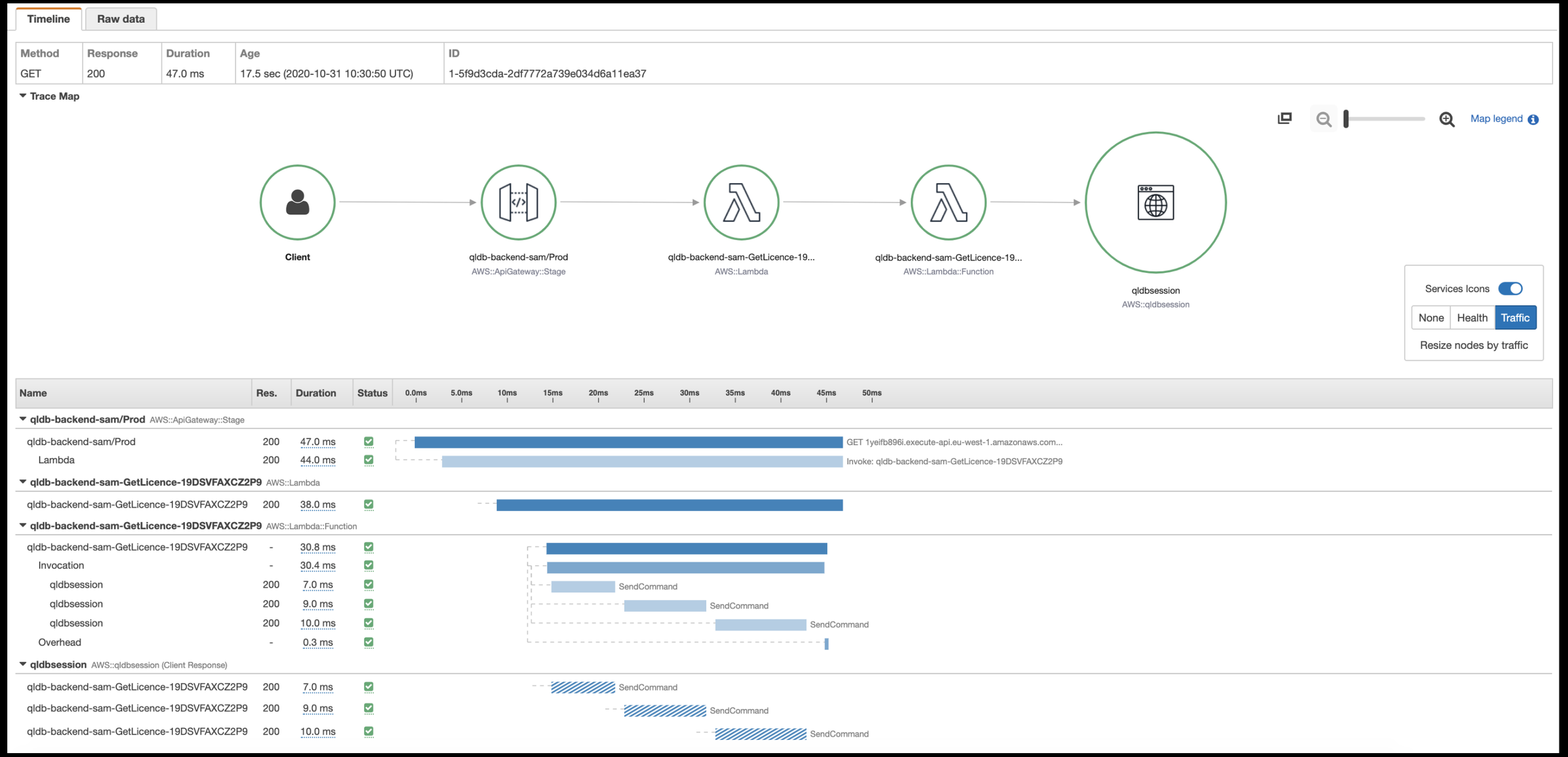
template.yml

```
Globals:
  Api:
    TracingEnabled: true
  Function:
    Tracing: Active
```

JavaScript

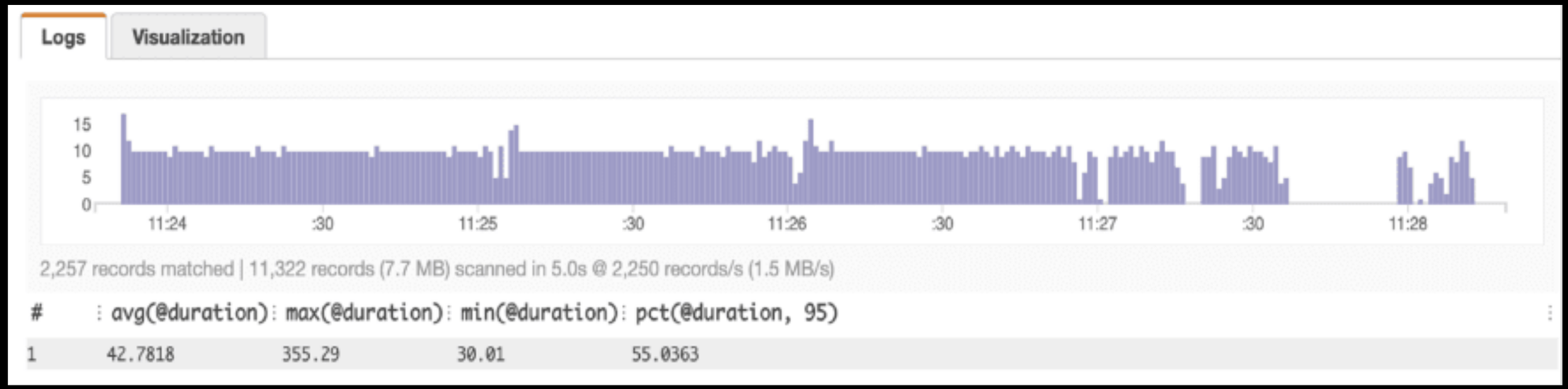
```
const AWSXRay = require('aws-xray-sdk-core’);
AWSXRay.captureAWS(require('aws-sdk'));
```

Enable X-Ray tracing



CloudWatch log insights

filter @type = "REPORT" | stats avg(@duration), max(@duration), min(@duration), pct(@duration, 95)



Enable HTTP keep-alive

template.yml

```
Globals:
  Function:
    Environment:
      Variables:
        AWS_NODEJS_CONNECTION_REUSE_ENABLED: 1
```

serverless.yml

```
provider:
  environment:
    AWS_NODEJS_CONNECTION_REUSE_ENABLED: 1
```

Reduce cold start times

Original package size

```
Serverless: Uploading service ...zip file to S3 (10.18 MB)...
```

Minimize dependencies using webpack

```
Serverless: Uploading service ...zip file to S3 (1.91 MB)...
```

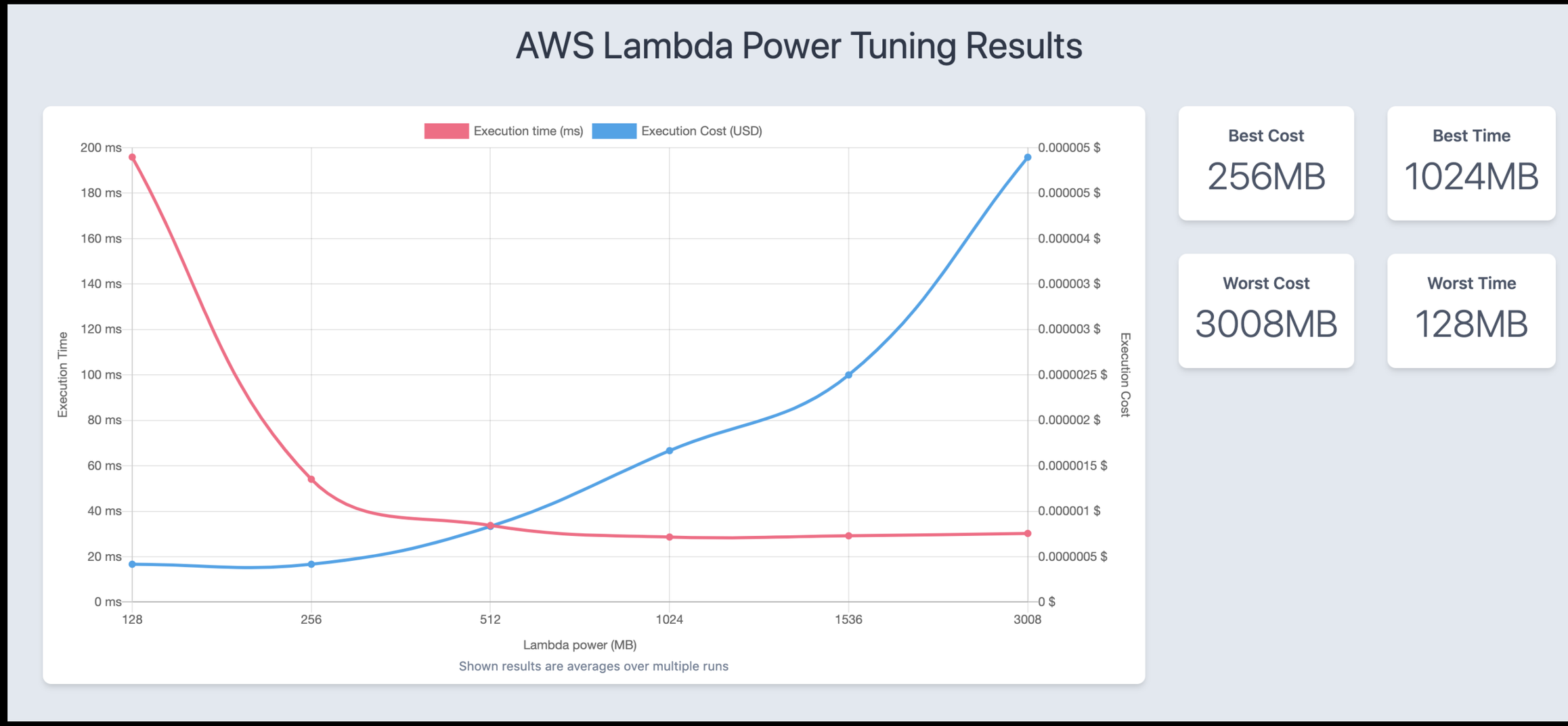
```
lumigo-cli analyze-lambda-cold-starts /
-m <mins> /
-n <function-name> /
-r <region>
```

Optimize AWS Lambda configuration

Use AWS Lambda Power Tuning for optimal memory allocation

```
lumigo-cli powertune-lambda \
-f <input-file> \
-n <function-name> \
-o <output-file> \
-r <region> \
-s balanced
```

Power Tuning visualization



QLDB streams



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

What is QLDB streams?



QLDB streams overview

- All committed changes streamed out from journal
- QLDB remains the immutable, complete, and verifiable source of truth
- At-least-once delivery guarantee
- Multiple streams from same ledger with different start / end dates and times

Create Kinesis data stream

```
LicenceStreamKinesis:
    Type: AWS::Kinesis::Stream
    Properties:
      Name: LicenceStreamKinesis
      RetentionPeriodHours: 24
      ShardCount: 1
```

QLDB assumes role with permissions policy

```
LicenceStreamsKinesisRole: # Used by QLDB to write to Kinesis Data Streams
    Type: AWS::IAM::Role
    Properties:
      AssumeRolePolicyDocument:
        Version: '2012-10-17'
        Statement:
          - Effect: Allow
            Principal:
              Service: qldb.amazonaws.com
            Action: sts:AssumeRole
```

QLDB permission policy

```
Policies:
  - PolicyName: QLDBStreamDynamoDBKinesisPermissions
    PolicyDocument:
      Version: '2012-10-17'
      Statement:
        - Effect: Allow
          Action:
            - kinesis:ListShards
            - kinesis:DescribeStream
            - kinesis:PutRecord*
          Resource: !GetAtt LicenceStreamKinesis.Arn
```

Create QLDB stream

```
LicenceQLDBStream:
    Type: AWS::QLDB::Stream
    DependsOn: LicenceStreamsKinesisRole
    Properties:
      InclusiveStartTime: "2020-05-29T00:00:00Z"
      KinesisConfiguration:
        AggregationEnabled: true
        StreamArn: !GetAtt LicenceStreamKinesis.Arn
      LedgerName: !ImportValue qldb-bicycle-licence
      RoleArn: !GetAtt LicenceStreamsKinesisRole.Arn
      StreamName: qldb-licence-dynamodb-sam
```

Configure event source mapping

```
MyEventSourceMapping:
   Type: AWS::Lambda::EventSourceMapping
   Properties:
     BatchSize: 50
     BisectBatchOnFunctionError: true
     DestinationConfig:
       OnFailure:
         Destination: !GetAtt StreamsFailureQueue.Arn
     EventSourceArn: !GetAtt LicenceStreamKinesis.Arn
     FunctionName: !GetAtt QldbStreamsKinesisLambdaFunction.Arn
     MaximumBatchingWindowInSeconds: 30
     MaximumRecordAgeInSeconds: -1
     MaximumRetryAttempts: 2
     ParallelizationFactor: 1
```

QLDB streams record types

- Control
- Block
- Revision details

```
{
  qldbStreamArn: string,
  recordType: CONTROL | BLOCK | REVISION_DETAILS,
  payload: {
    // data
  }
}
```

Process Kinesis records

```
async function processRecords(records) {
  await Promise.all(
    records.map(async (record) => {
      // Kinesis data is base64 encoded so decode here
      const payload = Buffer.from(record.data, "base64");

      // payload is ion binary record published by QLDB to the stream
      const ionRecord = ion.load(payload);

      // Only process records where the record type is REVISION_DETAILS
      if (ionRecord.recordType.stringValue() === “REVISION_DETAILS”) {
        await processIon(ionRecord);
      }
```

Extract values from Ion

```
async function processIon(ionRecord) {
  const version = ionRecord.payload.revision.metadata.version.numberValue();
  const id = ionRecord.payload.revision.metadata.id.stringValue();

  // Check to see if the data section exists.
  if (! ionRecord.payload.revision.data) {
    // document deleted
  } else {
    // New or updated document
    const name = ionRecord.payload.revision.data.name.stringValue();
    ...
  }
}
```

Handling duplicate records in DynamoDB

```
const params = {
  TableName: TABLE_NAME,
  Key: { pk: id },
  UpdateExpression: ‘..., version=:version’,
  ExpressionAttributeValues: {
    ...
    ':version': version,
  },
  ConditionExpression: 'attribute_not_exists(pk) OR version < :version',
};
```

Handling duplicate records in Elasticsearch

```
const doc = {
  ...
};
response = await sendRequest({
  httpMethod: 'PUT’,
  requestPath:
`/licence/_doc/${id}?version=${version}&version_type=external`,
  payload: doc,
});
```

Resources

- Code repository
  - https://github.com/AWS-South-Wales-User-Group/qldb-bicycle-licence
- QLDB Guide
  - https://qldbguide.com/
- Awesome QLDB
  - https://github.com/mlewis7127/awesome-qldb
- Go build!

Thank you!

Matthew Lewis

Chief Architect, UK Driver and Vehicle Licensing Agency (DVLA)
AWS Data Hero
@m_lewis

© 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.