INNOVATE2018
ONLINE CONFERENCE
DEVELOPER EDITION

# Amazon Neptune

**Ian Robinson, DBS Customer Advisory, EMEA | 2018**

**ianrob@amazon.com**

@awscloud

#AWSInnovate

## Amazon Neptune – A Fast, Reliable Graph Database

Optimized for storing and querying highly connected data

# Neptune Use Cases

Social Networking

Recommendations

Knowledge Graphs

Fraud Detection

Life Sciences

Network & IT Operations

## Characteristics of a Graph Workload

- Complex domain model
- Variable schema (per entity)
    - Optional attributes
- Variable structure (across domain)
    - Highly connected data
- Queries require joining entities or navigating relationships
    - Need to understand that things are connected
    - Need to understand *strength*, *weight* or *quality* of relationships

## Navigate a Web of Global Tax Policies

THOMSON REUTERS

*"Our customers are increasingly required to* ***navigate a complex web of global tax policies*** *and regulations. We need an approach to* ***model the sophisticated corporate structures*** *of our largest clients and deliver an end-to-end tax solution. We use a microservices architecture approach for our platforms and are beginning to leverage Amazon Neptune as a graph-based system to quickly* ***create links within the data****."*

Tim Vanderham, Chief Technology Officer, Thomson Reuters Tax & Accounting

# Amazon Neptune – A Fully-Managed Graph Database

**Fast**

Query billions of relationships with millisecond latency

**Reliable**

6 replicas of your data across 3 AZs with full backup and restore

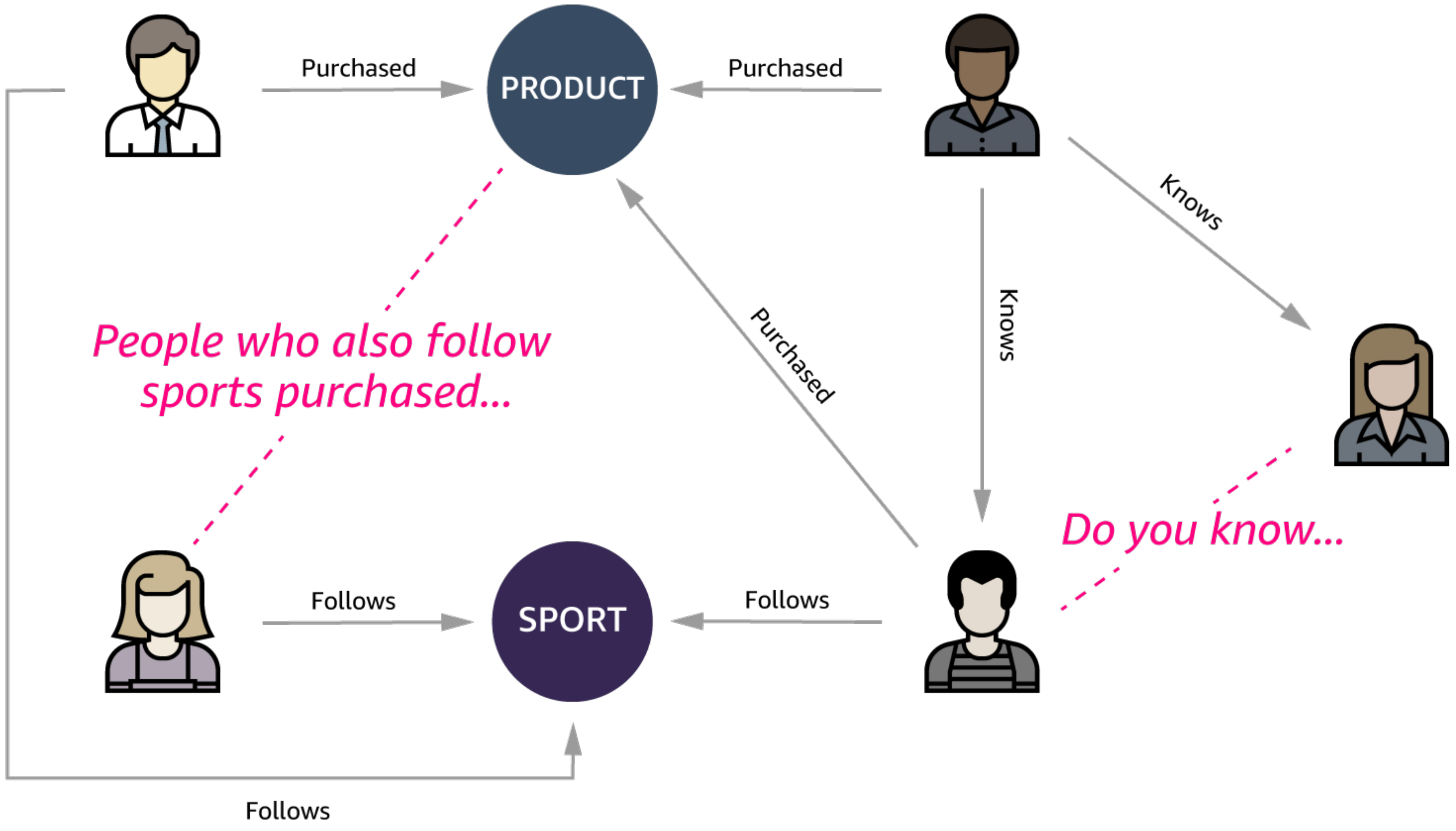
**Easy**

Build powerful queries easily with Gremlin and SPARQL

**Open**

Supports Apache TinkerPop & W3C RDF graph models

# Recommendations Based on Relationships

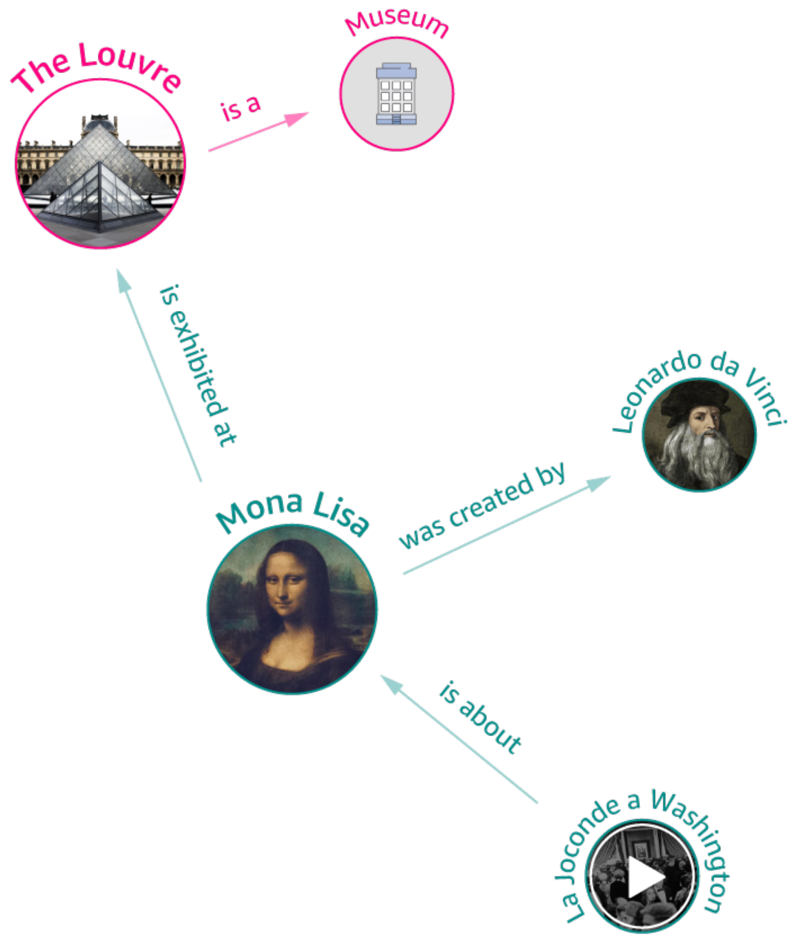


# Knowledge Graph Applications

Who painted the Mona Lisa?

What museums should Alice visit while in Paris?

What artists have paintings in The Louvre?



## Leading Graph Models and Frameworks

### Property Graph

- Vertices and edges (nodes and relationships) with properties
- Both record-like items

### Resource Description Framework (Rdf)

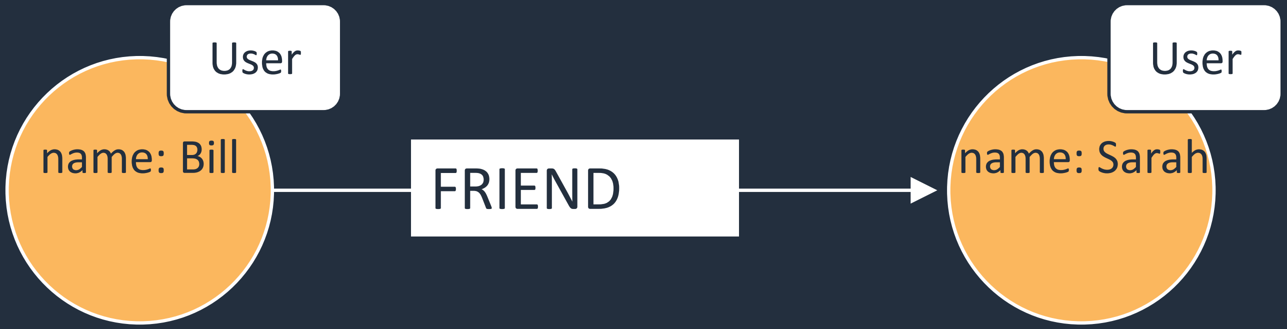
- Triples
- subject – predicate – object

# Creating a Property Graph

Add 2 vertices

```
g.addV('User').property('name','Bill');
g.addV('User').property('name','Sarah');

...

g.V().hasLabel('User').has('name','Sarah').as('a').
  V().hasLabel('User').has('name','Bill').
  addE('FRIEND').to('a');
```

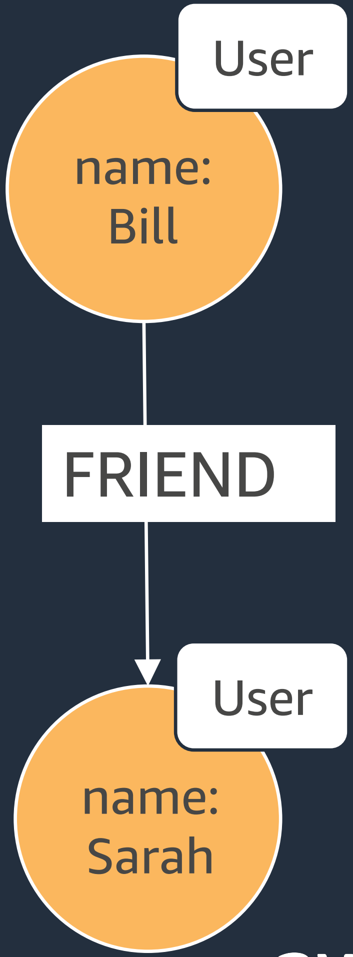
Connect with an edge



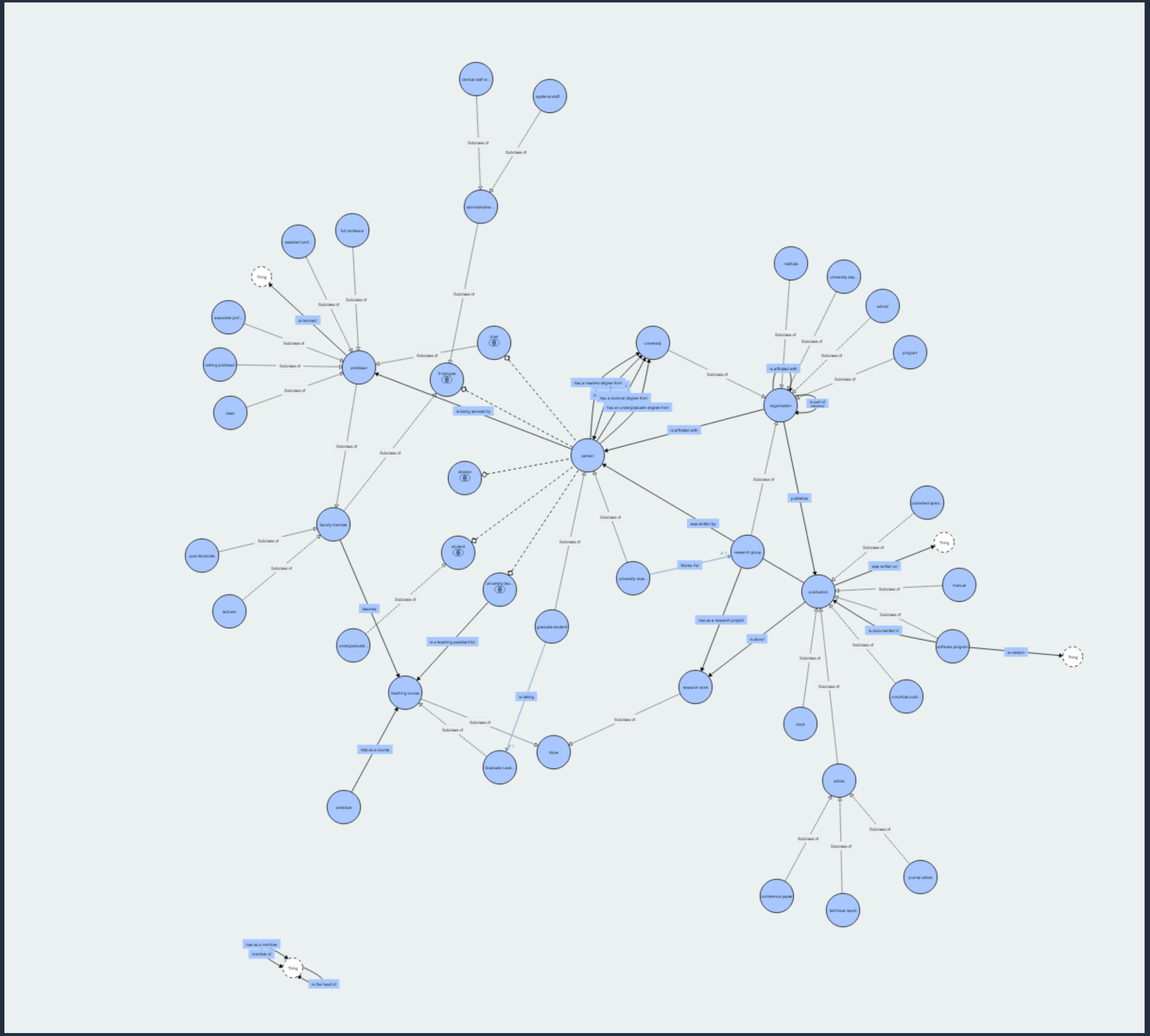
Gremlin (Apache TinkerPop 3.x)

## Creating an RDF Graph

```
@prefix contacts: <http://www.socialnetwork.com/people#>.


<http://www.socialnetwork.com/person#1>
   rdf:type contacts:User;
   contact:name: ”Bill” .

<http://www.socialnetwork.com/person#1>
   contacts:friend <http://www.socialnetwork.com/person#2> .



<http://www.socialnetwork.com/person#2>
   rdf:type contacts:User;
   contact:name: ”Sarah” .
```

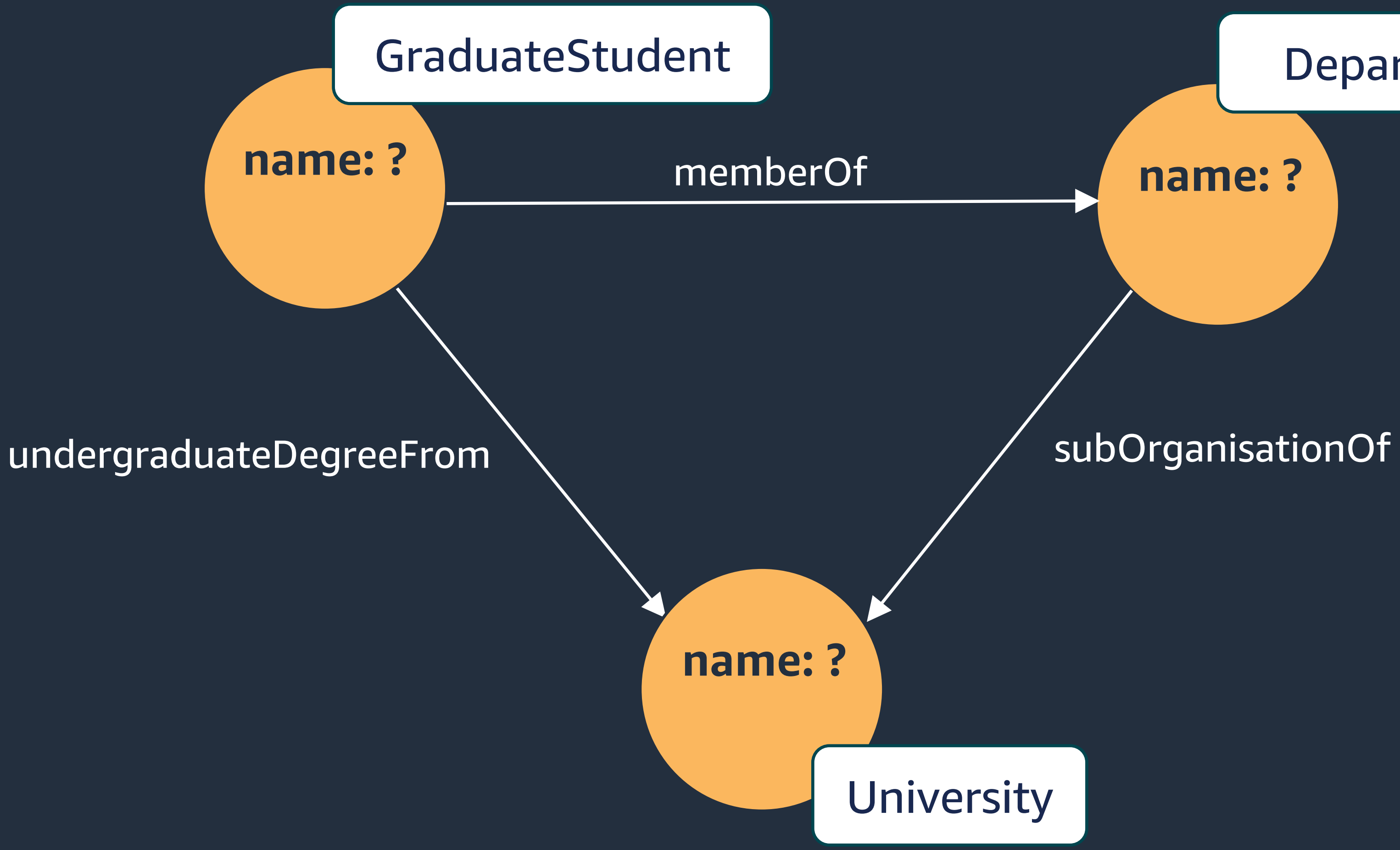


# Example: Universities



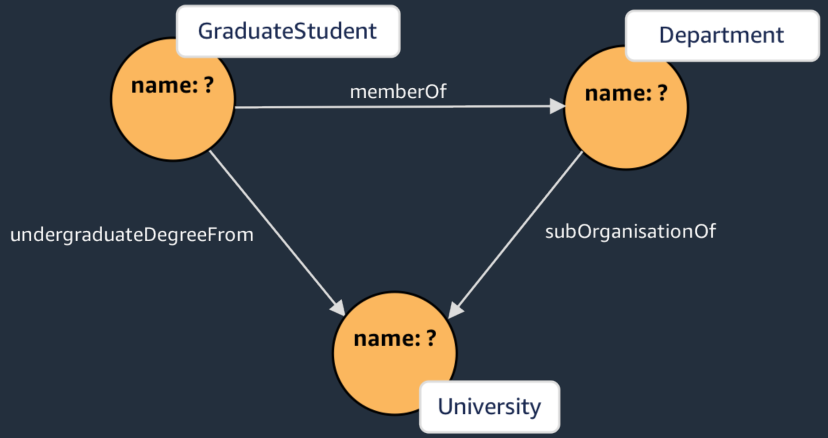
- Institutions
- Organisational structure
- Roles and job titles
- Research, teaching and administrative staff
- Subjects and course catalogs
- Timetables
- Undergraduate and graduate populations

## Find the graduate students who received an undergraduate degree from the same university



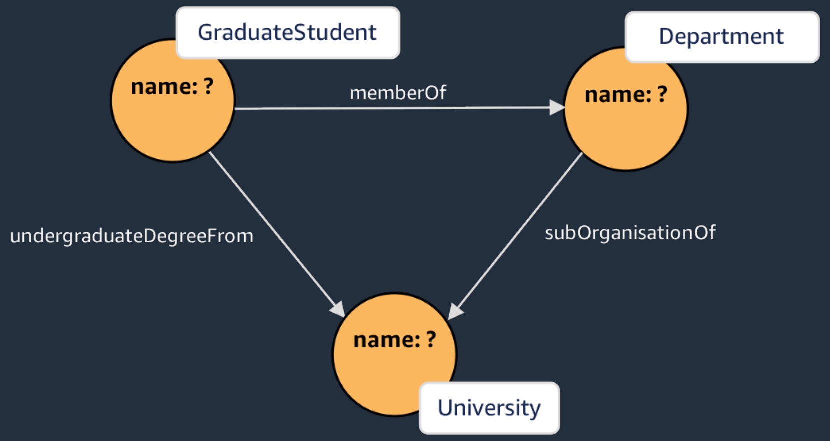
## Gremlin

```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



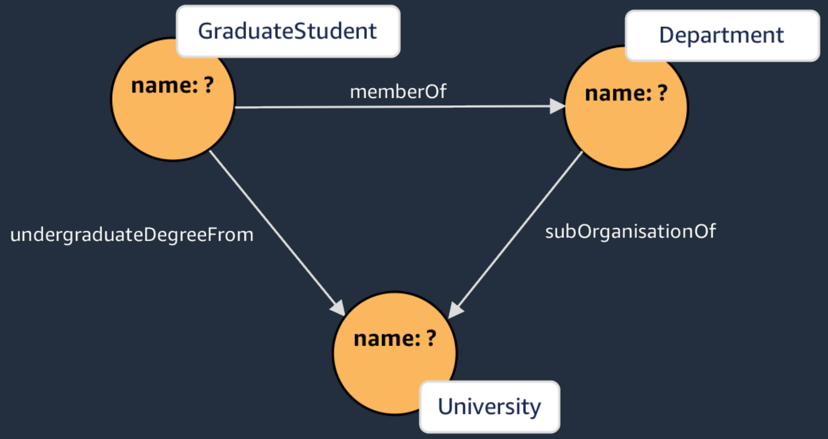
# Gremlin

```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



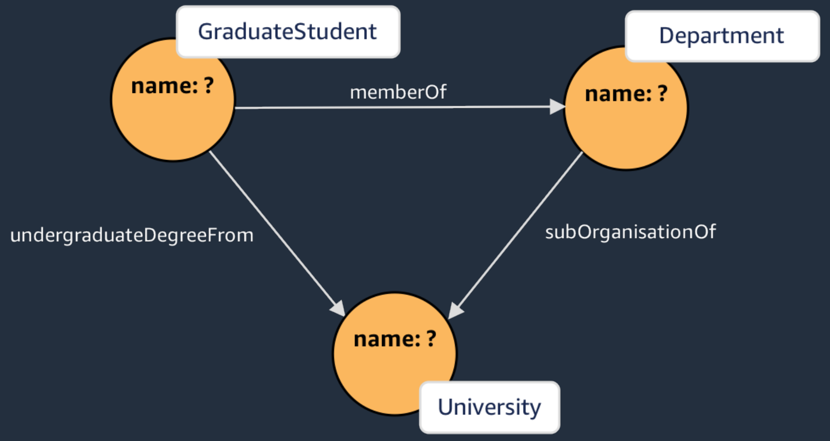
## Gremlin

```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



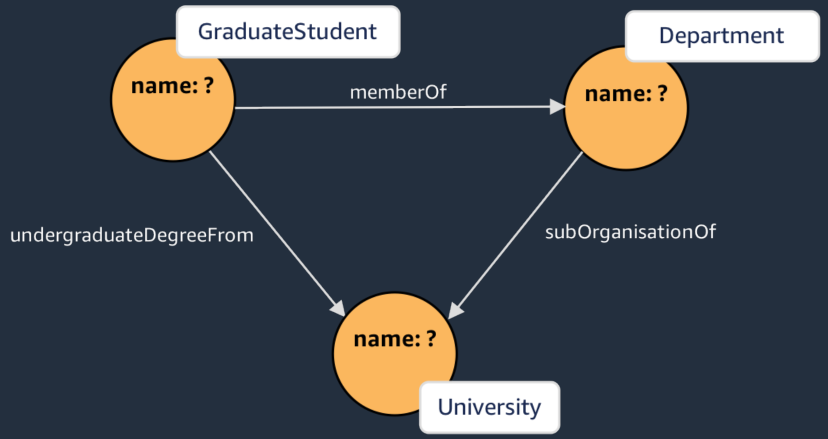
# Gremlin

```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



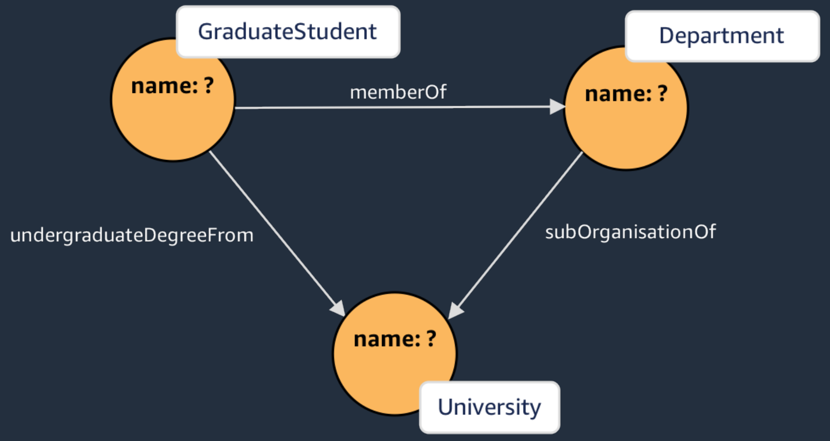
## Gremlin

```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



## Gremlin

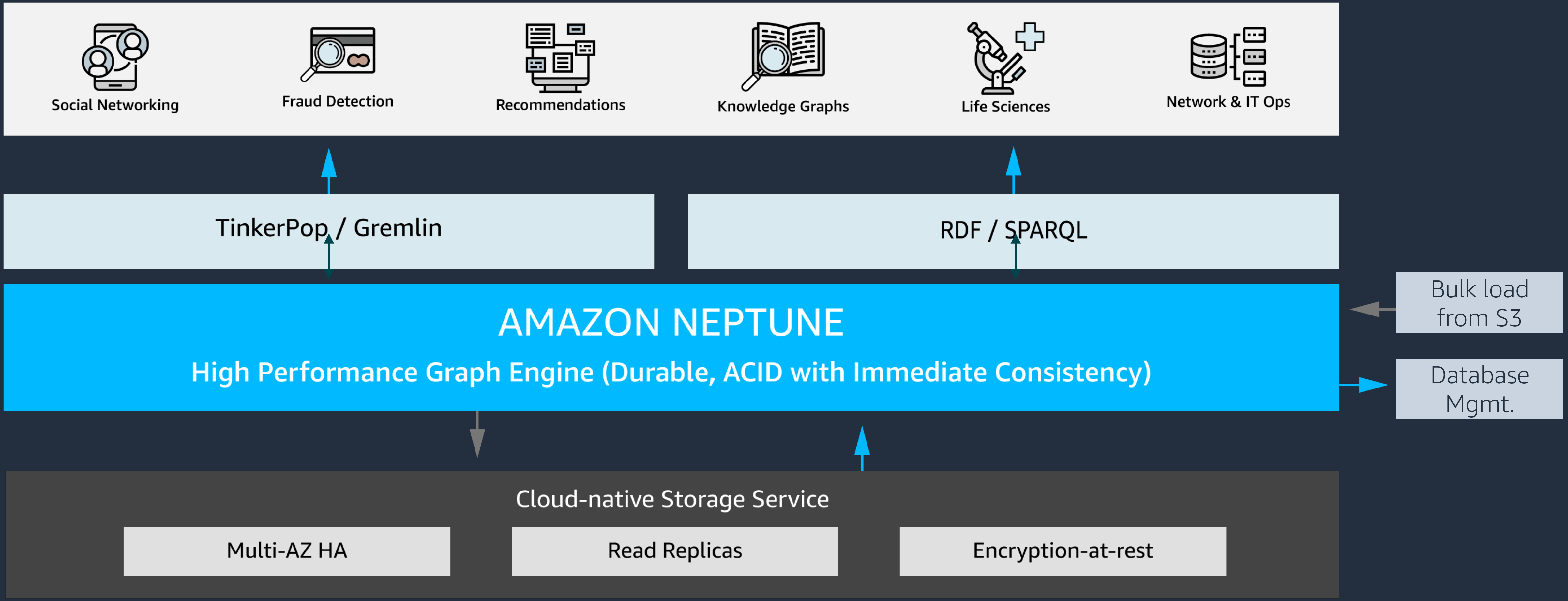
```
g.V().hasLabel('GraduateStudent').as('student').
  out('memberOf').
  out('subOrganisationOf').
  in('undergraduateDegreeFrom').
  where(eq('student'))
```



# SPARQL

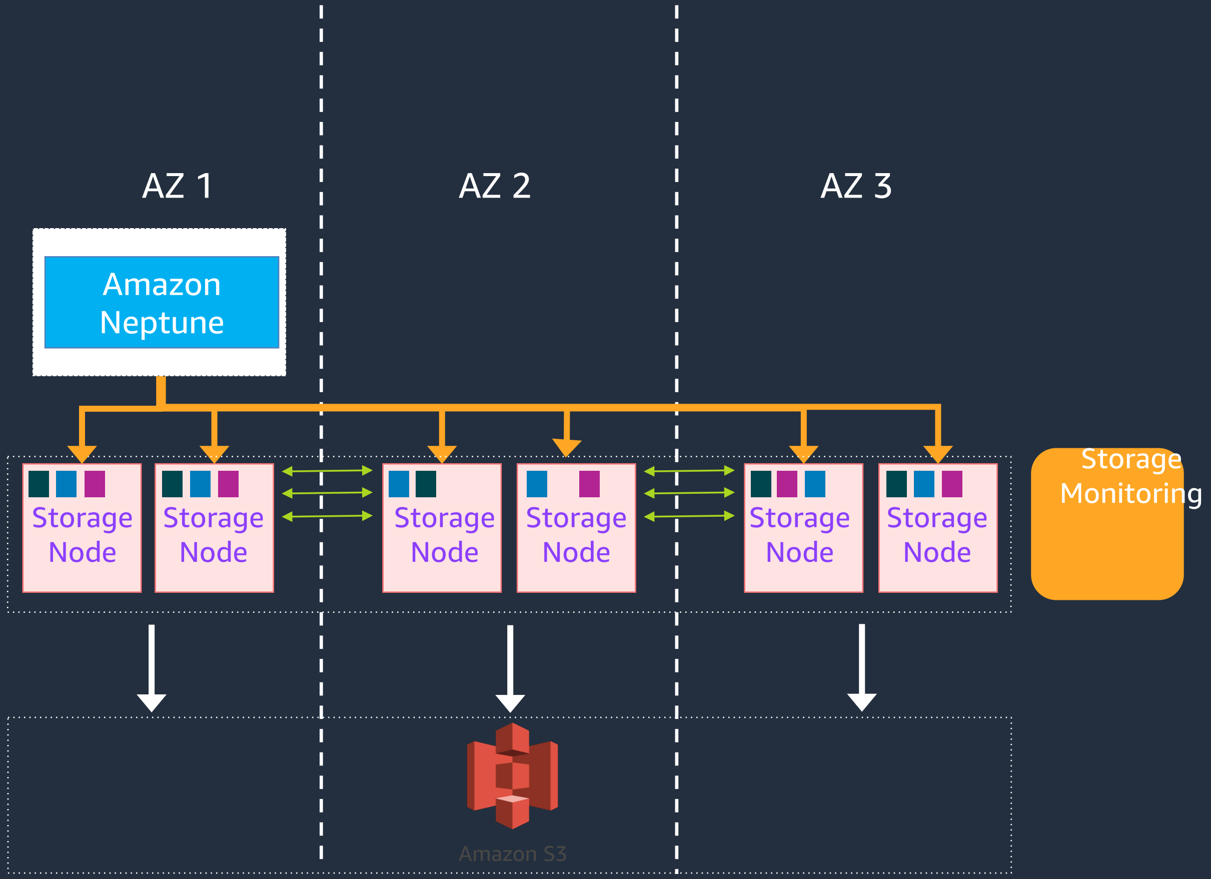
```
PREFIX rdf:http://www.w3.org/1999/02/22-rdf-syntax-ns#
PREFIX ub:http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl#

SELECT ?student WHERE {
      ?student rdf:type ub:GraduateStudent .
      ?univ rdf:type ub:University .
      ?dept rdf:type ub:Department .
      ?student ub:memberOf ?dept .
      ?dept ub:subOrganizationOf ?univ .
      ?student ub:undergraduateDegreeFrom ?univ
}
```

# Amazon Neptune High Level Architecture



# Cloud-Native Storage

- Data is replicated 6 times across 3 AZs
- Continuous backup to Amazon S3
  - built for 11 9s durability
- Continuous monitoring of nodes and disks
- 10 GB segments as unit of repair or hotspot rebalance
- Quorum system for read/write; latency tolerant
- Quorum membership changes do not stall writes
- Storage volume automatically grows up to 64 TB



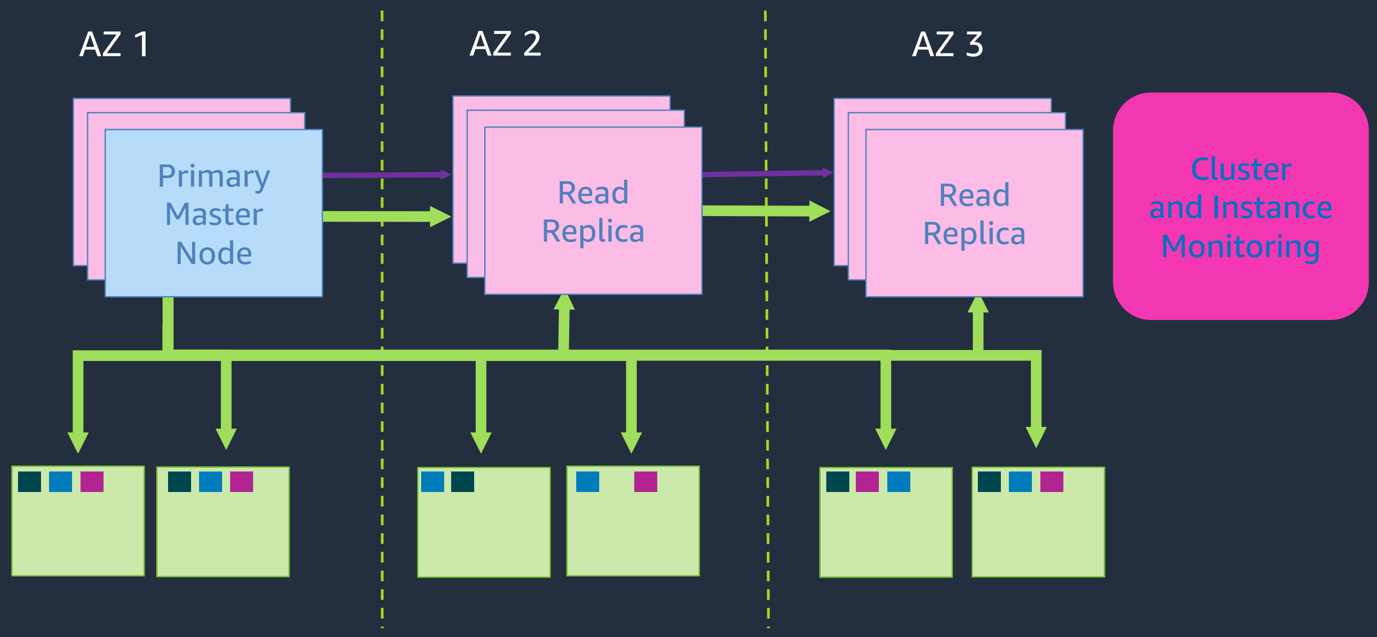
# Amazon Neptune Read Replicas

## Availability

- Failing database nodes are automatically detected and replaced
- Failing database processes are automatically detected and recycled
- Replicas are automatically promoted to primary if needed (failover)
  - Customer specifiable failover order

## Performance

- Customer applications can scale out read traffic across read replicas
- Read balancing across read replicas
  - Use reader endpoint



## Security

- Network isolation via Virtual Private Cloud
  - Use security groups to control ingress
- Encryption at rest using AWS Key Management Service (KMS)
  - underlying storage, automated backups, snapshots, and replicas in the same cluster
- IAM Policies to secure resources
- IAM Database Authentication
  - IAM Policies for database access
  - Each request must be signed using AWS Signature Version 4
  - Libraries for Gremlin and SPARQL clients

# Loading Data into Neptune

**Bulk Load API**

- UTF-8 encoded files in S3
- Supports `gzip` compression of single files
- Requires a VPC endpoint for Amazon S3
- Add IAM Role to Neptune allowing `s3::Get*` and `s3::List*` permissions for S3 bucket
- CSV formatted files for Gremlin; 4 standard formats for RDF (N-Triples, N-Quads, RDF/XML, Turtle)
- Load, get status and cancel job via `/loader` HTTP endpoint
- Append-only

## Backup and Restore

### Automated Backups

- Daily automated backups during backup window
- Full storage volume snapshot
- Taken from read replica if possible
- Retention period: 1-35 days

### Manual Snapshots

- Backup entire DB instance
- Can be shared with other AWS accounts

## Backup and Restore

**Restoring from a DB Snapshot (Automated or Manual)**

- Creates new DB instance
- Apply custom parameter groups and security groups after restore using **Modify** command

**Point-in-Time Restore**

- Restore from DB instance (not snapshot)
- Creates new DB instance
- Choose “Latest restorable time” or specify custom data and time
- 1 second granularity

# Monitoring

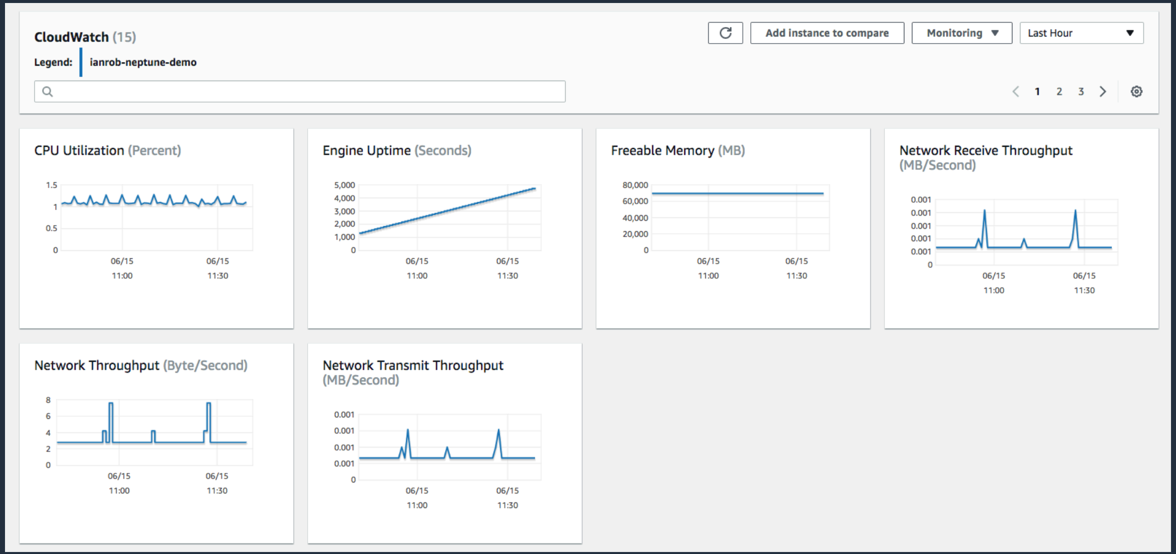
## AWS CloudTrail

- Log all Neptune API calls to S3 bucket

## Event Notifications

- Create SNS subscription via CLI or SDK
- Sources: db-instance | db-cluster | db-parameter-group | db-security-group | db-snapshot | db-cluster-snapshot

## Amazon CloudWatch

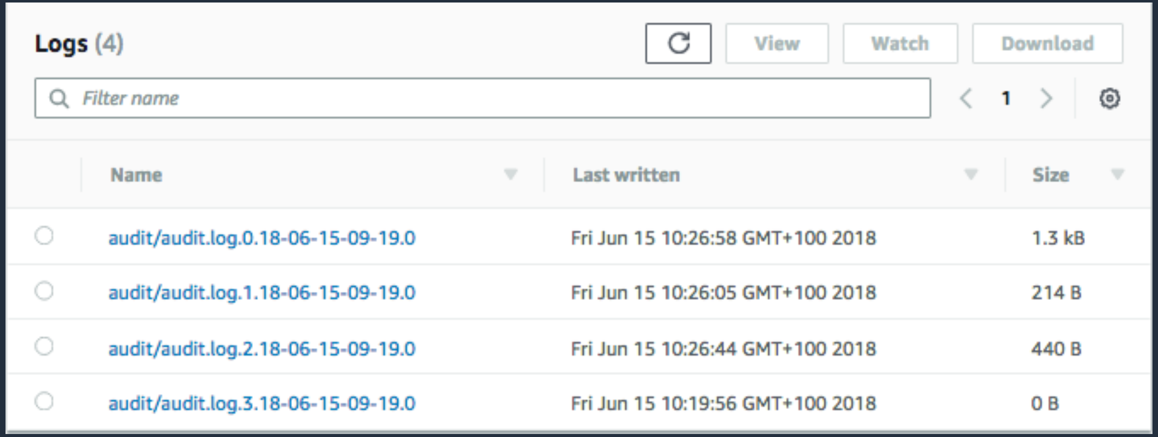


CloudWatch (15)
Add instance to compare
Monitoring
Last Hour
Legend: ianrob-neptune-demo
CPU Utilization (Percent)
Engine Uptime (Seconds)
Freeable Memory (MB)
Network Receive Throughput (MB/Second)
Network Throughput (Byte/Second)
Network Transmit Throughput (MB/Second)
06/15 11:00
06/15 11:30

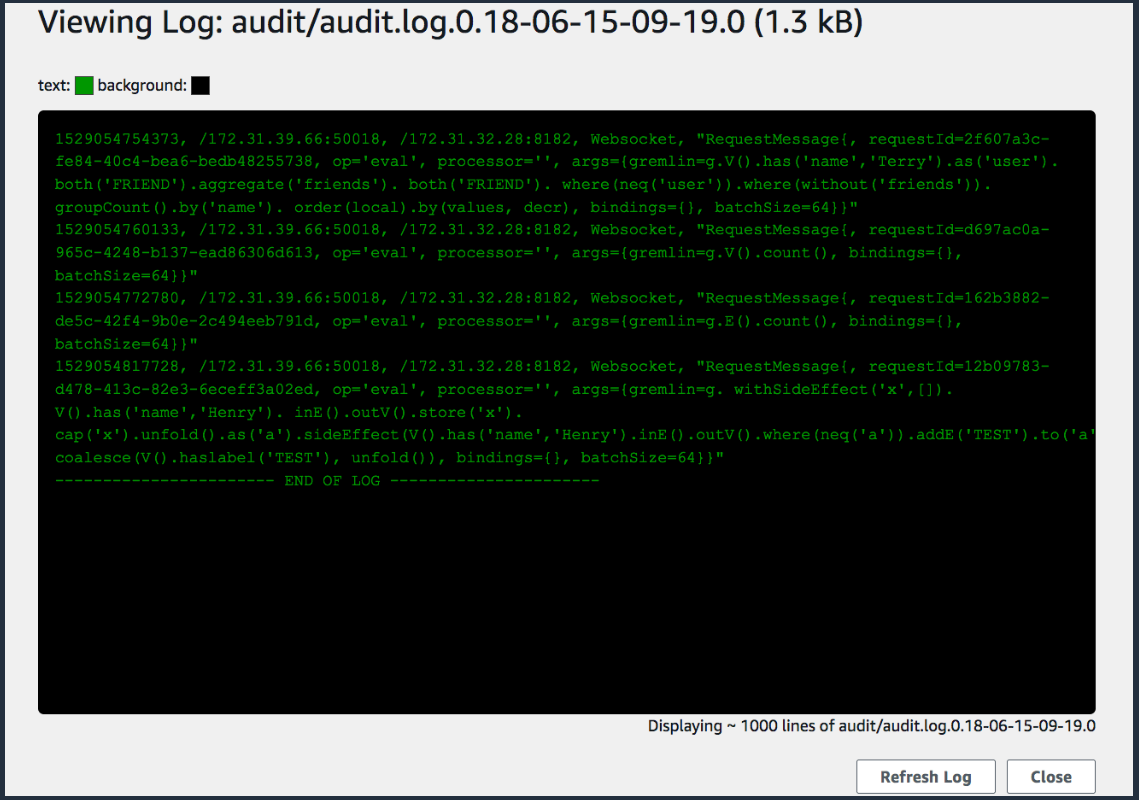
| | | | |
|---|---|---|---|
| CPUUtilization | GremlinRequestsPerSec | Http429 | SparqlErrors |
| ClusterReplicaLag | Http100 | Http500 | SparqlRequests |
| ClusterReplicaLagMaximum | Http101 | Http501 | SparqlRequestsPerSec |
| ClusterReplicaLagMinimum | Http200 | LoaderErrors | StatusErrors |
| EngineUptime | Http400 | LoaderRequests | StatusRequests |
| FreeableMemory | Http403 | NetworkReceiveThroughput | VolumeBytesUsed |
| GremlinErrors | Http405 | NetworkThroughput | VolumeReadIOPs |
| GremlinRequests | Http413 | NetworkTransmitThroughput | VolumeWriteIOPs |

# Audit Logs

## `neptune_enable_audit_log` DB cluster parameter



| | Name | Last written | Size |
|---|---|---|---|
| ○ | audit/audit.log.0.18-06-15-09-19.0 | Fri Jun 15 10:26:58 GMT+100 2018 | 1.3 kB |
| ○ | audit/audit.log.1.18-06-15-09-19.0 | Fri Jun 15 10:26:05 GMT+100 2018 | 214 B |
| ○ | audit/audit.log.2.18-06-15-09-19.0 | Fri Jun 15 10:26:44 GMT+100 2018 | 440 B |
| ○ | audit/audit.log.3.18-06-15-09-19.0 | Fri Jun 15 10:19:56 GMT+100 2018 | 0 B |



Viewing Log: audit/audit.log.0.18-06-15-09-19.0 (1.3 kB)

text: 🟩 background: ■

```
1529054754373, /172.31.39.66:50018, /172.31.32.28:8182, Websocket, "RequestMessage{, requestId=2f607a3c-
fe84-40c4-bea6-bedb48255738, op='eval', processor='', args={gremlin=g.V().has('name','Terry').as('user').
both('FRIEND').aggregate('friends'). both('FRIEND'). where(neq('user')).where(without('friends')).
groupCount().by('name'). order(local).by(values, decr), bindings={}, batchSize=64}}"
1529054760133, /172.31.39.66:50018, /172.31.32.28:8182, Websocket, "RequestMessage{, requestId=d697ac0a-
965c-4248-b137-ead86306d613, op='eval', processor='', args={gremlin=g.V().count(), bindings={},
batchSize=64}}"
1529054772780, /172.31.39.66:50018, /172.31.32.28:8182, Websocket, "RequestMessage{, requestId=162b3882-
de5c-42f4-9b0e-2c494eeb791d, op='eval', processor='', args={gremlin=g.E().count(), bindings={},
batchSize=64}}"
1529054817728, /172.31.39.66:50018, /172.31.32.28:8182, Websocket, "RequestMessage{, requestId=12b09783-
d478-413c-82e3-6eceff3a02ed, op='eval', processor='', args={gremlin=g. withSideEffect('x',[]).
V().has('name','Henry'). inE().outV().store('x').
cap('x').unfold().as('a').sideEffect(V().has('name','Henry').inE().outV().where(neq('a')).addE('TEST').to('a'
coalesce(V().haslabel('TEST'), unfold()), bindings={}, batchSize=64}}"
---------------------- END OF LOG ---------------------
```

Displaying ~ 1000 lines of audit/audit.log.0.18-06-15-09-19.0

- Timestamp
- ServerHost
- ClientHost
- ConnectionType
- RequestMesssage

INNOVATE 2018
ONLINE CONFERENCE

DEVELOPER EDITION

Thank you!

@awscloud

#AWSInnovate