

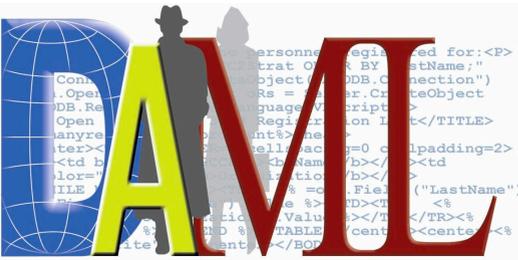


Putting Data into Context Using Amazon Neptune

Brad Bebee, Amazon Web Services (AWS)
Leader Product and Engineering, Amazon Neptune
May 12, 2020



A journey through graph...



The Semantic Web
 A new form of Web content that is meaningful to computers will unleash a revolution of new possibilities
 by TIM BERNERS-LEE, JAMES HENDLER and ORA LASSILA

The entertainment system was belting out the Beatles' "We Can Work It Out" when the phone rang. When Pete answered, his phone turned the sound down by sending a message to all the other local devices that had a volume control. His sister, Lucy, was on the line from the doctor's office: "Mom needs to see a specialist and then has to have a series of physical therapy sessions. Biweekly or something. I'm going to have my agent set up the appointments." Pete immediately agreed to share the chauffeur. At the doctor's office, Lucy instructed her Semantic Web agent through her handheld Web browser. The agent promptly retrieved information about Mom's prescribed treatment from the doctor's agent, looked up several lists of providers, and checked for the ones on-plan for Mom's insurance within a 20-mile radius of her home and with a rating of excellent or very good on trusted rating services. It then began trying to find a match between available appointment times (supplied by the agents of individual providers through their Web sites) and Pete's and Lucy's busy schedules. (The emphasized keywords indicate terms whose semantics, or meaning, were defined for the agent through the Semantic Web.)

Sem. Web. "Graph" Applications

Wikidata Query Service

Amazon Neptune General Availability



W3C

Resource Description Framework (RDF) Model and Syntax Specification
 W3C Recommendation 22 February 1999

REC-rdf-syntax-19990222

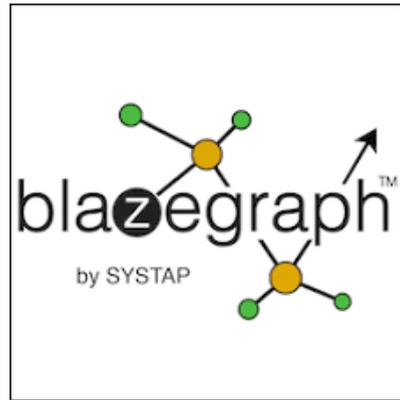
This Version: <http://www.w3.org/TR/1999/REC-rdf-syntax-19990222>
 Newest Version: <http://www.w3.org/TR/REC-rdf-syntax>

Editors:
 Ora Lassila ora.lassila@research.nokia.com, Nokia Research Center
 Ralph R. Swick rswick@w3.org, World Wide Web Consortium

Document Status

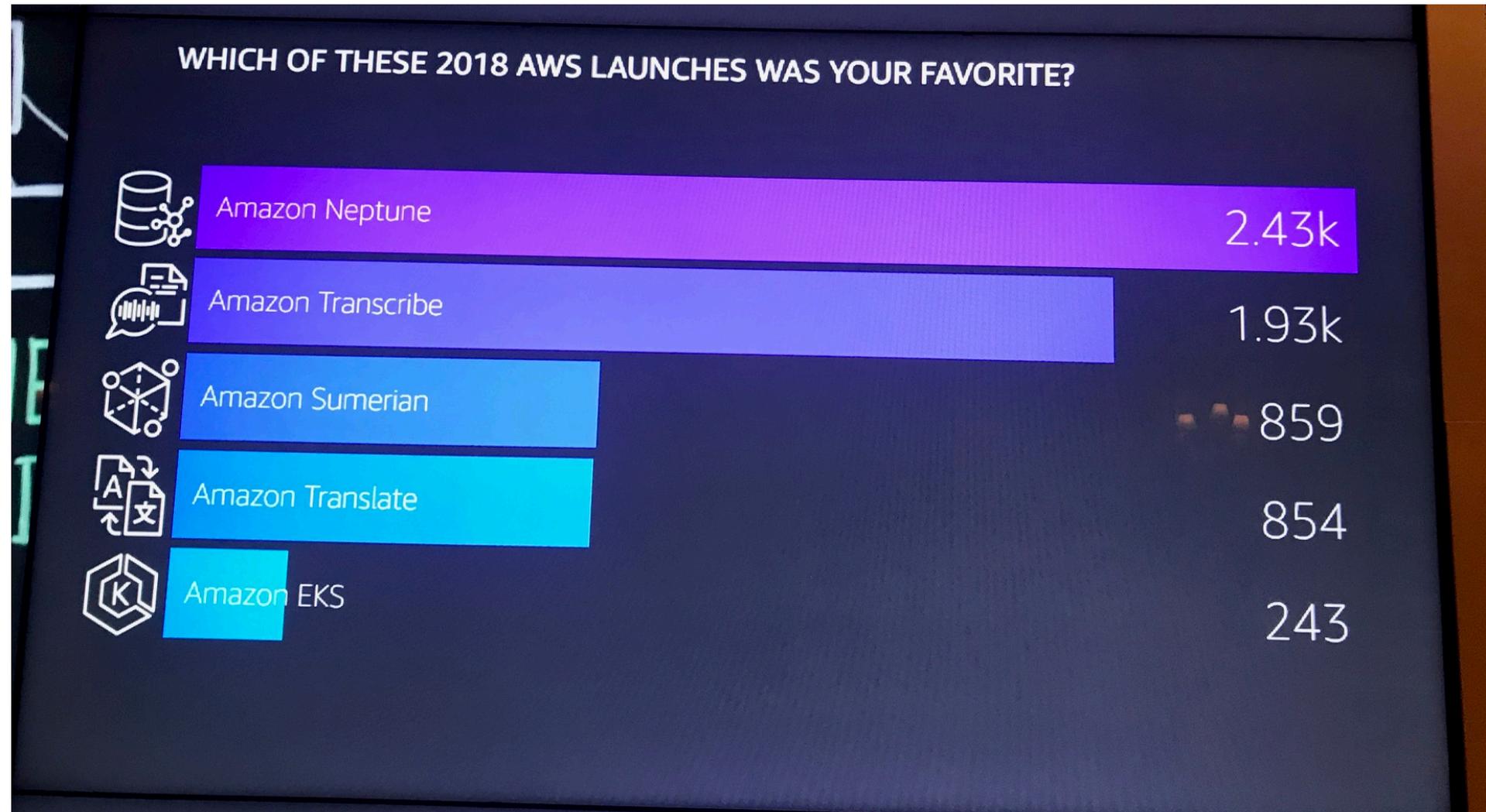
Copyright © 1997,1998,1999 W3C (MIT, INRIA, Keio). All Rights Reserved. W3C liability, trademark, document use and software licensing rules apply.

OWL Web Ontology Language Reference



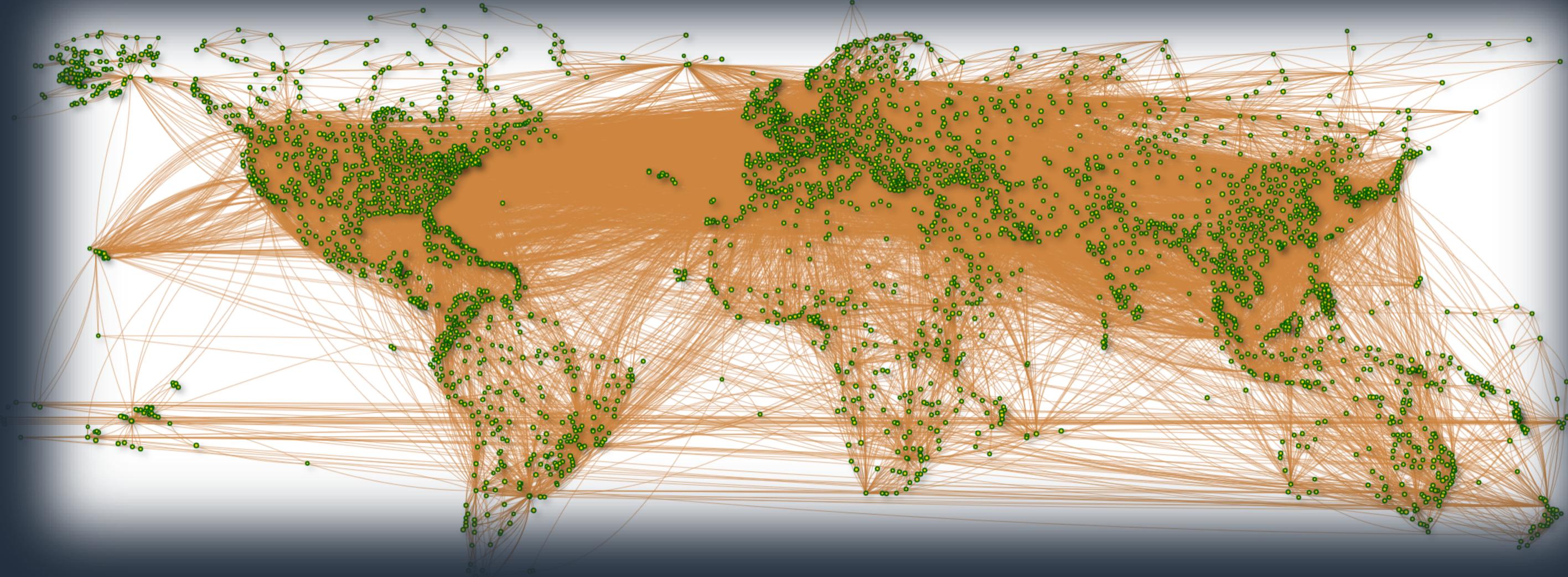
May'2020 Neptune turns two – still day one for graph!

Customers are excited about graph.



AWS re:Invent

Graphs are all around us



Graph use cases



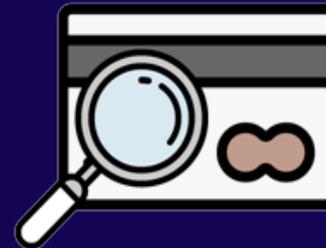
Social
networking



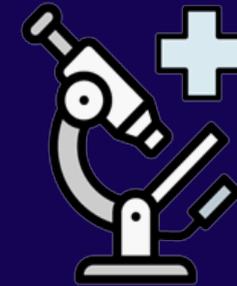
Recommendations



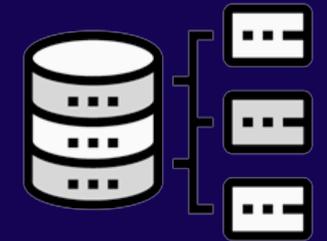
Knowledge
graphs



Fraud
detection



Life
Sciences



Network & IT
operations

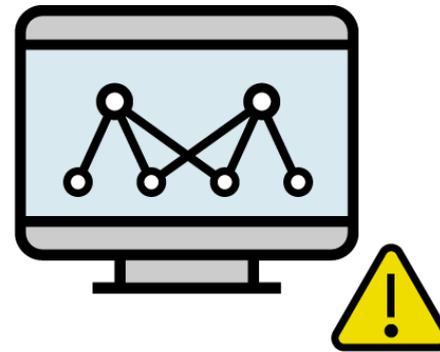
Connected data

- Navigate (variably) connected structure
- Filter or compute a result on the basis of the *strength*, *weight*, or *quality* of relationships

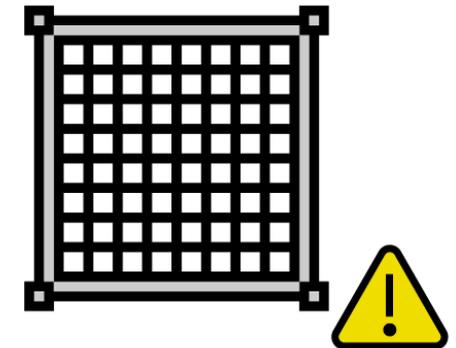
The challenges of building apps with highly connected data using a relational database (or key-value store)



Unnatural for
querying
graph

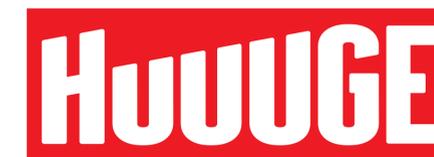


Inefficient
graph
processing



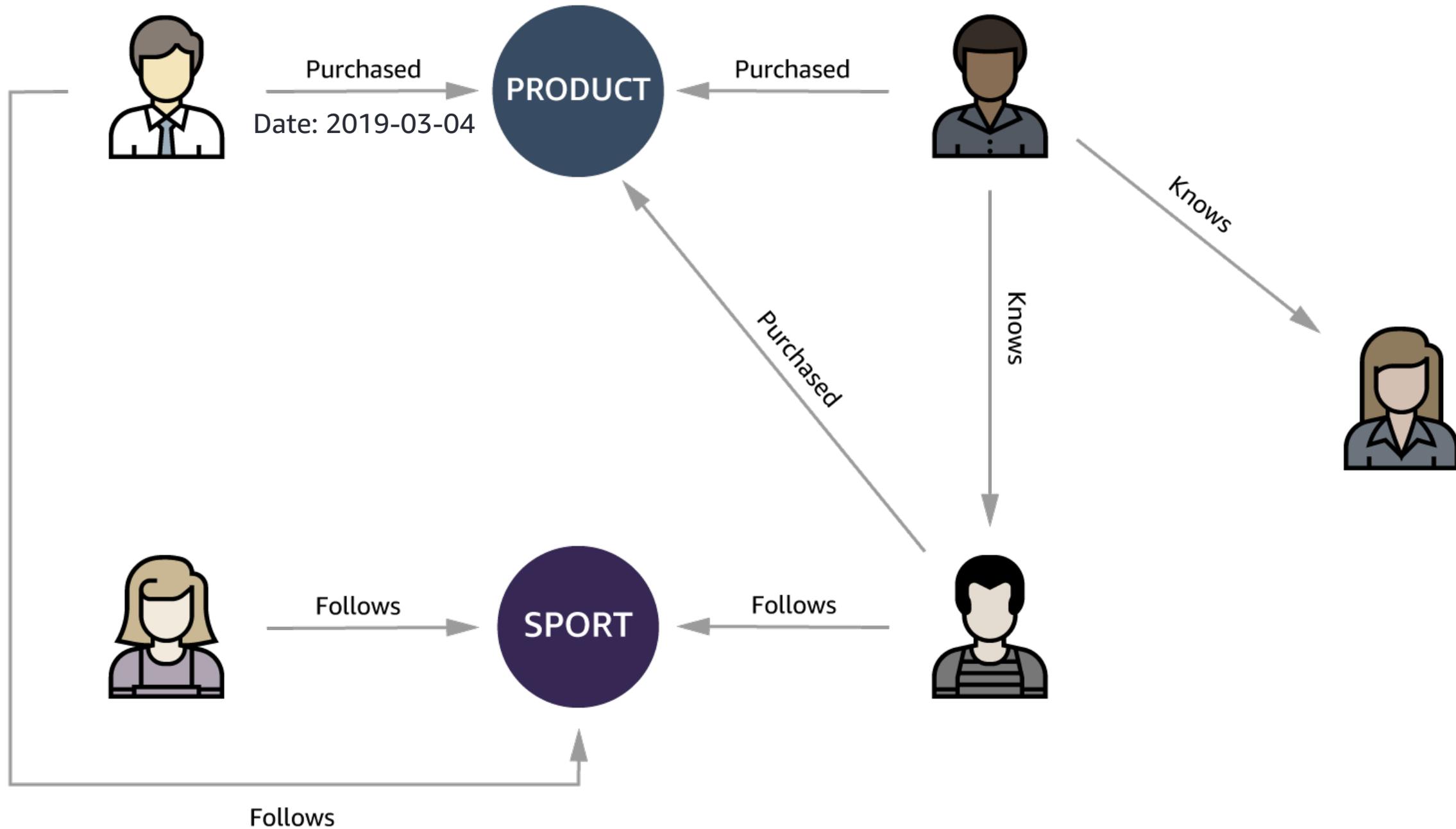
Rigid schema
inflexible
for changing data

Customers using Neptune

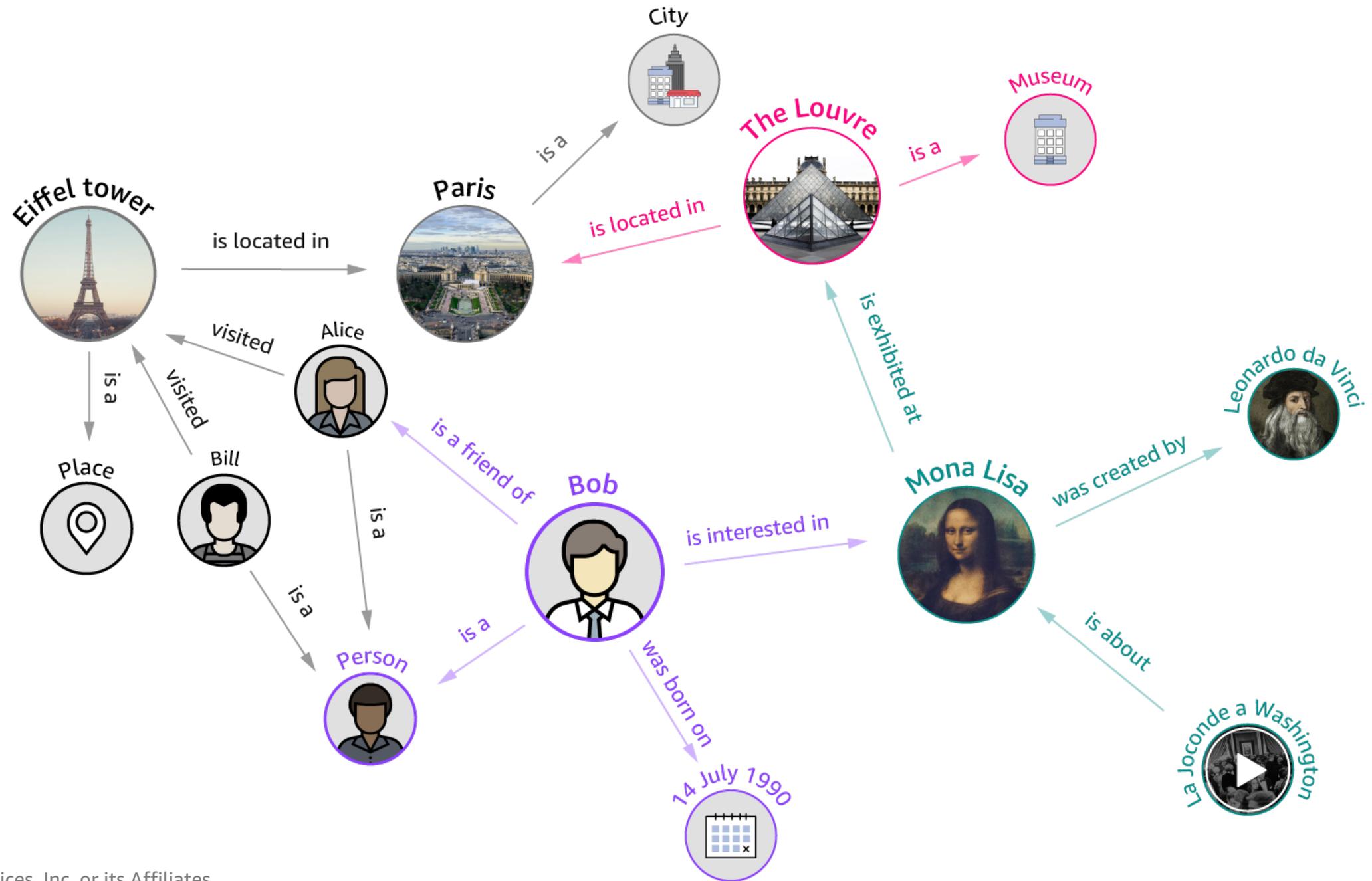


When a customer thinks of a graph

When a customer thinks of a graph



When a customer thinks of a graph



When a customer thinks of a graph

But they're typically not
thinking about...

Graph models and frameworks

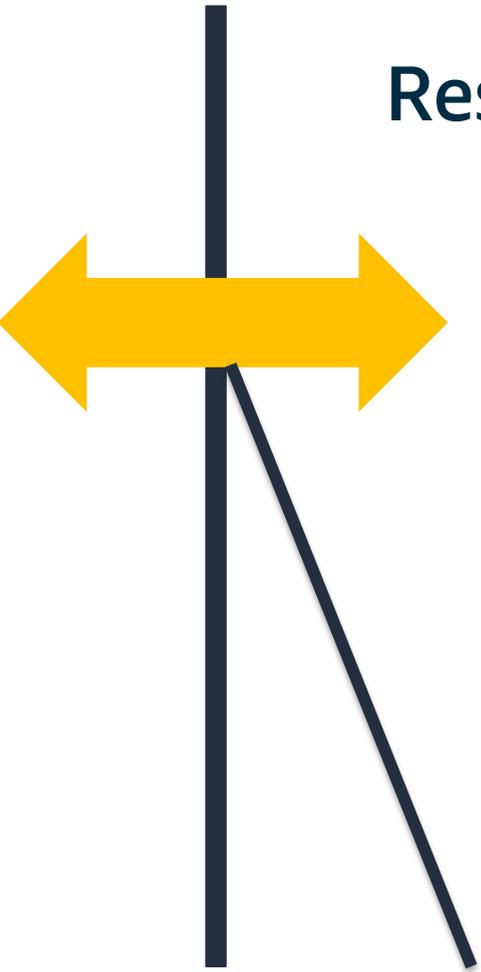
Property graph

Apache TinkerPop Gremlin
openCypher

Property Graph Query Language
(PGQL)
GQL
Others

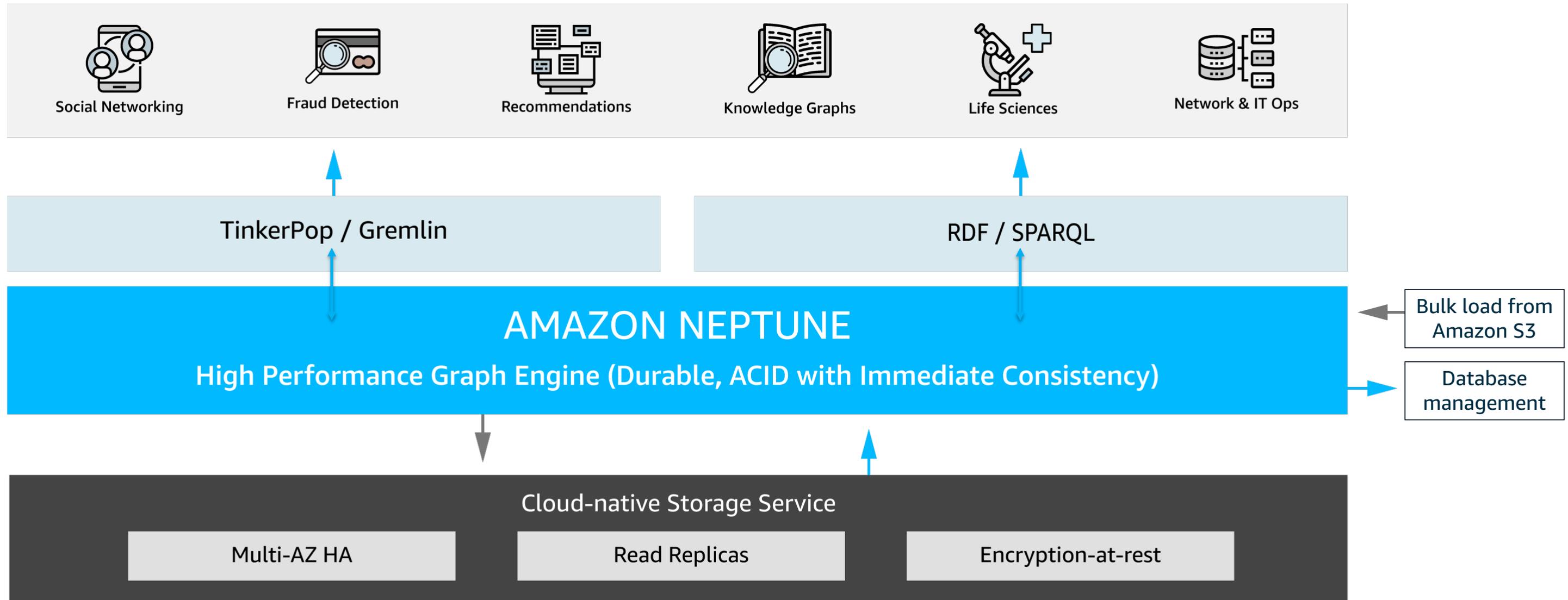
Resource Description Framework (RDF)

W3C standard
SPARQL Query Language



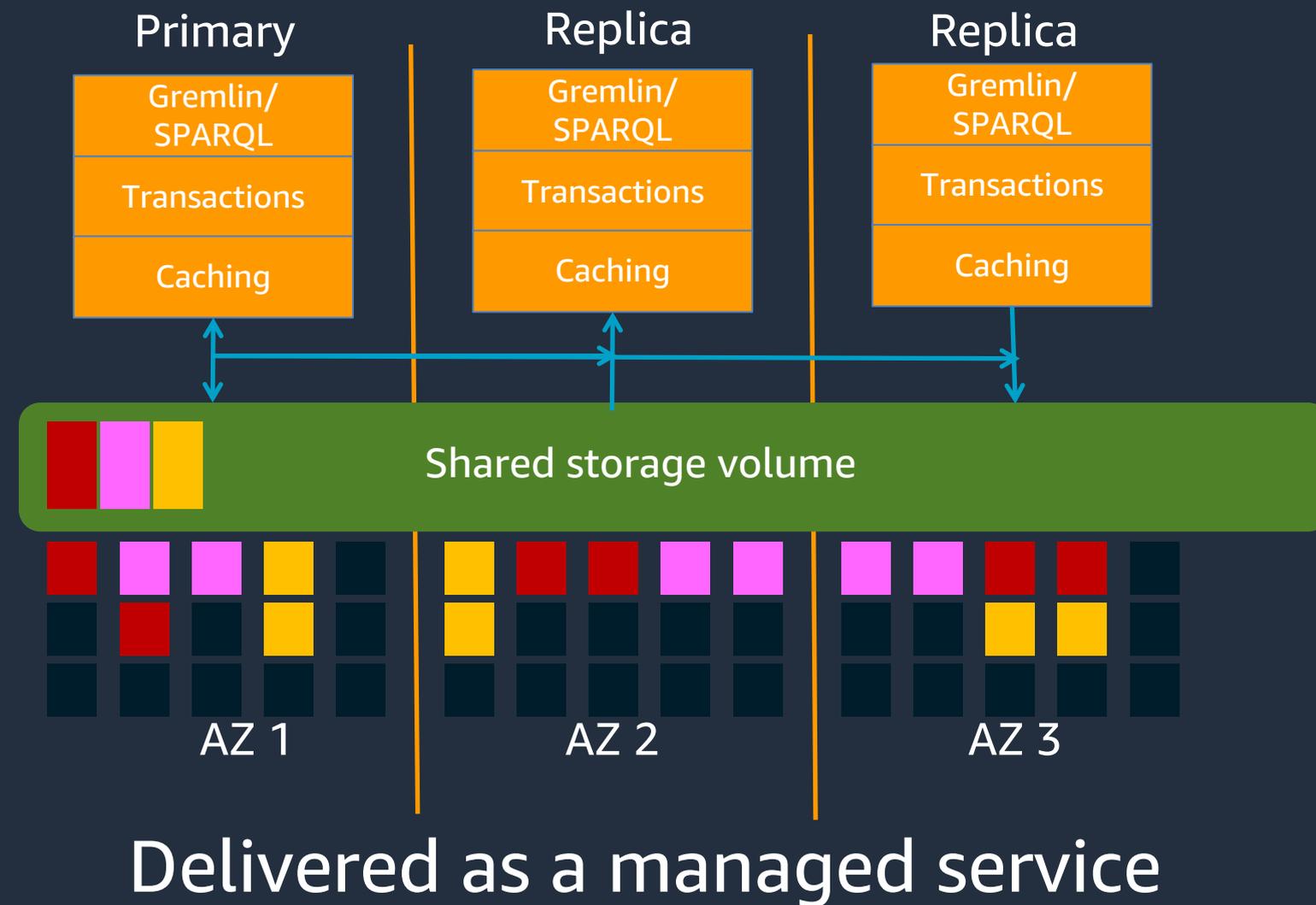
Many customers want both

Supporting both models is important to Amazon Neptune.



Distributed storage architecture

- Performance, availability, durability
- Scale-out replica architecture
- Shared storage volume with 10-GB segments striped across hundreds of nodes
- Data are replicated 6 times across 3 AZs
- Hotspot rebalance, fast database recovery
- Log applicator embedded in storage layer



Read replicas and high availability

Performance

- Applications can scale out read traffic across up to 15 read replicas

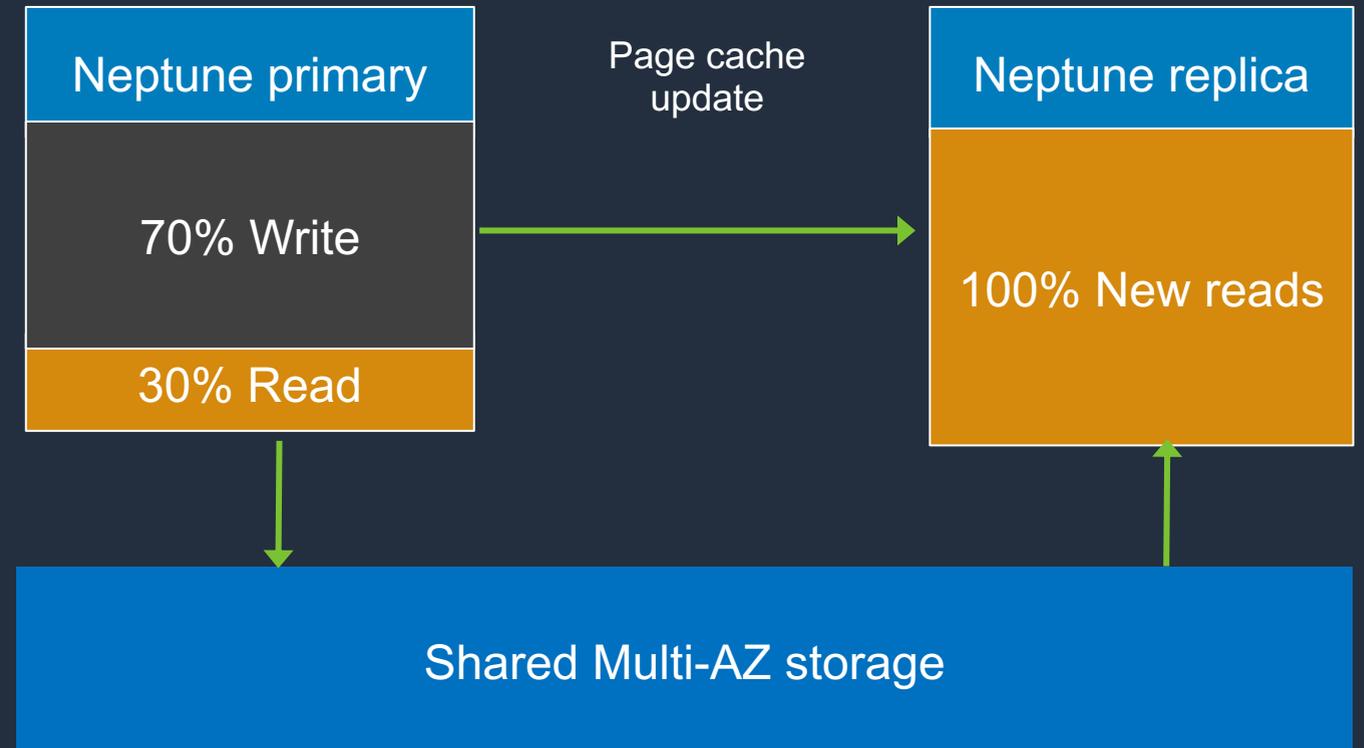
Low replica lag

- Typically <10 ms
- Master ships redo logs to replica
- Cached pages have redo applied
- Uncached pages from shared storage

Availability

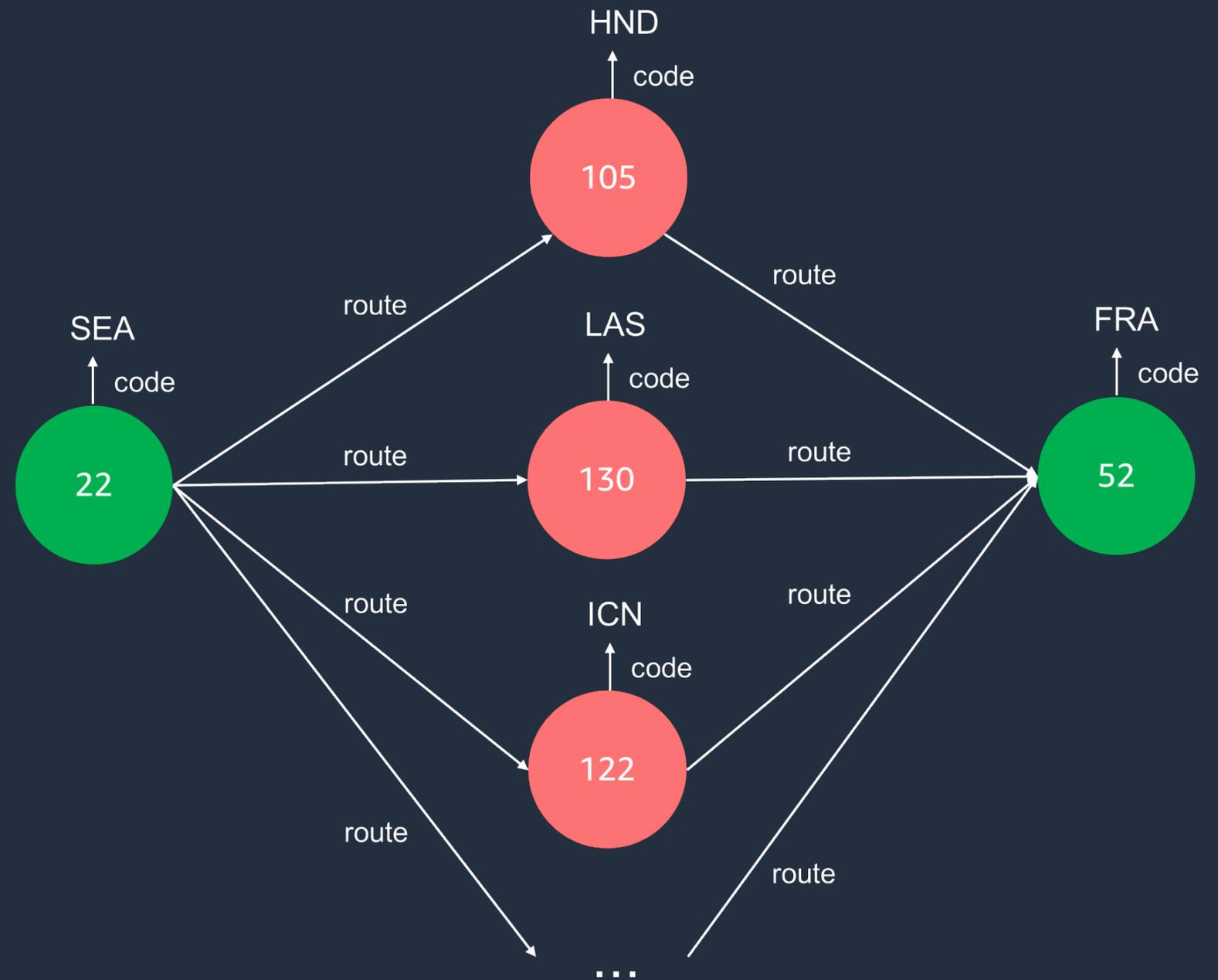
- Failing database nodes are automatically detected and replaced
- If primary fails, a replica replaces it (failover time typically <60 seconds)
- Primary upgrade by forced failover

Amazon Neptune read scaling



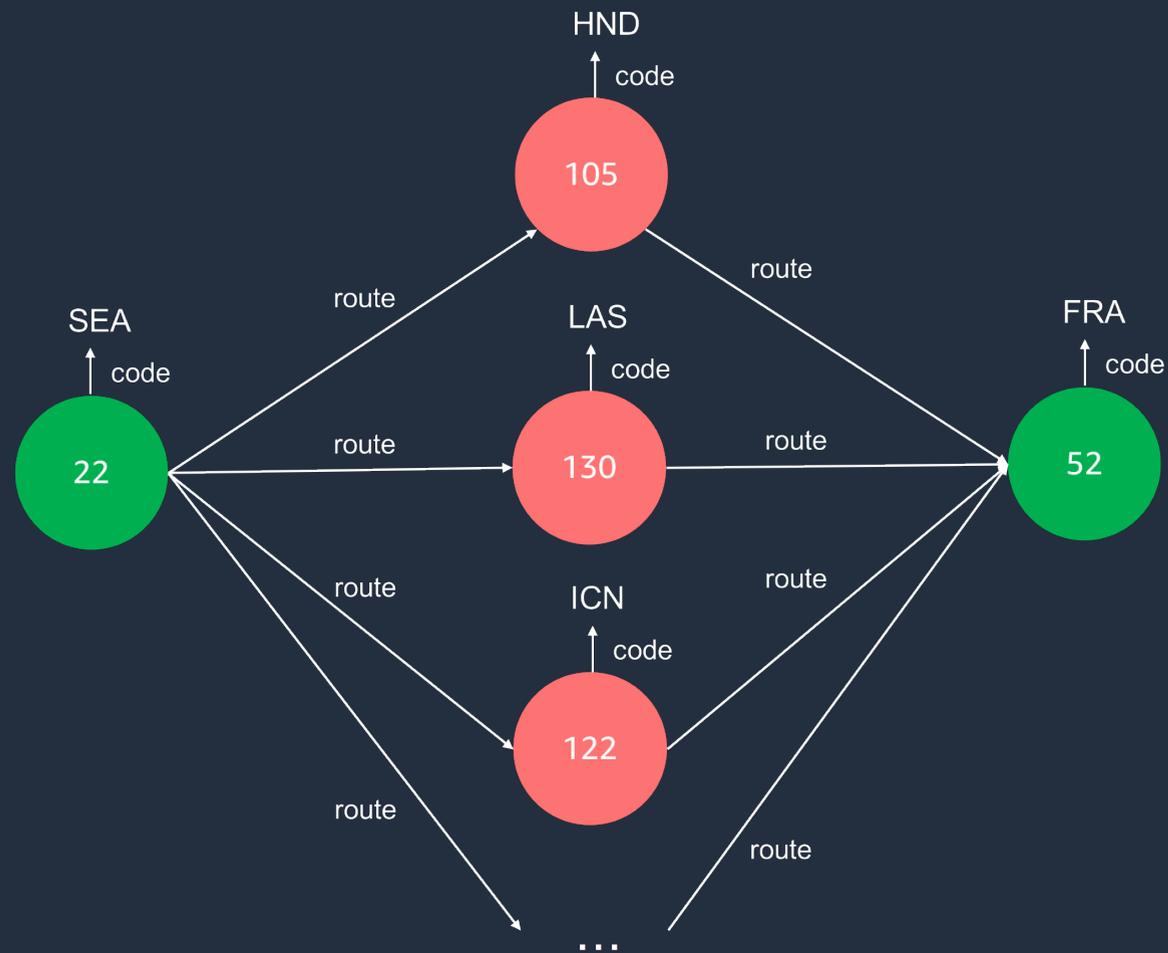
Air routes dataset

- Models the world's airline route network
- Queries operating over the airport connectivity graph
- Example: I can travel from Seattle (SEA) to Frankfurt (FRA) via Tokyo Haneda (HND), Las Vegas (LAS), or Seoul (ICN).



<https://github.com/krlawrence/graph/tree/master/sample-data>

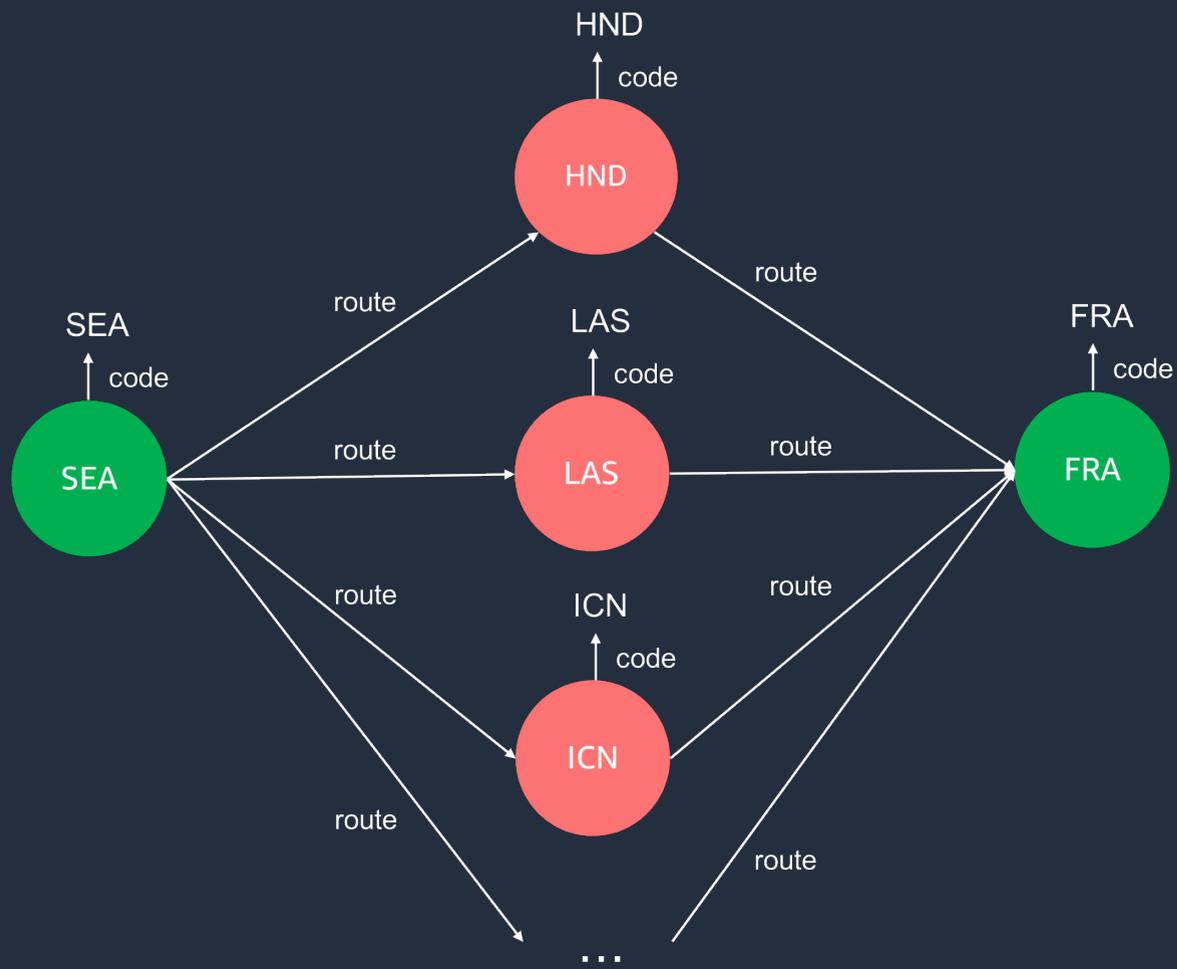
Neptune graph data model



Subject	Predicate	Object	Graph
22	code	"SEA"	default
22	route	105	e1
22	route	130	e2
22	route	122	e3
105	code	"HND"	default
105	route	52	e4
130	code	"LAS"	default
130	route	52	e5
122	code	"ICN"	default
122	route	52	e6
52	code	"FRA"	default

<https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html>

Neptune graph data model

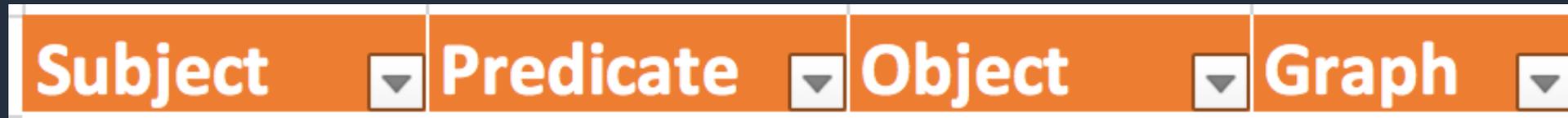


PREFIX resource: <http://kelvinlawrence.net/air-routes/resource/>
PREFIX airport: <http://kelvinlawrence.net/air-routes/resource/airport/>
PREFIX edge: <http://kelvinlawrence.net/air-routes/objectProperty/>
PREFIX prop: <http://kelvinlawrence.net/air-routes/datatypeProperty/>

Subject	Predicate	Object	Graph
airport:SEA	prop:code	"SEA"	default
airport:SEA	edge:route	airport:HND	resource:1
airport:SEA	edge:route	airport:LAS	resource:2
airport:SEA	edge:route	airport:ICN	resource:3
airport:HND	prop:code	"HND"	default
airport:HND	edge:route	airport:FRA	resource:4
airport:LAS	prop:code	"LAS"	default
airport:LAS	edge:route	airport:FRA	resource:5
airport:ICN	prop:code	"ICN"	default
airport:ICN	edge:route	airport:FRA	resource:6
airport:FRA	prop:code	"FRA"	default

<https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html>

Neptune graph data model – indices



SPOG – Uses a key composed of Subject + Predicate + Object + Graph

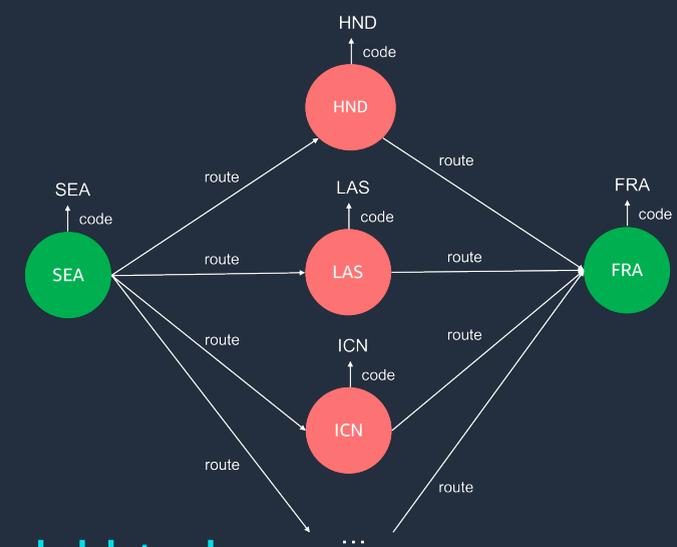
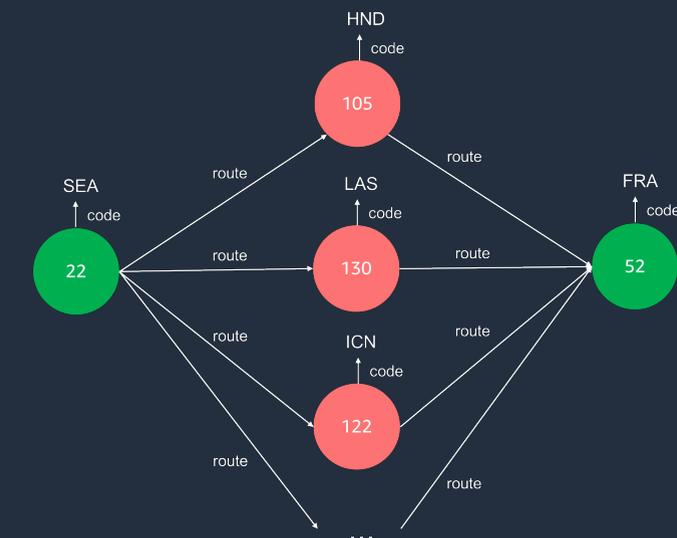
Efficient lookup whenever a prefix of the positions, such as the vertex (subject) or vertex and property identifier, is bound: *Find airport:SEA (22) with code "SEA"*

POGS – Uses a key composed of Predicate + Object + Graph + Subject

Efficient access when only the edge or property label stored in P position is bound: *What nodes have code "SEA"?*

GPSO – Uses a key composed of Graph + Predicate + Subject + Object

Efficient access with the graph (or edge ID) and a property identifier is bound: *What edges are have routes to "FRA"?*

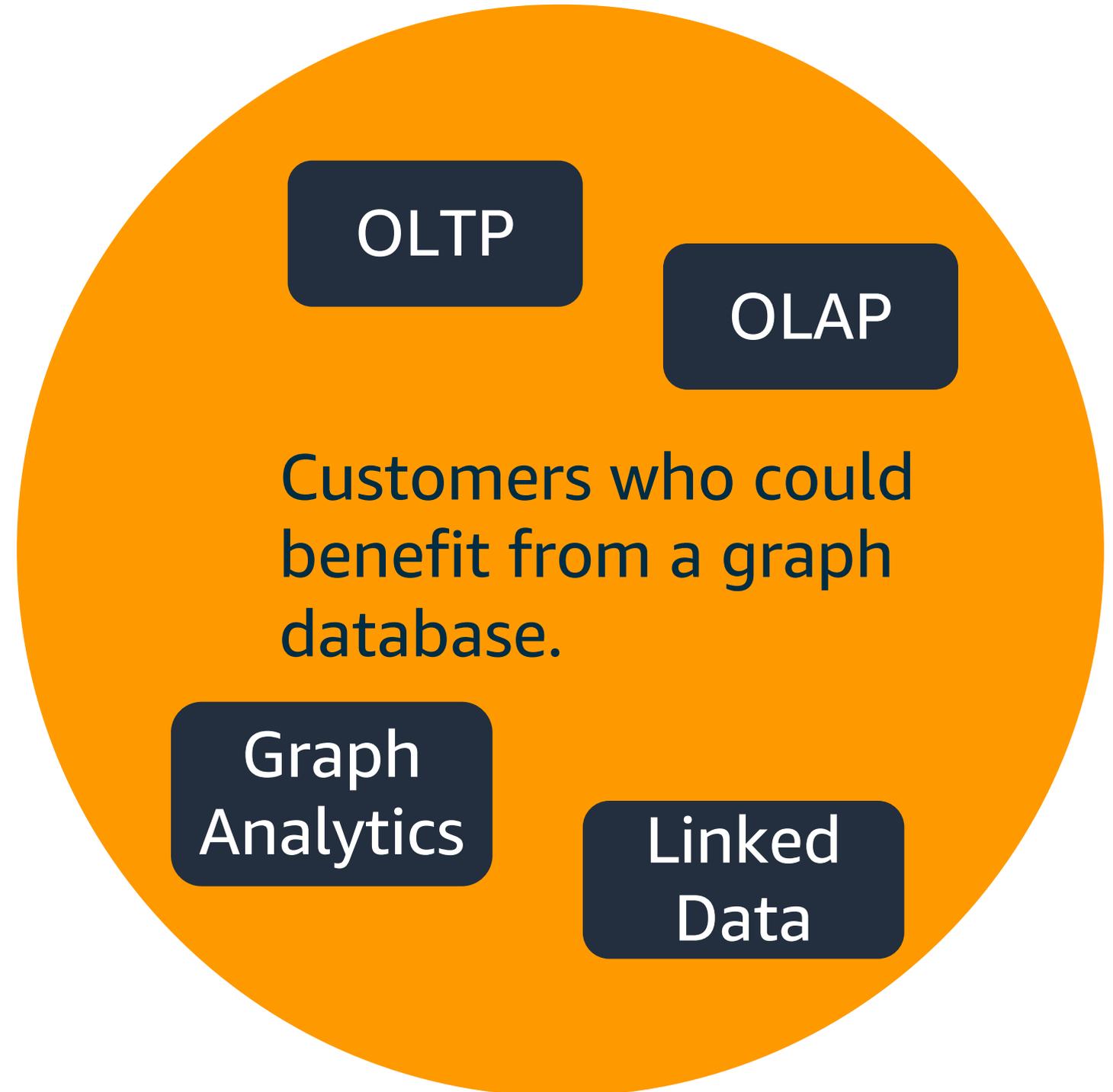


<https://docs.aws.amazon.com/neptune/latest/userguide/feature-overview-data-model.html>

How big is the graph market segment? What is it?



Customers who know they want a graph database.



Customers want graph and have use cases across graph data models.

- Many customers want both PG and RDF.
 - Developer team preference
 - Existing applications
- Semantic alignment / data canonicalization at an organization level (RDF – typically)
 - Business Applications (PG – often)
 - Graph Analytics

Customers also want to...

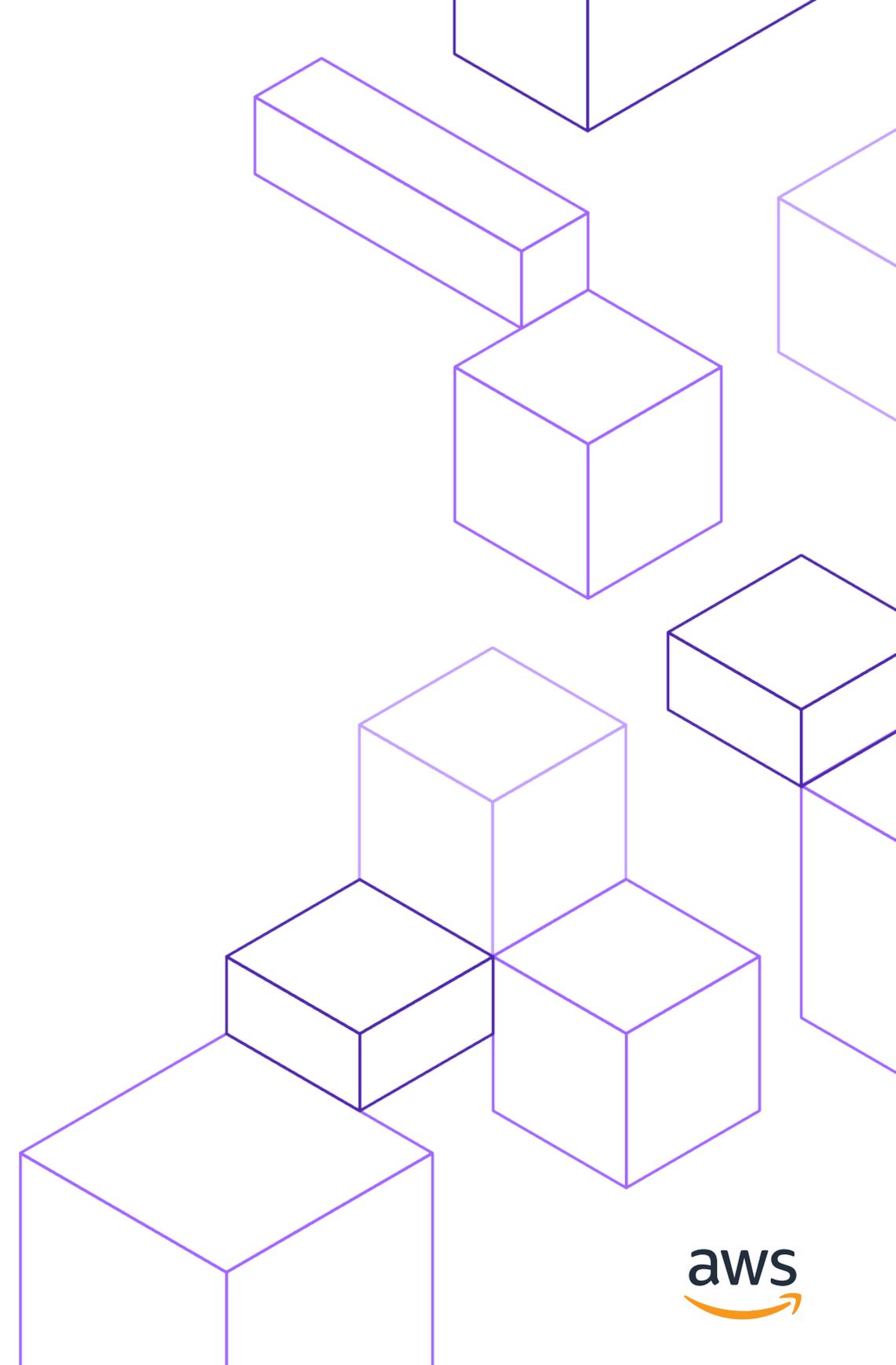
- Understand how to model their data in a graphs
- Query graph data
 - *Which PG query language should I use?*
 - *I can't query my PG data with SPARQL.*
 - *I have RDF Linked Data I want to use in my PG application.*
- Exchange data
 - Between property graph applications
 - RDF->PG and PG->RDF (like RDF* and SPARQL*)
 - Help to conceptualize graphs in RDF

Building approaches that enable commonality enables a larger set of graph users.

- A rising graph floats all nodes, edges, and properties (or resources).
 - Make it easier for customers to model their data as a graphs.
 - Make graph query that works across models more accessible to a broader set developers.
 - Give customers a way to interchange data between PG and RDF.

It's just graph; let's make it that way
for graph application builders.

Q&A



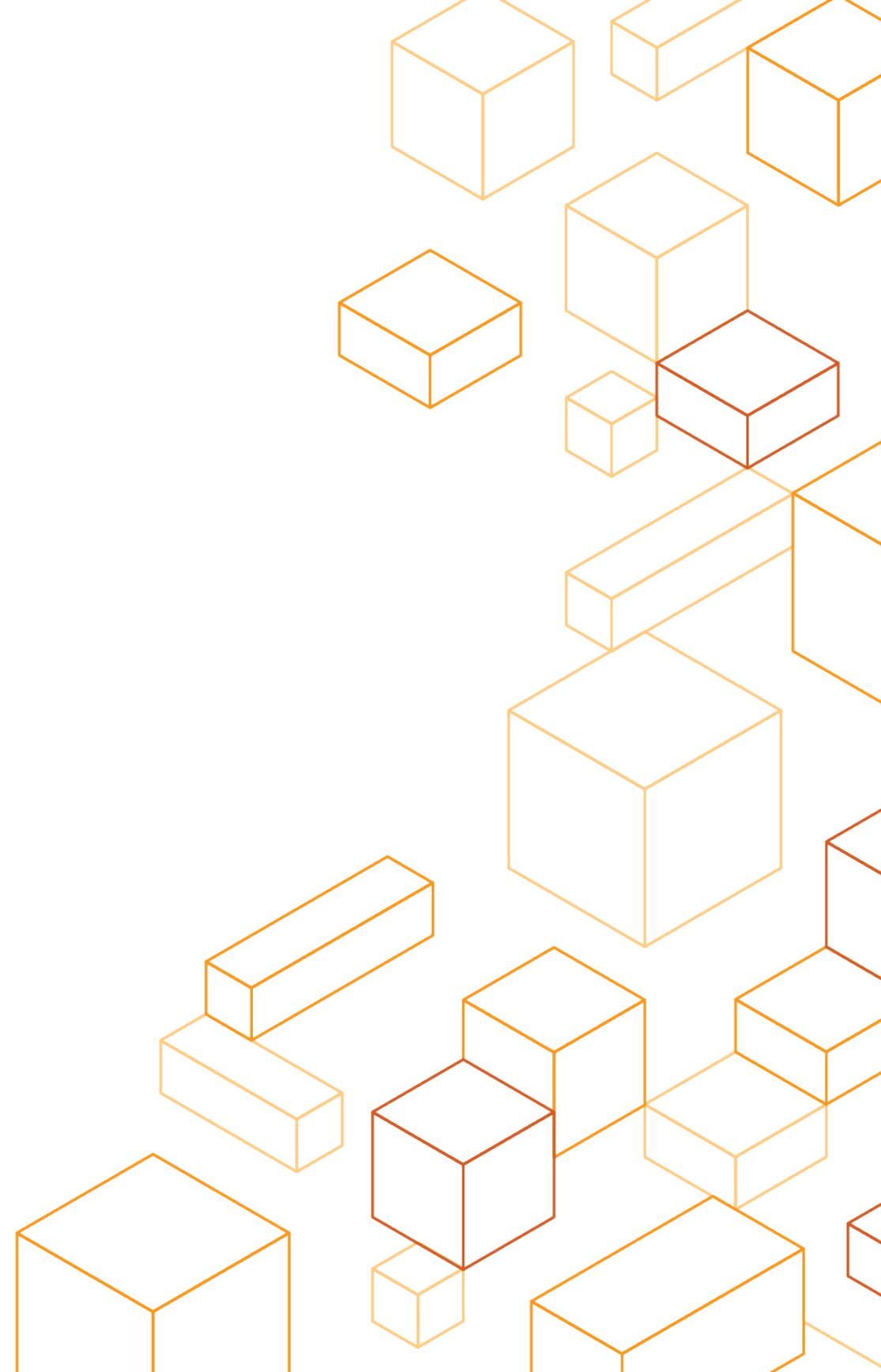


Thank you!

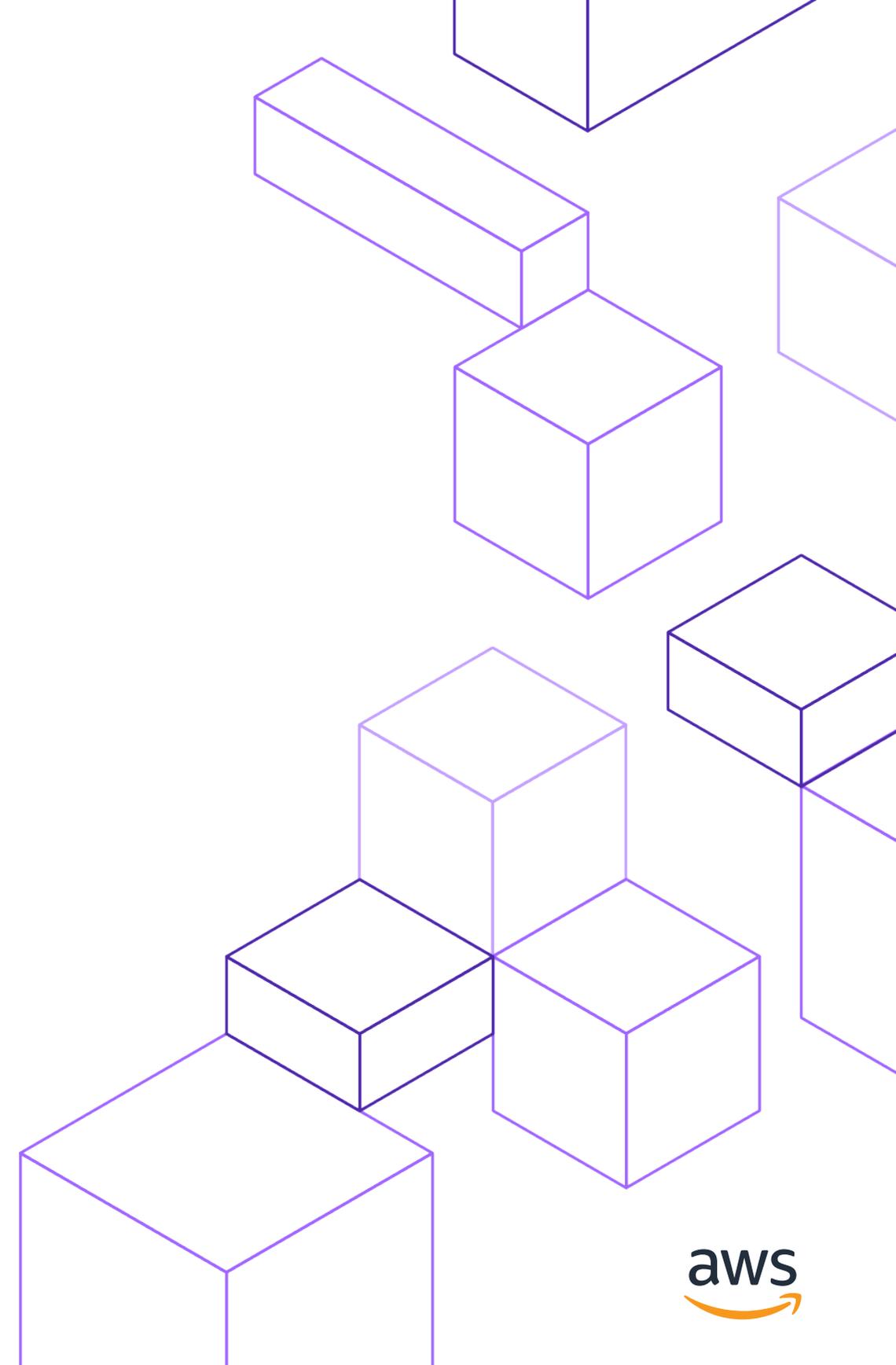
Brad Bebee

beeb@amazon.com

@b2ebs



Resources



Documentation

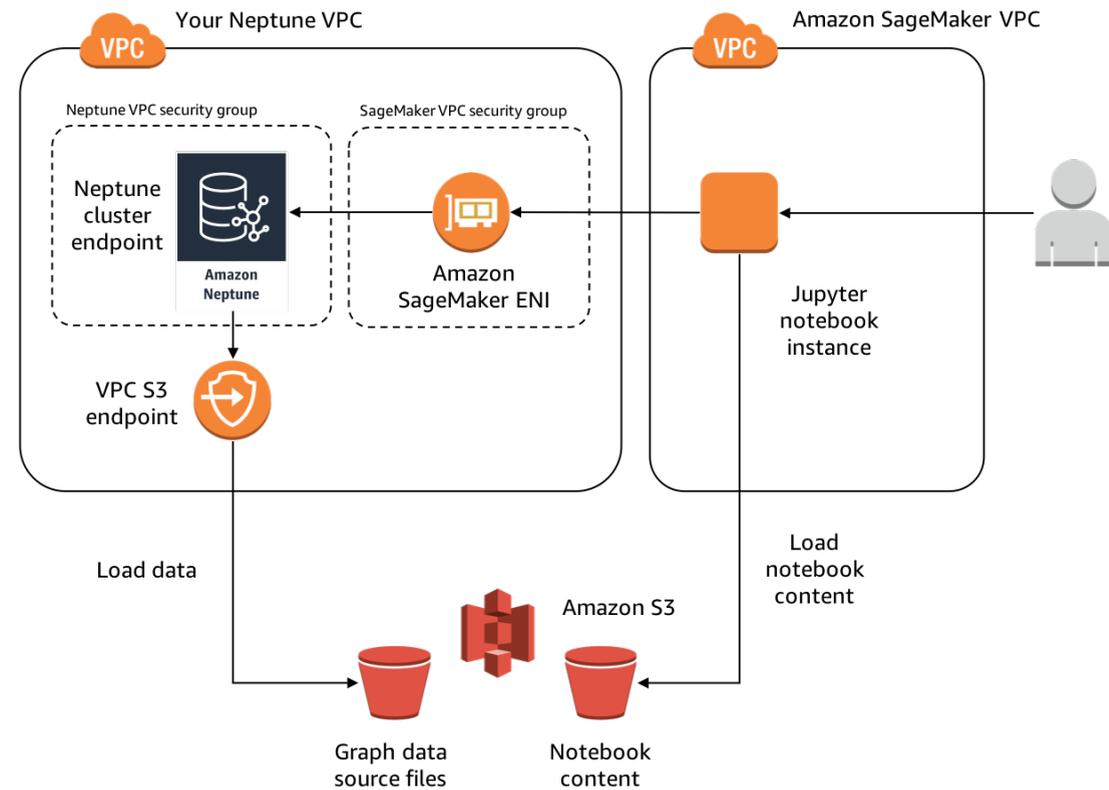
The screenshot shows the AWS Neptune documentation page. The breadcrumb trail is "AWS Documentation » Neptune » User Guide » Overview of Amazon Neptune Features". A callout box at the top states: "The AWS Documentation website is getting a new look! Try it now and let us know what you think. [Switch to the new look >>](#) You can return to the original look by selecting English in the language selector above." The main heading is "Overview of Amazon Neptune Features". The text below reads: "This section provides an overview of Neptune features, including clusters, instances, and storage characteristics of Neptune graphs." A "Note" section follows, stating: "This section does not cover access to the data in a Neptune graph. For information about how to connect to a running Neptune DB cluster with Gremlin, see [Accessing the Neptune Graph with Gremlin](#). For information about how to connect to a running Neptune DB cluster with SPARQL, see [Accessing the Neptune Graph with SPARQL](#)." Below the note is a "Topics" section with a list of links: "What Is a Graph Database?", "Amazon Neptune DB Clusters", "Amazon Neptune Graph Data Model", "Amazon Neptune Storage", "Amazon Neptune Reliability", "High Availability for Neptune", "Connecting to Amazon Neptune Endpoints", "Replication with Amazon Neptune", and "Changes and Updates to Amazon Neptune". The page footer includes "Document Conventions", "« Previous Next »", and a feedback form.

The screenshot shows the AWS Neptune documentation page for "Creating a New Neptune DB Cluster". The breadcrumb trail is "AWS Documentation » Neptune » User Guide » Creating a New Neptune DB Cluster". The main heading is "Using an AWS CloudFormation Stack to Create a Neptune DB Cluster". The text below reads: "You can use an AWS CloudFormation template to set up a Neptune DB Cluster." A numbered list follows: "1. To launch the AWS CloudFormation stack on the AWS CloudFormation console, choose one of the **Launch Stack** buttons in the following table." Below the list is a table with columns "Region", "View", "View in Designer", and "Launch". The table lists various regions and their corresponding "Launch Stack" buttons. The "Launch" column contains a "Launch Stack" button with a dropdown arrow. The numbered list continues: "2. On the **Select Template** page, choose **Next**." "3. On the **Specify Details** page, choose a key pair for the **EC2SSHKeyPairName**." Below the list, it states: "This key pair is required to access the EC2 instance. Ensure that you have the PEM file for". The page footer includes "Terms of Use | Privacy | © 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved.", "Did this page help you? Yes No", and "Feedback".

Region	View	View in Designer	Launch
US East (N. Virginia)	View	View in Designer	Launch Stack
US East (Ohio)	View	View in Designer	Launch Stack
US West (Oregon)	View	View in Designer	Launch Stack
EU (Ireland)	View	View in Designer	Launch Stack
EU (London)	View	View in Designer	Launch Stack
EU (Frankfurt)	View	View in Designer	Launch Stack
Asia Pacific (Singapore)	View	View in Designer	Launch Stack
Asia Pacific (Sydney)	View	View in Designer	Launch Stack
Asia Pacific (Tokyo)	View	View in Designer	Launch Stack
Asia Pacific (Mumbai)	View	View in Designer	Launch Stack
Asia Pacific (Seoul)	View	View in Designer	Launch Stack
EU (Stockholm)	View	View in Designer	Launch Stack
AWS GovCloud (US-West)	View	View in Designer	Launch Stack
AWS GovCloud (US-East)	View	View in Designer	Launch Stack

Explore use cases using Neptune

<https://aws.amazon.com/blogs/database/analyze-amazon-neptune-graphs-using-amazon-sagemaker-jupyter-notebooks/>



The screenshot shows a Jupyter notebook interface with a social network graph and a Gremlin query. The graph displays nodes for Henry, Mary, Gary, Chloe, Colin, Terry, Emily, Gordon, Alistair, Bill, Sarah, and Kate, connected by edges representing friendships. Below the graph, the notebook text discusses using friendship strength to improve recommendations and includes a Gremlin query.

Using Friendship Strength to Improve Recommendations

What if we wanted to base our recommendations only on reasonably strong friendship bonds?

If you look at the Gremlin we used to create our graph, you'll see that each FRIEND edge has a strength property. In the following query, the traversal applies a predicate to this strength property. Note that we use bothE() rather than both() to position the traversal on an edge, where we then apply the predicate. We proceed only where strength is greater than one.

```
In [5]: recommendations = (g.V().hasLabel('User').has('name', user).as('user').
bothE('FRIEND').
has('strength', P.gt(1)).otherV().
aggregate('friends').
bothE('FRIEND').
has('strength', P.gt(1)).otherV().
where(P.neq('user')).where(P.without('friends')).
groupCount().by('name').
order(Scope.local).by(Column.values, Order.decr).
next())

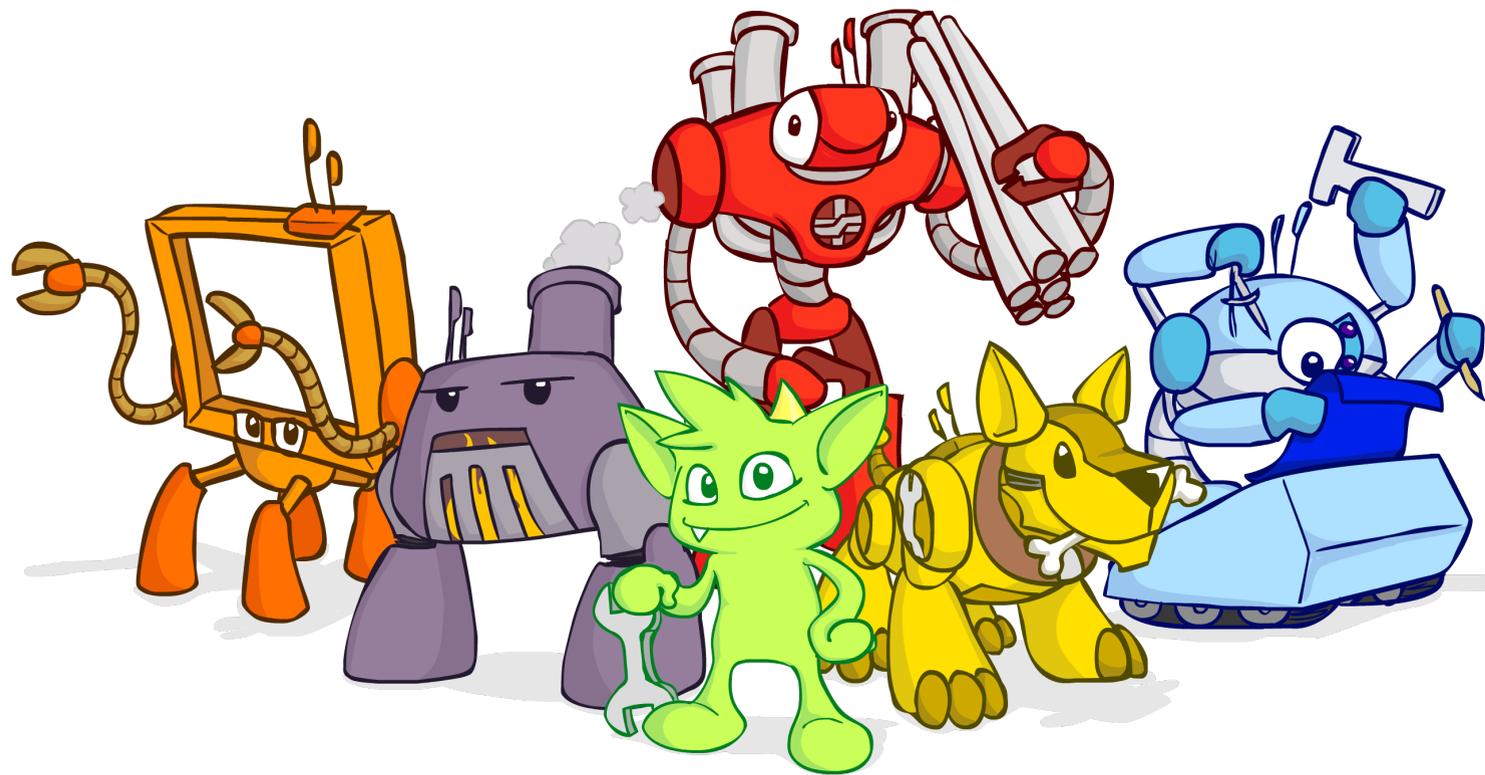
for key in recommendations.keys():
print(key + ': ' + str(recommendations[key]))
```

Colin: 2
Henry: 2
Chloe: 1

Learn Gremlin

<http://kelvinlawrence.net/book/Gremlin-Graph-Guide.html>

<https://github.com/krlawrence/graph>



PRACTICAL GREMLIN: An Apache TinkerPop Tutorial

Kelvin R. Lawrence – gfxman@yahoo.com – Version 282-preview, August 28th 2019

Table of Contents

- 1. INTRODUCTION
 - 1.1. How this book came to be
 - 1.2. Providing feedback
 - 1.3. Some words of thanks
 - 1.4. What is this book about?
 - 1.5. Introducing the book sources, sample programs and data
 - 1.6. A word about TinkerPop 3.4
 - 1.7. So what is a graph database and why should I care?
 - 1.8. A word about terminology
- 2. GETTING STARTED
 - 2.1. What is Apache TinkerPop?
 - 2.2.1. Downloading, installing and launching the console
 - 2.2.2. Saving output from the console to a file
 - 2.2. The Gremlin console
 - 2.3. Introducing TinkerGraph
 - 2.4. Introducing the air-routes graph
 - 2.4.1. Updated versions of the air-route data
 - 2.5. TinkerPop 3 migration notes
 - 2.5.1. Creating a TinkerGraph TP2 vs TP3
 - 2.5.2. Loading a graphML file TP2 vs TP3
 - 2.5.3. A word about the TinkerPop.sugar plugin

Learn SPARQL

https://logd.tw.rpi.edu/tutorial/a_crash_course_on_sparql

<https://www.w3.org/TR/sparql11-query/>

A screenshot of a web browser displaying the W3C SPARQL 1.1 Query Language page. The browser's address bar shows the URL 'w3.org/TR/sparql11-query/'. The page content includes the W3C logo, the title 'SPARQL 1.1 Query Language', and the date 'W3C Recommendation 21 March 2013'. It lists the current version, latest version, and previous version with their respective URLs. The editors are listed as Steve Harris, Garlik, and Andy Seaborne. The previous editor is Eric Prud'hommeaux. A note mentions referring to the errata for normative corrections. The page also includes a copyright notice for 2013 and an abstract section starting with 'RDF is a directed, labeled graph data format for representing information in the Web.'

W3C SPARQL 1.1 Query Language

W3C Recommendation 21 March 2013

This version:
<http://www.w3.org/TR/2013/REC-sparql11-query-20130321/>

Latest version:
<http://www.w3.org/TR/sparql11-query/>

Previous version:
<http://www.w3.org/TR/2012/PR-sparql11-query-20121108/>

Editors:
Steve Harris, Garlik, a part of Experian
Andy Seaborne, The Apache Software Foundation

Previous Editor:
Eric Prud'hommeaux, W3C

Please refer to the [errata](#) for this document, which may include some normative corrections.

See also [translations](#).

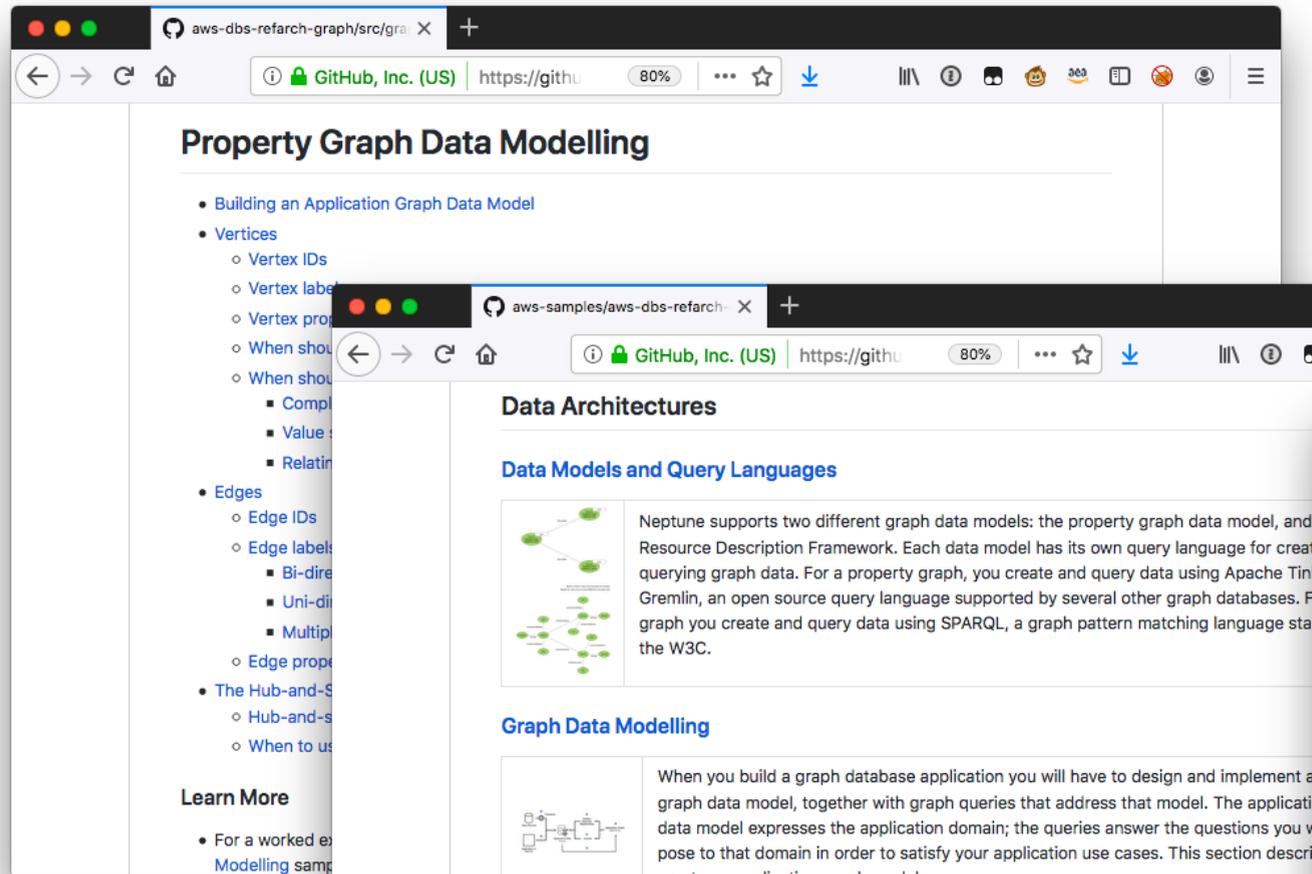
Copyright © 2013 W3C® (MIT, ERCIM, Keio, Beihang), All Rights Reserved. W3C [liability](#), [trademark](#) and [document use](#) rules apply.

Abstract

RDF is a directed, labeled graph data format for representing information in the Web. This specification defines the syntax and semantics of the SPARQL query language for RDF. SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware. SPARQL contains capabilities for querying required and optional graph patterns along

Reference architectures

<https://github.com/aws-samples/aws-dbs-refarch-graph/>

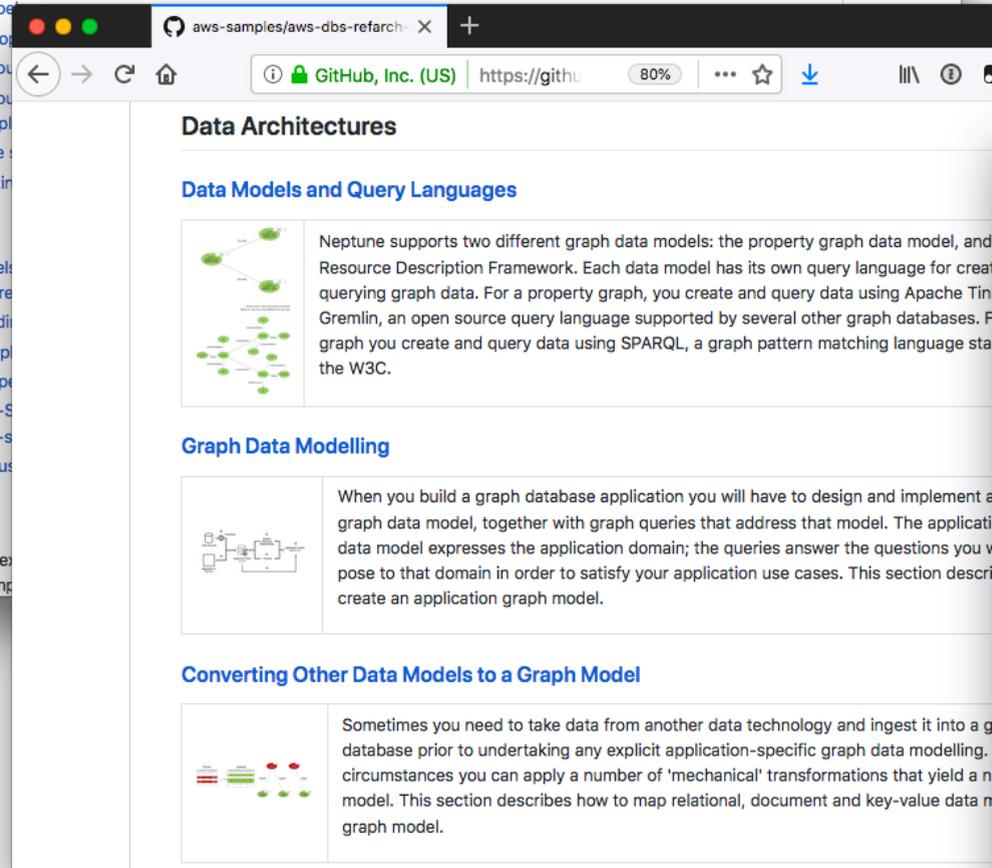


Property Graph Data Modelling

- Building an Application Graph Data Model
- Vertices
 - Vertex IDs
 - Vertex labels
 - Vertex properties
 - When should I use a vertex ID?
 - When should I use a vertex label?
- Edges
 - Edge IDs
 - Edge labels
 - Bi-directional
 - Unidirectional
 - Multiplexed
 - Edge properties
- The Hub-and-Spoke Model
 - Hub-and-Spoke
 - When to use a Hub-and-Spoke

Learn More

- For a worked example, see the [Property Graph Data Modelling sample](#).



Data Architectures

Data Models and Query Languages

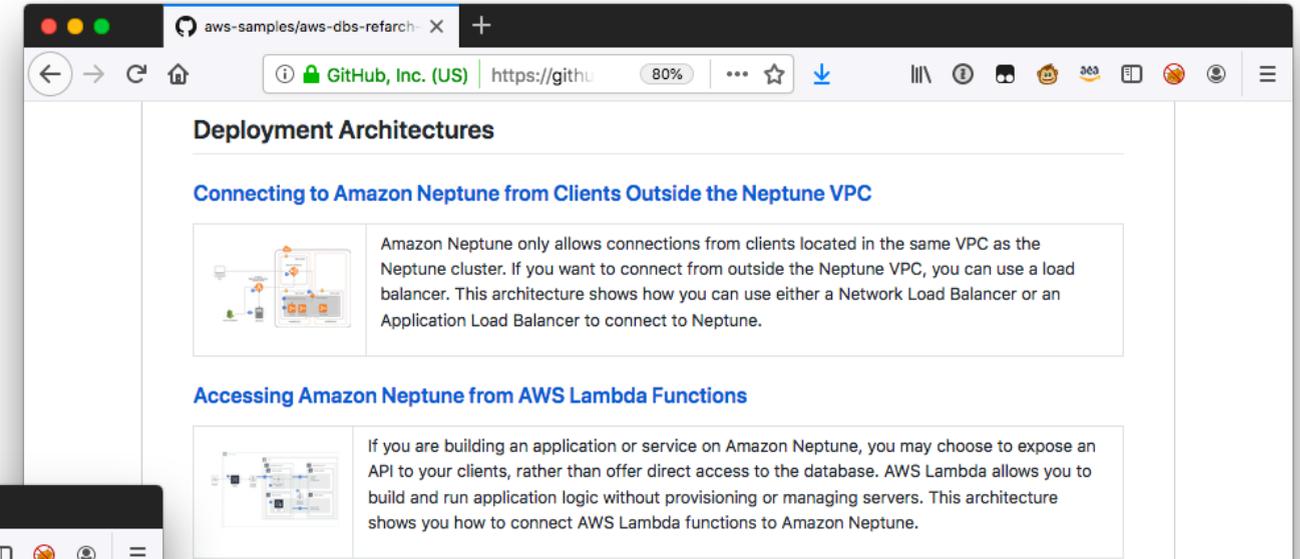
Neptune supports two different graph data models: the property graph data model, and the Resource Description Framework. Each data model has its own query language for creating and querying graph data. For a property graph, you create and query data using Apache TinkerPop Gremlin, an open source query language supported by several other graph databases. For a graph you create and query data using SPARQL, a graph pattern matching language standard in the W3C.

Graph Data Modelling

When you build a graph database application you will have to design and implement an application graph data model, together with graph queries that address that model. The application graph data model expresses the application domain; the queries answer the questions you have about that domain in order to satisfy your application use cases. This section describes how to create an application graph model.

Converting Other Data Models to a Graph Model

Sometimes you need to take data from another data technology and ingest it into a graph database prior to undertaking any explicit application-specific graph data modelling. In some circumstances you can apply a number of 'mechanical' transformations that yield a new graph model. This section describes how to map relational, document and key-value data models to a graph model.



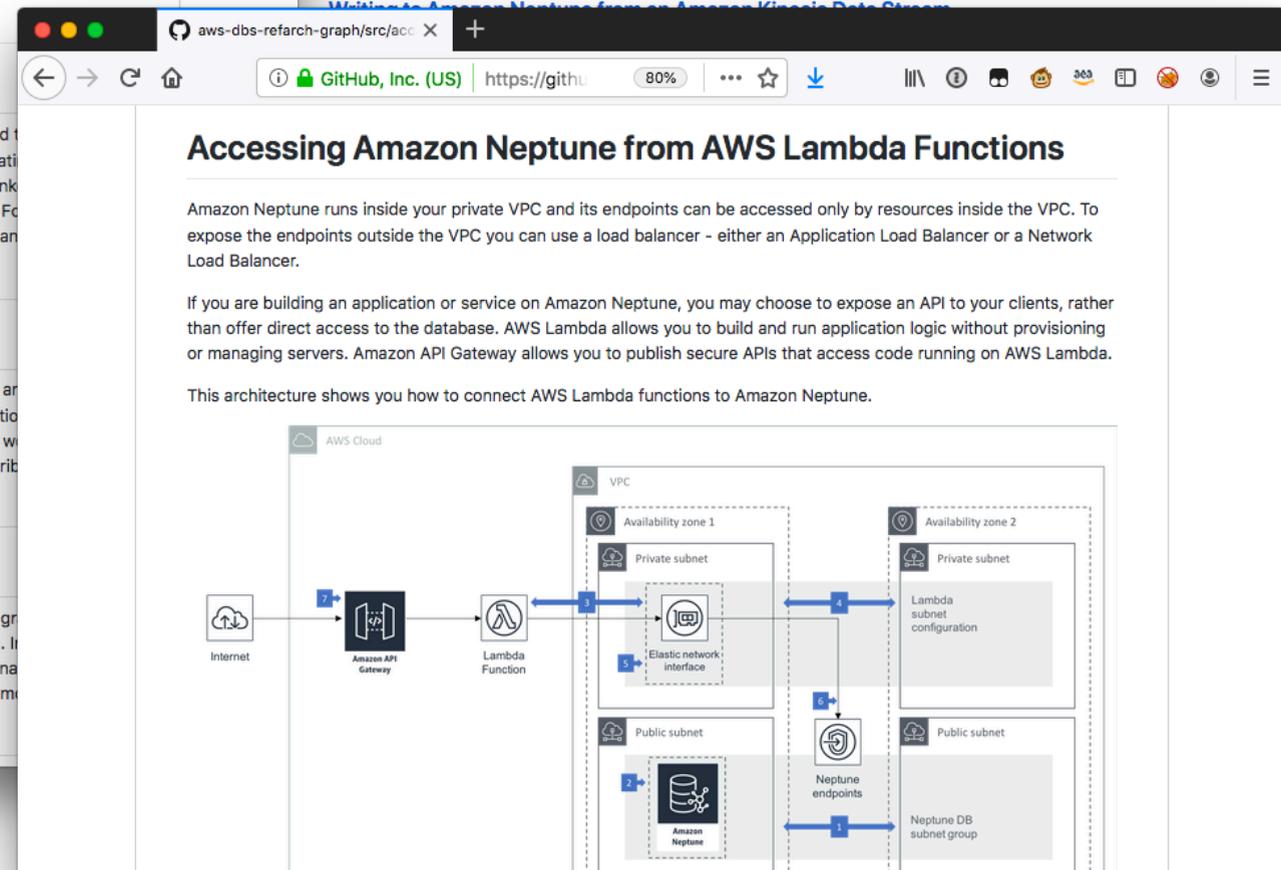
Deployment Architectures

Connecting to Amazon Neptune from Clients Outside the Neptune VPC

Amazon Neptune only allows connections from clients located in the same VPC as the Neptune cluster. If you want to connect from outside the Neptune VPC, you can use a load balancer. This architecture shows how you can use either a Network Load Balancer or an Application Load Balancer to connect to Neptune.

Accessing Amazon Neptune from AWS Lambda Functions

If you are building an application or service on Amazon Neptune, you may choose to expose an API to your clients, rather than offer direct access to the database. AWS Lambda allows you to build and run application logic without provisioning or managing servers. This architecture shows you how to connect AWS Lambda functions to Amazon Neptune.

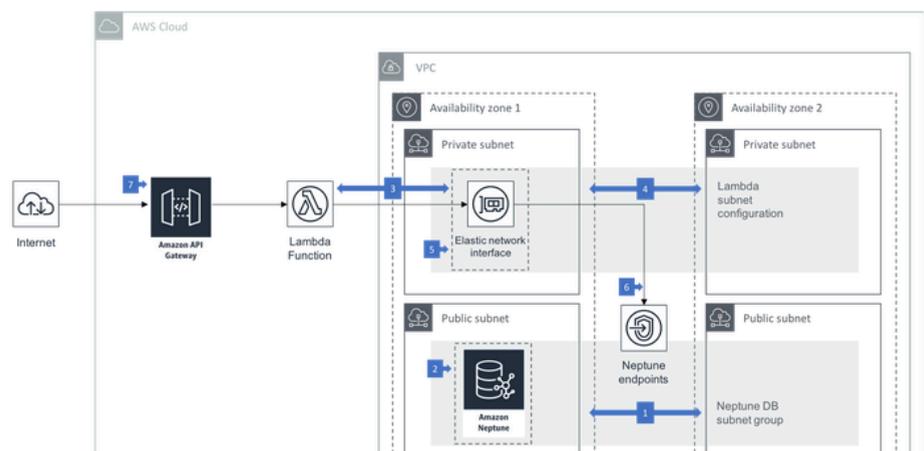


Accessing Amazon Neptune from AWS Lambda Functions

Amazon Neptune runs inside your private VPC and its endpoints can be accessed only by resources inside the VPC. To expose the endpoints outside the VPC you can use a load balancer - either an Application Load Balancer or a Network Load Balancer.

If you are building an application or service on Amazon Neptune, you may choose to expose an API to your clients, rather than offer direct access to the database. AWS Lambda allows you to build and run application logic without provisioning or managing servers. Amazon API Gateway allows you to publish secure APIs that access code running on AWS Lambda.

This architecture shows you how to connect AWS Lambda functions to Amazon Neptune.



The diagram illustrates the architecture for accessing Amazon Neptune from AWS Lambda functions. It shows the following components and their interactions:

- Internet:** External traffic enters from the Internet.
- Amazon API Gateway:** Acts as the entry point for external clients.
- Lambda Function:** Receives requests from the API Gateway and processes them.
- VPC (Virtual Private Cloud):** Contains the database and application resources.
 - Private subnet:** Contains the **Elastic network interface** and **Lambda subnet configuration**.
 - Public subnet:** Contains the **Amazon Neptune** instance and **Neptune endpoints**.
 - Neptune DB subnet group:** A group of subnets for the Neptune database instance.

Numbered steps (1-6) indicate the flow of traffic and data between these components.

Samples

<https://github.com/aws-samples/amazon-neptune-samples>

Setup

Install the components using CloudFormation:

Region	Stack
US East (N. Virginia)	Launch Stack
US East (Ohio)	Launch Stack
US West (Oregon)	Launch Stack
EU (Ireland)	Launch Stack
EU (London)	Launch Stack
EU (Frankfurt)	Launch Stack

You'll need to supply an `EC2KeyName` so that you can also specify a number of other parameters:

Parameter	
<code>DbInstanceType</code>	Neptune database instance type
<code>ShardCount</code>	Number of shards in the Kinesis Data Streams (default 1). The number of shards is multiplied by the number of vCPUs on the EC2 instance to create 64 shards. The number of shards is the number of functions writing batches to Neptune.
<code>BatchReadSize</code>	Number of records read from a batch (default 100). The number of records is the number of records written to Neptune.

Neptune/Getting-Started/ data-model-3

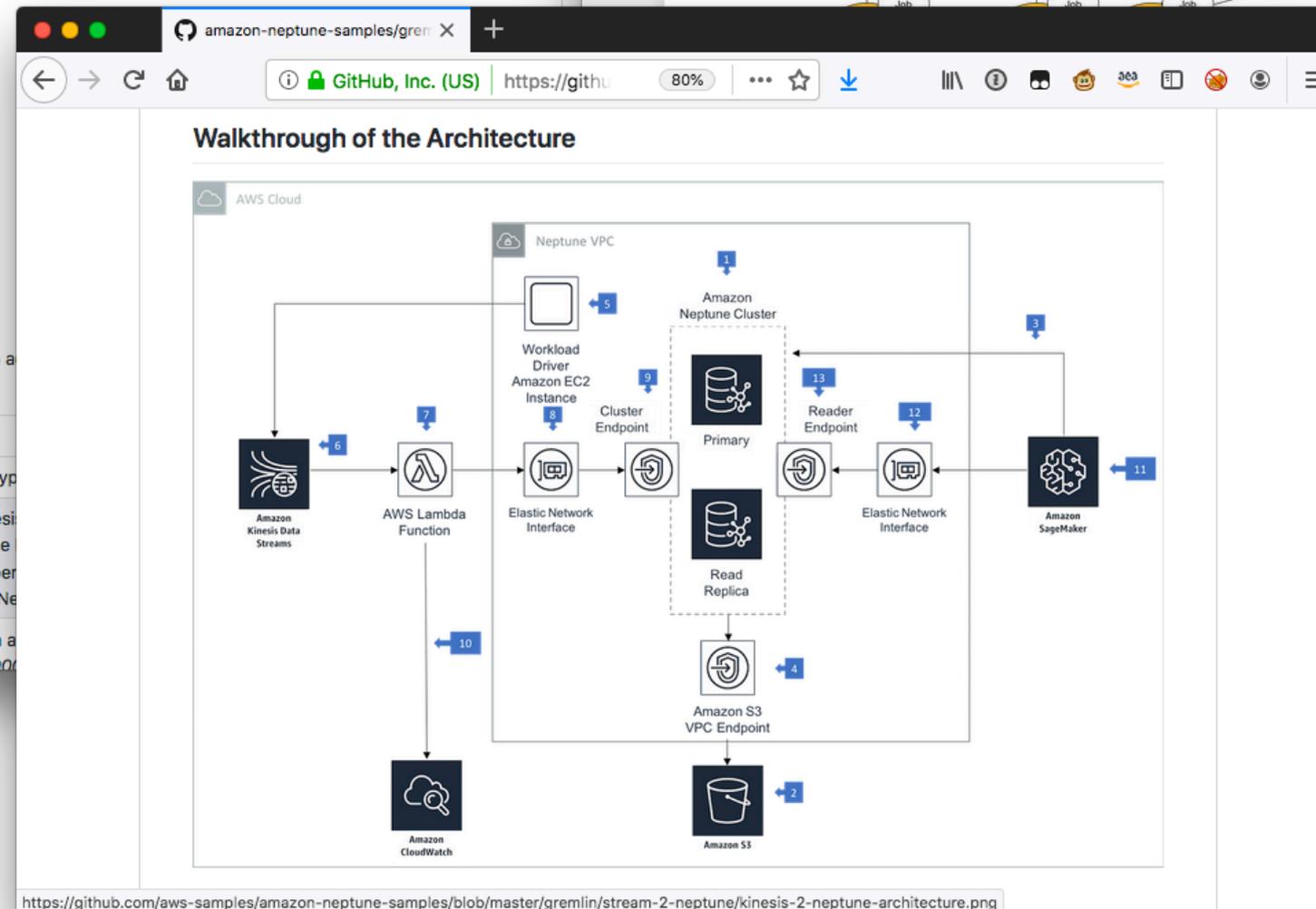
https://neptunenotebookinstan... 90%

Jupyter data-model-3 Last Checkpoint: a few seconds ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted conda_python3

Data model

ies where Li worked?



Use cases, videos, blog posts, and code

<https://aws.amazon.com/neptune/developer-resources/>

