



W E B I N A R

Boost ML-Powered Recommendation Engine & Fraud Detection Workloads

Kevin Phillips (he/him)

Amazon Neptune Specialist Solutions
Architect

What's On Today?

- Introducing Graph Databases
- Amazon Neptune
- Machine Learning on Graph
- Neptune ML
- Using Neptune for Fraud Detection
 - Demo: Performing real-time inference
- Summary
- Resources

Introducing Graph Databases



What is a graph?

Graphs are purpose-built to store and navigate relationships

Nodes represent real-world objects

Edges store relationships between objects

Properties and labels can be added to both nodes and edges



Technical challenges that graphs help solve



Combining data
across data silos



Finding common
connections or
paths

Technical challenges that graphs help solve



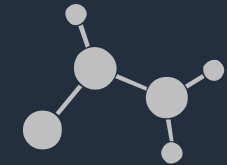
Combining data
across data silos



Finding common
connections or
paths



Working with
heterogeneous data
with complex
relationships



Data full of many-
to-many
relationships

Amazon Neptune



Amazon Neptune

Open



**Supports Apache
TinkerPop & W3C RDF
graph models**

Fast



**Query billions of
relationships with
millisecond latency**

Reliable



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

Flexible



**Build powerful queries
easily with Gremlin,
openCypher, and
SPARQL**

(Some) Service integrations



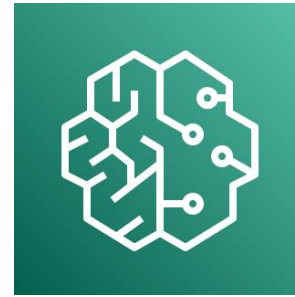
AWS OpenSearch Service



AWS Athena



AWS Glue



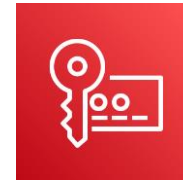
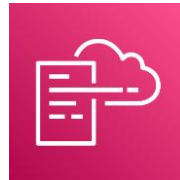
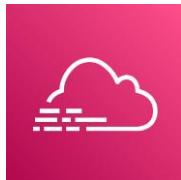
Amazon SageMaker



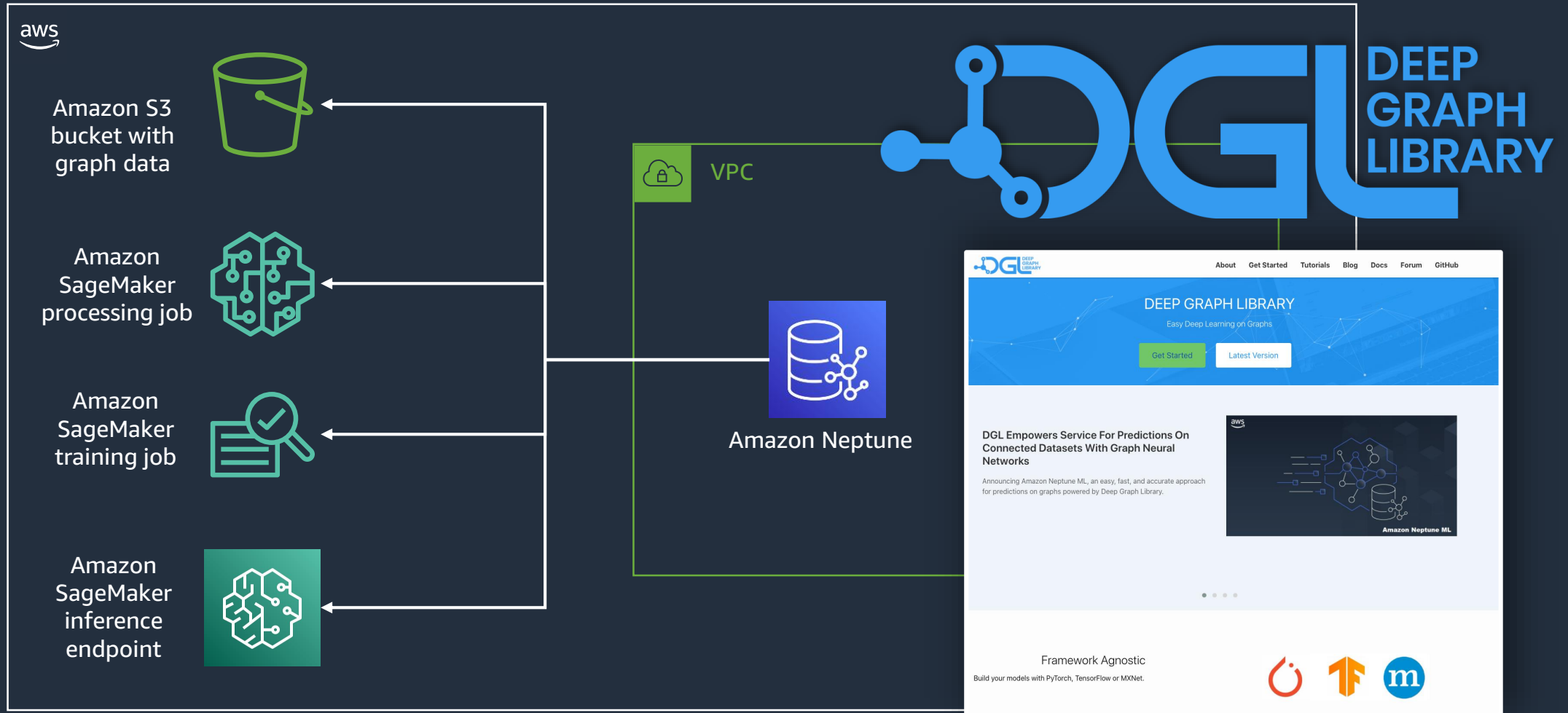
AWS Database Migration Service



Amazon CloudWatch



Neptune and SageMaker: Neptune ML



Machine Learning on Graph

Why Machine Learning on graph data?

Graphs provide machine learning with novel features based on connections within the data

Leverage connectedness

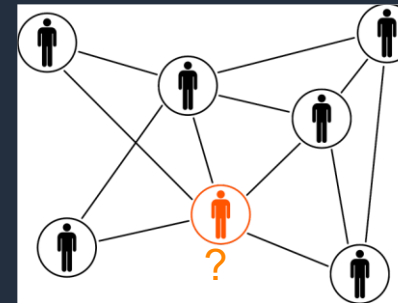
Unique insights and increased prediction accuracy leveraged by ML to detect patterns deep into data connections

Adapt over time

Common ML use-cases such as fraud, require the flexibility provided by graphs to model data and patterns as they evolve

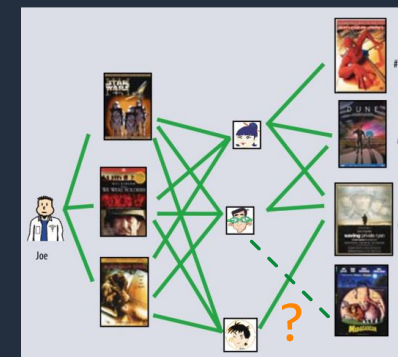
See data at scale

Graphs and graph embeddings provide a unique ability to efficiently represent data with billions of relationships



Node Classification & Regression

Customer targeting, malicious transactions, etc.



Link Prediction

Recommendations, hidden relationships, etc.

Why Machine Learning on graph data?

Graphs provide machine learning with novel features based on connections within the data

Leverage connectedness

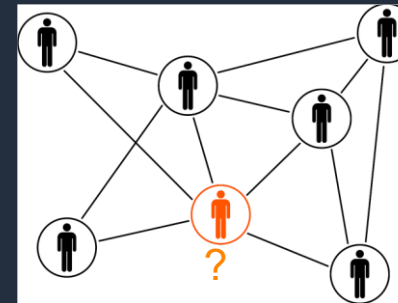
Unique insights and increased prediction accuracy leveraged by ML to detect patterns deep into data connections

Adapt over time

Common ML use-cases such as fraud, require the flexibility provided by graphs to model data and patterns as they evolve

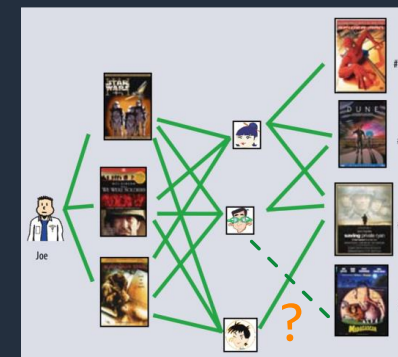
See data at scale

Graphs and graph embeddings provide a unique ability to efficiently represent data with billions of relationships



Node Classification & Regression

Customer targeting, malicious transactions, etc.



Link Prediction

Recommendations, hidden relationships, etc.

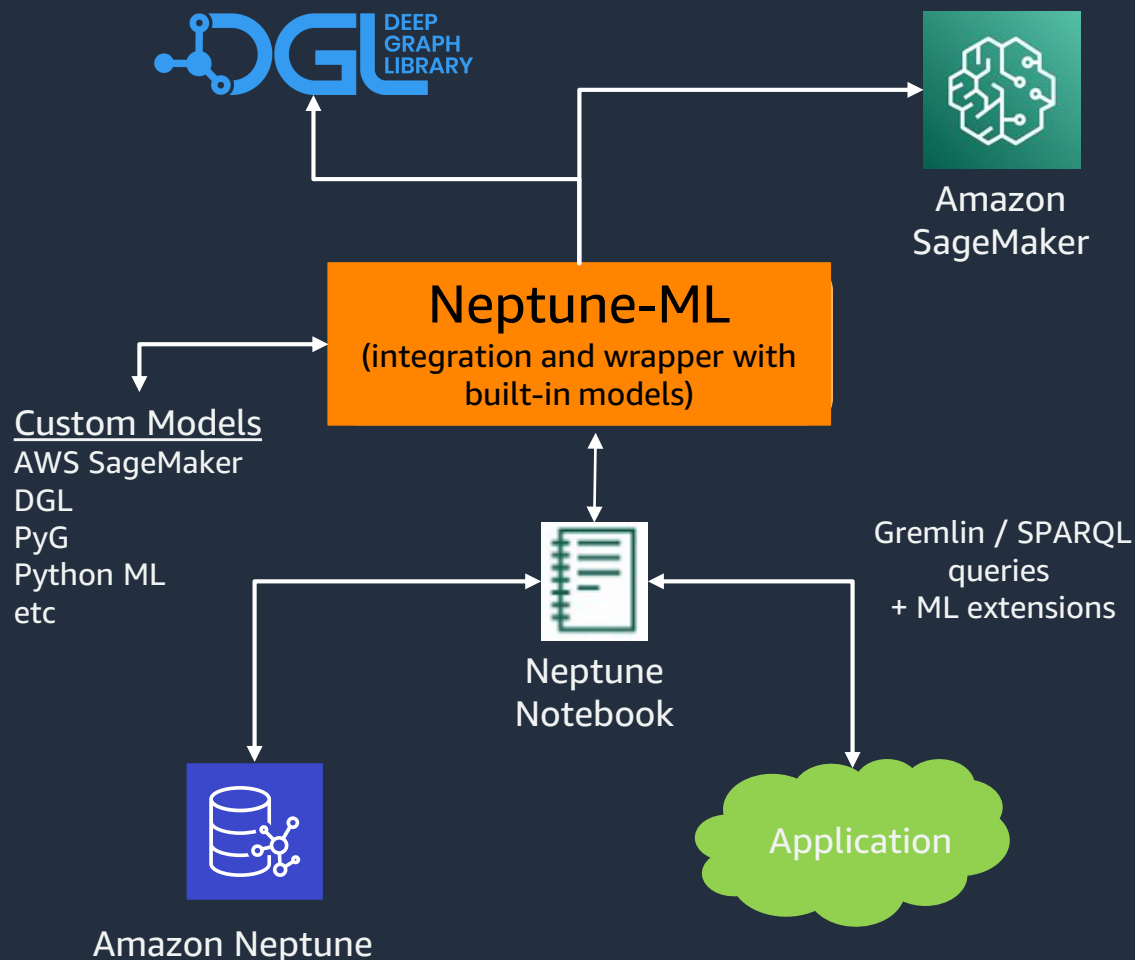
Traditional Path for Graph ML Solution

Building a ML solution for graph data requires:

- Experts in graph DBs, deep learning frameworks, productionizing models, etc.
- Export data from graph database
- Perform feature engineering
- Develop and iterate/innovate models
- Build infrastructure
- Select and deploy the best model
- Automate the process to adjust to dynamic data

Repetitive workflow requires sequence of manual steps and is work intensive

Graph Machine Learning **simplified** by Neptune ML



Graph predictions using Neptune ML

Automatically choose and train the best ML model
Seamless integration with AWS SageMaker and DGL

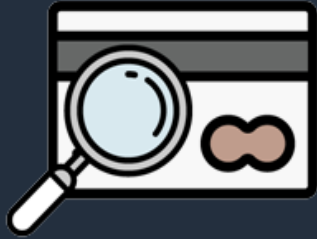
Use state-of-the-art graph ML

Higher accuracy by using GNNs embeddings and inference, leveraging the relationships and features in graphs

Scale runtime inferences to large graphs

Infer data within the query mid-traversal across the billions of nodes/relationships

Use-cases for Neptune Machine Learning



**Fraud
Detection**



**Product
Recommendations**



**Identity
Resolution**



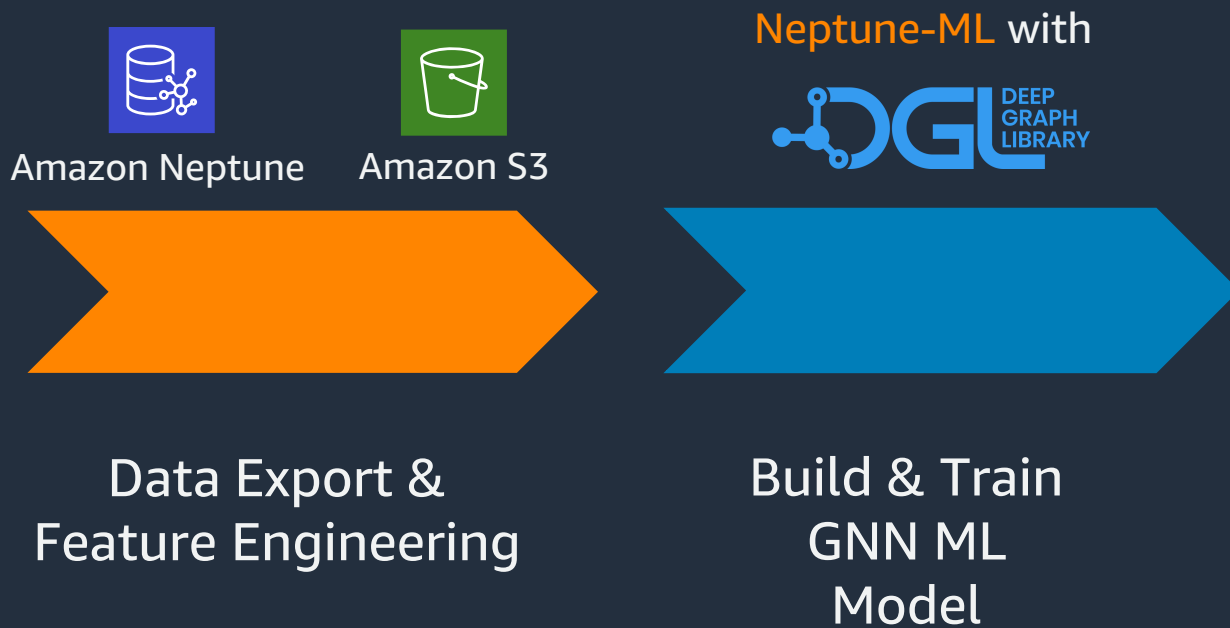
**Knowledge
Graph**

Neptune ML



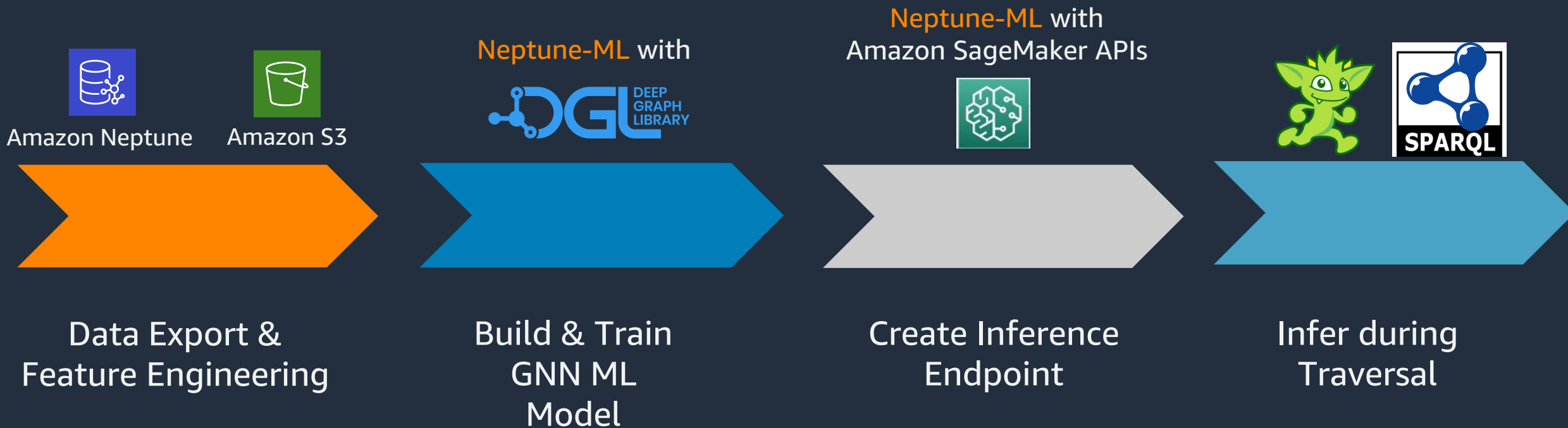
Amazon Neptune ML/GNN Workflow

Adds machine-learning based inference in 4 simple steps



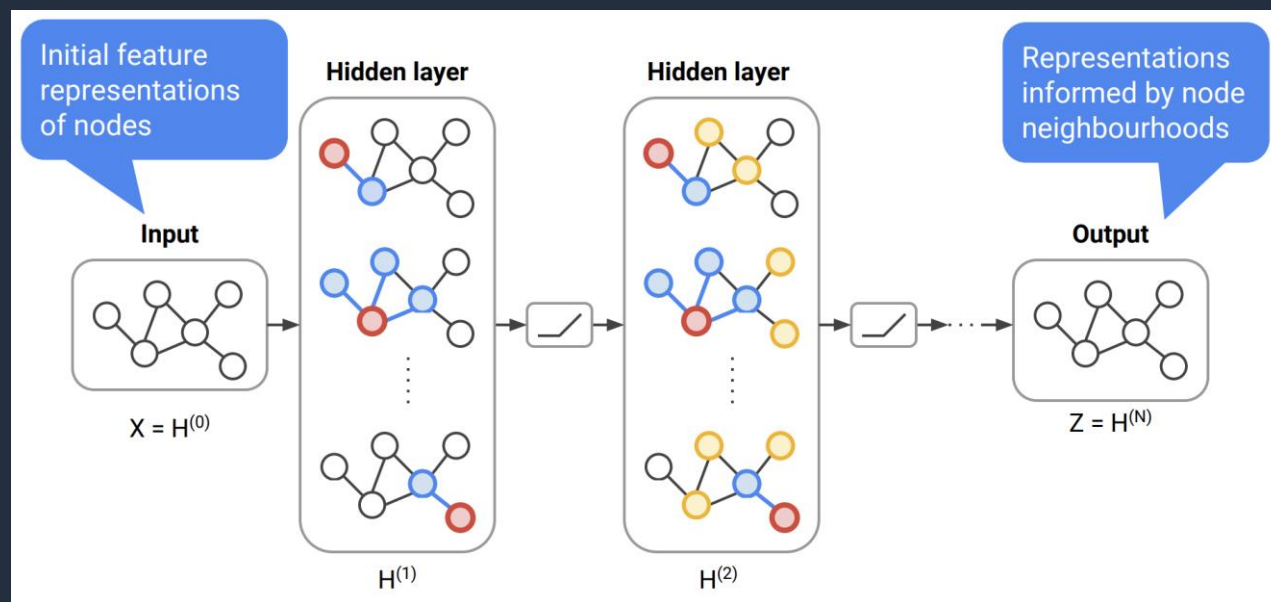
Amazon Neptune ML/GNN Workflow







Adds machine-learning based inference in 4 simple steps



What are Graph Neural Networks (GNNs)?

A family of deep neural networks that learn inductive node embeddings



	Class A	0.993
	Class A	0.679
	Class B	0.779
	Class A	0.823
	Class B	0.662
	Class A	0.913

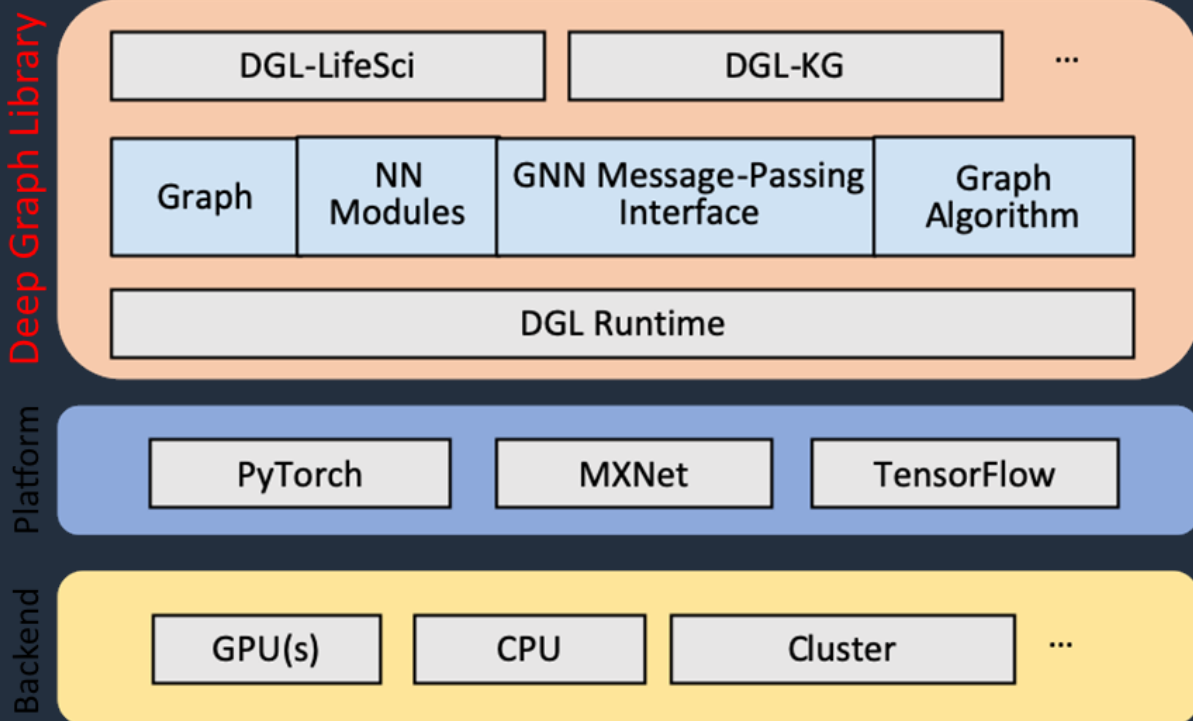
Compute embeddings using graph structure and node/edge features

Nonlinearly integrates topologically distant information

GNNs can be trained end-to-end

Deep Graph Library (DGL) powers Neptune ML

Easy deep learning on graphs



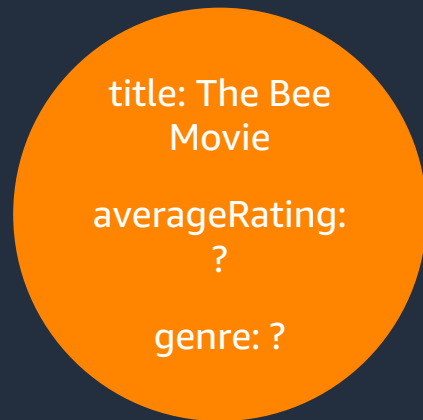
- Easy to develop new models
- Seamless integration with existing frameworks
- Fast and scalable
- Use built-in NN modules for popular GNN models e.g. GraphSage, GCN, RGCN, etc.



What can you do with Neptune ML?

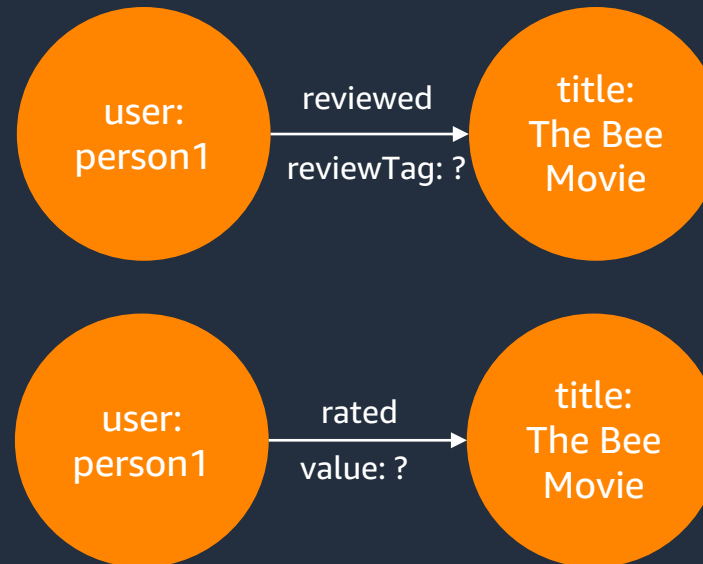
Node Prediction

Predict the values of categorical or numerical properties for nodes



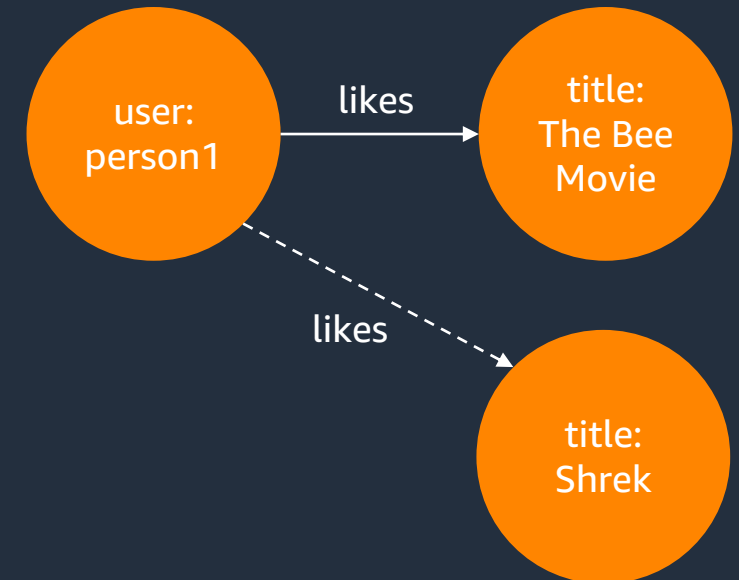
Edge Prediction

Predict the values of categorical or numerical properties for edges



Link Prediction

Predict the connections between nodes



What inferencing options do I have?

Transductive inference

Nodes/edges being queried for prediction were in the graph at the time of model training

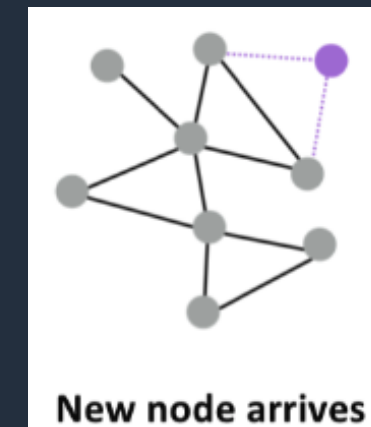
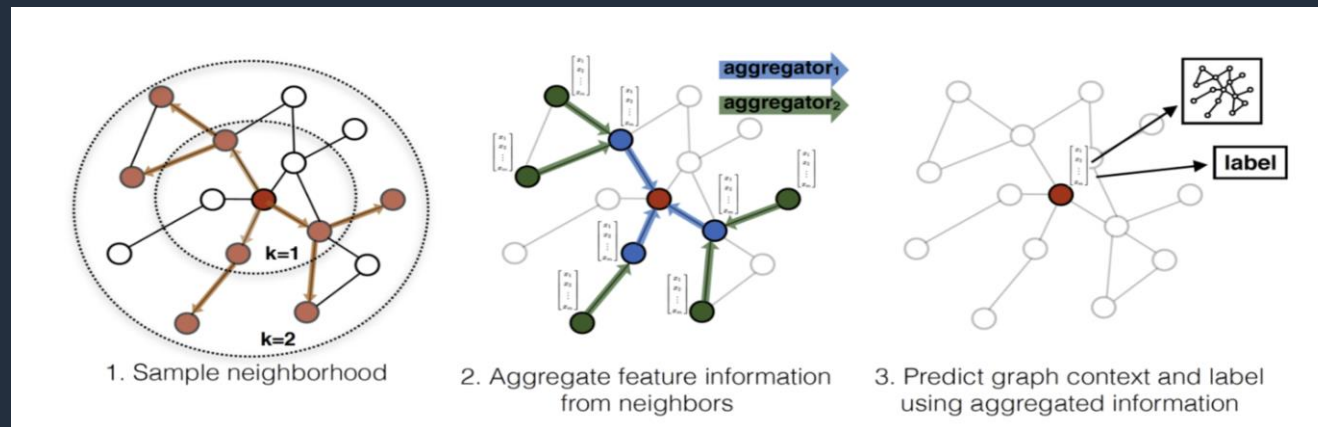
What inferencing options do I have?

Transductive inference

Nodes/edges being queried for prediction were in the graph at the time of model training

Inductive inference

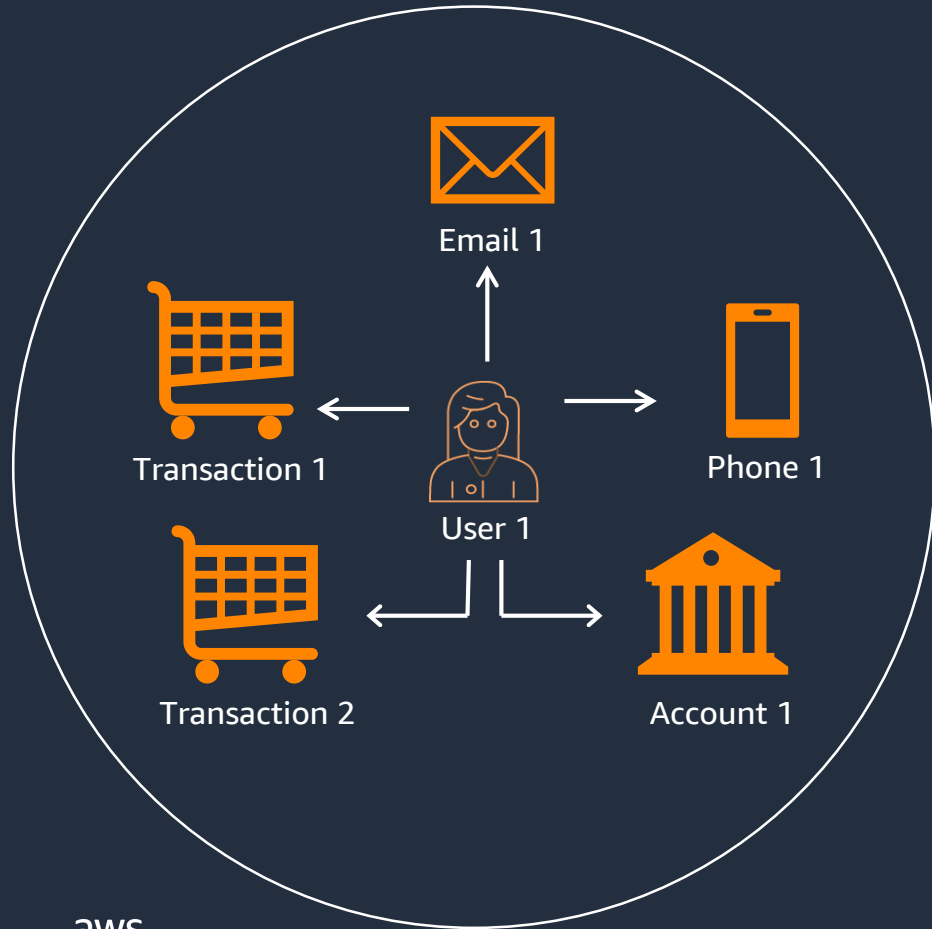
Enable predictions on nodes, edges and properties that were added to the graph after the ML model training process.



Using Neptune ML for Fraud Detection

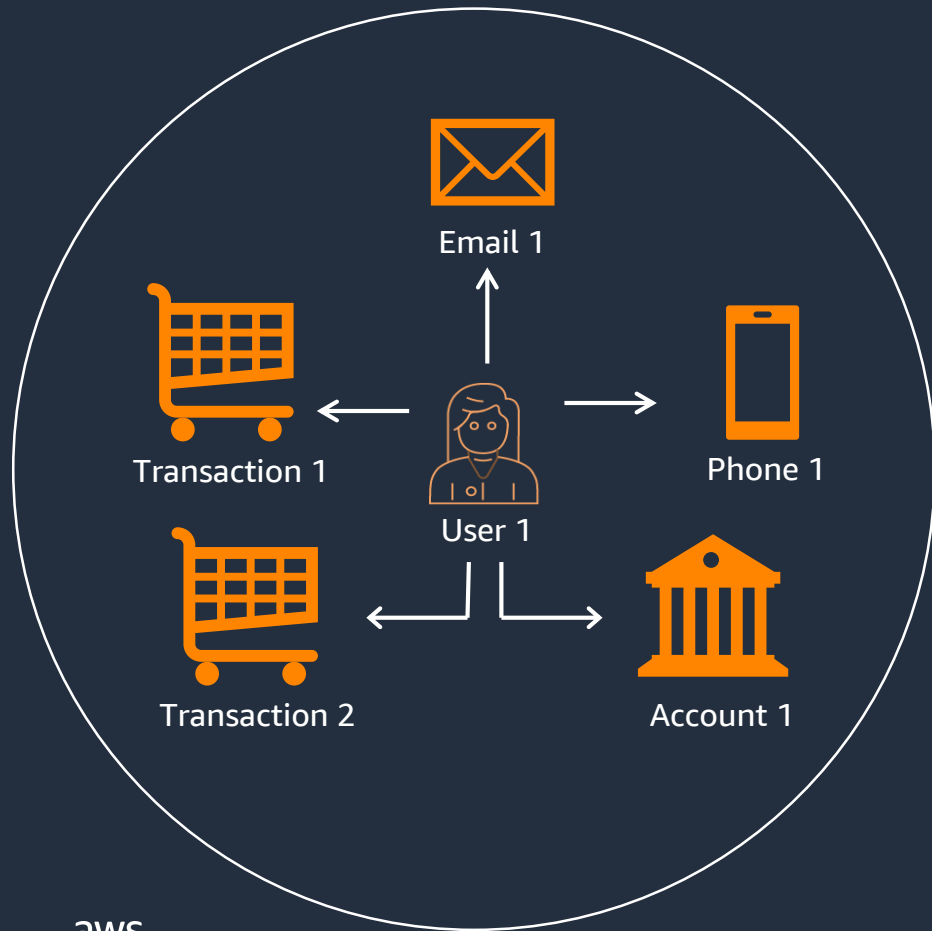
A fraud graph model

Conceptual fraud model

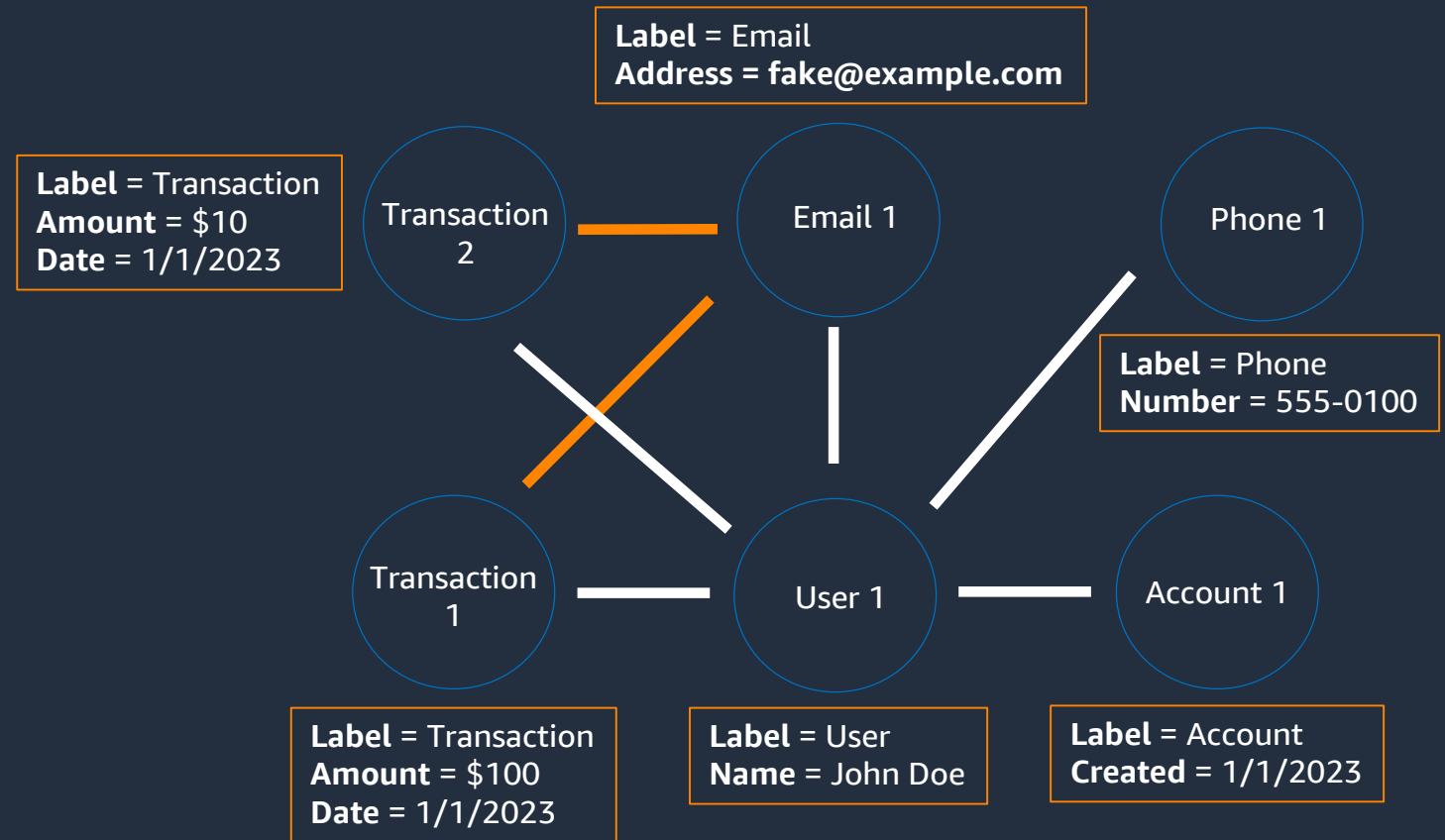


A fraud graph model

Conceptual fraud model



Fraud graph model



Fraud Detection with Neptune ML & GNN

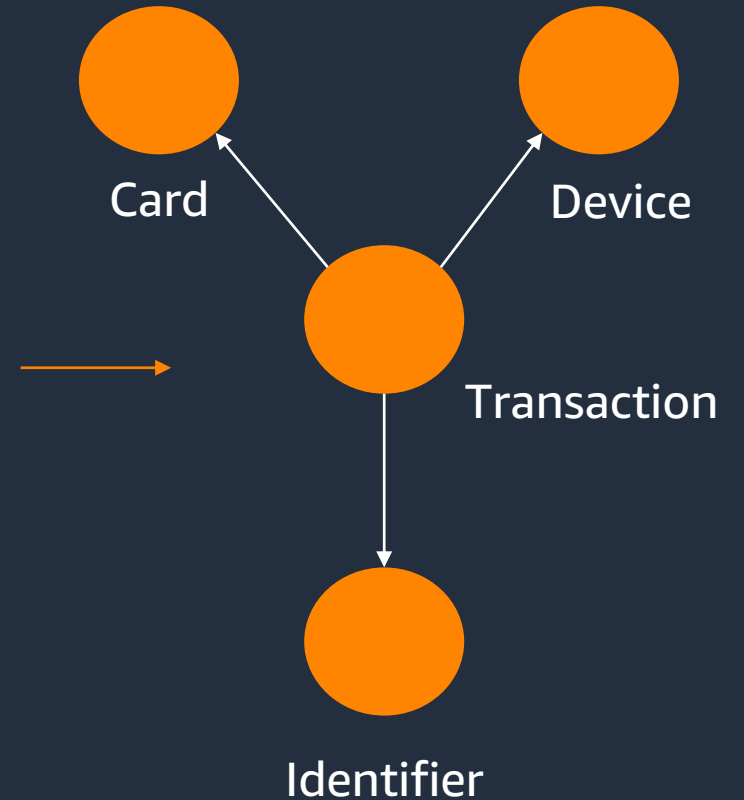
Identify source datasets

```
%%gremlin
g.V().groupCount().by(label).unfold().order().by(keys)
```

The output should contain:

- 406 Addresses
- 14286 Cards
- 1788 Devices
- 60 Emails
- 119193 Identifiers
- 5 Products
- 590540 Transactions

Use only the data needed to answer the business questions



Using IEEE CIS Dataset (Anonymised Fraud Dataset)

Data Source:

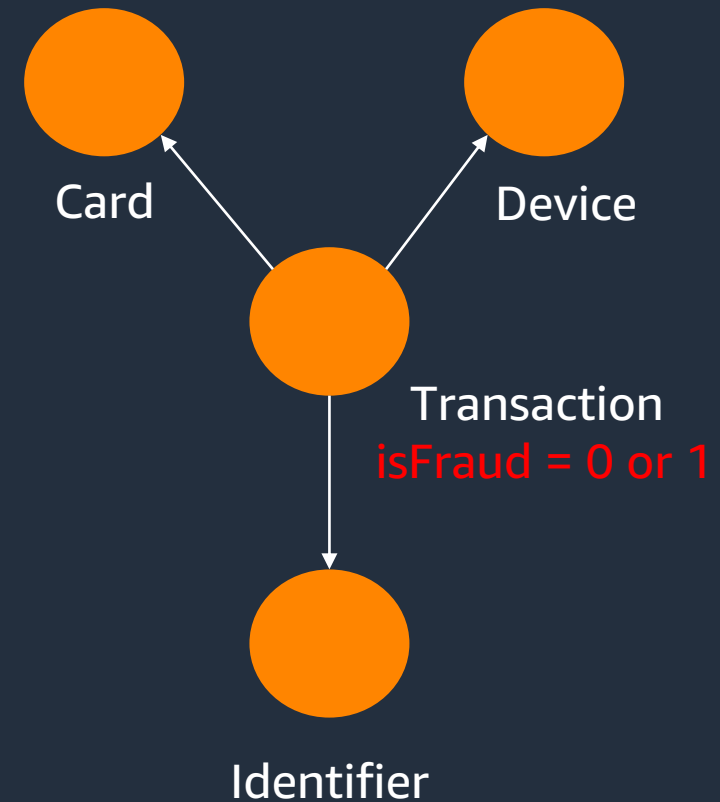
<https://www.kaggle.com/c/ieee-fraud-detection/data>

Fraud Detection with Neptune ML & GNN

Define Machine Learning Prediction Target

Identifying the prediction target

The data already contains 1.5% of fraudulent labelled transactions – defined by an “isFraud” property of 0 or 1.



Fraud Detection with Neptune ML & GNN

Define Machine Learning Prediction Target

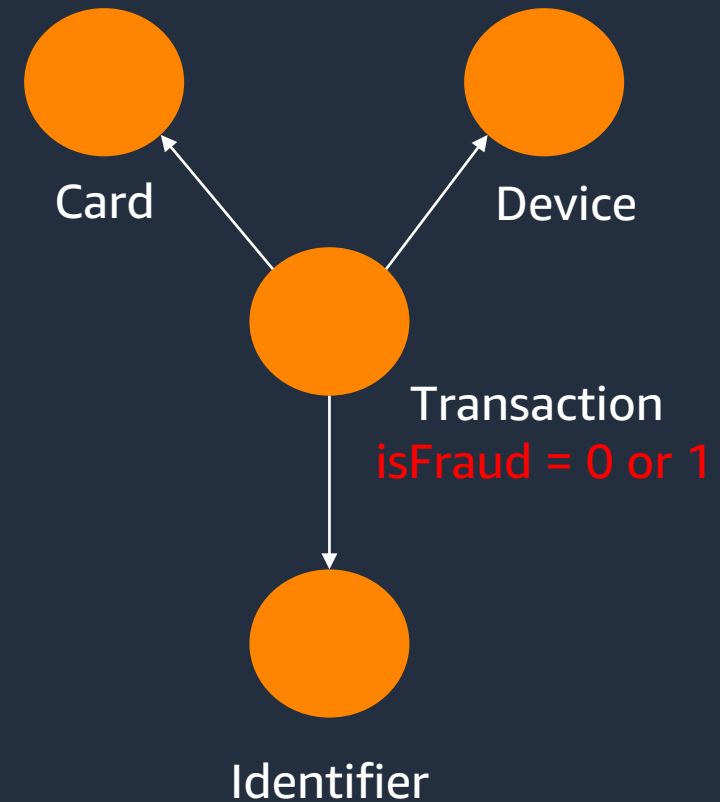
Identifying the prediction target

The data already contains 1.5% of fraudulent labelled transactions – defined by an “isFraud” property of 0 or 1.

Prediction target used for training

Classify the nodes labelled as “Transaction” with the property “isFraud”

```
"neptune_ml": {  
  "version": "v2.0",  
  "targets": [{  
    "node": "Transaction",  
    "property": "isFraud",  
    "type": "classification"  
  }]  
}
```

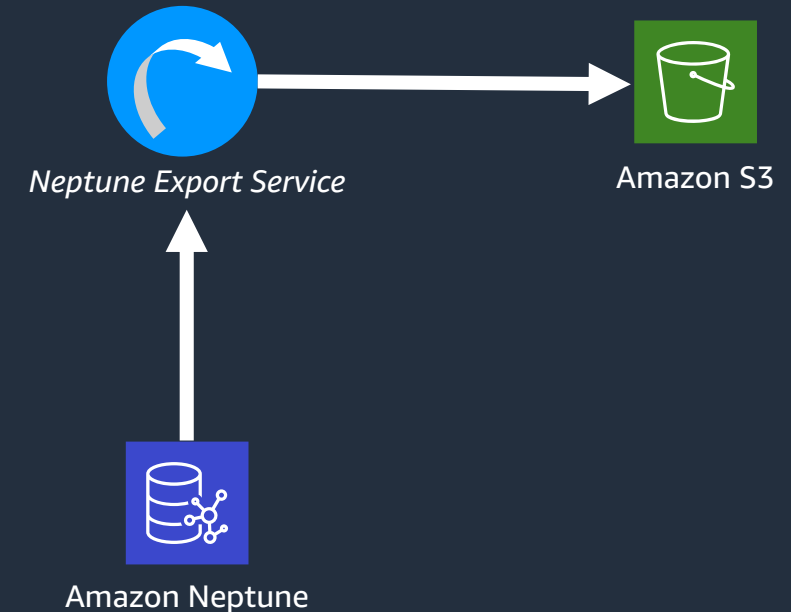


Export Data and Configuration

Neptune ML automates the data export and configuration generation process

```

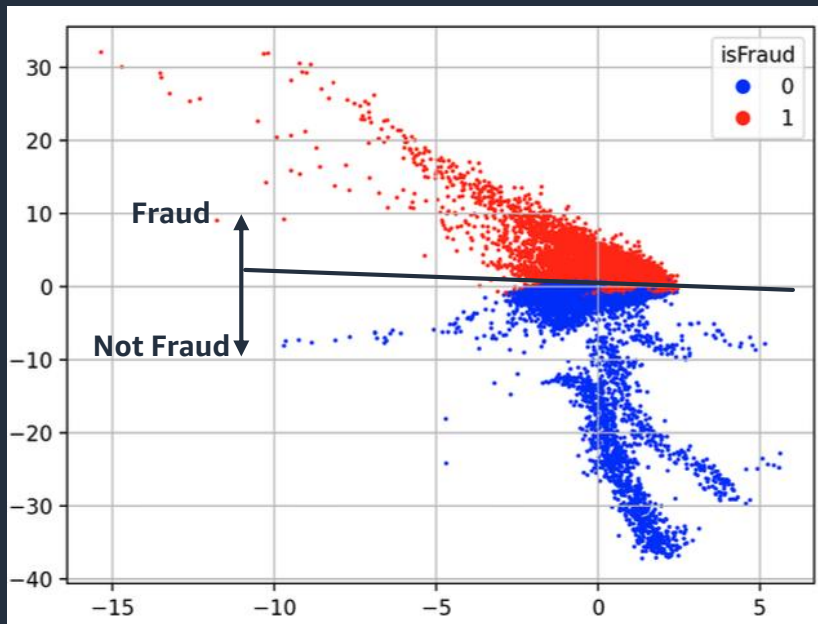
export_params={
  "command": "export-pg",
  "params": {
    "endpoint": neptune_ml.get_host(),
    "profile": "neptune_ml",
    "useIamAuth": neptune_ml.get_iam(),
    "cloneCluster": True,
    "cloneClusterInstanceType": "r5.12xlarge",
    "nodeLabels": ["Transaction", "Identifier", "Device", "Card"],
    "edgeLabels": ["identified_by", "purchased_by", "associated_with"]
  },
  "outputS3Path": f'{s3_bucket_uri}/neptune-export',
  "additionalParams": {
    "neptune_ml": {
      "version": "v2.0",
      "targets": [
        {
          "node": "Transaction",
          "property": "isFraud",
          "type": "classification"
        }
      ]
    }
  },
  "jobSize": "medium"
}
    
```



Fraud Detection with Neptune ML & GNN

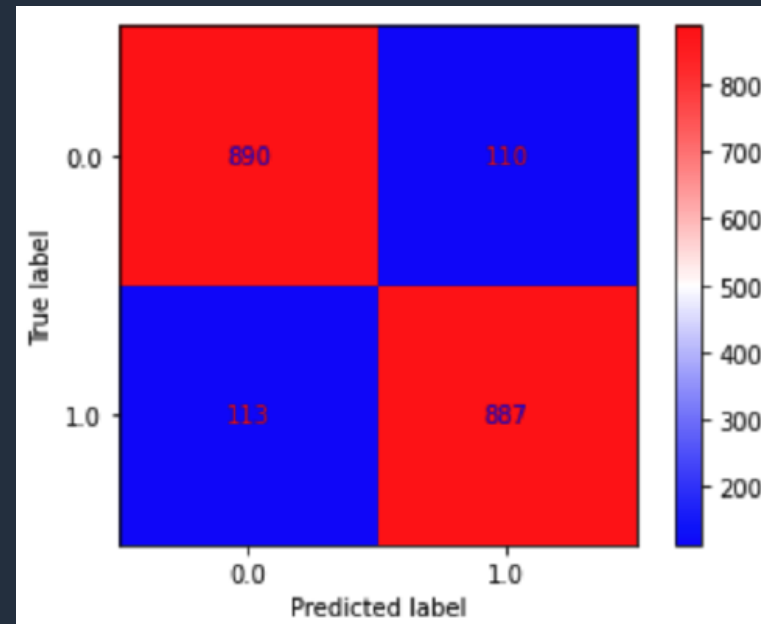
Resulting ML model and evaluating performance

Resulting Embeddings



570K non-fraud transactions
20K fraud transactions

Confusion Matrix



overall accuracy = 88.8%
correctly predicted legit = 89.0%
correctly predicted fraud = 88.7%

Fraud Detection with Neptune ML & GNN Example

Transductive prediction queries via Neptune Gremlin query language

Predict isFraud for “transaction” node “3001822”

```
g.with('Neptune#ml.endpoint', '${endpoint}')  
  .v('3001822').properties('isFraud', 'Neptune#ml.score')  
  .with('Neptune#ml.classification').value()
```

→ Result: 0, 0.644

Predict isFraud for all “transaction” nodes

```
g.with('Neptune#ml.endpoint', '${endpoint}')  
  .V().hasLabel('Transaction').hasNot('isFraud')  
  .properties('isFraud').with('Neptune#ml.classification').value()
```

Now, all the “Transaction” nodes have the label isFraud with value 0 or 1.

Fraud Detection with Neptune ML & GNN Example

Inductive prediction queries via Neptune Gremlin query language

Add a new transaction node

```
g.addV('Transaction').property(T.id, '9999999')
    .property('id_01', -5.0)
    .property('id_02', 222578.0)
    ...
    .property('id_12', 'NotFound')
    .property('id_15', 'New')
    .property('id_16', 'NotFound').as('newT')
.addE('associated_with')
    .from('newT')
    .to(V('DeviceType:mobile'))
    .property(T.id, 'transaction_9999999-associated-with-mobile')
.addE('purchased_by')
    .from('newT')
    .to(V('card3:150.0'))
    .property(T.id, 'transaction_9999999-purchased-by-card')
.addE('identified_by')
    .from('newT')
    .to(V('id_05:0.0'))
    .property(T.id, 'transaction_9999999-identified-by-id_05')
.addE('identified_by')
    .from('newT')
    .to(V('id_38:F'))
    .property(T.id, 'transaction_9999999-identified-by-id_38')
```

Fraud Detection with Neptune ML & GNN Example

Inductive prediction queries via Neptune Gremlin query language

Add a new transaction node

```
g.addV('Transaction').property(T.id, '9999999')
    .property('id_01', -5.0)
    .property('id_02', 222578.0)
    ...
    .property('id_12', 'NotFound')
    .property('id_15', 'New')
    .property('id_16', 'NotFound').as('newT')
.addE('associated_with')
    .from('newT')
    .to(V('DeviceType:mobile'))
    .property(T.id, 'transaction_9999999-associated-with-mobile')
.addE('purchased_by')
    .from('newT')
    .to(V('card3:150.0'))
    .property(T.id, 'transaction_9999999-purchased-by-card')
.addE('identified_by')
    .from('newT')
    .to(V('id_05:0.0'))
    .property(T.id, 'transaction_9999999-identified-by-id_05')
.addE('identified_by')
    .from('newT')
    .to(V('id_38:F'))
    .property(T.id, 'transaction_9999999-identified-by-id_38')
```

Inductive prediction on new node

```
g.with('Neptune#ml.endpoint', '${endpoint}')
  .V('9999999').properties('isFraud')
  .with('Neptune#ml.classification')
  .with('Neptune#ml.inductiveInference')
  .with('Neptune#ml.deterministic').value()
```

Result:

1, 0.5687



What started as a vehicle-for-hire app, Careem now connects consumers to a variety of services for transportation, delivery, and payments in over 100 cities across 14 countries.

Challenge:

Detect and stop losses from first-party and third-party fraud. Prior methods of fraud detection involved rules and ML models, but was reactive instead of proactive.

Solution:

- Identity graph in Neptune containing information about users and shared features
- Node classification using Neptune ML to infer fraud

Learn more: <https://go.aws/3loHGL5>





What started as a vehicle-for-hire app, Careem now connects consumers to a variety of services for transportation, delivery, and payments in over 100 cities across 14 countries.

Challenge:

Detect and stop losses from first-party and third-party fraud. Prior methods of fraud detection involved rules and ML models, but was reactive instead of proactive.

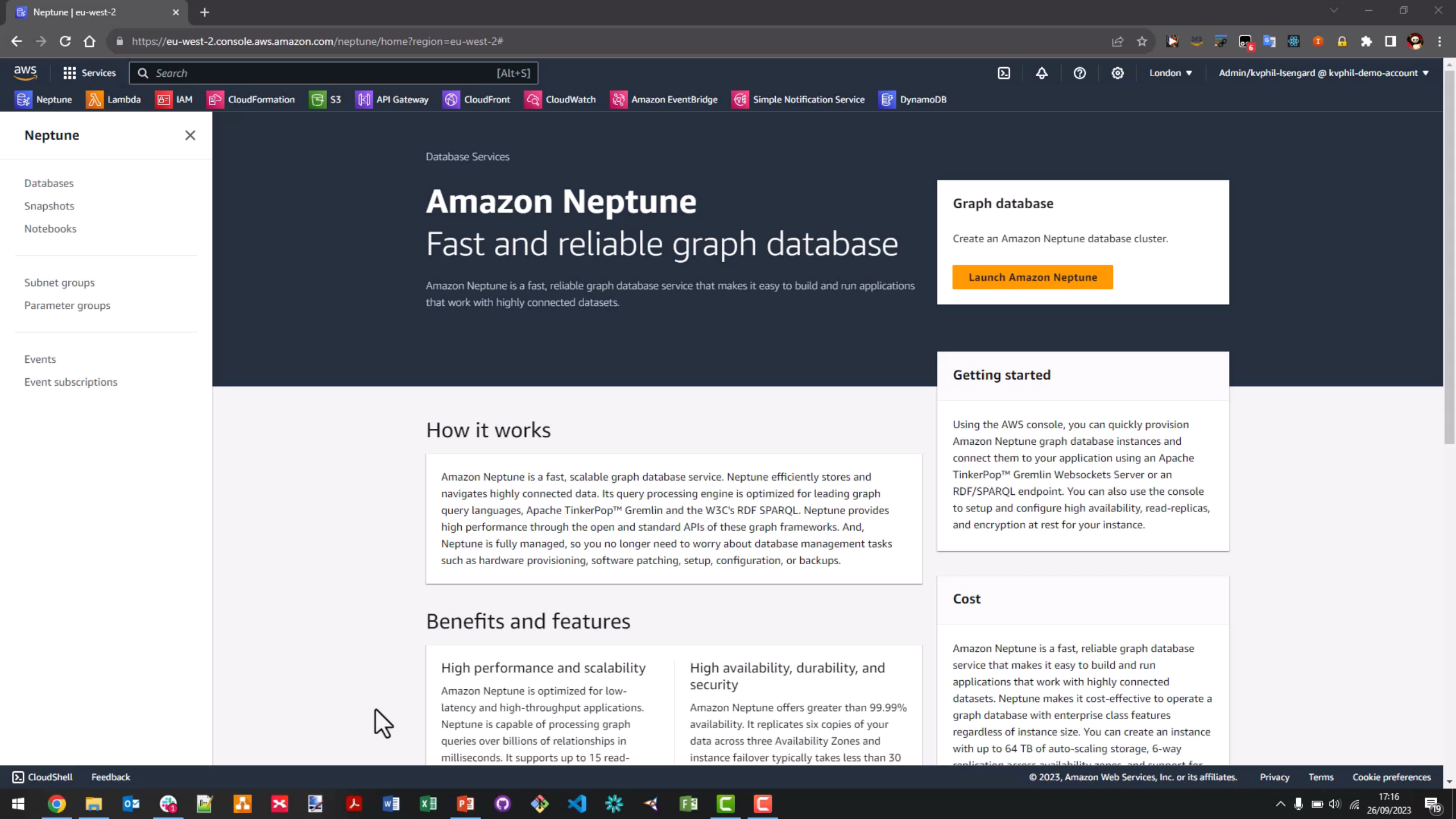
Solution:

- Identity graph in Neptune containing information about users and shared features
- Node classification using Neptune ML to infer fraud

Learn more: <https://go.aws/3loHGL5>



Demo: Performing real-time inference using Neptune ML



- Neptune
- Databases
- Snapshots
- Notebooks
- Subnet groups
- Parameter groups
- Events
- Event subscriptions

Database Services

Amazon Neptune

Fast and reliable graph database

Amazon Neptune is a fast, reliable graph database service that makes it easy to build and run applications that work with highly connected datasets.

Graph database

Create an Amazon Neptune database cluster.

[Launch Amazon Neptune](#)

Getting started

Using the AWS console, you can quickly provision Amazon Neptune graph database instances and connect them to your application using an Apache TinkerPop™ Gremlin Websockets Server or an RDF/SPARQL endpoint. You can also use the console to setup and configure high availability, read-replicas, and encryption at rest for your instance.

How it works

Amazon Neptune is a fast, scalable graph database service. Neptune efficiently stores and navigates highly connected data. Its query processing engine is optimized for leading graph query languages, Apache TinkerPop™ Gremlin and the W3C's RDF SPARQL. Neptune provides high performance through the open and standard APIs of these graph frameworks. And, Neptune is fully managed, so you no longer need to worry about database management tasks such as hardware provisioning, software patching, setup, configuration, or backups.

Benefits and features

- High performance and scalability**
Amazon Neptune is optimized for low-latency and high-throughput applications. Neptune is capable of processing graph queries over billions of relationships in milliseconds. It supports up to 15 read-
- High availability, durability, and security**
Amazon Neptune offers greater than 99.99% availability. It replicates six copies of your data across three Availability Zones and instance failover typically takes less than 30

Cost

Amazon Neptune is a fast, reliable graph database service that makes it easy to build and run applications that work with highly connected datasets. Neptune makes it cost-effective to operate a graph database with enterprise class features regardless of instance size. You can create an instance with up to 64 TB of auto-scaling storage, 6-way replication, 3-availability zones, and support for



Summary

Neptune ML enables Graph ML and GNN Pipeline

Seamless integration and abstract interfacing with AWS SageMaker and Deep Graph Library

No need for custom integrations, moving data around, learn separate tools, or even have deep ML experience

Make real-time Inductive Inference for applications in seconds

- **End-to-end data ingestion pipeline** for streaming events and bulk data
- Generate **continuous value** (confidence) score rather than a hard decision
- GNN - **better generalisation** for new (unseen) patterns
- **Efficient grouping of data** for **higher accuracy** classification or regressions
- Leverage data relationships as well as data features
- Use **real-time** graph data input **without the need to retrain**
- **Less labour intensive** than rule based or analytic based solutions



Amazon Neptune



Amazon SageMaker



Resources



Neptune ML Documentation

Start with the 'Getting Started' and 'Neptune Machine Learning' sections

The screenshot shows the AWS documentation page for 'Getting started with graph databases'. The page title is 'Getting started with graph databases' and it includes a search bar, navigation links, and a sidebar with a table of contents. The main content area contains an introduction to Amazon Neptune, a definition of graph databases, and a small graph diagram illustrating relationships between people and their hobbies.

Getting started with graph databases

Amazon Neptune is a fully managed graph database service that scales to handle billions of relationships and lets you query them with milliseconds latency, at a low cost for that kind of capacity.

If you know about graphs already, jump ahead to [Try out using graphs](#).

What exactly is a graph database?

Graph databases are optimized to store and query the *relationships* between data items.

They store data items themselves as *vertices* of the graph, and the relationships between them as *edges*. Each edge has a type, and is directed from one vertex (the start) to another (the end). Relationships can be called *predicates* as well as edges, and vertices are also sometimes referred to as *nodes*. In so-called property graphs, both vertices and edges can have additional *properties* associated with them too.

Here is a small graph representing friends and hobbies in a social network:

```

    graph TD
      Anna((Anna)) -- Friend --> Alana((Alana))
      Anna -- Likes --> Books((Books))
      Justin((Justin)) -- Friend --> Alana
      Justin -- Likes --> Movies((Movies))
      Alana -- Likes --> Movies
  
```

The screenshot shows the AWS documentation page for 'Using the Neptune ML AWS CloudFormation template to get started quickly'. The page title is 'Using the Neptune ML AWS CloudFormation template to get started quickly' and it includes a search bar, navigation links, and a sidebar with a table of contents. The main content area contains an introduction to the quick-start template and a table of regions with 'Launch Stack' buttons.

Using the Neptune ML AWS CloudFormation template to get started quickly

The easiest way to get started with Neptune ML is to use the AWS CloudFormation quick-start template. This template installs all necessary components including a Neptune DB cluster, enables Neptune ML in lab mode, and sets up the necessary IAM roles.

To create the Neptune ML quick-start stack

1. To launch the AWS CloudFormation stack on the AWS CloudFormation console, choose one of the **Launch Stack** buttons in the following table:

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 (N. California)	View	View in Designer	Launch Stack
US West (Oregon)	View	View in Designer	Launch Stack
Canada (Central)	View	View in Designer	Launch Stack
South America (São Paulo)	View	View in Designer	Launch Stack
Europe (Stockholm)	View	View in Designer	Launch Stack
Europe (Ireland)	View	View in Designer	Launch Stack
Europe (London)	View	View in Designer	Launch Stack

DGL Documentation

Start with the 'Getting Started' and 'Tutorials' sections

The screenshot shows the DGL homepage with a blue header containing the DGL logo and navigation links: About, Get Started, Tutorials, Blog, Docs, Forum, and GitHub. Below the header is a large blue banner with the text "DEEP GRAPH LIBRARY" and "Easy Deep Learning on Graphs". Two buttons, "Get Started" and "Latest Version", are positioned on the banner. The main content area features a white background with a section titled "DGL Empowers Service For Predictions On Connected Datasets With Graph Neural Networks". This section includes a sub-headline "Announcing Amazon Neptune ML, an easy, fast, and accurate approach for predictions on graphs powered by Deep Graph Library." and an image of the Amazon Neptune ML logo. At the bottom, it states "Framework Agnostic" and "Build your models with PyTorch, TensorFlow or MXNet." with logos for PyTorch, TensorFlow, and MXNet.

The screenshot shows a documentation page for "Graph neural networks and its variants". The page has a blue header with the DGL logo and version "0.5.x". A search bar is located below the header. The left sidebar contains a table of contents with sections: GET STARTED (Install DGL, Working with different backends, DGL at a Glance), USER GUIDE (Chapter 1: Graph, Chapter 2: Message Passing, Chapter 3: Building GNN Modules, Chapter 4: Graph Data Pipeline, Chapter 5: Training Graph Neural Networks, Chapter 6: Stochastic Training on Large Graphs, Chapter 7: Distributed Training), and API REFERENCE (dgl, dgl.data, dgl.dataloading, dgl.DGLGraph, dgl.distributed, dgl.function, dgl.nn, dgl.ops, dgl.sampling, User-defined Functions). The main content area has a title "Graph neural networks and its variants" and a list of bullet points describing different network types: Graph convolutional network (GCN), Graph attention network (GAT), Relational-GCN, and Line graph neural network (LGNN). Each bullet point includes links to research papers, tutorials, and code examples. Below the text are three visualizations: a Graph Convolutional Network (GCN) diagram, a Relational graph convolutional network (RGCN) diagram, and a Line graph neural network (LGNN) diagram.

Additional Resources



[Neptune Notebooks/Graph Notebook](#)



[Neptune Reference Architectures](#)



[Neptune Sample Applications](#)



[Use Cases, Videos, Blogs, Code](#)



The collage features several key resources:

- GitHub Article:** "Accessing Amazon Neptune from AWS Lambda Functions" by aws-dbs-refarch-graph/src/acc. It explains that Amazon Neptune runs in a private VPC and endpoints can only be accessed by resources within the VPC. To expose endpoints, a load balancer (Application Load Balancer or Network Load Balancer) is used.
- Walkthrough Diagram:** "Walkthrough of the Architecture" showing the integration of Amazon Neptune within a VPC, connected to an Elastic Network Interface and Amazon SageMaker.
- AWS Online Tech Talks:** A list of recorded sessions:
 - "Build event driven graph applications with AWS purpose-built databases" by Ian Robinson (48:03).
 - "Understanding Game Changes and Player Behavior with Graph Databases" by Nicholas Walsh (50:21).
 - "Deep dive on Amazon Neptune" by Brad Beebe, Karthik Bharathy, and Dirk Lassalle (1:01:01).
 - "Real-world customer use cases with Amazon Neptune" by Karthik Bharathy, Elliott Foster, and Yan Shumar (30:25).



Thank you!

Kevin Phillips

LinkedIn: [kevinphillips81](#)

