



W E B I N A R

# How Getir built a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB

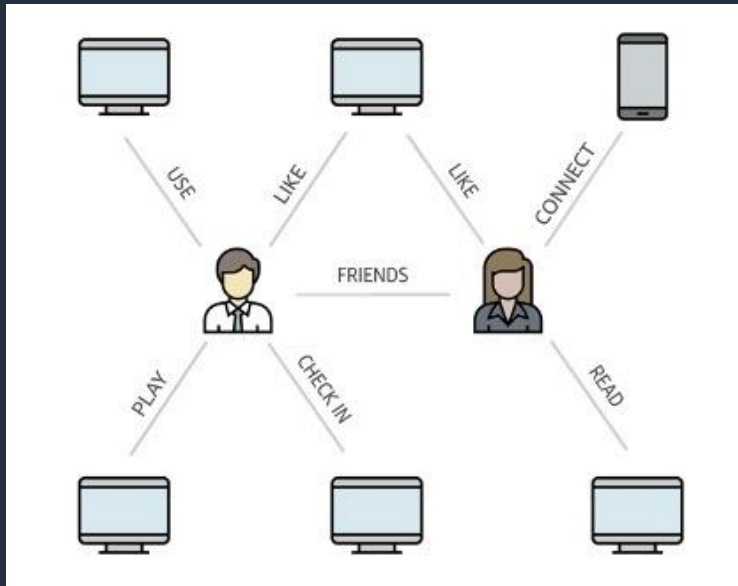
Berkay Berkman

Senior Data Engineer I  
Getir

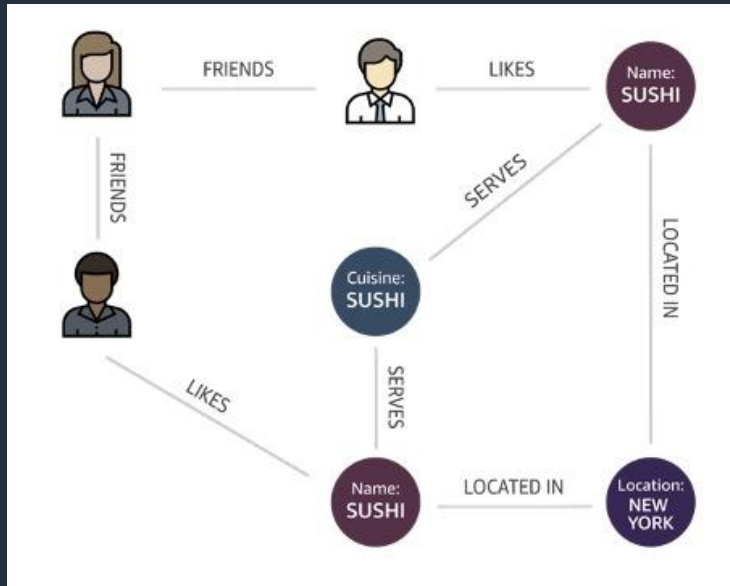
Esra Kayabali

Senior Solutions Architect  
AWS

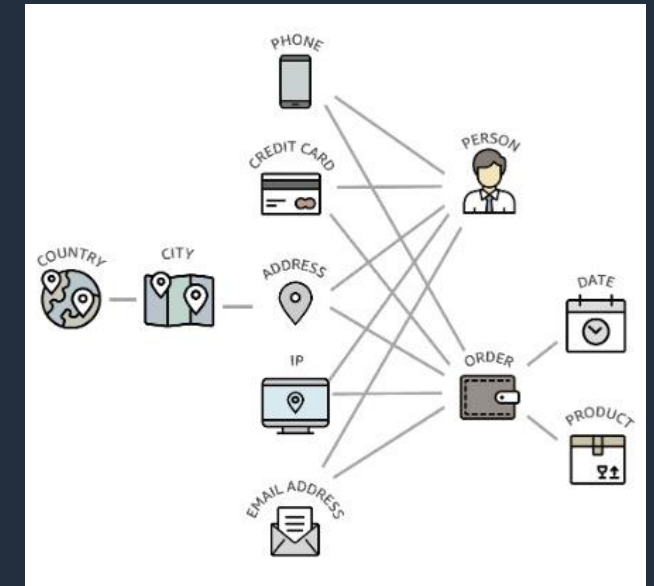
# Highly connected data



Social Networks



Restaurant Recommendations



Retail Fraud Detection

# Use cases for highly connected data



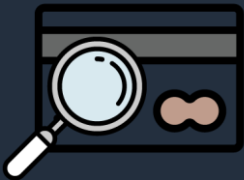
Social Networking



Recommendations



Knowledge Graphs



Fraud Detection



Life Sciences

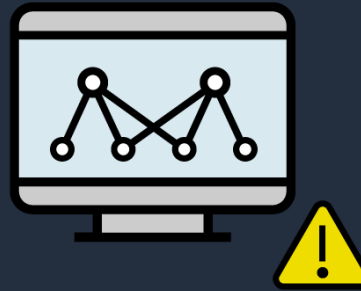


Network & IT Operations

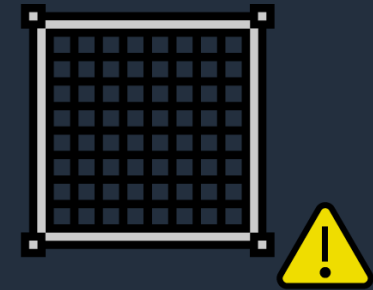
# Relational database challenges building apps with highly connected data



Unnatural for  
querying graph



Inefficient  
graph processing

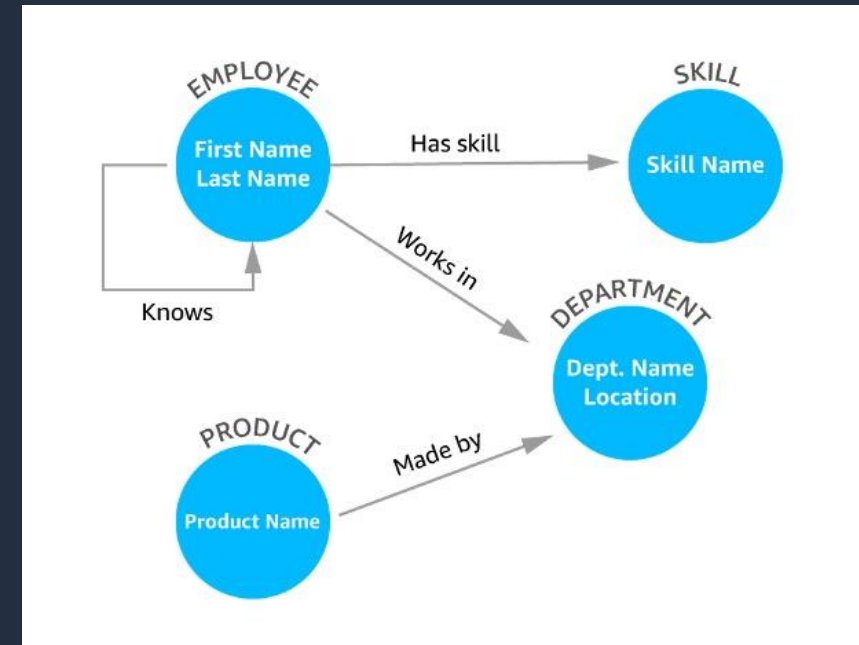
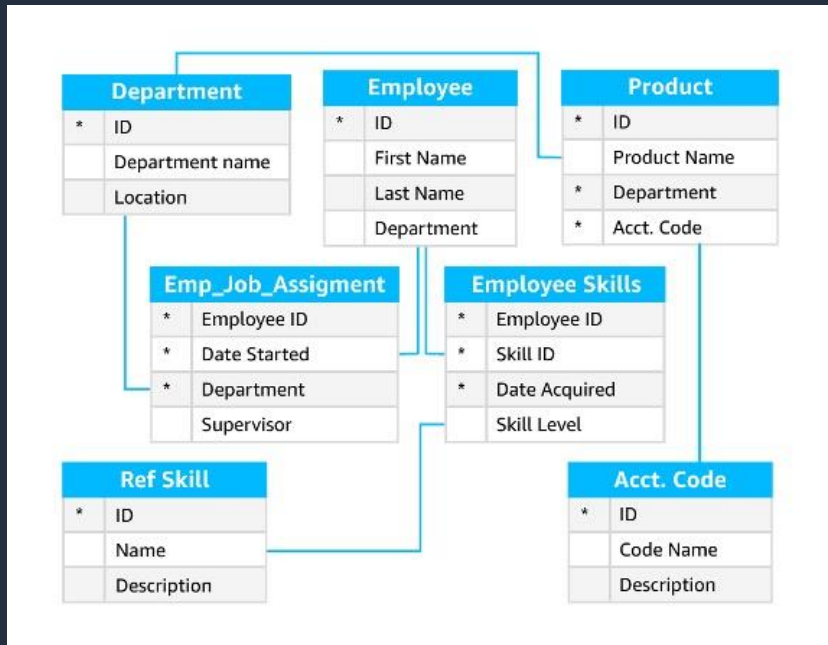


Rigid schema inflexible  
for changing data

# Different approaches for highly connected data

Purpose-built for a business process

Purpose-built to answer questions about relationships



# What is graph?

- Graph databases are purpose-built to store and navigate relationships
- Nodes represent real-world objects
- Edges store relationships between objects



# Amazon Neptune



# Amazon Neptune

Fast, reliable graph database built for the cloud



Supports Property Graph & W3C  
RDF graph models



Query billions of relationships  
with millisecond latency



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



Build powerful queries easily  
with Gremlin and SPARQL

Useful when

Relationships matter as much as the data  
Results depend on the *strength*, *weight*, or *quality* of relationships

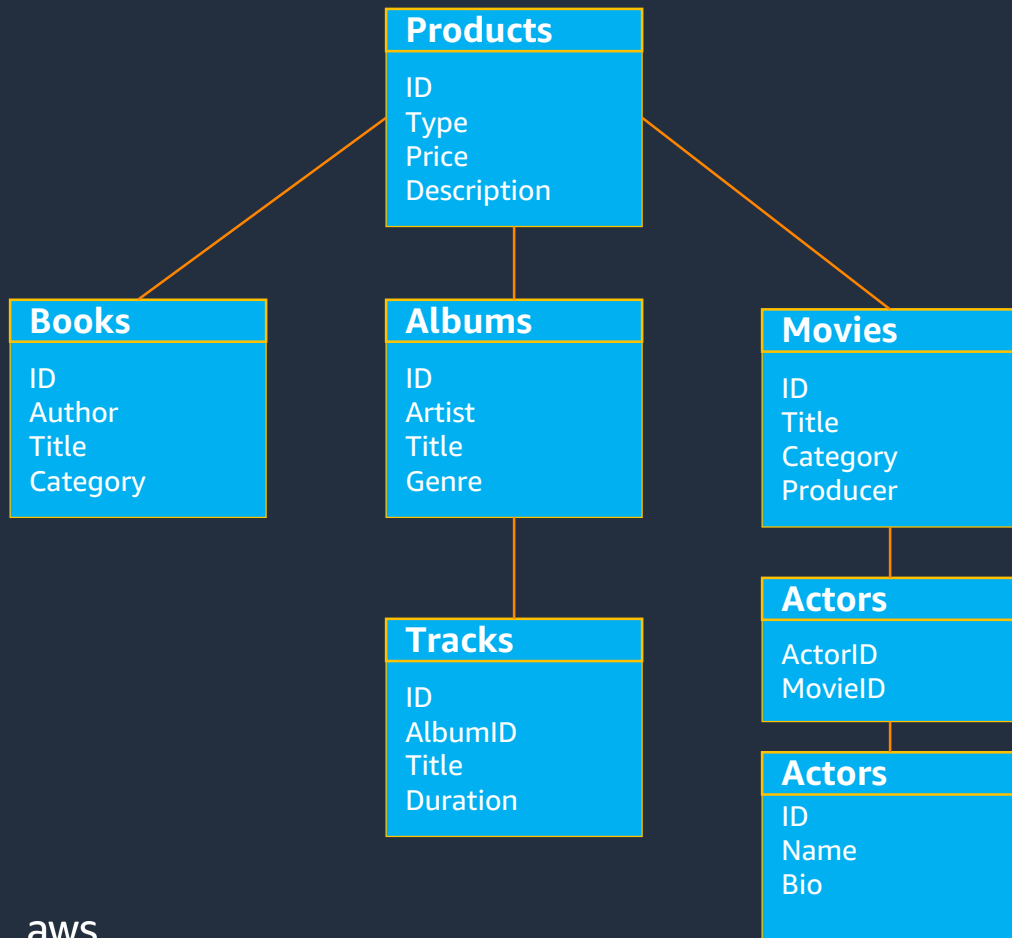
Amazon Neptune is ideal for

Social networking • Recommendations • Fraud detection • Life sciences



# Data Modeling: SQL vs. NoSQL

## SQL



## NoSQL



NoSQL design optimizes for compute instead of storage.

# Amazon DynamoDB



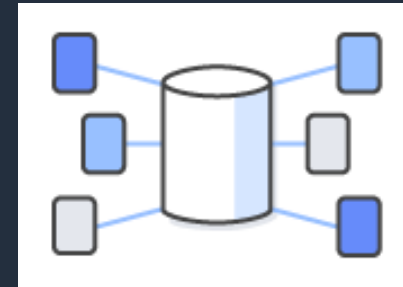
# Amazon DynamoDB



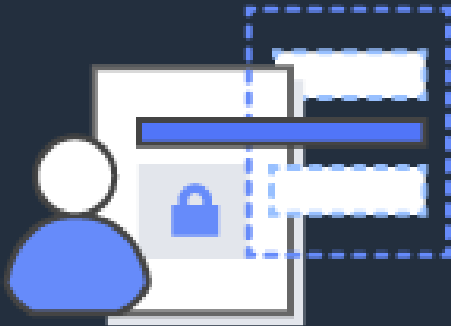
Fully managed



Consistently fast at any scale



Highly available and durable



Secure



Integrates with AWS Lambda, Amazon Redshift, and more



Cost-effective

# Customer story



# About Getir

- Pioneers of super fast delivery.
- Our founding idea: Groceries in 10 minutes, delivered to your door through our app.



- Founded in 2015, Getir means "**bring**" in Turkish
- Ultrafast delivery services, **anything at your door in 10'**
- All verticals – groceries, restaurants, water, car rental, jobs – offered through a **single Getir app**
- Available in **5 countries, in 3 continents**
- Super fast expansion, making it the **second-ever decacorn from Turkey** in only 6 years

# Fraud Detection: Fighting Financial Crime

- Fraud Detection comes with several challenges.
- Correctly identifying changing fraud patterns in real time is very difficult
- Detecting and preventing fraud adds friction to the customer experience
- Increases in manual reviews of suspicious activity drives up staffing costs

# Why Getir need a real-time fraud detection solution?

- Business teams were manually checking data daily. Meanwhile, the cost of fraud was very high.
- These manual controls became impossible as the use of the app increased.
- The request from the Audit Team is to identify these users based on rules using analytical methods.

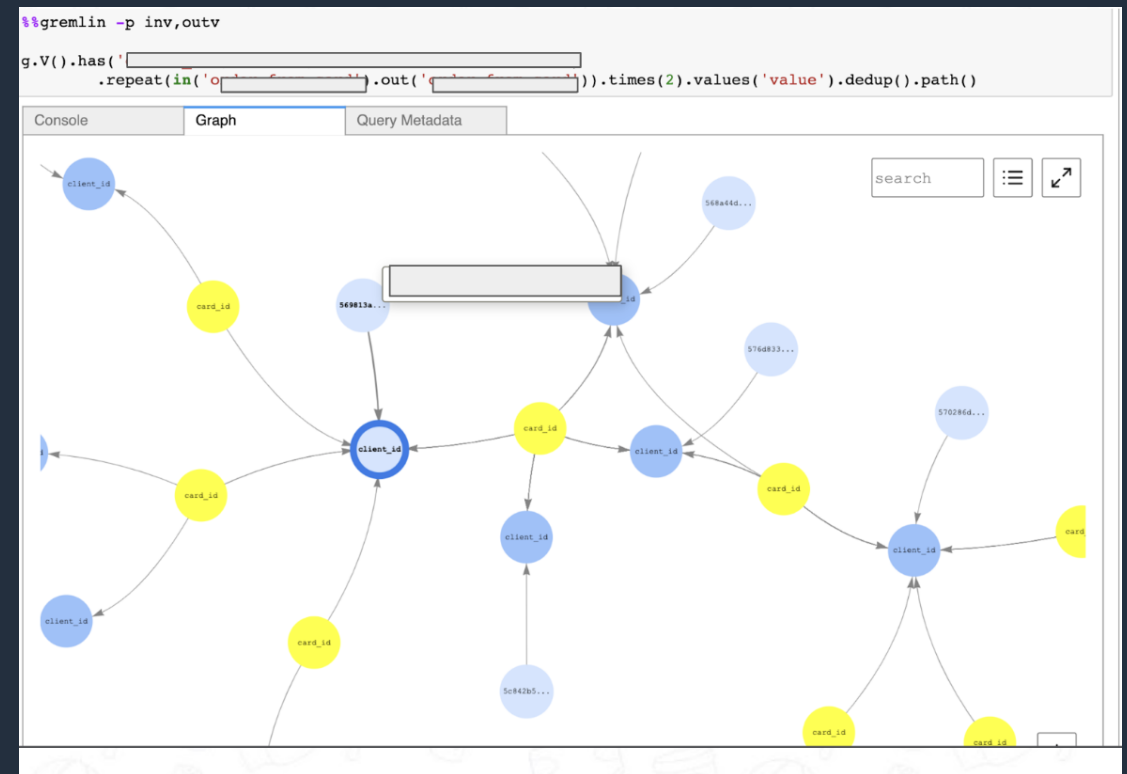
# Why Getir chose Amazon Neptune?

- Graph databases use nodes to store data entities, and edges to store relationships between entities. And this storage gives
- An edge always has a start node, end node, type, and direction, and an edge can describe parent-child relationships, actions, ownership, and the like.
- There is no limit to the number and kind of relationships a node can have.
- Amazon Neptune provides us with a much more effective environment for detecting fraud circles compared to Relational Databases. With SQL, we get our result in minutes, but using Amazon Neptune with Gremlin Graph Query Language, we get our answer in milliseconds.

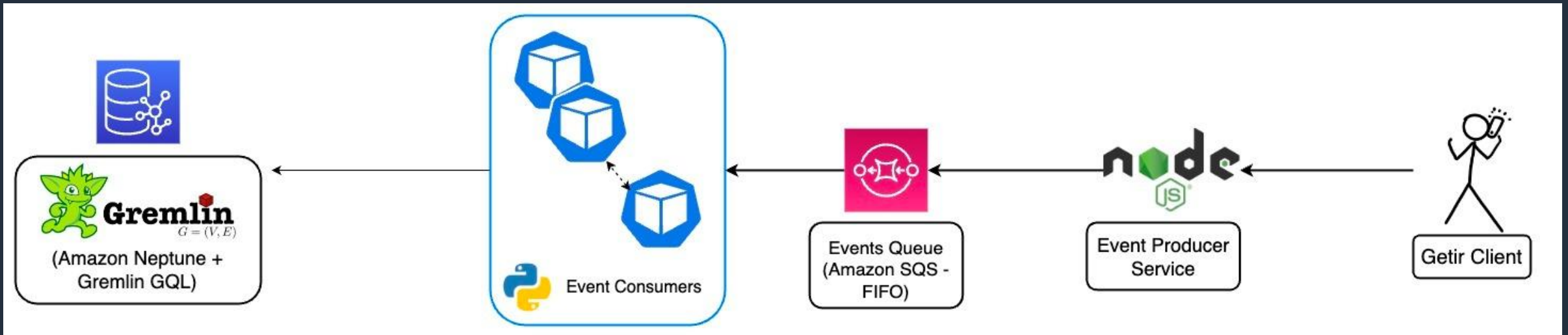


# Why Getir chose Amazon Neptune?

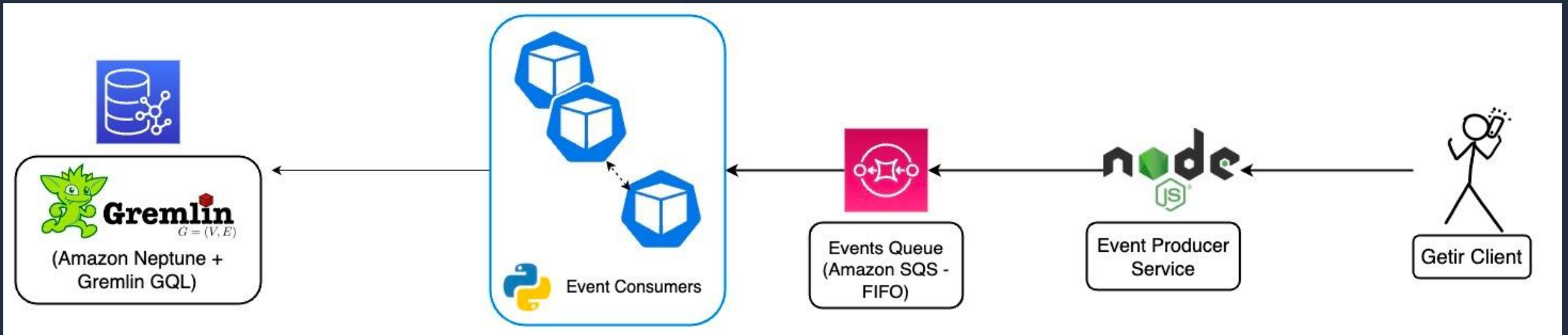
- Amazon Neptune is a fully managed database service built for the cloud that makes it easier to build and run graph applications.
- Neptune provides built-in security, continuous backups, serverless compute, and integrations with other AWS services.



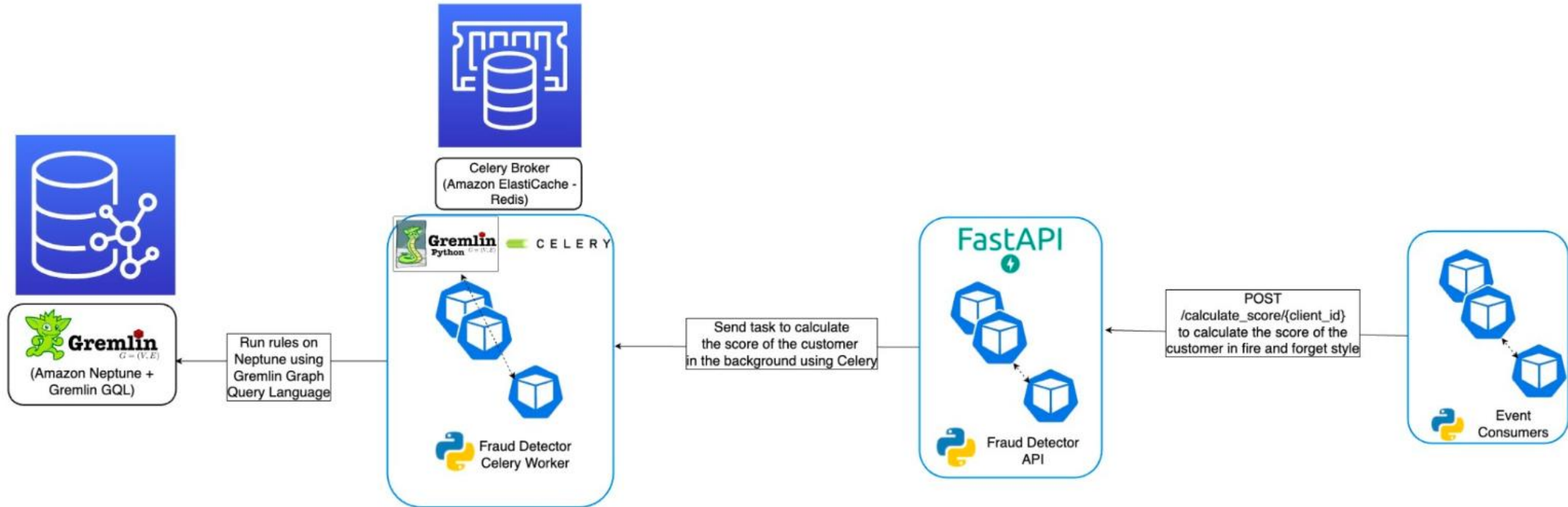
# Getir's fraud detection solution



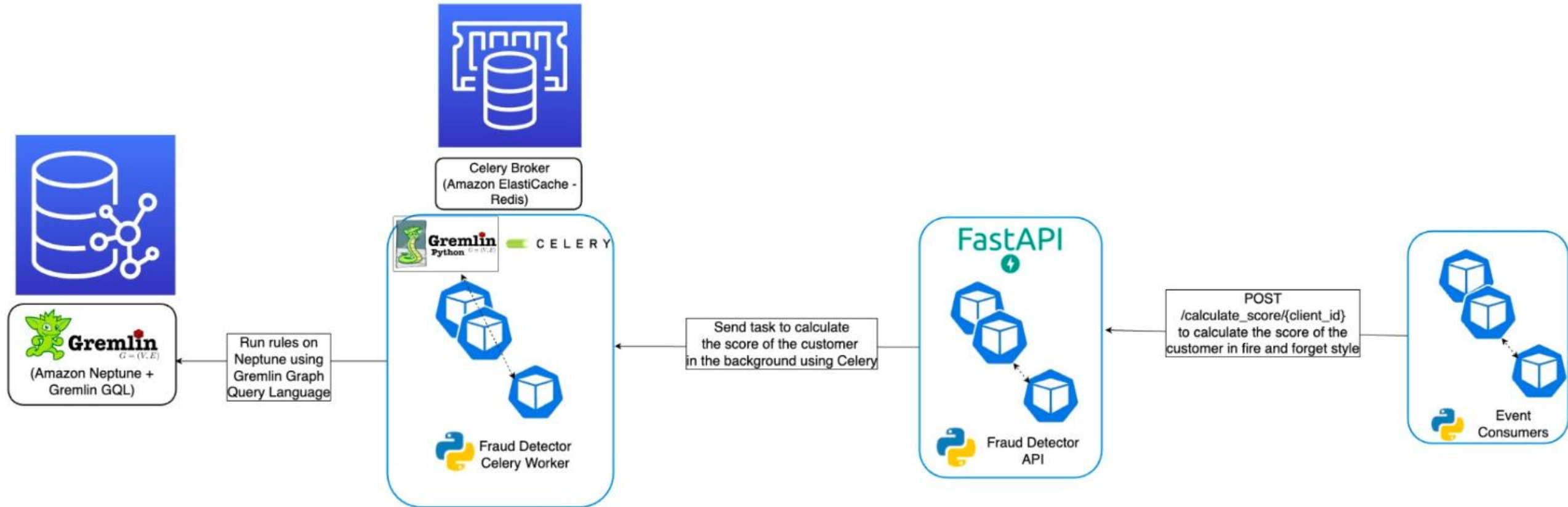
# Getir's fraud detection solution



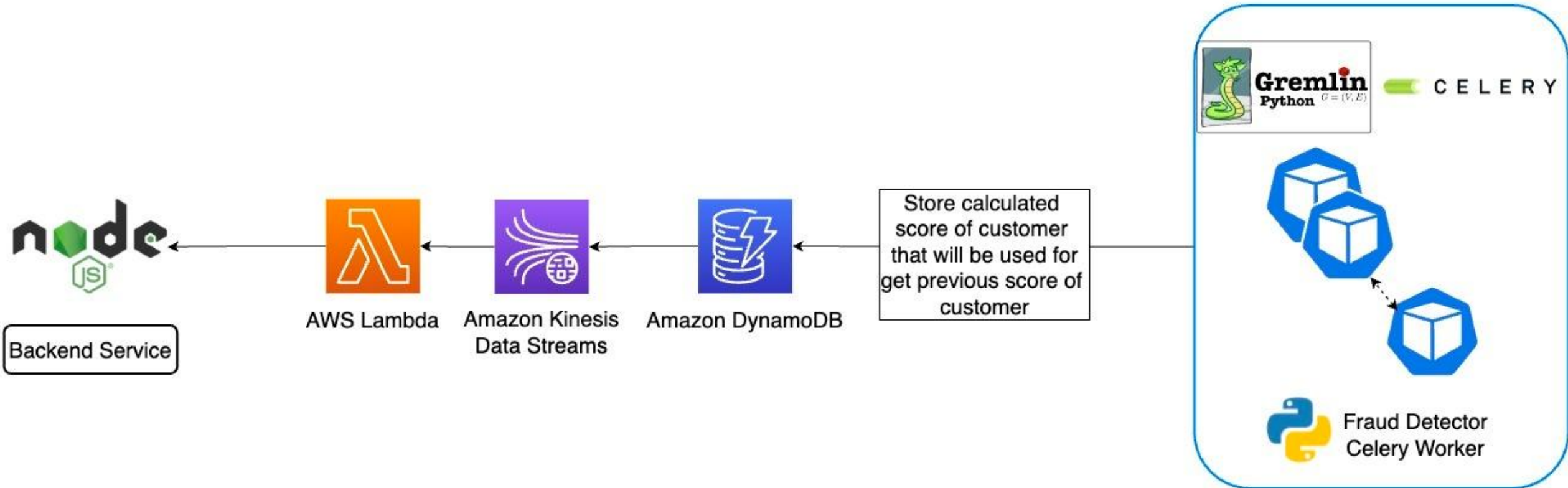
# Getir's fraud detection solution



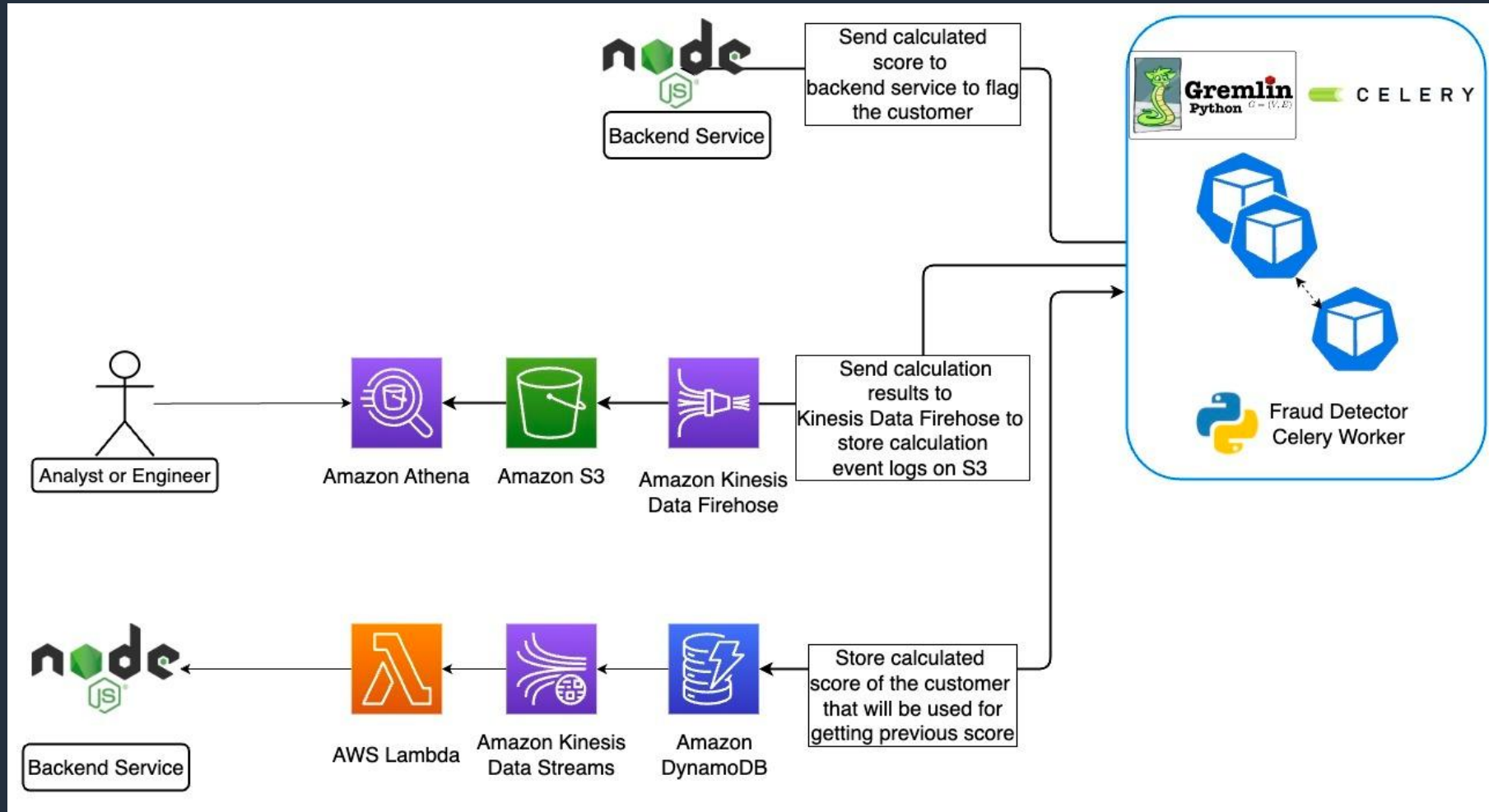
# Getir's fraud detection solution



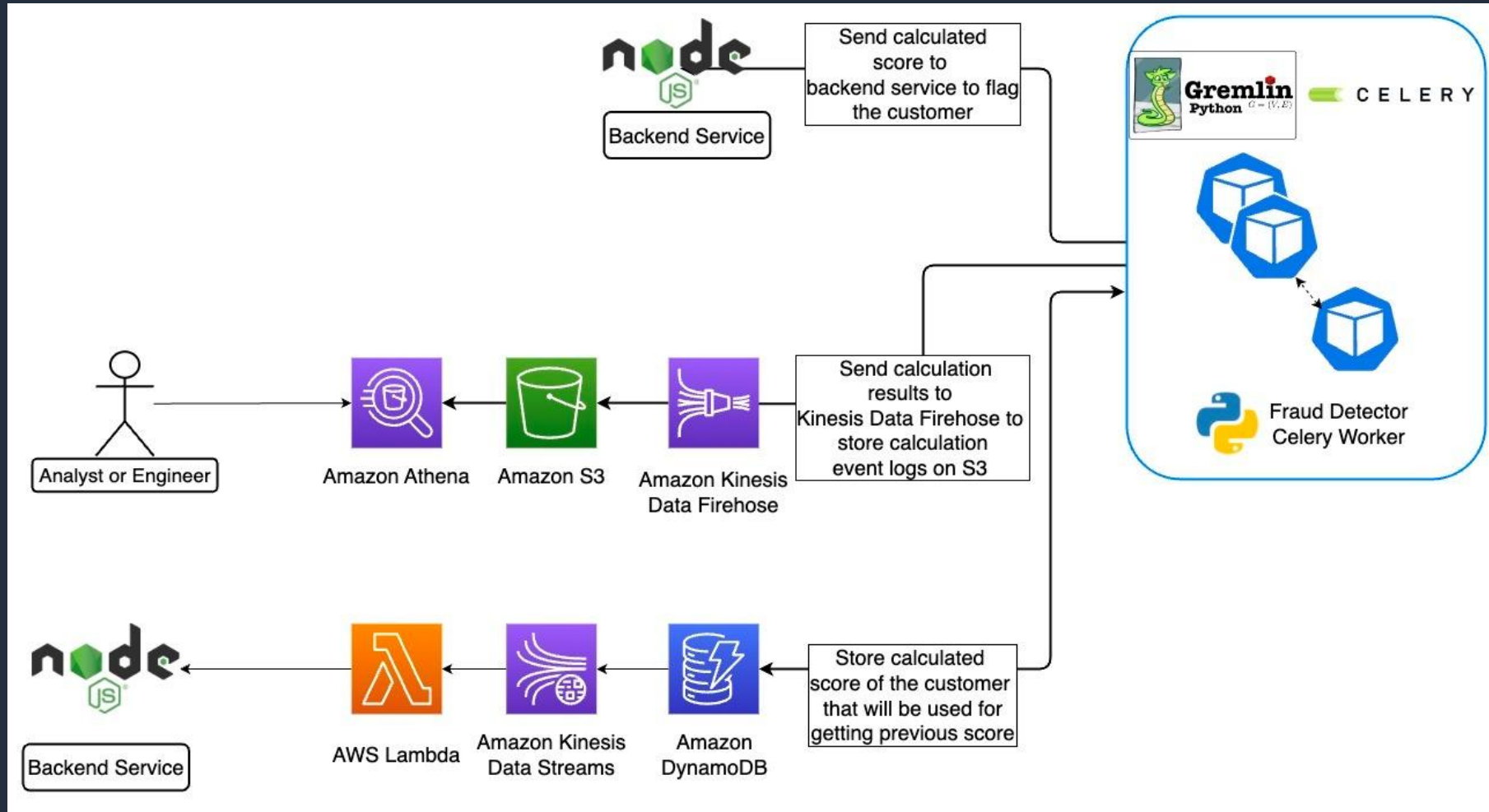
# Getir's fraud detection solution



# Getir's fraud detection solution



# Getir's fraud detection solution





# Demo



# Demo



Services

neptune



## Unified Se

Manage settings for

### Localization

Choose the language

Language

Browser default

Services (2)

Features (11)

Resources **New**

Documentation (3,263)

Knowledge Articles (18)

Marketplace (45)

Search results for 'neptune'

Try searching with longer queries for more relevant results

## Services

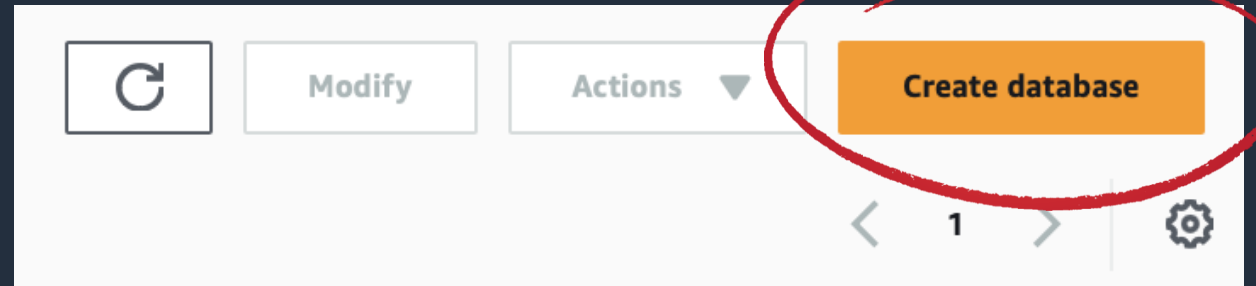
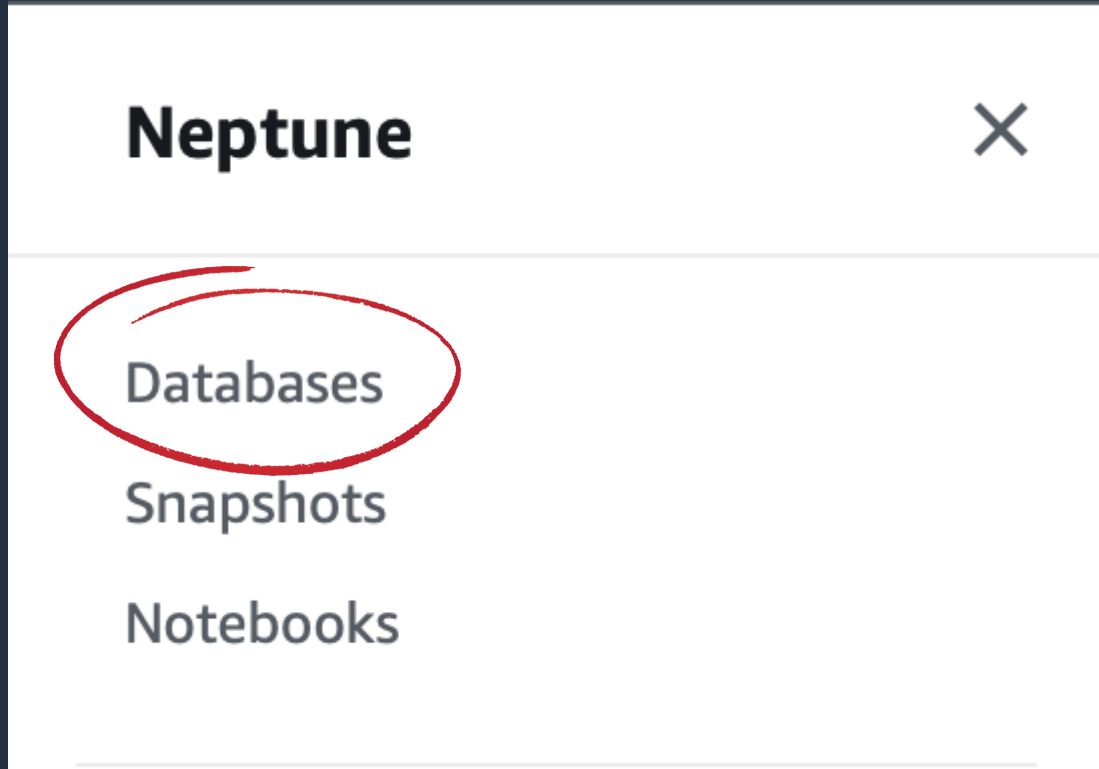


Neptune ☆

Fast, reliable graph database built for the cloud



# Demo



# Demo

Neptune > Databases > Create Database

## Create database

**Engine options**

Engine type

neptune

Provisioned

Serverless

Engine version [Info](#)

View the engine versions that support the following database features.

► Show filters

Version [Info](#)

Neptune 1.2.1.0.R6

**Settings**

**DB cluster identifier** [Info](#)

Type a name for your DB cluster. The name must be unique across all DB clusters owned by your AWS account in the current AWS Region.

fraud-detection

The DB cluster identifier is case-insensitive, but is stored as all lowercase (as in "mydbcluster"). Constraints: 1 to 60 alphanumeric characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphen.

**Templates**

Choose a template to meet your use case.

**Production**  
Use defaults for high availability and fast, consistent performance.

**Development and Testing**  
This instance is intended for development use outside of a production environment.

## DB instance size

**DB instance class** [Info](#)

Choose a DB instance class that meets your processing power and memory requirements. The DB instance class options below are limited to those supported by the engine you selected above.

Memory Optimized classes (includes r classes)

db.r6g.xlarge  
4 vCPUs 32 GIB RAM EBS: 4750 Mbps

Include previous generation classes

## Availability & durability

**Multi-AZ deployment** [Info](#)

Create read replica in different zone

No

## Connectivity

**Virtual Private Cloud (VPC)** [Info](#)

VPC that defines the virtual networking environment for this DB cluster.

Default VPC (vpc-09e98451d5576fcc1)

Only VPCs with a corresponding DB subnet group are listed.

**After a database is created, you can't change the VPC selection.**

► Additional connectivity configuration

# Demo

**Additional configuration**  
Database options, encryption enabled, failover, backup enabled, backtrack disabled, maintenance, CloudWatch Logs, delete protection enabled

**Database options**

DB instance identifier [Info](#)  
fraud-detection-db-1  
If you do not provide one, a default identifier based on the cluster identifier will be used.

DB cluster parameter group [Info](#)  
default.neptune1.2

DB parameter group [Info](#)  
default.neptune1.2

IAM DB authentication [Info](#)  
 Enable IAM DB authentication  
Manage your database user credentials through AWS IAM users and roles.

Failover priority  
No preference

**Backup**  
Creates a point in time snapshot of your database

Backup retention period [Info](#)  
Choose the number of days that Neptune should retain automatic backups for this instance.  
1 day

Copy logs to CloudWatch Logs

**Encryption**

Enable encryption  
Choose to encrypt the given instance. Master key ids and aliases appear in the list after they have been created using the Key Management Service (KMS) console. [Info](#)

Master key [Info](#)  
(default) aws/rds

**Log exports**  
Select the log types to publish to Amazon CloudWatch Logs

Audit log  
 Slow query log

**IAM role**  
The following service-linked role is used for publishing logs to CloudWatch Logs.

**RDS Service Linked Role**

**Info** Audit logging must be enabled for logs to be published in CloudWatch Logs. Please set the neptune\_enable\_audit\_log parameter to enable (1) in the parameter group that is used in this database cluster.

**Maintenance**  
Auto minor version upgrade [Info](#)

Enable auto minor version upgrade  
Enabling auto minor version upgrade will automatically upgrade to new minor versions as they are released. The automatic upgrades occur during the maintenance window for the database.

**Maintenance window** [Info](#)  
Select the period you want pending modifications or maintenance applied to the database.

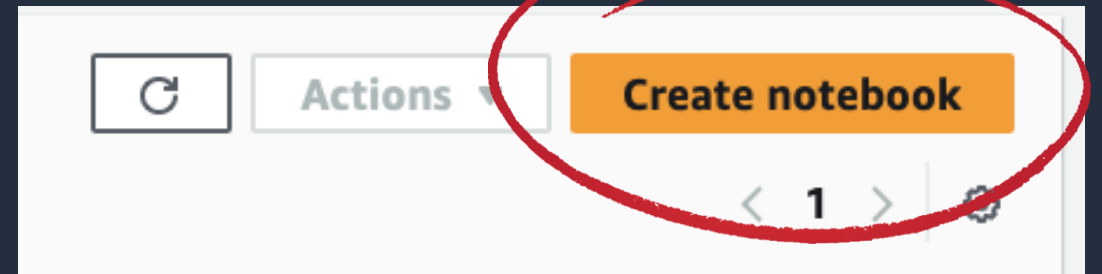
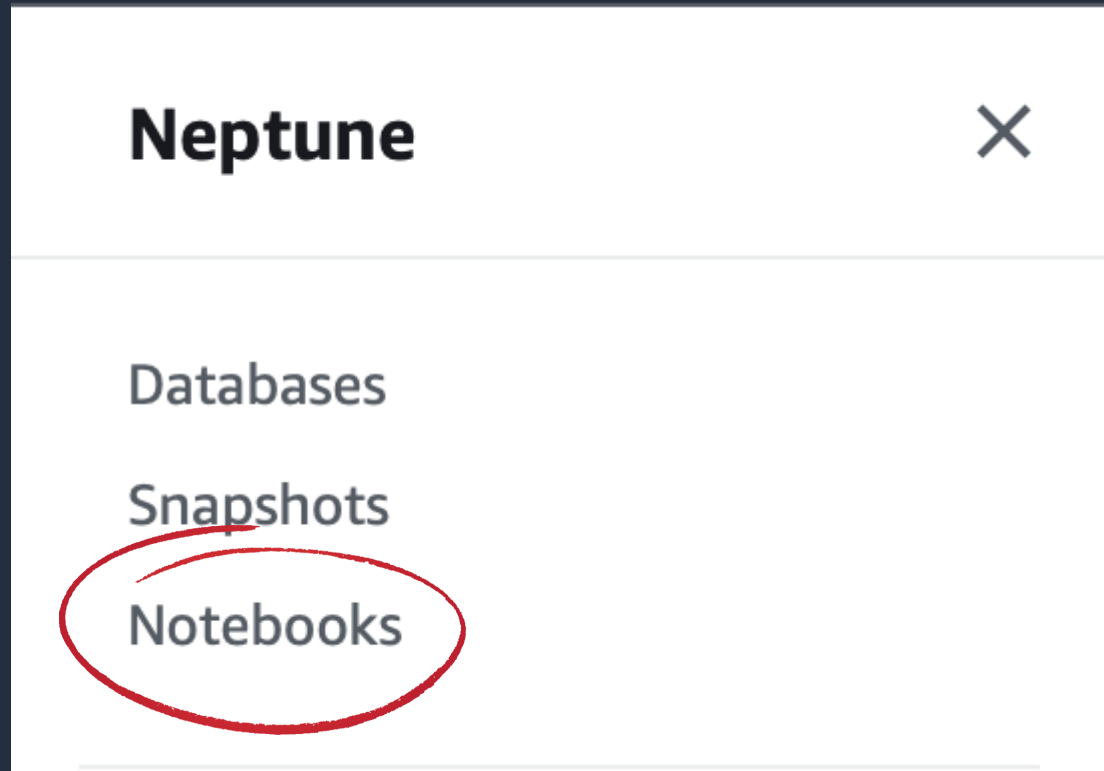
Select window  
 No preference

**Deletion protection**

Enable deletion protection  
Protects the database from being deleted accidentally. While this option is enabled, you can't delete the database.

Cancel **Create database**

# Demo



# Demo

Neptune > Notebooks > Create Notebook

### Create notebook

Create a Jupyter notebook to easily query your Neptune database. Jupyter notebooks are hosted and billed through SageMaker at your standard SageMaker usage rates. [Learn more](#)

**Notebook configuration**

**Cluster**  
Associate this notebook with an existing cluster. If no cluster exists, create a cluster  
fraud-detection

**Notebook instance type**  
ml.t3.medium

**Notebook name**  
aws-neptune-fraud-detection-demo  
Names may only contain letters (A-Z), numbers (0-9), or hyphens (-).

**Description - optional**  
256 character max

**IAM role name**  
Notebook instances require permissions to call other services including SageMaker and S3. Create a new role or update an existing role. You can also manage roles on the [IAM console](#)  
 Create an IAM role  
 Choose an existing IAM role

**IAM role**  
AWSNeptuneNotebookRole- demo-notebook  
To create an IAM role you must have CreateRole, CreatePolicy and AttachRolePolicy permissions.

**Lifecycle configuration**  
Customize your notebook environment with default scripts and plugins. [Learn more](#)  
 Use the Neptune default configuration  
 Create a new lifecycle configuration  
 Use an existing configuration

**Network configuration**  
Manage how your notebook connects to the internet. You can edit this configuration in the Amazon SageMaker console after creating your notebook. Your notebook will use your cluster's network configuration by default.

Name	Cluster	Status
aws-neptune-fraud-detection-demo	fraud-detection	Pending

Name	Cluster	Status
aws-neptune-fraud-detection-demo	fraud-detection	Ready

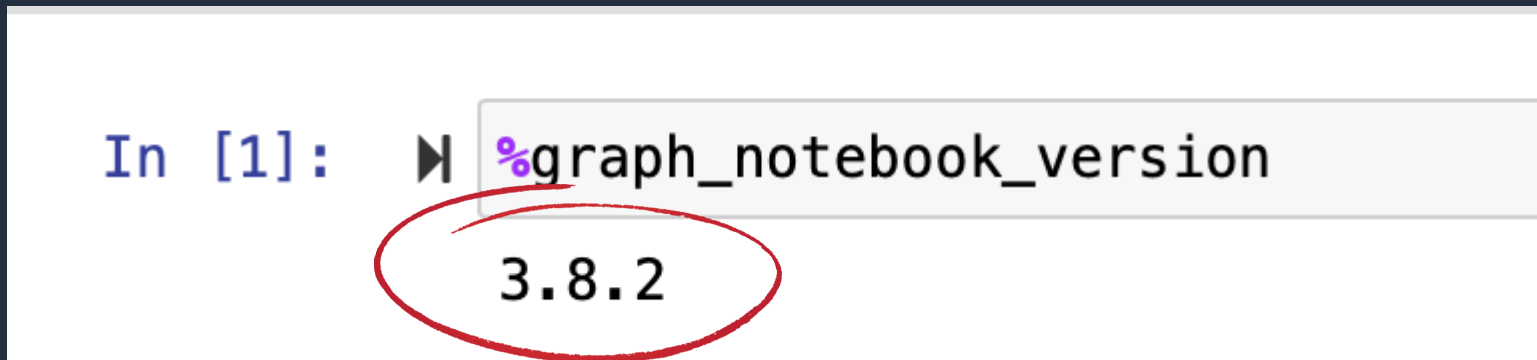
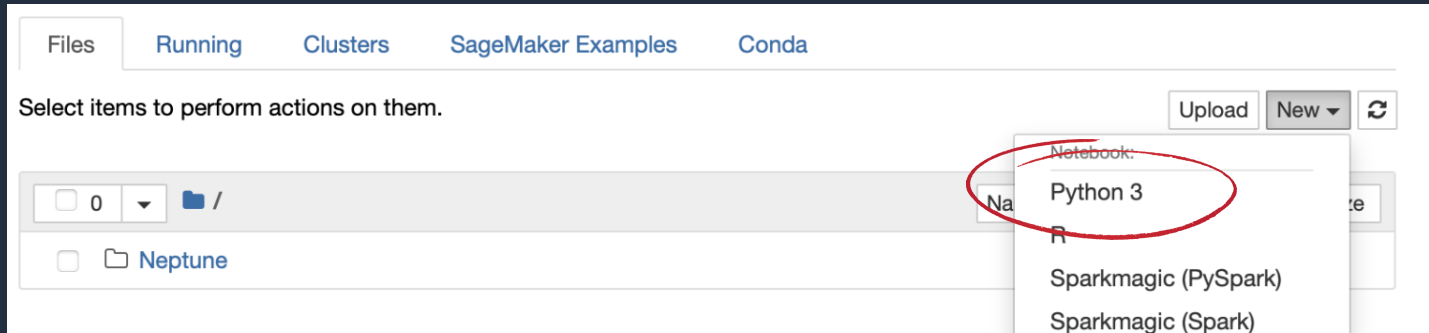
Notebooks (1/2)

Find notebooks

Name	Cluster	Status	Creation time
aws-neptune-fraud-detection-demo	fraud-detection	Ready	September 26th 2023, 2:10:21 am UTC-3 (local)

Actions: Open Jupyter, Open JupyterLab, Open Graph Explorer, Start

# Demo





# Demo

```
%%gremlin

// Add Users
g.addV('User').property('userID', 'U1').property('name', 'Alice').property('email', 'alice@example.com').next()
g.addV('User').property('userID', 'U2').property('name', 'AliceClone').property('email', 'alice2@example.com').next()

// Add Devices
g.addV('Device').property('deviceID', 'D1').property('deviceName', 'MacBookPro').next()
g.addV('Device').property('deviceID', 'D1').property('deviceName', 'MacBookPro').next()

// Add Addresses
g.addV('Address').property('addressID', 'A1').property('location', '123 Fake St').next()
g.addV('Address').property('addressID', 'A2').property('location', '124 Fake St').next()
```

Console    Query Metadata

Show 10 entries    Search:

#	Result
1	v[e2c56756-3c47-a20b-9495-230cd5c11d6f]

Showing 1 to 1 of 1 entries    Previous    1    Next

# Demo

▶ %%gremlin

```
// Add IP addresses
g.addV('IP').property('ipID', 'IP1').property('ipAddress', '192.168.1.1').next()
g.addV('IP').property('ipID', 'IP2').property('ipAddress', '192.168.1.2').next()

// Add Promotions
g.addV('Promotion').property('promoID', 'P1').property('promoCode', 'NEWUSER10').property('promoValue', '10% off')
```

Console

Query Metadata

Show 10 entries

Search:

# Result

#	Result
1	v[66c56756-b687-38e6-5638-3bcc069d082b]

# Demo

```
gremlin

// Link Users to Devices, Addresses, IPs, and Promotions
g.V().has('User', 'userID', 'U1').addE('USED_DEVICE').to(g.V().has('Device', 'deviceID', 'D1')).next()
g.V().has('User', 'userID', 'U2').addE('USED_DEVICE').to(g.V().has('Device', 'deviceID', 'D1')).next()

g.V().has('User', 'userID', 'U1').addE('HAS_ADDRESS').to(g.V().has('Address', 'addressID', 'A1')).next()
g.V().has('User', 'userID', 'U2').addE('HAS_ADDRESS').to(g.V().has('Address', 'addressID', 'A2')).next()

g.V().has('User', 'userID', 'U1').addE('USED_IP').to(g.V().has('IP', 'ipID', 'IP1')).next()
g.V().has('User', 'userID', 'U2').addE('USED_IP').to(g.V().has('IP', 'ipID', 'IP1')).next()

g.V().has('User', 'userID', 'U1').addE('CLAIMED_PROMOTION').to(g.V().has('Promotion', 'promoID', 'P1')).next()
g.V().has('User', 'userID', 'U2').addE('CLAIMED_PROMOTION').to(g.V().has('Promotion', 'promoID', 'P1')).next()
```

Console Query Metadata

Show 10 entries Search:

#	Result
1	e[14c56757-b7c4-6260-f08f-58a4bd6e2752] [1ec56756-3c29-b17d-fe20-ba8a0504e655-CLAIMED_PROMOTION->66c56756-b687-38e6-

# Demo

```
gremlin

// Link Users to Devices, Addresses, IPs, and Promotions
g.V().has('User', 'userID', 'U1').addE('USED_DEVICE').to(g.V().has('Device', 'deviceID', 'D1')).next()
g.V().has('User', 'userID', 'U2').addE('USED_DEVICE').to(g.V().has('Device', 'deviceID', 'D1')).next()

g.V().has('User', 'userID', 'U1').addE('HAS_ADDRESS').to(g.V().has('Address', 'addressID', 'A1')).next()
g.V().has('User', 'userID', 'U2').addE('HAS_ADDRESS').to(g.V().has('Address', 'addressID', 'A2')).next()

g.V().has('User', 'userID', 'U1').addE('USED_IP').to(g.V().has('IP', 'ipID', 'IP1')).next()
g.V().has('User', 'userID', 'U2').addE('USED_IP').to(g.V().has('IP', 'ipID', 'IP1')).next()

g.V().has('User', 'userID', 'U1').addE('CLAIMED_PROMOTION').to(g.V().has('Promotion', 'promoID', 'P1')).next()
g.V().has('User', 'userID', 'U2').addE('CLAIMED_PROMOTION').to(g.V().has('Promotion', 'promoID', 'P1')).next()
```

Console Query Metadata

Show 10 entries Search:

#	Result
1	e[14c56757-b7c4-6260-f08f-58a4bd6e2752] [1ec56756-3c29-b17d-fe20-ba8a0504e655-CLAIMED_PROMOTION->66c56756-b687-38e6-

# Demo

```
In [29]: %%gremlin -p outv
g.V().has('User', 'name', 'Alice').out().in().dedup().values().path()
```

Console | **Graph** | Query Metadata

The graph visualization displays a network of nodes and edges. A central yellow node labeled 'Device' is connected to several blue nodes labeled 'User'. Red nodes include 'Alice', 'U1', 'U2', 'alice2@...', and 'AliceClone'. Arrows indicate directed edges between nodes. The interface includes a search bar and various control icons.

# Demo

```
In [29]: %%gremlin -p outv
g.V().has('User', 'name', 'Alice').out().in().dedup().values().path()
```

Console | **Graph** | Query Metadata

The graph visualization displays a network of nodes and edges. The nodes are categorized by color: red nodes represent individual users or identifiers (Alice, U1, U2, alice2@..., AliceClone), blue nodes represent 'User' entities, and a yellow node represents a 'Device'. Directed edges (arrows) show relationships between these entities. A central yellow 'Device' node is connected to multiple blue 'User' nodes. Red nodes are also connected to blue 'User' nodes, with some connections being bidirectional. The interface includes a search bar and various control icons at the top right.

# What is next?



**Improve operational efficiency**



**Make more informed decisions**



**Accelerate innovation**



**getir**

# Additional resources



[Neptune notebooks/graph notebook](#)



[Neptune reference architectures](#)



[Use cases, videos, blogs, code ...](#)



[Customer success stories](#)



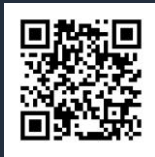
[How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB](#)



The screenshot shows a GitHub repository page for 'aws-dbs-refarch-graph/src' and an AWS Database Blog article. The article is titled 'How Getir build a comprehensive fraud detection system using Amazon Neptune and Amazon DynamoDB' and is authored by Berkay Berkman, Mahmut Turan, Mutlu Polatcan, Umut Cemal Kiraç, Yağız Yanıkoğlu, and Esra Kayabali. The article discusses how Getir built an end-to-end fraud detection system using Amazon Neptune and Amazon DynamoDB. It mentions that Neptune is a fully managed database service built for the cloud that makes it simple to build and run graph applications. It also mentions that DynamoDB is a fast and flexible non-relational database service for any scale. The article includes a video player for 'Build event driven graph applications with AWS purpose-built databases' and a video player for 'Deep dive on Amazon Neptune'.



# Additional resources



[Amazon DynamoDB](#)



[Getting Started](#)



[Amazon DynamoDB resources](#)



[Customer success stories](#)

The screenshot displays the Amazon DynamoDB product page. At the top, a diagram highlights key features: NoSQL Workbench, Global tables, Encryption at rest, Point-in-time recovery, On-demand capacity mode, and PartiQL supports. It also lists integration options like Amazon S3, AWS Glue Elastic Views, Amazon Kinesis Data Streams, AWS CloudTrail, and Amazon CloudWatch. Below this, the 'Configure Key Features' section lists built-in security, backup and restore, flexible capacity modes, multi-Region replication, in-memory caching, and data modeling tools. The main content area features a 'Getting Started with Amazon DynamoDB' article, a 'New AWS Training and Certification Courses for DynamoDB' section with a list of modules, and a 'PAGE CONTENT' sidebar with links to Training and Certification Courses, Tutorials, Virtual Workshops, Learning path, AWS Free Tier, Videos, and Additional training.





# Thank you!

Berkay Berkman

[berkay.berkman@getir.com](mailto:berkay.berkman@getir.com)

Esra Kayabali

[kayabali@amazon.com](mailto:kayabali@amazon.com)