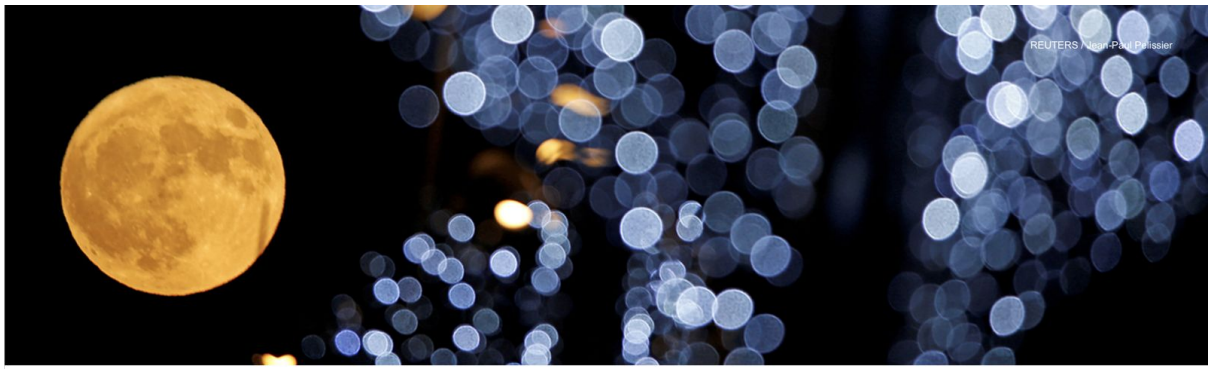


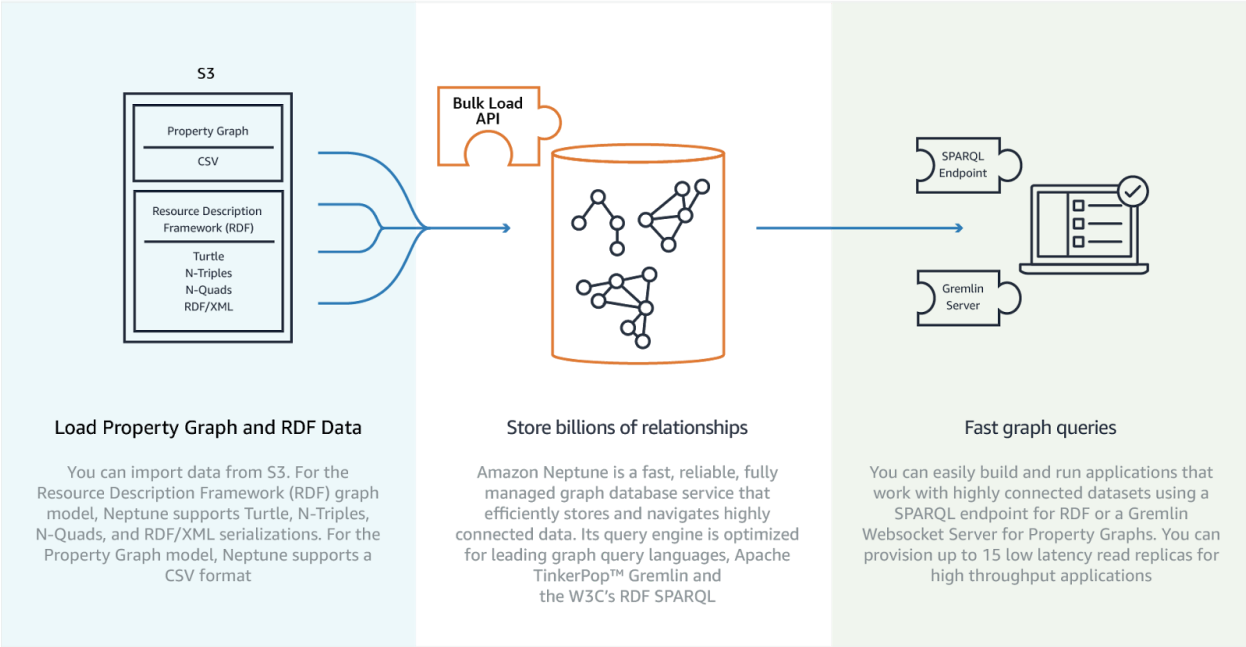
Thomson Reuters Graph Feed & Amazon Neptune

Tutorial on how to use Amazon Neptune as a graph-database alternative to explore Thomson Reuters Graph Feed.

Pratik Pandey
Jan 24th, 2018

the answer company™
THOMSON REUTERS®

End of November last year, Randall Hunt, an author and tech evangelist at Amazon, announced Amazon's foray into graph database with Neptune. Graph databases are dime a dozen, but Neptune stands out as being a fully-managed service with the ability to store billions of relationships of Property Graph and RDF data.



Neptune in a Nutshell

In this first blog post on Neptune, we will explore how we can export data from Thomson Reuters Knowledge Graph Feed and import it into Neptune. Thomson Reuters Knowledge Graph Feed is an API that delivers the Thomson Reuters Knowledge Graph. The raw size of Graph Feed export requires a graph analytics application to consume and query, and we'll see how to use Neptune for this.

1. Export the content set from Graph Feed.

2. Setup Neptune

3. Create an S3 import bucket

4. Create Endpoint

5. Launch EC2 instance in one of the subnets inside the Neptune VPC

6. SSH into the EC2 instance and initiate load command

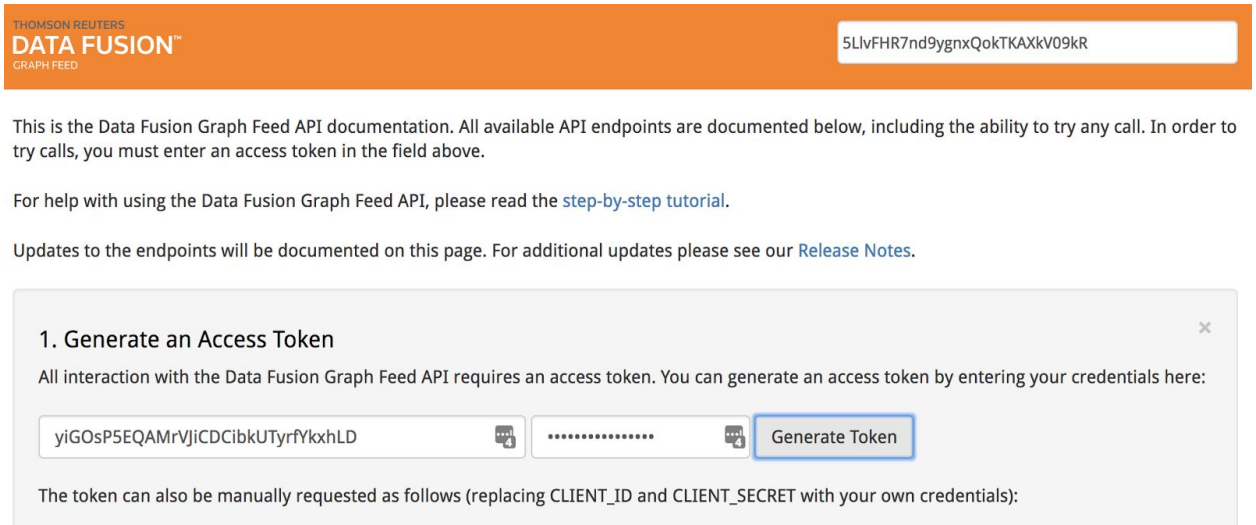
7. Validate

By the end of this tutorial, you'll export content-set from Thomson Reuters Graph Feed, spin up your own Neptune instance, load it onto Neptune and query the data.

1. Export the content set from Graph Feed.

Let's go to DFGF to download a content set that you would like to import into Neptune. You can do that in few ways (Swagger Docs, Command-line with CURL, or from a client to the API), but at the end, it all boils down to a REST call to the Graph Feed APIs.

Here we'll be using the docs. First, use the client_id and client_secret to generate a short-lived token. If you do not have a client_id and secret, please reach out to Brian Rohan. Let's go to the [Graph Feed site.]



Use /contentSet to find a list of content-set that you're entitled to.

Grab the ID of the content set you want to export and supply that to /contentSet/{id}/download API endpoint. The Response here is a short-lived URL to download the content set. Go ahead and start the download while we setup Neptune.

2. Setup Neptune

At this time of this writing, Neptune is only available in the us-east-1 region with a request for a preview, which you can do here. https://pages.awscloud.com/NeptunePreview.html. Once you have access to Neptune you can setup Neptune service in AWS (https://**yukon**.aws.amazon.com/rds/gdb?region=us-east-1). Notice that this is not the regular console URL for AWS. Once you login to the yukon.aws console, setting up Neptune is a quick two-step process. First specify database details.

Neptune > Instances > Launch DB instance

Specify DB details

Instance specifications

Estimate your monthly costs for the DB Instance using the AWS Simple Monthly Calculator

DB engine

neptune

DB instance class info

db.r4.4xlarge — 16 vCPU, 122 GiB RAM

db.r4.4xlarge — 16 vCPU, 122 GiB RAM

db.r4.8xlarge — 32 vCPU, 244 GiB RAM

Settings

DB instance identifier info

Specify a name that is unique for all DB instances owned by your AWS account in the current region.

dfgf-to-neptune

DB instance identifier is case insensitive, but stored as all lower-case, as in "mydbinstance".
Constraints:

- Must contain from 1 to 63 alphanumeric characters or hyphens.
- First character must be a letter.
- Cannot end with a hyphen or contain two consecutive hyphens.

Cancel Next

Second in the *Configure Advanced Settings* make selection of your choices on VPC, backups, security etc. I went with creating a new VPC and choose defaults for other choices.

Step 1
Specify DB details

Step 2
Configure advanced settings

Neptune > Instances > Launch DB instance

Configure advanced settings

Network & Security

Virtual Private Cloud (VPC) info

VPC defines the virtual networking environment for this DB instance.

Create new VPC

Only VPCs with a corresponding DB subnet group are listed.

Subnet group info

DB subnet group that defines which subnets and IP ranges the DB instance can use in the VPC you selected.

Create new DB Subnet Group

Availability zone info

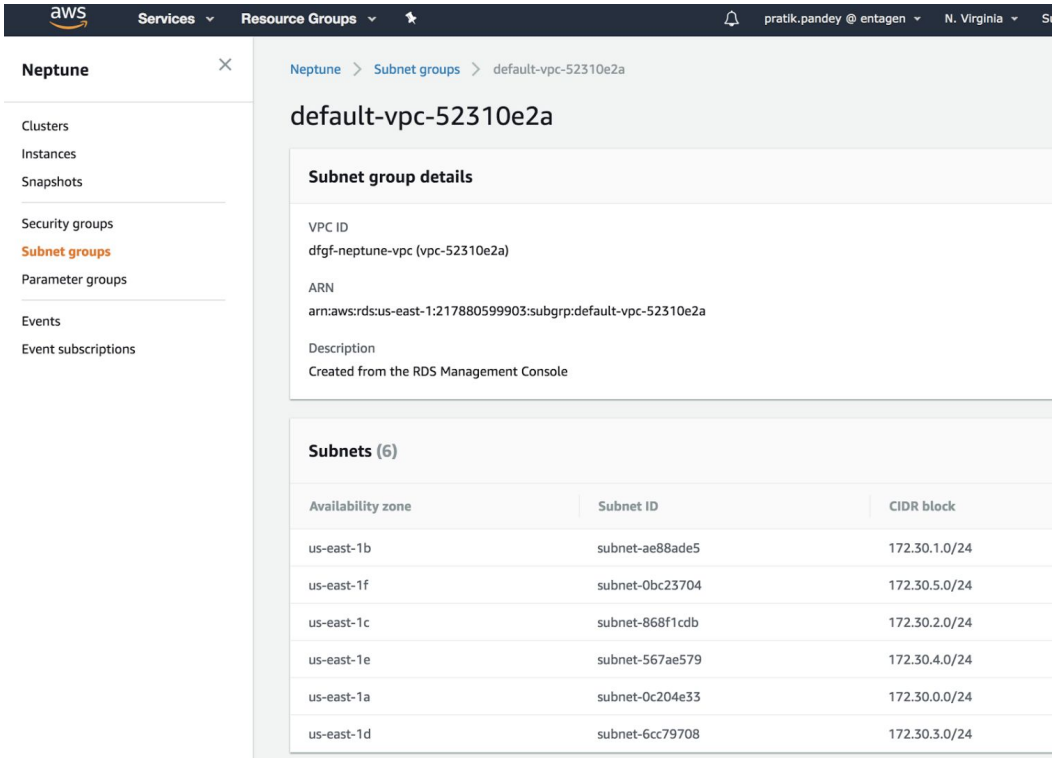
No preference

VPC security groups

Security groups have rules authorizing connections from all the EC2 instances and devices that need to access the DB instance.

- (●) Create new VPC security group
- ( ) Select existing VPC security groups

Once you configure everything and hit submit AWS will take a few minutes to instantiate the cluster. If everything is successful you will see the VPC and all the subnets Neptune cluster created.



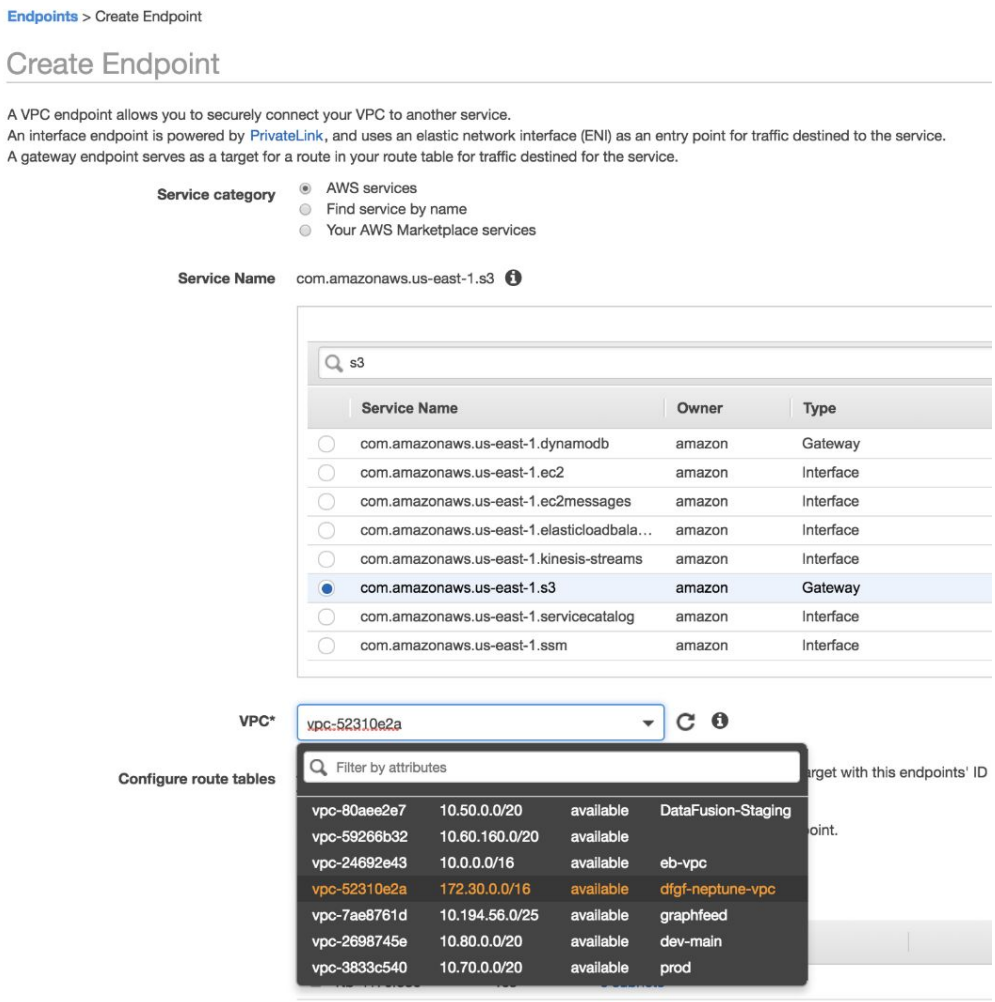
Now we will need to do few things in AWS so set up a pipeline to bulk import data into Neptune instance. Although in this tutorial we're focused on importing data from Graph Feed, this setup is required for bulk upload from any data source into Neptune.

3. Create an S3 import bucket

Create a new S3 bucket. Make sure you pick US East region. Once you have the bucket setup upload the file you downloaded from GraphFeed.

4. Create Endpoint

Creating an endpoint simplifies access to S3 resources from within a VPC and provides a secure connection to S3 that does not require a gateway or NAT instances. You can access endpoint configuration under VPC services. Make sure you choose com.amazonaws.**us-east**.s3 for service and the Neptune VPC you created earlier.



5. Launch EC2 instance in one of the subnets inside the Neptune VPC

Create an EC2 instance in one of the subnets inside the Neptune VPC. The only thing to make sure here is that you pick the VPC that Neptune created.

AMI Details — Edit AMI

Amazon Linux AMI 2017.09.1 (HVM), SSD Volume Type - ami-97785bed

Free tier eligible

The Amazon Linux AMI is an EBS-backed, AWS-supported image. The default image includes AWS command line tools, Python, Ruby, Perl, and Java. The repositories include Docker, PHP, MySQL, PostgreSQL, and other packages.

Root Device Type: ebs Virtualization type: hvm

Instance Type — Edit instance type

Instance Type	ECUs	vCPUs	Memory (GiB)	Instance Storage (GB)	EBS-Optimized Available	Network Performance
t2.micro	Variable	1	1	EBS only	-	Low to Moderate

Security Groups — Edit security groups

Instance Details — Edit instance details

Number of instances	1
Network	vpc-52310e2a
Subnet	subnet-0c204e33
EBS-optimized	No
Monitoring	No
Termination protection	No
Shutdown behavior	Stop
IAM role	None
Tenancy	default
T2 Unlimited	Disabled
Host ID	
Affinity	Off
Kernel ID	Use default
RAM disk ID	Use default
User data	
Assign Public IP	Use subnet setting (Enable)
Purchasing option	On demand

Cancel Previous Launch

6. SSH into the EC2 instance and initiate load command

Now it is just a matter of running a CURL with a POST command in the ec2 instance to load data into Neptune. Make sure to replace your own Neptune Endpoint, source, format, and credentials. If all goes well, you'll see a HTTP 200 response and a loadId in the payload.

```
curl -X POST \
    -H 'Content-Type: application/json' \
    https://your_neptune_endpoint:8182/loader -d '
    {
      "source" : "s3://graphfeed-neptune/55.cache.gz",
      "format" : "ntriples",
      "accessKey" : "your_access_key",
      "secretKey" : "your_secret",
      "region" : "us-east-1",
      "failOnError" : "FALSE"
    }'
```

```
}[ec2-user@ip-172-30-0-119 ~]$ ./neptune-loader.sh
{
    "status" : "200 OK",
    "payload" : {
        "loadId" : "91c29b70-fad5-4474-81ed-941f96c022de"
    }
```

7. Validate

Once you see the response with loadId, you can use that to check the status of the import with a GET call to Neptune endpoint. As you see here in the following screenshot, there were 7,225,094 total records in this content set.

```
[ec2-user@ip-172-30-0-119 ~]$ curl -X GET http://neptune.cluster-cubddbuiu6ju.us-east-1-beta
2/loader/91c29b70-fad5-4474-81ed-941f96c022de
{
    "status" : "200 OK",
    "payload" : {
        "feedCount" : [
            {
                "LOAD_COMPLETED" : 1
            }
        ],
        "overallStatus" : {
            "fullUri" : "https://s3.amazonaws.com/graphfeed-neptune/55.cache.txt",
            "runNumber" : 1,
            "retryNumber" : 0,
            "status" : "LOAD_COMPLETED",
            "totalTimeSpent" : 78,
            "totalRecords" : 7225094,
            "totalDuplicates" : 0,
            "parsingErrors" : 0,
            "datatypeMismatchErrors" : 0,
            "insertErrors" : 0
        }
    }
```

You can now use SPARQL to query your data with an HTTP POST.

```
~/dev  ssh -i "neptune-uploader.pem" ec2-user@ec2-54-84-79-122.compute-1.amazonaws.com

Warning: Identity file neptune-uploader.pem not accessible: No such file or directory.
Last login: Mon Jan 22 16:42:18 2018 from 70.123.51.50

       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-ami/2017.09-release-notes/
2 package(s) needed for security, out of 2 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-30-0-119 ~]$ curl -X POST --data-binary 'query=select ?s ?p ?o where {?s ?p ?o} limit 10' http://neptune.clu
dbuiu6ju.us-east-1-beta.rds.amazonaws.com:8182/sparql
{
  "head" : {
    "vars" : [ "s", "p", "o" ]
  },
  "results" : {
    "bindings" : [ {
      "s" : {
        "type" : "uri",
        "value" : "http://data.thomsonreuters.com/sc/snippet/4295859594_5037971263_141_134_nL5N1FN76X-2017-02-07"
      },
      "p" : {
        "type" : "uri",
        "value" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#type"
      },
      "o" : {
        "type" : "uri",
        "value" : "http://ontology.thomsonreuters.com/supplyChain#Snippet"
      }
    }, {
```