KOENIG
step forward

Monitoring Neo4J

Monitoring in Neo4J

- Authentication events are written to the security.log file
- Event categories that we monitor in log files include:
  - Queries
  - Transactions
  - Connections
  - Memory

Collecting metrics

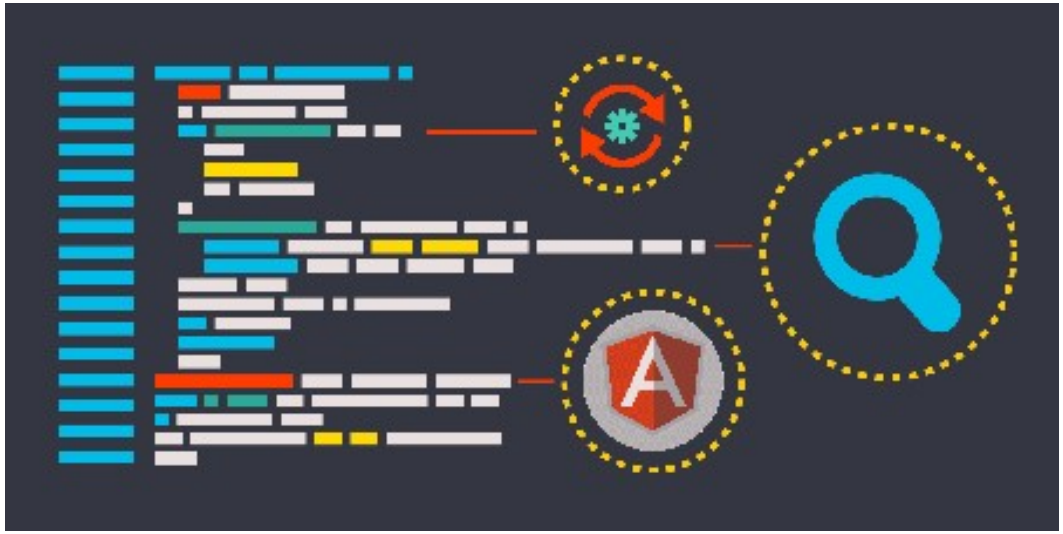
- Can configure to collect metrics that are related to events
- Can be viewed in tools - such as Grafana
- Can configure a tool such as Nagios to provide alerts when certain metrics are detected in Neo4j

Monitoring queries

- Write information about queries to the query.log file
- Log information about queries that took a long time to complete
- Can also monitor currently running queries and if need be, kill them if they are taking too long.

Configuring query logging

- Should enable logging for queries
- Set a threshold for the length of time queries take
- Regularly inspect the query.log file
- Determine if queries are taking longer duration



Configuring query logging

- Examples:
  - Options: [OFF, INFO, VERBOSE]
  - dbms.logs.query.enabled=VERBOSE

  - #If the execution of query takes more time than this threshold, the query is logged once completed
  - dbms.logs.query.threshold=1000ms

  - #Log parameters for the executed queries being logged
  - dbms.logs.query.parameter_logging_enabled=true

  - #Log detailed time information for the executed queries being logged
  - dbms.logs.query.time_logging_enabled=true

Configuring query logging

KOENIG
step forward

- Examples:
  - #Log amount of total allocated bytes for the executed queries being logged. The logged number is cumulative over the duration of the query
  - dbms.logs.query.allocation_logging_enabled=true

  - #Log page hits and page faults for the executed queries being logged.
  - dbms.logs.query.page_logging_enabled=true

  - #Enables or disables tracking of how much time a query spends actively executing on the CPU
  - dbms.track_query_cpu_time=true

  - #Enables or disables tracking of how many bytes are allocated by the execution of a query.
  - If enabled, calling dbms.listQueries will display the allocated bytes
  - dbms.track_query_allocation=true

Dynamic settings

KOENIG
step forward

- All the above listed settings are dynamic.
- To view the list of settings which can be set dynamically:
  - CALL dbms.listConfig()
  - YIELD name, dynamic
  - WHERE dynamic
  - RETURN name
  - ORDER BY name
- To configure dynamic setting:
  - CALL dbms.setConfigValue('dbms.logs.query.enabled', 'info')

Read the logs

- tail -f -n 50 /logs/query.log

Create some nodes

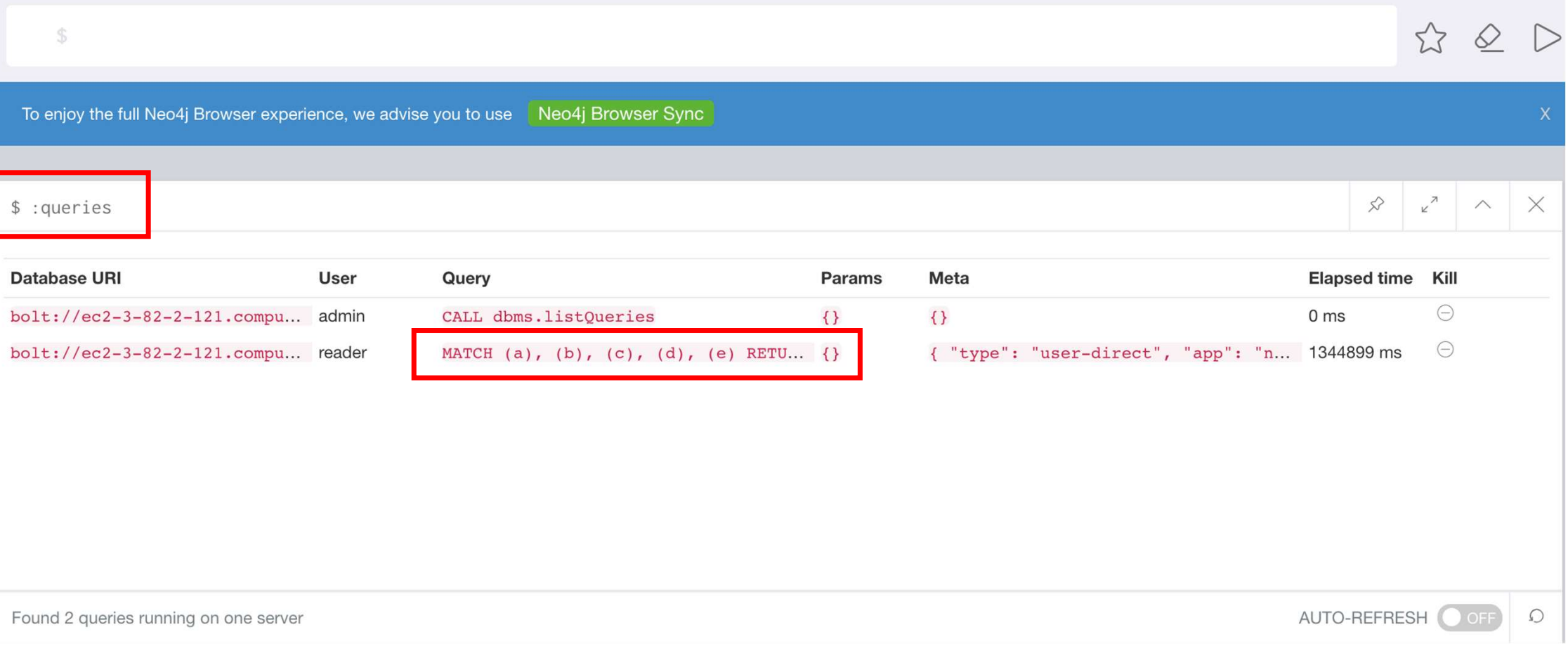
- cypher-shell -u neo4j -p secret -d neo4j --format plain
  - :use neo4j;
  - MATCH (n) DETACH DELETE n;
  - #Create some records
  - FOREACH (i IN RANGE(1,200) | CREATE (:Person {name:'Person' + i}));

Exercise: Monitoring queries

- cypher-shell -u neo4j -p secret -d neo4j --format plain
  - #Execute a query that runs for longer than 1000 ms:
  - MATCH (a), (b), (c), (d) RETURN count(id(a));

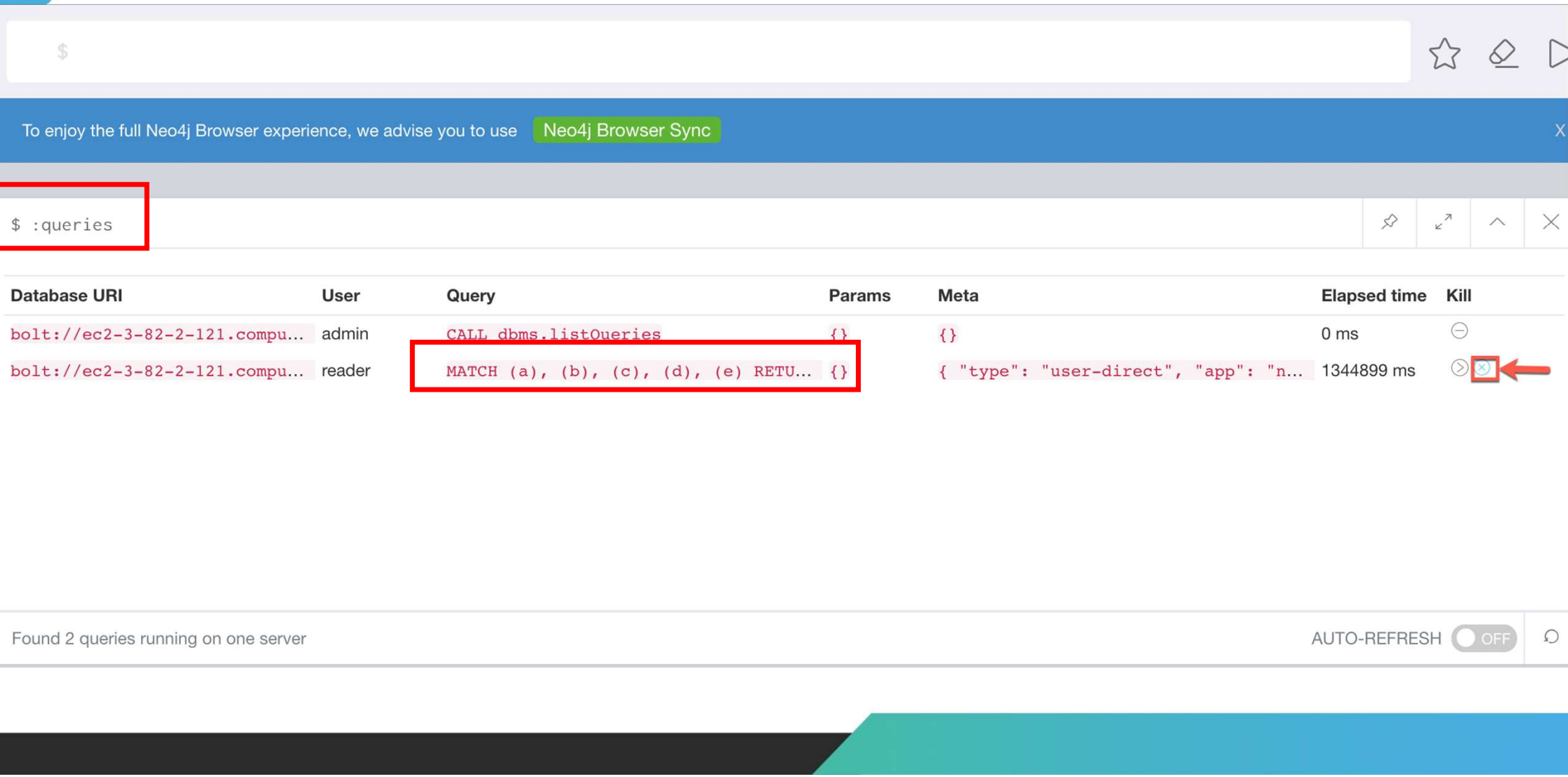
Viewing currently running queries

KOENIG
step forward



Killing a long-running query

KOENIG
step forward



Exercise #1: Monitoring queries

KOENIG
step forward

- Modify the neo4j.conf file to create a log record if a query exceeds 1000 ms
- vim /var/lib/neo4j/conf/neo4j.conf
  - dbms.logs.query.enabled=VERBOSE
  - dbms.logs.query.threshold=1000ms
  - dbms.logs.query.parameter_logging_enabled=true
  - dbms.logs.query.time_logging_enabled=true
  - dbms.logs.query.allocation_logging_enabled=true
  - dbms.logs.query.page_logging_enabled=true
  - dbms.track_query_cpu_time=true
  - dbms.track_query_allocation=true
- neo4j restart

Exercise: Monitoring queries

- cypher-shell -u neo4j -p secret -d neo4j --format plain
  - #Execute a query that runs for longer than 1000 ms:
  - MATCH (a), (b), (c), (d) RETURN count(id(a));

Exercise: Monitoring queries

KOENIG
step forward

- Wait about a minute, it should complete.

```
Terminal
ubuntu@ip-172-31-24-255:~$ /usr/bin/cypher-shell --format plain
username: reader
password: ******
neo4j> MATCH (a), (b), (c), (d) RETURN count(id(a));
count(id(a))
855036081
neo4j>
```

- View the query.log. Is there a record for this query?

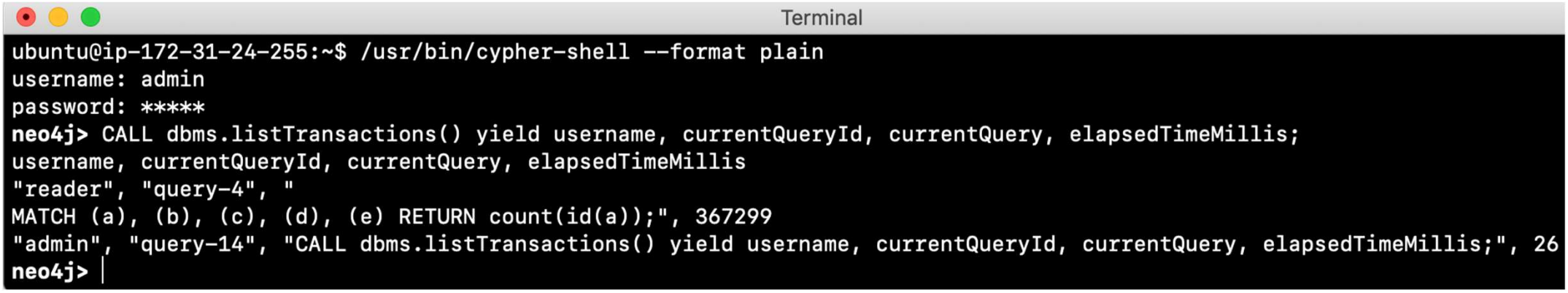
```
ubuntu@ip-172-31-24-255:/var/log/neo4j$ sudo systemctl restart neo4j
ubuntu@ip-172-31-24-255:/var/log/neo4j$ tail query.log
2019-02-15 18:33:48.069+0000 INFO  1929 ms: (planning: 1898, cpu: 1812, waiting: 0) - 260597464 B - 0 page hits, 0 page faults - embedd
ed-session               - MATCH (a:` This query is just used to load the cypher compiler during warmup. Please ignore `) RETURN a LIM
IT 0 - {} - {}
2019-02-15 18:35:45.642+0000 INFO  26738 ms: (planning: 176, cpu: 26729, waiting: 0) - 26865112 B - 29415 page hits, 0 page faults - bo
lt-session       bolt    reader  neo4j-java/dev          client/127.0.0.1:39750      server/127.0.0.1:7687>  reader - MATCH (a), (b), (
c), (d) RETURN count(id(a)); - {} - {}
ubuntu@ip-172-31-24-255:/var/log/neo4j$
```

Exercise #1: Monitoring queries

KOENIG step forward

- In cypher-shell session, enter query that will execute for an even longer time
  - MATCH (a), (b), (c), (d), (e) RETURN count(id(a));
- Open a new terminal window and log in to cypher-shell
- Execute the Cypher statement to list transactions
- Do you see the query?

CALL dbms.listTransactions() yield username, currentQueryId, currentQuery, elapsedTimeMillis;



Exercise #1: Monitoring queries

KOENIG
step forward

- Execute the Cypher statement to kill the long-running query.
  - CALL dbms.listTransactions() yield username, currentQueryId, currentQuery, elapsedTimeMillis;
  - CALL dbms.killQuery('query-id');

```
neo4j> CALL dbms.listTransactions() yield username, currentQueryId, currentQuery, elapsedTimeMillis;
username, currentQueryId, currentQuery, elapsedTimeMillis
"reader", "query-4", "
MATCH (a), (b), (c), (d), (e) RETURN count(id(a));", 367299
"admin", "query-14", "CALL dbms.listTransactions() yield username, currentQueryId, currentQuery, elap
neo4j> CALL dbms.killQuery('query-4');
queryId, username, message
"query-4", "reader", "Query found"
neo4j>
```

- Observe in other session that the query has been killed.

Automating monitoring of queries

KOENIG
step forward

- Automate the killing of long-running queries:
  - CALL dbms.listQueries() YIELD query, elapsedTimeMillis, queryId, username
  - WHERE NOT query CONTAINS toLower('LOAD')
  - AND elapsedTimeMillis > 1000
  - WITH query, collect(queryId) AS q
  - CALL dbms.killQueries(q) YIELD queryId
  - RETURN query, queryId;

Configuring transaction guard

KOENIG
step forward

- Example of lock acquisition timeout and transaction guard
- vim /var/lib/neo4j/conf/neo4j.conf
  - # transaction guard: max duration of any transaction
  - dbms.transaction.timeout=1s
  - # max time to acquire write lock
  - dbms.lock.acquisition.timeout=10ms
- neo4j restart
- Enter this statement to create multiple Person nodes:
  - FOREACH (i IN RANGE(1,1000000) | CREATE (:Person {name:'Person' + i}));
- Do you receive an error?
  - A record is written to the debug.log file

Configuring transaction guard

- View the record written to debug.log

```
ubuntu@ip-172-31-24-255:/var/log/neo4j$ tail -n 2 debug.log
2019-02-15 23:08:31.803+0000 INFO [o.n.i.d.DiagnosticsManager] --- SERVER STARTED END ---
2019-02-15 23:08:58.522+0000 WARN [o.n.k.i.a.t.m.KernelTransactionMonitor] Transaction KernelTransactionImplementationHandle{txReuseCou
nt=0, tx=KernelTransaction[0]} timeout.
ubuntu@ip-172-31-24-255:/var/log/neo4j$
```

Monitoring connections

- Can view the current connections to a Neo4j instance from cypher-shell
  - Call dbms.listConnections();

```
neo4j> call dbms.listConnections();
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| connectionId | connectTime                | connector | username    | userAgent                                                   | serverAddress         | clientAddress         |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+
| "bolt-97"    | "2019-02-19T20:35:30.513Z" | "bolt"    | "admin"     | "neo4j-javascript/1.7.2"                                    | "172.31.24.255:7687" | "216.45.71.93:57415" |
| "bolt-840"   | "2019-02-19T21:58:04.65Z"  | "bolt"    | "publisher" | "neo4j-java/1.7.2-7165cef0d4b602da30b65613977744ad661c2e67" | "127.0.0.1:7687"     | "127.0.0.1:40344"    |
| "bolt-779"   | "2019-02-19T21:52:05.978Z" | "bolt"    | "admin"     | "neo4j-java/dev"                                            | "127.0.0.1:7687"     | "127.0.0.1:40338"    |
| "bolt-839"   | "2019-02-19T21:58:04.358Z" | "bolt"    | "publisher" | "neo4j-java/1.7.2-7165cef0d4b602da30b65613977744ad661c2e67" | "127.0.0.1:7687"     | "127.0.0.1:40340"    |
+--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------+

4 rows available after 1 ms, consumed after another 0 ms
neo4j>
```

- Terminate the connection
  - dbms.killConnection()

```
neo4j> CALL dbms.killConnections(['bolt-1362','bolt-1363']);
+---------------------------------------------------+
| connectionId | username    | message            |
+---------------------------------------------------+
| "bolt-1362"  | "publisher" | "Connection found" |
| "bolt-1363"  | "publisher" | "Connection found" |
+---------------------------------------------------+

2 rows available after 14 ms, consumed after another 0 ms
neo4j>
```

Logging HTTP requests

- You can set this property in neo4j.conf to log HTTP requests:
  - vim /var/lib/neo4j/conf/neo4j.conf
    - dbms.logs.http.enabled=true
  - neo4j restart
- View the records in the http.log file
  - tail -f -n 10 /logs/http.log

```
debug.log  http.log  query.log  security.log
ubuntu@ip-172-31-24-255:/var/log/neo4j$ cat http.log
2019-02-20 14:41:07.614+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:07.609+0000]  216.45.71.93 -  [Wed Feb 20 14:41:07 UTC 2019] "/?null" 303 -1 "" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, l
ike Gecko) Chrome/72.0.3626.109 Safari/537.36" 40
2019-02-20 14:41:07.697+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:07.697+0000]  216.45.71.93 -  [Wed Feb 20 14:41:07 UTC 2019] "/browser/?null" 200 2895 "" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 10
2019-02-20 14:41:08.773+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:08.773+0000]  216.45.71.93 -  [Wed Feb 20 14:41:08 UTC 2019] "/browser/main.chunkhash.bundle.js?null" 200 1650952 "http://ec2-54-158-36-157.compute-1.amazona
ws.com:7474/browser/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 946
2019-02-20 14:41:11.042+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:11.042+0000]  216.45.71.93 -  [Wed Feb 20 14:41:11 UTC 2019] "/browser/vendors~main.chunkhash.bundle.js?null" 200 3301516 "http://ec2-54-158-36-157.compute-1
.amazonaws.com:7474/browser/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 3217
2019-02-20 14:41:12.503+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:12.502+0000]  216.45.71.93 -  [Wed Feb 20 14:41:12 UTC 2019] "/?null" 200 232 "http://ec2-54-158-36-157.compute-1.amazonaws.com:7474/browser/" "Mozilla/5.0 (
Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 57
2019-02-20 14:41:14.020+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:14.020+0000]  216.45.71.93 -  [Wed Feb 20 14:41:14 UTC 2019] "/browser/assets/fonts/OpenSans-Semibold.ttf?null" 200 221328 "http://ec2-54-158-36-157.compute-
1.amazonaws.com:7474/browser/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 3
2019-02-20 14:41:14.024+0000 INFO [REQUEST] [AsyncLog @ 2019-02-20 14:41:14.023+0000]  216.45.71.93 -  [Wed Feb 20 14:41:14 UTC 2019] "/browser/assets/fonts/OpenSans-Light.ttf?null" 200 222412 "http://ec2-54-158-36-157.compute-1.a
mazonaws.com:7474/browser/" "Mozilla/5.0 (Macintosh; Intel Mac OS X 10_14_3) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/72.0.3626.109 Safari/537.36" 2
ubuntu@ip-172-31-24-255:/var/log/neo4j$
```

Monitoring memory usage

- Default configurations in neo4j.conf are useful for small database
- In production, make sure that settings for the JVM are the best ones

Initial memory settings for database

- Obtain recommendation for settings related to memory:
  - neo4j-admin memrec
- Provides recommended memory settings

```
ubuntu@ip-172-31-24-255:/var/log/neo4j$ /usr/bin/neo4j-admin memrec --database=movie3.db
# Memory settings recommendation from neo4j-admin memrec:
#
# Assuming the system is dedicated to running Neo4j and has 7700m of memory,
# we recommend a heap size of around 3500m, and a page cache of around 1800m,
# and that about 2400m is left for the operating system, and the native memory
# needed by Lucene and Netty.
#
# Tip: If the indexing storage use is high, e.g. there are many indexes or most
# data indexed, then it might advantageous to leave more memory for the
# operating system.
#
# Tip: The more concurrent transactions your workload has and the more updates
# they do, the more heap memory you will need. However, don't allocate more
# than 31g of heap, since this will disable pointer compression, also known as
# "compressed oops", in the JVM and make less effective use of the heap.
#
# Tip: Setting the initial and the max heap size to the same value means the
# JVM will never need to change the heap size. Changing the heap size otherwise
# involves a full GC, which is desirable to avoid.
#
# Based on the above, the following memory settings are recommended:
dbms.memory.heap.initial_size=3500m
dbms.memory.heap.max_size=3500m
dbms.memory.pagecache.size=1800m
#
# The numbers below have been derived based on your current data volume in database and ind
# They can be used as an input into more detailed memory analysis.
# Lucene indexes: 0.0
# Data volume and native indexes: 54800k
```

Recommendation for log files

- Each type of log file should
  - Have its maximum size defined and
  - # of log files to keep
- # Number of HTTP logs to keep.
- dbms.logs.http.rotation.keep_number=5
- # Size of each HTTP log that is kept. (k,m,g)
- dbms.logs.http.rotation.size=20m
- # Number of query logs to keep.
- dbms.logs.query.rotation.keep_number=5
- # Size of each query log that is kept.
- dbms.logs.query.rotation.size=20m

Collecting metrics

- Automatically collects metrics
- Can disable them by setting
  - metrics.enabled=false
- Metrics are collected in CSV format
  - cd /var/lib/neo4j/metrics
  - ls -al