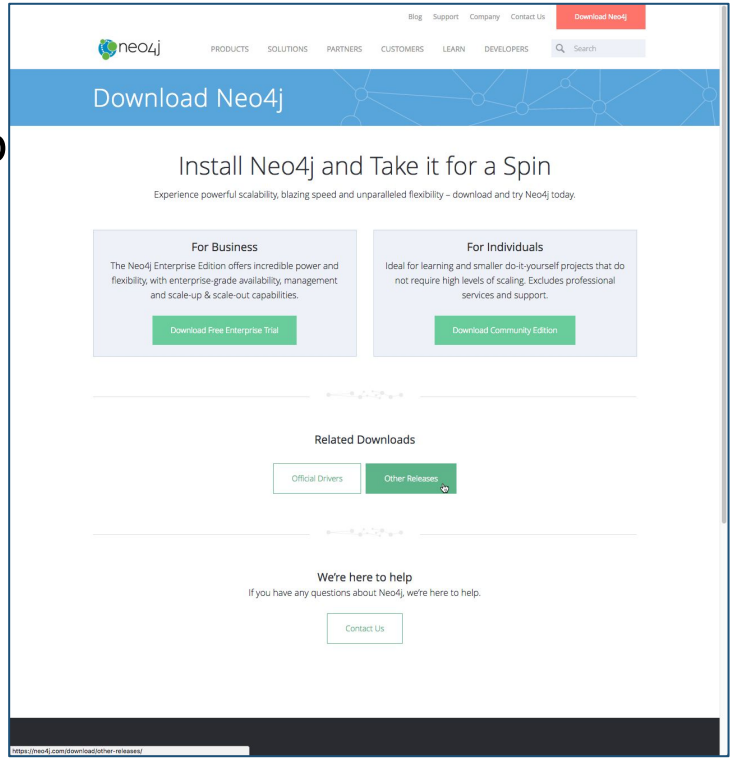
An operational intro to Neo4j

What are we going to do?

- Download and install Neo4j
- Overview of what Neo4j consists of
- Launch Neo4j and import a demo dataset
- Intro to Neo4j's command line tools

Obtain Neo4j Enterprise Edition

- Get Neo4j:
  - Copy Neo4j from your USB stick onto your machine, or
  - download it from http://neo4j.com/download
- Unpack Neo4j
  - Double-click, or
  - `tar -xf neo4j-enterprise-3.1.4-unix.tar.gz`



Obtaining Neo4j

There are several other ways to obtain Neo4j:

- Debian package
- Docker image
- Desktop Installer (Community only - not for production)

bin

This is where command line tools live.

```
bin/
├── cypher-shell
├── neo4j
└── neo4j-admin
```

Updating the default password

We need to update the default password on first use. Run the following command to do this:

```
./bin/neo4j-admin set-initial-password gctraining
```

bin/neo4j

Script to:

- start
- stop
- restart
- view the status of

the database.

bin/neo4j

Run the following command from your Neo4j home directory:

```
./bin/neo4j start
```

```
Starting Neo4j.
Started neo4j (pid 3844). By default, it is available at http://localhost:7474/
There may be a short delay until the server is ready.
See logs/neo4j.log for current status.
```

Watch the startup process

Have a look at the log file to see what's going on while the database is starting up.

```
tail -f logs/neo4j.log
```

Check if the database is up and running

We can also call the status or console commands to view the status of the database:

```
./bin/neo4j console
Neo4j is already running (pid 86209).
```

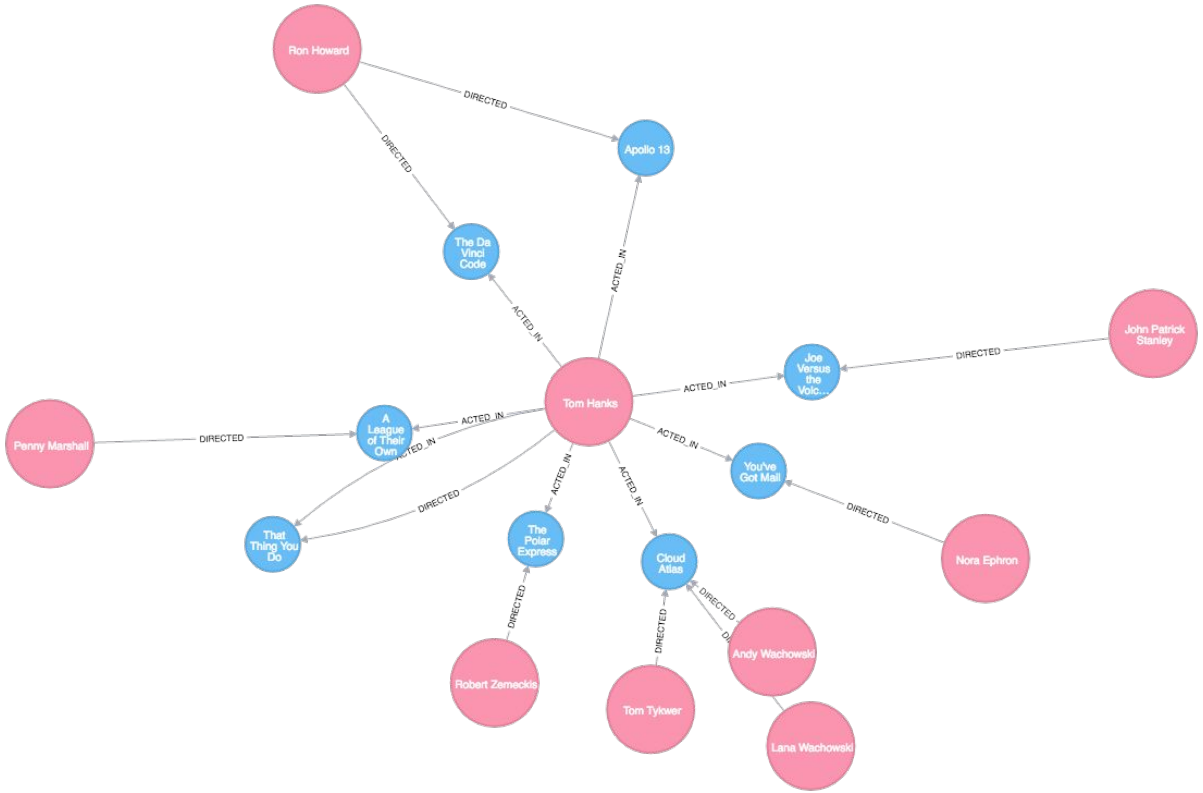
```
./bin/neo4j status
Neo4j is already running (pid 86209).
```

:play movies

Navigate to http://localhost:7474/ and type the following command into the query bar:

```
:play movies
```

:play movies


Ron Howard
DIRECTED
Apollo 13
DIRECTED
The Da Vinci Code
ACTED_IN
ACTED_IN
John Patrick Stanley
DIRECTED
Joe Versus the Volc...
ACTED_IN
Tom Hanks
Penny Marshall
DIRECTED
A League of Their Own
ACTED_IN
ACTED_IN
ACTED_IN
You've Got Mail
DIRECTED
Nora Ephron
DIRECTED
That Thing You Do
ACTED_IN
The Polar Express
ACTED_IN
Cloud Atlas
DIRECTED
DIRECTED
Robert Zemeckis
DIRECTED
Tom Tykwer
Andy Wachowski
Lana Wachowski

Neo4j Browser 101

- Run queries with CMD (CTRL) + Enter
- Insert new-line with SHIFT + Enter
- Expand the query bar with ESC
- CMD (CTRL) + Up Arrow to scroll through past queries
- Use `:clear` to clear past results
- Double-click a node and see what happens!
- Change node colors
- Change which node property is displayed

:play sysinfo

Type the following command into the query bar to get an overview of the state of the server:

```
:play sysinfo
```

:play sysinfo

:play sysinfo

Store Sizes	
Array Store	8.00 KiB
Logical Log	2.68 KiB
Node Store	8.00 KiB
Property Store	7.97 KiB
Relationship Store	0 B
String Store Size	8.00 KiB
Total Store Size	133.33 KiB

ID Allocation	
Node ID	1
Property ID	1
Relationship ID	0
Relationship Type ID	0

Page Cache			
Faults	33	Flushes	0
Evictions	0	Eviction Exceptions	0
File Mappings	44	File Unmappings	28
Bytes Read	213263	Bytes Written	0

Transactions	
Last Tx Id	20
Current	0
Peak	3
Opened	184
Committed	155

High Availability

Not enabled.

Cluster

No cluster.

Note: some metrics only available in Neo4j Enterprise.

AUTO-REFRESH OFF

bin/cypher-shell

A command line shell where you can execute Cypher against an instance of Neo4j. Interacts with the database using the Bolt Protocol.

Replaces `neo4j-shell`.

bin/cypher-shell

Run the following command from your Neo4j home directory:

```
./bin/cypher-shell -u neo4j -p <password>
```

```
Connected to Neo4j at bolt://localhost:7687 as user neo4j.

Type :help for a list of available commands.

neo4j>
```

Trying out bin/cypher-shell

Type the following command into `cypher-shell`:

```
CALL db.labels();
```

Viewing all the procedures

You can see a list of all the available procedures by running the following procedure:

```
CALL dbms.procedures();
```

Try out some of the other procedures to get the hang of what they do.

Deploying procedures

We can also deploy our own custom procedures into the `plugins` directory.

The Neo4j community are maintaining a library of useful procedures in the apoc project:

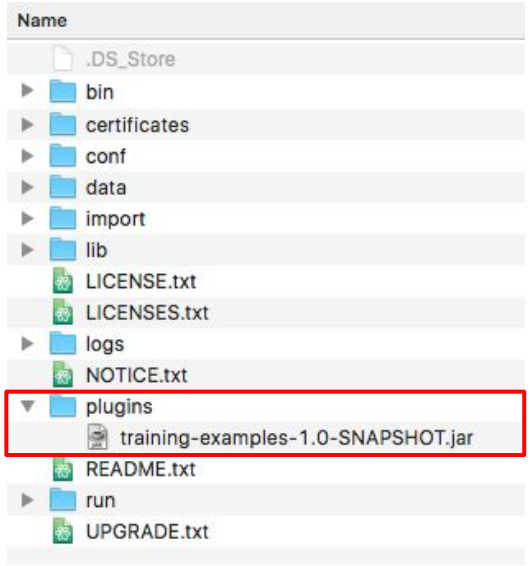
https://github.com/neo4j-contrib/neo4j-apoc-procedures/releases

Deploying procedures

Copy the `plugins` folders from **USB Stick** to the **`default.graphdb`** folder (or **`$NEO4J_HOME`**)

Then restart Neo4j:

```
./bin/neo4j restart
```



Viewing all the procedures

Now run the procedure again:

```
CALL dbms.procedures();
```

```
$ call dbms.procedures()
```

dbms.security.changePassword	dbms.security.changePassword(password :: STRING?, requirePasswordChange = false :: BOOLEAN?) :: VOID	Change the current user's password.
dbms.security.changeUserPassword	dbms.security.changeUserPassword(username :: STRING?, newPassword :: STRING?, requirePasswordChange = true :: BOOLEAN?) :: VOID	Change the given user's password.
dbms.security.clearAuthCache	dbms.security.clearAuthCache() :: VOID	Clears authentication and authorization cache.
dbms.security.createRole	dbms.security.createRole(roleName :: STRING?) :: VOID	Create a new role.
dbms.security.createUser	dbms.security.createUser(username :: STRING?, password :: STRING?, requirePasswordChange = true :: BOOLEAN?) :: VOID	Create a new user.
dbms.security.deleteRole	dbms.security.deleteRole(roleName :: STRING?) :: VOID	Delete the specified role. Any role assignments will be removed.
dbms.security.deleteUser	dbms.security.deleteUser(username :: STRING?) :: VOID	Delete the specified user.
dbms.security.listRoles	dbms.security.listRoles() :: (role :: STRING?, users :: LIST? OF STRING?)	List all available roles.
dbms.security.listRolesForUser	dbms.security.listRolesForUser(username :: STRING?) :: (value :: STRING?)	List all roles assigned to the specified user.
dbms.security.listUsers	dbms.security.listUsers() :: (username :: STRING?, roles :: LIST? OF STRING?, flags :: LIST? OF STRING?)	List all local users.
dbms.security.listUsersForRole	dbms.security.listUsersForRole(roleName :: STRING?) :: (value :: STRING?)	List all users currently assigned the specified role.
dbms.security.removeRoleFromUser	dbms.security.removeRoleFromUser(roleName :: STRING?, username :: STRING?) :: VOID	Unassign a role from the user.
dbms.security.showCurrentUser	dbms.security.showCurrentUser() :: (username :: STRING?, roles :: LIST? OF STRING?, flags :: LIST? OF STRING?)	Show the current user.
dbms.security.suspendUser	dbms.security.suspendUser(username :: STRING?) :: VOID	Suspend the specified user.
dbms.setTXMetaData	dbms.setTXMetaData(data :: MAP?) :: VOID	*(empty)*
training.moviesOnly	training.moviesOnly(name :: STRING?) :: (title :: STRING?)	Find movies by an actor

Returned 35 records in 92 ms.

conf

This is where configuration files live.

conf

This is where configuration files live.

```
conf/
├── jmx.access
├── jmx.password
└── neo4j.conf
```

conf/neo4j.conf

```
# The name of the database to mount
#dbms.active_database=graph.db

# Whether requests to Neo4j are authenticated.
# To disable authentication, uncomment this line
#dbms.security.auth_enabled=false

#*****************************************************************
# Core/Edge Configuration
#*****************************************************************
```

Change HTTP port

```
# HTTP Connector
dbms.connector.http.type=HTTP
dbms.connector.http.enabled=true
#dbms.connector.http.listen_address=:7474
```

Change HTTP port

```
# HTTP Connector
dbms.connector.http.type=HTTP
dbms.connector.http.enabled=true
dbms.connector.http.listen_address=:8484
```

Restart Neo4j

Run the following command:

```
./bin/neo4j restart
```

Navigate to http://localhost:8484 in your browser

data

This is where the data lives.

data

This is where the data lives.

```
data/
├── databases
│   └── graph.db
│       ├── schema
│       │   └── label
│       │       └── lucene
│       │           └── labelStore
│       └── store_lock
└── dbms
    ├── auth
    └── roles
```

data

This is where the data lives.

```
data/
├── databases
│   └── graph.db
│       ├── schema
│       │   └── label
│       │       └── lucene
│       │           └── labelStore
│       └── store_lock
└── dbms
    ├── auth
    └── roles
```

```
dbms.directories.data=data
```

data

This is where the data lives.

```
data/
├── databases
│   └── graph.db
│       ├── schema
│       │   └── label
│       │       └── lucene
│       │           └── labelStore
│       └── store_lock
└── dbms
    ├── auth
    └── roles
```

```
dbms.active_database=graph.db
```

data/databases/graph.db

```
neostore
neostore.counts.db.a
neostore.counts.db.b
neostore.id
neostore.labeltokenstore.db
neostore.labeltokenstore.db.id
neostore.labeltokenstore.db.names
neostore.labeltokenstore.db.names.id
neostore.nodestore.db
neostore.nodestore.db.id
neostore.nodestore.db.labels
neostore.nodestore.db.labels.id
neostore.propertystore.db
neostore.propertystore.db.arrays
neostore.propertystore.db.arrays.id
neostore.propertystore.db.id
neostore.propertystore.db.index
neostore.propertystore.db.index.id
neostore.propertystore.db.index.keys
neostore.propertystore.db.index.keys.id
neostore.propertystore.db.strings
neostore.propertystore.db.strings.id
neostore.relationshipgroupstore.db
neostore.relationshipgroupstore.db.id
neostore.relationshipstore.db
neostore.relationshipstore.db.id
neostore.relationshiptypestore.db
neostore.relationshiptypestore.db.id
neostore.relationshiptypestore.db.names
neostore.relationshiptypestore.db.names.id
neostore.schemastore.db
neostore.schemastore.db.id
neostore.transaction.db.0
schema
store_lock
```

Transaction log

```
neostore
neostore.counts.db.a
neostore.counts.db.b
neostore.id
neostore.labeltokenstore.db
neostore.labeltokenstore.db.id
neostore.labeltokenstore.db.names
neostore.labeltokenstore.db.names.id
neostore.nodestore.db
neostore.nodestore.db.id
neostore.nodestore.db.labels
neostore.nodestore.db.labels.id
neostore.propertystore.db
neostore.propertystore.db.arrays
neostore.propertystore.db.arrays.id
neostore.propertystore.db.id
neostore.propertystore.db.index
neostore.propertystore.db.index.id
neostore.propertystore.db.index.keys
neostore.propertystore.db.index.keys.id
neostore.propertystore.db.strings
neostore.propertystore.db.strings.id
neostore.relationshipgroupstore.db
neostore.relationshipgroupstore.db.id
neostore.relationshipstore.db
neostore.relationshipstore.db.id
neostore.relationshiptypestore.db
neostore.relationshiptypestore.db.id
neostore.relationshiptypestore.db.names
neostore.relationshiptypestore.db.names.id
neostore.schemastore.db
neostore.schemastore.db.id
neostore.transaction.db.0
schema
store_lock
```

Store files

neostore
neostore.counts.db.a
neostore.counts.db.b
neostore.id
neostore.labeltokenstore.db
neostore.labeltokenstore.db.id
neostore.labeltokenstore.db.names
neostore.labeltokenstore.db.names.id
neostore.nodestore.db
neostore.nodestore.db.id
neostore.nodestore.db.labels
neostore.nodestore.db.labels.id
neostore.propertystore.db
neostore.propertystore.db.arrays
neostore.propertystore.db.arrays.id
neostore.propertystore.db.id
neostore.propertystore.db.index
neostore.propertystore.db.index.id
neostore.propertystore.db.index.keys
neostore.propertystore.db.index.keys.id
neostore.propertystore.db.strings
neostore.propertystore.db.strings.id
neostore.relationshipgroupstore.db
neostore.relationshipgroupstore.db.id
neostore.relationshipstore.db
neostore.relationshipstore.db.id
neostore.relationshiptypestore.db
neostore.relationshiptypestore.db.id
neostore.relationshiptypestore.db.names
neostore.relationshiptypestore.db.names.id
neostore.schemastore.db
neostore.schemastore.db.id
neostore.transaction.db.0
schema
store_lock

Metadata file

```
neostore
neostore.counts.db.a
neostore.counts.db.b
neostore.id
neostore.labeltokenstore.db
neostore.labeltokenstore.db.id
neostore.labeltokenstore.db.names
neostore.labeltokenstore.db.names.id
neostore.nodestore.db
neostore.nodestore.db.id
neostore.nodestore.db.labels
neostore.nodestore.db.labels.id
neostore.propertystore.db
neostore.propertystore.db.arrays
neostore.propertystore.db.arrays.id
neostore.propertystore.db.id
neostore.propertystore.db.index
neostore.propertystore.db.index.id
neostore.propertystore.db.index.keys
neostore.propertystore.db.index.keys.id
neostore.propertystore.db.strings
neostore.propertystore.db.strings.id
neostore.relationshipgroupstore.db
neostore.relationshipgroupstore.db.id
neostore.relationshipstore.db
neostore.relationshipstore.db.id
neostore.relationshiptypestore.db
neostore.relationshiptypestore.db.id
neostore.relationshiptypestore.db.names
neostore.relationshiptypestore.db.names.id
neostore.schemastore.db
neostore.schemastore.db.id
neostore.transaction.db.0
schema
store_lock
```

Schema

```
neostore
neostore.counts.db.a
neostore.counts.db.b
neostore.id
neostore.labeltokenstore.db
neostore.labeltokenstore.db.id
neostore.labeltokenstore.db.names
neostore.labeltokenstore.db.names.id
neostore.nodestore.db
neostore.nodestore.db.id
neostore.nodestore.db.labels
neostore.nodestore.db.labels.id
neostore.propertystore.db
neostore.propertystore.db.arrays
neostore.propertystore.db.arrays.id
neostore.propertystore.db.id
neostore.propertystore.db.index
neostore.propertystore.db.index.id
neostore.propertystore.db.index.keys
neostore.propertystore.db.index.keys.id
neostore.propertystore.db.strings
neostore.propertystore.db.strings.id
neostore.relationshipgroupstore.db
neostore.relationshipgroupstore.db.id
neostore.relationshipstore.db
neostore.relationshipstore.db.id
neostore.relationshiptypestore.db
neostore.relationshiptypestore.db.id
neostore.relationshiptypestore.db.names
neostore.relationshiptypestore.db.names.id
neostore.schemastore.db
neostore.schemastore.db.id
neostore.transaction.db.0
schema
store_lock
```

Schema

```
data/databases/graph.db/schema/
├── index
│   └── lucene
│       └── 1
│           ├── 1
│           └── failure-message
└── label
    └── lucene
        └── labelStore
            ├── 1
            └── failure-message
```

Indexes and Unique Constraints

```
data/databases/graph.db/schema/
├── index
│   └── lucene
│       └── 1
│           ├── 1
│           └── failure-message
└── label
    └── lucene
        └── labelStore
            ├── 1
            └── failure-message
```

Label scan store

```
data/databases/graph.db/schema/
├── index
│   └── lucene
│       └── 1
│           ├── 1
│           └── failure-message
└── label
    └── lucene
        └── labelStore
            ├── 1
            └── failure-message
```

logs

This is where log files live.

logs

This is where log files live.

`neo4j.log` The standard log, where general information about Neo4j is written.

`debug.log` Information useful when debugging problems with Neo4j.

logs/neo4j.log

```
2016-09-14 09:04:14.185+0000 INFO  Starting…
2016-09-14 09:04:14.793+0000 INFO  Bolt enabled on localhost:7687.
2016-09-14 09:04:14.802+0000 INFO  Initiating metrics…
2016-09-14 09:04:16.386+0000 INFO  Started.
2016-09-14 09:04:16.508+0000 INFO  Mounted REST API at: /db/manage
2016-09-14 09:04:17.013+0000 INFO  Remote interface available at
http://localhost:7474/
```

logs/debug.log

```
2016-09-14 10:15:16.155+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Starting check
pointing...

2016-09-14 10:15:16.157+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Starting store
flush...

2016-09-14 10:15:16.158+0000 INFO  [o.n.k.i.s.c.CountsTracker] About to rotate counts store at transaction 19 to
[/tmp/neo4j-enterprise-3.1.0/data/databases/graph.db/neostore.counts.db.b], from
[/tmp/neo4j-enterprise-3.1.0/data/databases/graph.db/neostore.counts.db.a].

2016-09-14 10:15:16.159+0000 INFO  [o.n.k.i.s.c.CountsTracker] Successfully rotated counts store at transaction 19 to
[/tmp/neo4j-enterprise-3.1.0/data/databases/graph.db/neostore.counts.db.b], from
[/tmp/neo4j-enterprise-3.1.0/data/databases/graph.db/neostore.counts.db.a].

2016-09-14 10:15:16.161+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Store flush
completed

2016-09-14 10:15:16.161+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Starting
appending check point entry into the tx log...

2016-09-14 10:15:16.162+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Appending check
point entry into the tx log completed

2016-09-14 10:15:16.162+0000 INFO  [o.n.k.i.t.l.c.CheckPointerImpl] Check Pointing triggered by scheduler for time threshold [19]:  Check pointing
completed
```

End of Module
An operational intro to Neo4j

Questions ?

Backup/Restore

What are we going to do?

- Backup a database
- Mess up the data in the database
- Restore from the backup
- Tamper with the store files
- Check consistency of the store files

bin/neo4j-admin

A collection of tools for executing administrative tasks against the database.

bin/neo4j-admin

Run the following command from your Neo4j home directory:

```
./bin/neo4j-admin
```

```
neo4j-admin restore --from=<backup-directory> --database=<database-name> [--force]
    Restore a backed up database.

neo4j-admin users <subcommand> [<username>] [<password>] [--requires-password-change]
    Runs several possible sub-commands for managing the native users repository

neo4j-admin roles <subcommand> [<roleName>] [<username>]
    Runs several possible sub-commands for managing the native roles repository

neo4j-admin backup [--from=<address>] --to=<backup-path> [--check-consistency] [--additional-config=<config-file-path>] [--timeout=<timeout>]
    Perform a backup, over the network, from a running Neo4j server into a local copy of the database store (the backup).

neo4j-admin check-consistency --database=<database> [--additional-config=<file>] [--verbose]
    Check the consistency of a database.

neo4j-admin import --mode={database|csv} --database=<database-name>
    Import a collection of CSV files with --mode=csv, or a database from a pre-3.0 installation with --mode=database.
```

bin/neo4j-admin backup

Perform an online backup of the database.

bin/neo4j-admin backup

Run the following command from your Neo4j home directory:

```
./bin/neo4j-admin backup \
--from localhost:6362 \
--to /tmp/backup-graph-connect
```

Mess up the database

Let’s mess up some of the nodes in the database.

Execute the following query:

```
MATCH (p:Person)
SET p.name = "Emil"
```

Restoring a backup

We'd better restore the database. First we'll need to stop the database:

```
./bin/neo4j stop
```

bin/neo4j-admin restore

Restore a backed up database.

bin/neo4j-admin restore

Run the following command from your Neo4j home directory:

```
./bin/neo4j-admin restore \
--from /tmp/backup-graph-connect \
--database graph.db \
--force
```

Restoring a backup

Now start the database up...

```
./bin/neo4j start
```

...and run the following query to check the data has been restored:

```
MATCH (p:Person)
RETURN p
```

Tamper with the store files

Restore the backup to another database name:

```
./bin/neo4j-admin restore \
--from /tmp/backup-graph-connect \
--database tamper.db \
--force
```

Tamper with one of the files in `data/databases/tamper.db`

e.g. type some random characters into `neostore.nodestore.db`

Check consistency of the database

```
./bin/neo4j-admin check-consistency
```

```
neo4j-admin check-consistency --database=<database>
[--additional-config=<file>] [--verbose]
```

Check consistency of the database

```
./bin/neo4j-admin check-consistency --database tamper.db
```

Check consistency of the database

- Automatically runs as part of the backup process
- Consistency check is an expensive process that will compete for resources with the database; therefore
  - You can disable consistency check in the backup process
  - You can move the backup to a different server and run consistency check there

End of Module Backup/Restore

Questions ?

Data Import

What are we going to do?

- An intro to the `neo4j-import` tool
- Create a new database from CSV files

Planning for data import

Data import is special for graph databases because it needs to pre-compute joins (**relationships**) between records (**nodes**).

If you have a lot of data it’s most efficient to do this offline, outside your normal application architecture.

bin/neo4j-admin import

Creates a new database from a collection of CSV files. Would typically be used for initial data import.

Import CSV files

Copy the `gcsf2016-import.zip` file from the USB stick onto your machine and unpack it. If you don't have a USB stick you can download it here:

https://s3.amazonaws.com/example-data.neo4j.org/files/production-import.zip

Import CSV files

```
$ unzip production-import.zip

Archive:  production-import.zip
extracting: crimes_header.csv
inflating: crimes.csv
inflating: beats.csv
inflating: crimesBeats.csv
inflating: crimesPrimaryTypes.csv
inflating: primaryTypes.csv
```

Import CSV files

crimes.csv

:ID(Crime)	id	:LABEL	date	description
4730813	4730813	Crime	05/09/2006 08:20:00 AM	POCKET-PICKING
7150780	7150780	Crime	09/28/2009 01:00:00 AM	CHILD ABANDONMENT
4556970	4556970	Crime	12/16/2005 08:39:24 PM	POSS: CANNABIS 30GMS OR LESS
9442492	9442492	Crime	12/28/2013 12:15:00 PM	OVER $500
7524560	7524560	Crime	05/25/2010 12:00:00 PM	AGGRAVATED: HANDGUN

beats.csv

:ID(Beat)	id	:LABEL
1132	1132	Beat
0813	0813	Beat
0513	0513	Beat
0532	0532	Beat
1311	1311	Beat

crimesBeats.csv

:START_ID(Crime)	:END_ID(Beat)	:TYPE
6978096	0911	ON_BEAT
3170923	2511	ON_BEAT
3073515	1012	ON_BEAT
8157905	0113	ON_BEAT
6915474	0912	ON_BEAT

bin/neo4j-admin import

Run the following command from your Neo4j home directory:

```
./bin/neo4j-admin import
```

```
neo4j-admin import --mode={database|csv} --database=<database-name> [--additional-config=<config-file-path>]
[--from=<source-directory>] [--nodes[:Label1:Label2]="<file1>,<file2>,..."]
[--relationships[:RELATIONSHIP_TYPE]="<file1>,<file2>,..."] [--input-encoding=<character-set>] [--id-type=<id-type>]

    Import a collection of CSV files with --mode=csv, or a database from a pre-3.0 installation with --mode=database.
    --mode=database        Import a database from a pre-3.0 Neo4j installation.
    --mode=csv Import a database from a collection of CSV files.
    --nodes[:Label1:Label2]="<file1>,<file2>,..."
    --relationships[:RELATIONSHIP_TYPE] "<file1>,<file2>,..."
    --id-type <id-type>
    --input-encoding <character-set>
    --page-size <page-size>
```

bin/neo4j-admin import

```
./bin/neo4j-admin import \
--mode csv \
--database imported.db \
--nodes crimes_header.csv,crimes.csv \
--nodes beats.csv  \
--nodes primaryTypes.csv \
--relationships crimesBeats.csv  \
--relationships:PRIMARY_TYPE crimesPrimaryTypes.csv
```

Active database

conf/neo4j.conf

```
# The name of the database to mount
#dbms.active_database=graph.db
```

Active database

conf/neo4j.conf

```
# The name of the database to mount
dbms.active_database=imported.db
```

Restart the database

Restart the database:

```
./bin/neo4j restart
```

And run the following command in the browser to see the new data:

```
:sysinfo
```

Import summary

Execute the following query to get a summary of what we've imported into the database:

```
MATCH ()-[]->()
WITH COUNT(*) AS count
RETURN "relationships" AS type, count
UNION
MATCH ()
WITH COUNT(*) AS count
RETURN "nodes" AS type, count
```

End of Module
Data Import

Questions ?

Monitoring

What are we going to do?

- View currently running queries
- Kill running queries
- Setup metrics

Killing long running queries

Currently running queries

We can run the following procedure to view the currently running queries:

```
CALL dbms.listQueries();
```

Currently running queries

We can run the following procedure to view the currently running queries:

```
CALL dbms.listQueries();
```

```
$ CALL dbms.listQueries;
```

queryId	username	query	parameters	startTime	elapsedTime	connectionDetails	metaData
query-382	neo4j	CALL dbms.listQueries;	*(empty)*	2016-09-28T11:29:01.338+01:00	00:00:00.001	server-session http 127.0.0.1 /db/data/transaction/167/commit neo4j	*(empty)*

Returned 1 row in 10 ms.

Start a long running query

Let’s create a long running query and then kill it. Execute the following query:

```
MATCH (m), (n), (o), (p), (q)
RETURN COUNT(*) AS count
```

Currently running queries

Now run the list queries procedure again:

```
CALL dbms.listQueries()
```

Currently running queries

Now run the list queries procedure again:

```
CALL dbms.listQueries()
```

```
$ CALL dbms.listQueries()
```

Rows | Text | Code

queryId	username	query	parameters	startTime	elapsedTime	connectionDetails	metaData
query-88	neo4j	MATCH (m), (n), (o), (p), (q) RETURN COUNT(*) AS count	*(empty)*	2016-10-05T16:10:31.773+01:00	00:00:11.170	server-session http 127.0.0.1 /db/data/transaction/67/commit neo4j	*(empty)*
query-90	neo4j	CALL dbms.listQueries()	*(empty)*	2016-10-05T16:10:42.943+01:00	00:00:00.001	server-session http 127.0.0.1 /db/data/transaction/69/commit neo4j	*(empty)*

Returned 2 records in 10 ms.

Currently running queries

We have a built in tool in the browser. Type `:qs` into the query pane:

:qs

Host	User	Query	Params	Meta	Elapsed time	Kill
http://localhost:7474	neo4j	CALL dbms.listQueries()	{}	{}	00:00:00.001	Confirm Kill query
http://localhost:7474	neo4j	MATCH (m), (n), (o), (p), (q) RETURN COUNT(*) A...	{}	{}	00:01:17.251	

Found 2 queries running on one server

AUTO-REFRESH OFF

Killing a query

We have a built in tool in the browser. Type `:queries` into the query

:qs

Host	User	Query	Params	Meta	Elapsed time	Kill
http://localhost:7474	neo4j	CALL dbms.listQueries()	{}	{}	00:00:00.001	Confirm Kill query
http://localhost:7474	neo4j	MATCH (m), (n), (o), (p), (q) RETURN COUNT(*) A...	{}	{}	00:01:17.251	

Found 2 queries running on one server

$ MATCH (m), (n), (o), (p), (q) RETURN COUNT(*) AS count

Errors

Code

The transaction has been terminated. Retry your operation in a new transaction, and you should see a successful resul
t. Explicitly terminated by the user.

Neo.DatabaseError.Statement.ExecutionFailed

Kill running queries

We can also kill the query manually using the kill query procedure:

```
$ CALL dbms.listQueries()
```

queryId	username	query	parameters	startTime	elapsedTime	connectionDetails	metaData
query-88	neo4j	MATCH (m), (n), (o), (p), (q) RETURN COUNT(*) AS count	*(empty)*	2016-10-05T16:10:31.773+01:00	00:00:11.170	server-session http 127.0.0.1 /db/data/transaction/67/commit neo4j	*(empty)*
query-90	neo4j	CALL dbms.listQueries()	*(empty)*	2016-10-05T16:10:42.943+01:00	00:00:00.001	server-session http 127.0.0.1 /db/data/transaction/69/commit neo4j	*(empty)*

Returned 2 records in 10 ms.

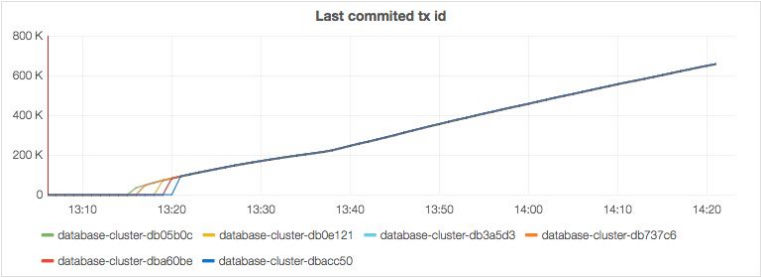
```
CALL dbms.killQuery("query-88")
```

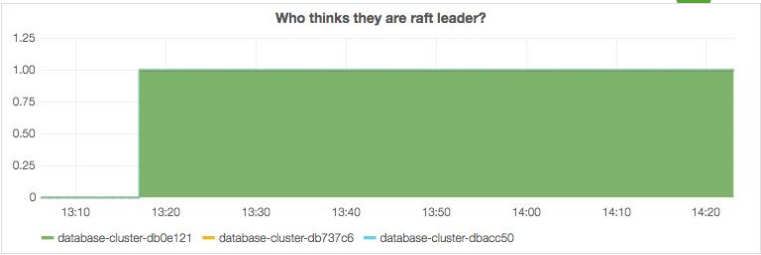
Metrics

Enabling metrics

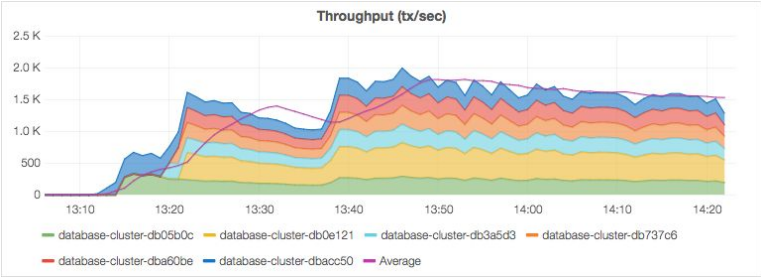
```
// default setting for enabling all supported metrics
metrics.enabled=true
metrics.graphite.enabled=true
metrics.graphite.server=<ip>:2003
metrics.graphite.interval=<how often to send data, defaults to 3s>
metrics.prefix=<Neo4j instance name, e.g. wwwneo1>
```

Grafana

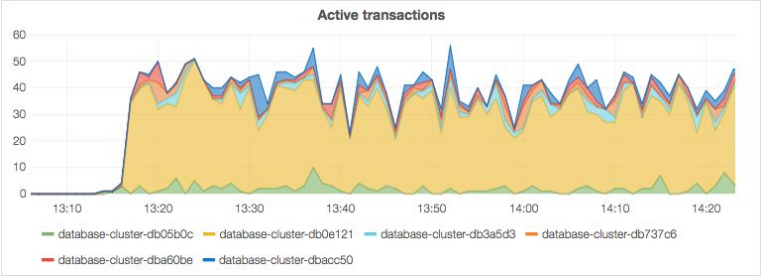

Last commited tx id
800 K
600 K
400 K
200 K
0
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
database-cluster-db05b0c
database-cluster-db0e121
database-cluster-db3a5d3
database-cluster-db737c6
database-cluster-dba60be
database-cluster-dbacc50


Who thinks they are raft leader?
1.25
1.00
0.75
0.50
0.25
0
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
database-cluster-db0e121
database-cluster-db737c6
database-cluster-dbacc50

Total Data Store (bytes)
9 GiB
7 GiB
6 GiB
4 GiB
2 GiB
0 B
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
db05b0c.all-data
db0e121.all-data
db3a5d3.all-data
db737c6.all-data
dba60be.all-data
dbacc50.all-data


Throughput (tx/sec)
2.5 K
2.0 K
1.5 K
1.0 K
500
0
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
database-cluster-db05b0c
database-cluster-db0e121
database-cluster-db3a5d3
database-cluster-db737c6
database-cluster-dba60be
database-cluster-dbacc50
Average

Available instances
7
6
5
4
3
2
1
0
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
database-cluster-db05b0c
database-cluster-db0e121
database-cluster-db3a5d3
database-cluster-db737c6
database-cluster-dba60be
database-cluster-dbacc50


Active transactions
60
50
40
30
20
10
0
13:10
13:20
13:30
13:40
13:50
14:00
14:10
14:20
database-cluster-db05b0c
database-cluster-db0e121
database-cluster-db3a5d3
database-cluster-db737c6
database-cluster-dba60be
database-cluster-dbacc50

End of Module Monitoring

Questions ?

Security

What are we going to do?

- Role based access control
- Procedure annotated access control

Role Based Access Control

Create a user for yourself

Type the following command in the query pane:

```
:server user list
```

:server user list

Username	Role(s)	Status	Password Change	Delete
neo4j	admin	Activated		

Add new user

Create a user for yourself

Now add yourself as a new user and assign yourself the admin role.

```
:server user list
```

Username	Role(s)	Status	Password Change	Delete
neo4j	admin	Activated		
graphconnect	admin	Activated		

Add new user

Security logging

- Enable query logging and then restart Neo4j `dbms.logs.query.enabled=true`
- Open two browser windows
  - Note: credentials are cached in the browser, so it's easiest to use Incognito windows so that they are independent from each other
- Log into different users in each window
- Run queries
- Inspect the security log for users and their respective queries

Security logging

```
logs/
├── debug.log
├── neo4j.log
├── query.log
└── security.log
```

Security logging

```
logs/
├── debug.log
├── neo4j.log
├── query.log
└── security.log
```

logs/security.log

```
2016-10-05 16:16:22.045+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.045+0000]  [neo4j]: logged in
2016-10-05 16:16:22.045+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.045+0000]  [neo4j]: logged in
2016-10-05 16:16:22.088+0000 INFO  [AsyncLog @ 2016-10-05 16:16:22.088+0000]  [neo4j]: logged in
2016-10-05 16:16:26.256+0000 INFO  [AsyncLog @ 2016-10-05 16:16:26.256+0000]  [neo4j]: logged in
2016-10-05 16:16:26.260+0000 INFO  [AsyncLog @ 2016-10-05 16:16:26.260+0000]  [neo4j]: logged in
2016-10-05 16:16:50.354+0000 INFO  [AsyncLog @ 2016-10-05 16:16:50.354+0000]  [neo4j]: logged in
2016-10-05 16:16:50.357+0000 INFO  [AsyncLog @ 2016-10-05 16:16:50.357+0000]  [neo4j]: logged in

2016-10-05 16:17:14.021+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.020+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.033+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.033+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.077+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.077+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.078+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.078+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.091+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.091+0000]  [graphconnect]: logged in
2016-10-05 16:17:14.091+0000 INFO  [AsyncLog @ 2016-10-05 16:17:14.091+0000]  [graphconnect]: logged in
```

logs/query.log

```
2016-10-05 16:30:19.378+0000 INFO  10 ms: server-session   http   127.0.0.1    /db/data/transaction/commit
graphconnect - EXPLAIN CREATE (:Event {name: "Graph Connect San Francisco 2016"}) - {} - {}

2016-10-05 16:30:19.792+0000 INFO  4 ms: server-session    http   127.0.0.1    /db/data/transaction/33/commit
graphconnect - CREATE (:Event {name: "Graph Connect San Francisco 2016"}) - {} - {}
```

User Roles

Predefined roles

Privileges	Reader	Publisher	Architect	Admin
Change own password	•	•	•	•
Read data	•	•	•	•
View own details	•	•	•	•
Terminate own query	•	•	•	•
Write/update/delete data		•	•	•
Manage index/constraints			•	•
View all users and roles				•
Create/delete users				•
Assign/delete user roles				•
Suspend/activate users				•
Terminate others' queries				•
Change others' passwords				•

User roles

```
CALL dbms.security.listRoles()
```

role	users	
reader	[]	Can read
architect	[]	Can read, write, and execute schema operations
admin	[graphconnect, neo4j]	Can do anything
publisher	[]	Can read and write

Create a new user with read only permission

Execute the following procedure calls to create a new user and assign them to the ‘reader’ role:

```
CALL dbms.security.createUser("restrictedUser", "neo4j", false)

CALL dbms.security.addRoleToUser("reader", "restrictedUser")
```

User roles

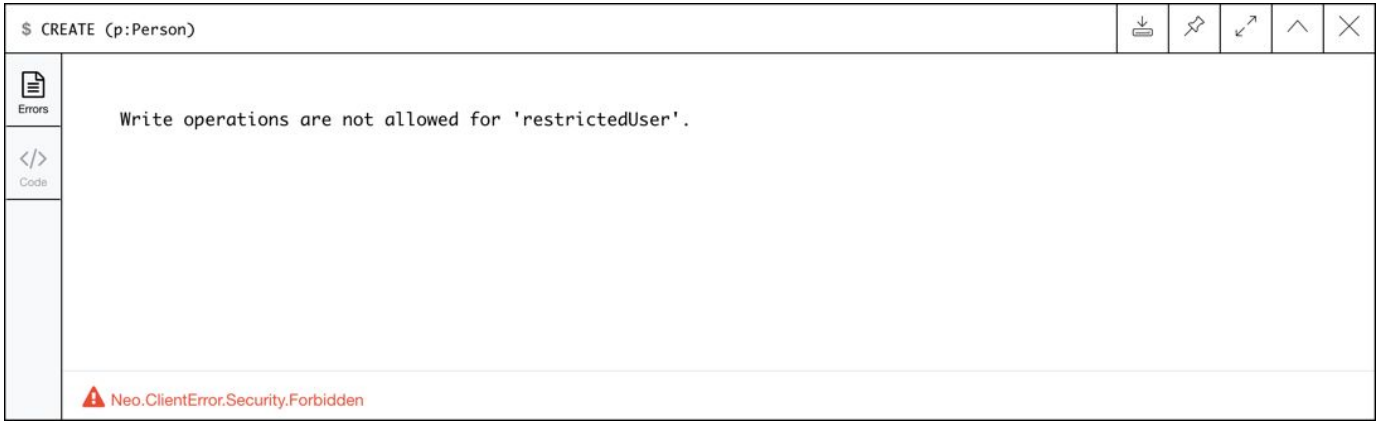
```
CALL dbms.security.listRoles()
```

role	users
reader	[restrictedUser]
architect	[]
admin	[graphconnect, neo4j]
publisher	[]

Create data: denied!

Now login as that user in another browser session and try and create some data:

```
CREATE (p:Person);
```



Procedure Annotated Access Control

Locked down roles

We can also create completely locked down roles which can only make calls to specific procedures.

e.g. imagine we want to create a role which can only find the movies that people have acted in but can’t do anything else!

Create locked down role

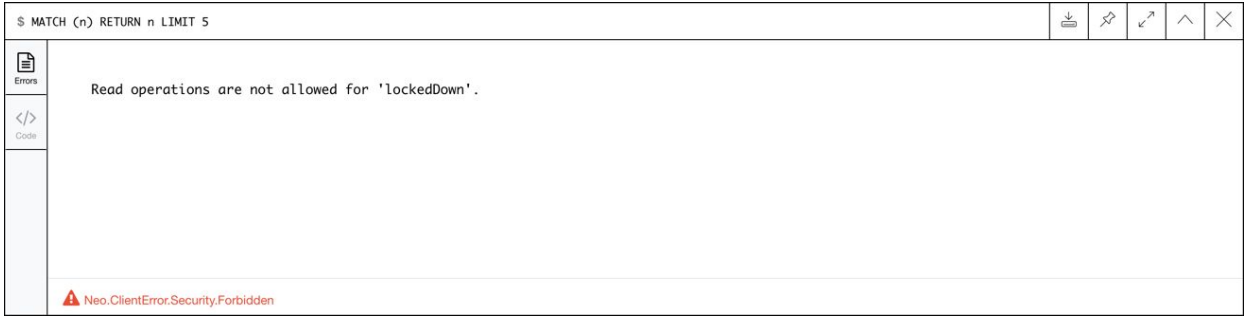
Let’s create a locked down role and a new user to assign to that role. Run the following commands:

```
CALL dbms.security.createRole("movies_only")
CALL dbms.security.createUser("lockedDown", "abc", false)
CALL dbms.security.addRoleToUser("movies_only", "lockedDown")
```

Create locked down role

Login as our new user and try to execute any Cypher query e.g.

```
MATCH (n)
RETURN n
LIMIT 5
```



Our user can’t do anything!

Locked down roles

```
@Context
public GraphDatabaseService db;

@Description( "Find movies by an actor" )
@Procedure( name = "training.moviesOnly", allowed = "movies_only", mode = Mode.READ )
public Stream<Movie> moviesOnly( @Name( "name" ) String name )
       throws InvalidArgumentsException, IOException
{
   String query = "MATCH (:Person {name: {name}})-[:ACTED_IN]->(movie) RETURN movie";
   return db.execute( query, map("name", name) )
           .stream()
           .map( Movie::new );
}
```

Locked down roles

```
dbms.security.procedures.roles=
     training.moviesOnly:movies_only;
     training.writeProcedure:allowed_role;
     training.waitFor:allowed_role;
```

Locked down roles

```
dbms.security.procedures.roles=
     training.moviesOnly:movies_only;
     training.writeProcedure:allowed_role;
     training.waitFor:allowed_role;
```

Users with the 'movies_only' role can only execute the 'training.moviesOnly' procedure

Locked down roles

```
dbms.security.procedures.roles=
    training.moviesOnly:movies_only;
    training.writeProcedure:allowed_role;
    training.waitFor:allowed_role;
```

Users with the 'allowed_role' role can only execute the following procedures:

- training.writeProcedure
- training.waitFor

Locked down roles

```
dbms.security.procedures.roles=training.moviesOnly:movies_only;
training.writeProcedure:allowed_role;training.waitFor:allowed_role;
```

Add this to neo4j.conf and restart neo4j

Locked down roles

Try calling our locked down procedure:

```
CALL training.moviesOnly("Halle Berry")
```

End of Module
Security (single server)

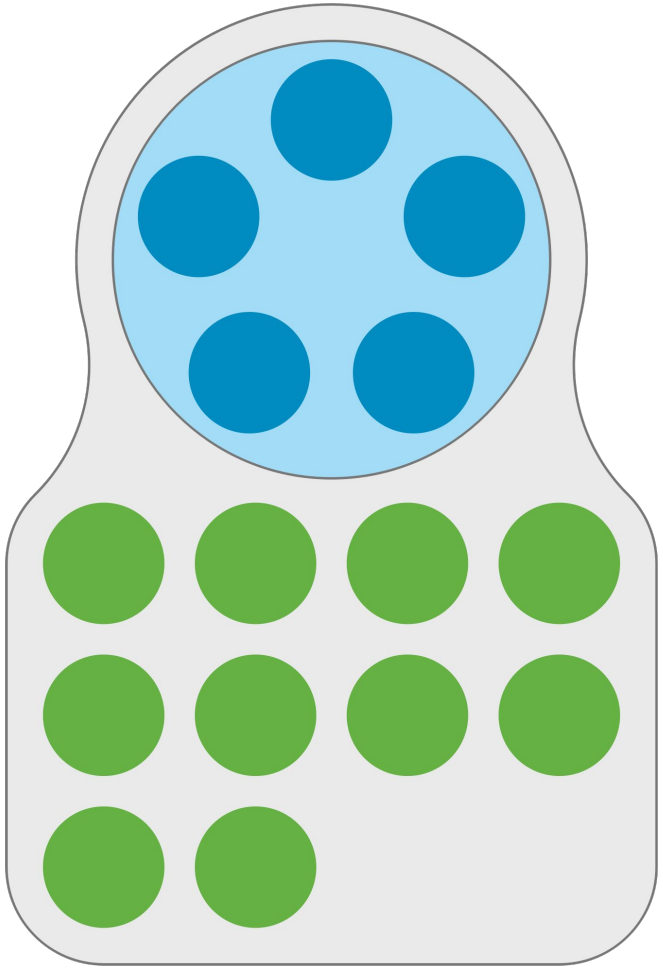
Questions ?

Causal Clustering

What are we going to do?

- Intro to Causal clustering
- Spin up a Causal cluster
- Cluster Security
- Causal Clustering

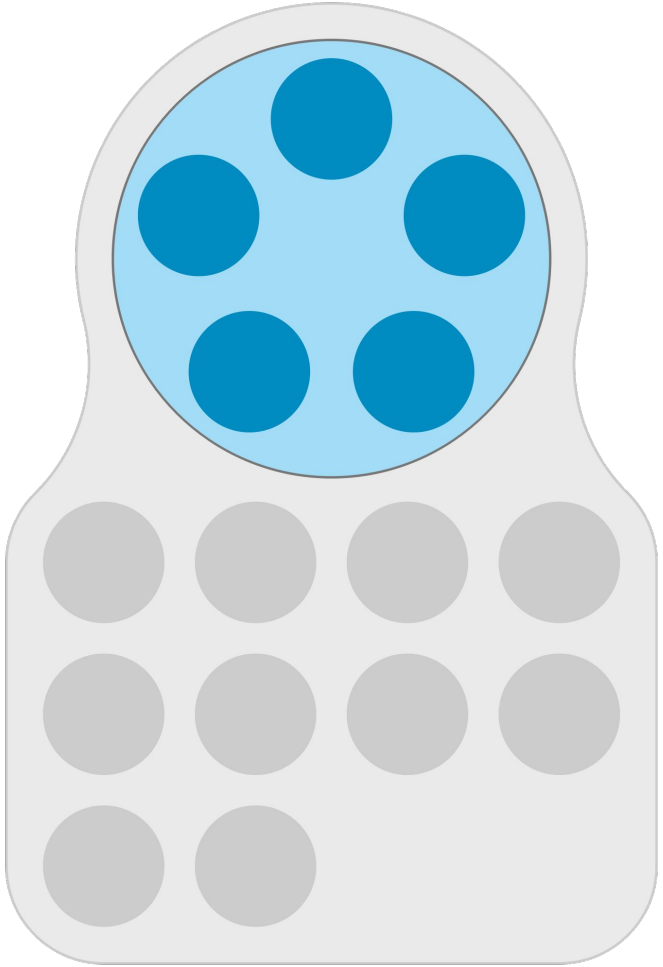
Intro to Causal clustering



neo4j

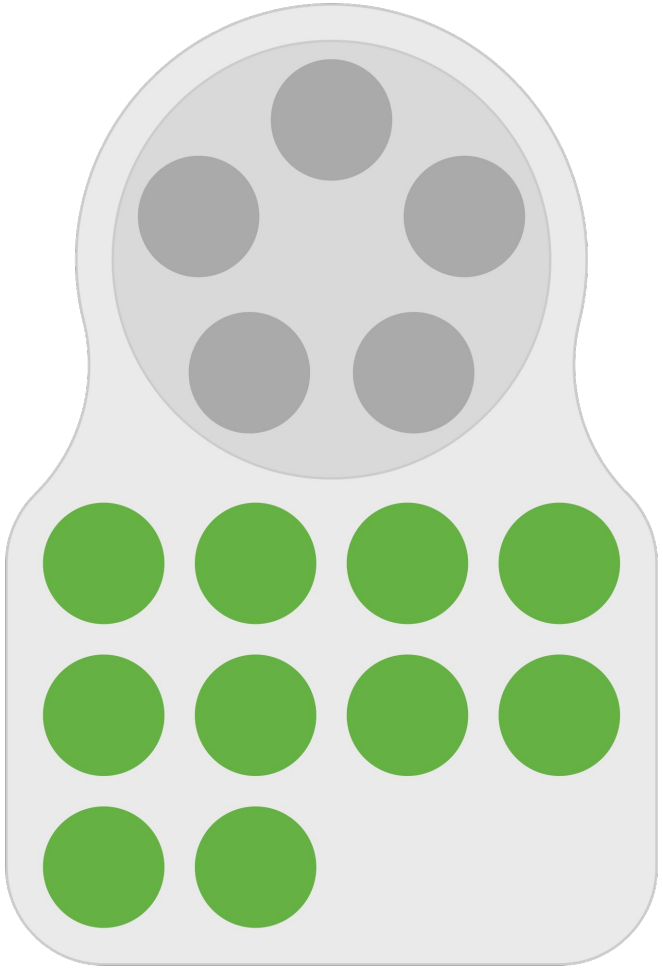
Core

Replica



Core

- Small group of Neo4j databases
- Implements Consensus Commit
- Responsible for data safety



Replica

- For massive query throughput
- Read-only replicas
- Not involved in Consensus Commit
  => less overhead
- Disposable, suitable for auto-scaling

Spinning up a Causal cluster

Download Neo4j Enterprise

Make sure you've got Neo4j Enterprise 3.1.4 on your machine. You can copy this from the USB stick or download from neo4j.com/download

Spin up a cluster

If you can't use the script, you can spin up a cluster manually:

- Make 4 copies of the neo4j directory e.g. `core_1 core_2 core_3 edge_1`
- In each directory:
  - Replace the config properties from bit.ly/2qHFnLC in

    `conf/neo4j.conf`
  - Start up each of the servers:

    `bin/neo4j start`

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *core_1/conf/neo4j.conf*

```
dbms.mode=CORE

dbms.connector.bolt.listen_address=:7691
dbms.connector.http.listen_address=:7481
dbms.connector.https.listen_address=:7371

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
causal_clustering.discovery_listen_address=127.0.0.1:5001
causal_clustering.transaction_listen_address=:6001
causal_clustering.raft_listen_address=:7001
```

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *core_1/conf/neo4j.conf*

```
dbms.mode=CORE

dbms.connector.bolt.listen_address=:7691
dbms.connector.http.listen_address=:7481
dbms.[illegible]=:7371

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
causal_clustering.[illegible]address=127.0.0.1:5001
causal_clustering.transaction_listen_address=:6001
causal_clustering.raft_listen_address=:7001
```

The default mode is 'SINGLE' so if we want a cluster we need to explicitly configure the mode.

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *core_1/conf/neo4j.conf*

```
dbms.mode=CORE

dbms.connector.bolt.listen_address=:7691
dbms.connector.http.listen_address=:7481
dbms.connector.https.listen_address=:7371

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
causal_clustering.discovery_listen_address=127.0.0.1:5001
causal_clustering.transaction_listen_address=:6001
causal_clustering.raft_listen_address=:7001
```

Set the various ports to listen on. These are used by the Neo4j browser and language drivers.

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *core_1/conf/neo4j.conf*

```
dbms.mode=CORE

dbms.connector.bolt.listen_address=:7691
dbms.connector.http.listen_address=:7481
dbms.connector.https.listen_address=:7371

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
causal_clustering.discovery_listen_address=127.0.0.1:5001
causal_clustering.transaction_listen_address=:6001
causal_clustering.raft_listen_address=:7001
```

The servers that form the cluster. This value will be the same on every cluster member

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *core_1/conf/neo4j.conf*

```
dbms.mode=CORE

dbms.connector.bolt.listen_address=:7691
dbms.connector.http.listen_address=:7481
dbms.connector.https.listen_address=:7371

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
causal_clustering.discovery_listen_address=127.0.0.1:5001
causal_clustering.transaction_listen_address=:6001
causal_clustering.raft_listen_address=:7001
```

Ports to listen on for cluster discovery, serving transactions, and consensus commit.

conf/neo4j.conf

These are the config properties that we've changed:

e.g. *edge_1/conf/neo4j.conf*

```
dbms.mode=READ_REPLICA

dbms.connector.bolt.listen_address=:7601
dbms.connector.http.listen_address=:7491
dbms.connector.https.listen_address=:7381

causal_clustering.initial_discovery_members=localhost:5001,localhost:5002,localhost:5003
```

:sysinfo

Navigate to http://localhost:7481 in your browser and type the following command:

```
:sysinfo
```

Core-Edge Cluster Members

Role	Addresses	Actions
READ_REPLICA	bolt://localhost:7601, http://localhost:7491, https://localhost:7381	Open
LEADER	bolt://localhost:7691, http://localhost:7481, https://localhost:7371	
FOLLOWER	bolt://localhost:7692, http://localhost:7482, https://localhost:7372	Open
FOLLOWER	bolt://localhost:7693, http://localhost:7483, https://localhost:7373	Open

Navigate to LEADER

If the server you just opened doesn’t have the LEADER role click on the 'Open' link next to the server which does have that role.

Core-Edge Cluster Members

Role	Addresses	Actions
READ_REPLICA	bolt://localhost:7601, http://localhost:7491, https://localhost:7381	Open
LEADER	bolt://localhost:7691, http://localhost:7481, https://localhost:7371	
FOLLOWER	bolt://localhost:7692, http://localhost:7482, https://localhost:7372	Open
FOLLOWER	bolt://localhost:7693, http://localhost:7483, https://localhost:7373	Open

Create data

Execute the following query to create some data:

```
UNWIND range(0, 100) AS value
MERGE (p1:Person {id: value})
MERGE (p2:Person {id: TOINT(100.0 * rand())})
MERGE (p1)-[:FRIENDS]->(p2)
```

Create data

Go to one of the servers with the `FOLLOWER` role by clicking on the 'Open' link next to it. Once you've done that execute the following query:

```
MATCH path = (p:Person)-[:FRIENDS]-(friend)
RETURN path
LIMIT 10
```

You can see that the data has been replicated.

Stop the LEADER

Find the server which has the LEADER role:

Core-Edge Cluster Members

Role	Addresses	Actions
READ_REPLICA	bolt://localhost:7601, http://localhost:7491, https://localhost:7381	Open
LEADER	bolt://localhost:7691, http://localhost:7481, https://localhost:7371	
FOLLOWER	bolt://localhost:7692, http://localhost:7482, https://localhost:7372	Open
FOLLOWER	bolt://localhost:7693, http://localhost:7483, https://localhost:7373	Open

And stop it using the following command:

```
./core_1/bin/neo4j stop
```

Look for the new LEADER

```
:sysinfo
```

Core-Edge Cluster Members

Role	Addresses	Actions
READ_REPLICA	bolt://localhost:7601, http://localhost:7491, https://localhost:7381	Open
FOLLOWER	bolt://localhost:7692, http://localhost:7482, https://localhost:7372	Open
LEADER	bolt://localhost:7693, http://localhost:7483, https://localhost:7373	

Create some more data

Execute the following query to create some more data:

```
UNWIND range(50, 150) AS value
MERGE (p1:Person {id: value})
MERGE (p2:Person {id: TOINT(150.0 * rand())})
MERGE (p1)-[:FRIENDS]->(p2)
```

Bring the old LEADER back again

Now start the old LEADER back up again.

Core-Edge Cluster Members

Role	Addresses	Actions
READ_REPLICA	bolt://localhost:7601, http://localhost:7491, https://localhost:7381	Open
LEADER	bolt://localhost:7691, http://localhost:7481, https://localhost:7371	
FOLLOWER	bolt://localhost:7692, http://localhost:7482, https://localhost:7372	Open
FOLLOWER	bolt://localhost:7693, http://localhost:7483, https://localhost:7373	Open

```
./core_1/bin/neo4j start
```

Cluster Security

Native Users Cluster

Flexible Authentication Options

Configuring LDAP connector

Use LDAP..

..at this address

..with system account defined

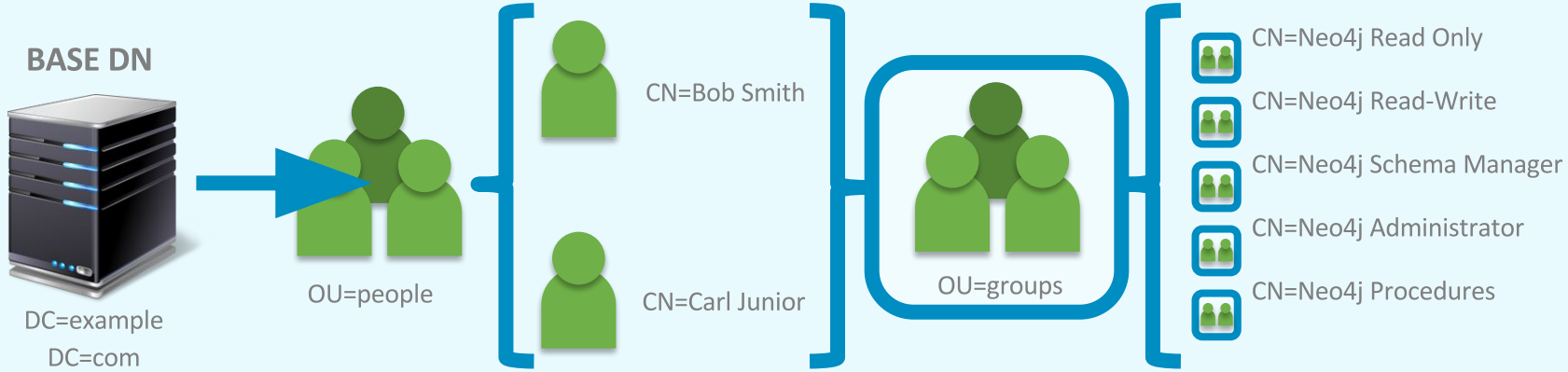
..with this directory structure

```
# Choose LDAP connector as both authentication and authorization provider
dbms.security.auth_provider=ldap

# Configure LDAP connector to point to the AD server
dbms.security.ldap.host=ldap://myactivedirectory.example.com

# In case where defined users are not allowed to search for themselves,
# we can specify credentials for user with read access to all users and groups
dbms.security.ldap.authorization.use_system_account=true
dbms.security.ldap.system_username=CN=admin,OU=people,DC=example,DC=com
dbms.security.ldap.system_password=admin-password

# Provide details on user structure within LDAP
dbms.security.ldap.user_dn_template=CN={0},OU=people,DC=example,DC=com
dbms.security.ldap.authorization.user_search_base=OU=people,dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=*)(CN={0}))
dbms.security.ldap.authorization.group_membership_attributes=memberOf
```

./conf/neo4j.conf

Flexible Authentication Options

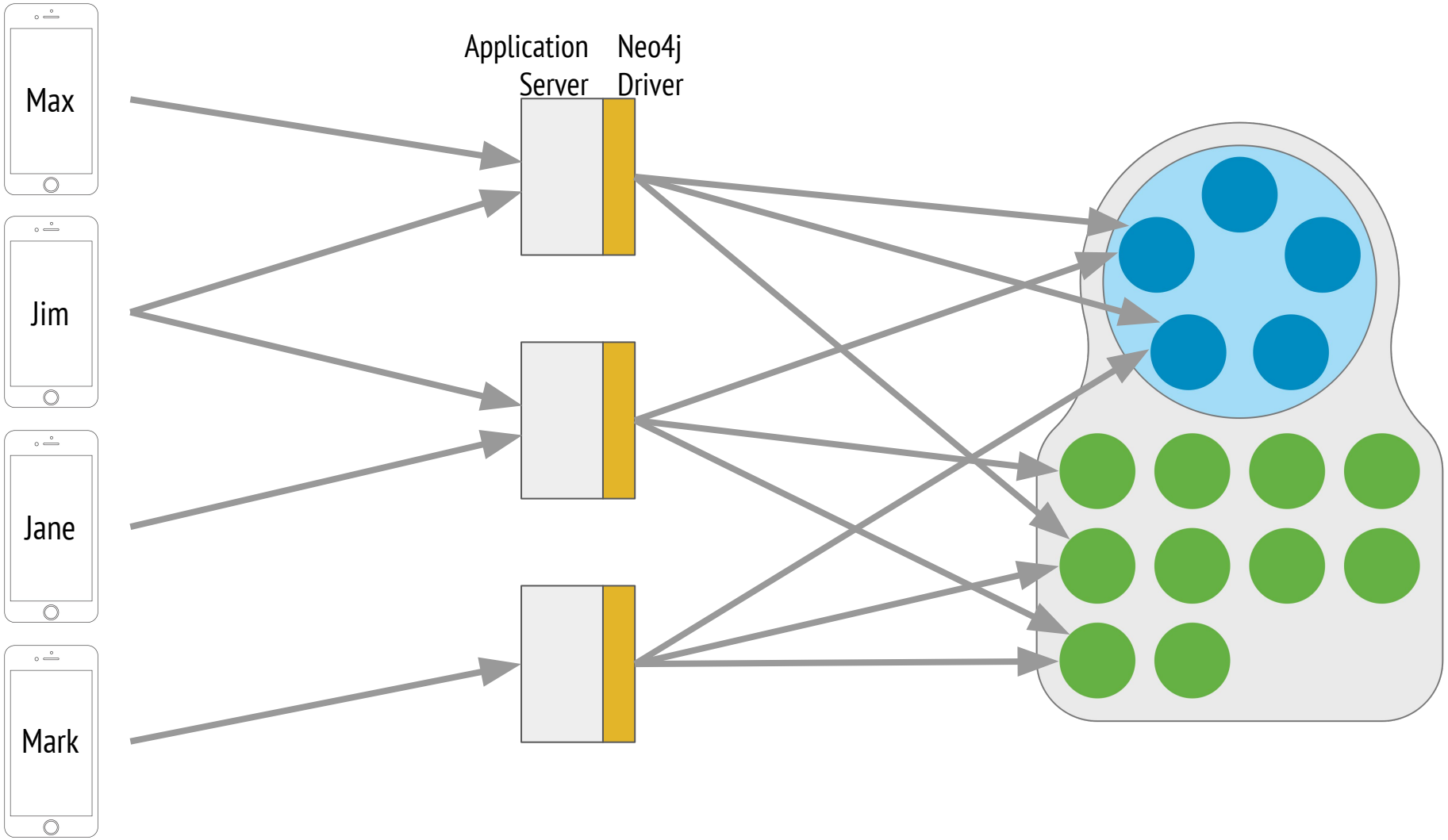
LDAP Group to Role Mapping

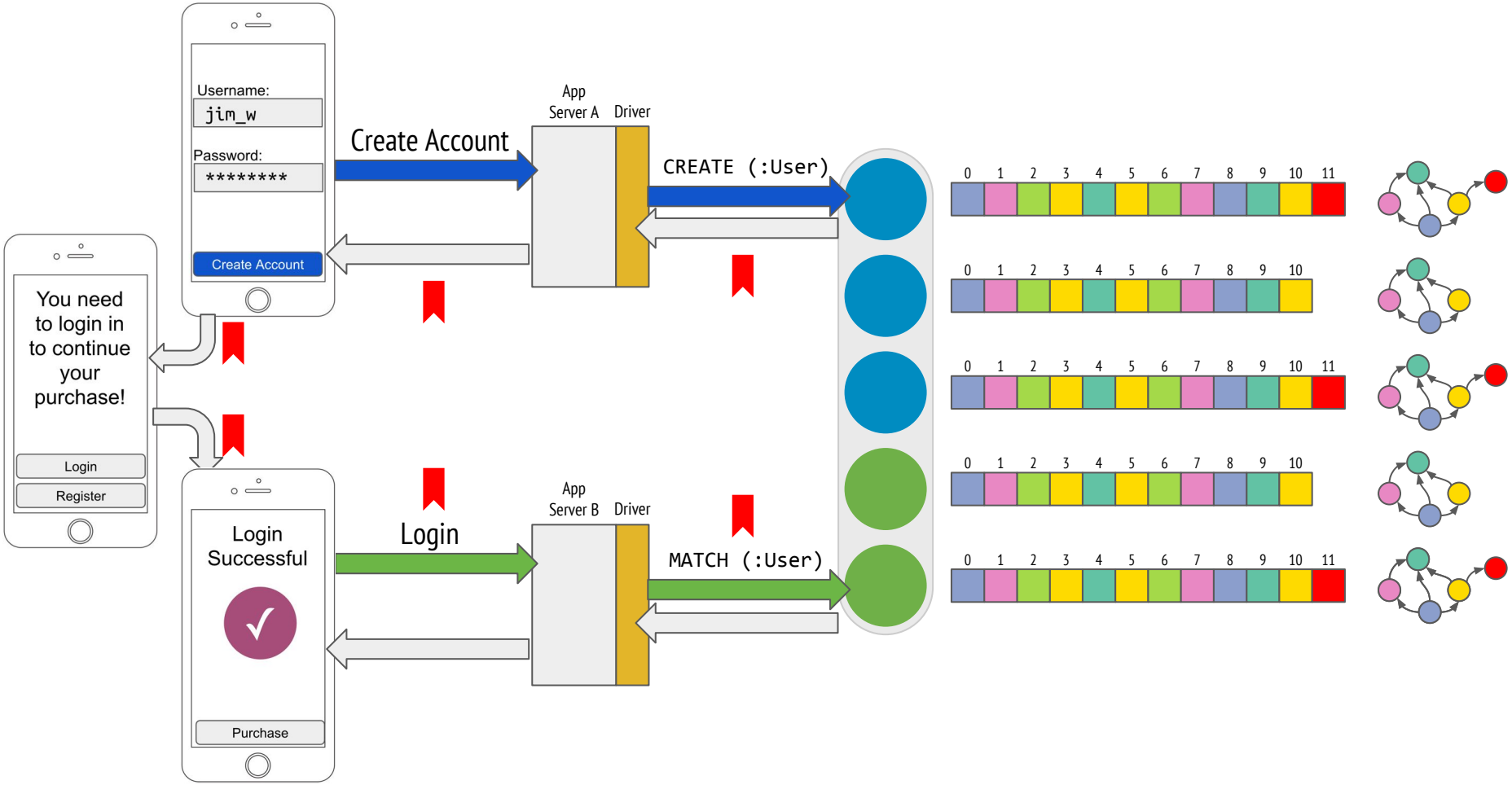



```
# Configure the actual mapping between groups in the LDAP and roles in
Neo4j
dbms.security.ldap.authorization.group_to_role_mapping= \
  "CN=Neo4j Read Only,OU=groups,DC=example,DC=com"      = reader; \
  "CN=Neo4j Read-Write,OU=groups,DC=example,DC=com"     = publisher; \
  "CN=Neo4j Schema Manager,OU=groups,DC=example,DC=com" = architect; \
  "CN=Neo4j Administrator,OU=groups,DC=example,DC=com"  = admin; \
  "CN=Neo4j Procedures,OU=groups,DC=example,DC=com"     = allowed_role
```

./conf/neo4j.conf

Causal Clustering


Max
Jim
Jane
Mark
Application Server
Neo4j Driver


Username:
jim_w
Password:
****
Create Account
Create Account
App
Server A
Driver
CREATE (:User)
You need to login in to continue your purchase!
Login
Register
Login Successful
Purchase
Login
App
Server B
Driver
MATCH (:User)
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11
0 1 2 3 4 5 6 7 8 9 10
0 1 2 3 4 5 6 7 8 9 10 11

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
   try ( Session session = driver.session() ) {

       session.writeTransaction( (tx) -> {
           return tx.run( "CREATE (person:Person {name: {name}, title: {title}})",
                   parameters( "name", "Webber", "title", "Dr" ) );
       } );

       String title = session.readTransaction( (tx) -> {
           return tx.run( "MATCH (person:Person {name: {name}}) RETURN person.title",
                           parameters( "name", "Webber" ) ).single().get( 0 ).asString();
       } );

       System.out.println( "Webber's title is " + title );
   }
}
```

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
    try        ion session = driver.session() ) {

                                    -> {
                                    rson:Person {name: {name}, title: {title}})",
                                    , "Webber", "title", "Dr" ) );

        String title = session.readTransaction( (tx) -> {
            return tx.run( "MATCH (person:Person {name: {name}}) RETURN person.title",
                           parameters( "name", "Webber" ) ).single().get( 0 ).asString();
        } );

        System.out.println( "Webber's title is " + title );
    }
}
```

Application holds one global instance of the driver. The Driver is thread safe.

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
    try        ion session = driver.session() ) {

                                    -> {
                                    rson:Person {name: {name}        : {title}})",
                                    , "Webber", "title", "Dr"

        String title = session.readTransaction( (tx) -> {
            return tx.run( "MATCH (person:Person {name: {
                           parameters( "name", "Webber" )
        } );

        System.out.println( "Webber's title is " + title );
    }
}
```

Application holds one global instance of the driver. The Driver is thread safe.

Bolt routing URI allows access to automatic cluster member routing behaviour.

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
    try ( Session session = driver.session() ) {

        session.writeTransaction( (tx) -> {
            return tx.run( "CREATE (person:Person {name: {name}, title: {title}})",
                           parameters( "name", "Dr", "title", "Dr" ) );
        } );

        String title = session.readTransaction( (tx) -> {
            return tx.run( "MATCH (person:Person {name: {name}}) RETURN person.title",
                           parameters( "name", "Webber" ) ).single().get( 0 ).asString();
        } );

        System.out.println( "Webber's title is " + title );
    }
}
```

The Session is a logical context for transactional units of work. A Session is not thread safe.

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
   try ( Session session = driver.session() ) {

       session.writeTransaction( (tx) -> {
           return tx.run( "CREATE (person:Person {name: {name}, title: {title}})",
                   parameters( "name", "Webber", "title", "Dr" ) );
       } );

       String title =                  n.readTransaction( (tx) -> {
           ret                                         me: {name}}) RETURN person.title",
                                                       er" ) ).single().get( 0 ).asString();
       } );

       System.                                         itle );
   }
}
```

A unit of work is encapsulated within a *Transaction Function*. If execution fails, retries are attempted automatically.

Causal Clustering: Read your own writes

Cypher results should be fully consumed within a transaction function but values can be returned.

```
try ( Driver driver = G                                    lhost:7693",
                                                           eo4j", "abc" ) ) ) {
   try ( Session sessio

      session.writeTra
         return tx.ru                                      title: {title}})",
                  parameters                  Webber ,  title ,  Dr  ) );
      } );

      String title = session.readTransaction( (tx) -> {
         return tx.run( "MATCH (person:Person {name: {name}}) RETURN person.title",
                        parameters( "name", "Webber" ) ).single().get( 0 ).asString();
      } );

      System.out.println( "Webber's title is " + title );
   }
}
```

Causal Clustering: Read your own writes

```
try ( Driver driver = GraphDatabase.driver( "bolt+routing://localhost:7693",
                                            AuthTokens.basic( "neo4j", "abc" ) ) ) {
   try ( Session session = driver.session() ) {

       session.writeTransaction( (tx) -> {
           return tx.run( "CREATE (person:Person {name: {name}, title: {title}})",
                   parameters( "name", "Webber", "title", "Dr" ) );
       } );

       String title = session.readTransaction( (tx) -> {
           return tx.run( "MATCH (person:Person {name: {name}}) RETURN person.title",
                           parameters( "name", "Webber" ) ).single().get( 0 ).asString();
       } );

       System.out.println( "Webber's title is " + title );
   }
}
```

Causal Clustering Demo

End of Module
Core-Edge Clustering

Questions?

What are we going to do?

Now we're ready to move from our single server to a clustered server setup.

Step 1: Stop the database
Step 2: Create a backup
Step 3: Restore the backup on each of the servers
Step 4: Start the cluster

What are we going to do?

Run a query on the leader and watch it propagate

Demo of config on separate servers?

Gist that we can copy/paste from

Docker instructions? Steal Jonas' script (try it out)

Download cluster bootstrapping script

Your USB stick contains a file called `neo4j-cluster-rsync-auth-master.zip` which we'll use to create a local cluster. Copy this file onto your machine and unpack it.

If you don't have a USB stick you can download it from here:

https://github.com/craigtaverner/neo4j-cluster-rsync-auth

Spin up a cluster

Set the following environment variables:

```
export NEO4J_VERSION="3.1.4"
export NEO4J_PASSWORD="abc"
export NUMBER_CORES=3
export NUMBER_EDGES=1
```

Run the installation script:

```
./install.sh
```

Spin up a cluster

Set the following environment variables:

```
export NEO4J_VERSION="3.1.4"
export NEO4J_PASSWORD="abc"
export NUMBER_CORES=3
export NUMBER_EDGES=1
```

Run the installation script:

```
./install.sh
```

If you can't use the script, you can spin up a cluster manually:

- Make 3 copies of the neo4j directory e.g. `core_1 core_2 core_3 edge_1`
- In each directory:
  - Copy config from http://bit.ly/gcsf2016-prod into `conf/neo4j.conf`
  - Start up the server:

    `bin/neo4j start`

Native Users Cluster Demo

Flexible Authentication Options

Configuring LDAP connector

Use LDAP..

..at this address

..with system account defined

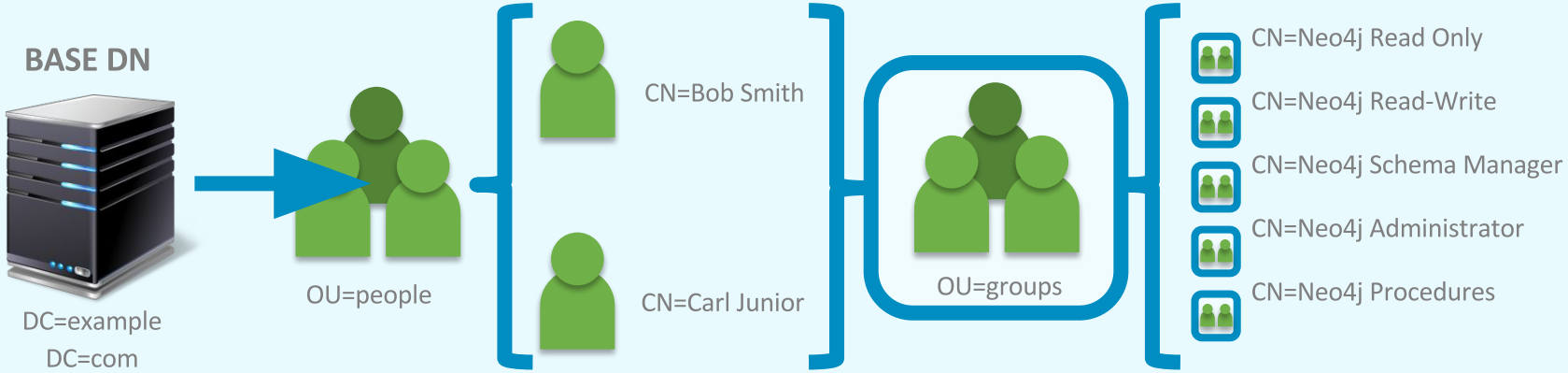
..with this directory structure

```
# Choose LDAP connector as both authentication and authorization provider
dbms.security.auth_provider=ldap

# Configure LDAP connector to point to the AD server
dbms.security.ldap.host=ldap://myactivedirectory.example.com

# In case where defined users are not allowed to search for themselves,
# we can specify credentials for user with read access to all users and groups
dbms.security.ldap.authorization.use_system_account=true
dbms.security.ldap.system_username=CN=admin,OU=people,DC=example,DC=com
dbms.security.ldap.system_password=admin-password

# Provide details on user structure within LDAP
dbms.security.ldap.user_dn_template=CN={0},OU=people,DC=example,DC=com
dbms.security.ldap.authorization.user_search_base=OU=people,dc=example,dc=com
dbms.security.ldap.authorization.user_search_filter=(&(objectClass=*)(CN={0}))
dbms.security.ldap.authorization.group_membership_attributes=memberOf
```

./conf/neo4j.conf

Flexible Authentication Options

LDAP Group to Role Mapping




```
# Configure the actual mapping between groups in the LDAP and roles in
Neo4j
dbms.security.ldap.authorization.group_to_role_mapping= \
  "CN=Neo4j Read Only,OU=groups,DC=example,DC=com"      = reader; \
  "CN=Neo4j Read-Write,OU=groups,DC=example,DC=com"     = publisher; \
  "CN=Neo4j Schema Manager,OU=groups,DC=example,DC=com" = architect; \
  "CN=Neo4j Administrator,OU=groups,DC=example,DC=com"  = admin; \
  "CN=Neo4j Procedures,OU=groups,DC=example,DC=com"     = allowed_role
```

./conf/neo4j.conf

Active Directory Demo

Reconfigure cluster for Active Directory

Run the following script:

```
./configure_provider.sh AD
```

If you can't use the script, you can configure the cluster manually:

- For each server:
  - Copy the contents of neo4j-AD.conf from http://bit.ly/neo4j-prod-security into the end of `conf/neo4j.conf`
  - Restart the server:

    ```
    bin/neo4j restart
    ```

Reconfigure cluster for Active Directory

After the cluster has restarted, reload the browser tabs and login again, but now the password will have changed and the admin user is called ‘morpheus’.