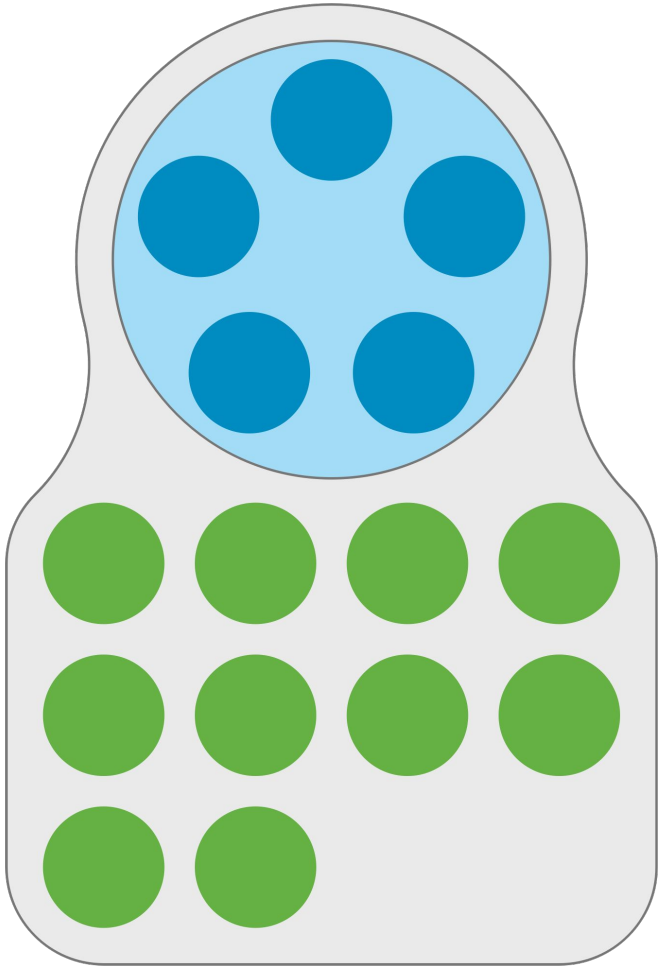
Causal Clustering

What are we going to do?

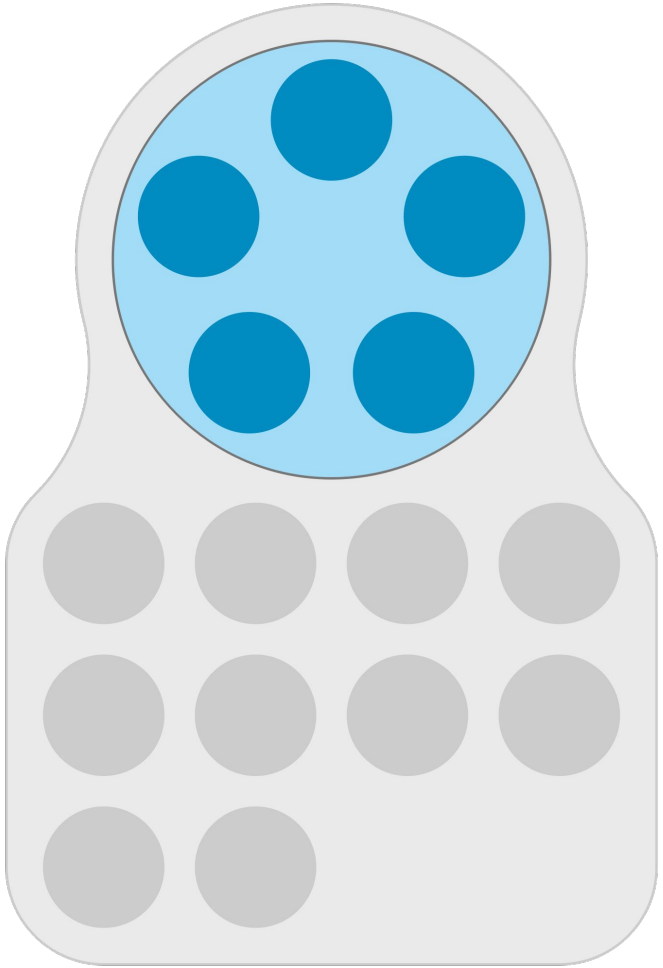
- Intro to Causal clustering
- Spin up a Causal cluster
- Cluster Security
- Causal Clustering

Intro to Causal clustering



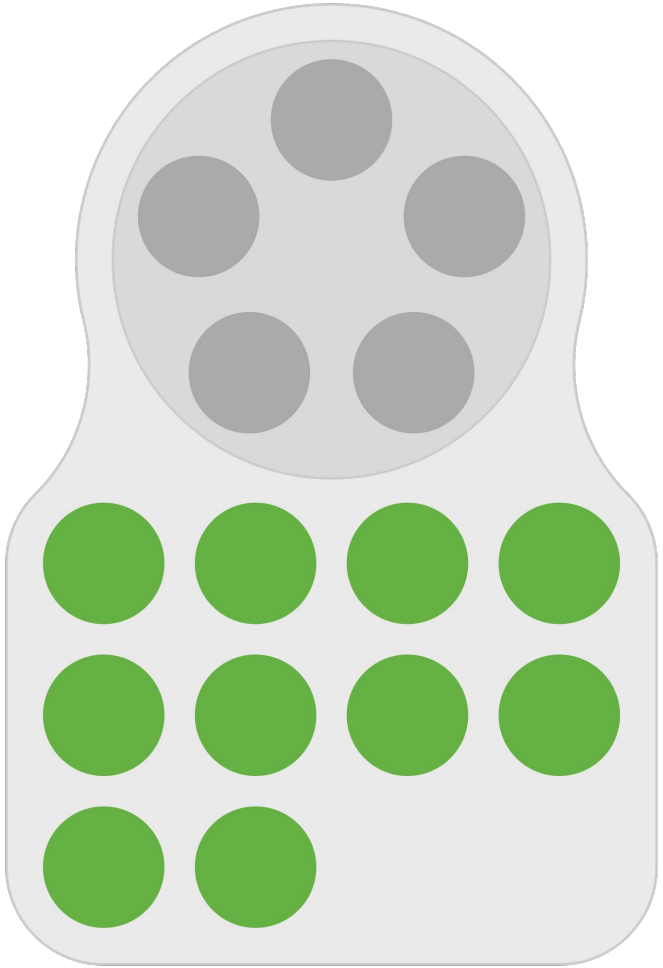
Core

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 3.4.5 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.backup.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 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:

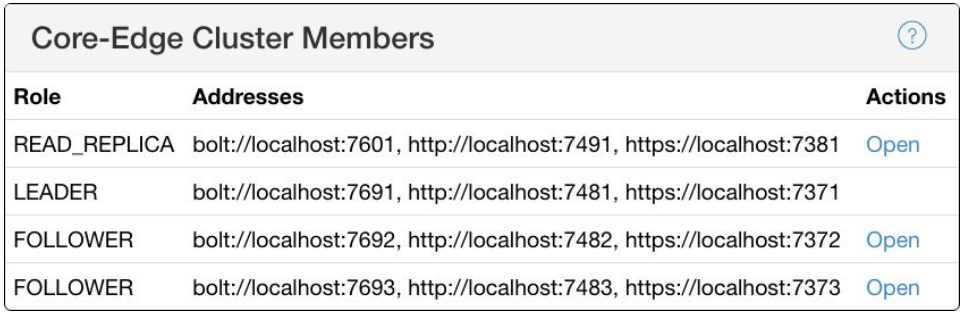
e.g. *edge_1/conf/neo4j.conf*

```
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

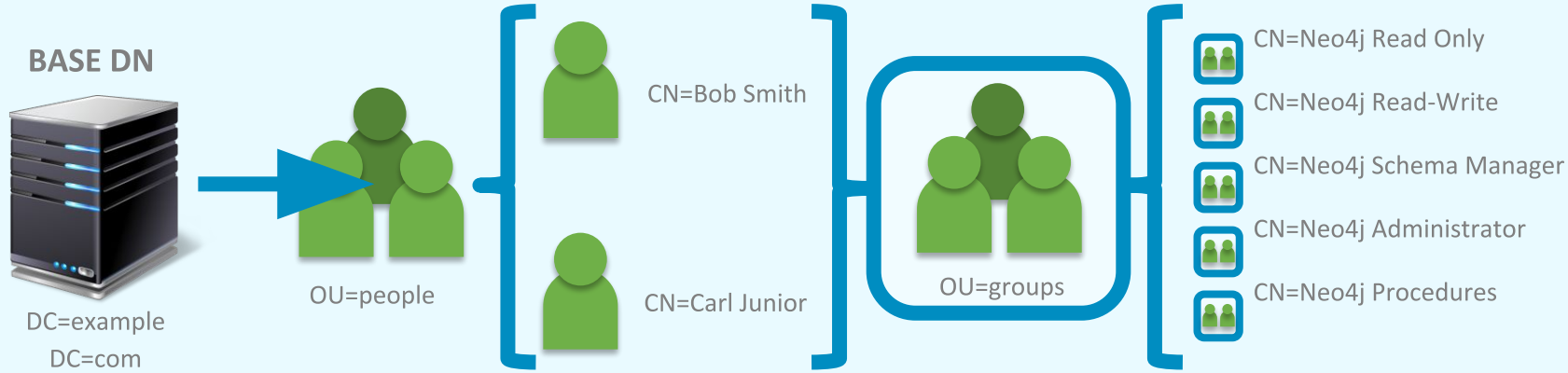
..with this directory structure

```
# 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

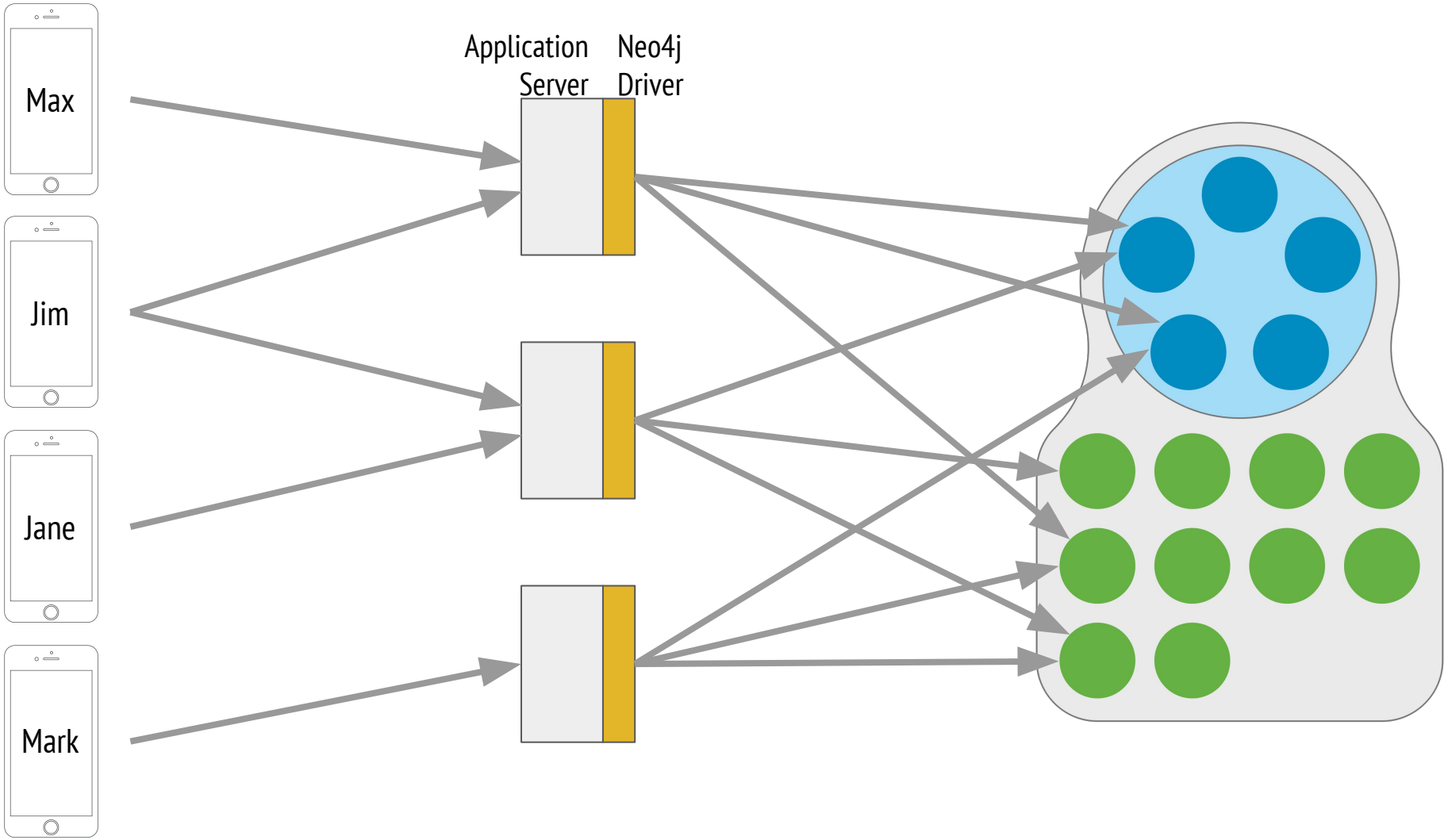
LDAP Group to Role Mapping

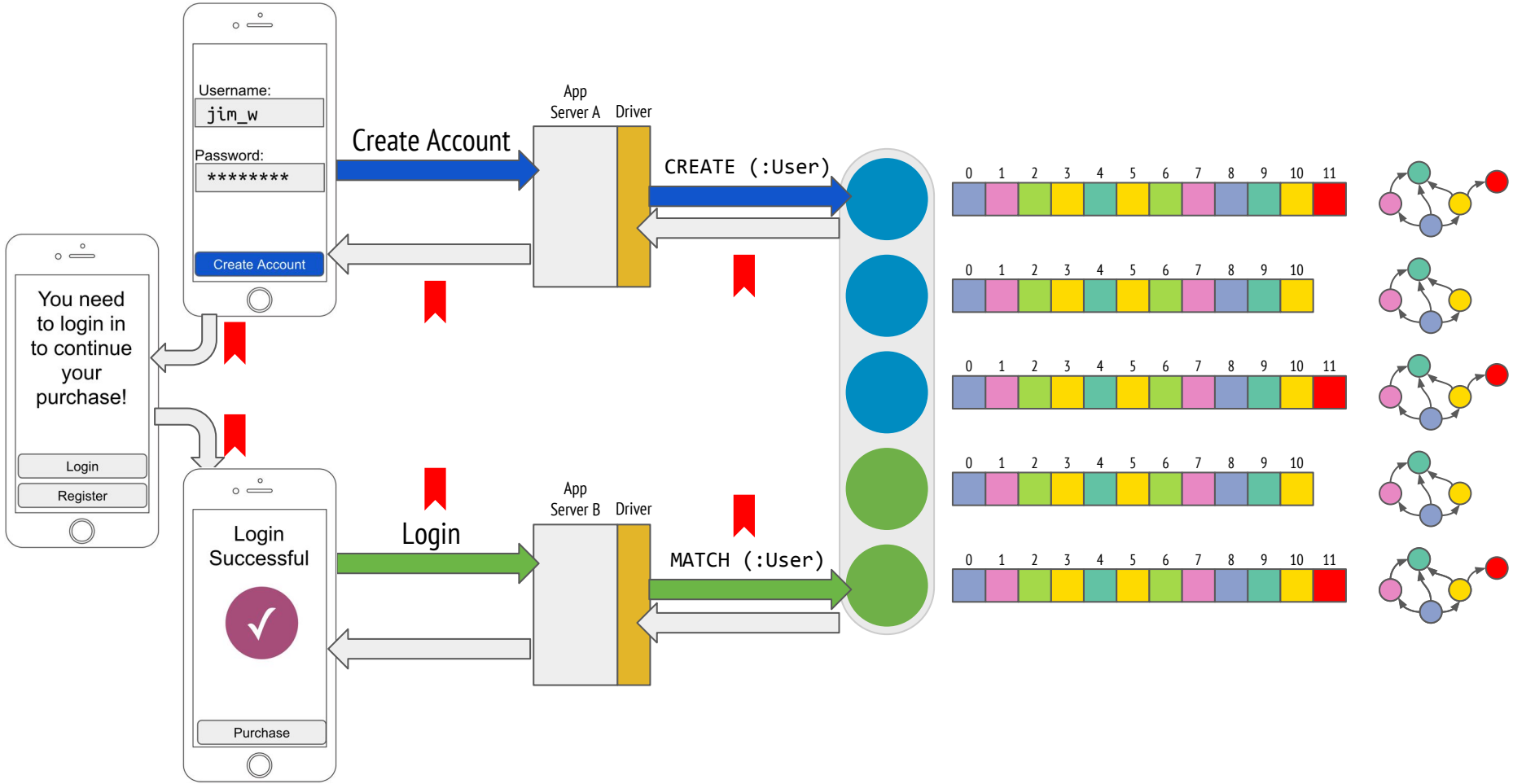



```
# Configure the actual mapping between groups in the LDAP and roles in
Neo4j
dbms.security.ldap.authorization.group_to_role_mapping= \
  "CN=Neo4j Read Only,OU=groups,DC=example,DC=com"      = reader; \
  "CN=Neo4j Read-Write,OU=groups,DC=example,DC=com"     = publisher; \
  "CN=Neo4j Schema Manager,OU=groups,DC=example,DC=com" = 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
You need to login in to continue your purchase!
Login
Register
Login Successful
Purchase
Create Account
Login
App Server A
Driver
App Server B
Driver
CREATE (:User)
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 ( Session session = driver.session() ) {

        session.writeTransaction( (tx) -> {
            tx.run( "MERGE (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 );
    }
}
```

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 ( Session session = driver.session() ) {

        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.writ      saction( (tx) -> {
            return      "CREATE (person:Person {name: {name}, title: {title}})",
                                                  "title", "Dr" ) );
        } );

        Strin                                     tx) -> {
                                                  {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 = session.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.

```
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

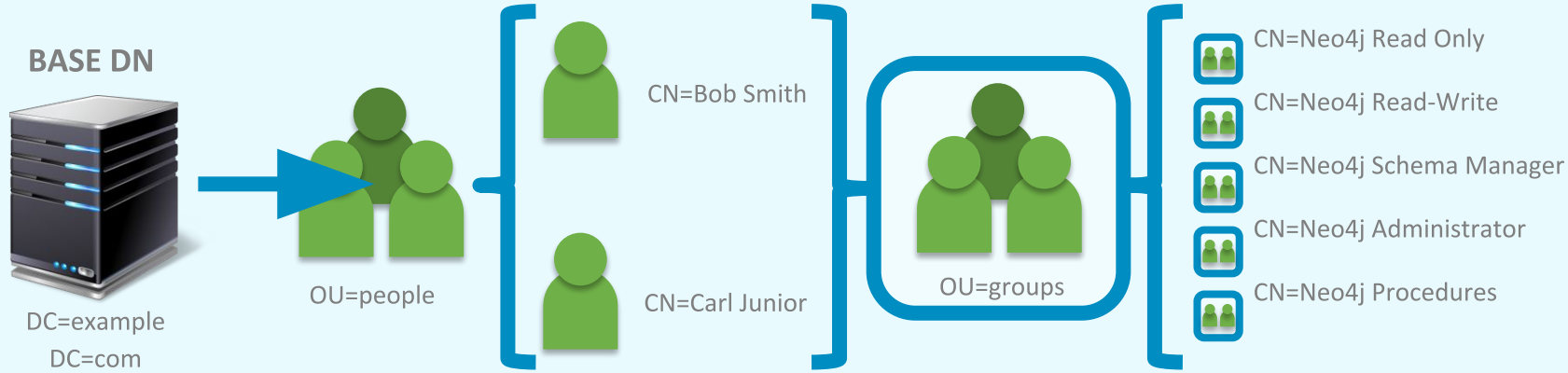
..with this directory structure

```
# 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’.