Managing Cassandra

Apache Cassandra:
Operations and Performance Tuning

Learning Objectives

- **Add new nodes into a cluster**
- Understand cleanup operations
- Remove downed nodes
- Decommission nodes
- Replace downed nodes

Node Setup Review

- Four parameters of a node
  - cluster_name
  - rpc_address
  - listen_address
  - -seeds
- These are configured in the *cassandra.yaml* file.

Why would you add new nodes to a cluster?

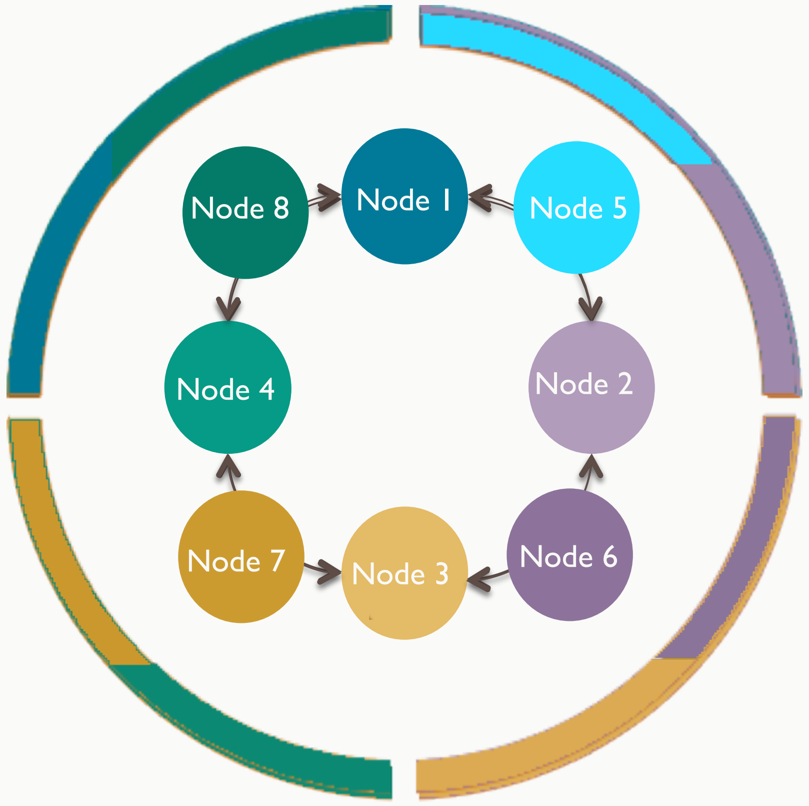
- Too much data – Your data has outgrown the node's hardware capacity.
  - Disk space may be too full – scale up.
- Too much traffic – Your application needs more rapid response with less latency.
  - If nodes have less data, then the hit rate on each will decrease.
  - Memory may be too small for data – scale out.
- Not enough operational headroom
  - Need more resources for node repair, compaction, and other resource intensive operations.

Starting the first node

- Seed node provider has to equal the listen_address
  - You are your own seed, and thus you are a single node cluster.
- You will never gossip with anyone else.

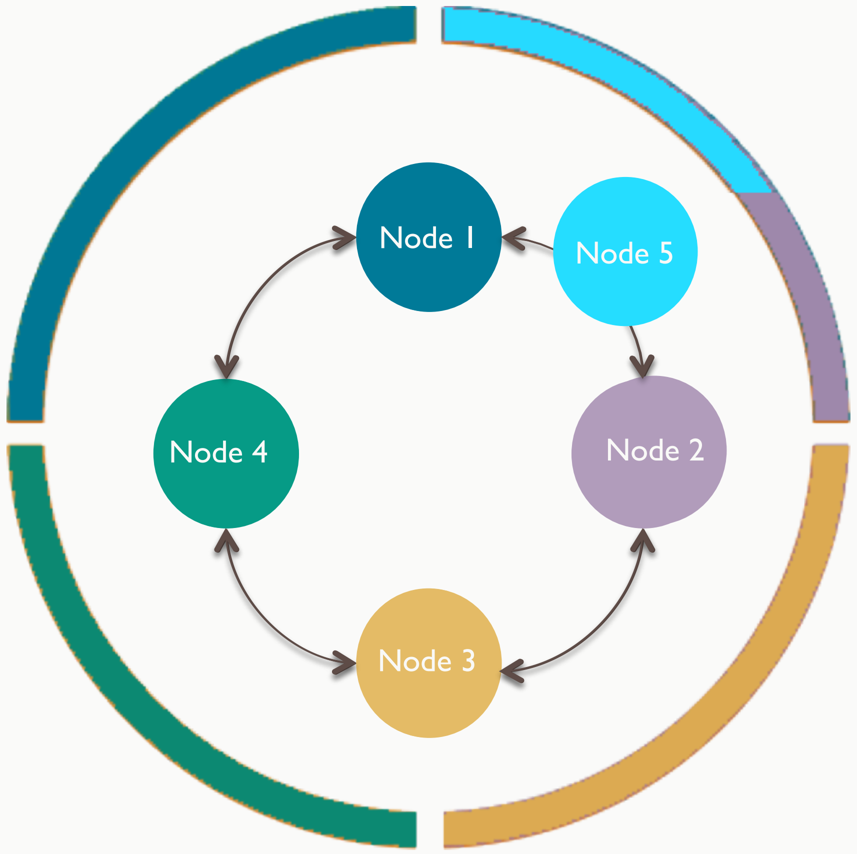
What are best practices for adding nodes?

- Single-token clusters: double the size of a cluster
  - Can minimize latency impact to a production load, where token recalculation and data movement can affect performance.
  - Hot spots are minimized during data movement to new nodes.



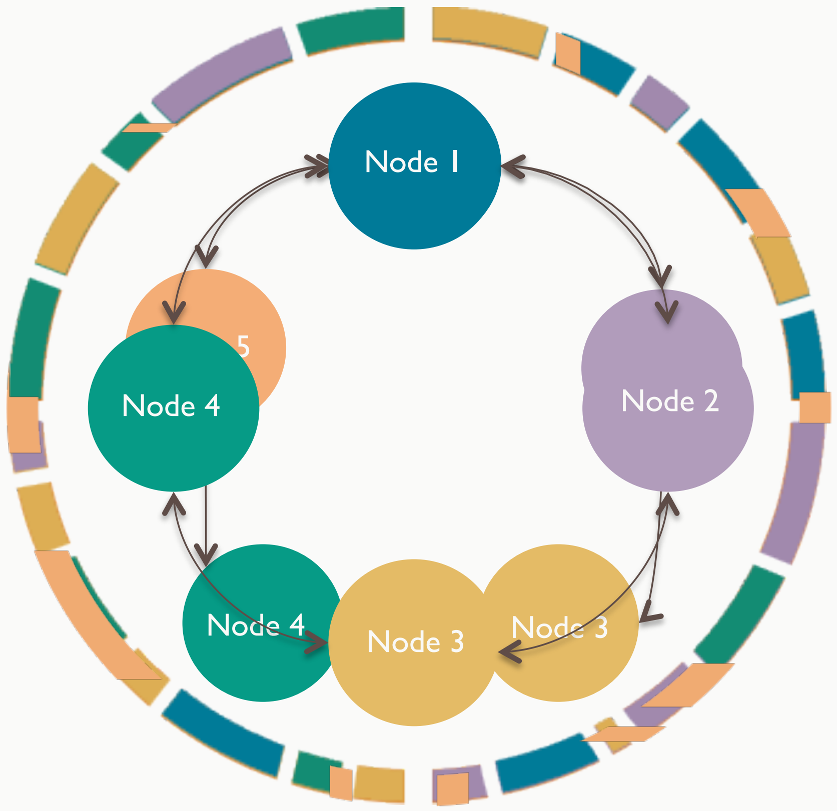
What are NOT best practices for adding nodes?

- Single-token clusters: adding one node at a time
  - This may cause cluster to become unbalanced.
  - Don't leave the cluster unbalanced.



What are best practices for adding nodes?

- Vnode clusters: Increment the size of the cluster if more nodes are needed.
  - Token ranges are distributed, and token range assignment is automatic.
  - Several token ranges will be split on each node, therefore each node will stream some data.



What are best practices for adding nodes?

- Token generation for a doubled single-token cluster is simple.
  - Halve the size of each range.
- Adding a single node creates a much more complex token generation scheme.
  - A token inserted between two others will leave cluster unbalanced.
  - Have to regenerate all tokens and move data to rebalance ring.
- Simple formula may be used to generate tokens.
  - Divide the hash range by the number of nodes in cluster.
  - Murmur3Partitioner (default) – Find the maximum possible range by calculating -2^{63} to $(2^{63} - 1)$ and divide by number of nodes.

What are best practices for adding nodes?

- Adding multiple nodes to a cluster at the same time
  - Vnode cluster – start up all nodes at once, otherwise you will end up pushing the same data to new nodes more than once as the data reshuffles its position.
  - Single-token cluster – each node added will split one node in half. Starting them all at the same time will not change how data is written to each new node.

What is the best solution in this case?

- 3-node cluster configuration
  - 100 GB of data spread over three nodes
  - Each node has 50 GB hard disk
  - 8-core CPU
  - 8 GB of RAM
  - Configured with regular nodes
  - Starting to experience latencies of over 1 ms

What is the best solution in this case?

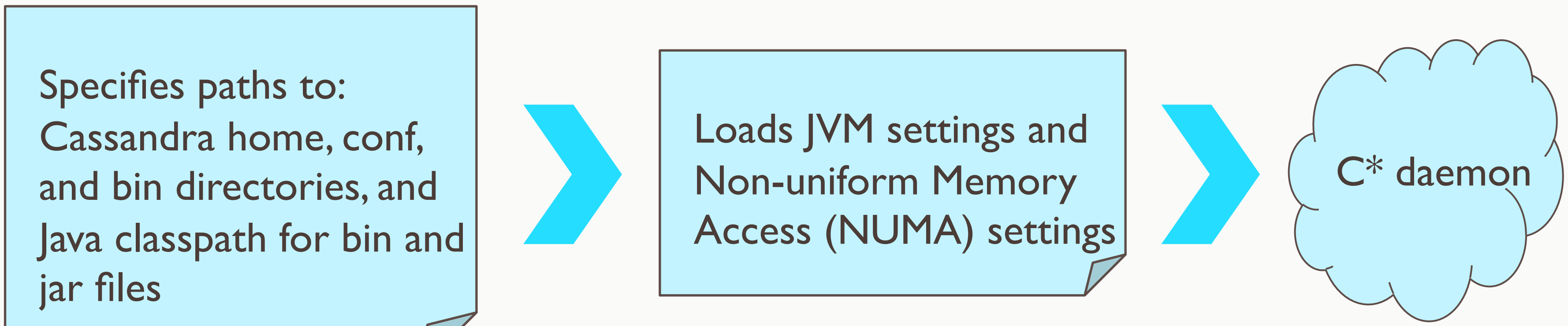
- 100-node cluster configuration
  - 10 TB of data spread over 100 nodes
  - Each node has 100 GB hard disk
  - 8-core CPU
  - 8 GB of RAM
  - Configured with Vnodes
  - Available disk space is filled at 95% capacity

What is bootstrapping in Cassandra?

- After a node joins a ring, bootstrapping is the process by which the new node gets its data from other nodes.
- Before bootstrapping starts:
  - The Cassandra shell file executes
  - The Cassandra daemon runs
- Finally, bootstrapping kicks off

How is Cassandra started?

- Run “cassandra”
  - Shell script that runs initialization actions
  - Sets paths
  - Loads settings
  - Starts the daemon



Specifies paths to: Cassandra home, conf, and bin directories, and Java classpath for bin and jar files

Loads JVM settings and Non-uniform Memory Access (NUMA) settings

C* daemon

Searches for cassandra.in.sh file and sources it

References cassandra-env.sh file

Cassandra daemon starts

What is the bootstrapping process?

1

- Joining nodes contact a seed node.

2

- Seed node sees incoming communication from the joining node.
- Shares cluster information with the joining node
  - Includes existing tokens that are assigned
- Seed node and joining node handshake

3

- Joining node calculates token(s) it is responsible for
- Shares with the seed node
- Seed node and joining node handshake again

4

- Existing nodes prepare to stream data to joining node
- 30 second pause occurs to do the following tasks for durability, in case of bootstrap failure
  - Flush data in Memtables to disk
  - New system keyspace information flushes to disk for joining node

What is the bootstrapping process? (con't.)

6
- Existing nodes locate appropriate keys from SSTables for data to stream. streamed

7
- Existing nodes stream only the SSTables that hold keys for the new node.
  - No data is removed from existing nodes – could be lost!

8
- Writes during this streaming period continue to be written to the existing node which owns it.
- These writes get forwarded to the new node.

9
- Joining node switches from JOINING to NORMAL state.
  - Write and read requests will now go to the new node.

10
- New node starts listener service for CQL calls (port 9042) and Thrift calls (port 9160).

How do you bootstrap new nodes?

- Provision a machine to hold new Cassandra instance
- Install Java and JNA
- Install Cassandra
- Start Cassandra on new node and it will automatically join
- Verify the node is up using *nodetool status*

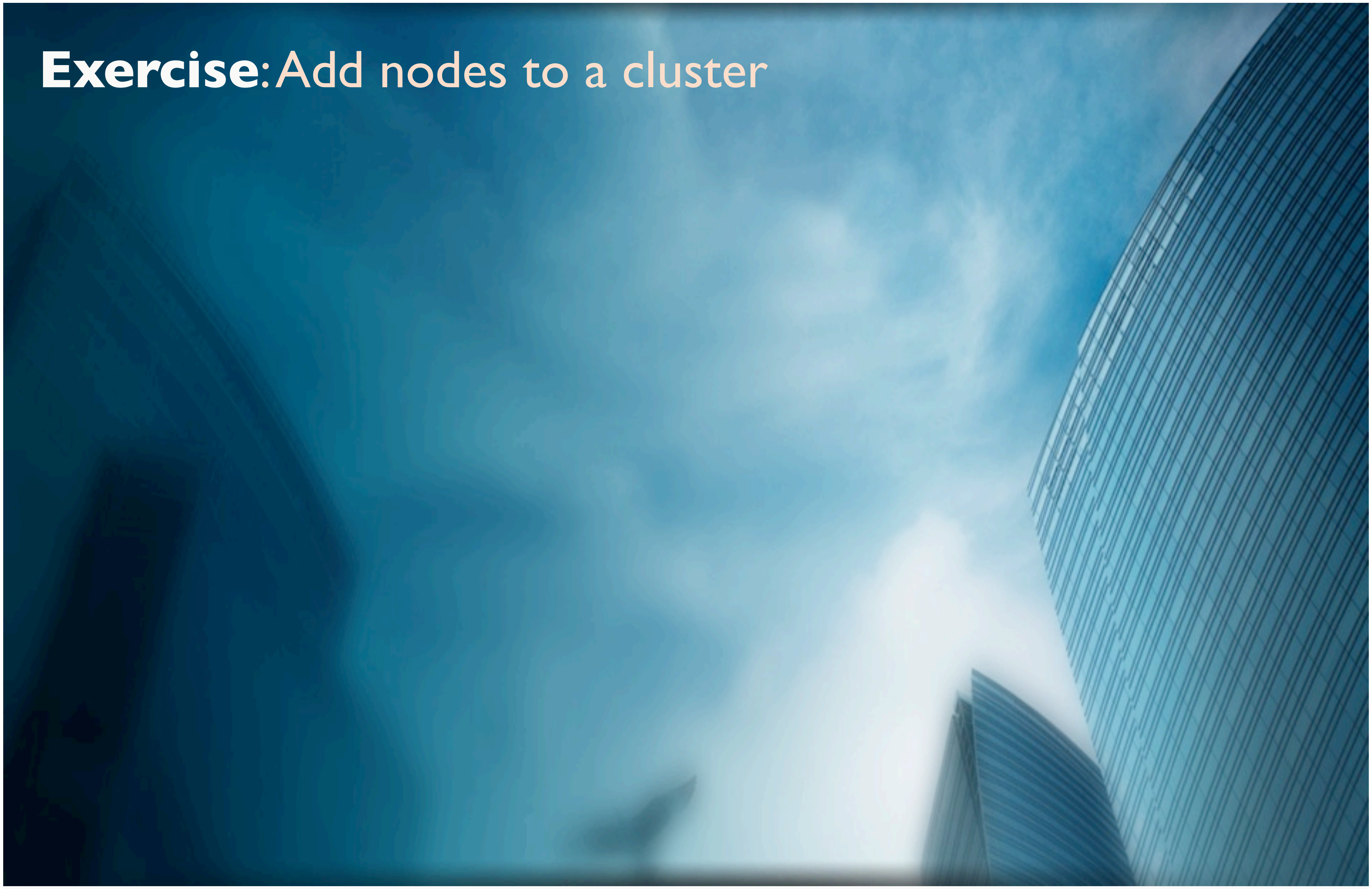
When not to automatically bootstrap a node

- *auto_bootstrap* is a setting that can be added to the *cassandra.yaml* file to disable automatic bootstrapping.
- Might be useful if you are adding several nodes and want to start them individually with *auto_bootstrap: false*
- Same for when adding a data center

Node Troubleshooting

- If a node fails to come up or bootstrap, you need to:
  - Examine the log file to understand what's going on
  - Shut down the node
  - Fix the configuration
  - Wipe your data directories, for example:
    - rm -rf /var/lib/cassandra/*/*
  - Start node again

Exercise: Add nodes to a cluster



Learning Objectives

- Add new nodes into a cluster
- **Understand cleanup operations**
- Remove downed nodes
- Decommission nodes
- Replace downed nodes

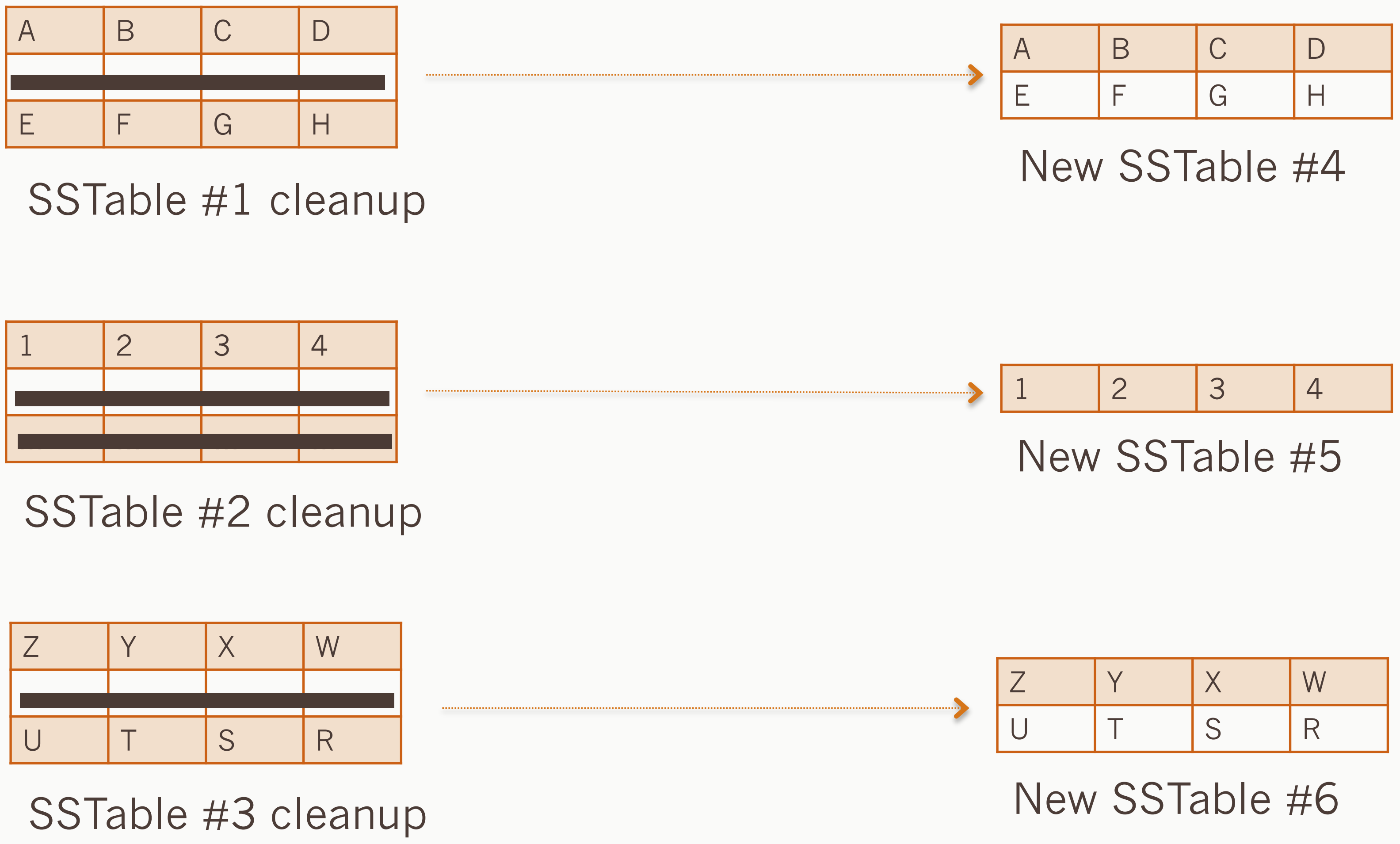
Why would I need to run a cleanup operation?

- Added new nodes to cluster, decreased replication factor or moved tokens
  - When a new node is added, each node in cluster shifts part of its partition range to the new node.
  - The stale data is not automatically removed from "old" nodes until a cleanup operation is done.
  - If the replication factor is decreased, similar stale data will exist on some nodes in cluster.
- Not needed to be done regularly
  - Main reason for doing cleanup is to recover disk space.
- Cleanup operations can be delayed until no performance impact will occur.

What does a cleanup operation do?

- Clean keyspace by writing new SSTables by skipping the keys that no longer belong on the node.
- Rewrites every SSTable
  - The largest SSTable dictates the amount of space needed for operation.
  - All SSTables are cleaned one table at a time.
  - If SSTable is already clean, cleanup operation skips it.
- Cleanup is basically a compaction!

What does cleanup look like?



How do you execute cleanup using *nodetool*?

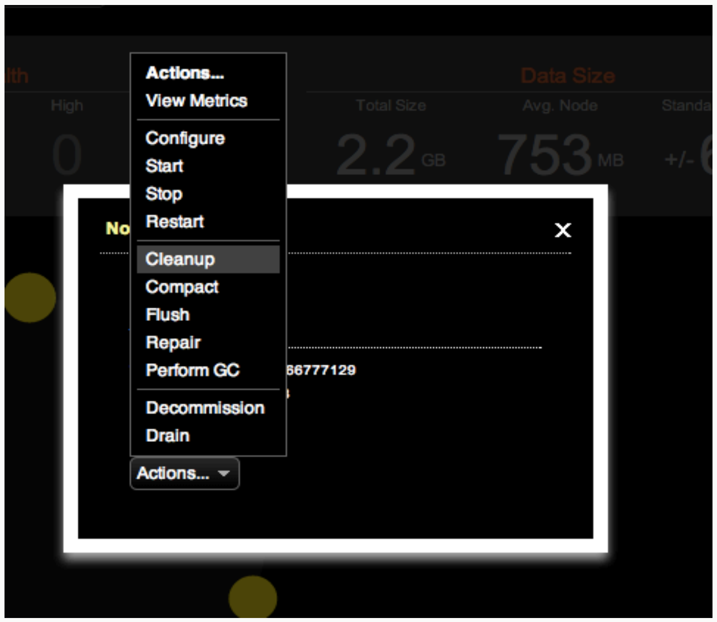
- The *nodetool cleanup* command cleans up all data in a keyspace and table(s) that are specified

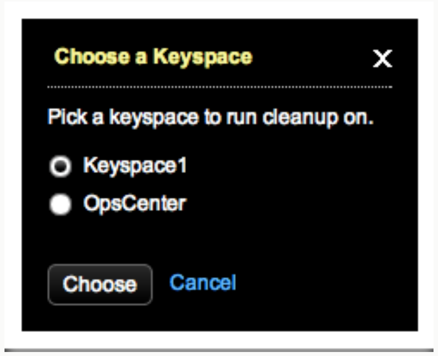
```
bin/nodetool [options] cleanup -- <keyspace> (<table> …)
```

- Same options as *nodetool* command
  - -h [host] | [IP address]
  - -p port
  - -pw password
  - -u username
- *nodetool cleanup* command will clean all keyspaces if no keyspace is specified

How do you execute cleanup using OpsCenter?

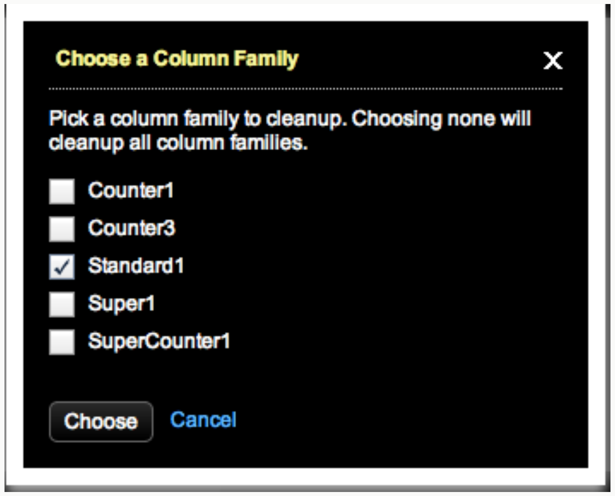
- Cleanup runs as part of the rebalancing process on *OpsCenter*
  - *OpsCenter* will automatically move data to/from nodes in a cluster to even out the partition load balance.
  - Choose cleanup action for a node, then choose keyspaces and tables to cleanup.
  - Once data is moved, cleanup removes data that each node is no longer responsible for.




Choose a Keyspace

Pick a keyspace to run cleanup on.

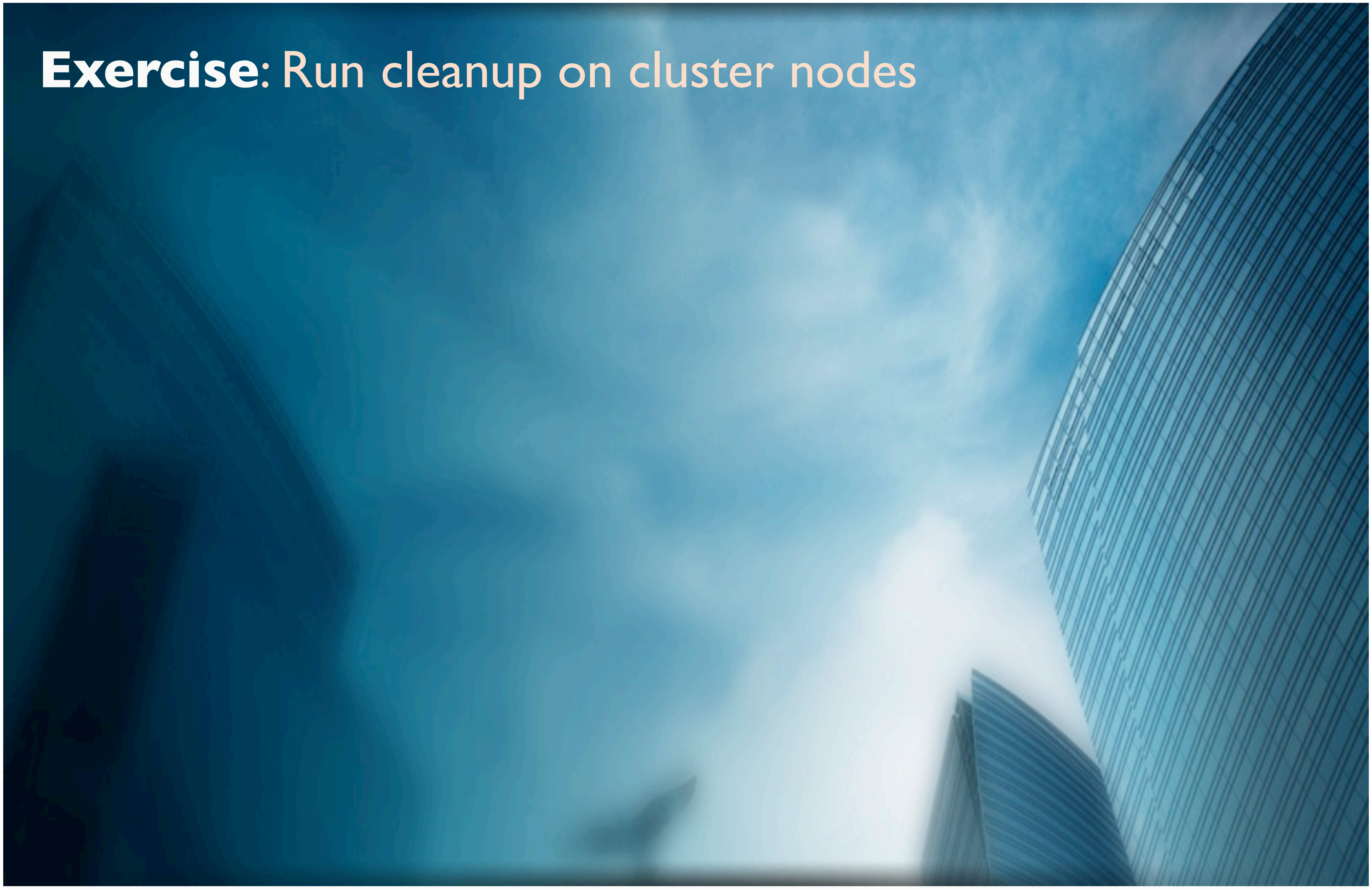
- ○ Keyspace1
- ● OpsCenter


Choose a Column Family

Pick a column family to cleanup. Choosing none will cleanup all column families.

- ☐ Counter1
- ☐ Counter3
- ☑ Standard1
- ☐ Super1
- ☐ SuperCounter1

Exercise: Run cleanup on cluster nodes



Learning Objectives

- Add new nodes into a cluster
- Understand cleanup operations
- **Remove downed nodes**
- Decommission nodes
- Replace downed nodes

What causes a node to go down?

- Network conditions
  - Transient changes in connectivity can cause a node to go down.
- Workload
  - Too much workload for a particular partition
- Hardware failure
  - Disk failure, especially rotation disk, can occur.
- A node not reporting high enough heartbeat rate to the gossip process
  - Gossip uses an accrual detection mechanism to calculate a per-node threshold.
  - The conditions listed above can affect the perceived heartbeat rate.
  - The parameter *phi_convict_threshold* in the *cassandra.yaml* file can control sensitivity to node failure detection.
    - Default value is appropriate for most situations.
    - May need to be changed for cloud environments.

What causes a node to go down?

- *Chaos Monkey* is an interesting tool for simulating failures to test system robustness.
  - Netflix developed this tool to continuously test their production systems during business hours.
  - Failures happen and system design should tolerate it.
  - Randomly terminates one of the nodes in a cluster.

- Information and download: https://github.com/Netflix/SimianArmy/wiki/Chaos-Monkey

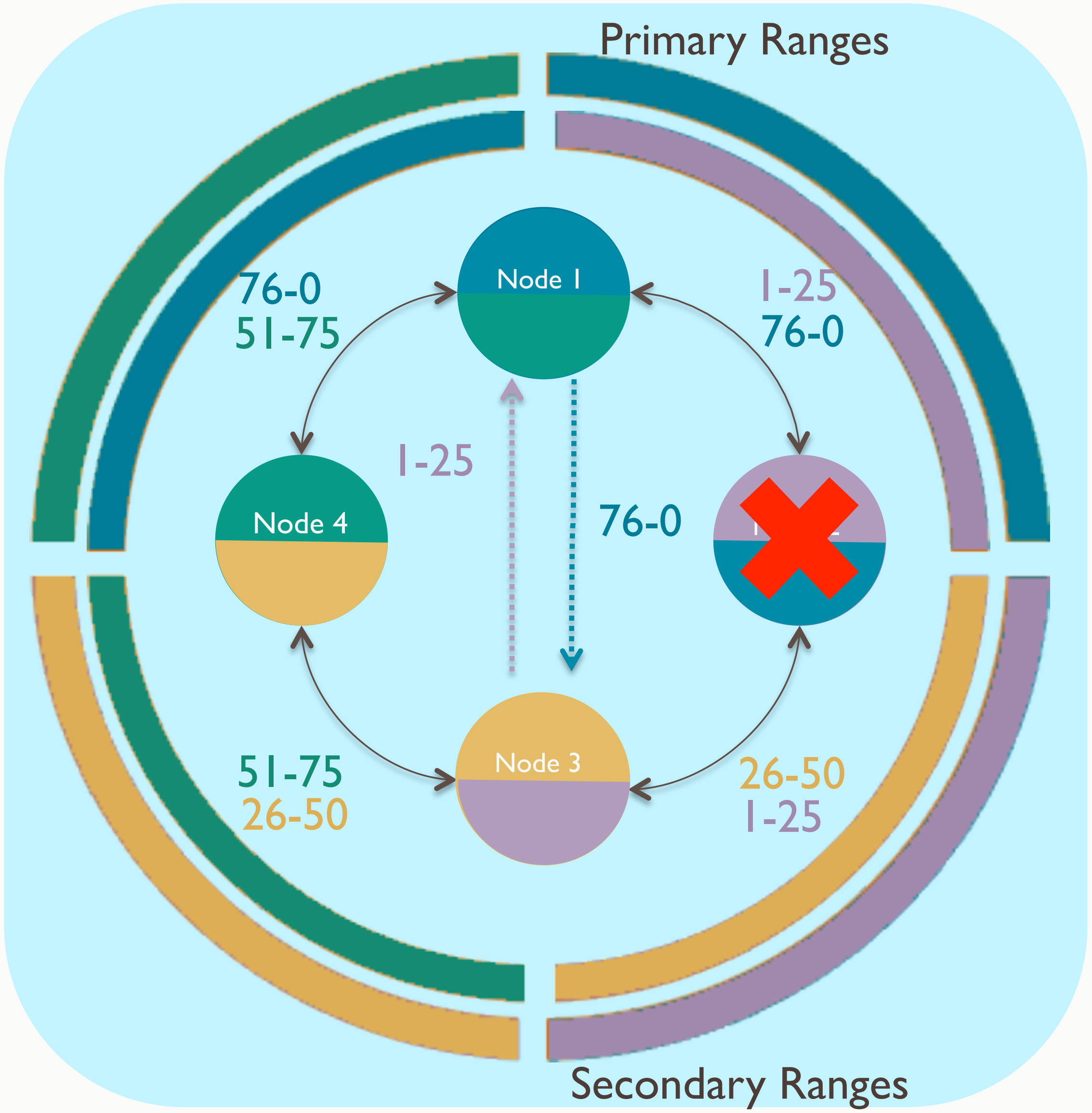
Can a downed node be removed from the cluster?

- Downed node can be removed but does not have to be.
- Downed node is not automatically removed because it may come back online after an outage.
- Outages are often transient.
  - Other nodes periodically try to initiate gossip contact with failed nodes.
  - If a node comes back online, it may have missed writes for its replicas.
  - Other replicas will store the missed writes and provide a hinted handoff to assist the returning node in getting the missed data.

- When should a downed node be removed?
  - If node is not going to recover.
  - Enough time has passed that it would be better to replace node.

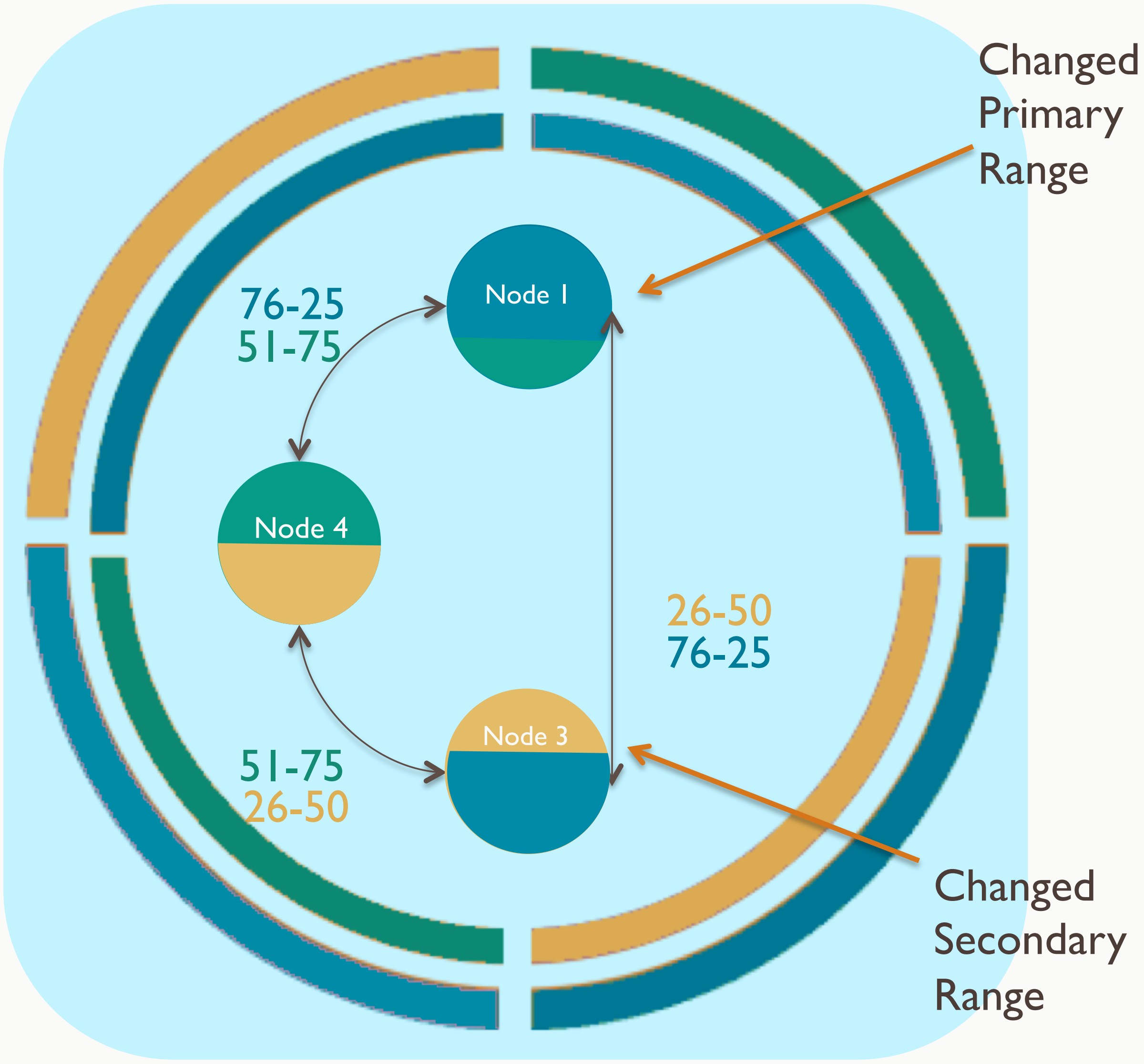
What happens when a node is removed?

- Data will be reassigned to other nodes—another node will gain primary range of removed node.
- Secondary range will also needs duplication.
- The data is streamed from remaining replicas.



What happens when a node is removed?

- Cluster becomes unbalanced—because two nodes have more data.
- Tokens must be rebalanced manually.
- Vnode clusters will not experience unbalancing, as the redistributed partitions are smaller and more evenly spread.



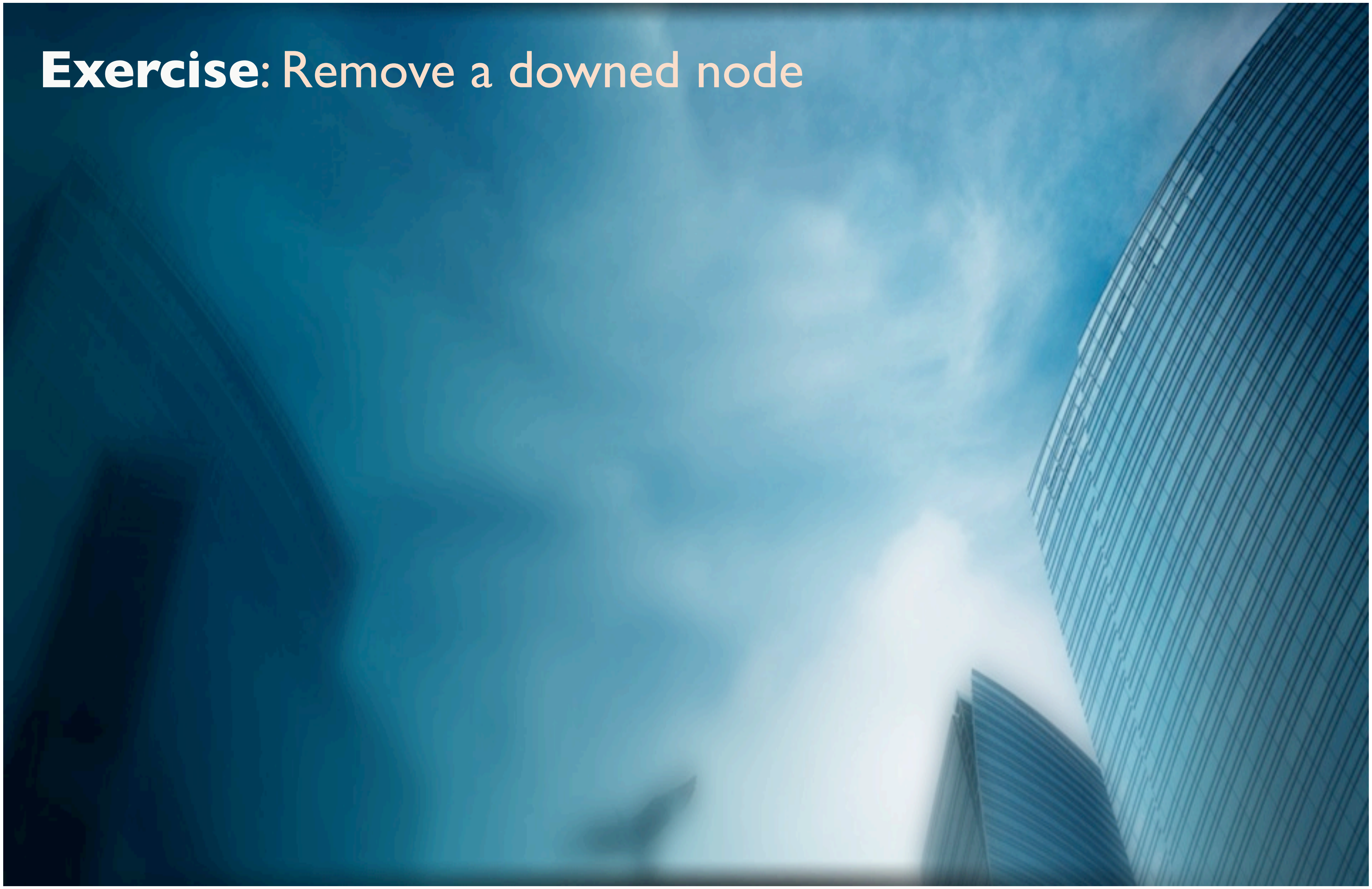
How do you remove a node using *nodetool*?

- The *nodetool removenode* command removes the node that is specified by *host id*

```
bin/nodetool [options] removenode -- <status> | <force> |
[host id]
```

- Same options as *nodetool* command
  - -h [host] | [IP address]
  - -p port
  - -pw password
  - -u username
- Additional arguments can be specified
  - *status* provides status information
  - *force* forces completion of a pending removal
- *nodetool status* can be used to get the *host id*

Exercise: Remove a downed node



Learning Objectives

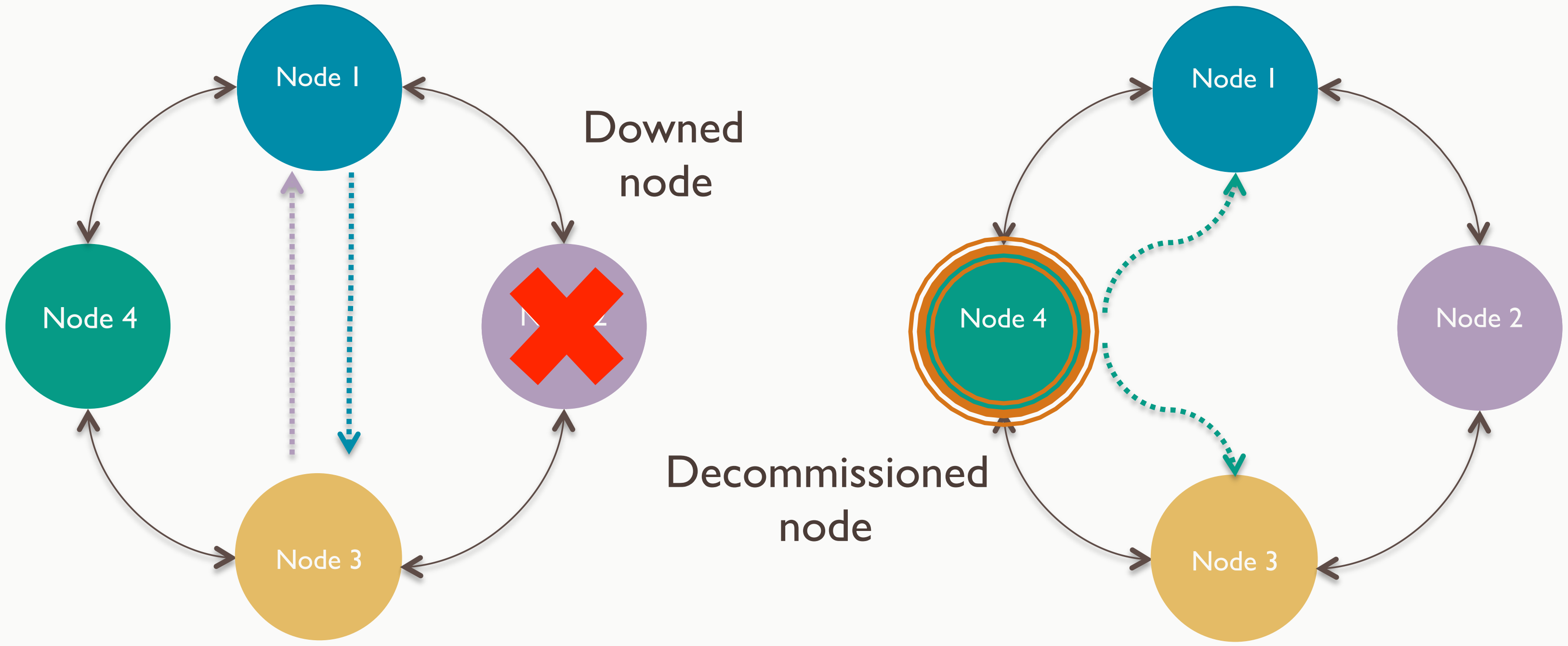
- Add new nodes into a cluster
- Understand cleanup operations
- Remove downed nodes
- **Decommission nodes**
- Replace downed nodes

What if you want to decommission a node and decrease the size of your cluster?

- You might want to remove a live node from a cluster.
- Decommissioning a node will assign the ranges of the old node to other nodes and replicate the appropriate data on the new nodes.
  - The data will be streamed from the decommissioned node to the new nodes.
  - No data is removed automatically from the decommissioned node.
  - If you wish to put the node back into service, you must manually remove the data.
- Another reason for decommissioning might be to swap out an older machine with a newer machine.

What is the difference between a downed node and a decommissioned node?

- Downed node's data will be streamed from other replicas.
- Decommissioned node's data will be streamed from the decommissioning node itself.
- Once data has been moved to other nodes, the process for removing or replacing is similar for both.



What happens when a node is decommissioned?

- Node is marked as "LEAVING" and will stream data to other live nodes.
- Once the streaming is complete, node will quit reporting in *nodetool status*.
- The data directories will still exist – remove these if the node will go back into production.
  - Leaving the data in place will cause confusion if node is brought back into the ring.
  - Commit log does not maintain out-of-date schema information which could lead to tokens that are assigned to more than one node.
    - Bad things happen! Data can disappear!
- Cassandra may still be running – need to manually shut down the process.

How do you decommission a node using *nodetool*?

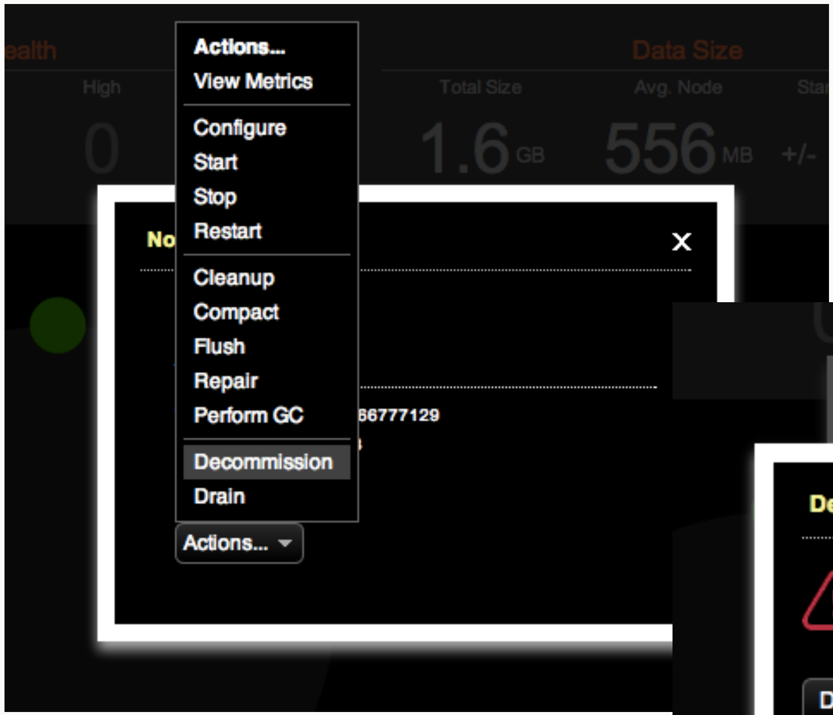
- The *nodetool decommission* command removes node that is specified by the *host id*

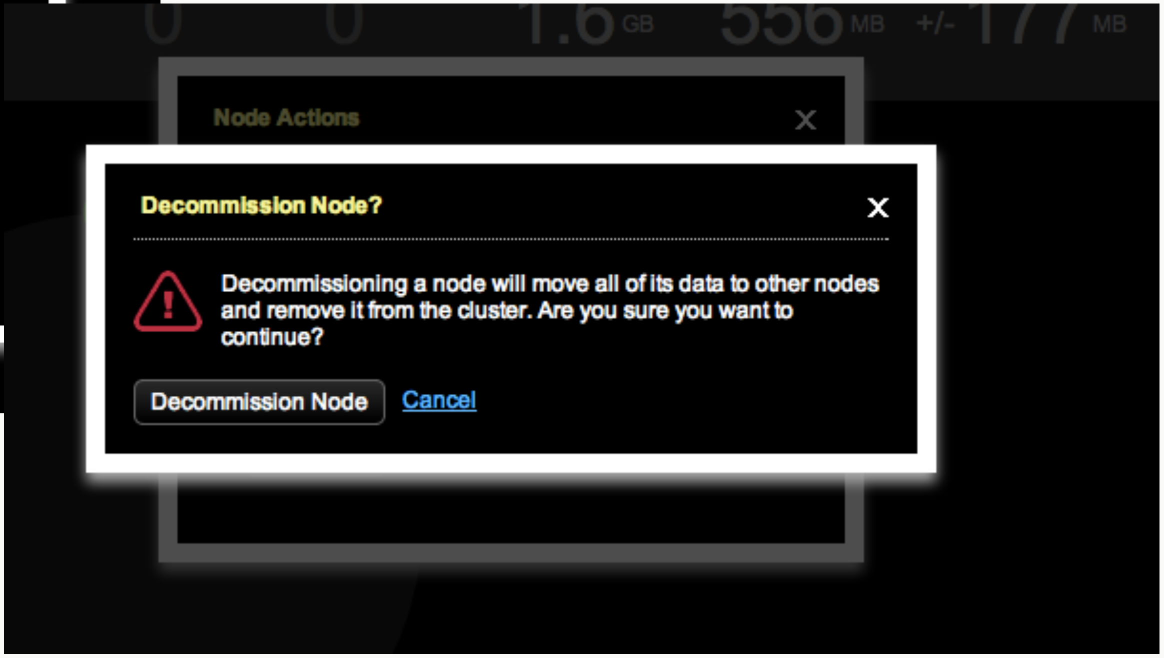
```
bin/nodetool [options] decommission
```

- Same options as *nodetool* command
  - -h [host]|[IP address]
  - -p port
  - -pw password
  - -u username
- *nodetool netstats* can be used to monitor the progress

How do you decommission node using OpsCenter?

- *Decommission* option can be used for any node





Decommissioning a node will move all of its data to other nodes and remove it from the cluster. Are you sure you want to continue?

Learning Objectives

- Add new nodes into a cluster
- Understand cleanup operations
- Remove downed nodes
- Decommission nodes
- **Replace downed nodes**

Can I replace a downed node?

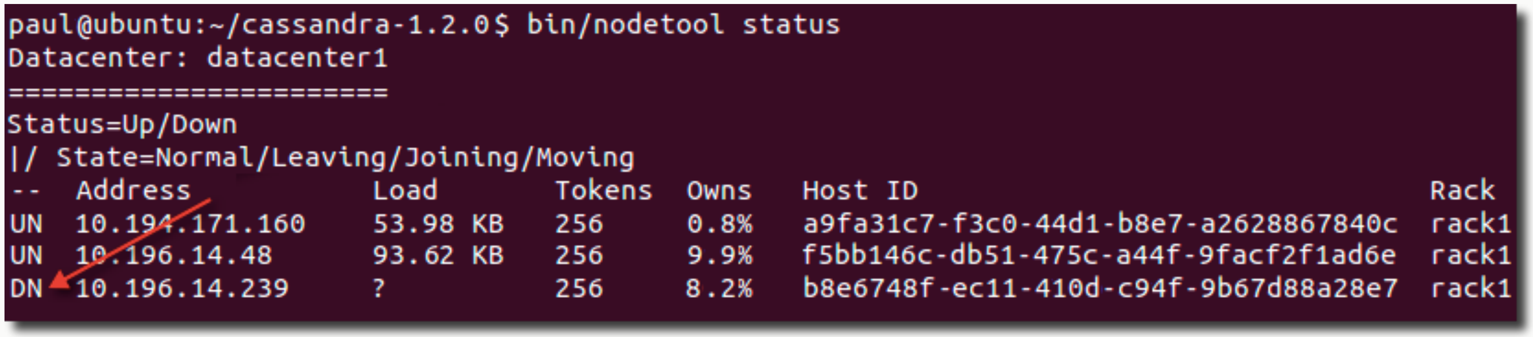
- Yes!
- Benefits of replacing downed nodes
  - You don’t have to move the data twice, as you would if you add a node, then remove a node, and allow the data to be rewritten to the new node with a new token.
  - Backup for a node will work for a replaced node, because same tokens are used to bring replaced node into cluster.
- These are important operations considerations.

Is there a difference if the node is also a seed?

- New seed node IP address will need to be added to the list of seed nodes in the *cassandra.yaml* file for each node.
- Cassandra will not allow a seed node to be automatically bootstrapped if it is listed as a seed node.
- Thus the new seed node will need to have repair run on it to manually bootstrap node.

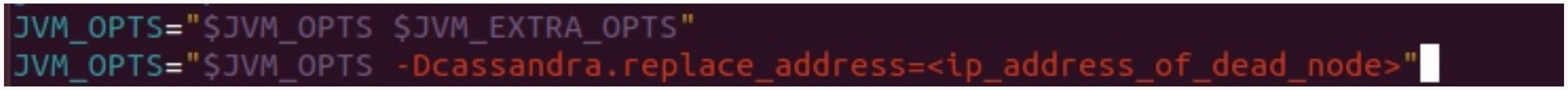
How do you replace a downed node using *nodetool*?

- Find the Address of the down node.



```
paul@ubuntu:~/cassandra-1.2.0$ bin/nodetool status
Datacenter: datacenter1
=======================
Status=Up/Down
|/ State=Normal/Leaving/Joining/Moving
--  Address          Load       Tokens  Owns   Host ID                               Rack
UN  10.194.171.160   53.98 KB   256     0.8%   a9fa31c7-f3c0-44d1-b8e7-a2628867840c  rack1
UN  10.196.14.48     93.62 KB   256     9.9%   f5bb146c-db51-475c-a44f-9facf2f1ad6e  rack1
DN  10.196.14.239    ?          256     8.2%   b8e6748f-ec11-410d-c94f-9b67d88a28e7  rack1
```

- Bootstrap a new node in, using the IP address of the dead node as the replace_address value in the JVM option in the *cassandra-env.sh* file.



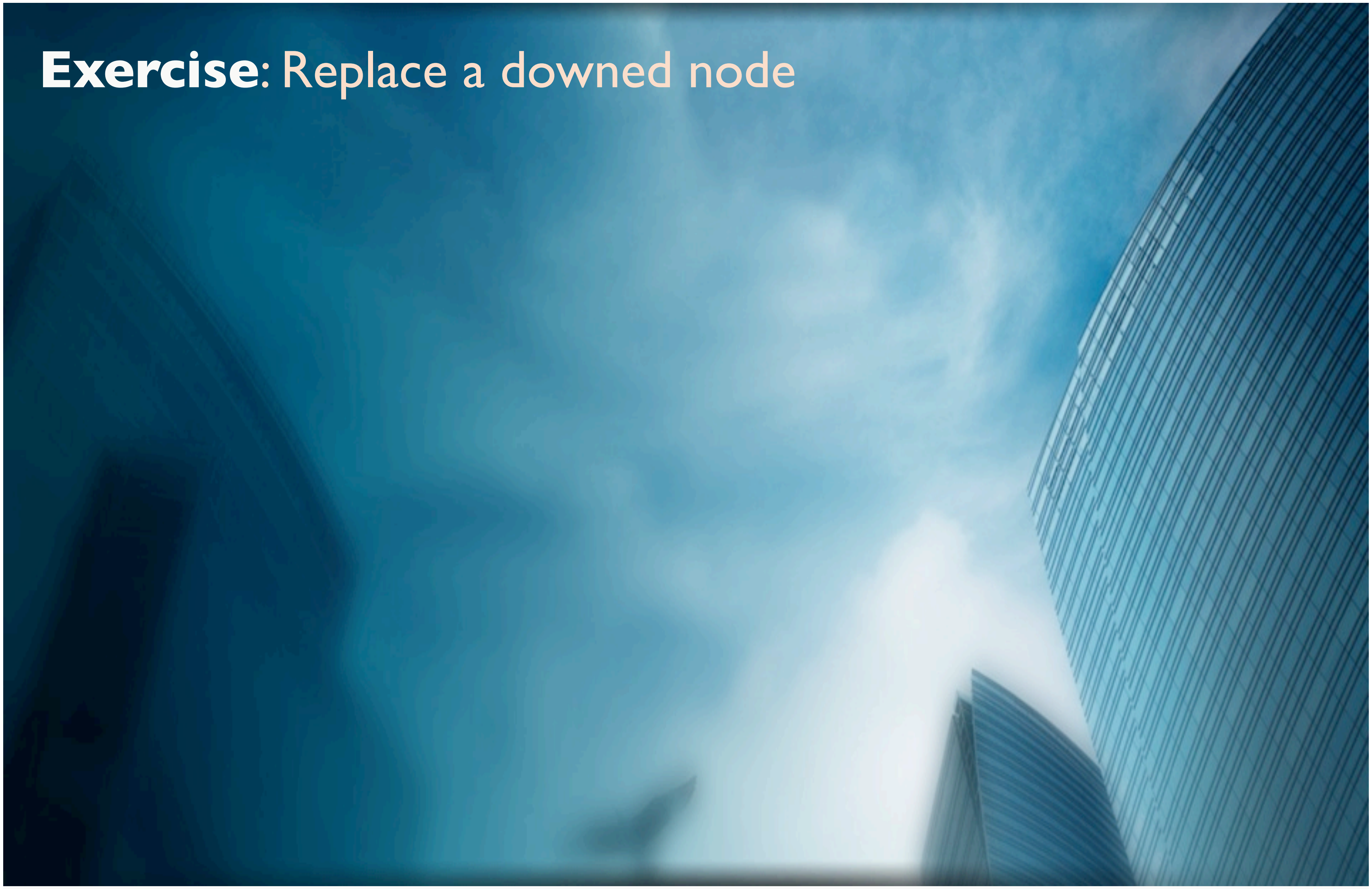
```
JVM_OPTS="$JVM_OPTS $JVM_EXTRA_OPTS"
JVM_OPTS="$JVM_OPTS -Dcassandra.replace_address=<ip_address_of_dead_node>"
```

- Use *nodetool removenode* to remove the dead node.
  - Use the *force* option if necessary.
- Monitor using *nodetool netstats*.

How do you replace a downed seed node using *nodetool*?

- Add a new node making the necessary changes to the *cassandra.yaml* file.
- One difference – specify a seed node in the *cassandra.yaml* file.
  - This is a good illustration of why you want more than one seed node.
  - Make sure in the new node to specify an active seed node.
- Start Cassandra on the new seed node.
- Run *nodetool repair* on the new seed node to manually bootstrap.
- Remove the old seed node using *nodetool removenode* with the Host ID of the downed node.
- Good idea to run *nodetool cleanup* on each previously existing node to remove keys that have been moved to the new node.

Exercise: Replace a downed node



Summary

- Bootstrapping in Cassandra refers to process that a new node goes through to join a cluster.
- Cleanup is the process of removing data from a node when that data has been transferred to a new node.
- Downed nodes are those that quit responding to the cluster.
  - Partial failure, or temporary failure is common, so a threshold is used to manage whether or not a heartbeat is considered permanently gone.
- Removing a node uses replica data to move a downed node's data to other nodes.
- Decommissioning a node is similar to removing a node, but the node being decommissioned can first write its data to other nodes.
- Replacing a node has advantages—replacement node has the same token as the original node being replaced
- Replacing a seed node involves a few extra steps

Review Questions

- What has to happen before a node can be bootstrapped?
- What is the main purpose of bootstrapping in Cassandra?
- Why is cleanup necessary?
- What is the difference between removing a node and decommissioning a node?
- What is the procedure for replacing a downed node?
- What are the differences between replacing a seed node vs. a regular node?