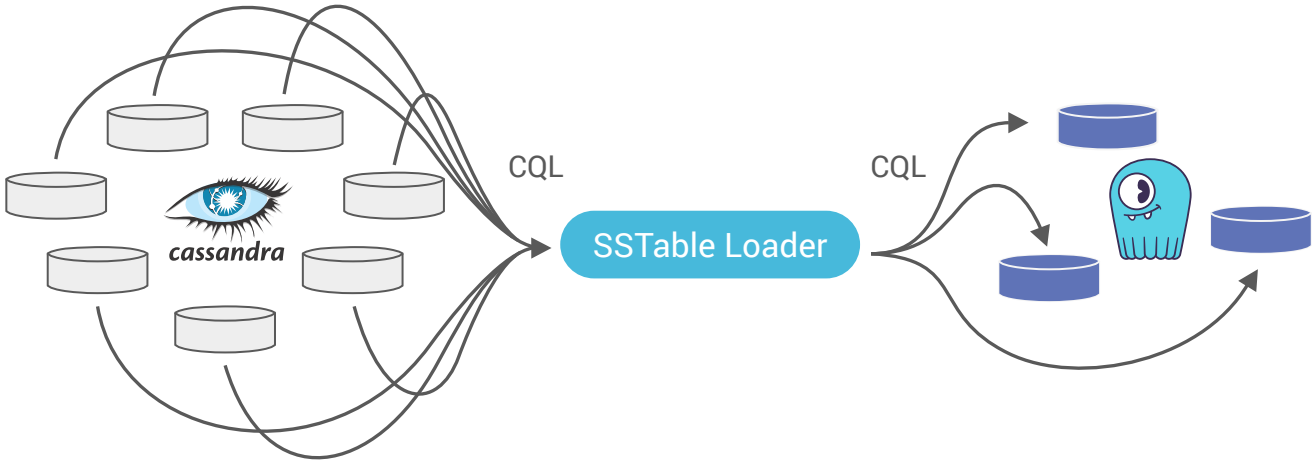
MIGRATION GUIDE:
Apache Cassandra to Scylla

Scylla is a drop-in replacement for Apache Cassandra. Transitioning from Cassandra to Scylla is very straight forward. This guide will cover the multi-step process of migrating the data and verifying its consistency. We'll start by listing the high-level steps, as follow:

1. Creating the same schema from Apache Cassandra in Scylla (there can be some variation)
2. Configuring your application/s to perform dual writes (still reading only from Apache Cassandra)
3. Taking a snapshot of all to-be-migrated data from Apache Cassandra
4. Loading the sstable files to Scylla using the Scylla sstableloader tool + data validation
5. Verification period: Dual writes and reads, Scylla serves reads. Logging mismatches, until minimal data mismatch threshold is reached
6. Apache Cassandra end of life: Scylla only for reads and writes procedure

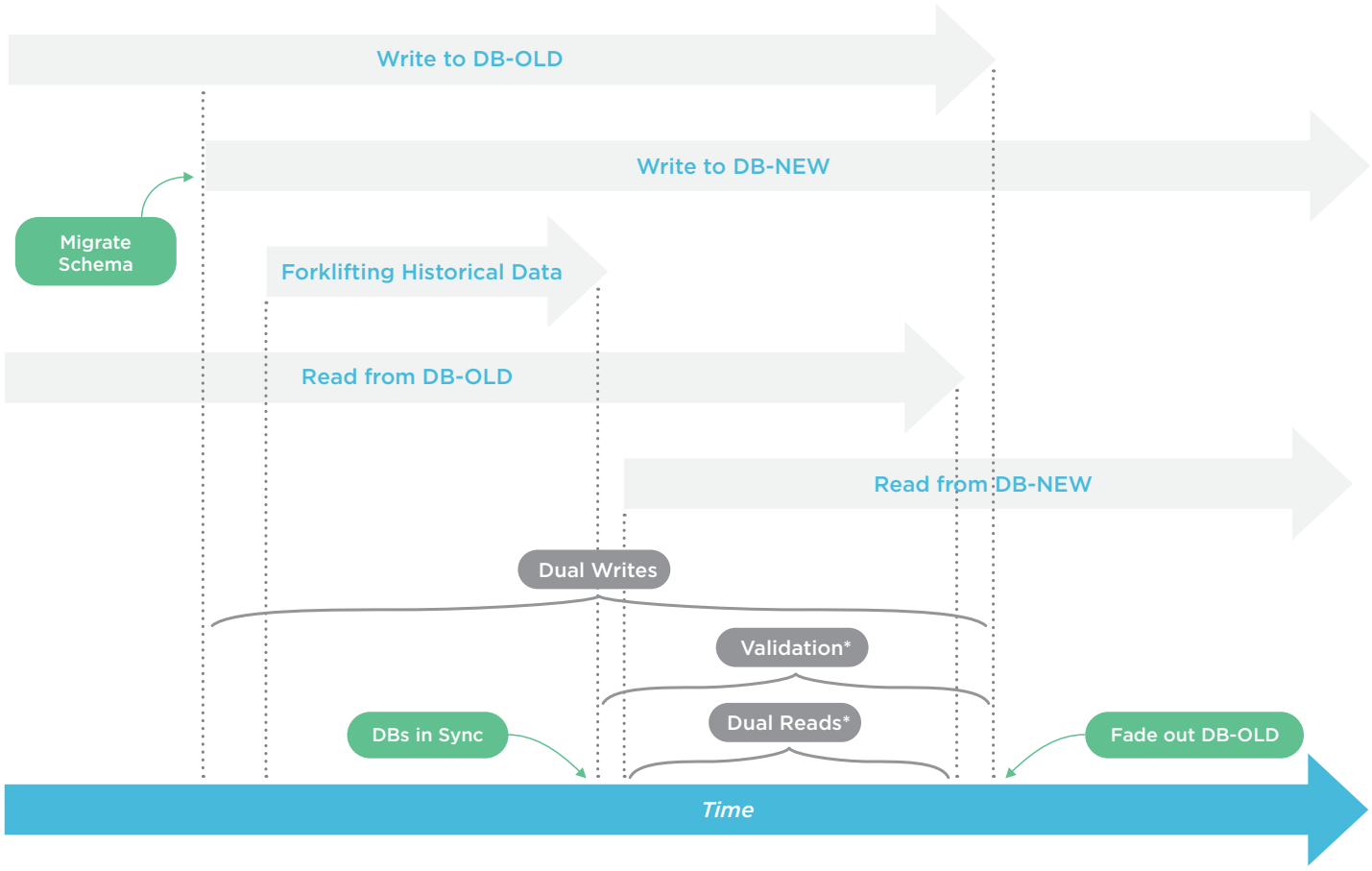
Note: Steps 2 and 5 are required for live migration only (meaning with on-going traffic and no downtime).

Dual Writes: Application logic is updated to write to both DBs



Forklifting: Migrate historical data from Apache Cassandra SSTables to Scylla

Dual Reads: Ongoing validation of data sync between the two DBs



Live Migration: Migrating from DB-OLD to DB-NEW timeline

PROCEDURE

1 Create manually / Migrate your schema (keyspaces, tables and user defined type, if used) on / to your Scylla cluster. When migrating from Apache Cassandra 3.x some schema updates are required (for limitations and known issues, please refer to docs.scylladb.com).

- Export schema from Apache Cassandra:

```
cqlsh [IP] "-e DESC SCHEMA" > orig_schema.cql
```

- Import schema to Scylla:

```
cqlsh [IP] --file 'adjusted_schema.cql'
```

2 If you wish to perform the migration process without any downtime, please configure your application/s to perform dual writes to both data stores, Apache Cassandra and Scylla (see below code snippet for dual writes). Before doing that, and as a general guidance, make sure to use client generated timestamp (writetime). If you do not, the data on Scylla and Apache Cassandra can be considered different, while it is the same.

Note: Your application/s should continue reading and writing from Apache Cassandra until the entire migration process is completed, data integrity is validated and the dual writes and reads verification has been performed to your satisfaction.

Dual writes and client-generated timestamp Python code snippet

```
# put both writes (cluster 1 and cluster 2) into a list
    writes = []

    #insert 1st statement into db1 session, table 1
    writes.append(db1.execute_async(insert_statement_prepared[0], values))
    #insert 2nd statement into db2 session, table 2
    writes.append(db2.execute_async(insert_statement_prepared[1], values))

    # loop over futures and output success/fail
    results = []
    for i in range(0,len(writes)):
        try:
            row = writes[i].result()
            results.append(1)
        except Exception:
            results.append(0)
            #log exception if you like
            #logging.exception('Failed write: %s', ('Cluster 1' if (i==0) else 'Cluster 2'))

    results.append(values)
    log(results)
```

```
    #did we have failures?
    if (results[0]==0):
        #do something, like re-write to cluster 1
        log('Write to cluster 1 failed')
    if (results[1]==0):
        #do something, like re-write to cluster 2
        log('Write to cluster 2 failed')

for x in range(0,RANDOM_WRITES):
    #explicitly set a writetime in microseconds
    values = [ random.randrange(0,1000) , str(uuid.uuid4()) , int(time.time()*1000000) ]

    execute( values )
```

See the full code example here

3 On each Apache Cassandra node, take a snapshot for every keyspace using nodetool snapshot command. This will flush all sstables to disk and generate a snapshots folder with an epoch timestamp for each underlying table in that keyspace.

Folder path post snapshot:

```
/var/lib/cassandra/data/keyspace/table-[uuid]/snapshots/[epoch_timestamp]/
```

4 We strongly advise against running the sstableloader tool directly on the Scylla cluster, as it will consume resources from Scylla, instead you should run the sstableloader from intermediate node/s. To do that, you need to install `scylla-tools-core` package (it includes the sstableloader tool).

You need to make sure you have connectivity to both the Apache Cassandra and Scylla clusters. There are two ways to do this, both of which require having a file system in place (RAID is optional):

- **Option 1 (recommended):** Copy the sstable files from the Apache Cassandra cluster to a local folder on the intermediate node.
- **Option 2:** NFS mount point on the intermediate node to the sstable files located in the Apache Cassandra nodes.
  - NFS mount on CentOS
  - NFS mount on Ubuntu

1. After installing the relevant pkgs (detailed in the links), edit `/etc/exports` file on each Apache Cassandra node and add the following in a single line:

```
[Full path to snapshot 'epoch' folder] [Scylla_IP]
(rw,sync,no_root_squash,no_subtree_check)
```

2. Restart NFS server `sudo systemctl restart nfs-kernel-server`

3. Create a new folder on one of the Scylla nodes and use it as a mount point to the Apache Cassandra node

Example:

```
sudo mount [Cassandra_IP]:[Full path to snapshots 'epoch' folder] /[ks]/[table]
```

Note: both the local folder, or the NFS mount point paths must end with `/[ks]/[table]` format, used by the sstableloader for parsing purposes (see `sstableloader help` for more details).

5 If you cannot use intermediate node/s (see the previous step), then you have two options:

- Option 1: Copy the sstable files to a local folder on one of your Scylla cluster nodes. Preferably on a disk or disk-array which is not part of the Scylla cluster RAID, yet still accessible for the sstableloader tool.

  Note: Copying it to the Scylla RAID, will require sufficient disk space (Apache Cassandra sstable snapshots size x2 < 50% of Scylla node capacity), in order to contain both the copied sstables files and the entire data migrated to Scylla (keyspace RF should also be taken into account).

- Option 2: NFS mount point on Scylla nodes to the sstable files located in the Apache Cassandra nodes (see NFS mount instructions in the previous step). This saves the additional disk space needed for the 1st option.

  Note: Both the local folder or the NFS mount point paths must end with `/[ks]/[table]` format, used by the sstableloader for parsing purposes (see `sstableloader help` for more details).

6 Use Scylla sstableloader tool (**NOT** the Apache Cassandra one that has the same name) to load the sstables. Running without any parameters will present the list of options and usage. Most important are the sstables directory and the Scylla node IP.

Note: We recommend using the `-x` flag for prepared statements, as it boosts the performance (up to an x5 factor).

Examples:

```
sstableloader -x -d [Scylla IP] .../[ks]/[table]
```

```
sstableloader -x -d [scylla IP] .../[mount point] (in /[ks]/[table] format)
```

7 We recommend running several sstableloaders in parallel and utilizing all Scylla nodes as targets for sstable loading. Start with one keyspace and its underlying sstable files from all Apache Cassandra nodes. After completion, continue to the next keyspace and so on.

Note: Limit the sstableloader speed by using the throttling `-t` parameter, considering your physical HW, live traffic load and network utilization (see sstableloader help for more details).

8 Once you completed loading the sstable files from all keyspaces, you can use `cqlsh` or any other tool to validate the data migrated successfully. We strongly recommend configuring your application to perform both writes and reads to/from both data stores. Apache Cassandra (as is, up to this point) and Scylla (now as primary) for a verification period. Keep track of the number of requests for which the data in both these data stores is mismatched.

9 **Apache Cassandra end of life:** Once you are confident in your Scylla cluster, you can flip the flag in your application/s, stop writes and reads against the Cassandra cluster, and make Scylla your sole target/source.

FAILURE HANDLING

What should I do if sstableloader fails?

Each loading job is per keyspace/table_name, that means in any case of failure, you need to repeat the loading job. As you are loading the same data (partially loaded before the failure), compactions will take care of any duplication.

What should I do if an Apache Cassandra node fails?

If the node that failed was a node you were loading sstables from, then the sstableloader will also fail. If you were using RF>1 then the data exists on other node/s. Hence you can continue with the sstable loading from all the other Cassandra nodes. Once completed, all your data should be on Scylla.

What should I do if a Scylla node fails?

If the node that failed was a node you were loading sstables to, then the sstableloader will also fail. Restart the loading job and use a different Scylla node as your target.

How to rollback and start from scratch?

1. Stop the dual writes to Scylla
2. Stop Scylla service `sudo systemctl stop scylla-server`
3. Use `cqlsh` to perform `truncate` on all data already loaded to Scylla
4. Start the dual writes again to Scylla
5. Take a new snapshot of all Cassandra nodes

For more information on migrating from Cassandra to Scylla, please visit docs.scylladb.com.

ABOUT US

Scylla is a drop-in Apache Cassandra replacement that powers your applications with ultra-low latency and extreme throughput.

Leveraging the best from Apache Cassandra in high availability, fault tolerance, and its rich ecosystem, Scylla offers developers a dramatically higher-performing and resource effective NoSQL database to power modern and demanding applications.

GET STARTED TODAY

Download Scylla Open Source at scylladb.com/download or contact us at: sales@scylladb.com

SCYLLADB.COM

SCYLLA.

United States Headquarters
1900 Embarcadero Rd
Palo Alto, CA 94303 U.S.A.
Phone: +1 (650) 332-9582
Email: info@scylladb.com

Israel Headquarters
11 Galgalei Haplada
Herzelia, Israel