New Replica Types

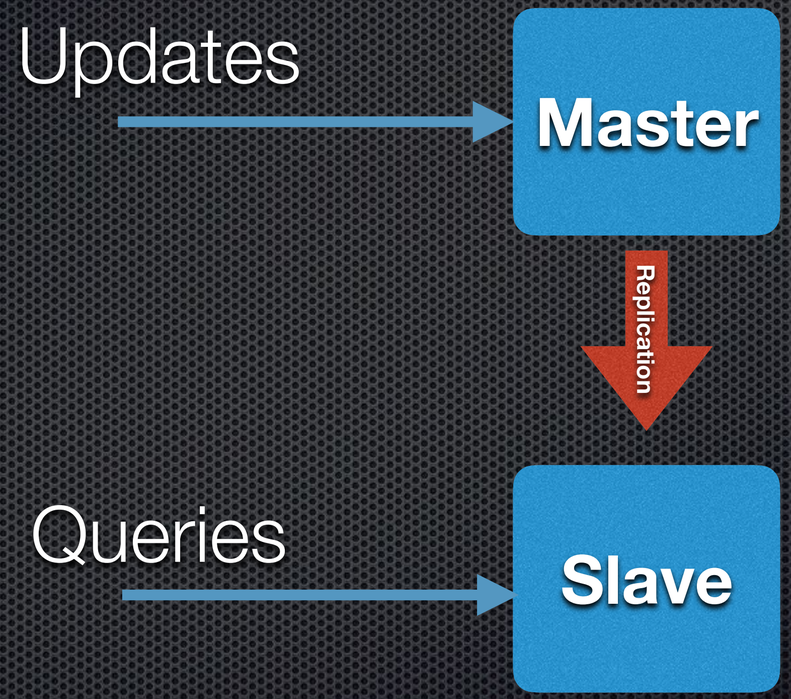
Tomás Fernández Löbbe (tflobbe@…)

Agenda

- Scaling Solr pre 4.0
- SolrCloud
- Why Replica Types?
- Replica Types Added
- Master/Slave in SolrCloud
- How to use Replica Types
- TODOs and future work

Scaling Solr pre 4.0

Scaling Solr pre 4.0

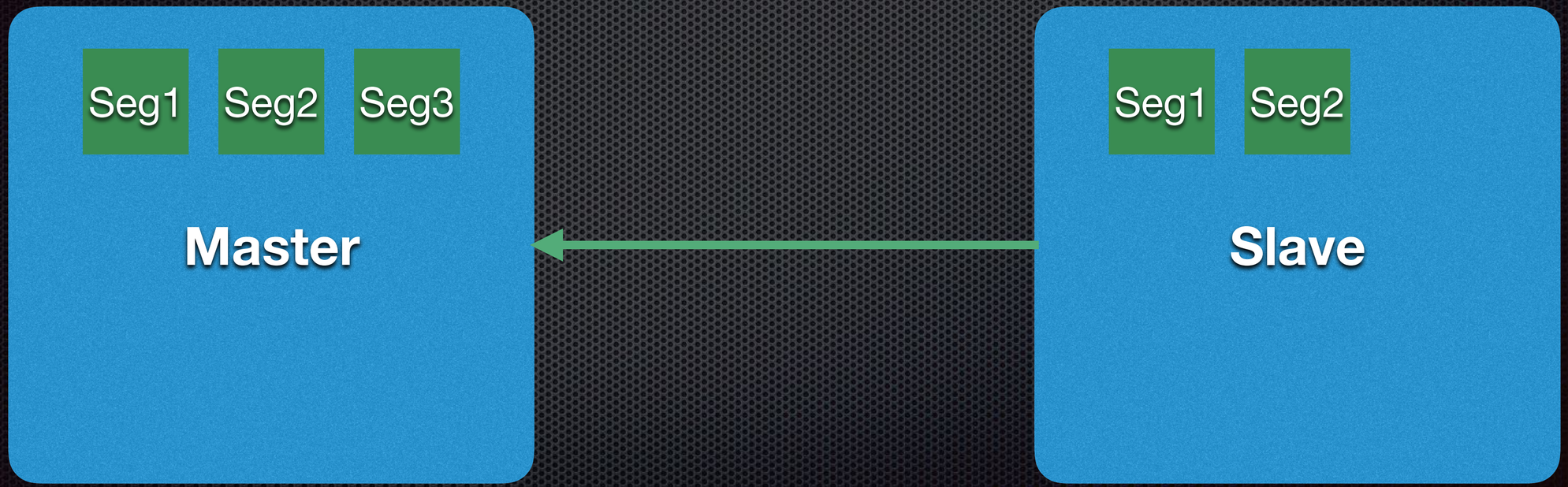


Scaling Solr pre 4.0

- Solr is built on top of Lucene
- Lucene writes segments to disk as new documents are added
- Lucene writes once. **Files do not change once they are flushed to disk**
- A background thread merges segments

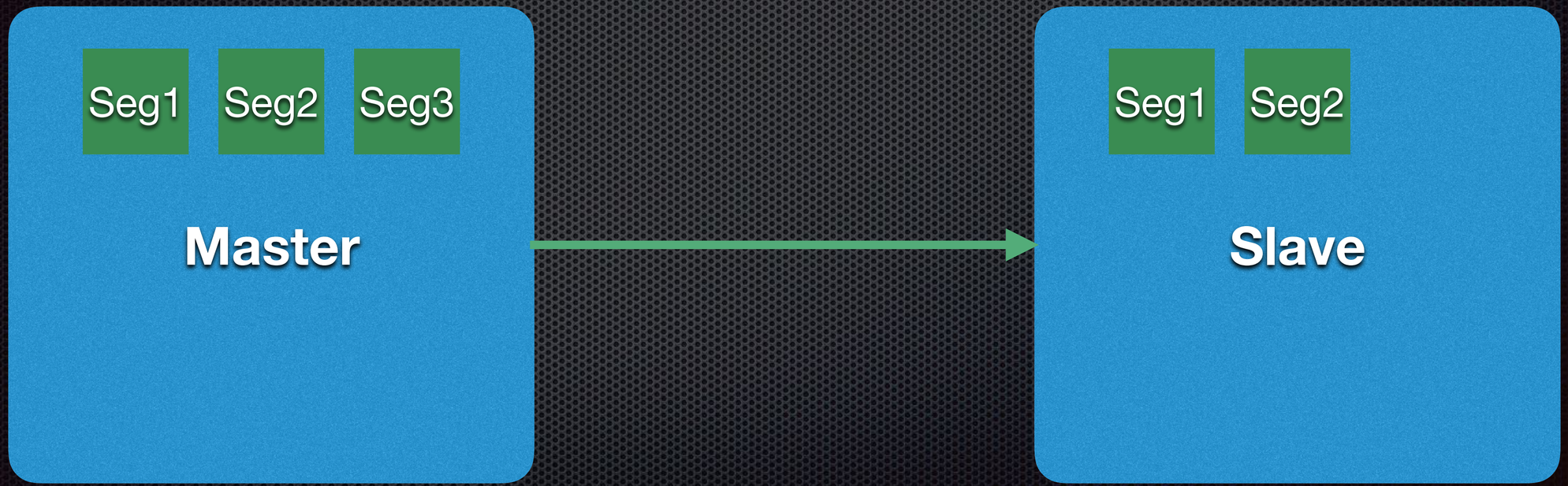
Scaling Solr pre 4.0

- Solr segment file replication works by incrementally downloading new segments from master server
- Not NRT (does not support “softCommits”)



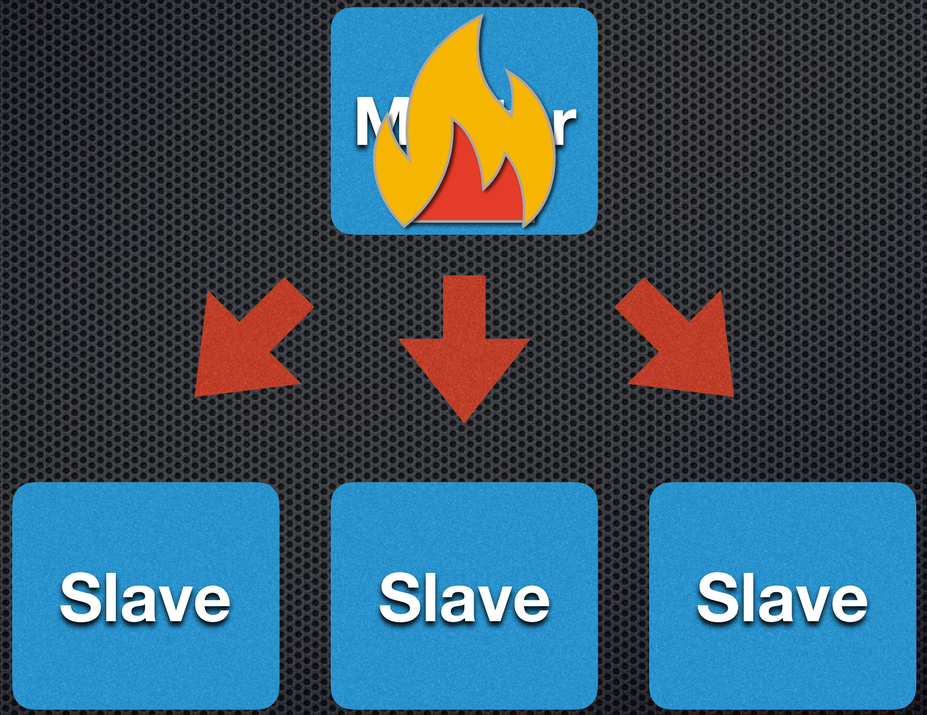
Scaling Solr pre 4.0

- Solr segment file replication works by incrementally downloading new segments from master server
- Not NRT (does not support “softCommits”)



Scaling Solr pre 4.0

- If master server goes down, writes to the shard will fail. Search would still work



SolrCloud

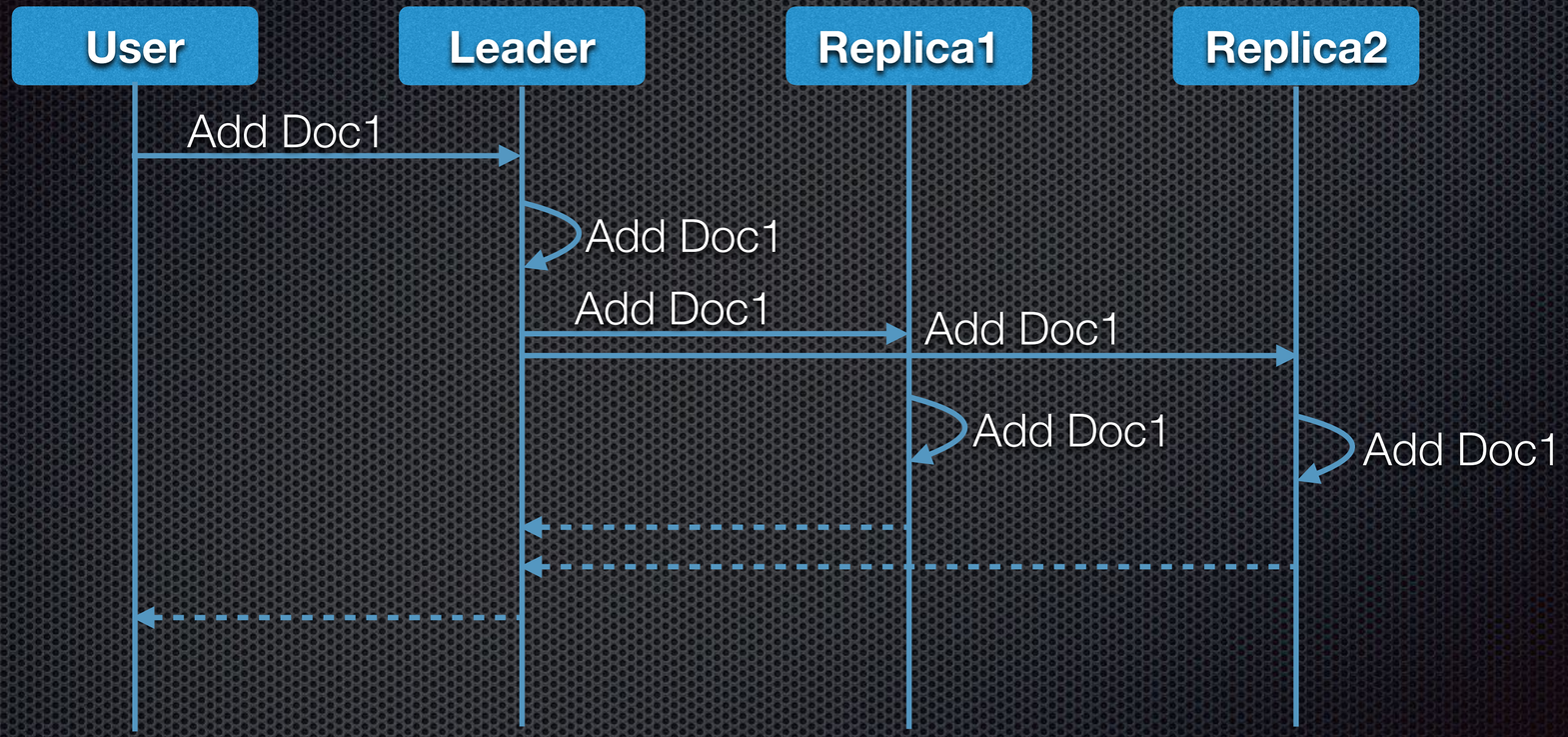
SolrCloud

- The set of features and capabilities of Solr to support:
  - Distributed indexing and searching
  - Automatic load balancing for queries
  - Central configuration
  - Node discovery

Scaling with SolrCloud

- One replica per shard is elected to be leader
- Leader versions the update, applies it locally and forwards it to the replicas
- Every update is sent to all replicas of a shard
- If a replica fails a response, it needs to recover

Scaling with SolrCloud



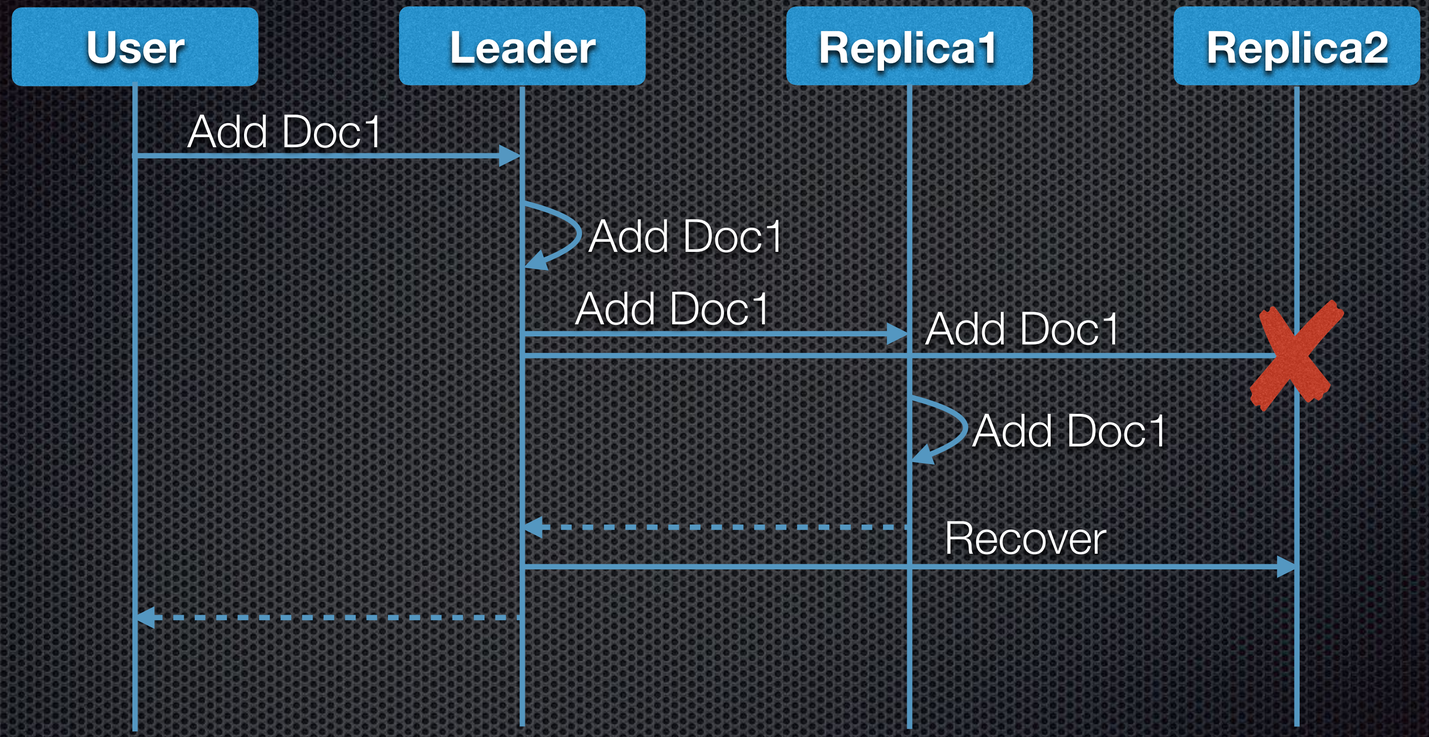
Scaling with SolrCloud

- In addition to the Lucene index, each replica keeps a transaction log
- Contains at least the updates made since the last commit.
- Required in the recovery process (in addition to RealTime Gets)

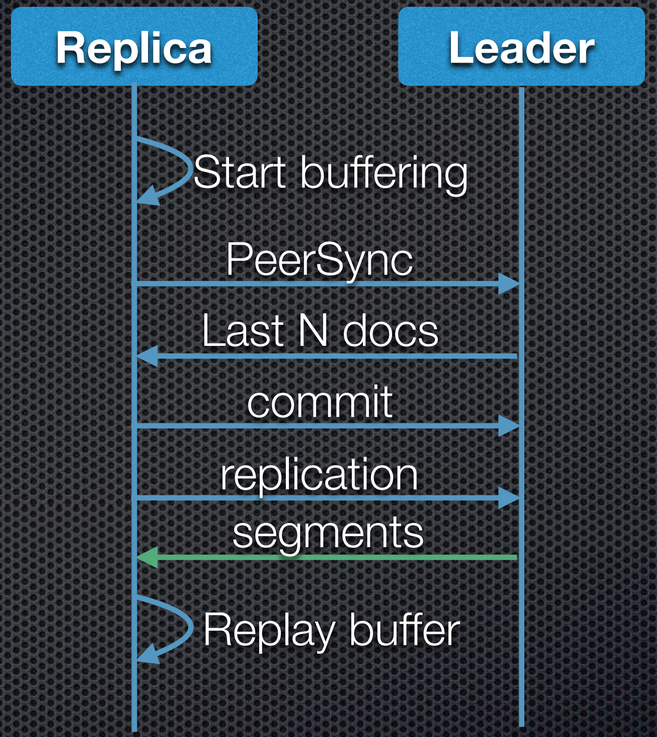
Scaling with SolrCloud

- Replicas that miss updates (or new replicas added to the shard) need to recover from the leader
- While on RECOVERY state, replicas don’t serve query traffic

Scaling with SolrCloud



Scaling with SolrCloud



Why replica types?

Why replica types?

- In Master/Slave architecture, updates and queries are sent to different nodes, so the resources used by one process don’t affect the other process.
  - An expensive query doesn’t affect update throughput
  - An expensive document update/segment merge doesn’t affect query throughput

This was not possible in SolrCloud mode

Why replica types?

- Some use cases are OK with serving slightly out of date data

Why replica types?

- Leader Initiated Recovery can become a problem

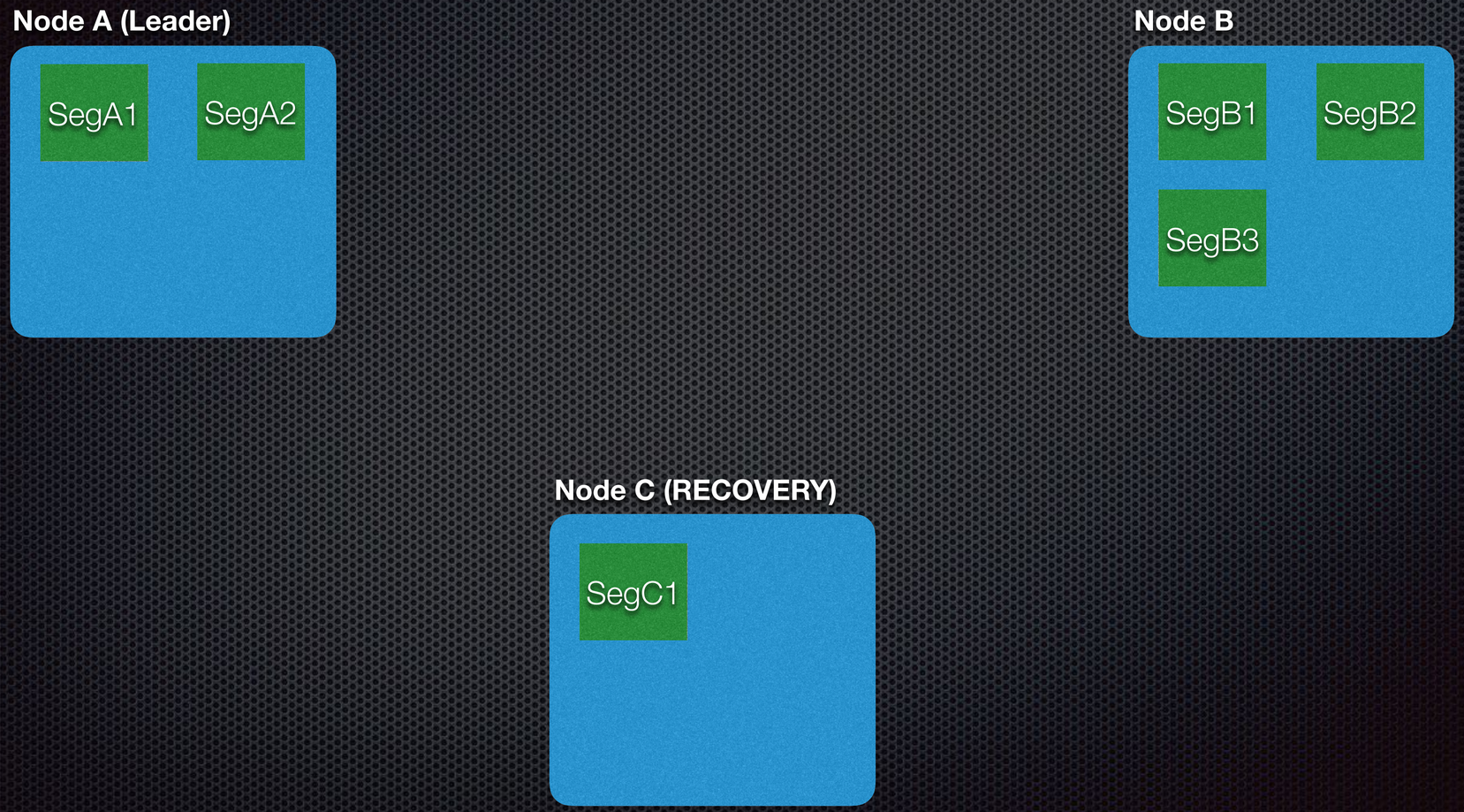
Why replica types?

- In clusters with many replicas per shard, making every replica index, commit and merge can be wasteful
  - On a 3 node shard, each update is sent to 3 replicas and indexed 3 times
  - On a 50 node shard, each update is sent to 50 replicas and indexed 50 times

Why replica types?

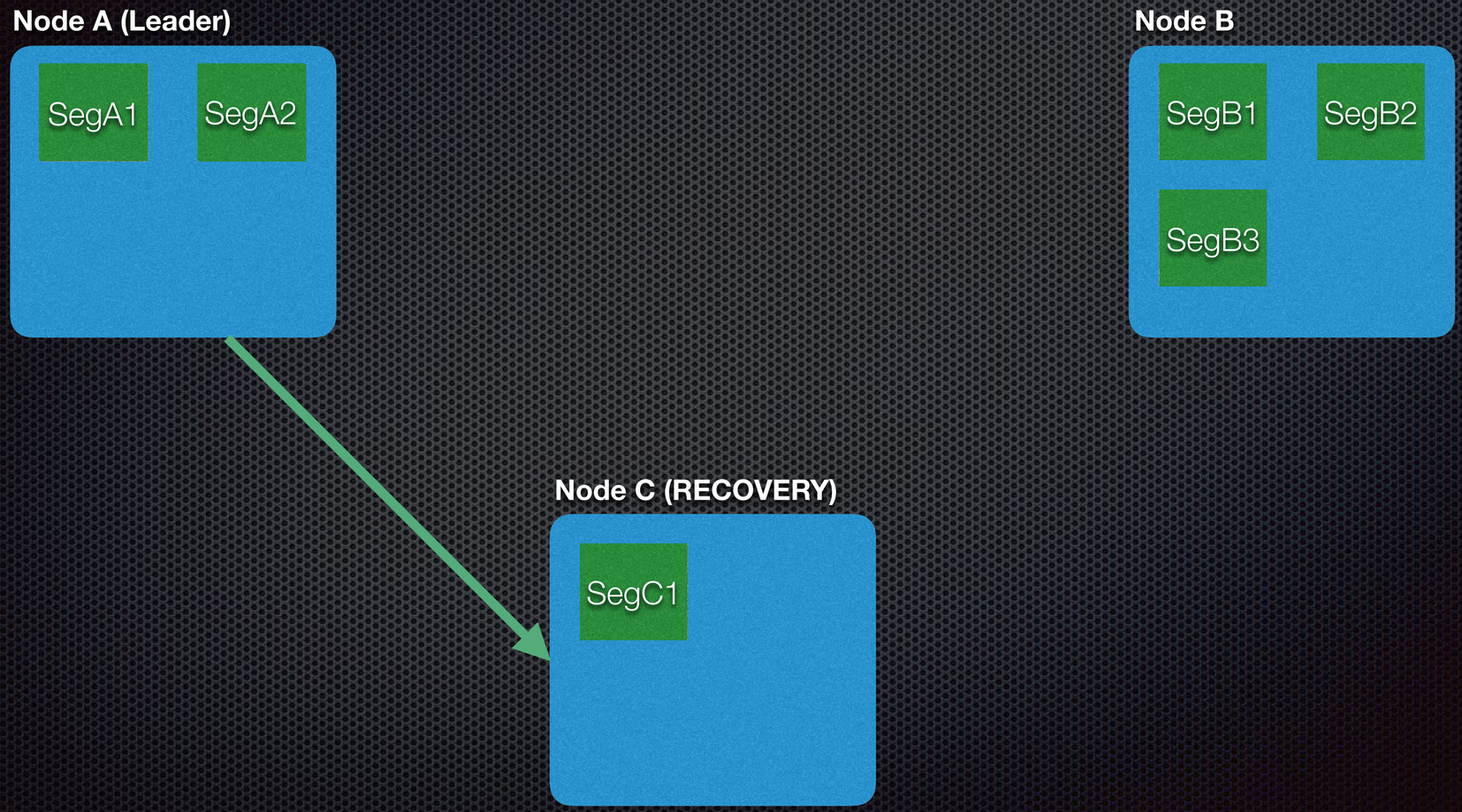
- In high indexing throughput the transaction log sync process has little chance of succeeding
- In high indexing throughput the number of segment files to copy from the leader grows
- If there was a leader change, a full index replication may be needed

Why replica types?



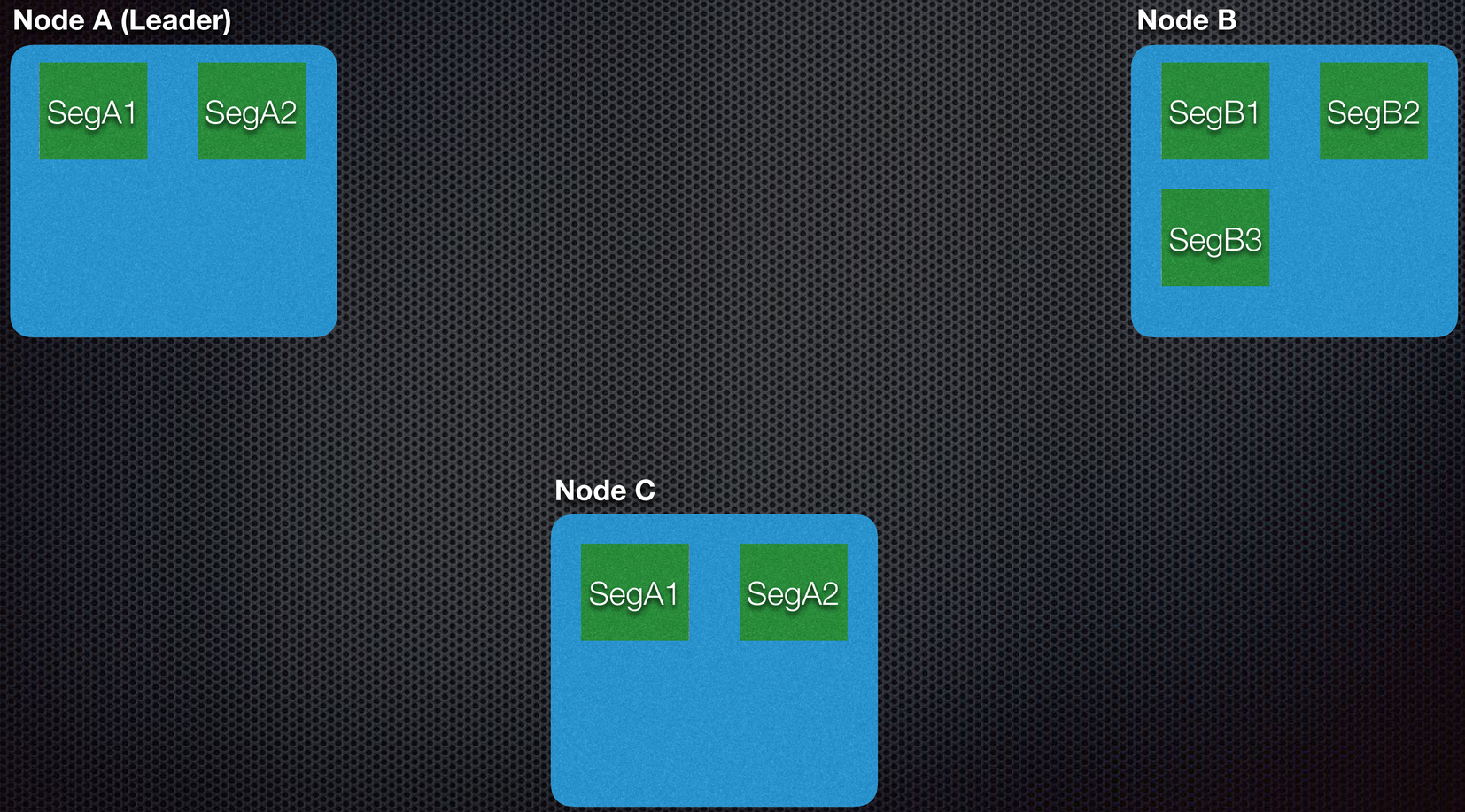
Why replica types?

Full Index Recovery issue in SolrCloud



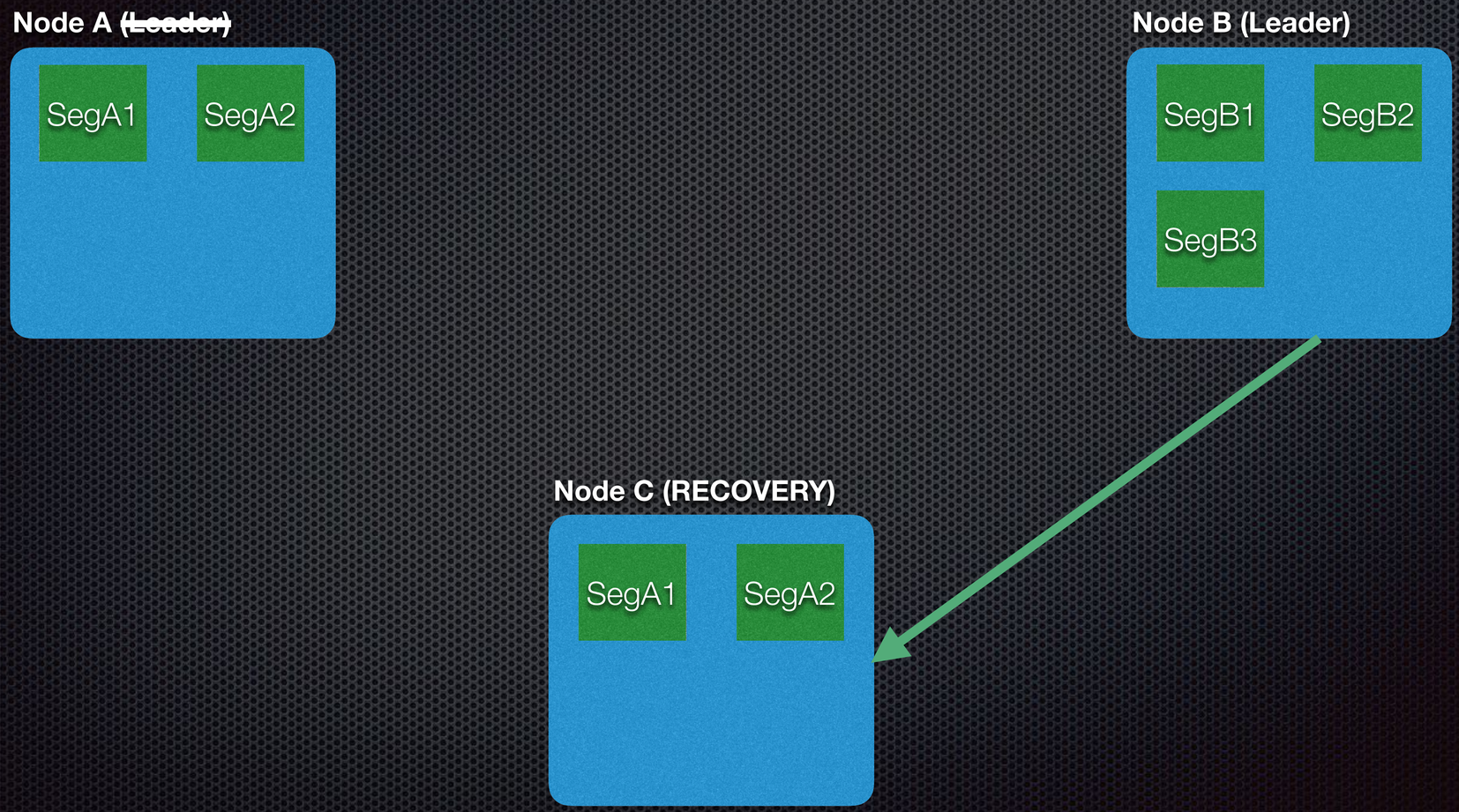
Why replica types?

Full Index Recovery issue in SolrCloud



Why replica types?

Full Index Recovery issue in SolrCloud



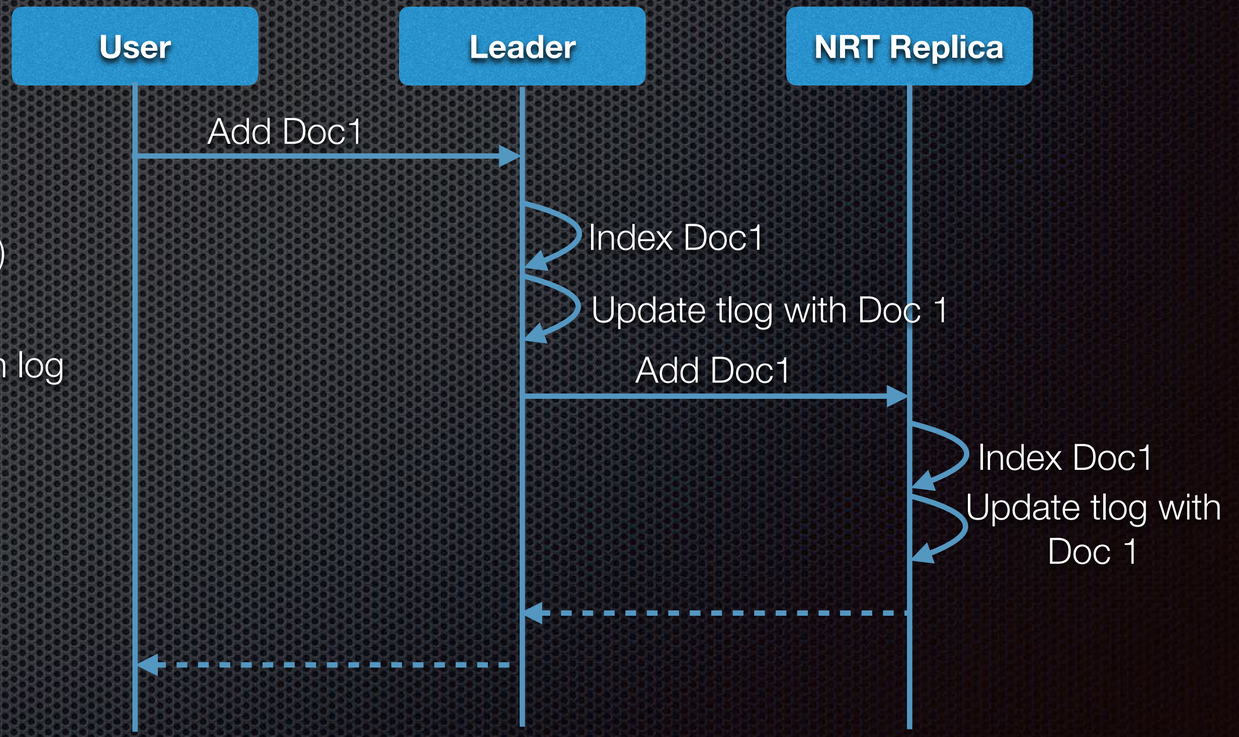
Replica types added

Replica Types Added

- NRT - Near Real Time
- TLOG - Transaction Log
- PULL - …Pulls indices only

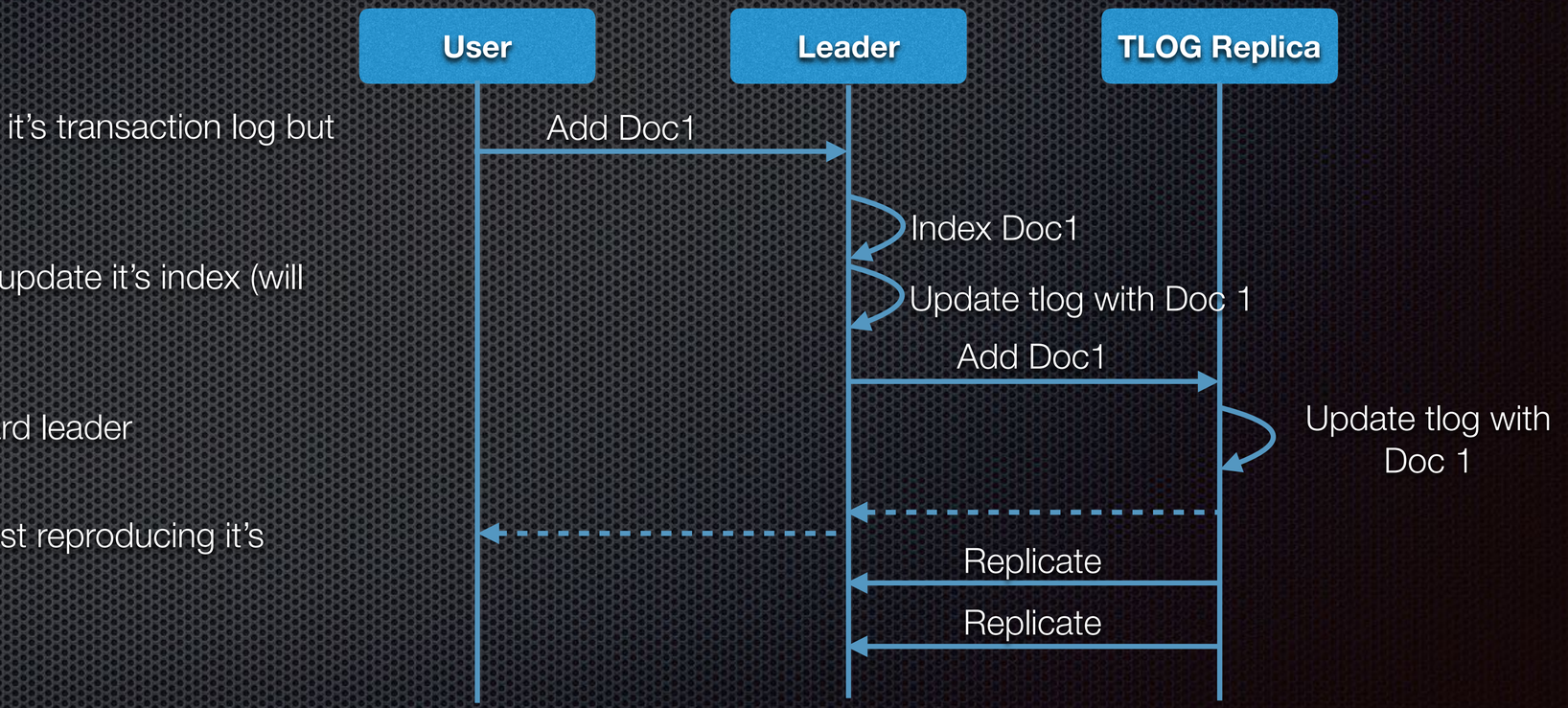
NRT Replicas

- The only existing type until 7.0 and the default type
- The only type of replica that supports Near-RealTime (softCommits)
- For every document, NRT replicas update it's index and transaction log
- Any NRT replica of the shard can become leader
- The only type of replica that supports RealTime Get



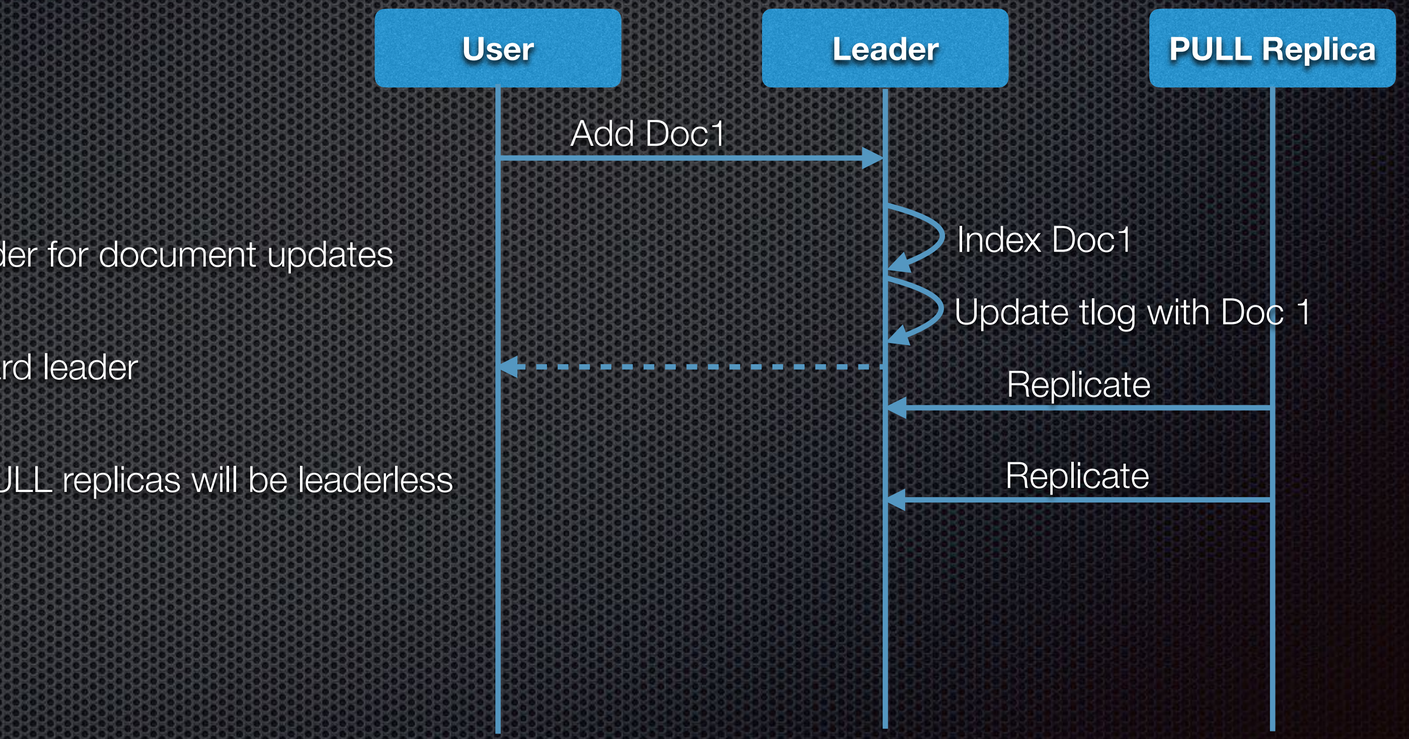
TLOG Replicas

- For every document, TLOG replicas update it's transaction log but not the index*
- A TLOG replica that is a shard leader WILL update it's index (will behave like a NRT type)
- Periodically replicate segment files from shard leader
- Any TLOG replica can become leader, by first reproducing it's transaction log



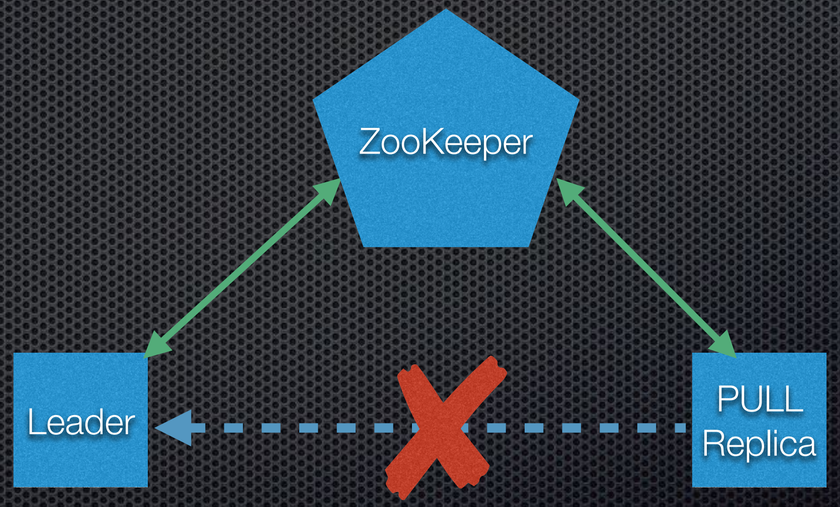
PULL Replicas

- PULL replicas are not contacted by the leader for document updates
- Periodically replicate segment files from shard leader
- Can’t become leaders. A shard with only PULL replicas will be leaderless



PULL Replicas

- PULL replicas can’t be in LIR, because are not contacted by the leader for document updates
- They can be out of date, for a long time if they can’t talk with the leader



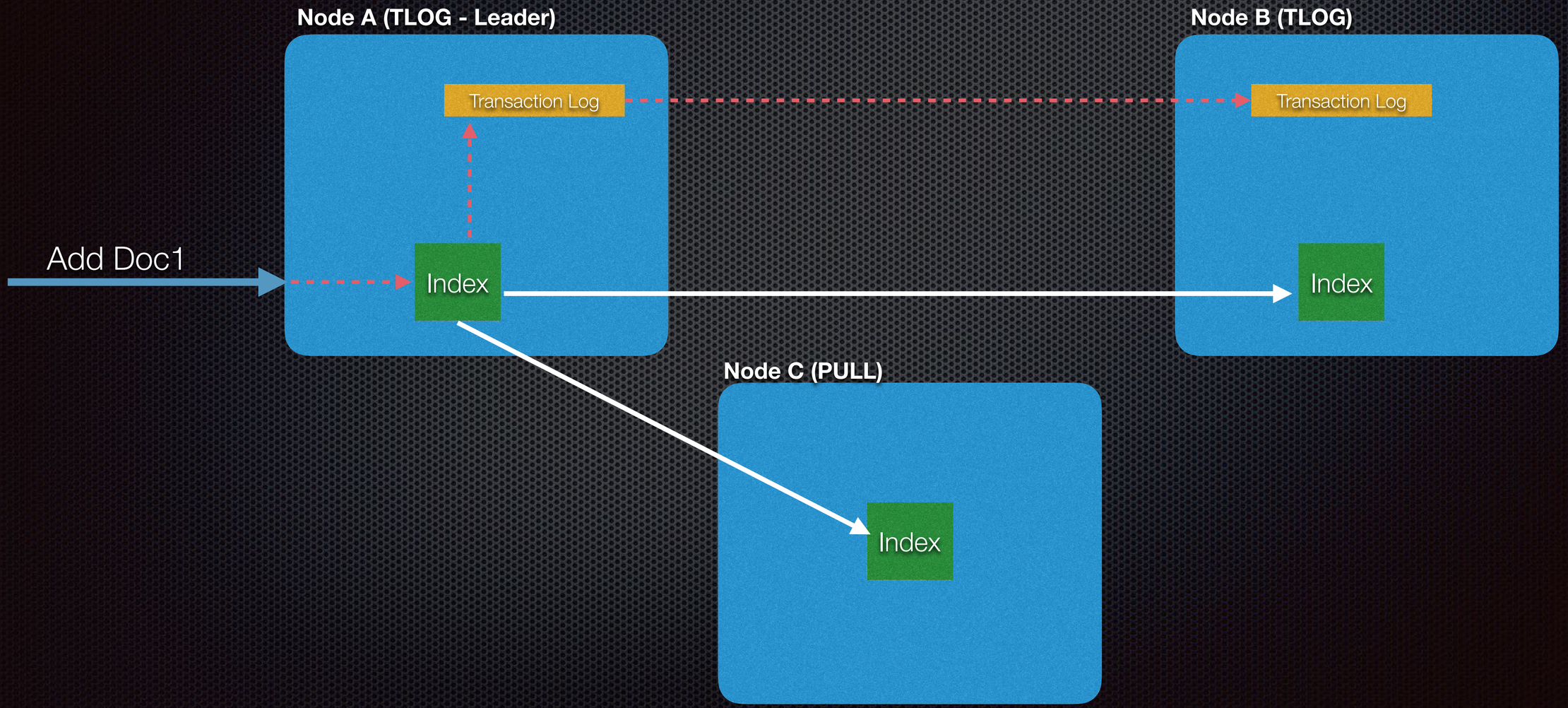
Replica Types Summary

What do they do?

	NRT	TLOG	PULL
Writes Index	YES	NO*	NO
Writes Transaction Log	YES	YES	NO
Receives every update	YES	YES	NO
Replicates periodically	NO	YES	YES

Replica Types Summary

What do they do?



Replica Types Summary

Supported features

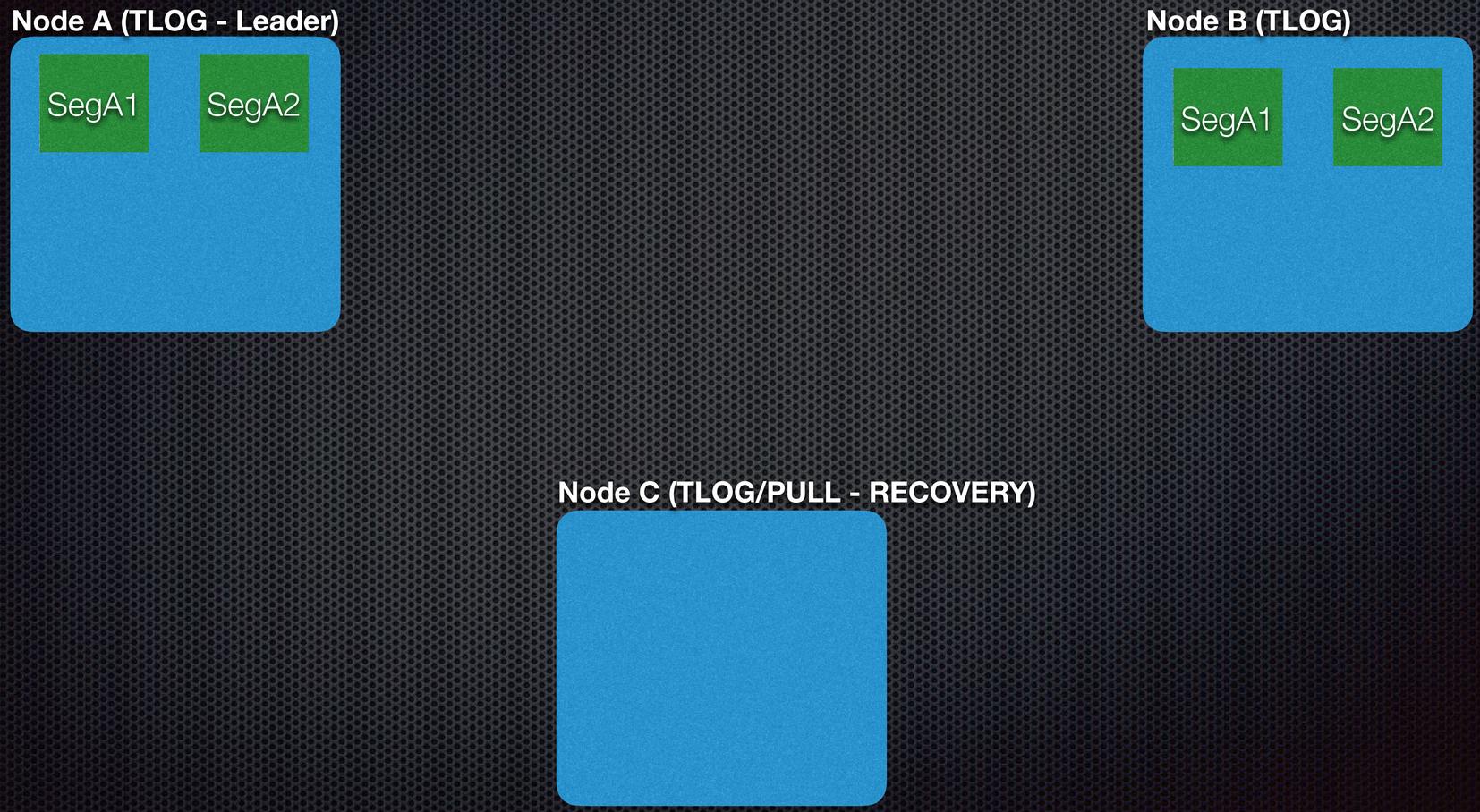
	NRT	TLOG	PULL
Supports Soft Commits (NRT)	YES	NO	NO
Supports RealTime Get	YES	NO*	NO
Can become leader	YES	YES	NO
Can be in LIR	YES	YES	NO

When creating a collection (or a shard), users can now choose how many replicas of each type they want, however only some combination of replica types are recommended

Combination of replica types in clusters

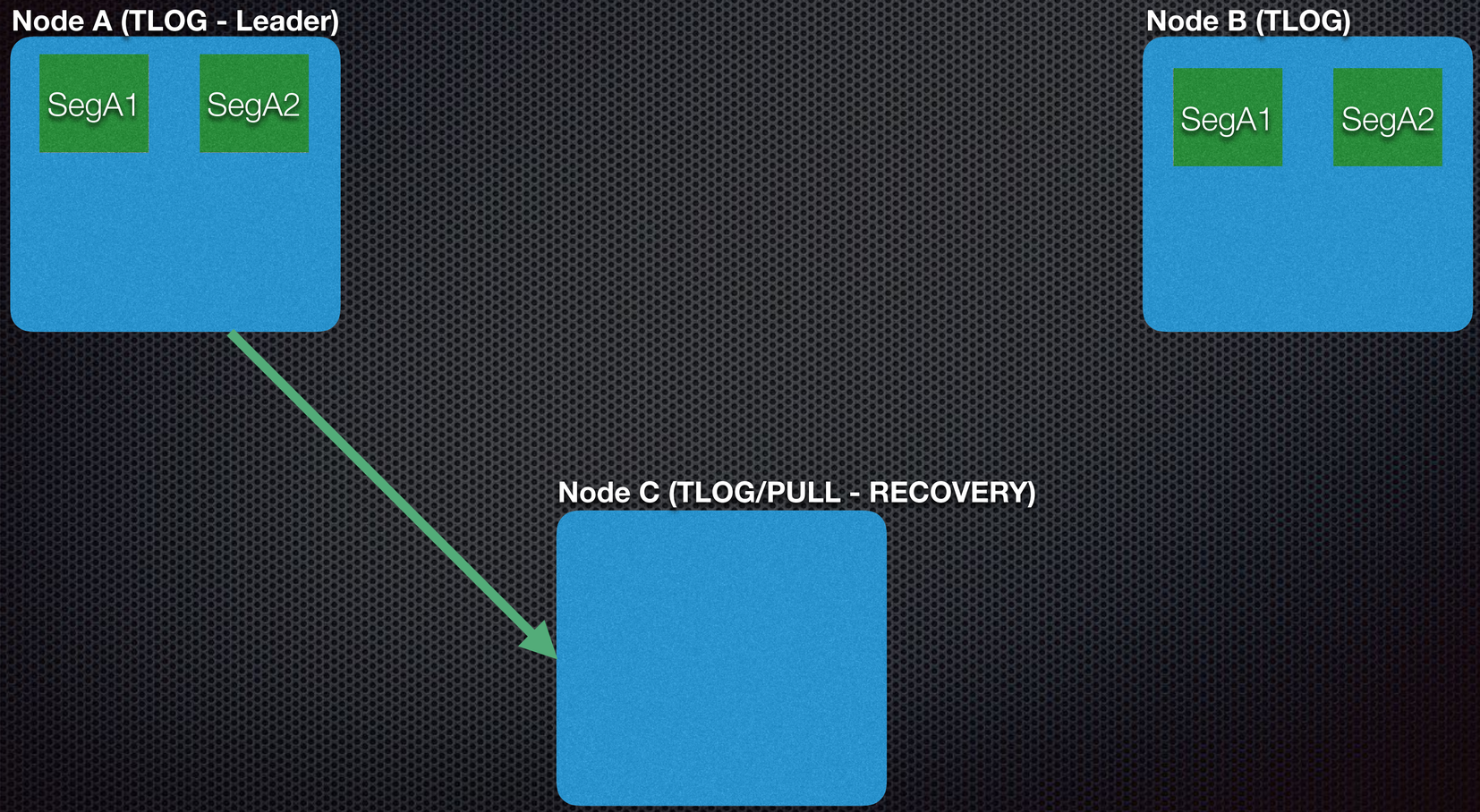
Replica Type combination	When to use?
All NRT	* This is the default configuration and the only combination before 7.0 * Use always when Near-Real-Time is needed * Small to medium size clusters, or with low to medium indexing throughput
All TLOG	* Near-Real-Time is not needed. * High update throughput * Medium to large clusters, but want all replicas to have all documents always
TLOG + PULL	* Near-Real-Time is not needed * High update throughput * Medium to large clusters, prefer availability of search over updates

Easier Recovery



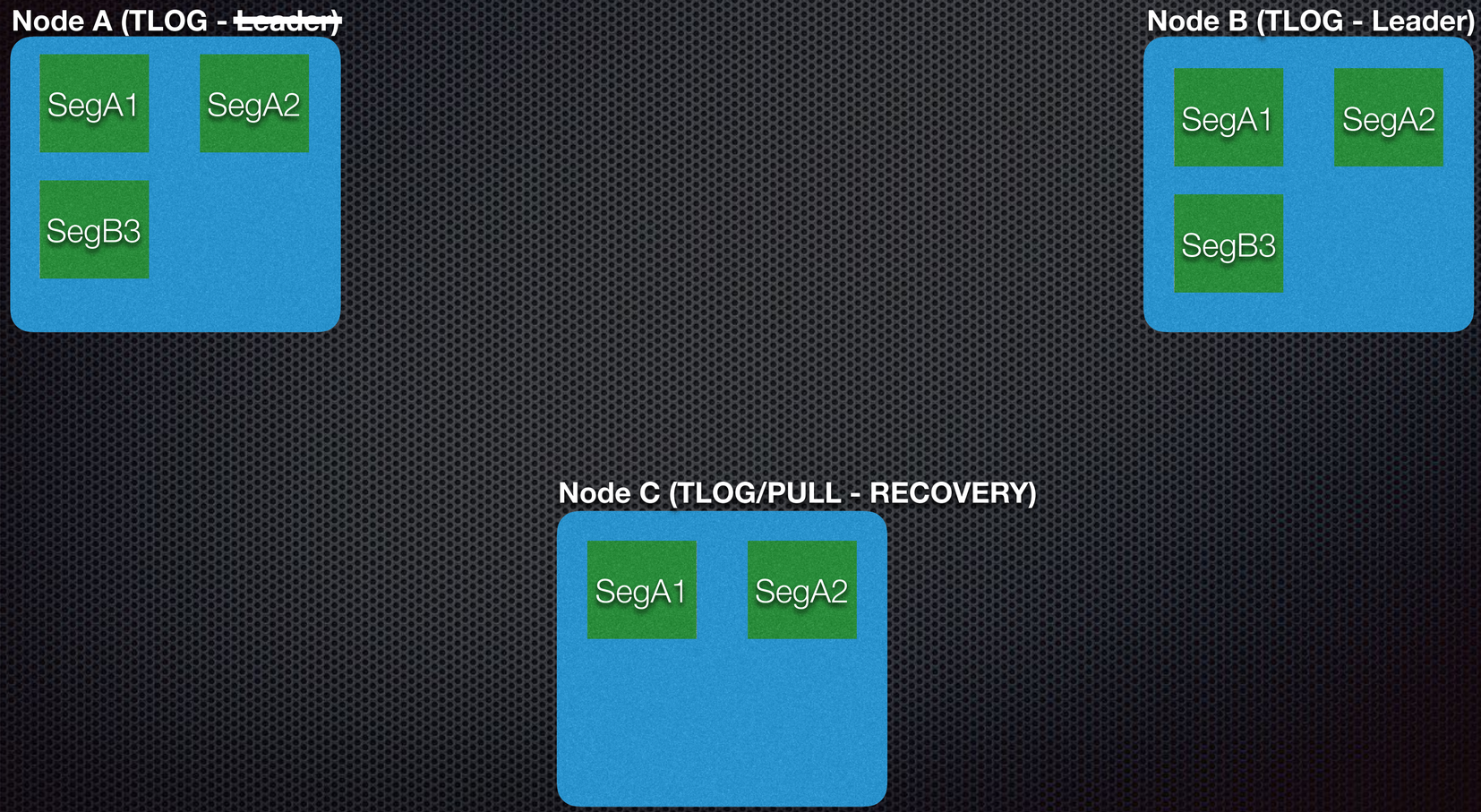
Easier Recovery

TLOG and PULL replicas share segments



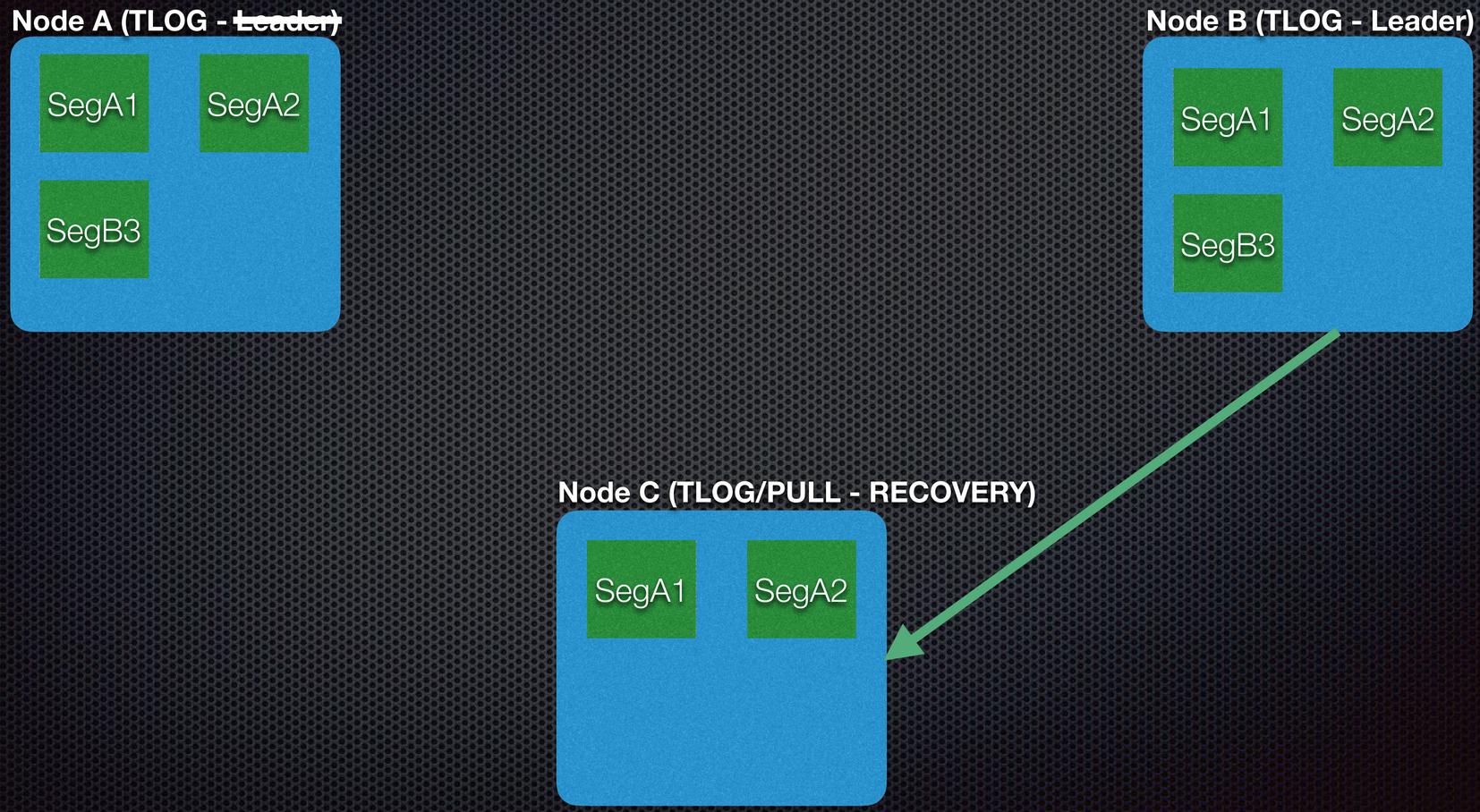
Easier Recovery

TLOG and PULL replicas share segments



Easier Recovery

TLOG and PULL replicas share segments

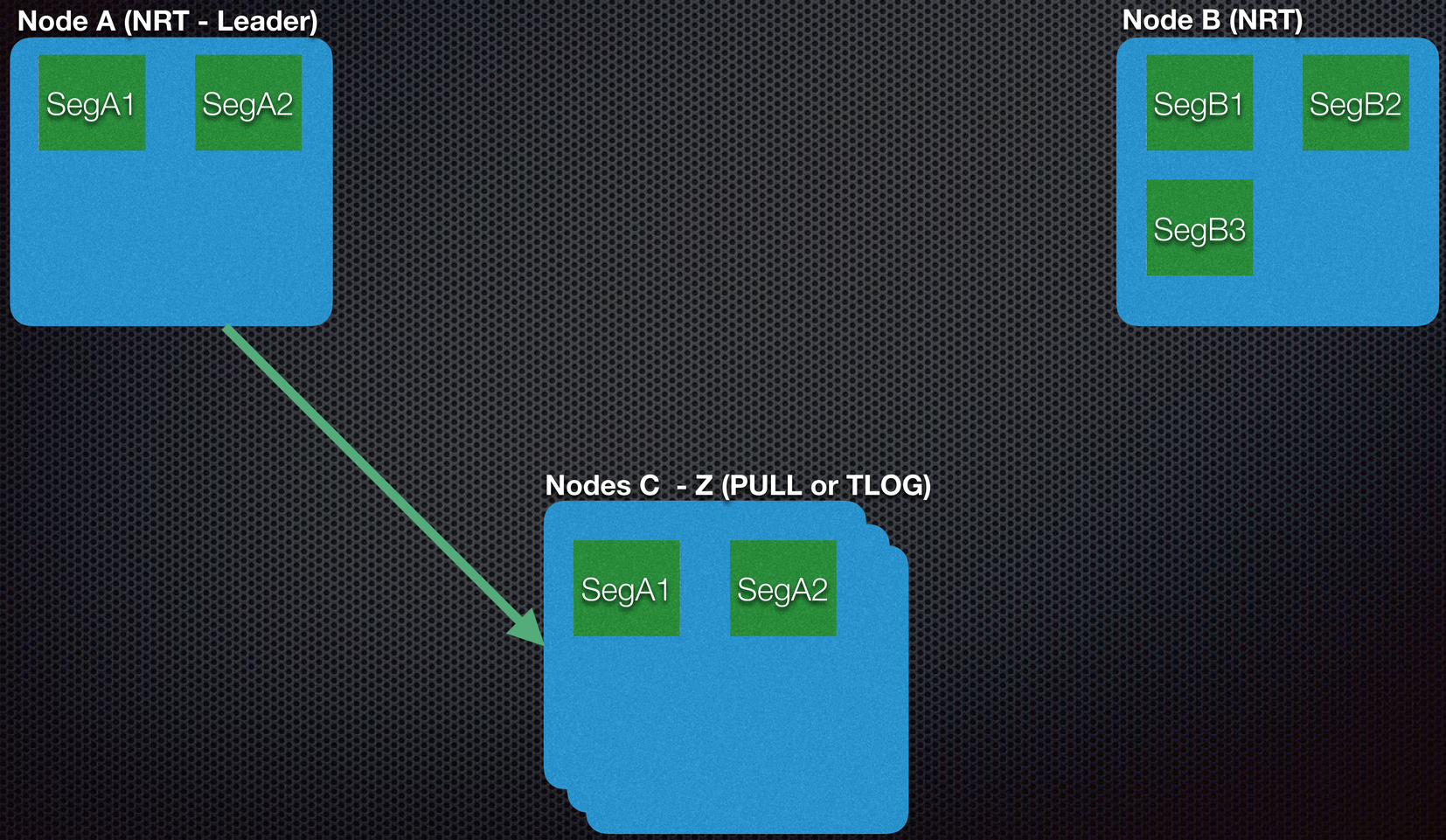


Combination of replica types in clusters

- If two or more nodes in the cluster write their own indices, any change of leadership between them will cause all TLOG and PULL replicas to require all the new index!

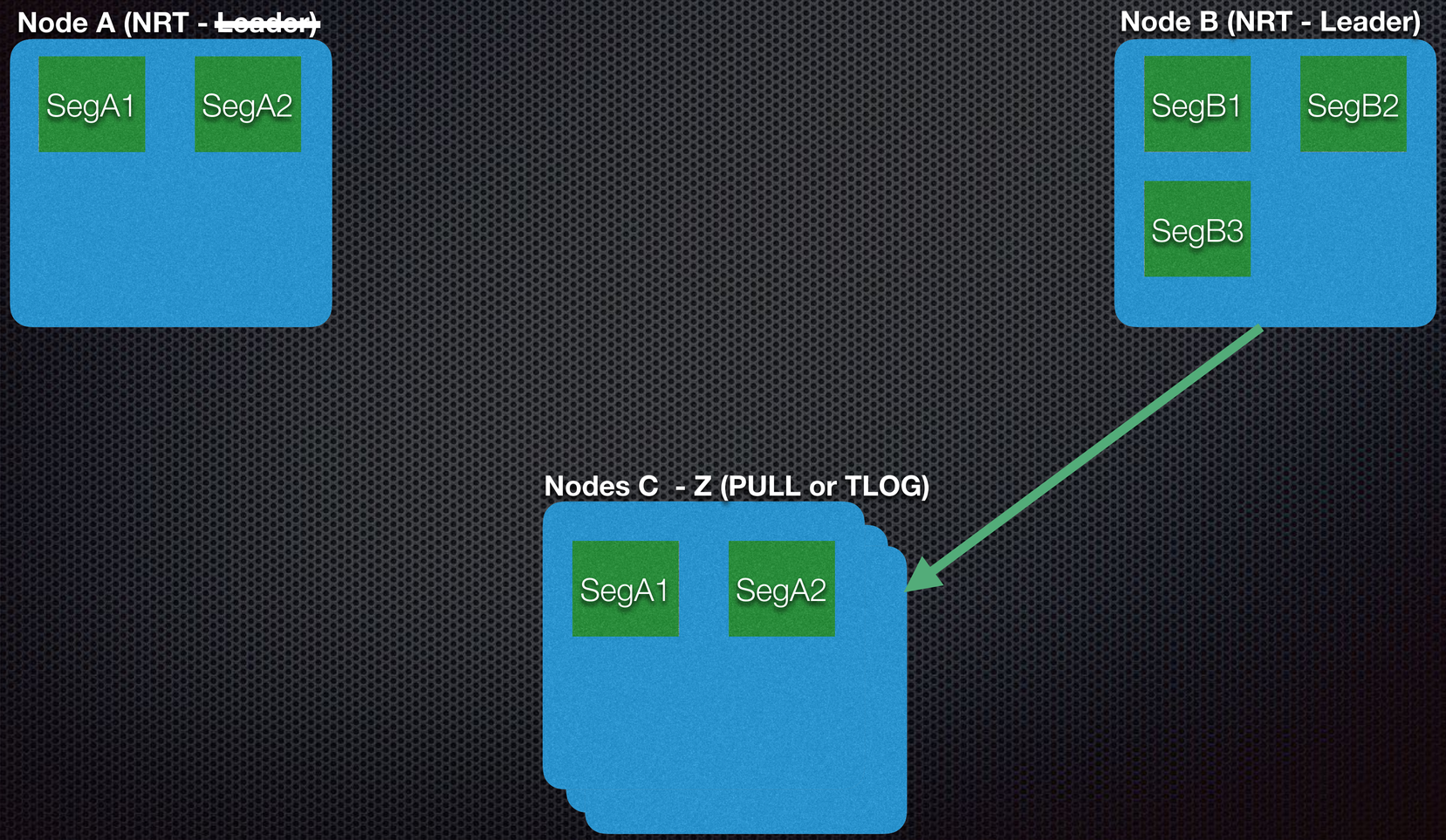
Combination of replica types in clusters

Not Recommended - Mix NRT with TLOG or PULL



Combination of replica types in clusters

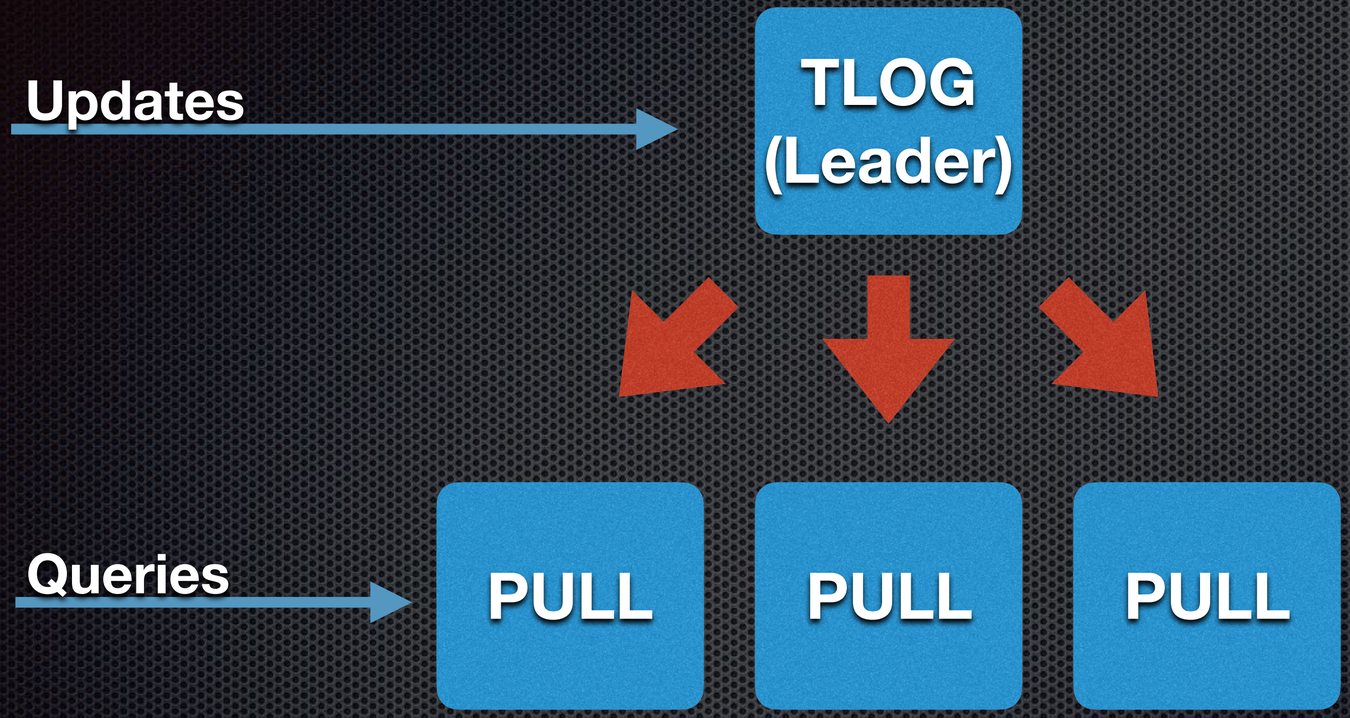
Not Recommended - Mix NRT with TLOG or PULL

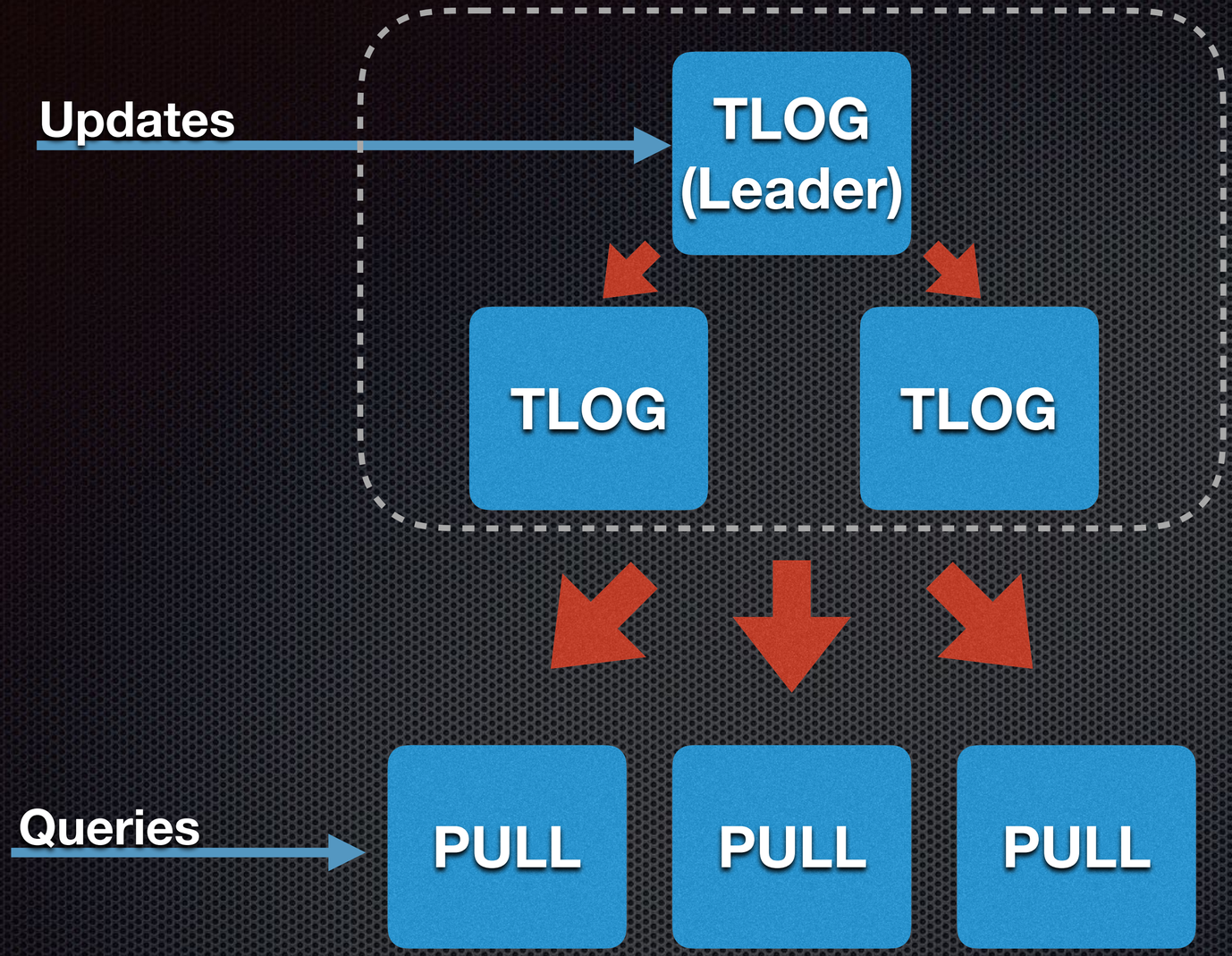


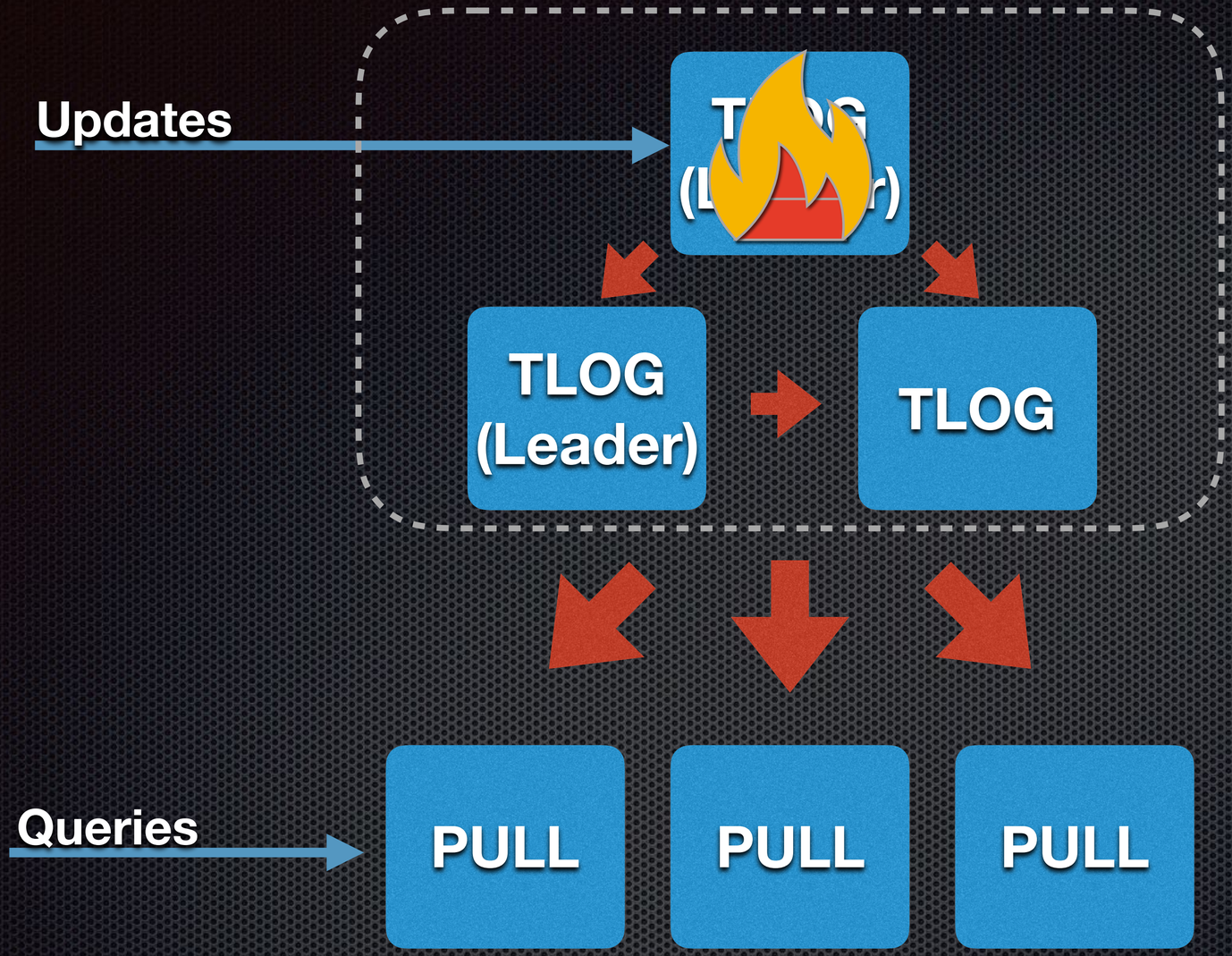
Combination of replica types in clusters

- PULL replicas can’t be leaders. A shard with only PULL replicas is a leaderless shard

Master/Slave in SolrCloud


Updates
TLOG
(Leader)
Queries
PULL
PULL
PULL


Updates
TLOG
(Leader)
TLOG
TLOG
Queries
PULL
PULL
PULL


Updates
TLOG
(Leader)
TLOG
(Leader)
TLOG
Queries
PULL
PULL
PULL

What does this mean?

- Prefer availability of search queries over document updates and NRT (no LIR)
- Separation of responsibilities. Updates can go to some replicas while queries will go to others*

What does this mean?

- High availability of writes
- Load balancing of query traffic and updates
- Collections API
- CloudSolrClient support
- Node discovery
- …

Multiple shards or collections

- Can use Autoscaling rules if you want to separate responsibilities for the whole node

How to use Replica Types

How to use Replica Types

V1:

- /admin/collections?action=CREATE…&**nrtReplicas=X&tlogReplicas=Y&pullReplicas=Z**
- /admin/collections?action=ADDREPLICA…&**type=[nrt/tlog/pull]**

V2:

- POST "http://host:port/v2/collections" -d ‘{create:{… **nrtReplicas=X,tlogReplicas=Y,pullReplicas=Z**}}’
- POST "http://host:port/v2/collections/myCollection/shards" -d ‘{add-replica:{…,**type:[NRT/TLOG/PULL]**}}’

How to use Replica Types

```
try (CloudSolrClient client = new CloudSolrClient.Builder().withSolrUrl("http://host:port/solr").build()) {
    CollectionAdminRequest.createCollection("myCollection", "_default", 1, 0, 2, 2)
        .process(client);
}
```

```
try (CloudSolrClient client = new CloudSolrClient.Builder().withSolrUrl("http://host:port/solr").build()) {
    CollectionAdminRequest.addReplicaToShard("myCollection", "shard1", Replica.Type.PULL)
        .process(client);
}
```

Autoscaling policy framework

- {"replica":"1", "shard":"#ANY" ,"port":8983, **"type":"NRT"**}
- {"replica":"1", "shard":"#ANY" ,"port":7574, **"type":"PULL"**}
- {"replica":"1", "shard":"#ANY" ,"port":7573, **"type":"TLOG"**}

Identifying types of replicas

```
…
"shards":{"shard1":{
        "range":"80000000-7fffffff",
        “state":"active",
        "replicas":{
          "core_node3":{
            "core":"myCollection_shard1_replica_t1",
            "base_url":"http://10.0.0.108:7574/solr",
            "node_name":"10.0.0.108:7574_solr",
            "state":"active",
            "type":"TLOG"},
…
```

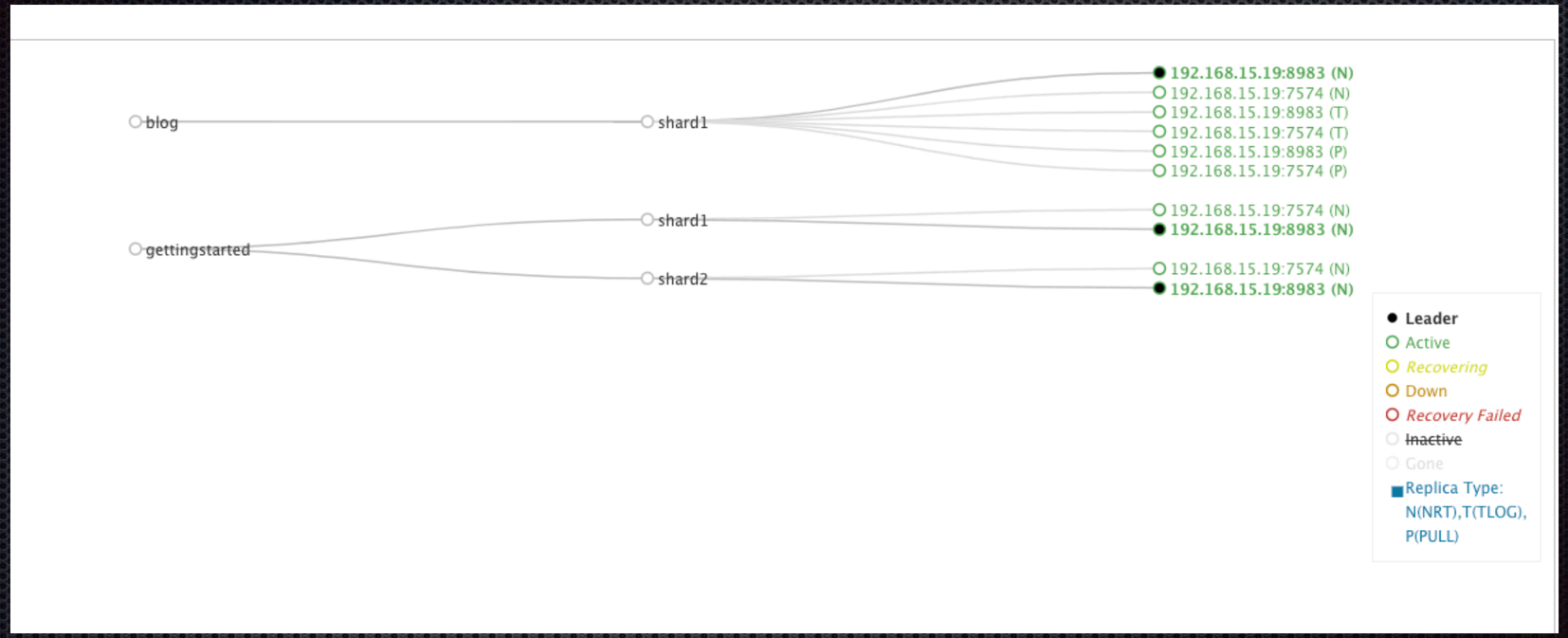
Identifying types of replicas

INFO [c:myCollection s:shard1 r:core_node8 **x:myCollection_shard1_replica_t1**] o.a.s.h.IndexFetcher; Master's generation: 1

INFO [c:myCollection s:shard1 r:core_node8 **x:myCollection_shard1_replica_t1**] o.a.s.h.IndexFetcher; Master's version: 0

…

Identifying types of replicas



https://issues.apache.org/jira/browse/SOLR-11578

Preferring some types over others for queries

/select?q=*:*&**shards.preference=replica.type:PULL**

Filtering types that can be used

/select?q=*:*&**shards.filter=replica.type:PULL** ?

https://issues.apache.org/jira/browse/SOLR-10880

TODOs and future work

TODOs and future work

- How old is my data? SOLR-10775
- Replication doesn’t always need to be from the shard leader
- Integration with CLI - SOLR-10772
- Replica types preference for single shard collections: SOLR-12217
- Allowing mixing NRT with PULL/TLOG
- NRT Replication

Thanks!