Distributed AG Cutover Runbook

Template - Zero-Downtime SQL Server Migration Checklist

> This is a sanitized template based on a real production cutover. All server names, AG names, IP addresses, and account names have been replaced with generic placeholders from the sample environment below. Copy this document, fill in your own environment, and walk through it step by step during your maintenance window. For the concepts behind it, see the companion articles on distributed availability groups and zero-downtime migration on powershelldba.de/blog.

Date: _____ Start: _____ End: _____

Responsible DBA: _____ AD Team: _____ App Team: _____

System Overview (Sample Environment)

Role	Server / Object Name	Remark
Old Cluster Primary	SQLA01	Source - decommissioned after migration
Old Cluster Secondary	SQLA02	
New Cluster Primary	SQLB01	Target - becomes new production system
New Cluster Secondary	SQLB02	
Old AG	AG_DatacenterA	Becomes secondary after migration
New AG	AG_DatacenterB	Becomes primary after failover
Distributed AG	DAG_Production	Connects both clusters
Listener PROD (moves along)	AG_DatacenterA	IP: 10.20.0.10 (example)
Listener TEMP (stays)	AG_DatacenterB	Temporary - delete after migration
Service Account	CONTOSO\SqlServiceAccount	

Prerequisites (verify before CutOver)

- ☐ All databases on SQLB01 show SYNCHRONIZED
- ☐ log_send_queue_size = 0 on all databases
- ☐ SQLB01 and SQLB02 are online and reachable
- ☐ AD team is ready and has been informed
- ☐ App team is ready for shutdown
- ☐ Rollback plan communicated and understood
- ☐ Maintenance window confirmed
- ☐ Backup available

Phase 1: Stop Application (App Team)

#	Executor	Action / T-SQL	OK
1	App Team	Shut down all applications in an orderly manner	☐
2	App Team	Confirm no active connections remain	☐
3	DBA	**SQLA01:** `SELECT count(*) FROM sys.dm_exec_sessions WHERE is_user_process = 1` *-- Expected: 0*	☐

Phase 2: Final Sync Check (DBA - on SQLA01)

#	Executor	Action / T-SQL	OK
4	DBA	**SQLA01:** `SELECT AGs.name, DB_NAME(database_id) AS DBName, synchronization_state_desc, last_hardened_lsn FROM sys.dm_hadr_database_replica_states replicaStates JOIN sys.availability_groups AGs ON AGs.group_id = replicaStates.group_id WHERE (is_distributed = 1) OR (is_primary_replica = 1)`	☐
5	DBA	ALL databases: synchronization_state_desc = SYNCHRONIZED, log_send_queue_size = 0 *-- If deviating, WAIT until sync is complete!*	☐

Phase 3: Distributed AG Failover (DBA - on SQLA01)

CRITICAL: After this step, SQLB01 is the new Primary. Applications still connect via AG_DatacenterA to the old cluster!

#	Executor	Action / T-SQL	OK
6	DBA	**SQLA01:** `ALTER AVAILABILITY GROUP [DAG_Production] MODIFY AVAILABILITY GROUP ON 'AG_DatacenterA' WITH (AVAILABILITY_MODE = SYNCHRONOUS_COMMIT), 'AG_DatacenterB' WITH (AVAILABILITY_MODE = SYNCHRONOUS_COMMIT)` *-- Switch to SYNC before failover!*	☐
7	DBA	**SQLA01:** `SELECT AGs.name, DB_NAME(database_id) AS DBName, synchronization_state_desc, last_hardened_lsn FROM sys.dm_hadr_database_replica_states replicaStates JOIN sys.availability_groups AGs ON AGs.group_id = replicaStates.group_id WHERE (is_distributed = 1) OR (is_primary_replica = 1)` *-- Wait until: synchronization_state_desc = SYNCHRONIZED*	☐
8	DBA	**SQLA01:** `-- SQL Server 2022: SET ROLE = SECONDARY first! ALTER AVAILABILITY GROUP [DAG_Production] SET (ROLE = SECONDARY)` *-- WARNING: the DAG is now briefly unavailable!*	☐
9	DBA	**SQLB01:** `-- Perform failover on the forwarder: ALTER AVAILABILITY GROUP [DAG_Production] FORCE_FAILOVER_ALLOW_DATA_LOSS`	☐

#	Executor	Action / T-SQL	OK
		-- No data loss - previously SYNC + SYNCHRONIZED!	
10	DBA	Wait 30 seconds	☐
11	DBA	**SQLB01:** `SELECT ars.role_desc FROM sys.dm_hadr_availability_replica_states ars JOIN sys.availability_replicas ar ON ar.replica_id = ars.replica_id WHERE ars.is_local = 1` *-- Expected: PRIMARY*	☐

Phase 4: Prepare Listener (DBA - BEFORE AD Team!) on SQLA01

CRITICAL: THIS STEP MUST BE PERFORMED BEFORE THE AD TEAM! If the cluster resource is deleted before the listener has been removed from the SQL AG, ALL databases immediately go into RECOVERY MODE!

#	Executor	Action / T-SQL	OK
9	DBA	**SQLA01:** `ALTER AVAILABILITY GROUP [AG_DatacenterA] REMOVE LISTENER N'AG_DatacenterA'`	☐
10	DBA	**SQLA01:** `SELECT name, state_desc FROM sys.databases WHERE database_id > 4` *-- ALL must remain ONLINE! If RECOVERY MODE: STOP, fix the problem!*	☐
11	DBA	Inform AD team: "Green light - cluster resource can now be swapped"	☐

Phase 5: Listener Swap in the Cluster (AD Team)

#	Executor	Action / T-SQL	OK
12	AD Team	Delete cluster resource AG_DatacenterA on old cluster (SQLA01/02)	☐
13	AD Team	Create new cluster resource AG_DatacenterA on new cluster (SQLB01/02)	☐
14	AD Team	Configure IP address 10.20.0.10	☐
15	AD Team	Bring cluster resource AG_DatacenterA online and verify	☐
16	AD Team	Update SPNs for CONTOSO\SqlServiceAccount: MSSQLSvc/AG_DatacenterA:1433 MSSQLSvc/AG_DatacenterA.contoso.com:1433	☐
17	AD Team	`nslookup AG_DatacenterA` *-- Must resolve to 10.20.0.10*	☐
18	AD Team	Inform DBA: "Cluster resource AG_DatacenterA is online"	☐

TIP - sqmSQLTool: Before handing step 16 to the AD team, run Get-sqmSpnReport against the new listener/instance. It checks the existing SPNs for the service account, flags anything missing or duplicated, and generates the exact setspn.exe commands (copied straight to the clipboard) so the AD team gets a ready-to-run list instead of typing SPNs by hand.

Phase 6: Activate Listener (DBA - on SQLB01)

#	Executor	Action / T-SQL	OK
19	DBA	**SQLB01:** `Get-ClusterResource | Where-Object Name -eq 'AG_DatacenterA'` *-- Must show State: Online!*	☐
20	DBA	**SQLB01:** `ALTER AVAILABILITY GROUP [AG_DatacenterB] ADD LISTENER N'AG_DatacenterA' (PORT = 1433)` *-- NO WITH IP! Cluster resource already exists.*	☐
21	DBA	**SQLB01:** `SELECT dns_name, port FROM sys.availability_group_listeners` *-- AG_DatacenterA must appear*	☐
22	DBA	**SQLB01:** `SELECT name, state_desc FROM sys.databases WHERE database_id > 4` *-- ALL must be ONLINE!*	☐

Phase 7: Validation (DBA + App Team)

#	Executor	Action / T-SQL	OK
23	DBA	`sqlcmd -S AG_DatacenterA -Q "SELECT @@SERVERNAME"` *-- Must return SQLB01!*	☐
24	DBA	ipconfig /flushdns on all relevant servers and clients	☐
25	App Team	Start applications	☐
26	App Team	Perform functional test (all critical functions)	☐
27	App Team	Confirm: application runs correctly on new system	☐

DONE: CutOver complete! Migration successful. Note the time: _____

Phase 8: Completion (DBA) - ONLY after a stable production phase (min. 24h)

CRITICAL: Do not delete anything prematurely! The old AG and Distributed AG are the rollback plan. Only clean up after 24h of stable operation.

#	Executor	Action / T-SQL	OK
28	DBA	`DROP AVAILABILITY GROUP [DAG_Production]`	☐
29	DBA	`DROP AVAILABILITY GROUP [AG_DatacenterA]`	☐
30	DBA	`ALTER AVAILABILITY GROUP [AG_DatacenterB] REMOVE LISTENER N'AG_DatacenterB'` *-- Delete temp listener*	☐
31	DBA	Decommission old servers SQLA01/02 (after change approval)	☐

Rollback Scenarios

Scenario 1: Problems after Phase 3, before listener swap (simple rollback)

NOTE: Distributed AG still present - rollback is easily possible!

#	Executor	Action / T-SQL	OK
R1	DBA	**SQLB01:** `ALTER AVAILABILITY GROUP [DAG_Production] FAILOVER` *-- AG_DatacenterA becomes Primary again*	☐
R2	DBA	Verify: AG_DatacenterA is Primary again on SQLA01	☐
R3	App Team	Start applications	☐

Scenario 2: Problems after Phase 6, after listener swap

#	Executor	Action / T-SQL	OK
R1	DBA	**SQLB01:** `ALTER AVAILABILITY GROUP [AG_DatacenterB] REMOVE LISTENER N'AG_DatacenterA'`	☐
R2	AD Team	Move listener AG_DatacenterA back to the old cluster	☐
R3	DBA	**SQLA01:** `ALTER AVAILABILITY GROUP [AG_DatacenterA] ADD LISTENER N'AG_DatacenterA' (PORT = 1433)`	☐
R4	DBA	**SQLB01:** `ALTER AVAILABILITY GROUP [DAG_Production] FAILOVER` *-- Back to the old system*	☐
R5	App Team	Start and test applications	☐

Scenario 3: Databases in RECOVERY MODE after listener swap

CRITICAL: Cause: AD team deleted the cluster resource before step 9 was performed!

#	Executor	Action / T-SQL	OK
R1	AD Team	Bring cluster resource AG_DatacenterA online on new cluster (if not already done)	☐
R2	DBA	**SQLB01:** `ALTER AVAILABILITY GROUP [AG_DatacenterB] ADD LISTENER N'AG_DatacenterA' (PORT = 1433)` *-- Databases automatically come out of RECOVERY MODE*	☐
R3	DBA	`SELECT name, state_desc FROM sys.databases WHERE database_id > 4` *-- All must become ONLINE*	☐

Schedule / Log

Phase	Description	Planned	Started	Completed	Status
Phase 1	App Team: Stop application				
Phase 2	DBA: Final sync check				
Phase 3	DBA: Distributed AG failover				
Phase 4	DBA: Prepare listener				
Phase 5	AD Team: Listener swap				
Phase 6	DBA: Activate listener				
Phase 7	App Team: Validation				
Phase 8	DBA: Completion (after 24h)				

Notes / Issues During CutOver

Appendix: How to Create a Distributed AlwaysOn Availability Group

Reference procedure for building the Distributed AG used in this runbook, using the same sample environment (SQLA01/SQLA02 = old cluster, SQLB01/SQLB02 = new cluster).

Prerequisites

- ☐ Two separate AlwaysOn Availability Groups already exist (one on each cluster)
- ☐ Both primary replicas are synchronous and healthy
- ☐ Network connectivity between clusters is stable
- ☐ Both clusters run SQL Server 2016 SP2 or later (2019+ recommended)
- ☐ All databases on both primaries are in SYNCHRONIZED state
- ☐ Database names match exactly on both clusters
- ☐ Login accounts are synchronized (or will be, via sqmSQLTool)

Architecture Overview

A Distributed AlwaysOn Availability Group connects two separate AlwaysOn AGs across different clusters. Replication is asynchronous by design due to network latency, and each cluster maintains its own synchronous replicas internally.

Key differences from a standard AG

- ☐ Replication is asynchronous (by design, due to network latency)
- ☐ Each cluster maintains its own synchronous replicas internally
- ☐ Distributed AG is a logical grouping, visible only on SQL Server
- ☐ Failover between clusters requires manual intervention (not automatic)

Step 1: Verify Current Availability Groups

Run the sync-state query from Phase 2 on SQLA01 to verify local AG details and database sync state before you start.

Step 2: Create Distributed AG on the Primary Cluster

Execute on SQLA01 to create the Distributed AG:

```
CREATE AVAILABILITY GROUP DAG_Production
WITH (DISTRIBUTED)
AVAILABILITY GROUP ON
'AG_DatacenterA' WITH (
    LISTENER_URL = 'TCP://ag-dca-listener.contoso.com:5022',
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
    FAILOVER_MODE = MANUAL,
    SEEDING_MODE = AUTOMATIC
),
'AG_DatacenterB' WITH (
    LISTENER_URL = 'TCP://ag-dcb-listener.contoso.com:5022',
    AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT,
    FAILOVER_MODE = MANUAL,
    SEEDING_MODE = AUTOMATIC
);
```

Expected output: is_distributed = 1, with both the local replicas and the secondary replica listed.

Step 3: Join Distributed AG on the Secondary Cluster

Execute on SQLB01 to join the secondary AG to the Distributed AG (same AVAILABILITY GROUP ON block as Step 2, using ALTER ... JOIN instead of CREATE).

Step 4: Verify Distributed AG Replication

Run the sync-state query on SQLA01 to check replication status. Wait for all databases on the secondary cluster to reach SYNCHRONIZED state.

Step 5: Monitor Seeding / Initial Replication

Use PowerShell with dbatools, or sqmSQLTool's Test-sqmDistributedAgReadiness / Get-sqmDistributedAgHealth, to monitor:

- ☐ AG health status
- ☐ Database sync state
- ☐ Log send queue size

Step 6: Synchronize Logins Between Clusters

Before any cutover, ensure all logins, jobs, and linked servers are replicated, for example with sqmSQLTool's Sync-sqmAgNode, which copies logins (including SID/password), SQL Agent jobs, linked servers, operators, and alerts from the primary to all secondaries.

Step 7: Verify SPNs Before Cutover

Kerberos authentication depends on the listener and instance SPNs being registered correctly for the service account. Run Get-sqmSpnReport (sqmSQLTool) against each instance and the new listener ahead of Phase 5. It checks the four expected MSSQLSvc SPNs per instance plus the listener SPNs, and, for anything missing, prepares ready-to-use setspn.exe commands (copied to the clipboard) that can go straight to the AD team instead of being typed by hand under time pressure during the cutover window.

Step 8: Prepare for Cutover

- ☐ All databases are SYNCHRONIZED on the secondary cluster
- ☐ Log send queue = 0 (zero replication lag)
- ☐ SPNs verified for the new listener (Get-sqmSpnReport)
- ☐ Rollback plan is documented and communicated
- ☐ Maintenance window is confirmed
- ☐ Application team is ready for connection failover
- ☐ AD team is ready for listener migration (if applicable)

Common Issues & Troubleshooting

Databases stuck in SYNCHRONIZING

Cause: network latency or a large transaction log. Solution: verify network connectivity and check log send queue size.

ALTER AVAILABILITY GROUP fails with a permission error

Cause: the service account lacks privileges. Solution: ensure the service account has sysadmin on both clusters.

Secondary replica shows DISCONNECTED

Cause: network or firewall blocking port 5022. Solution: check firewall rules and verify the mirroring endpoint exists.

Distributed AG not visible on the secondary

Cause: the join command failed silently. Solution: re-run ALTER AVAILABILITY GROUP ... JOIN on the secondary.

Applications fail with Kerberos / "cannot generate SSPI context" after listener swap

Cause: SPNs for the listener were not registered (or duplicated on the old cluster's computer account) before cutover. Solution: run Get-sqmSpnReport against the new listener, apply the generated setspn commands, then have clients re-authenticate (new Kerberos ticket).

Rollback: Remove the Distributed AG

You can remove the Distributed AG without affecting the local replicas. Execute on SQLA01:

```
ALTER AVAILABILITY GROUP [DAG_Production] REMOVE REPLICA ON N'AG_DatacenterB';
```

No action is needed on the secondary AG; it reverts to standard AG status.