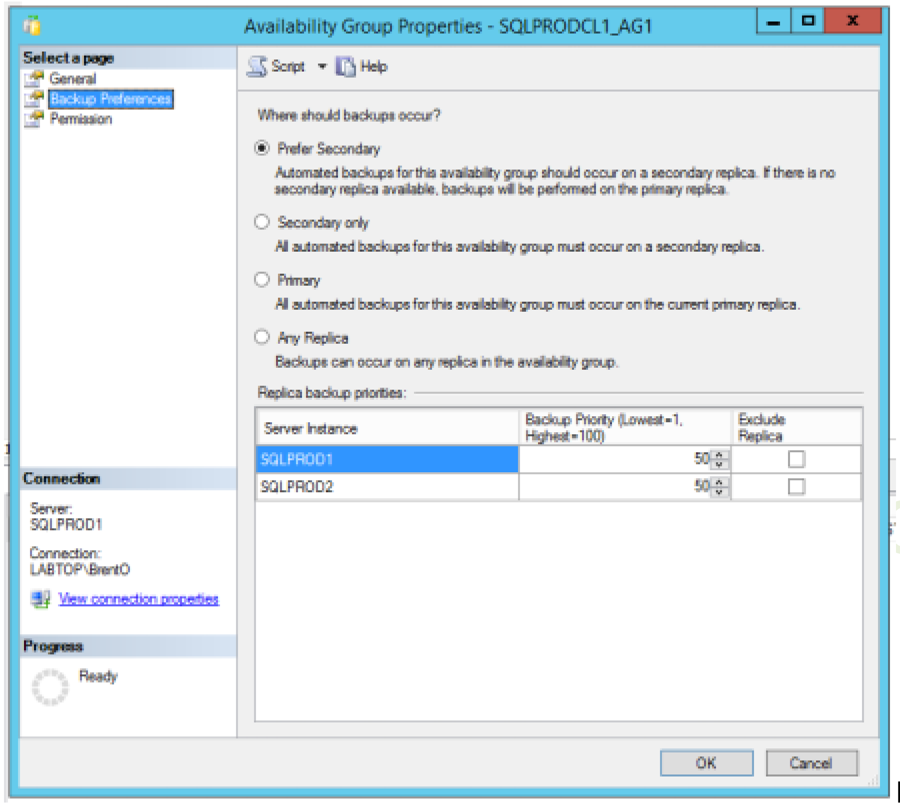
Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Owner: Brent Ozar

With SQL Server AlwaysOn Availability Groups, you can offload backups to a replica rather than running them on the primary. Here's how to do it:

1. Install Ola Hallengren's utility scripts on all of the replicas. During the install, it creates a laundry list of SQL Agent jobs, but doesn't set up schedules for any of them. More on that in a couple of steps.



Where should backups occur?

[✓ Prefer Secondary]
Automated backups for this availability group should occur on a secondary replica. If there is no secondary replica available, backups will be perfomed on the primary replica.

Secondary only
All automated backups for this availability group must occur on a secondary replica.

Primary
All automated backups for this availability group must occur on the current primary replica.

Any Replica
Backups can occur on any replica in the availability group.

Replica backup priorities:

Server Instance	Backup Priority (Lowest=1, Highest=100)	Exclude Replica
SQLPROD1	50	☐
SQLPROD2	50	☐

Connection
Server:
SQLPROD1
Connection:
LABTOP\BrentO
View connection properties

Progress
Ready

Backup preference settings

champs.com

2. Set your backup preferences. In SSMS, right-click on your Availability Group, click Properties, and click the Backup Preferences pane.

The first option, "Prefer Secondary," means that your backups will be taken on a secondary server unless all secondaries go offline, at which point they'll be taken on the primary. There's some risk here: if communication falls behind, your secondary may be running backups of old data, as Anthony Nocentino explains. In that case, you won't get backup failure alerts, but you won't be able to meet your RPO. Monitoring for that is an exercise left for the reader.

In the "Replica backup priorities" window, rank your replicas to choose who will do the backups first.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

Say I have 3 servers – two in my primary data center, and one in my disaster recovery site. I'd rather have my full backups running in my primary data center because if I need to do a restore, I want the backups nearby. (You can also run backups in both places – and I would – but more on that in a future post.)

To configure that, I'd set priorities as:

- SQLPROD1 and SQLPROD2 (my primary data center replicas) – both 50 priority
- SQLDR1 (my disaster recovery replica) – priority 40

3. Test your backup preferences. Run this query on each replica:

```
SELECT d.database_name,
  sys.fn_hadr_backup_is_preferred_replica (d.database_name) AS IsPreferredBackupReplicaNow
FROM sys.availability_databases_cluster d
```

This returns a list of databases in an AG, and whether or not they're the preferred backup replica right now. Check that on all of your replicas to make sure backups are going to run where you expect, and if not, revisit your backup preferences in the last step.

4. Configure Ola's Agent full backup jobs. On any replica, in the Agent job list, right-click on the "DatabaseBackup – USER_DATABASES – FULL" job, click Properties, click Steps, and edit the first step. By default, it looks like this:

https://www.sqldbachamps.com

```
sqlcmd -E -S $(ESCAPE_SQUOTE(SRVR)) -d master -Q "EXECUTE [dbo].[DatabaseBackup] @Databases = 'USER_DATABASES', @Directory = N'C:\Backup', @BackupType = 'FULL', @Verify = 'Y', @CleanupTime = NULL, @CheckSum = 'Y', @LogToTable = 'Y'" -b
```

You need to change these parts:

- @Directory – set this to your backup path. I like using a UNC path that all of the replicas can access.
- @Verify – I'd recommend turning this off to make your backup jobs go faster. If you really want to verify your backups, restore them on another server.
- If you want to run the backups on a secondary replica rather than the primary, add a parameter for @CopyOnly='Y'
- If you only want to back up specific databases, modify the @Databases parameter. I don't like doing that – I'd rather have one job for all of my backups. The way this is set now (USER_DATABASES), this one job will back up all of my databases that aren't in an AG, plus it'll back up the AG-contained databases where this replica is the preferred backup right now.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

With AlwaysOn AGs, replicas can only run copy-only backups, and people often think that's a problem. It's only a problem if you want to use differential backups – otherwise, with AGs, it doesn't affect transaction log handling at all. I don't recommend using differentials with AlwaysOn AGs, but if you insist on doing it, you'll be running your full backups on your primary replica.

Other parameters you may want to set:

- @CleanupTime – how many hours of backup files you want to keep around
- @Compress – Ola inherits the default compression setting at the server level, but you can hard code this to Y if you want to make sure backups are always compressed

So here's what my Agent job script ends up looking like, with my changed parts in bold:

sqlcmd -E -S $(ESCAPE_SQUOTE(SRVR)) -d master -Q "EXECUTE [dbo].[DatabaseBackup] @Databases = 'USER_DATABASES', **@Directory = N'\\FILESERVER1\SQLBackups'**, **@CopyOnly='Y'**, @CleanupTime=48, @Compress='Y', @BackupType = 'FULL', @Verify = 'N', @CleanupTime = NULL, @CheckSum = 'Y', @LogToTable = 'Y'" -b

5. Copy these changes to all replicas and test the full jobs. Make sure each replica that isn't supposed to run the fulls, doesn't, and the replica that IS supposed to run the fulls, DOES. In really mission-critical environments where we're building the new AG servers from the ground up, we actually fail the AG around to different servers to test behavior when different servers are the primary – and when entire data centers go down.

https://www.sqldbachamps.com

6. Repeat the setup with your log backup jobs. Right-click on the "DatabaseBackup – USER_DATABASES – LOG" job and click Properties, Steps, and edit the first step. Set the @Directory and @Verify steps as we did earlier.

Here's where things get a little tricky – you don't have to add @CopyOnly='Y' for the log backup steps. There's no such thing as a copy-only log backup in an Availability Group secondary, much to my dismay.

You might also consider setting the @ChangeBackupType parameter to Y. By default, if Ola can't do a transaction log backup (like if it's a brand new database that has never had a full backup before), then the log backup is skipped. If you set @ChangeBackupType='Y', then Ola will do a full backup in that situation, and then do a log backup. However, if it's a large database, this might take a while to perform the full, and this will tie up your log backup job while it runs. Say the full takes 20 minutes to perform – this might blow your RPO/RTO commitments.

Praveen Madupu
Mb: +91 98661 30093
Sr. Database Administrator

7. Copy these changes to all replicas and test the jobs. Same thing here that we did with the full backups.

8. Configure your DBCC CHECKDB jobs. You need to check for corruption on any server where you're running backups – here's why.

9. Design your monitoring. Sadly, SQL Server doesn't centralize backup history, so it's up to you to poll all of your replicas to find out where backups are happening for any given Availability Group. In one case, I had a DBA change the backup preferences and Ola's job settings incorrectly, and all of the backup jobs were succeeding – but none of them were backing up one of his Availability Groups.

10. Set yourself a weekly reminder to test restores. AG backups are notoriously complex, and if you cared enough to set up this whole expensive infrastructure, then you should care enough to test it. Make sure you have a good, restorable set of backups.

https://www.brentozar.com/archive/2015/06/how-to-configure-alwayson-ag-backups-with-ola-hallengrens-scripts/

https://www.sqldbachamps.com