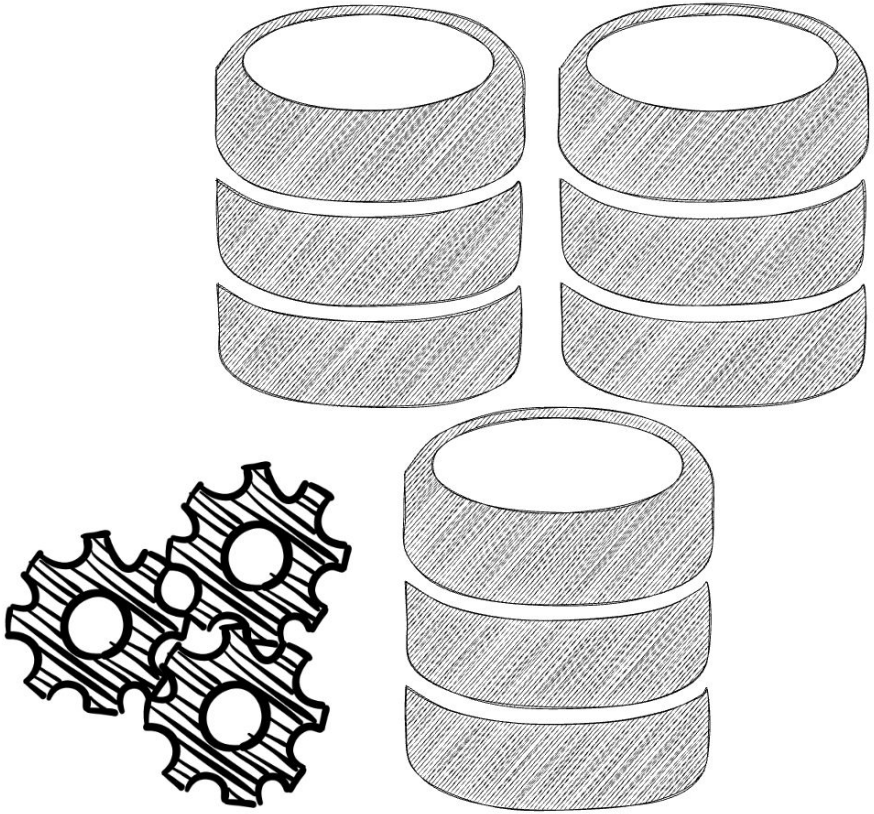
Next-Gen High Availability with Contained Availability Groups



Nader Sharara

Who am I?

Microsoft CERTIFIED
IT Professional
Database Administrator 2008

Microsoft CERTIFIED
Solutions Expert
Data Platform

aws certified
Database
SPECIALTY

sqlspark.com

nader-sharara-db

Database Administrator & Architect with ~15 years of experience.

Passionate about databases, especially SQL Server.

Microsoft & AWS Certified in Databases.

Played different roles from DBA to Database Engineering Manager to Cloud Database Architect.

Currently focused on building SQL Server cloud managed services for STACKIT.

Share technical insights on sqlspark.com.

Enjoy playing Squash.

Egyptian, living in Berlin for 7 years now.

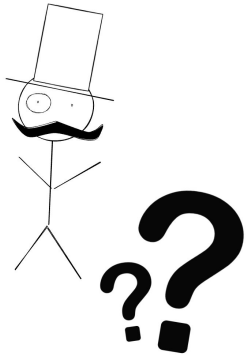
What are we going to talk about today??

- The “4 questions” about Contained Availability Groups:
  - Why? → The problem definition.
  - What? → Introducing contained availability groups.
  - How? → Digging deep inside contained availability groups.
  - When? → Common use cases for contained availability groups.

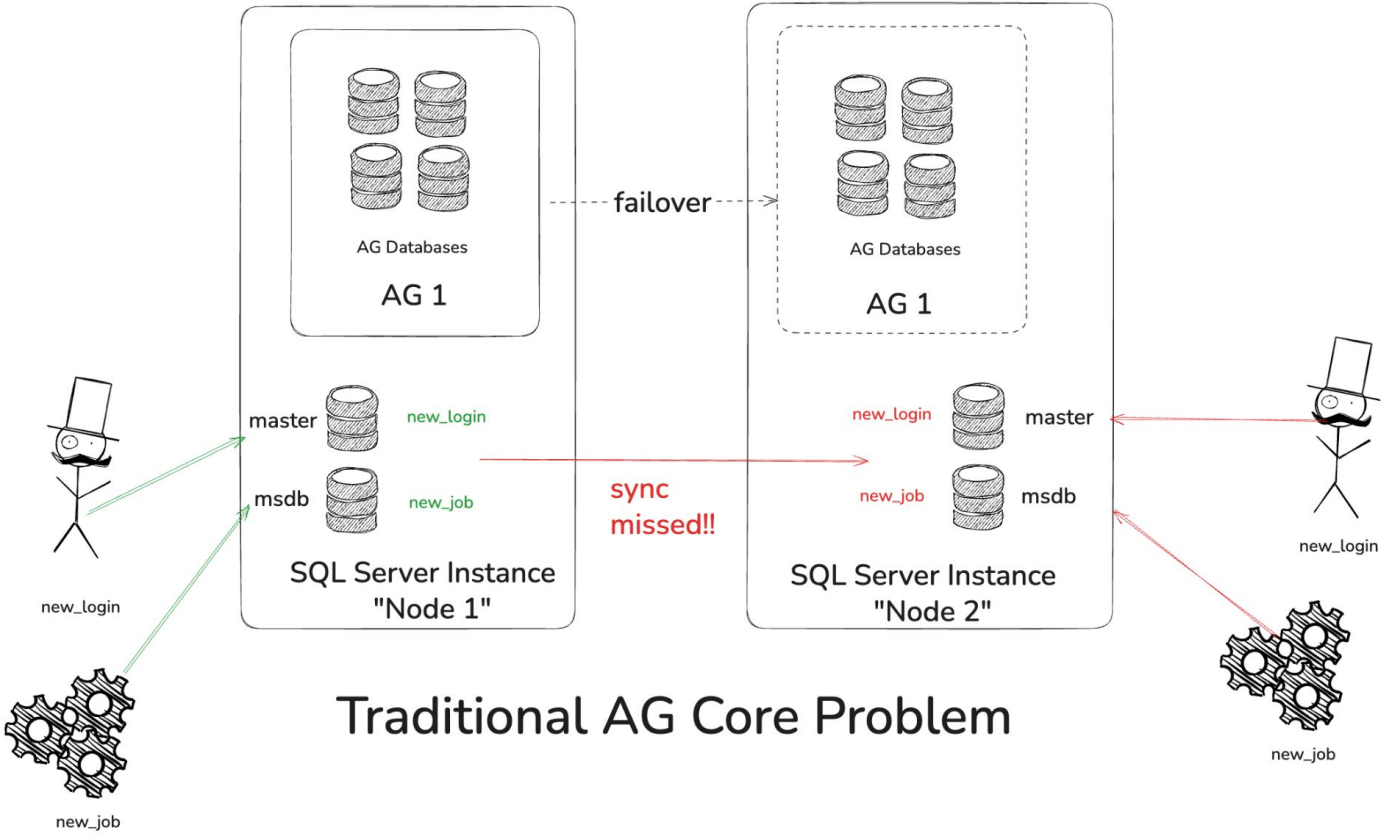
- Contained availability groups limitations.

- Some small demos.

Why Contained AGs??



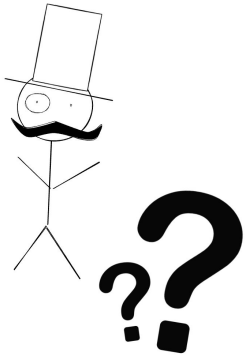
What is the problem with traditional AGs?



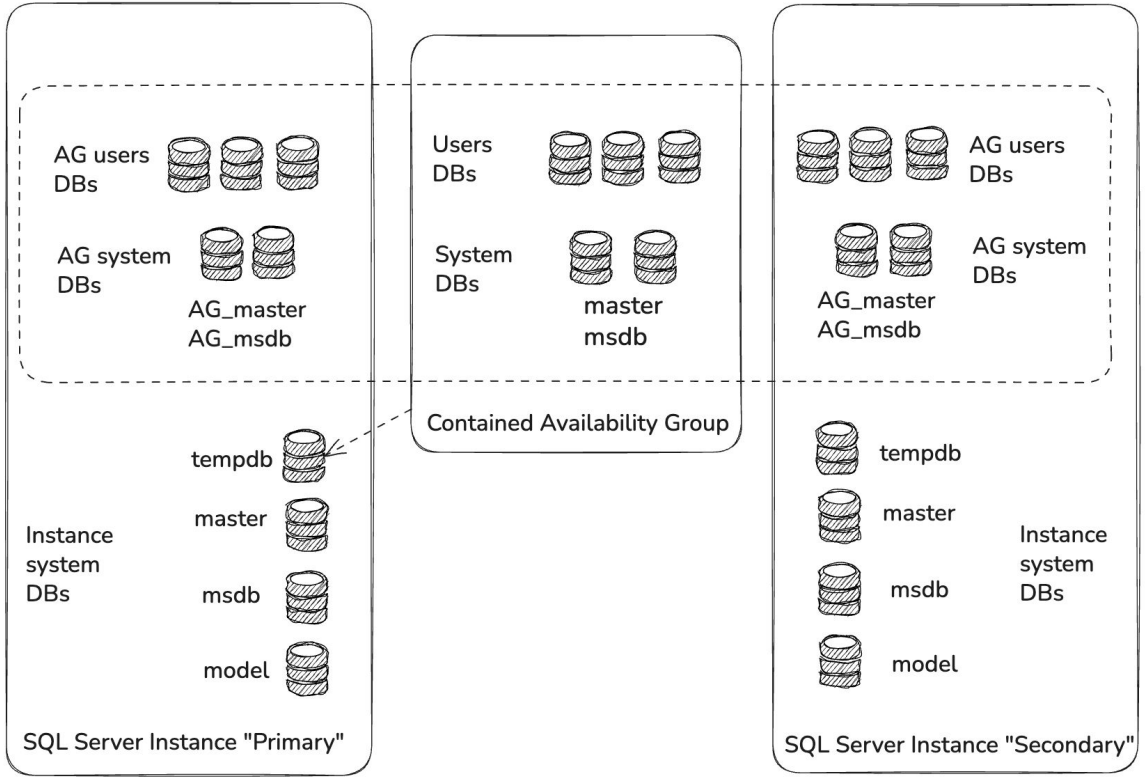
What is the problem with traditional AGs? (cont.)

- Objects dependency & lack of portability.
- Manual synchronization is required.
- Increased complexity by adding a dependency.
- Application Outages: A failover occurring before object synchronization is complete leads to connection failures and job execution errors.

What is a Contained AG??

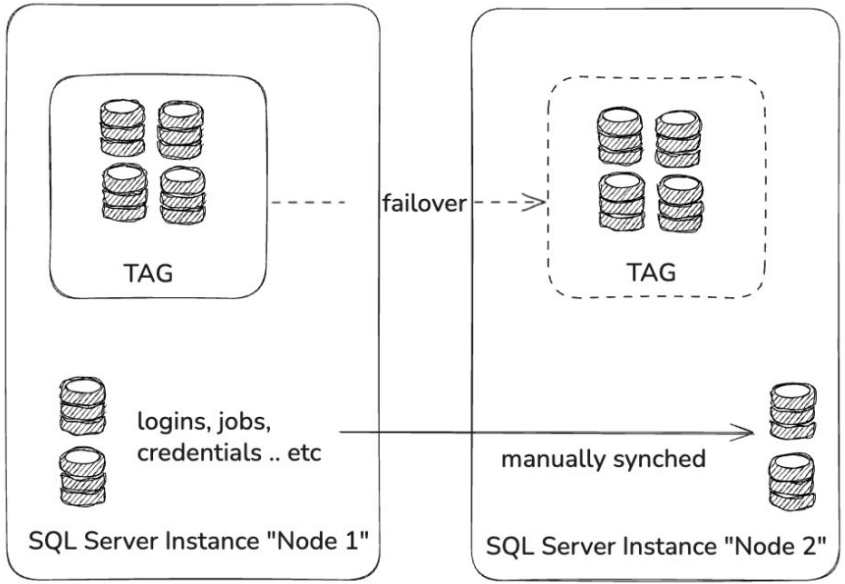


Contained Availability Group Architecture

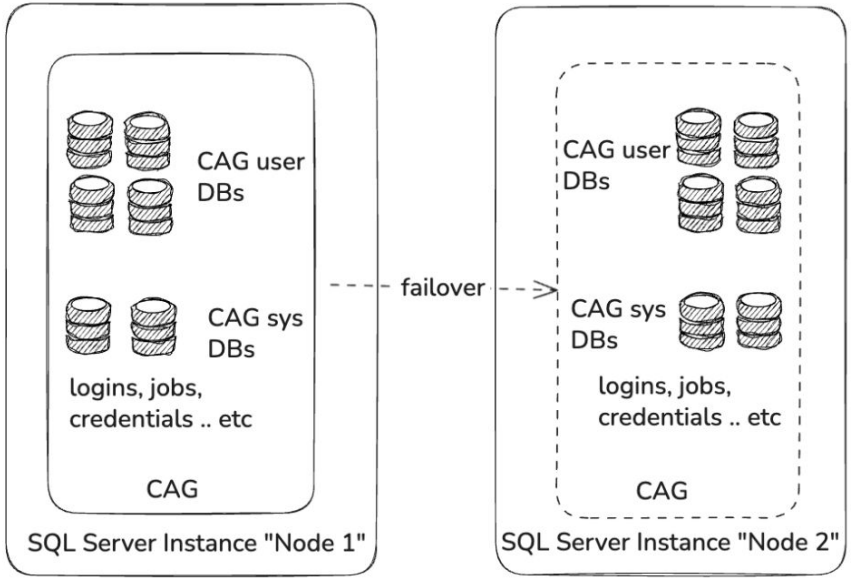


Contained Availability Group Architecture

Contained Availability Group vs. Traditional Availability Group

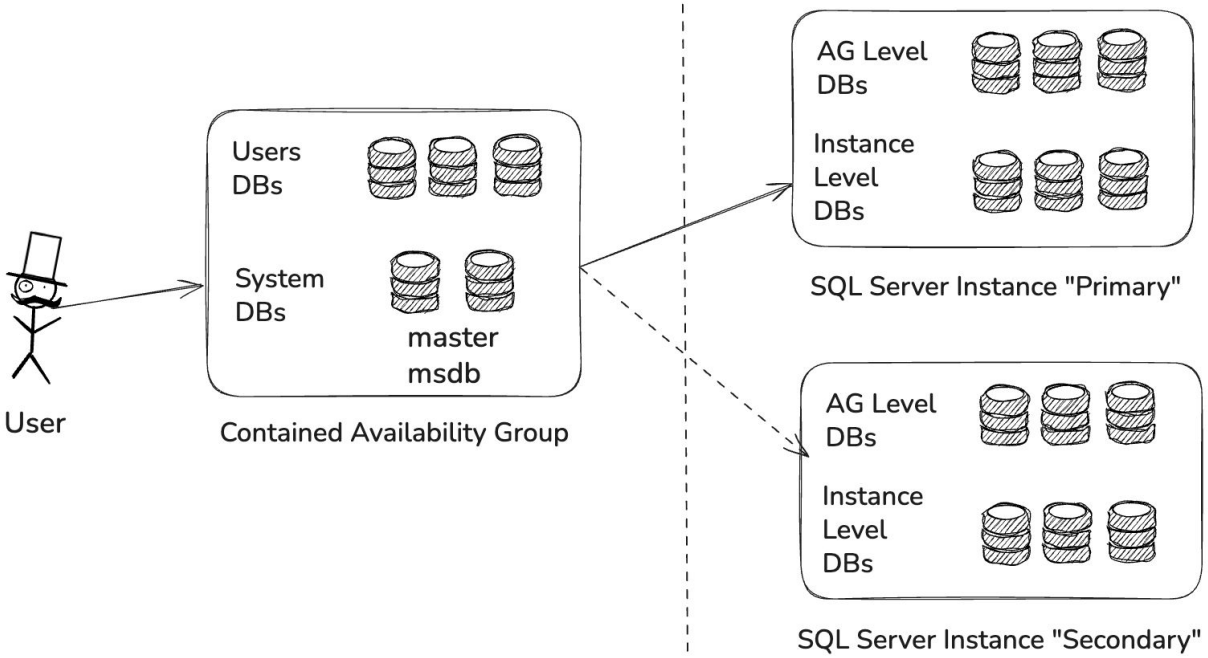


Traditional Availability Group



Contained Availability Group

The Contained AG Abstracted Layer

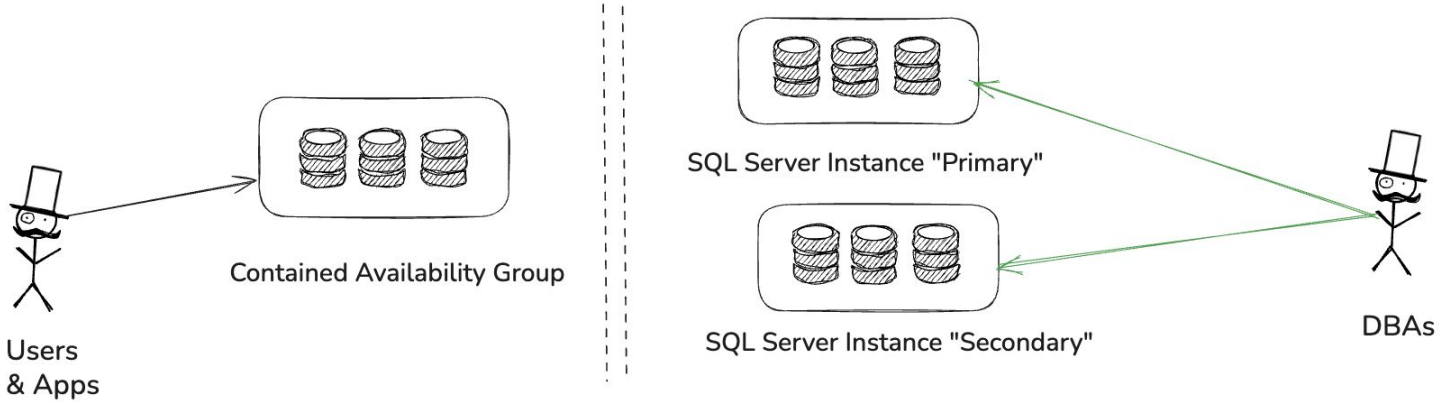


The Contained AG Abstracted Layer

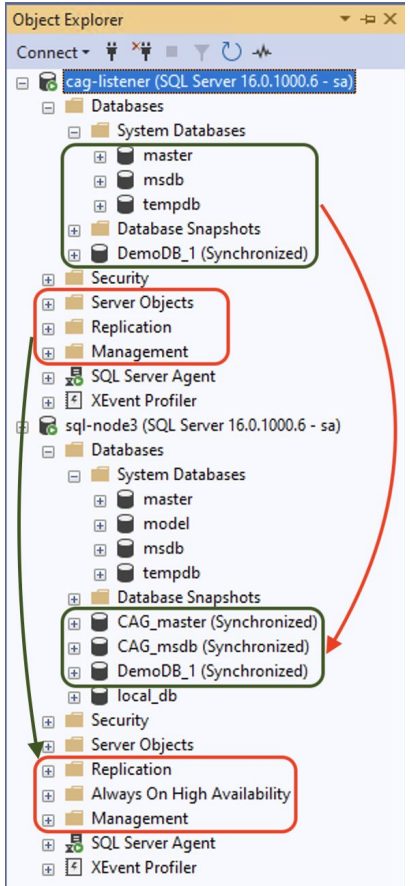
The Contained AG Abstracted Layer (cont.)

- The primary differentiator is that a Contained AG listener connects to an abstracted, self-contained environment, while a Traditional AG listener transparently redirects the connection to the physical primary instance.

- A logical separation that confines a user's rights and actions to a specific environment, distinct from the broader administrative infrastructure.



The Contained AG Abstracted Layer (cont.)



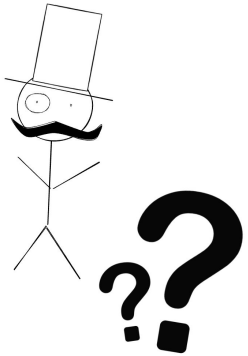
- No access to Always On High Availability from SSMS.
- Access via T-SQL requires “sysadmin or [VIEW SERVER STATE + VIEW ANY DEFINITION]”.
- The contained AG system databases have a different physical name “[AG name] + _ + [system DB name]

	name	physical_database_name	physical_name
1	msdb	CAG_msdb	D:\SQLData\MSSQL16.MSSQLSERVER\MSSQL\DATA\CAG_msdbData.mdf
2	msdb	CAG_msdb	D:\SQLData\MSSQL16.MSSQLSERVER\MSSQL\DATA\CAG_msdbLog.ldf
3	master	CAG_master	D:\SQLData\MSSQL16.MSSQLSERVER\MSSQL\DATA\CAG_master.mdf
4	master	CAG_master	D:\SQLData\MSSQL16.MSSQLSERVER\MSSQL\DATA\CAG_masterlog.ldf

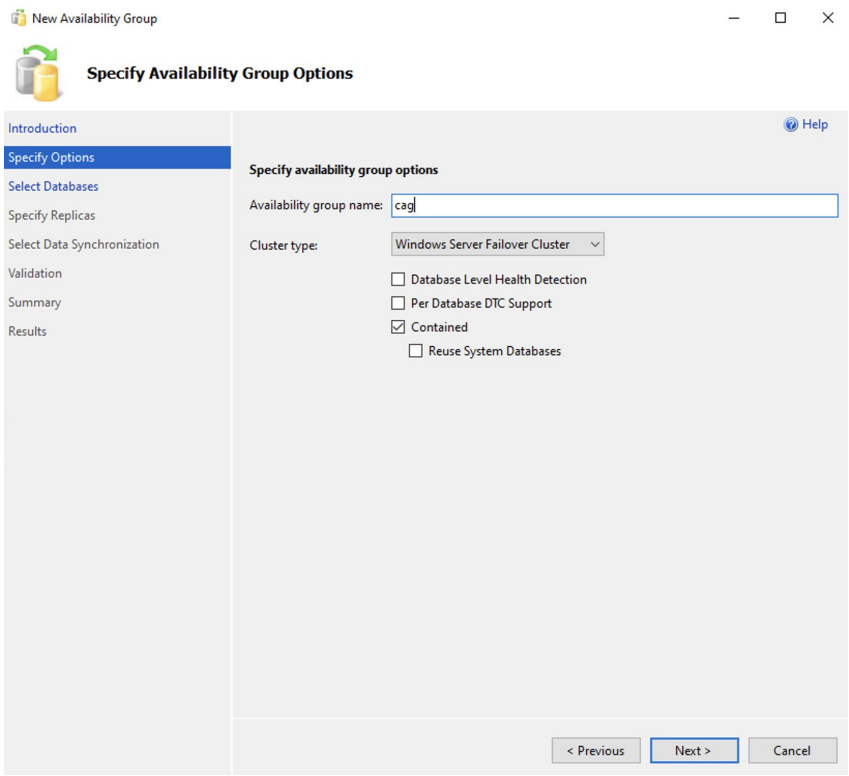
	AgName	NodeId	Role	ConnectState	SyncHealth	OperatState	DbCount	SyncDBs	NotSyncDBs	MaxLatencySec	SyncMode	FailoverMode	BackupPriority	ReadableSecondary
1	CAG	4	PRIMARY	CONNECTED	HEALTHY	ONLINE	3	3	0	NULL	SYNCHRONOUS_COMMIT	AUTOMATIC	50	NO
2	CAG	3	SECONDARY	CONNECTED	HEALTHY	NULL	3	3	0	2575	SYNCHRONOUS_COMMIT	AUTOMATIC	50	NO

	AgName	NodeId	Role	Database	SyncState	DBHealth	IsSuspended	SuspendReason	LastHardenedTime	LastCommitTime	SendQueueKB	SendRateKbSec	RedoQueueKB	RedoRateKbSec	LatencySec	FailoverReadiness
1	CAG	4	PRIMARY	CAG_master	SYNCHRONIZED	HEALTHY	0	N/A	NULL	2025-08-13 07:19:48.080	NULL	NULL	NULL	NULL	NULL	
2	CAG	4	PRIMARY	CAG_msdb	SYNCHRONIZED	HEALTHY	0	N/A	NULL	2025-08-13 07:19:48.303	NULL	NULL	NULL	NULL	NULL	
3	CAG	4	PRIMARY	DemoDB_1	SYNCHRONIZED	HEALTHY	0	N/A	NULL	2025-08-13 07:24:49.443	NULL	NULL	NULL	NULL	NULL	
4	CAG	3	SECONDARY	CAG_master	SYNCHRONIZED	HEALTHY	0	N/A	2025-08-13 07:19:57.043	2025-08-13 07:19:48.080	0	0	0	25333	2575	Failover Ready
5	CAG	3	SECONDARY	CAG_msdb	SYNCHRONIZED	HEALTHY	0	N/A	2025-08-13 07:19:57.123	2025-08-13 07:19:48.303	0	0	0	16000	2575	Failover Ready
6	CAG	3	SECONDARY	DemoDB_1	SYNCHRONIZED	HEALTHY	0	N/A	2025-08-13 07:24:49.447	2025-08-13 07:24:49.443	60	40000	0	17600	2283	Failover Ready

How to Implement a Contained AG??

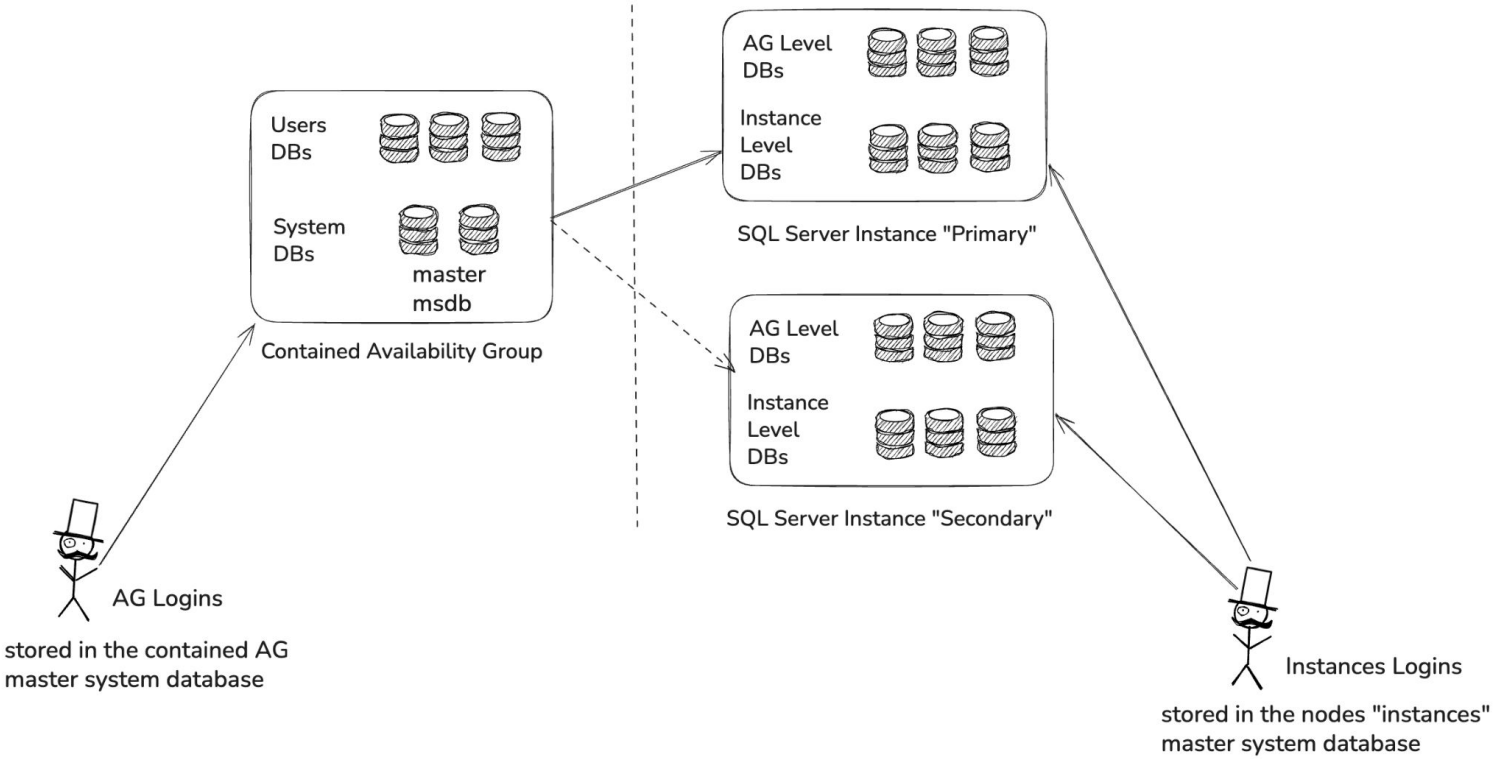


How to Create a Contained AG



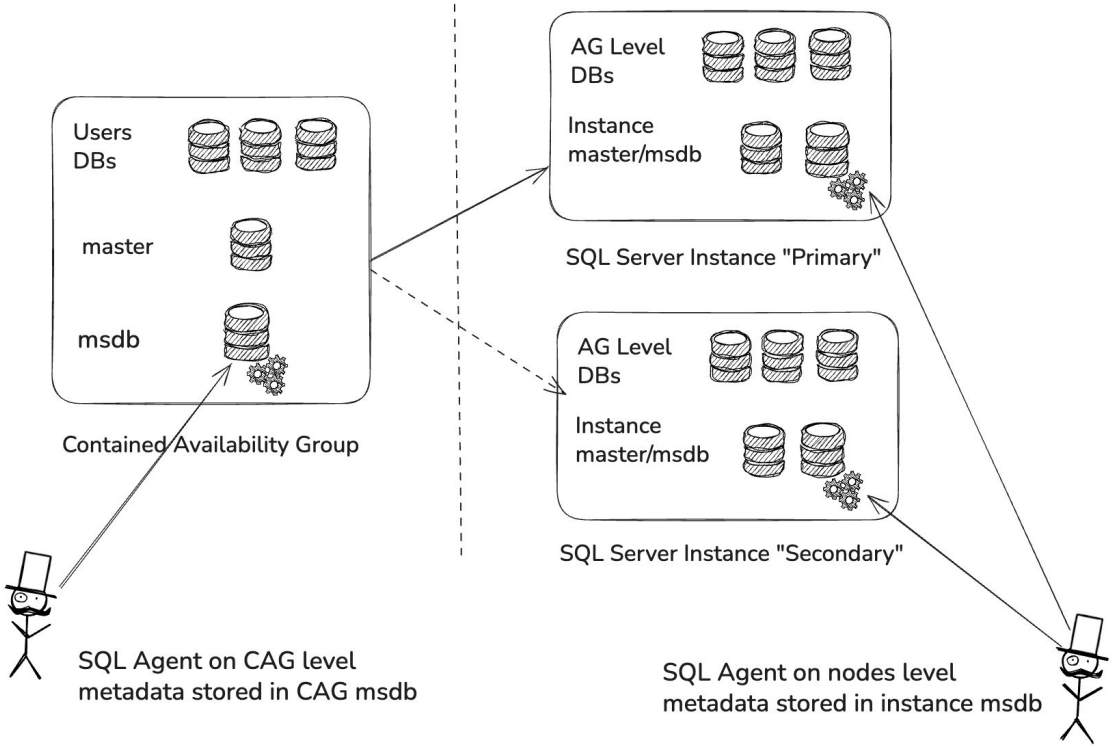

```
/* create contained availability group with 2 replicas */
CREATE AVAILABILITY GROUP [cag]
WITH (
   CLUSTER_TYPE = WSFC,
   CONTAINED = ON
)
FOR REPLICA ON
   N'sql-node3' WITH (
      ENDPOINT_URL = N'tcp://sql-node3:5022',
      AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
      FAILOVER_MODE = AUTOMATIC,
      SEEDING_MODE = AUTOMATIC
   ),
   N'sql-node4' WITH (
      ENDPOINT_URL = N'tcp://sql-node4:5022',
      AVAILABILITY_MODE = SYNCHRONOUS_COMMIT,
      FAILOVER_MODE = AUTOMATIC,
      SEEDING_MODE = AUTOMATIC
   );
GO
/* grant permission for auto seeding */
ALTER AVAILABILITY GROUP [cag] GRANT CREATE ANY DATABASE;
GO
```

Contained AG Logins vs. AG Node’s Logins



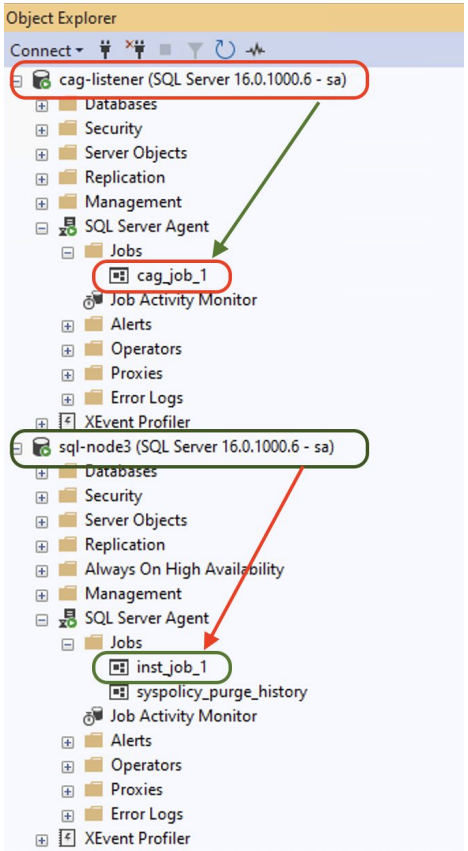
Contained AG Logins vs. AG Node’s Logins

Contained AG Agent vs. AG Node’s Agent



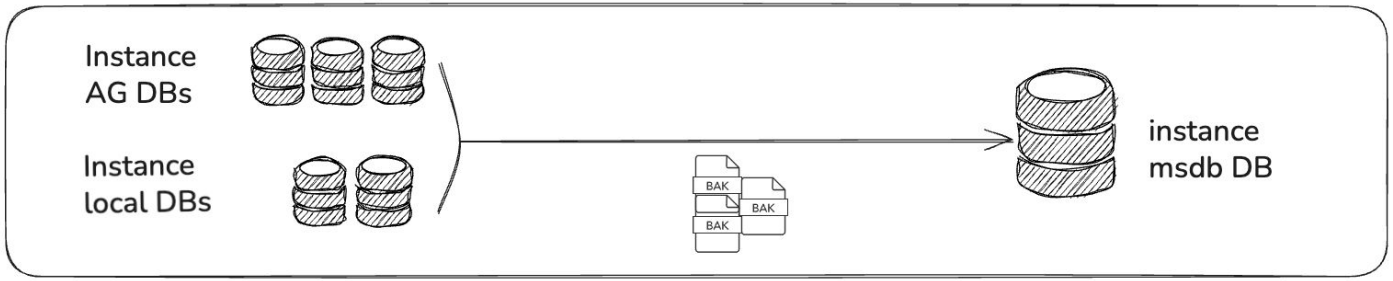
Contained AG Agent vs. AG Node’s Agent

Contained AG Agent vs. AG Node's Agent (cont.)

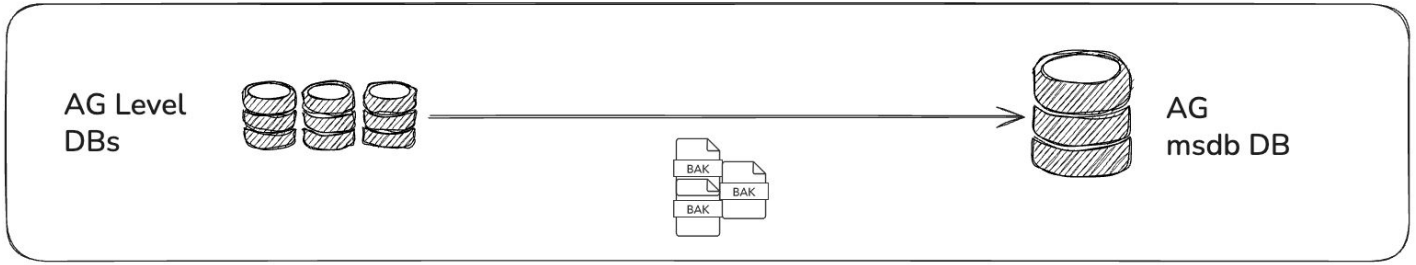


- Contained AG has its own SQL Agent.
- Agent metadata is stored in contained AG msdb.
- Jobs are automatically synchronized as a core concept.
- Separated from SQL Agent on AG nodes level.
- Some limitations/bugs for applications and non-sysadmin logins "a common use case for contained AGs".

Backups in Contained AG



SQL Server Instance
"Primary/Secondary"



Contained Availability Group

Sysadmins in Contained AGs

- sa login is there, mirrored from the local primary node sa during AG creation.

- Creating sysadmin logins on contained AG level is possible but risky!!

- Any login created on contained AG level with sysadmin permission will have the possibility to connect to instance level databases and perform administrative operations there!!

- While instance-level databases are not listed in the Object Explorer or by querying sys.databases catalog view when connected to the listener, a sysadmin can still connect to them directly if the database name is known.

- It is a best practice to disable sa and deny sysdmin permission from created logins on contained AG level.

- This also complies with the idea of having all necessary DBA tasks done on instance level not on the contained AG level.

Supported Features in Contained AGs

- Linked Servers.
- SQL Server Agent.
- Change Data Capture.
- Contained Databases.
- Backup operations from AG level.
- Transparent Data Encryption “TDE”.
- Distributed Availability Groups “starting from SQL Server 2025”.
- Log Shipping “only as a primary role, the same like traditional AGs”.

Unsupported Features in Contained AGs

- Database creation is not supported from contained AG level, it is possible only from primary node level following the default process of creating and adding databases into availability groups.

- The same goes by default to database restoration, it is also not possible to be done by nature on contained AG level, and needs to follow the same process of database creation by restoring databases on primary node then adding them to availability groups.

- Server configuration is read only “by design”.

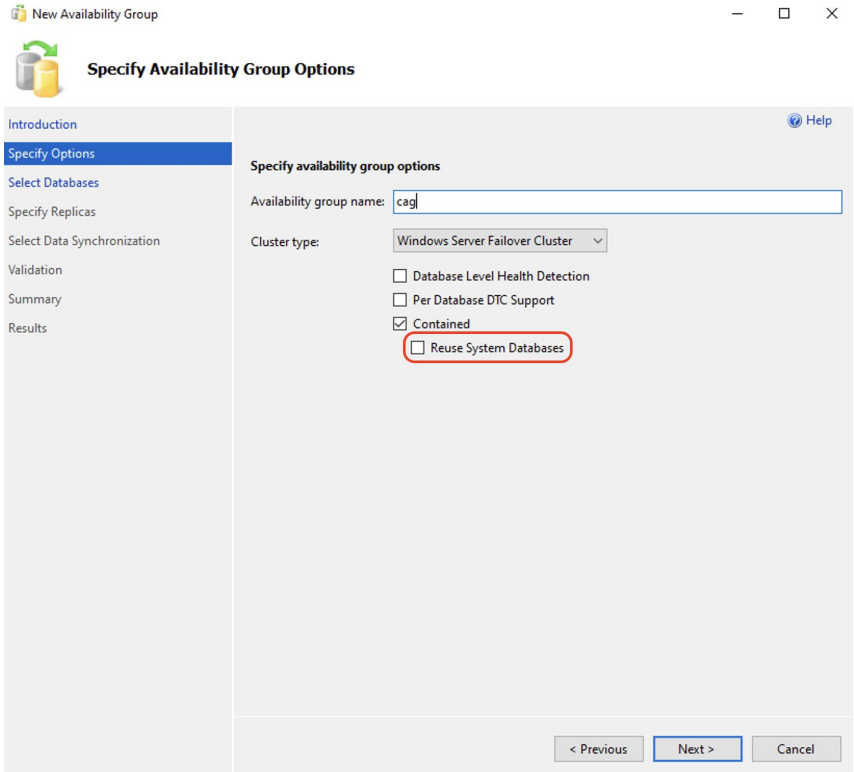
- SQL Agent jobs cannot be executed from secondaries “ag_msdb is a read only DB, and this is also by design”.

- Some limitations like database creation and restoration can have some workarounds!!

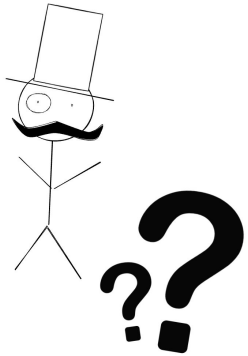
Recovering a Contained AG from a Disaster

- Contained system databases can be normally recovered like any other databases in contained AG → restore on instance level primary node then added to AG.

- Also, contained AGs can reuse existing contained system databases → by using option REUSE_SYSTEM_DATABSES parameter during AG creation process.



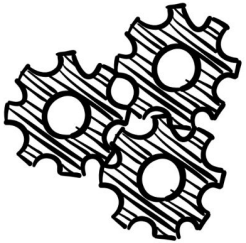
When to use Contained AGs??



When to Use Contained Availability Groups

- As a common practice for avoiding the AG objects dependency problem.
- A good option for delivering managed services due to its abstracted layer.
- Also a good option for a multi-tenant environment.
- For workloads that are compatible with its current feature set and limitations.
  - There is no news so far for addressing such limitations in the near future as they are still persist in SQL Server 2025 preview.

Demo



Questions