Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

In SQL Server, **Distributed Availability Groups** (Distributed AGs) provide an advanced high-availability and disaster recovery (HADR) solution that allows you to extend the capabilities of **Always On Availability Groups (AGs)** across multiple **SQL Server instances** and **availability groups**, including across different **SQL Server versions** or **geographies**.

A **Distributed Availability Group** is a logical extension of the **Always On Availability Group** feature and enables a **two-tier** architecture for SQL Server replication. This architecture consists of:

1. A **primary Availability Group** (AG) at the source.
2. A **secondary Availability Group** (AG) at the target.

This feature allows the databases in the **primary AG** to be replicated and synchronized with the databases in the **secondary AG**, enabling advanced configurations like cross-geo replication and cross-version SQL Server support.

Key Concepts of Distributed Availability Groups

1. **Two-Tier Replication Architecture**:
    - The architecture involves **two separate availability groups**:
        - **Primary AG**: The availability group that is the source for replication (the "primary" replica).
        - **Secondary AG**: The target availability group where the databases from the primary AG will be replicated (the "secondary" replica).
    - This two-tier architecture allows you to scale out replication across regions, data centers, or even different versions of SQL Server.
2. **Distributed AG Components**:
    - **Primary AG**: Contains the databases that will be replicated to a secondary AG.
    - **Secondary AG**: Receives replicated data from the primary AG and can itself have multiple replicas (but these replicas are part of a different AG).
    - **Listener**: A listener for the Distributed AG can be used to provide a single point of access to the primary replica in the primary AG.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

3. **Transactional Replication**:
    - Distributed AGs rely on **transactional replication** to move data between the primary AG and the secondary AG.
    - The **primary AG** sends its logs to the **secondary AG**, which must be on a different server and typically located in a separate site or geographical region.
4. **Cross-Version Support**:
    - A key benefit of Distributed AGs is that they allow you to use **different versions** of SQL Server between the primary and secondary AGs. This is particularly useful when upgrading from older versions of SQL Server to a newer version without requiring a full migration of all replicas at once.
5. **Replication Across Regions**:
    - With Distributed AGs, you can create availability groups across geographically dispersed data centers or Azure regions, providing greater disaster recovery capabilities.
    - This feature allows **cross-region replication** for scenarios where you have to replicate your database across distant locations (e.g., North America to Europe).
6. **Failover and Recovery**:
    - Failovers can occur within a single AG (within the primary AG or within the secondary AG), but **Distributed AGs** focus on extending the **availability** of the primary AG to a remote AG.
    - A failover in the primary AG **does not automatically failover** the secondary AG. Failover of the secondary AG is independent and must be handled manually, unless configured otherwise.

How Distributed Availability Groups Work

1. **Primary Availability Group (Tier 1)**:
    - The **primary availability group** can be a typical AG configuration with synchronous or asynchronous commit replicas.
    - The **primary AG** will handle all the writes and transaction log generation.
2. **Secondary Availability Group (Tier 2)**:
    - The **secondary availability group** receives **log shipping** from the primary AG.
    - The data replicated to the secondary AG may be in a **different geographic location** or on a different version of SQL Server.
3. **Log Shipping Between AGs**:
    - The log shipping is handled using **transactional replication**. The transaction logs from the primary AG are replicated to the secondary AG.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

- The secondary AG does not need to be synchronous with the primary AG, which allows for more flexibility, especially in geographically dispersed environments.

4. **Replication Flow**:
   - **Primary AG**: The source AG that generates transaction logs as part of the SQL Server's normal database activity.
   - **Replication to Secondary AG**: The transaction logs are sent to the secondary AG, which applies the logs to its own databases.
   - **Data Synchronization**: Data between the primary and secondary AGs is synchronized via log shipping, allowing the secondary AG to have an almost identical copy of the primary AG databases.

Benefits of Distributed Availability Groups

1. **Geographical Disaster Recovery**:
   - By replicating the primary AG to a secondary AG that is located in a different geographical region, Distributed AGs provide an additional layer of disaster recovery. If the primary region experiences failure (e.g., due to network issues or a data center outage), the secondary AG can take over, ensuring the business continuity of your applications.

2. **Cross-Version Support**:
   - One of the standout features of Distributed AGs is **cross-version support**. You can use SQL Server 2016 on the primary AG and SQL Server 2019 on the secondary AG, allowing for easier migration and version upgrades.
3. **Scaling Out Disaster Recovery**:
   - By enabling multiple secondary AGs across different regions or environments, organizations can implement **multi-tier disaster recovery** strategies where multiple levels of redundancy exist for high availability.
4. **Upgrade Strategy**:
   - Distributed AGs allow for rolling upgrades of SQL Server versions. The primary AG can run on an older version of SQL Server, while the secondary AG can run on a newer version. This allows for seamless version migration with minimal downtime.
5. **Offload Reporting/Analytics**:
   - The secondary AG can be used for offloading read-only workloads such as reporting or analytics, allowing the primary AG to handle transactional workloads. In some cases, you can offload such workloads to a separate data center or region entirely.

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

Setting Up a Distributed Availability Group

Here's a basic overview of setting up a Distributed Availability Group in SQL Server:

Step 1: Create the Primary Availability Group

1. Set up the **primary availability group** as you would for a typical Always On AG. This involves configuring a WSFC cluster, creating the availability group, and adding replicas.

```
CREATE AVAILABILITY GROUP [PrimaryAG]

FOR DATABASE [YourDatabase]

REPLICA ON

    N'PrimaryReplica1' WITH (ENDPOINT_URL = 'TCP://PrimaryReplica1.domain.com:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = AUTOMATIC),

    N'PrimaryReplica2' WITH (ENDPOINT_URL = 'TCP://PrimaryReplica2.domain.com:5022', AVAILABILITY_MODE = ASYNCHRONOUS_COMMIT, FAILOVER_MODE = MANUAL);
```

Step 2: Create the Secondary Availability Group

2. Set up the **secondary availability group** on a different SQL Server instance or data center, ensuring that the databases are in a **restoring state**.

Step 3: Create the Distributed Availability Group

3. Once both the primary and secondary AGs are created, you can configure the **Distributed Availability Group** using **T-SQL**.
   Example:

-- Create Distributed Availability Group

```
CREATE AVAILABILITY GROUP [DistributedAG]

    FOR DATABASE [YourDatabase]

    REPLICA ON

        N'PrimaryReplica1' WITH (ENDPOINT_URL = 'TCP://PrimaryReplica1.domain.com:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = AUTOMATIC)

    SECONDARY AG

        N'SecondaryReplica1' WITH (ENDPOINT_URL = 'TCP://SecondaryReplica1.domain.com:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = AUTOMATIC);
```

Praveen Madupu +91 98661 30093
Sr SQL Server DBA, Dubai
praveensqldba12@gmail.com

Step 4: Test and Monitor

- After setting up the **Distributed AG**, you should test the failover and failback process and monitor replication health between the primary and secondary availability groups.

Step 5: Set Up the Listener

- Configure an **Availability Group Listener** for the Distributed AG, which provides a single connection point for client applications.
  Example:

```
ALTER AVAILABILITY GROUP [DistributedAG]

ADD LISTENER 'AGListener' (WITH IP ((10.0.0.100, 255.255.255.0)), PORT = 1433);
```

Considerations

- **Replication Latency**: Ensure that the network between the primary and secondary AG is robust to handle the replication traffic, especially in cross-geo scenarios where latency might be a concern.
- **Failover Handling**: Understand that failovers between the primary AG and the secondary AG are independent, meaning that a failover on the primary AG does not trigger a failover on the secondary AG. Manual intervention may be required for failover operations.
- **Backup Strategy**: A solid backup strategy should be implemented across both AGs, as well as ensuring that the data replicated to the secondary AG is properly backed up for disaster recovery scenarios.

Summary:

Distributed Availability Groups in SQL Server allow you to extend the capabilities of Always On Availability Groups to support complex, cross-region, and cross-version configurations.

This feature is ideal for organizations looking to scale their disaster recovery solutions, upgrade SQL Server versions with minimal downtime, and ensure the continuity of business operations across geographically distributed data centers.

However, configuring and managing Distributed AGs requires careful planning, particularly in terms of network configuration, failover strategies, and replication monitoring.