AlwaysOn Availability Groups
Failover Best Practices

Choirul Amri
Choirul @ gmail.com
About Me

- Microsoft MVP SQL (2005 – 2011)
- Microsoft: Senior Consultant (2011 – 2014)

Founder: SQL Server User Group Indonesia
- [https://groups.yahoo.com/neo/groups/sqlserver-indo/info](https://groups.yahoo.com/neo/groups/sqlserver-indo/info)

Co-Founder: IlmuKomputer.Com

Ebiz Cipta Solusi: VP Enterprise Division
Session Objectives

• Key Takeaway
  - Understanding SQL Server Cluster Vs Availability Groups
  - Design decision – best practices
  - Failover process and demo
  - Focus on failover, not configuration

• What’s not in session
  - Failover clustering details (unless you ask 😊)
  - Active Secondary and routing
  - Basic availability groups
What’s the “Groups” mean?

- Failover happens on “group of databases” rather than individual DBs.
- You can also create a group with only one database.
- Why: You want to make sure that related DBs are failed-over together.
## AlwaysOn Landscape

<table>
<thead>
<tr>
<th>Failover Cluster Instance (FCI)</th>
<th>Availability Groups (AG)</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Instance level</td>
<td>• Database level</td>
</tr>
<tr>
<td>• <strong>Shared</strong> storage</td>
<td>• No shared storage</td>
</tr>
<tr>
<td>• Failover per instance</td>
<td>• Failover per DB group</td>
</tr>
<tr>
<td>• No active secondary</td>
<td>• Active secondary (read only)</td>
</tr>
<tr>
<td>• Support auto, transparent failover</td>
<td>• Support auto, transparent failover</td>
</tr>
</tbody>
</table>
Availability Groups – Common Implementation

Windows Server Failover Cluster

KOMODOSQL1 - Primary Server
- Ticketing
  - Primary Replica
- CustomerDB
  - Primary Replica

KOMODOSQL2 - Secondary Server
- Ticketing
  - Secondary Replica
- CustomerDB
  - Secondary Replica
- Readable Secondary

KOMODOSQL3 - Secondary Server
- Ticketing
  - Secondary Replica
- CustomerDB
  - Secondary Replica
- Non-Readable (standby only)

Asynchronous

Synchronous

Availability Listener
SQLAG-LS
Availability Groups Advantage

• No shared storage, easier and faster to implement
• Active secondary for better box utilization
  - Offload reporting to secondary
  - Backup from secondary
• Support automatic failover
• Transparent client failover
Design Decision: FCI Vs AG

Failover Cluster Instance (FCI)
- Supporting “Legacy” apps
- Upgrading from older cluster
- Limited DBA resources
- Shared storage is available

Availability Groups (AG)
- Apps with AG support
- Limited shared storage availability
- Isolate failover per DB
- Separate reporting is required
Availability Groups – Availability Mode

Asynchronous
- Better performance
- Primary commits without waiting notification from secondary
- Possible data loss in failover

Synchronous
- Primary waits for notification from secondary before committing transaction
- Performance penalty
Design Decision: Synchronous Vs Asynchronous

**Synchronous**

- "Good" bandwidth is available. Start with latency 5-10ms
- Machines in the same DC
- **Automatic failover** is "absolutely" required
- Necessity instead of desire

**Asynchronous**

- **DR scenario**: manual failover is a "must"
- If network/bandwith not "enough"
- You can "afford" to have data loss
Failover Mode

**Automatic**
- Require **Synchronous** mode
- No data loss
- Machine shutdown or service unavailable

**Planned - Manual**
- Require **Synchronous** mode
- No data loss
- Use SSMS or T-SQL

**Forced - Manual**
- Replicas are in **asynchronous** commit mode
- Forced means: You can afford **loosing** data
- Obviously, data loss is possible
Demo
Planned – Manual Failover
Automatic Failover - Requirement

Requirement

- **Minimum 1 pair** databases are in Synchronous-commit with Automatic failover
- Secondary databases are in synchronized state
- Windows Cluster is in good condition and fulfill the quorum requirement
Automatic Failover – Ready to failover

KomodoSQLAG: hosted by KOMODOSQL1\SQL2014 (Replica role: Primary)

<table>
<thead>
<tr>
<th>Availability group state:</th>
<th>Healthy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Primary instance:</td>
<td>KOMODOSQL1\SQL2014</td>
</tr>
<tr>
<td>Failover mode:</td>
<td>Automatic</td>
</tr>
<tr>
<td>Cluster state:</td>
<td>WIN2012-SQLAG (Normal Quorum)</td>
</tr>
</tbody>
</table>

**Availability replica:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Role</th>
<th>Failover Mode</th>
<th>Synchronization State</th>
<th>Issues</th>
</tr>
</thead>
<tbody>
<tr>
<td>KOMODOSQL1\SQL2014</td>
<td>Primary</td>
<td>Automatic</td>
<td>Synchronized</td>
<td></td>
</tr>
<tr>
<td>KOMODOSQL2\SQL2014</td>
<td>Secondary</td>
<td>Automatic</td>
<td>Synchronized</td>
<td></td>
</tr>
<tr>
<td>KOMODOSQL3\SQL2014</td>
<td>Secondary</td>
<td>Manual</td>
<td>Synchronized</td>
<td></td>
</tr>
</tbody>
</table>

**Replica Group:**

<table>
<thead>
<tr>
<th>Name</th>
<th>Replica</th>
<th>Synchronization State</th>
<th>Failover Ready...</th>
</tr>
</thead>
<tbody>
<tr>
<td>KOMODOSQL1\SQL2014</td>
<td>AdventureWorks2014</td>
<td>Synchronized</td>
<td>No Data Loss</td>
</tr>
<tr>
<td>KOMODOSQL1\SQL2014</td>
<td>AdventureWorks2014</td>
<td>Synchronized</td>
<td>No Data Loss</td>
</tr>
<tr>
<td>KOMODOSQL3\SQL2014</td>
<td>AdventureWorks2014</td>
<td>Synchronized</td>
<td>No Data Loss</td>
</tr>
</tbody>
</table>
Demo Automatic Failover
Real life decision: Which failover to use

**Automatic**
- Servers in the same data center with low latency network
- Unexpected machine crash or SQL service disruption
- Happens in the background without notice

**Planned - Manual**
- Patching OS or SQL Server
- HA/DR exercise
- Upgrading hardware

**Forced - Manual**
- Real disaster happens – failover to DR site
- 3 servers, but 2 machines in primary DC are down
- There is no primary available in the configuration!
Forced Manual Failover – Fixing “Resolving” condition

<table>
<thead>
<tr>
<th>Server Instance</th>
<th>Availability Mode</th>
<th>Failover Mode</th>
<th>Role</th>
</tr>
</thead>
<tbody>
<tr>
<td>KOMODOSQL1\SQL2014</td>
<td>Synchronous commit</td>
<td>Automatic</td>
<td>Primary</td>
</tr>
<tr>
<td>KOMODOSQL2\SQL2014</td>
<td>Synchronous commit</td>
<td>Automatic</td>
<td>Secondary</td>
</tr>
<tr>
<td>KOMODOSQL3\SQL2014</td>
<td>Asynchronous commit</td>
<td>Manual</td>
<td>Secondary</td>
</tr>
</tbody>
</table>

What happen if:

1) KOMODOSQL2\SQL2014 is down; then
2) Client still can access the primary as usual (but now there is no “automatic” pair)
3) KOMODOSQL1\SQL2014 is down
4) Automatic failover to KOMODOSQL3\SQL2014 will not happen!
5) DBs in KOMODOSQL3\SQL2014 replica will be in “resolving” mode
6) Client application cannot connect
Recovering the “last man standing” node

Your situation:

- **Primary** and **secondary** in production are **lost**
- You only have 1 node in DR, but it is in resolving state, inaccessible from client
Bring Your "resolving" replica alive

--- YOU MUST EXECUTE THE FOLLOWING SCRIPT IN SQLCMD MODE

--- KOMODOSQL3 is the replica in resolving state

:Connect KOMODOSQL3\SQL2014

ALTER AVAILABILITY GROUP [KomodoSQLAG]
FORCE_FAILOVER_ALLOW_DATA_LOSS;
Demo
Recovering from “Resolving” state
Recommended reference

Download my e-book 😊

SQL Server AlwaysOn Availability Groups
Perencanaan, Konfigurasi, dan Optimasi


https://leanpub.com/sqlagindo

Work in progress, If You find typos please @mention me at @mchoirul