Sync Logins in SQL AlwaysOn Replicas-Manually – Author: DANIEL HUTMACHER

Source: https://sqlsunday.com/2016/10/11/how-to-sync-logins-between-availability-group-replicas/

https://raw.githubusercontent.com/strdco/ag-sync/boss/SyncLogins.sql

1. Execute the below SP on all SQL AG Replica's.

```
USE master;
GO
IF (OBJECT_ID('dbo.SyncLogins') IS NULL) EXEC('CREATE PROCEDURE dbo.SyncLogins AS SELECT 0;');
GO
/*

Copyright Daniel Hutmacher under Creative Commons 4.0 license with attribution.
http://creativecommons.org/licenses/by/4.0/

Source: http://sqlsunday.com/downloads/

DISCLAIMER: This script may not be suitable to run in a production environment. I cannot assume any responsibility regarding
        the accuracy of the output information, performance impacts on your server, or any other consequence. If
        your juristiction does not allow for this kind of waiver/disclaimer, or if you do not accept these terms,
        you are NOT allowed to store, distribute or use this code in any way.
        Please test stuff before you put it in your production environment.

USAGE:
Use master
EXECUTE dbo.SyncLogins
          @primary_replica,              -- {name of SOURCE linked server}
          @allow_drop_logins,                    -- 1=allow script to drop logins
          @print_only,                           -- 1=only print the T-SQL.
          @exclude_logins                        -- comma-separated list of logins to exclude. Example: 'abc,def,DOMAIN\ghi'

TODO:     Owners of logins.
          Permissions on endpoints.
VERSION:   2020-05-15

*/
ALTER PROCEDURE dbo.SyncLogins
          @primary_replica  sysname=NULL,
          @allow_drop_logins          bit=0,
          @print_only                          bit=0,
          @check_policy               bit=0,
          @exclude_logins             nvarchar(max)=NULL
AS

SET NOCOUNT ON;
DECLARE @sql nvarchar(max), @msg nvarchar(max);

--- Thanks, sqlDimsJesper!
IF (@primary_replica NOT IN (SELECT [name] FROM sys.servers)) BEGIN;
  THROW 50000, N'Primary replica is not a linked server.', 16;
  RETURN;
END;


IF (@primary_replica IS NULL) BEGIN;
          SELECT @primary_replica=primary_replica
  FROM sys.dm_hadr_availability_group_states;

          IF (@@ROWCOUNT>1) BEGIN;
                    THROW 50000, N'More than one availability group exists on server, please specify @primary_replica.', 16;
                    RETURN;
```

```
        END;

        IF (@primary_replica IS NULL) BEGIN;
                THROW 50000, N'No availability group found, please specify @primary_replica.', 16;
                RETURN;
        END;
END;

IF (@primary_replica=@@SERVERNAME) BEGIN;
        PRINT N'This server is the primary replica. No changes will be made.';
        RETURN;
END;

--- These are the logins (Windows user and groups, SQL logins) from
--- the primary replica.
DECLARE @primaryLogins TABLE (
        [name]                                  sysname NOT NULL,
        [sid]                                   varbinary(85) NOT NULL,
        [type]                                  char(1) NOT NULL,
        is_disabled                             bit NULL,
        default_database_name   sysname NULL,
        default_language_name   sysname NULL,
        is_policy_checked               bit NULL,
        is_expiration_checked           bit NULL,
        password_hash                           varbinary(256) NULL,
        PRIMARY KEY CLUSTERED ([sid])
);

SET @sql=N'
SELECT sp.[name], sp.[sid], sp.[type], sp.is_disabled, sp.default_database_name,
    sp.default_language_name, l.is_policy_checked, l.is_expiration_checked, l.password_hash
FROM ['+@primary_replica+'].master.sys.server_principals AS sp
LEFT JOIN ['+@primary_replica+'].master.sys.sql_logins AS l ON sp.[sid]=l.[sid]
WHERE sp.[type] IN (''U'', ''G'', ''S'') AND
   UPPER(sp.[name]) NOT LIKE ''NT SERVICE\%'' AND
        UPPER(sp.[name]) NOT LIKE ''##%##'' AND'+ISNULL('
        UPPER(sp.[name]) NOT IN ('''+REPLACE(NULLIF(@exclude_logins, ''), ',', ''',''')+''') AND', '')+'
        sp.[name] NOT IN (''NT AUTHORITY\SYSTEM'')';

INSERT INTO @primaryLogins
EXECUTE master.sys.sp_executesql @sql;


--- These are the server roles on the primary replica.
DECLARE @primaryRoles TABLE (
        [sid]                                   varbinary(85) NOT NULL,
        [name]                                  sysname NOT NULL,
        PRIMARY KEY CLUSTERED ([sid])
);

SET @sql=N'
SELECT sr.[sid], sr.[name]
FROM ['+@primary_replica+'].master.sys.server_principals AS sr
WHERE sr.is_fixed_role=0 AND
   sr.[type]=''R'';

INSERT INTO @primaryRoles
EXECUTE master.sys.sp_executesql @sql;


--- These are the role members of the server roles on
--- the primary replica.
DECLARE @primaryMembers TABLE (
        role_sid                        varbinary(85) NOT NULL,
        member_sid                              varbinary(85) NOT NULL,
```

```
        PRIMARY KEY CLUSTERED (role_sid, member_sid)
);

SET @sql=N'
SELECT r.[sid], m.[sid]
FROM ['+@primary_replica+N'].master.sys.server_principals AS r
INNER JOIN ['+@primary_replica+N'].master.sys.server_role_members AS rm ON r.principal_id=rm.role_principal_id
INNER JOIN ['+@primary_replica+N'].master.sys.server_principals AS m ON rm.member_principal_id=m.principal_id';

INSERT INTO @primaryMembers
EXECUTE master.sys.sp_executesql @sql;


--- These are the server-level permissions on the
--- primary replica.
DECLARE @primaryPermissions TABLE (
        state_desc                      nvarchar(120) NOT NULL,
        [permission_name] nvarchar(256) NOT NULL,
        principal_name          sysname NOT NULL,
        PRIMARY KEY CLUSTERED ([permission_name], principal_name)
);

SET @sql=N'
SELECT p.state_desc, p.[permission_name], sp.[name]
FROM ['+@primary_replica+'].master.sys.server_permissions AS p
INNER JOIN ['+@primary_replica+'].master.sys.server_principals AS sp ON p.grantee_principal_id=sp.principal_id
WHERE p.class=100';

INSERT INTO @primaryPermissions
EXECUTE master.sys.sp_executesql @sql;


--- This table variable contains the "run queue" of all commands
--- we want to execute on the local (secondary) replica, ordered
--- by "seq".
DECLARE @queue TABLE (
        seq                                     int IDENTITY(1, 1) NOT NULL,
        [sql]                           nvarchar(max) NOT NULL,
        PRIMARY KEY CLUSTERED (seq)
);
--------------------------------------------------------------------------------
--- Login doesn't exist on the primary - DROP.
INSERT INTO @queue ([sql])
SELECT N'
        DROP LOGIN ['+sp.[name]+N'];'
FROM master.sys.server_principals AS sp
WHERE sp.[type] IN ('U', 'G', 'S') AND
    UPPER(sp.[name]) NOT LIKE 'NT SERVICE\%' AND
        sp.[name] NOT IN ('NT AUTHORITY\SYSTEM') AND
        sp.[sid] NOT IN (SELECT [sid] FROM @primaryLogins) AND
        @allow_drop_logins=1;


--- Login doesn't exist on the secondary - CREATE.
INSERT INTO @queue ([sql])
SELECT N'
        CREATE LOGIN ['+p.[name]+'] '+(CASE
                        WHEN p.[type]='S'
                        THEN N'WITH PASSWORD=0x'+CONVERT(nvarchar(max), p.password_hash, 2)+N' HASHED, CHECK_POLICY='+(CASE
WHEN @check_policy=1 THEN N'ON' ELSE N'OFF' END)+N', SID=0x'+CONVERT(nvarchar(max), p.[sid], 2)+N', '
                        WHEN p.[type] IN ('U', 'G')
                        THEN N'FROM WINDOWS WITH ' END)+
                N'DEFAULT_DATABASE=['+p.default_database_name+N']'+
                ISNULL(N', DEFAULT_LANGUAGE='+p.default_language_name, N'')+N';'
FROM @primaryLogins AS p
```

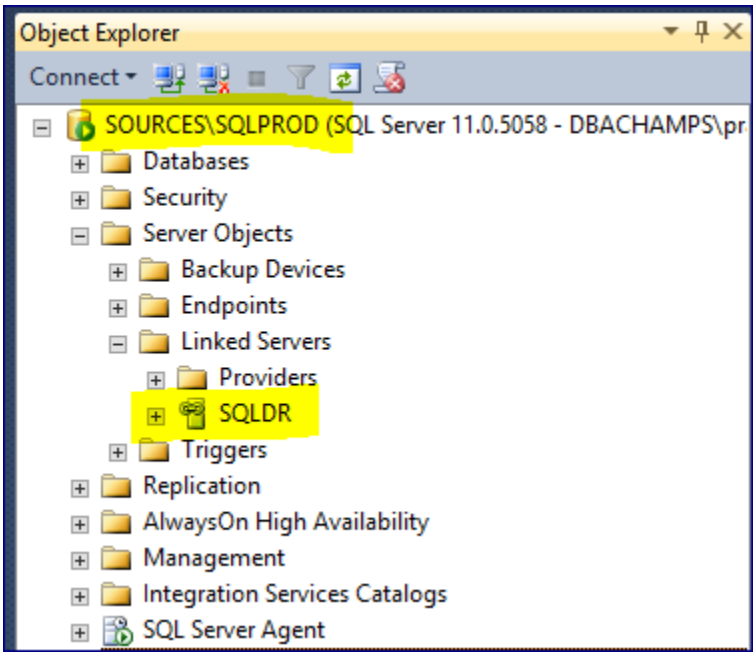
```
WHERE p.[sid] NOT IN (SELECT [sid] FROM master.sys.server_principals) AND
            p.[type] IN ('U', 'G', 'S');


--- Login exists but has been enabled/disabled - ALTER.
INSERT INTO @queue ([sql])
SELECT N'
        ALTER LOGIN ['+ISNULL(sp.[name], x.[name])+']'+
                (CASE WHEN x.is_disabled=0 AND  sp.is_disabled=1 THEN N' ENABLE'
                    WHEN x.is_disabled=1 AND (sp.is_disabled=0 OR sp.[sid] IS NULL) THEN N' DISABLE' END)+N';'
FROM @primaryLogins AS x
LEFT JOIN master.sys.server_principals AS sp ON x.[sid]=sp.[sid]
WHERE x.is_disabled!=sp.is_disabled OR
    x.is_disabled=1 AND sp.[sid] IS NULL;


--- Login exists but has changed in some respect - ALTER.
INSERT INTO @queue ([sql])
SELECT N'
        ALTER LOGIN ['+sp.[name]+'] WITH '+
                SUBSTRING(
                (CASE WHEN x.password_hash!=l.password_hash
                    THEN N', PASSWORD=0x'+CONVERT(nvarchar(max), x.password_hash, 2)+N' HASHED, CHECK_POLICY=OFF'
                        ELSE N'' END)+
                (CASE WHEN ISNULL(x.default_database_name, N'master')!=ISNULL(sp.default_database_name, N'master')
                    THEN ', DEFAULT_DATABASE=['+x.default_database_name+N']'
                        ELSE N'' END)+
                (CASE WHEN x.default_language_name!=sp.default_language_name
                    THEN ', DEFAULT_LANGUAGE='+x.default_language_name
                        ELSE N'' END)+
                (CASE WHEN x.[name]!=sp.[name]
                    THEN ', NAME=['+x.[name]+N']'
                        ELSE N'' END)+
                (CASE WHEN x.is_policy_checked!=l.is_policy_checked
                    THEN ', CHECK_POLICY='+(CASE x.is_policy_checked WHEN 1 THEN N'ON' ELSE N'OFF' END)
                        ELSE N'' END)+
                (CASE WHEN x.is_expiration_checked!=l.is_expiration_checked
                    THEN ', CHECK_EXPIRATION='+(CASE x.is_expiration_checked WHEN 1 THEN N'ON' ELSE N'OFF' END)
                        ELSE N'' END), 3, 10000)+N';'
FROM @primaryLogins AS x
INNER JOIN master.sys.server_principals AS sp ON x.[sid]=sp.[sid]
LEFT JOIN master.sys.sql_logins AS l ON sp.[sid]=l.[sid]
WHERE x.password_hash!=l.password_hash OR
            ISNULL(x.default_database_name, N'master')!=ISNULL(sp.default_database_name, N'master') OR
            ISNULL(x.default_language_name, N'us_english')!=ISNULL(sp.default_language_name, N'us_english') OR
            x.[name]!=sp.[name] OR
            ISNULL(x.is_policy_checked, 0)!=ISNULL(l.is_policy_checked, 0) OR
            ISNULL(x.is_expiration_checked, 0)!=ISNULL(l.is_expiration_checked, 0);

INSERT INTO @queue ([sql])
SELECT N'
  ALTER LOGIN ['+sp.[name]+'] WITH CHECK_POLICY=ON;'
FROM @primaryLogins AS x
INNER JOIN master.sys.server_principals AS sp ON x.[sid]=sp.[sid]
LEFT JOIN master.sys.sql_logins AS l ON sp.[sid]=l.[sid]
WHERE x.password_hash!=l.password_hash AND
ISNULL(x.is_policy_checked, 0)=1;

INSERT INTO @queue ([sql])
SELECT N'
  ALTER LOGIN ['+sp.[name]+'] WITH CHECK_EXPIRATION=ON;'
FROM @primaryLogins AS x
INNER JOIN master.sys.server_principals AS sp ON x.[sid]=sp.[sid]
LEFT JOIN master.sys.sql_logins AS l ON sp.[sid]=l.[sid]
WHERE x.password_hash!=l.password_hash AND
```

```
ISNULL(x.is_expiration_checked, 0)=1;

--------------------------------------------------------------------------------
--- Roles that don't exist on the primary - DROP.
INSERT INTO @queue ([sql])
SELECT N'
            DROP ROLE ['+sp.[name]+N'];'
FROM master.sys.server_principals AS sp
WHERE is_fixed_role=0 AND
              sp.[type]='R' AND
              sp.[sid] NOT IN (SELECT [sid] FROM @primaryRoles);


--- Roles that don't exist on the secondary - CREATE.
INSERT INTO @queue ([sql])
SELECT N'
            CREATE SERVER ROLE ['+r.[name]+N'];'
FROM @primaryRoles AS r
WHERE [sid] NOT IN (
            SELECT [sid]
            FROM sys.server_principals
            WHERE is_fixed_role=0 AND
                        [type]='R');
--------------------------------------------------------------------------------
--- Revoke role memberships:
INSERT INTO @queue ([sql])
SELECT N'
            ALTER SERVER ROLE ['+r.[name]+N'] DROP MEMBER ['+m.[name]+N'];'
FROM sys.server_role_members AS rm
INNER JOIN sys.server_principals AS r ON r.principal_id=rm.role_principal_id
INNER JOIN sys.server_principals AS m ON m.principal_id=rm.member_principal_id
LEFT JOIN @primaryMembers AS pm ON pm.member_sid=m.[sid] AND pm.role_sid=r.[sid]
WHERE pm.role_sid IS NULL;


--- Add server role memberships:
INSERT INTO @queue ([sql])
SELECT N'
  ALTER SERVER ROLE ['+r.[name]+N'] ADD MEMBER ['+pl.[name]+N'];'
FROM @primaryMembers AS pm
INNER JOIN @primaryLogins AS pl ON pm.member_sid=pl.[sid]
LEFT JOIN sys.server_principals AS r ON pm.role_sid=r.[sid] AND r.[type]='R'
LEFT JOIN sys.server_principals AS m ON pm.member_sid=m.[sid]
LEFT JOIN sys.server_role_members AS rm ON r.principal_id=rm.role_principal_id AND m.principal_id=rm.member_principal_id
WHERE rm.role_principal_id IS NULL;

--------------------------------------------------------------------------------
--- GRANT/DENY server-level permissions:
INSERT INTO @queue ([sql])
SELECT N'
            '+pp.state_desc+N' '+pp.[permission_name]+N' TO ['+pp.principal_name+'];'
FROM @primaryPermissions AS pp
INNER JOIN sys.server_principals AS sp ON pp.principal_name=sp.[name]
LEFT JOIN sys.server_permissions AS p ON
   p.grantee_principal_id=sp.principal_id AND
            p.[permission_name] COLLATE database_default=pp.[permission_name] AND
            p.class=100
WHERE pp.state_desc!=p.state_desc COLLATE database_default;


--------------------------------------------------------------------------------
--- Ready to roll:

SET @sql=N'';
SELECT @sql=@sql+[sql] FROM @queue ORDER BY seq;
```

```
--- @print_only=1: PRINT the queue.
WHILE (@print_only=1 AND @sql!=N'') BEGIN;
        PRINT LEFT(@sql, CHARINDEX(CHAR(13), @sql+CHAR(13))-1);
        SET @sql=SUBSTRING(@sql, CHARINDEX(CHAR(13), @sql+CHAR(13))+2, LEN(@sql));
END;


--- @print_only=0: Execute the queue.
IF (@print_only=0)
        EXECUTE master.sys.sp_executesql @sql;
GO
```

2. Create Linked Servers as follows.

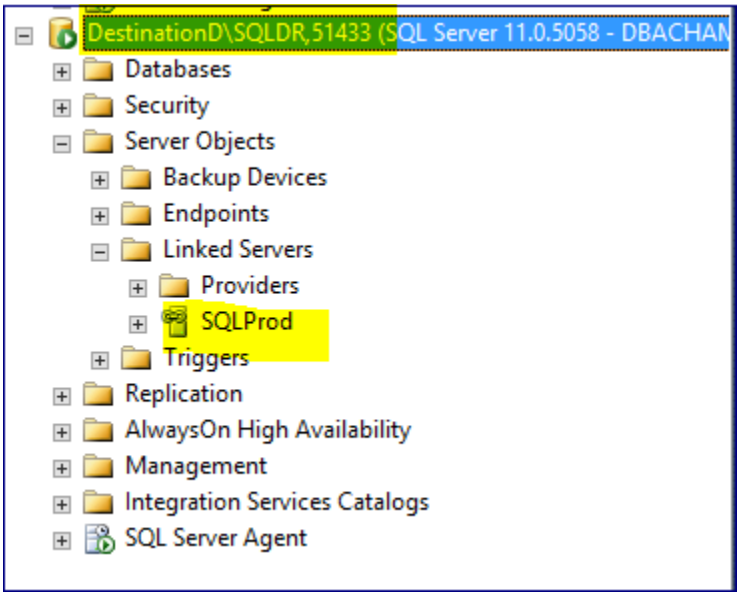
Secondary Replica Linked Server on Primary Replica. (Script given below for example)




```
USE [master]
GO

/****** Object:  LinkedServer [SQLDR]    Script Date: 8/14/2020 9:25:05 PM ******/
EXEC master.dbo.sp_addlinkedserver @server = N'SQLDR', @srvproduct=N'SQLServer', @provider=N'SQLNCLI', @datasrc=N'DestinationD\SQLDR'
 /* For security reasons the linked server remote logins password is changed with ######## */
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname=N'SQLDR',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL

GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'collation compatible', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'data access', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'dist', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'pub', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'rpc', @optvalue=N'true'
```

```
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'rpc out', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'sub', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'connect timeout', @optvalue=N'0'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'collation name', @optvalue=null
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'lazy schema validation', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'query timeout', @optvalue=N'0'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'use remote collation', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLDR', @optname=N'remote proc transaction promotion', @optvalue=N'true'
GO
```

Primary Replica Linked Server on Secondary Replica. (Script given below for example)




```
USE [master]
GO

/****** Object:  LinkedServer [SQLProd]    Script Date: 8/14/2020 9:25:52 PM ******/
EXEC master.dbo.sp_addlinkedserver @server = N'SQLProd', @srvproduct=N'SQLServer', @provider=N'SQLNCLI', @datasrc=N'SourceS\SQLProd'
 /* For security reasons the linked server remote logins password is changed with ######## */
EXEC master.dbo.sp_addlinkedsrvlogin @rmtsrvname=N'SQLProd',@useself=N'True',@locallogin=NULL,@rmtuser=NULL,@rmtpassword=NULL
EXEC master.dbo.sp_addlinkedsrvlogin
@rmtsrvname=N'SQLProd',@useself=N'True',@locallogin=N'DBACHAMPS\SQLService',@rmtuser=NULL,@rmtpassword=NULL

GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'collation compatible', @optvalue=N'false'
GO
```

```
EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'data access', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'dist', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'pub', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'rpc', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'rpc out', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'sub', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'connect timeout', @optvalue=N'0'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'collation name', @optvalue=null
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'lazy schema validation', @optvalue=N'false'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'query timeout', @optvalue=N'0'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'use remote collation', @optvalue=N'true'
GO

EXEC master.dbo.sp_serveroption @server=N'SQLProd', @optname=N'remote proc transaction promotion', @optvalue=N'true'
GO
```

3. Finally, execute the below SP on Secondary Replica to generate missing logins creation script.

```
Use master
EXECUTE dbo.SyncLogins
   @primary_replica = 'SQLProd',       -- {name of SOURCE linked server}
   @allow_drop_logins = 1,             -- 1=allow script to drop logins
   @print_only=1,                       -- 1=only print the T-SQL.
   @check_policy=1,                     -- 1=force check password policy to on
   @exclude_logins = 'sa'
```

Copy the execution output and execute it on Secondary Replica to create the missing logins.