# Automate the Failover of Always on Cluster

To Automate the failover of SQL Server Always on Availability Groups, follow these steps:

1. Prepare the Script

Copy the PowerShell script provided in the next step and save it as a `.ps1` file on your server where SQL Server is installed or on a management workstation that has connectivity to your SQL Server instances.

2. Open PowerShell:

Open PowerShell with administrative privileges on the server or workstation where you saved the script.

3. Set Execution Policy (if needed)

If your system's execution policy doesn't allow running scripts, you might need to set it to allow running scripts. You can set it to RemoteSigned by running the following command:

```

Set-ExecutionPolicy RemoteSigned

```

4. Navigate to the Script Location (if needed):

If you saved the script in a specific directory, navigate to that directory using the `cd` command in PowerShell.

5. Execute the Script:

To execute the script, simply type the name of the script file and press Enter:

```

.\YourScriptName.ps1

```

6. Monitor Execution:

Once the script is running, PowerShell will continuously monitor the health of the primary replica according to the conditions specified in the script. It will automatically initiate a failover if necessary.

7. Exit the Script:

To stop the script, you can press `Ctrl + C` in the PowerShell window where the script is running. This will terminate the script execution.

8. Testing and Validation:

Before deploying the script in a production environment, thoroughly test it in a non-production environment to ensure it behaves as expected and performs failovers correctly.

9. Deployment to Production:

After successful testing, you can deploy the script to your production environment. Make sure to monitor its execution and performance closely after deployment.

1. **PowerShell Script** - In this script, a scheduled failover time is specified (e.g., "02:00" for 2:00 AM). The script continuously checks the current time, and if the current time matches the scheduled failover time and the primary replica is healthy, it initiates the failover process.

Please note that the script assumes that the failover operation completes within a minute. If the failover operation takes longer, you may need to adjust the sleep interval accordingly to prevent multiple executions within the same scheduled time window.

# Define variables

$AGGroupName = "YourAvailabilityGroupName"

$ListenerName = "YourAGListenerName"

$ScheduledFailoverTime = "02:00" # Scheduled failover time in 24-hour format (e.g., "02:00" for 2:00 AM)

$ThresholdSeconds = 60 # Adjust as needed based on your monitoring intervals

# Function to check if primary replica is healthy

function Test-PrimaryReplicaHealth {

# Implement logic to check the health of the primary replica

# Return $true if healthy, $false otherwise

# Example: $PrimaryReplicaHealth = Get-SqlAvailabilityGroup -Name $AGGroupName | Get-SqlAvailabilityGroupReplica -Name $PrimaryReplica | Test-SqlAvailabilityGroupReplica -ErrorAction SilentlyContinue

# Example: return $PrimaryReplicaHealth.State -eq "Online"

}

# Function to initiate failover

function Invoke-Failover {

# Implement logic to initiate failover

# Example: Set-SqlAvailabilityGroup -Name $AGGroupName -Failover -ErrorAction SilentlyContinue

}

# Main script

while ($true) {

$CurrentTime = Get-Date

$ScheduledFailoverDateTime = Get-Date -Date $CurrentTime.Date.Add([TimeSpan]::Parse($ScheduledFailoverTime))

if ($CurrentTime -ge $ScheduledFailoverDateTime -and $CurrentTime -lt ($ScheduledFailoverDateTime.AddMinutes(1)) -and (Test-PrimaryReplicaHealth)) {

Write-Host "Scheduled failover time reached. Initiating failover..."

Invoke-Failover

# Add logic here to prevent multiple executions of failover during the same scheduled time window

Start-Sleep -Seconds 60 # Wait for a minute to avoid multiple executions within the same minute

}

Start-Sleep -Seconds 60 # Check every minute

}

```

Thoroughly test this script in a non-production environment before deploying it to production and ensure that the account running this script has appropriate permissions to perform failover operations on SQL Server.

1. **Steps to Schedule the Script:**

To schedule the execution of the PowerShell script on a specific day and time of the week, you can utilize the Windows Task Scheduler. Here's how you can do it:

1. Save the Script: Save the PowerShell script (.ps1 file) on your server or workstation where you want it to be executed.

2. Open Task Scheduler:

- Open the Start menu, type "Task Scheduler," and select it from the search results.

- Alternatively, you can press `Win + R`, type `taskschd.msc`, and press Enter to open Task Scheduler directly.

3. Create a New Task:

- In Task Scheduler, click on "Create Task..." from the right-hand menu.

4. General Tab:

- Enter a name and description for the task.

- Ensure that "Run whether user is logged on or not" is selected.

- Check "Run with highest privileges".

5. Triggers Tab:

- Click on "New..." to create a new trigger.

- Select "Weekly" from the drop-down menu.

- Choose the specific day(s) of the week you want the script to run.

- Set the start time and any other settings as needed.

6. Actions Tab:

- Click on "New..." to create a new action.

- Set the action to "Start a program".

- In the "Program/script" field, enter the path to `powershell.exe`.

- In the "Add arguments" field, enter the full path to your PowerShell script.

- Make sure to enclose the paths in double quotes if they contain spaces.

7. Conditions Tab (Optional):

- Adjust settings such as power, network, or idle conditions if necessary.

8. Settings Tab (Optional):

- Adjust settings such as allowing the task to be run on demand or stopping the task if it runs longer than a certain duration.

9. Save and Close:

- Click "OK" to save the task.

10. Provide Credentials:

- If prompted, provide the credentials of a user account that has the necessary permissions to execute the script and access resources it requires.

Once configured, the Task Scheduler will execute your PowerShell script at the specified day(s) and time(s) each week automatically.

Ensure that you thoroughly test the scheduled task to ensure it runs as expected, and monitor its execution in the Task Scheduler to detect any issues.