1. Configure a new Azure Availability Set from the Azure Portal
    a. Name the Availability Set LabAG
    b. Select your Subscription and assign the Resource Group
    c. Set the Location to match the location for the Azure VM's that will be created
    d. Change Fault Domains and Update Domains to 1 for the purposes of this lab
2. Configure three VM's with RHEL 7.6 and SQL Server 2017 installed:
    a. sqlagonrhel01
    b. sqlagonrhel02
    c. sqlagonrhel03
3. Configure the root account password
    a. sudo -s (enter the lab user password)
    b. passwd root (enter a new password for the root account)
    c. exit
4. Configure the Availability Group
    a. Configure hosts file on each replica with IP Address and names of the servers
        i. sudo ip addr show
            1. The IP Addresses are also available in the Azure Portal. Use the Private IP address of each server for the hosts file configuration
        ii. sudo nano /etc/hosts

```
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6
10.0.0.4    sqlagonrhel01
10.0.0.5    sqlagonrhel02
10.0.0.6    sqlagonrhel03
```

Note: You can rely on DNS registration if configured properly in the environment. This is just a "hack" around not having DNS in a test environment to allow it to be able to talk to the other nodes by name resolution. If the IP subnets are different in your network, use the correct IP's for each server node.

    b. Enable the HADR feature for each server replica and restart SQL Server
        i. `sudo /opt/mssql/bin/mssql-conf set hadr.hadrenabled 1`
        ii. `sudo systemctl restart mssql-server`
    c. `Choose a server to be the primary replica for configuration`
    d. `Connect to SQL Server using sqlcmd -Slocalhost,51433 -Usa and the SA password previously created`
        i. `Create a master key in the master database`
            1. CREATE MASTER KEY ENCRYPTION BY PASSWORD = '$tr0ngP@$$w0rd';
            2. GO
        ii. `Create a certificate for endpoint authentication between instances (NOTE: Specify an EXPIRY_DATE for the certificate manually, otherwise the certificate will expire one year from creation and require manual rotation across all replicas)`

1. CREATE CERTIFICATE hadr_cert WITH SUBJECT = 'HADR Endpoint Certificate', EXPIRY_DATE = '2030-01-01';
2. GO

iii. Backup the certificate (NOTE: For the purposes of this lab we will be backing up the certificate and the private key together. This violates security best practices and each replica should create a separate certificate that is exchanged without the private key to other replicas to minimize exposure risk should the certificate and private key be compromised)
1. BACKUP CERTIFICATE hadr_cert TO FILE = '/var/opt/mssql/data/hadr_cert.cer' WITH PRIVATE KEY (FILE = '/var/opt/mssql/data/hadr_cert.pvk', ENCRYPTION BY PASSWORD = '$tr0ngP@$$w0rd');
2. GO

iv. Exit sqlcmd with exit

e. Copy the certificate and private key backups to each additional replica
i. su (enter root password)
ii. cd /var/opt/mssql/data
iii. scp -P 52019 hadr_cert.* root@sqlagonrhel02:/var/opt/mssql/data/ (enter root password for each secondary server)
iv. scp -P 52019 hadr_cert.* root@sqlagonrhel03:/var/opt/mssql/data/ (enter root password for each secondary server)
v. exit

f. Assign permissions to the local mssql account to the certificate and private key files on all nodes of the cluster
i. su (enter root password)
ii. cd /var/opt/mssql/data
iii. chown mssql:mssql hadr_cert.*
iv. exit

g. Connect to SQL Server using sqlcmd -Slocalhost,51433 -Usa and the SA password previously created
i. Create a master key for the master database on each of the secondary servers
1. CREATE MASTER KEY ENCRYPTION BY PASSWORD = '$tr0ngP@$$w0rd';
2. GO

ii. Create the certificate on each of the secondary servers with the private key attached
1. CREATE CERTIFICATE HADR_Cert FROM FILE = '/var/opt/mssql/data/HADR_Cert.cer' WITH PRIVATE KEY (FILE = '/var/opt/mssql/data/HADR_Cert.pvk', DECRYPTION BY PASSWORD = '$tr0ngP@$$w0rd');
2. GO

iii. Create the mirroring endpoint for HADR on each instance and start the endpoint (NOTE: The default port used by SQL Server

Management Studio is 5022 for the mirroring endpoint, but any other port may be used)

1. CREATE ENDPOINT [hadr_endpoint] AS TCP (LISTENER_PORT = 5022) FOR DATABASE_MIRRORING (ROLE = ALL, AUTHENTICATION = CERTIFICATE hadr_cert, ENCRYPTION = REQUIRED ALGORITHM AES);
2. ALTER ENDPOINT [hadr_endpoint] STATE = STARTED;
3. GO

iv. Open the firewall TCP rules to allow the TCP listener port for the endpoint

1. sudo firewall-cmd --zone=public --add-port=5022/tcp –permanent
2. sudo firewall-cmd --reload

v. Add a firewall rule in Azure for the TCP port and only allow activity between the Private IP Addresses (NOTE: If stretching an on premise Availabilty Group to Azure VM's the public IP Address will be necessary)

h. Configure the Availability Group
   i. On the primary replica
      1. CREATE AVAILABILITY GROUP [LabAG] WITH (CLUSTER_TYPE = EXTERNAL) FOR REPLICA ON N'sqlagonrhel01' WITH (ENDPOINT_URL = N'tcp://sqlagonrhel01:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = EXTERNAL, SEEDING_MODE = AUTOMATIC), N'sqlagonrhel02' WITH (ENDPOINT_URL = N'tcp://sqlagonrhel02:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = EXTERNAL, SEEDING_MODE = AUTOMATIC), N'sqlagonrhel03' WITH (ENDPOINT_URL = N'tcp://sqlagonrhel03:5022', AVAILABILITY_MODE = SYNCHRONOUS_COMMIT, FAILOVER_MODE = EXTERNAL, SEEDING_MODE = AUTOMATIC);
      2. GO
      3. ALTER AVAILABILITY GROUP [LabAG] GRANT CREATE ANY DATABASE;
      4. GO
   ii. On each of the secondary replicas
      1. ALTER AVAILABILITY GROUP [LabAG] JOIN WITH (CLUSTER_TYPE = EXTERNAL);
      2. GO
      3. ALTER AVAILABILITY GROUP [LabAG] GRANT CREATE ANY DATABASE;
      4. GO

i. Create a new database to test the AG configuration
   i. CREATE DATABASE [TestAGDB];
   ii. ALTER DATABASE [TestAGDB] SET RECOVERY FULL;
   iii. BACKUP DATABASE [TestAGDB] TO DISK = 'TestAGDB.bak';
   iv. GO
   v. ALTER AVAILABILITY GROUP [LabAG] ADD DATABASE [TestAGDB];
   vi. GO

j. Check secondary replicas for new database:

i. SELECT * FROM sys.databases;
ii. GO

5. Configure PaceMaker High Availability
    a. Register with RedHat for a developer account (https://developers.redhat.com)
        i. Click on Log In then under Don't have an account? Click Create one now link.
    b. Accept the End User License Agreement for the developer subscription in the account created.
    c. On each server node register the subscription manager to Redhat
        i. sudo subscription-manager register
            1. This will prompt for the user name and password that you just created.
            2. If you haven't already accepted the end user license agreement for the subscription, copy/paste the link presented and Log In to accept the agreement then retry the subscription registration again.
        ii. sudo subscription-manager list –available
            1. This will list all the available subscriptions to the account that was registered.

SKU: RH00798
Contract:
Pool ID: <PoolID>
Provides Management: No
Available: 15
Suggested: 1
Service Level: Self-Support
Service Type:
Subscription Type: Standard
Starts: 12/11/2018
Ends: 12/11/2019
System Type: Physical

            2. The bottom of the information includes the Pool ID that we are going to use to attach the server to so that we can use the HA features for RHEL.
        iii. sudo subscription-manager attach --pool=<PoolID>
            1. This should output the following message:

Successfully attached a subscription for: Red Hat Developer Subscription

    d. On each server node enable the RHEL HA repository and install Pacemaker
        i. `sudo subscription-manager repos --enable=rhel-ha-for-rhel-7-server-rpms`

```
Repository 'rhel-ha-for-rhel-7-server-rpms' is enabled for this system.
```

        ii. sudo yum install pacemaker pcs fence-agents-all resource-agents
    e. On each server open the firewall ports for the high-availability service and reload the firewall
        i. sudo firewall-cmd --permanent --add-service=high-availability

ii. sudo firewall-cmd –reload

f. Set the password for the default hacluster account used by pacemaker on each node that will participate in the cluster and then enable the pcsd and pacemaker services and start pcsd so that the nodes can rejoin the cluster after any reboot.
   i. sudo passwd hacluster
   ii. `sudo systemctl enable pcsd`
   iii. `sudo systemctl start pcsd`
   iv. `sudo systemctl enable pacemaker`

g. Authorize each of the nodes that will participate in the cluster. This will prompt you to enter the password for the hacluster account that was just set in the previous step.
   i. sudo pcs cluster auth sqlagonrhel01 sqlagonrhel02 sqlagonrhel03 -u hacluster

h. Create the cluster in pacemaker across all three servers and start it.
   i. sudo pcs cluster setup --name sqlagcluster sqlagonrhel01 sqlagonrhel02 sqlagonrhel03 --force --start --all --enable
      1. This will stop the pacemaker services on each node, exchange the remote authkey for pacemaker across the specified servers, send the cluster config files to the nodes, synchronize the pcsd certificates across servers, and then restart pcsd to reload the certificates
      2. The --force option is not required unless you've previously created a cluster on the same nodes and need it to be destroyed.

i. Install SQL Server resource agent for SQL Server on all of the nodes
   i. `sudo yum install mssql-server-ha`

j. Pacemaker clusters require STONITH to be enabled and a fencing device to be configured to bring the cluster into a known state when the cluster resource manager cannot determine the state of a node or resource within the cluster. STONITH stands for 'shoot the other node in the head'. Pacemaker supports numerous fencing devices and the configuration depends heavily on the specific environment so for the purposes of this lab fencing is going to be disabled (it can be configured later for production systems based on their requirements).
   i. `sudo pcs property set stonith-enabled=false`

k. Create SQL Server Login for Pacemaker on all of the nodes and add the login to the sysadmin fixed server role in SQL Server
   i. `USE [master]`
   ii. `GO`
   iii. `CREATE LOGIN [pacemaker] with PASSWORD= N'P@c3M@ker!!'`
   iv. `ALTER SERVER ROLE [sysadmin] ADD MEMBER [pacemaker]`
   v. `GO`

l. Store the pacemaker login and password information in the pacemaker-passwd file in /var/opt/mssql/secrets/passwd
   i. `echo 'pacemaker' >> ~/pacemaker-passwd`
   ii. `echo 'P@c3M@ker!!' >> ~/pacemaker-passwd`
   iii. `sudo mv ~/pacemaker-passwd /var/opt/mssql/secrets/passwd`

m. Mark the pacemaker password file as owned by root and only readable by the root account for security
   i. `sudo chown root:root /var/opt/mssql/secrets/passwd`

ii. sudo chmod 400 /var/opt/mssql/secrets/passwd

n. Create the Availability Group resource inside the cluster
   i. sudo pcs resource create sqlrhelag ocf:mssql:ag ag_name=LabAG meta failure-timeout=60s master notify=true

o. Modify the Availability Group resource for the purposes of the lab environment to set the port attribute for the ag-helper resource agent to connect to the instances. NOTE: This is not required if SQL Server is listening on the default listener port 1433.
   i. sudo pcs resource update sqlrhelag attrib port=51433

p. Check everything's working with
   i. sudo pcs status

6. Configure Availability Group Listener IP
   a. Connect to SQL Server on the primary replica using sqlcmd -Slocalhost,51433 -Usa and the SA password previously
   b. Execute the following command to create the listener inside of the Availability Group with the IP address of 10.0.0.100 listening on port 51433. For Linux Availability Groups, the Listener Name does not register inside of pacemaker as a virtual name resource
      i. ALTER AVAILABILITY GROUP LabAG ADD LISTENER 'LabAGListener' (WITH IP ('10.0.0.100'), PORT = 51433);
      ii. GO
   c. Create an Azure Load Balancer to support the Availability Group IP Address – This is an Azure VM specific configuration requirement to allow the IP Address to transition between servers in the cloud properly.
      i. Select a subscription and the same resource group as the Availability Set used for the VM creation
      ii. Name the Load Balancer LabAG
      iii. Select the same region for the Load Balancer as the Availability Set for the VMs
      iv. Create the load balancer as Internal for the purposes of this lab. For an externally available listener, a Public Load Balancer would also need to be configured for the external IP Address.
      v. If the Virtual Machines were not created in an Availability Set, or in the same Availability Set, or were created in multiple availability zones in Azure, a Standard Load Balancer will be required. For the purposes of this lab a Basic Load Balancer can be used for a single Availability Set configuration.
      vi. Select the virtual network associated with the Virtual Machines, Availability Set or Zones for the Availability Group VMs
      vii. Select the subnet for the IP address
      viii. Set a Static IP Address from the available IP's in the range.
      ix. For a Standard Load Balancer configure it to be Zone-redundant. (Not Required for a Basic Load Balancer)
      x. Create the Load Balancer
      xi. Edit the Load Balancer in Azure Portal
      xii. Add a Backend Pool to the Load Balancer
         1. Name the pool LabAGPool
         2. Associated to Availability Set

3. Choose the LabAG Availability Set
4. Add a target Network IP Configuration for each VM in the Availabilty Set
5. Click OK to add the Pool

xiii. Add a health probe to the Load Balancer
1. Name it AGHealthProbe
2. Protocol TCP
3. Port 62000
4. Interval 5
5. Unhealthy threshold 2
6. Click OK to create the health probe.

xiv. Add a Load Balancing Rule to the Load Balancer
1. Name AGIPAddress
2. Frontend IP Address 10.0.0.100
3. Port 51433
4. Backend Port 51433
5. Backend Pool LabAGPool
6. Health Probe AGHealthProbe(TCP:62000)
7. Floading IP (direct server return) Enabled
8. Click OK to create the Rule

xv. Add a TCP inbound Network Security Group rule for Port 62000 to each of the VM Networks

d. Install the pacemaker resource agents for Azure to add support for the azure-lb resource type in pacemaker
   i. sudo yum install nmap-ncat resource-agents
e. Create the azure-lb resource for the load balancer probe port configured in the Azure Load Balancer for the Availability Group
   i. sudo pcs resource create AzureLBProbe azure-lb port=62000
f. Create a colocation constraint in pacemaker for the Azure load balancer probe port resource to follow the sqlrhelag resource
   i. sudo pcs constraint colocation add AzureLBProbe sqlrhelag-master INFINITY with-rsc-role=Master
g. Create a order constraint on the sqlrhelag resources to start first during a failover and then start the Azure Load Balancer probe port for the IP address.
   i. sudo pcs constraint order promote sqlrhelag-master then start AzureLBProbe
h. Create a firewall rule to allow the TCP port for the probe through
   i. sudo firewall-cmd --zone=public --add-port=62000/tcp --permanent
   ii. sudo firewall-cmd –reload

7. Create a BASH shell script to show AG failover servernames
   a. Edit the script with the following command and past the contents below into the file
   b. sudo nano /usr/local/bin/showagprimary
      i. #!/bin/bash
      ii. sqlcmd -S10.0.0.100,51433 -Usa -P 'pass@word1' -Q "SELECT @@SERVERNAME"
   c. Ctl+o to write the file contents out
   d. Ctl+X to exit
   e. Mark the script for execution
      i. sudo chmod +x /usr/local/bin/showagprimary

f. Export the path to the script so it can be used anywhere
    i. export PATH=$PATH:/usr/local/bin/
g. Test the script
    i. showagprimary
h. Move the Availability Group
    i. sudo pcs resource move sqlrhelag-master sqlagonrhel01 –master
i. Remove the location constraint created by failover
    i. sudo pcs constraint remove cli-prefer-sqlrhelag-master
j. Check the status
    i. sudo pcs status
k. Check the Primary servername
    i. showagprimary