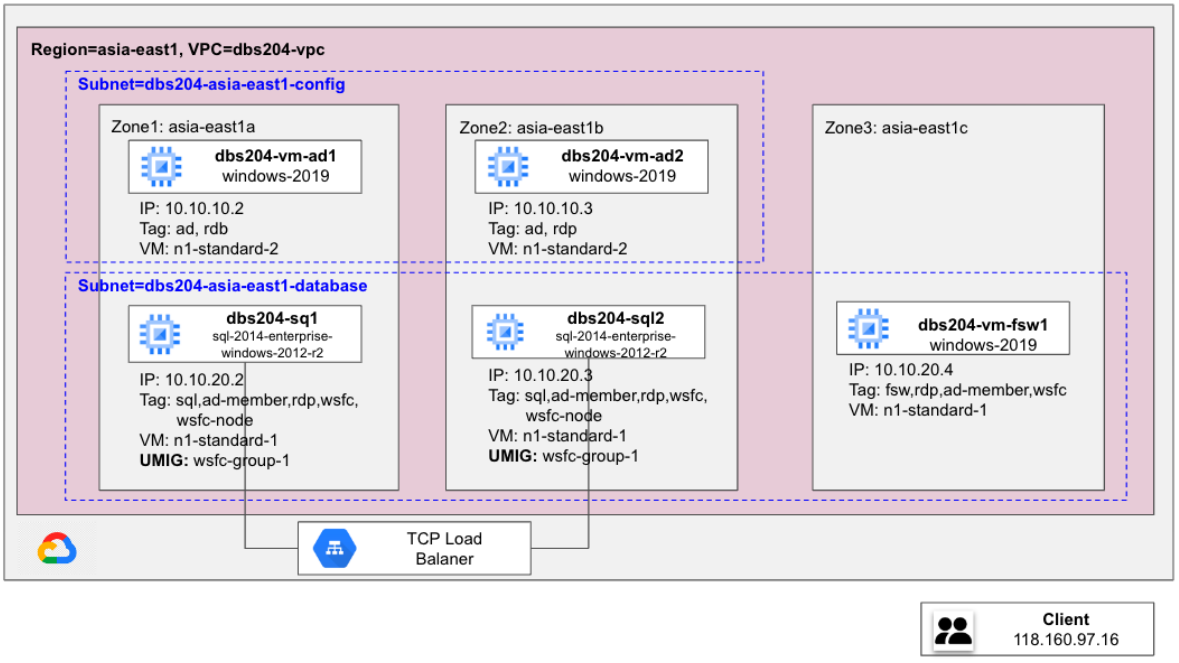
設定Single Subnet Multi Zone的SQL Server AlwaysON Availability Group和Internal Load Balancer的步驟

我們將設定步驟分成以下三個部分 - Network部分，Instance部分，和Internal Load Balancer部分來做說明：

- 灰色部份表示是在gcloud shell實作
- 藍色部分表示用RDP登入Windows，在PowerShell操作
- 黃色部份表示登入SQL Server，在SQL Server 操作
- 橘色部份表示使用VM等Editor編寫

System Architecture



Overview

- 在台灣區(asia-east1)建立一個VPC - dbs204-vpc，同時根據不同的Tag組別來設立防火牆。
- 建立兩個VM(ad1和ad2)來做AD的High Availability。
- 建立兩個VM(sql1和sql2)分別做Primary和Secondary，並且以fsw1來做File Share Witeness。
- 把這兩個VM加入AD的Group，接受AD的管理。
- 在sql1啟動SQL AlwaysOn的設定，並且設定Secondary和File Share Witness。
    - 將Listenner的Ip指令為Load Balancer IP
- 分別針對sql1和sql2設立 Unmanaged Instance Group(UMIG) - wsfc-group1和wsfc-group2。
- 建立TCP Load Balancer，並且把wsfc-group1和wsfc-group2當做TCP Load Balancer的Backend。
- 我們分別使用gcloud command和terraform的方式來建造SQL AlwaysOn Cluster
    - gcloud command是建立一個Primary(sql1)和一個Secondary(sql2)，Primary和Secondary在不同Zone。
    - terraform是建立一個Primary(sql1)和兩個Secondary(sql2和sql3)，Primary和Secondary在不同Zone。

Network部分

測試環境設定簡化設定

- 設定環境變數

```
CloudShell>
/*設定region在asia-east1*/
export region=asia-east1

/*設定三個Zone，Zone1在asia-east-1a，Zone2在asia-east-1b，Zone3在asia-east-c*/
export zone_1=${region}-a
export zone_2=${region}-b
export zone_3=${region}-c

/*設定VPC名稱*/
export vpc_name=dbs204-vpc

/*設定Project名稱*/
export project_id=lf-db-poc

/*設定子網域名稱*/
export subnet_name=dbs204-asia-east1-config

/*設定資料庫的子網域名稱*/
export sql_subnet_name=dbs204-asia-east1-database
```

- 設定Google Cloud CLI設定檔

```
CloudShell>
/*設定預計使用的Project ID在Google Cloud CLI設定檔中*/
```

```
gcloud config set project ${project_id}

/*設定預計使用的Region在Google Cloud CLI設定檔中*/
gcloud config set compute/region ${region}
```

設定VPC網路來管理資料庫環境

- 根據前面設定的環境變數${vpc_name}來建立VPC - dbs204-vpc，並且加上說明 - VPC network to deploy Active Directory，並且使用custom模式手動建立子網域

```
CloudShell>
gcloud compute networks create ${vpc_name}  \
   --description "VPC network to deploy Active Directory" \
   --subnet-mode custom
```

- 設定在VPC - dbs204-vpc和Region - asia-east1中設定兩個子網域 - dbs204-asia-east1-single-subnet 和dbs204-subnet-asia-east1-sql
    - ${subnet_name}是dbs204-asia-east1-single-subnet的環境變數
    - ${sql_subnet_name}是dbs204-subnet-asia-east1-sql的環境變數
    - ${region}是asia-east的環境變數
    - --enable-private-ip-google-access 啟動內網IP設定，只允許內網連接

```
CloudShell>
/*設定子網域dbs204-asia-east1-single-subnet*/
gcloud compute networks subnets create ${subnet_name} --network=${vpc_name}
--region=${region} --range=10.10.10.0/24 --enable-private-ip-google-access

/*設定子網域dbs204-subnet-asia-east1-sql*/
gcloud compute networks subnets create ${sql_subnet_name} --network=${vpc_name}
--region=${region} --range=10.10.20.0/24 --enable-private-ip-google-access
```

依照Tag和Client IP來設定防火牆

- 設定防火牆，只允許特定對象彼此溝通.
    - 118.160.97.16/32 是Client端的IP Range
    - Client端和Domain Controller間的防火牆常見的Rules可設定為
        - –rules=tcp:53,tcp:88,tcp:135,tcp:137,tcp:139,tcp:389,tcp:445,tcp:464,tcp:636,tcp:3268,tcp:3269,tcp:49152-65535,udp:53,udp:88,udp:123,udp:135,udp:137,udp:138,udp:389,udp:445,udp:464,udp:49152-65535

```
CloudShell>
/*設定Domain Controller的防火牆*/
gcloud compute firewall-rules create dbs204-fw-ad-controller --direction=INGRESS --priority=1000
--network=${vpc_name} --action=ALLOW --rules=all --source-tags=ad --target-tags=ad

/*設定Client端和Dommain Controller間的防火牆*/
gcloud compute firewall-rules create dbs204-fw-ad-member --direction=INGRESS --priority=1000
--network=${vpc_name} --action=ALLOW --rules=all ---source-ranges=10.10.10.0/24,10.10.20.0/24
--target-tags=ad, ad-member
```

```
/*設定SQL Server間的防火牆*/
gcloud compute firewall-rules create dbs204-fw-sql --direction=INGRESS --priority=1000
--network=${vpc_name} --action=ALLOW --rules=all --source-tags=sql --target-tags=sql

/*設定SQL Server和File Share Witness間的防火牆*/
gcloud compute firewall-rules create dbs204-fw-fs --direction=INGRESS --priority=1000
--network=${vpc_name} --action=ALLOW --rules=tcp:445 --source-tags=sql --target-tags=fsw

/*設定Client和SQL Server之間的防火牆*/
gcloud compute firewall-rules create dbs204-fw-sql-client --direction=INGRESS --priority=1000
--network=${vpc_name} --action=ALLOW --rules=tcp:1433 --source-ranges=118.160.97.16/32
--target-tags=sql


/*設定Tag是rdp的VM可以彼此溝通的防火牆*/
gcloud compute firewall-rules create allow-rdp --network ${vpc_name} --allow tcp:3389
--source-ranges 118.160.97.16/32 --target-tags=rdp
```

Instance部分

分為Domain Controller和SQL Server AlwaysOn Availability Group這兩個部分來做說明:

設定Activity Directory來做Domain Controller

建立兩個VM當作Domain Controller

- 建立第一個**Dommain Controller**的VM
    - 使用VM - n1-standard-2來建立AD - dbs204-vm-ad1
        - 使用以下參數在Project - if-db-poc上使用Persistent SSD (pd-ssd)當作OS Disk，且設定OS Disk為50GB。
            - --project=${project_id}
            - --boot-disk-type pd-ssd
            - --boot-disk-size 50GB
        - 使用以下參數來建立windows-2019的映像檔
            - --image-family windows-2019
            - --image-project windows-cloud
        - 使用以下參數在Zone - asia-east-1a和子網域 - dbs204-asia-east1-single-subnet中設定IP為10.10.10.2。
            - --zone ${zone_1} --subnet ${subnet_name}
            - --private-network-ip=10.10.10.2
        - 啟動Shieded VM機制來保護VM的資料，舉例shielded-secure-boot就是開機安全啟動，shielded-vtpm就是啟動Trusted Platform Module，shiedlded-integrity-monitoring就是啟動監控驗證資料完整性。
            - --shielded-secure-boot

        - --shielded-vtpm
        - --shielded-integrity-monitoring
    - 標註該VM的Tag為ad和rdp
        - --tags=ad,rdp
    - 並且以--score來啟動預設啟動的GoogleAPIs
        - --scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append

```
CloudShell>
/*建置dbs204-vm-ad1*/
gcloud compute instances create dbs204-vm-ad1 --machine-type n1-standard-2 \
   --project=${project_id} \
   --boot-disk-type pd-ssd \
   --boot-disk-size 50GB \
   --image-family windows-2019 --image-project windows-cloud \
   --zone ${zone_1} --subnet ${subnet_name} \
   --private-network-ip=10.10.10.2 \
   --shielded-secure-boot \
   --shielded-vtpm \
   --shielded-integrity-monitoring \
   --tags=ad,rdp \

--scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append
```

- 建立第二個**Dommain Controller**的VM
    - 使用VM - n1-standard-2來建立AD - dbs204-vm-ad2
        - 使用以下參數在Project - if-db-poc上使用Persistent SSD (pd-ssd)當作OS Disk，且設定OS Disk為50GB。
            - --project=${project_id}
            - --boot-disk-type pd-ssd
            - --boot-disk-size 50GB
        - 使用以下參數來建立windows-2019的映像檔
            - --image-family windows-2019
            - --image-project windows-cloud
        - 使用以下參數在Zone - asia-east-1a和子網域 - dbs204-asia-east1-single-subnet中設定IP為10.10.10.2。
            - --zone ${zone_1} --subnet ${subnet_name}
            - --private-network-ip=10.10.10.3
        - 啟動Shieded VM機制來保護VM的資料，舉例shielded-secure-boot就是開機安全啟動，shielded-vtpm就是啟動

Trusted Platform Module，shiedlded-integrity-monitoring就是啟動監控驗證資料完整性。

- --shielded-secure-boot
- --shielded-vtpm
- --shielded-integrity-monitoring

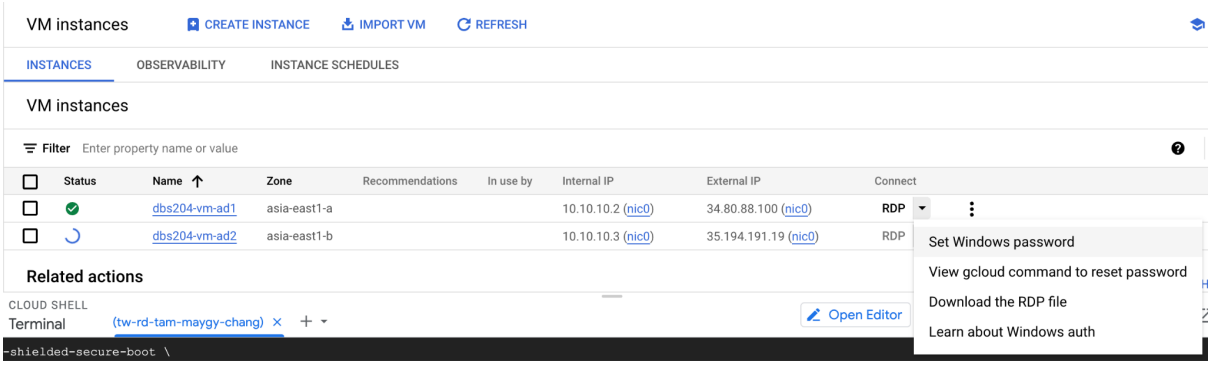
- 標註該VM的Tag為ad和rdp
  - --tags=ad,rdp
- 並且以--score來啟動預設啟動的GoogleAPIs
  - --scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append

CloudShell>
/*建置dbs204-vm-ad2*/

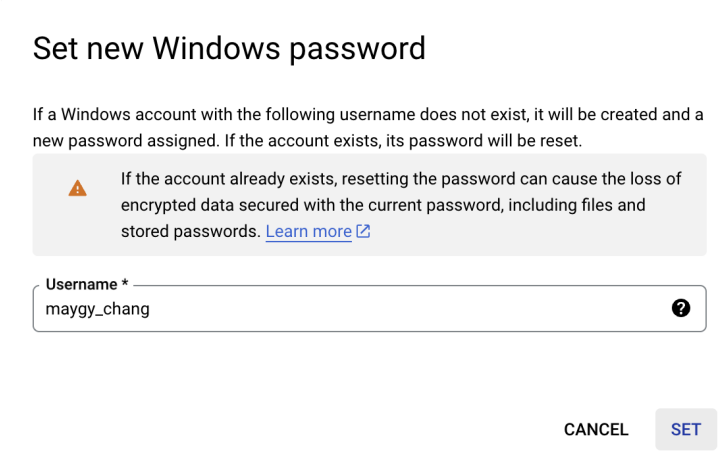
```
gcloud compute instances create dbs204-vm-ad2 --machine-type n1-standard-2 \
    --project=${project_id} \
    --boot-disk-type pd-ssd \
    --boot-disk-size 50GB \
    --image-family windows-2019 --image-project windows-cloud \
    --zone ${zone_2} --subnet ${subnet_name} \
    --private-network-ip=10.10.10.3 \
    --shielded-secure-boot \
    --shielded-vtpm \
    --shielded-integrity-monitoring \
    --tags=ad,rdp \

--scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/servicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleapis.com/auth/trace.append
```

設定Activity Directory來做高可用的Domain Controller

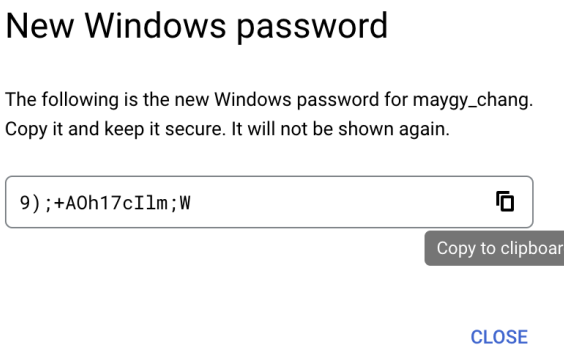
- 建立第一個**Domain Controller**
  - RDP 使用您在上一步中建立VM後產生憑據連接到Domain Controller
    - 點選RDP旁邊的下拉選單，選擇Set Windows password



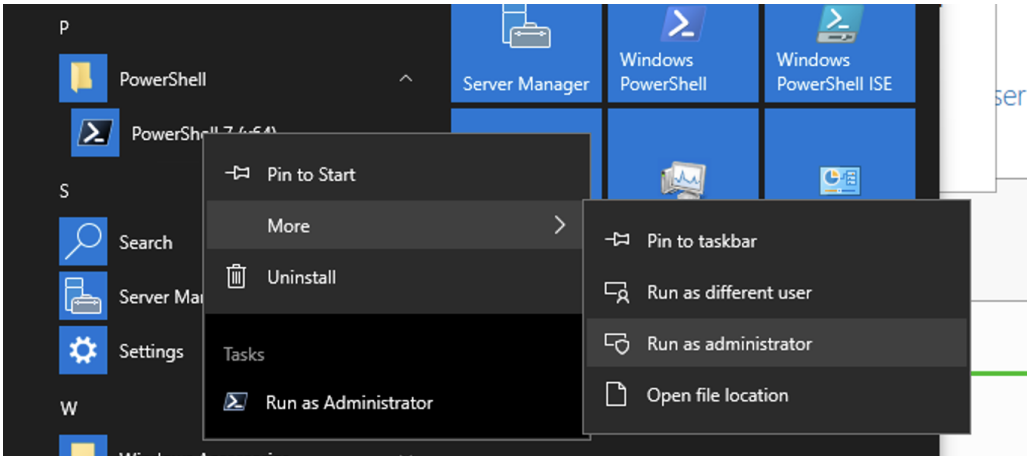
- 按SET來設定Windows password



- 複製password下來，以供之後RDP連線使用.



- 以管理員身份打開 PowerShell 終端。
  - 點擊“開始”，點選 “PowerShell”，按右鍵，選擇More，然後按Run as administrator
  - 或者點擊“開始”，輸入 “PowerShell”，然後按Shift+Ctrl+Enter，便會轉為管理者身份來登入PowerShell。



- 登入Powershell輸入以下指令
  - 假設Administrator的密碼為abc==123

PowerShell >
/*設定Administrator密碼，密碼設定需考慮密碼長度和複雜性，不然無法active*/
net user Administrator *

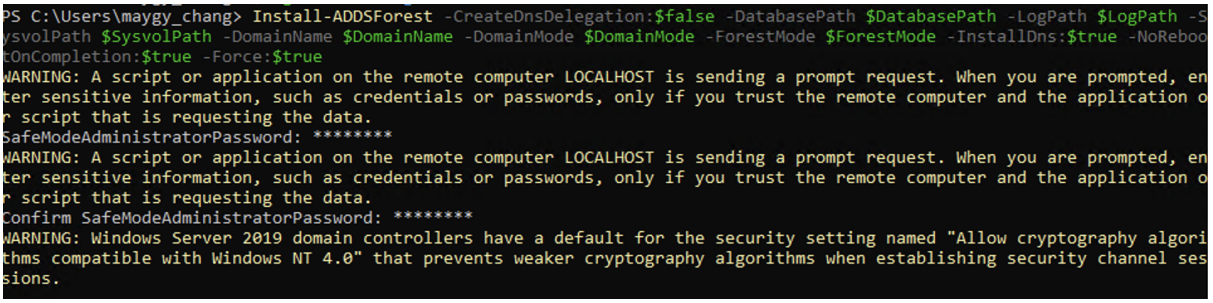
/*啟動Administrator*/
net user Administrator /active:yes

/*安裝AD Domain服務和管理工具*/
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools

/*設定變數*/
$DomainName = "ad.local"
$DomainMode = "7"
$ForestMode = "7"
$DatabasePath = "C:\Windows\NTDS"
$SysvolPath = "C:\Windows\SYSVOL"
$LogPath = "C:\Logs"

/*AD網域服務設定*/
Install-ADDSForest -CreateDnsDelegation:$false `
-DatabasePath $DatabasePath `
-LogPath $LogPath `
-SysvolPath $SysvolPath `
-DomainName $DomainName `
-DomainMode $DomainMode `
-ForestMode $ForestMode `
-InstallDNS:$true `
-NoRebootOnCompletion:$true `
-Force:$true

- 安裝ADDSForest的時候，系統會要你輸入Safe Mode Administrator的密碼。
  - 輸入兩次Administrator的密碼abc==123



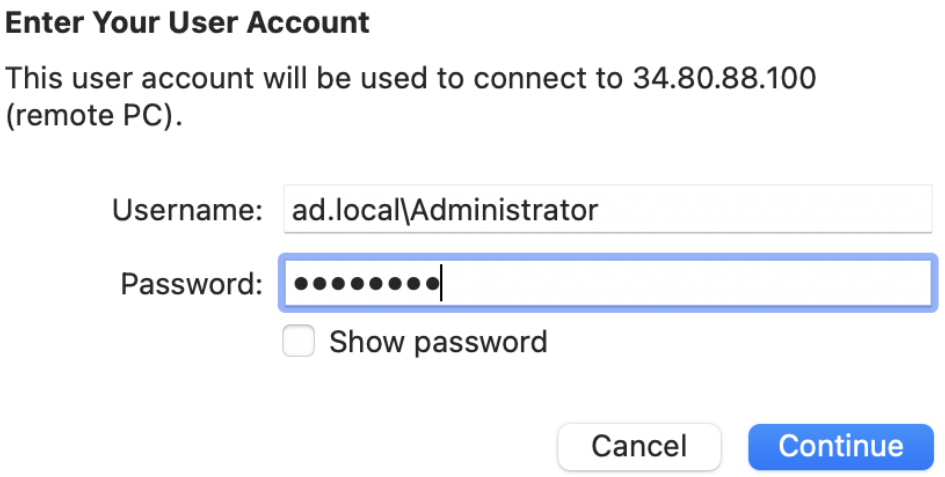
- 安裝完畢，可忽略系統出現以下的警告.

```
WARNING: This computer has at least one physical network adapter that does not have static IP address(es) assigned to it
s IP Properties. If both IPv4 and IPv6 are enabled for a network adapter, both IPv4 and IPv6 static IP addresses should
be assigned to both IPv4 and IPv6 Properties of the physical network adapter. Such static IP address(es) assignment shou
ld be done to all the physical network adapters for reliable Domain Name System (DNS) operation.


WARNING: A delegation for this DNS server cannot be created because the authoritative parent zone cannot be found or it
does not run Windows DNS server. If you are integrating with an existing DNS infrastructure, you should manually create
a delegation to this DNS server in the parent zone to ensure reliable name resolution from outside the domain "ad.local"
. Otherwise, no action is required.



RunspaceId     : 12b293e1-cffd-4966-aa4b-ceb213ef2b9d
Message        : You must restart this computer to complete the operation.

Context        : DCPromo.General.2
RebootRequired : True
Status         : Success
```

- 在PowerShell執行指令Restart-Computer來重啟VM - dbs204-vm-ad1

PowerShell >
Restart-Computer

- 使用 RDP 連接到域控制器AD1。
  - 請記住將域名添加為前綴，如 ad.local\Administrator
  - Username為ad.local\Administrator
  - Password為abc==123



- 以管理員身份打開 PowerShell 終端。
  - 點擊“開始”，輸入 “PowerShell”，然後按Shift+Ctrl+Enter，便會轉為管理者身份來登入PowerShell。

- 設置以下變量：

PowerShell >
/*設定第二個Domain Controller為DNS Primary*/
$DNSPrimary = "10.10.10.3"

/*設定第一個Domain Controller為DNS Secondary*/
$DNSSecondary = "127.0.0.1"

```
$LocalStaticIp = "10.10.10.2"
$DefaultGateway = "10.10.10.1"
```

- 設置IP地址和默認網閘:

```
PowerShell >
netsh interface ip set address name=Ethernet static $LocalStaticIp 255.255.255.0 $DefaultGateway
1
/*設定完RDP會重新連線*/
```

- 配置主 DNS 服務器為第二個Domain Controller:

```
PowerShell >
netsh interface ip set dns Ethernet static $DNSPrimary
```

- 此時第二個Domain Controller還未啟動，因此這個錯誤可忽略錯誤

```
The configured DNS server is incorrect or does not exist.
```

- 配置輔助 DNS 服務器:

```
PowerShell >
netsh interface ip add dns Ethernet $DNSSecondary index=2
```

- 建立第二個**Domain Controller**
  - 使用RDP 和在上一步中建立VM後產生憑據連接到Domain Controller
  - 以管理員身份打開 PowerShell 終端。
    - 點擊“開始”，輸入 “PowerShell”，然後按Shift+Ctrl+Enter。
  - 登入Powershell，安裝 Active Directory 域服務，包括管理工具:

```
PowerShell >
/*安裝AD Domain服務和管理工具*/
Install-WindowsFeature -Name AD-Domain-Services -IncludeManagementTools
```

- 設置以下變量:

```
PowerShell >
$DomainName = "ad.local"
$DomainPrefix = "AD"
$DNSPrimary = "10.10.10.2"
$DNSSecondary = "127.0.0.1"
$LocalStaticIp = "10.10.10.3"
$DefaultGateway = "10.10.10.1"
$DatabasePath = "C:\Windows\NTDS"
$SysvolPath = "C:\Windows\SYSVOL"
$LogPath = "C:\Logs"
```

- 配置第一個Domain Controller為主 DNS 服務器:

```
PowerShell >
netsh interface ip set dns Ethernet static $DNSPrimary
```

- 配置第二個Domain Controller為輔助 DNS 服務器:

```
PowerShell >
netsh interface ip add dns Ethernet $DNSSecondary index=2
```

- 此時第二個Domain Controller還未啟動，因此這個錯誤可忽略錯誤

```
The configured DNS server is incorrect or does not exist.
```

- 設置IP地址和默認網閘:

```
PowerShell >
netsh interface ip set address name=Ethernet static $LocalStaticIp 255.255.255.0 $DefaultGateway
1
```

- 運行以下 PowerShell 腳本，它將在第一個域控制器開始運行時通知您。
  - 等到看到 Domain controller is reachable消息。

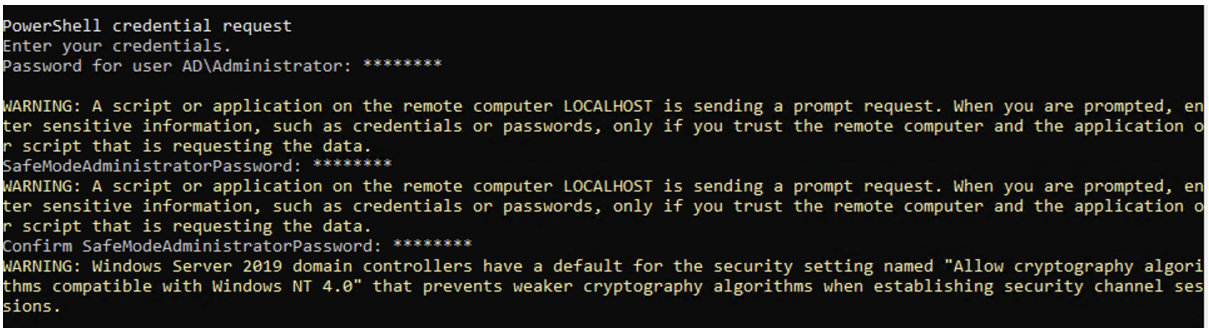
```
PowerShell >
$DomainIsReady=$False
For ($i=0; $i -le 30; $i++) {
    nltest /dsgetdc:$DomainName
    if($LASTEXITCODE -ne 0) {
        Write-Host "Domain not ready, wait 1 more minute, then retry"
        Start-Sleep -s 60
    }
    else {
        $DomainIsReady=$True
        Write-Host "Domain controller is reachable"
        break
    }
}
if($DomainIsReady -eq $False) {
    Write-Host "Domain not ready. Check if it was deployed ok"
}
```

- 將VM - dbs204-vm-ad2 作為第二個Domain Controller添加到Forest中:

```
PowerShell >
Install-ADDSDomainController `
    -Credential (Get-Credential "$DomainPrefix\Administrator") `
    -CreateDnsDelegation:$false `
    -DatabasePath $DatabasePath `
    -DomainName $DomainName `
```

```
-InstallDns:$true `
-LogPath $LogPath `
-SysvolPath $SysvolPath `
-NoGlobalCatalog:$false `
-SiteName 'Default-First-Site-Name' `
-NoRebootOnCompletion:$true `
-Force:$true
```

- 系統請您提供管理員帳戶的密碼時，即 ad.local\Administrator的密碼abc==123)。
- 系統也會請您提供SafeModeAdministrator的密碼，輸入之前設定的SafeModeAdministrator密碼(abc==123)。



- 在PowerShell執行指令來重啟VM - dbs204-vm-ad1

```
PowerShell >
Restart-Computer
```

- 測試**Domain Controller**安裝
  - 約等5-10分鐘等兩個Domain Controller的操作設定完成。
  - 以RDP連接第一個Domain Controller。
  - 以管理員身份打開 PowerShell 終端。
  - 測試複製是否正常工作：

```
PowerShell >
repadmin /replsum
```

- 輸出應類似於以下內容，沒有錯誤或失敗。
  - 如果過了10分鐘還是沒有出現兩台AD的訊息，應該是設定有錯誤，需要重新設定Domain Controller.

```
PS C:\Users\administrator> repadmin /replsum
Replication Summary Start Time: 2017-09-08 20:15:55

Beginning data collection for replication summary, this may take awhile:
  .....


Source DSA          largest delta    fails/total %%   error
 AD-DC1                     11m:11s    0 /   5    0
 AD-DC2                     10m:06s    0 /   5    0


Destination DSA     largest delta    fails/total %%   error
 AD-DC1                     10m:06s    0 /   5    0
 AD-DC2                     11m:11s    0 /   5    0
```

設定SQL Server AlwaysOn availability groups

VM和網路設定

- 設定環境變數
  - 當Cloud Shell閒置太久的時候，會中斷連線，需要重新登入。
  - 而重新登入的時候，環境變數設定可能會不見，需要重新設定.

CloudShell>
/*設定region在asia-east1*/
export region=asia-east1

/*設定三個Zone，Zone1在asia-east-1a，Zone2在asia-east-1b，Zone3在asia-east-c*/
export zone_1=${region}-a
export zone_2=${region}-b
export zone_3=${region}-c

/*設定Project名稱*/
export project_id=lf-db-poc

/*設定VPC名稱*/
export vpc_name=dbs204-vpc

/*設定資料庫的子網域名稱*/
export sql_subnet_name=dbs204-asia-east1-database

/*設定預計使用的Region在Google Cloud CLI設定檔中*/
gcloud config set compute/region ${region}

依照Tag建立SQL和WSFC的防火牆規則

- 返回CloudShell，設立subnet_cidr環境變數

CloudShell>
subnet_cidr=$(gcloud compute networks subnets describe $sql_subnet_name
--format=value\('ipCidrRange'\))

- 設立wsfc之間的防火牆

CloudShell>
```
gcloud compute firewall-rules create dbs204-fw-allow-all-between-wsfc-nodes \
  --direction=INGRESS \
  --action=allow \
  --rules=tcp,udp,icmp \
  --enable-logging \
  --source-tags=wsfc \
  --target-tags=wsfc \
  --network=$vpc_name \
  --priority 10000
```

- 設立sql和wsfc的防火牆

CloudShell>
```
gcloud compute firewall-rules create dbs204-fw-allow-sql-to-wsfc-nodes \
  --direction=INGRESS \
  --action=allow \
  --rules=tcp:1433 \
  --enable-logging \
  --source-ranges=$subnet_cidr \
  --target-tags=wsfc-node \
  --network=$vpc_name \
  --priority 10000
```

- 設立防火牆，允許從 Google Cloud 探測器的 IP 範圍進行Health Check。

CloudShell>
```
gcloud compute firewall-rules create dbs204-fw-allow-health-check-to-wsfc-nodes \
  --direction=INGRESS \
  --action=allow \
  --rules=tcp \
  --source-ranges=130.211.0.0/22,35.191.0.0/16 \
  --target-tags=wsfc-node \
  --network=$vpc_name \
  --priority 10000
```

用VM和Start Script來建立Primary，Secondary和Witness，並加入AD

- 返回Cloud Shell，執行以下指令，來寫一個Script - specialize-node.ps1

CloudShell>
```
cat << "EOF" > specialize-node.ps1

$ErrorActionPreference = "stop"

# Install required Windows features
Install-WindowsFeature Failover-Clustering -IncludeManagementTools
Install-WindowsFeature RSAT-AD-PowerShell

# Open firewall for WSFC
netsh advfirewall firewall add rule name="Allow SQL Server health check" dir=in action=allow
protocol=TCP localport=59997
```

```
# Open firewall for SQL Server
netsh advfirewall firewall add rule name="Allow SQL Server" dir=in action=allow protocol=TCP
localport=1433

# Open firewall for SQL Server replication
netsh advfirewall firewall add rule name="Allow SQL Server replication" dir=in action=allow
protocol=TCP localport=5022

# Format data disk
Get-Disk |
 Where partitionstyle -eq 'RAW' |
 Initialize-Disk -PartitionStyle MBR -PassThru |
 New-Partition -AssignDriveLetter -UseMaximumSize |
 Format-Volume -FileSystem NTFS -NewFileSystemLabel 'Data' -Confirm:$false

# Create data and log folders for SQL Server
md d:\Data
md d:\Logs
EOF
```

- 設定環境變數pd_size=200

```
CloudShell>
export pd_size=200
```

- 在Zone1建立SQL Server1(dbs204-sql1)
  - 在Project - if-db-poc中和Zone - asia-east1a中設置n1-standard-1的VM當作SQL Server
    - --machine-type n1-standard-1 --project=${project_id} --zone=${zone_1}
  - 以SQL2014和Windows2012的映像檔來建立Disk
    - --create-disk=auto-delete=yes,boot=yes,device-name=dbs204-sql1,image=projects/windows-sql-cloud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v20230510,mode=rw,size=50,type=projects/${project_id}/zones/${zone_1}/diskTypes/pd-balanced
  - 啟動Shieded VM機制來保護VM的資料，舉例shielded-secure-boot就是開機安全啟動，shielded-vtpm就是啟動Trusted Platform Module，shiedlded-integrity-monitoring就是啟動監控驗證資料完整性。
    - --shielded-secure-boot
    - --shielded-vtpm
    - --shielded-integrity-monitoring

```
CloudShell>
gcloud compute instances create dbs204-sql1 --machine-type n1-standard-1 \
   --project=${project_id} \
   --zone=${zone_1} \

--create-disk=auto-delete=yes,boot=yes,device-name=dbs204-sql1,image=projects/windows-sql-cl
oud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v20230510,mode=rw,size=50,type=pro
jects/${project_id}/zones/${zone_1}/diskTypes/pd-balanced \
--shielded-secure-boot \
--shielded-vtpm \
```

```
--shielded-integrity-monitoring \

--network-interface=network-tier=PREMIUM,private-network-ip=10.10.20.2,stack-type=IPV4_ONLY,
subnet=${sql_subnet_name} \
    --tags=sql,ad-member,rdp,wsfc,wsfc-node \

--scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/
logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/s
ervicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleap
is.com/auth/trace.append \
  --create-disk=name=dbs204-sql1-datadisk,size=$pd_size,type=pd-ssd,auto-delete=no \
  --metadata enable-wsfc=true \
  --metadata-from-file=sysprep-specialize-script-ps1=specialize-node.ps1
```

- 可使用以下指令確認SQL Server1是否Ready

```
CloudShell>
gcloud compute instances tail-serial-port-output dbs204-sql1
```

- 在Zone2建立SQL Server2(dbs204-sql1)

```
CloudShell>
gcloud compute instances create dbs204-sql2 --machine-type n1-standard-1 \
    --project=${project_id} \
    --zone=${zone_2} \

--create-disk=auto-delete=yes,boot=yes,device-name=dbs204-sql2,image=projects/windows-sql-cl
oud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v20230510,mode=rw,size=50,type=pro
jects/${project_id}/zones/${zone_2}/diskTypes/pd-balanced \
    --shielded-secure-boot \
    --shielded-vtpm \
    --shielded-integrity-monitoring \

--network-interface=network-tier=PREMIUM,private-network-ip=10.10.20.3,stack-type=IPV4_ONLY,
subnet=${sql_subnet_name} \
    --tags=sql,ad-member,rdp,wsfc,wsfc-node \

--scopes=https://www.googleapis.com/auth/devstorage.read_only,https://www.googleapis.com/auth/
logging.write,https://www.googleapis.com/auth/monitoring.write,https://www.googleapis.com/auth/s
ervicecontrol,https://www.googleapis.com/auth/service.management.readonly,https://www.googleap
is.com/auth/trace.append \
  --create-disk=name=dbs204-sql2-datadisk,size=$pd_size,type=pd-ssd,auto-delete=no \
  --metadata enable-wsfc=true \
  --metadata-from-file=sysprep-specialize-script-ps1=specialize-node.ps1
```

- 可使用以下指令確認SQL Server2是否Ready

```
CloudShell>
gcloud compute instances tail-serial-port-output dbs204-sql2
```

- 建立FSW1(dbs204-vm-fsw1)做Witness主機使用

```
CloudShell>
gcloud compute instances create dbs204-vm-fsw1 --machine-type n1-standard-1\
   --boot-disk-type pd-ssd \
   --boot-disk-size 50GB \
   --image-family windows-2019 --image-project windows-cloud \
   --zone ${zone_3} --subnet ${sql_subnet_name} \
   --private-network-ip=10.10.20.4 \
   --tags=fsw,rdp,ad-member,wsfc \
   --scopes=default \
   --metadata sysprep-specialize-script-ps1="add-windowsfeature FS-FileServer"
```

將VM加入AD的Domain管理

- 將dbs204-sql1, dbs204-sql2，dbs204-vm-fsw1等3 個 VM 加入 Active Directory的Domain
    - 使用RDP 和在上一步中建立VM後產生憑據分別連接到這三個VM
    - 以管理員身份打開 PowerShell 終端。
        - 點擊“開始”，輸入 “PowerShell”，然後按Shift+Ctrl+Enter，便會轉為管理者身份來登入PowerShell。
    - 對三個VM確認是能連結到Domain Controller(10.10.10.2, 10.10.10.3)：

```
PowerShell >
Test-NetConnection 10.10.10.2
```

    - 對三個VM分別執行以下操作：

```
PowerShell >
Set-DnsClientServerAddress -InterfaceAlias @("Ethernet") -ServerAddresses
@("10.10.10.2,10.10.10.3")
```

    - 將VM加入您的 Active Directory 域並重新啟動：

```
PowerShell >
Add-Computer -Domain ad.local -Restart
```

    - 執行指令的時候會跳出以下的視窗，在以下視窗輸入使用者為ad.local\Administrator和ad.local\Administrator的密碼(abc==123)：

- 解決在dbs204-vm-fsw1輸入Add-Computer可能報錯問題
    - dbs204-vm-fsw1可能“add-computer” is not recognized as a name of a cmdlet
        - 這是因為fsw1的系統是windows2019，預設沒有安裝PowerShell.Management的套件。
    - 以管理員身份打開 PowerShell 終端。
        - 點擊“開始”，輸入 “PowerShell”，然後按Shift+Ctrl+Enter，便會轉為管理者身份來登入PowerShell。
    - 先安裝以下套件，再輸入一次“Add-Computer -Domain ad.local -Restart”

```
PowerShell >
Import-Module Microsoft.PowerShell.Management -UseWindowsPowerShell
```

設定Load Balancer IP和Cluster IP供Listener設定

- 在 VPC 中保留了兩個靜態 IP 地址。
    - 一個 IP 地址用作 WSFC 群集 IP 地址，另一個由內部負載平衡器使用。
    - 步驟1. 為內部負載均衡器保留靜態 IP

```
CloudShell>
/*保留IP位址*/
gcloud compute addresses create wsfc \
  --subnet ${sql_subnet_name} \
  --region $(gcloud config get-value compute/region)

/*設定並顯示環境變數LOADBALANCER_ADDRESS，假設為10.10.20.10*/
LOADBALANCER_ADDRESS=$(gcloud compute addresses describe wsfc \
  --region $(gcloud config get-value compute/region) \
  --format=value\(address\)) && \
echo "Load Balancer IP: $LOADBALANCER_ADDRESS"
```

    - 步驟2.保留另一個用作集群 IP 的靜態 IP 地址：

```
CloudShell>
/*保留IP位址且設定並顯示CLUSTER_ADDRESS，假設該IP為10.10.20.11*/
```

```
gcloud compute addresses create wsfc-cluster \
  --subnet ${sql_subnet_name}  \
  --region $(gcloud config get-value compute/region) && \
CLUSTER_ADDRESS=$(gcloud compute addresses describe wsfc-cluster \
    --region $(gcloud config get-value compute/region) \
    --format=value\(address\)) && \
echo "Cluster IP: $CLUSTER_ADDRESS"
```

SQL Server設定

設定Primary和Secondary為故障轉移集群

- 以下SQL Server的操作皆使用ad.local\Administrator的身份來操作
- 準備 SQL Server
    - 以ad.local\Administrator的身份來使用RDP連接到sql1
    - 開啟PowerShell，為 SQL Server 和 SQL 代理建立域用戶帳戶和分配密碼：

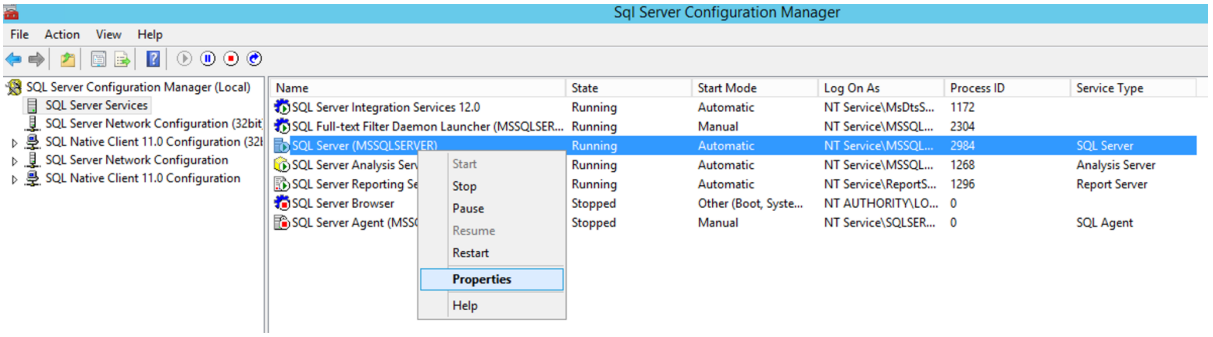
```
PowerShell>
/*設定User - sql_server的密碼*/
$Credential = Get-Credential -UserName sql_server -Message 'Enter password'

/*將User - sql_server加入AD*/
New-ADUser `
  -Name "sql_server" `
  -Description "SQL Admin account." `
  -AccountPassword $Credential.Password `
  -Enabled $true -PasswordNeverExpires $true
```

- 要配置 SQL Server，請對Primary(dbs204-sql1)和Secondary(dbs204-sql2和dbs204-sql3)執行以下步驟
    - 步驟1.以ad.local\Administrator的身份來登入sql1和sql2的PowerShell
        - 重命名 SQL Server(dbs204-sql1和dbs204-sql2)，使其名稱與主機名匹配：

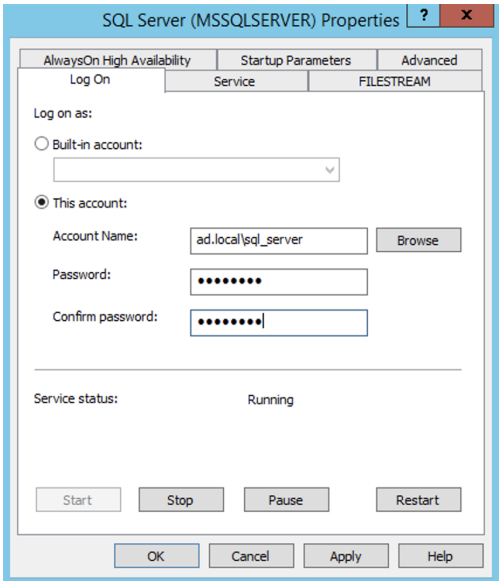
```
PowerShell>
Invoke-Sqlcmd -Query "
  sp_dropserver 'INST-INSTALL-SQ';
  GO
  sp_addserver '$env:computername', local;
  GO"
Restart-Service -Name MSSQLSERVER
```

    - 步驟2.打開SQL Server Configuration Manager
        - 搜尋SQL Server Configuration Manager，即看到SQL Server 2014 Configuration Manager。
    - 步驟3.在導航窗格中，選擇SQL Server Services

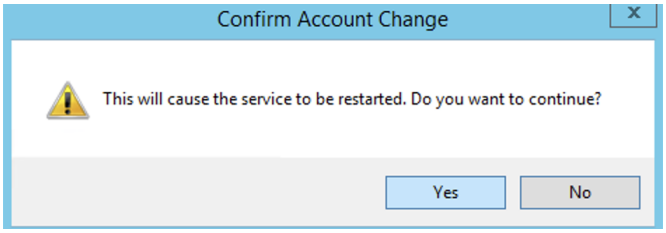
- 步驟4.在服務列表中，右鍵單擊SQL Server (MSSQLSERVER)並選擇Properties。



- 步驟5.在Log on as下，更改帳戶：
- 帳戶名稱：您的 Active Directory 域的 NetBIOS 名稱 ad.local\sql_server



- 步驟6.單擊確定(OK)。
- 步驟7.當提示重新啟動 SQL Server 時，選擇”Yes”。



- SQL Server 現在在域用戶帳戶(ad.local\sql_server)下運行。

在FSW主機上建立文件共享

- 在dbs204-vm-fsw1上建立兩個文件共享，以便該VM可以存儲 SQL Server 備份並充當文件共享見證：
  - 步驟1. 使用RDP連接到fsw1的VM。使用您的域用戶帳戶ad.local\Administrator登錄。
  - 步驟2. 啟動PowerShell

- 步驟3. 建立Witness文件共享並授予您自己和兩個 WSFC 節點(dbs204-sql1和dbs204-sql2)訪問文件共享的權限
- 該資料夾必須獨立於各節點

```
PowerShell>
/*建立Witness文件*/
New-Item "C:\QWitness" –type directory

/*授予權限給dbs204-sql1和dbs204-sql2*/
icacls C:\QWitness\ /grant 'dbs204-sql1$:(OI)(CI)(M)'
icacls C:\QWitness\ /grant 'dbs204-sql2$:(OI)(CI)(M)'

/*設定QWitness文件讓dbs204-sql1和dbs204-sql2存取*/
New-SmbShare `
  -Name QWitness `
  -Path "C:\QWitness" `
  -Description "SQL File Share Witness" `
  -FullAccess $env:username,dbs204-sql1$,dbs204-sql2$

/*倘若有兩個Secondary，設定QWitness文件讓dbs204-sql1, dbs204-sql2和dbs204-sql3存取*/
New-SmbShare `
  -Name QWitness `
  -Path "C:\QWitness" `
  -Description "SQL File Share Witness" `
  -FullAccess $env:username,dbs204-sql1$,dbs204-sql2$,dbs204-sql3$
```

- 步驟4.建立另一個文件共享來存儲備份並授予 SQL Server 完全訪問權限：

```
PowerShell>
/*建立Backup文件*/
New-Item "C:\Backup" –type directory

/*設定Backup文件共享權限*/
New-SmbShare `
  -Name Backup `
  -Path "C:\Backup" `
  -Description "SQL Backup" `
  -FullAccess  $env:USERDOMAIN\sql_server
```

- 授予ad/administrator 完全訪問權限：

```
PowerShell>
Grant-SmbShareAccess -Name "Backup" -AccountName $env:USERDOMAIN\administrator
-AccessRight Full
```

設定Witness和啟動SQL AlwaysOn設定

- 登入以下主機的RDP需皆使用ad.local\Administrator的帳號
- 現在已準備好建立故障轉移集群：
  - 步驟1. 用RDP來登入dbs204-sql1上

- 步驟2. 開啟PowerShell
- 步驟3. 建立一個新的Cluster:
  - 10.10.20.6 是前面建立的 cluster IP

```
PowerShell>
/*建立一個Primary和一個Secondary的叢集*/
New-Cluster `
  -Name sql-cluster `
  -Node dbs204-sql1,dbs204-sql2 `
  -NoStorage `
  -StaticAddress 10.10.20.6

/*建立一個Primary和兩個Secondary的叢集*/
New-Cluster `
  -Name sql-cluster `
  -Node dbs204-sql1,dbs204-sql2,dbs204-sql3 `
  -NoStorage `
  -StaticAddress 10.10.20.11
```

- 步驟4. 登入dbs204-vm-fsw1並開啟PowerShell，並授予集群的虛擬計算機對象訪問文件共享的權限：

```
PowerShell>
icacls C:\QWitness\ /grant 'sql-cluster$:(OI)(CI)(M)'
Grant-SmbShareAccess `
  -Name QWitness `
  -AccountName 'sql-cluster$' `
  -AccessRight Full `
  -Force
```

- 步驟5.登入dbs204-sq1的 PowerShell，並將集群配置為使用Witness服務器(dbs204-vm-fsw1)上的文件共享作為集群仲裁：

```
PowerShell>
Set-ClusterQuorum -FileShareWitness \\DBS204-VM-FSW1\QWitness
```

- 步驟6.在dbs204-sql1上驗證集群是否建立成功：

```
PowerShell>
Test-Cluster
```

- 步驟7.在所有db節點(dbs204-sql1，dbs204-sql2，和dbs204-sql3)上啟用 AlwaysOn 可用性組：

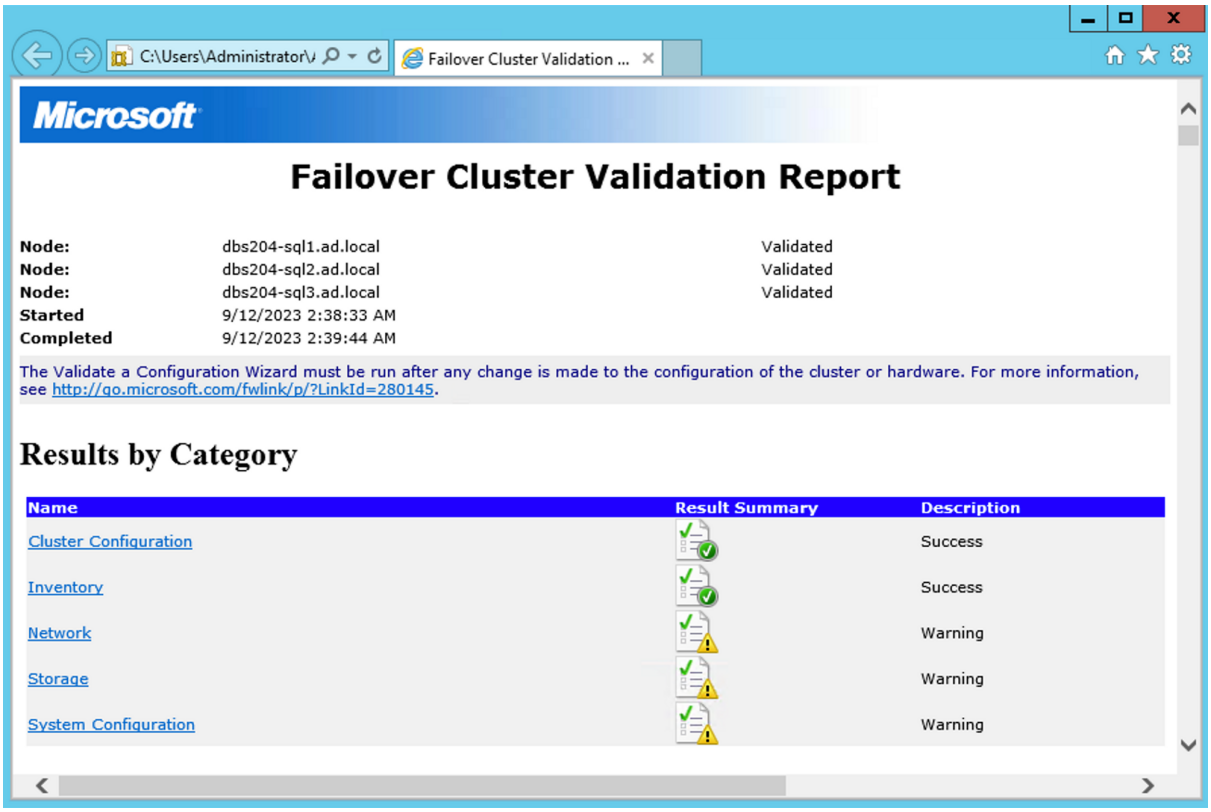
```
PowerShell>
/*在dbs204-sql1執行*/
Enable-SqlAlwaysOn -ServerInstance dbs204-sql1 -Force

/*在dbs204-sql2執行*/
Enable-SqlAlwaysOn -ServerInstance dbs204-sql2 -Force
```

/*在dbs204-sql3執行*/
Enable-SqlAlwaysOn -ServerInstance dbs204-sql3 -Force

- 測試叢集間的節點連線是否成功
    - 指令測試完畢，會產生一個Validation Report在 C:\Users\Administrator\AppData\Local\Temp\2\下

PowerShell >
Test-Cluster

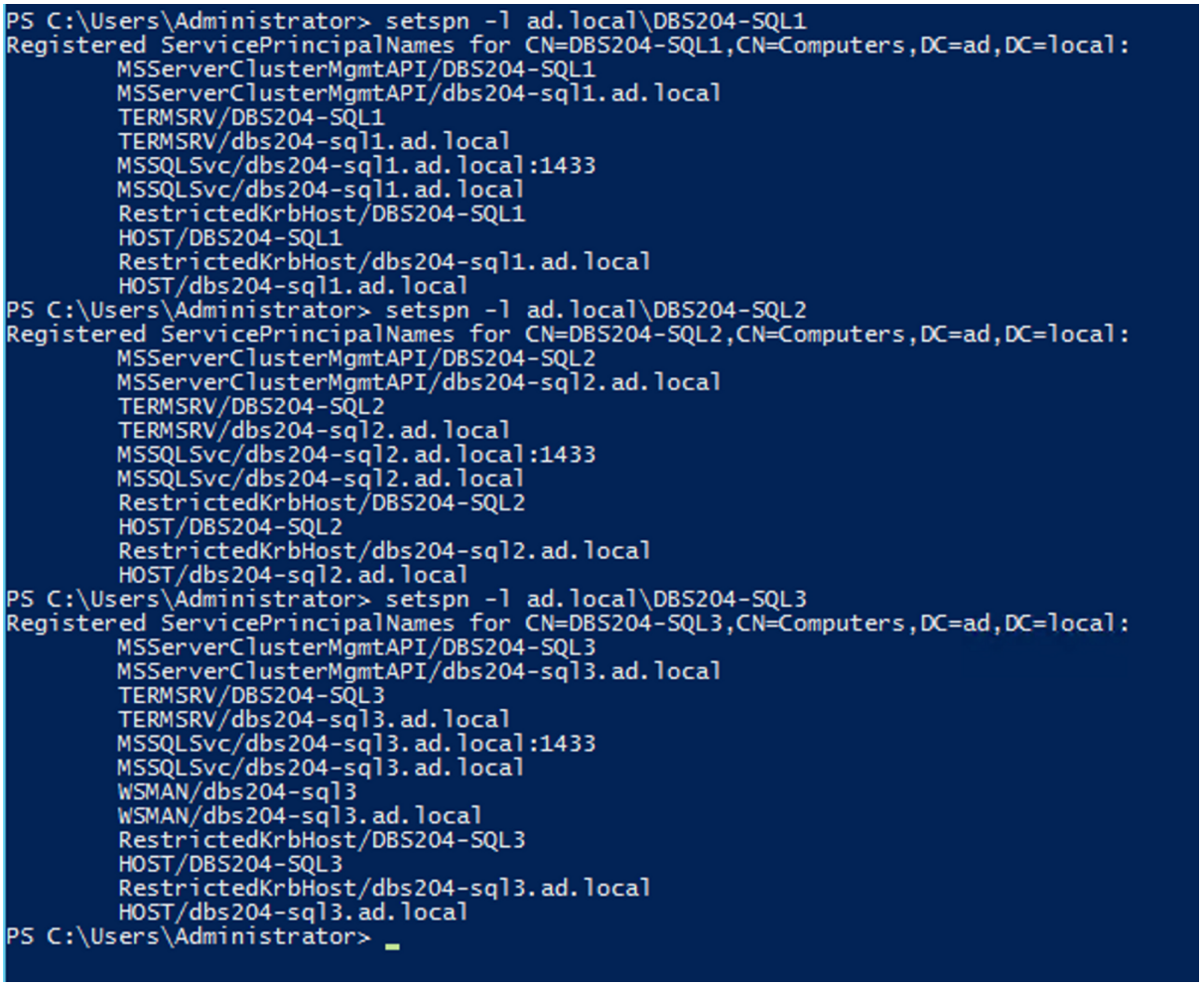
- 開啟Validation Report，確認所有Cluster的狀態都是都是Validated



設置 SPN

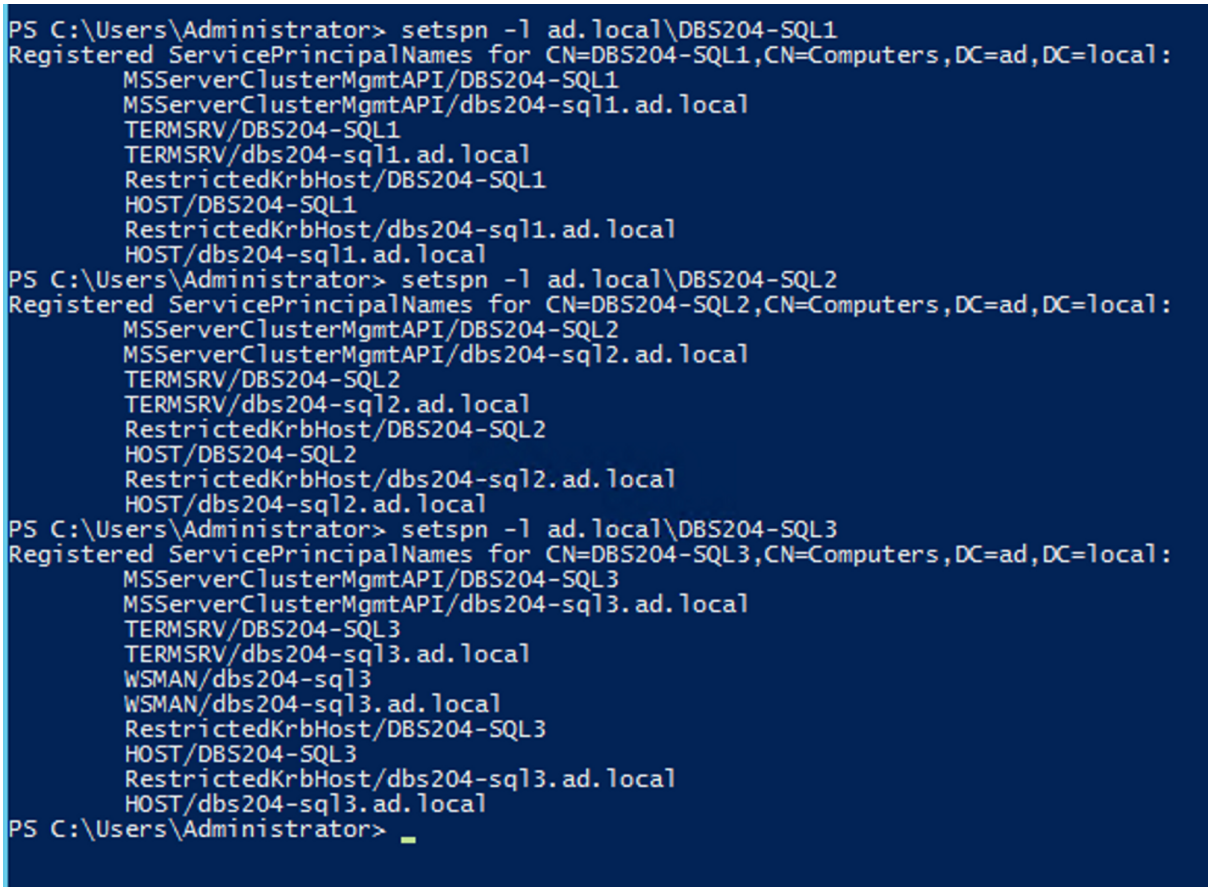
- 進入 dbs204-sql1 server
- 執行以下指令查看結果
    - SPN(Service Principal name)伺服器主體名稱，

PowerShell>
setspn -l ad.local\DBS204-SQL1
setspn -l ad.local\DBS204-SQL2
setspn -l ad.local\DBS204-SQL3



- 將`MSSQLSvc/*` 為多的設置等多餘的設置用以下指令刪除

PowerShell>
setspn -d MSSQLSvc/dbs204-sql1.ad.local:1433 DBS204-SQL1
setspn -d MSSQLSvc/dbs204-sql1.ad.local DBS204-SQL1
setspn -d MSSQLSvc/dbs204-sql2.ad.local:1433 DBS204-SQL2
setspn -d MSSQLSvc/dbs204-sql2.ad.local DBS204-SQL2
setspn -d MSSQLSvc/dbs204-sql3.ad.local:1433 DBS204-SQL3
setspn -d MSSQLSvc/dbs204-sql3.ad.local DBS204-SQL3



- 在這些DB的VM執行以下指令來重啟 MSSQLSERVER

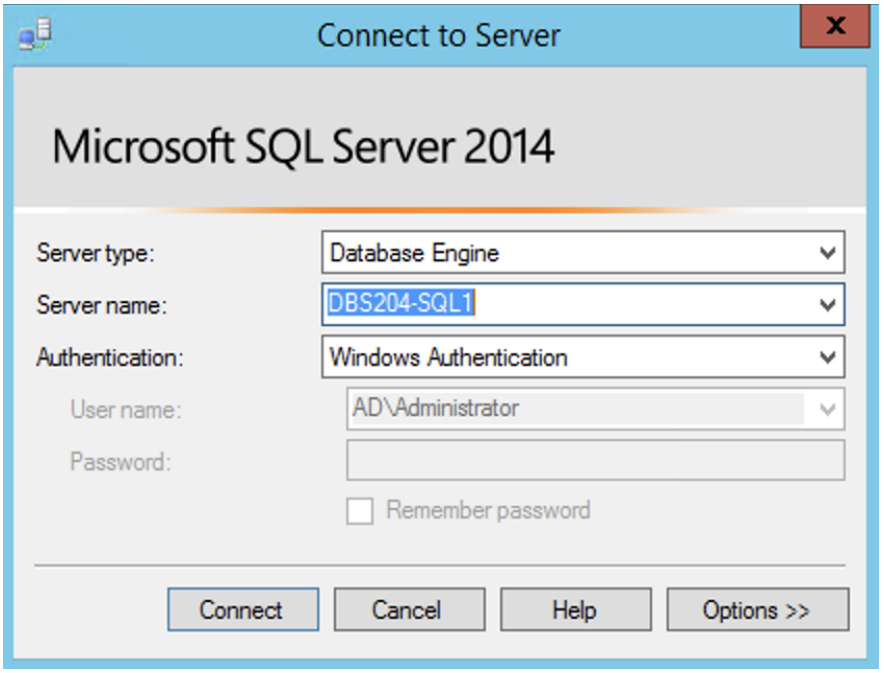
PowerShell>
Restart-Service -Name MSSQLSERVER

建立DB並設定其為可用

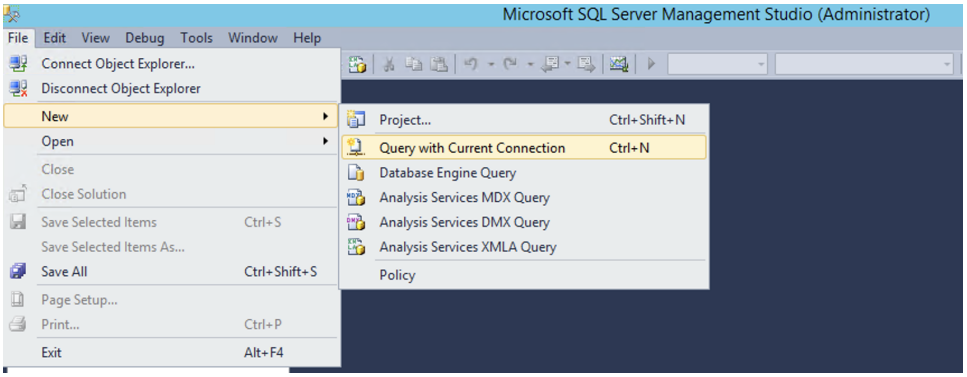
建立一個示例資料庫bookshelf，將其包含在名為 bookshelf-ag 的新可用性組中並配置高可用性。

登入DB主機建立資料庫

- 登入SQL主機的RDP需皆使用ad.local\Administrator的帳號
- 操作步驟
  - 步驟1. 使用RDP登入dbs204-sql1 上的
  - 步驟2.開啟SQL Server Management Studio
  - 步驟3.在“連接到服務器”對話框中，確認服務器名稱設置為 dbs204-sql1 並選擇“連接”。



- 步驟4.在選單中，選擇 File > New > Query with current connection



- 步驟5.接著會出現一個Editor視窗，在Editor貼上以下SQL script:

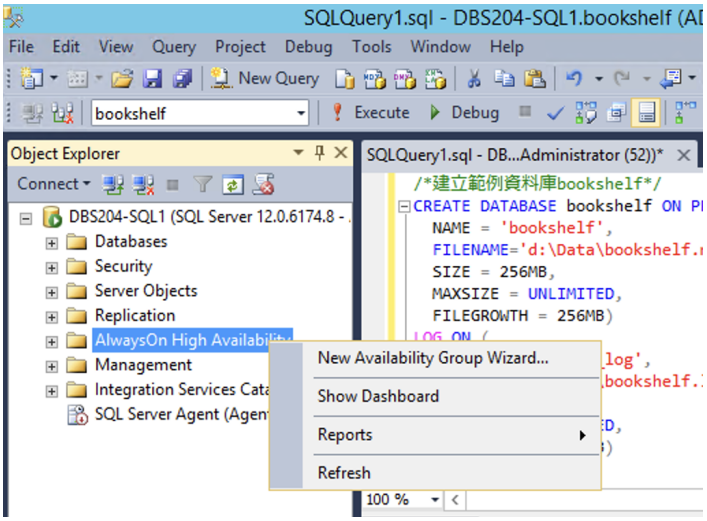
```
SQL Console>
/*建立範例資料庫bookshelf*/
CREATE DATABASE bookshelf ON PRIMARY (
  NAME = 'bookshelf',
  FILENAME='d:\Data\bookshelf.mdf',
  SIZE = 256MB,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 256MB)
LOG ON (
  NAME = 'bookshelf_log',
  FILENAME='d:\Logs\bookshelf.ldf',
  SIZE = 256MB,
  MAXSIZE = UNLIMITED,
  FILEGROWTH = 256MB)
GO

USE [bookshelf]
SET ANSI_NULLS ON
```

```
SET QUOTED_IDENTIFIER ON
GO

/*建立Table dbo*/
CREATE TABLE [dbo].[Books] (
  [Id] [bigint] IDENTITY(1,1) NOT NULL,
  [Title] [nvarchar](max) NOT NULL,
  [Author] [nvarchar](max) NULL,
  [PublishedDate] [datetime] NULL,
  [ImageUrl] [nvarchar](max) NULL,
  [Description] [nvarchar](max) NULL,
  [CreatedById] [nvarchar](max) NULL,
  CONSTRAINT [PK_dbo.Books] PRIMARY KEY CLUSTERED ([Id] ASC) WITH (
    PAD_INDEX = OFF,
    STATISTICS_NORECOMPUTE = OFF,
    IGNORE_DUP_KEY = OFF,
    ALLOW_ROW_LOCKS = ON,
    ALLOW_PAGE_LOCKS = ON) ON [PRIMARY]
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
GO

/*建立備份*/
EXEC dbo.sp_changedbowner @loginame = 'sa', @map = false;
  ALTER DATABASE [bookshelf] SET RECOVERY FULL;
  GO
  BACKUP DATABASE bookshelf to disk = '\\DBS204-VM-FSW1\Backup\bookshelf.bak' WITH INIT
GO
```

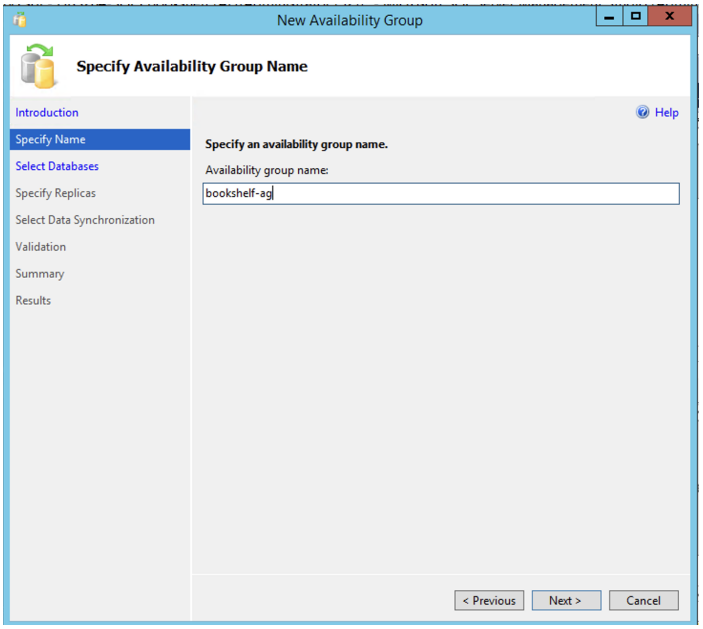
- 步驟6. 選擇 Execute按鈕來執行SQL script.

DB中配置加入Replicas，並設置高可用性

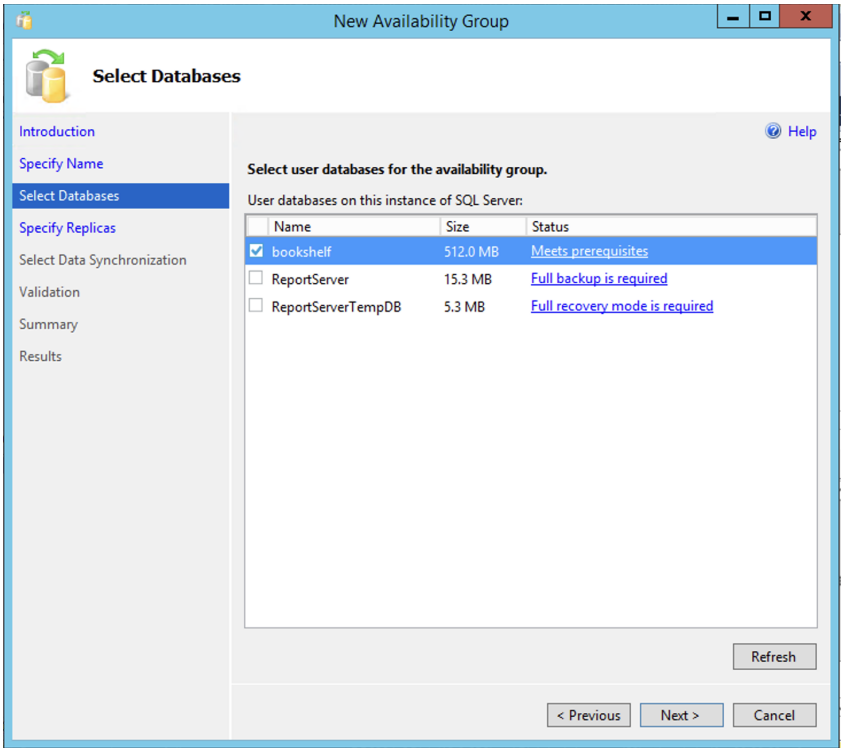
- 操作步驟
  - 步驟1.在“Object Explorer”窗口中，右鍵單擊“Always On High Availability”，然後選擇“New Availability Group Wizard”。



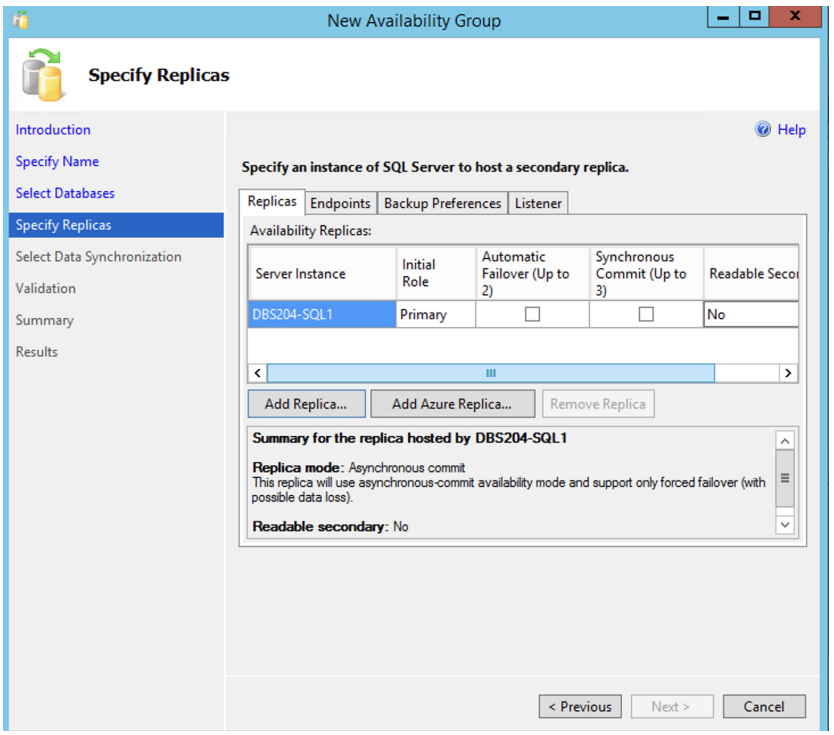
- 步驟2.在“Specify Name”頁面中，將可用性組名稱設置為 bookshelf-ag，然後選擇“下一步”。



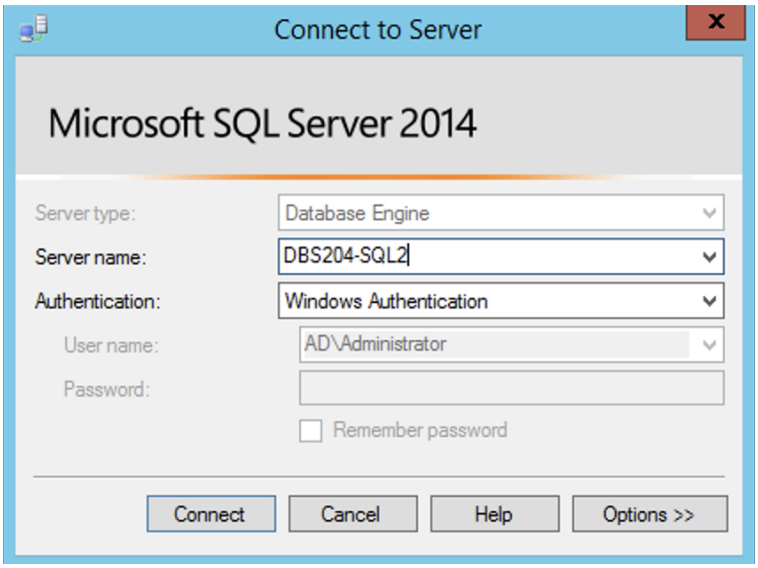
- 步驟3.在“Select Databases”頁面上，選擇 bookshelf 資料庫，然後選擇“NEXT”。



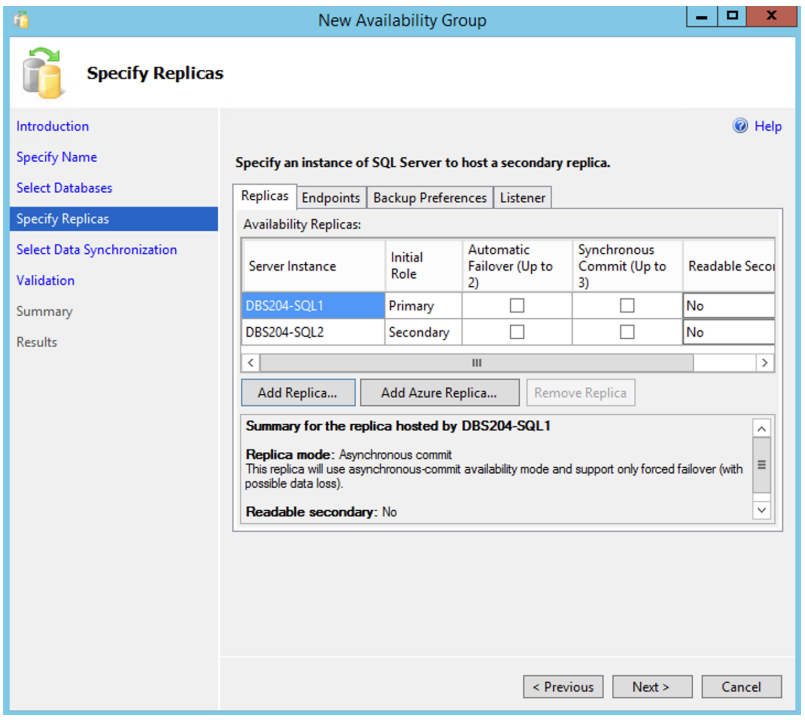
- 步驟4.在“Specify Replicas”頁面上，選擇“Replicas”tab：



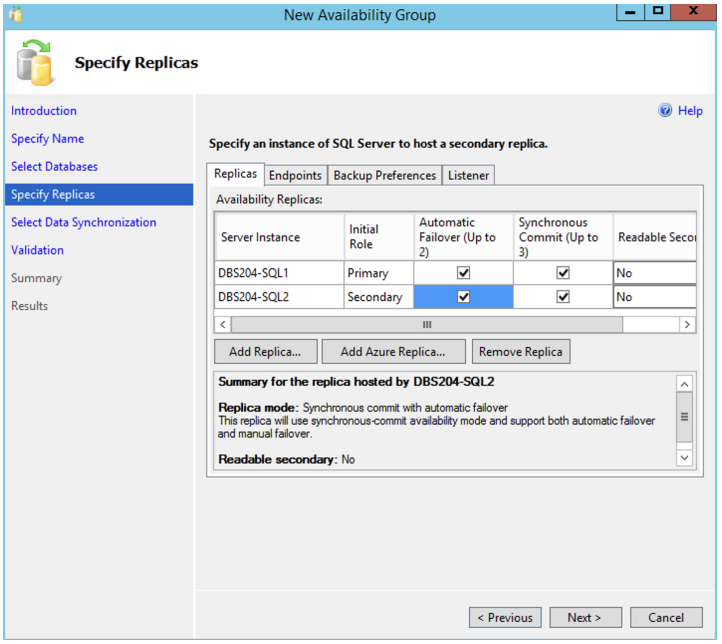
- 選擇 Add replica
- Connect to server視窗, 輸入dbs204-sql2並且按Connect.



- 接著Availability Replicas會顯示兩個可用的SQL Replica-dbs204-sql1和dbs204-sql2

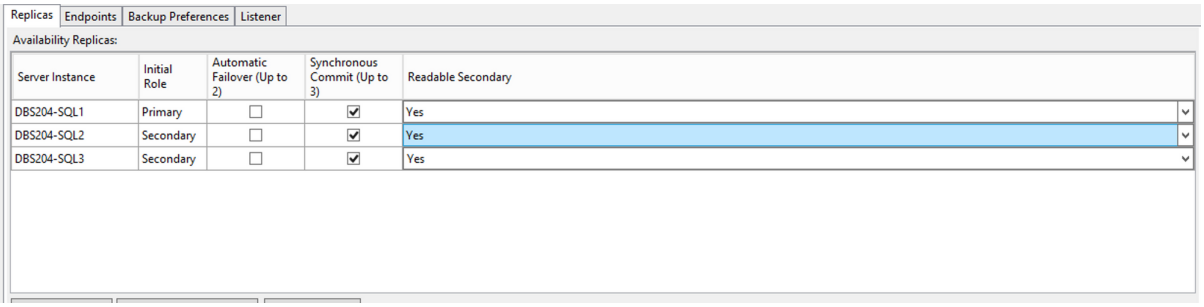


- 如果有超過兩個以上的Secondary，全部都必須填入
- 依照需求設 replicate 節點
  - 若共 2個節點可以開 auto failover, 則發生錯誤時會自動轉換到另一個 node
    - 在Replicas Tab設定兩個Replicas的Automatic failover為Enabled(勾選第三欄 - Automatic Failover (Up to 2))

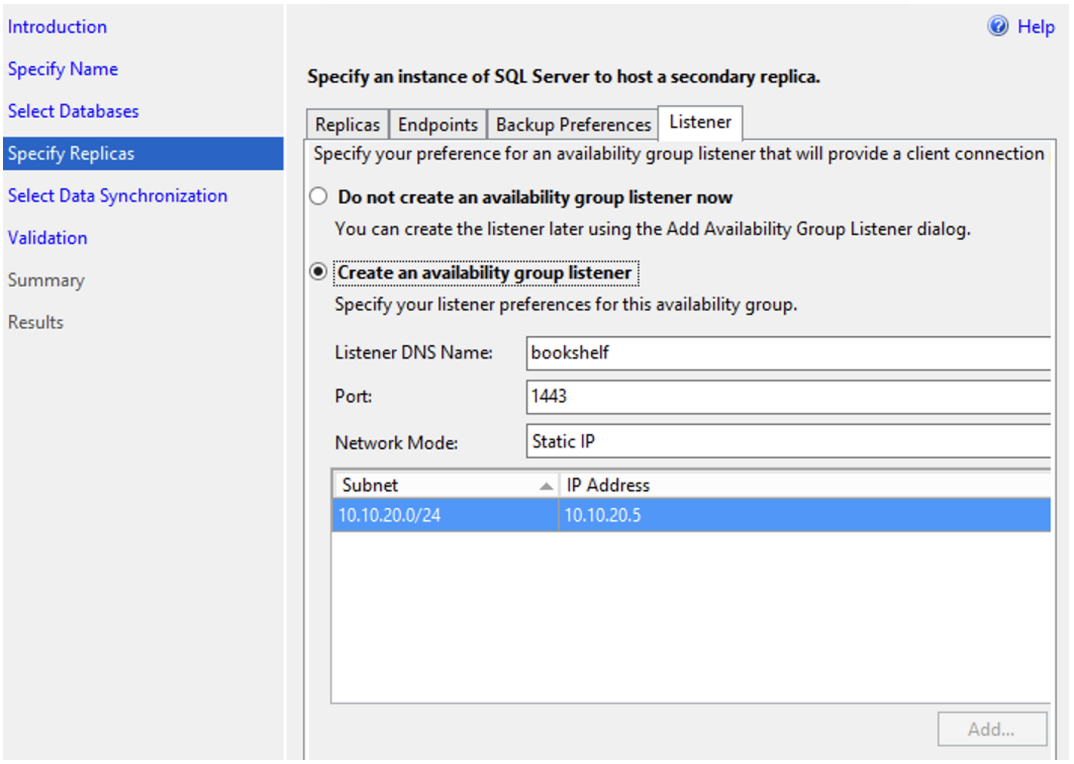


  - 若不超過 3 個 node 可以開啟 sync commit, 可以確保 3 個節點資料同步 (但相應 query 速度降低)

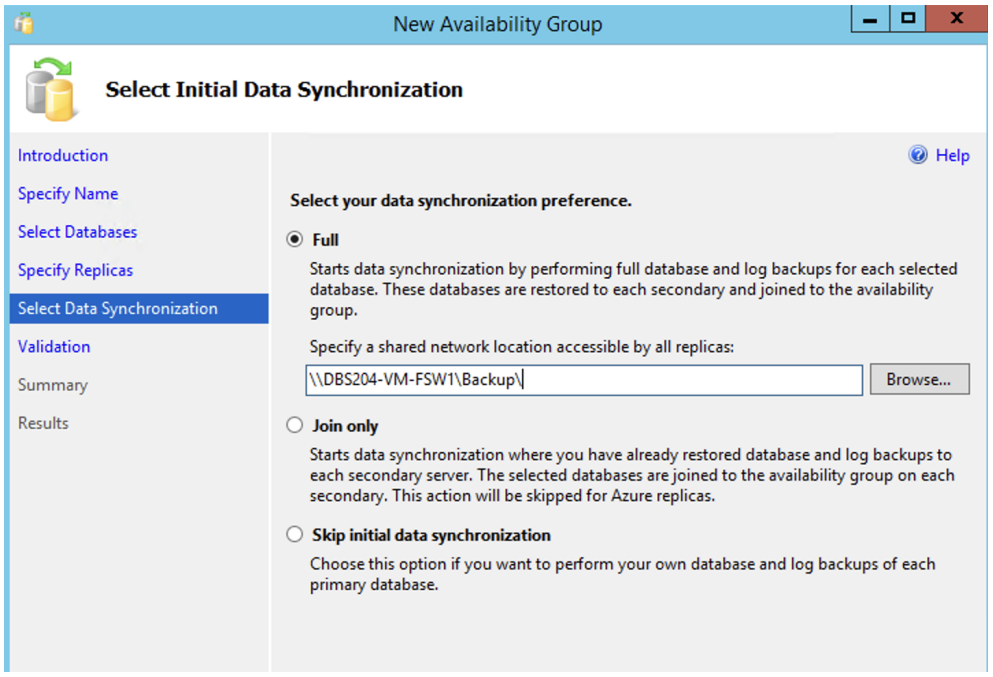
- 在Replicas Tab設定兩個Replicas的Availability mode 為Synchrounous commit(勾選第四欄 - Synchronous Commit(Up to 3)).
- 若希望可以對 secondary read (read/ write split) 此處要開啟 readable secondary



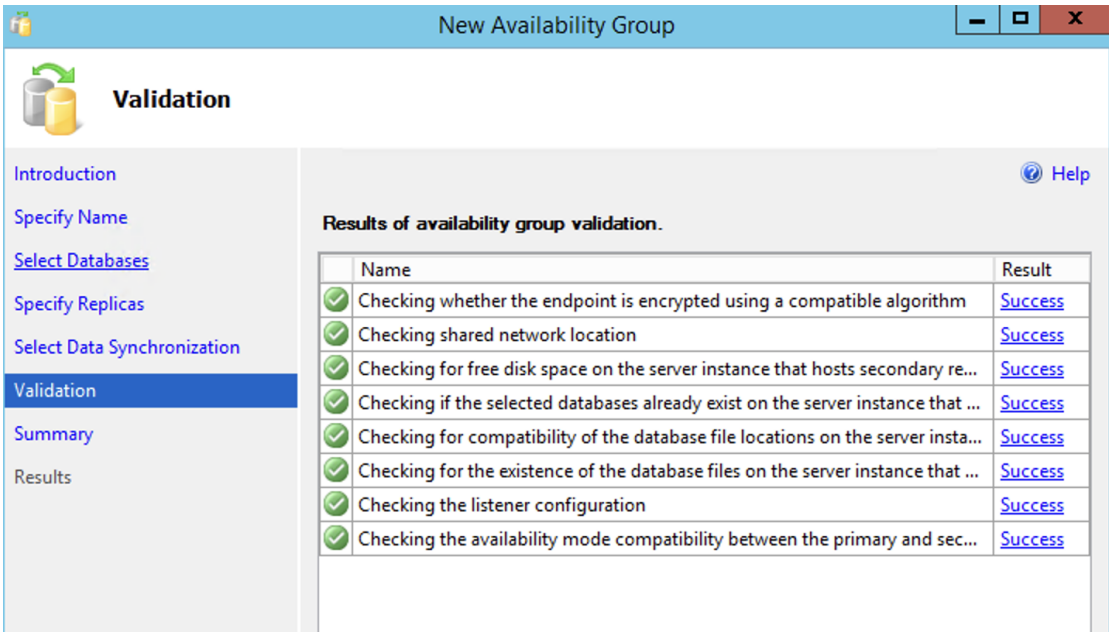
- 選擇Listener Tab
  - 選擇Create an availability group listener
  - 輸入下列設定:
    - Listener DNS name: bookshelf
    - Port: 1433
    - Network mode: Static IP
  - 選擇 Add並輸入之前設定的Load Balancer IP，並選擇按OK，接著按下Next
  - Listener所代表的是該可用性群組，因此在應用程式的連線字串，應該使用Listener的名稱作為資料庫伺服器名稱連線，當Availability Group的Failover發生時，應用程式才能對應連結到目前Activity的資料庫伺服器。



- 步驟5.在“Select Data Synchronization”頁面上，選擇“Full”。並指定位置\\DBS204-VM-FSW1\Backup\，並按Next



- 步驟6. 在Validation的頁面，確認所有Checks都是Success，接著按Next

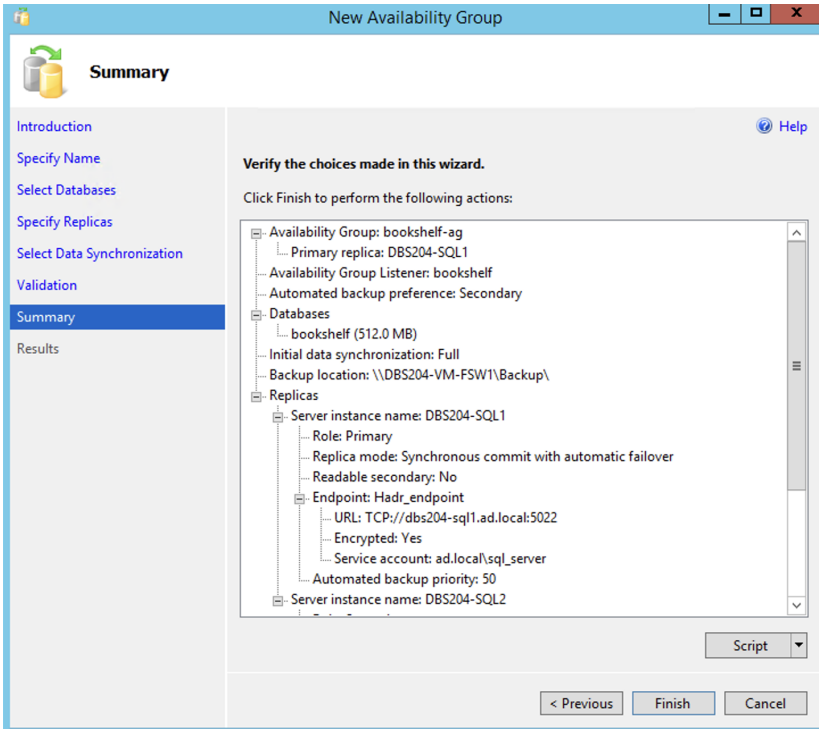


  - 倘若join 階段卡住 3 分鐘以上, 表示該 db 沒有權限, 連入該 db 後用 sql 下此指令即可
    - 在選單中，選擇 File > New > Query with current connection

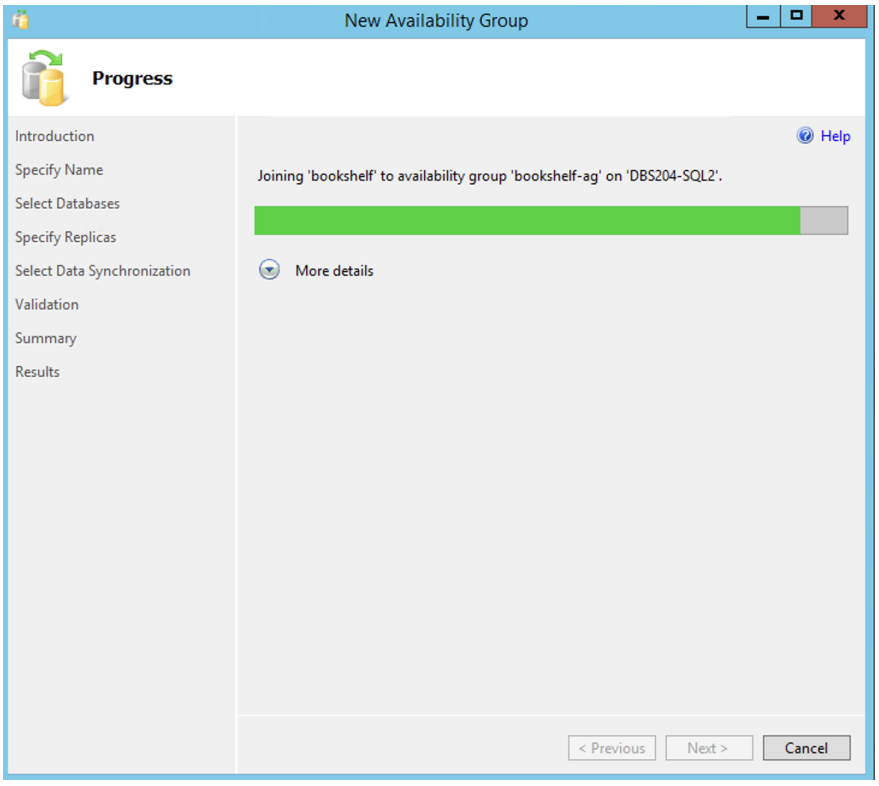
SQL Console>
USE [master]
GO
CREATE LOGIN [AD\sql_server] FROM WINDOWS WITH DEFAULT_DATABASE=[master]

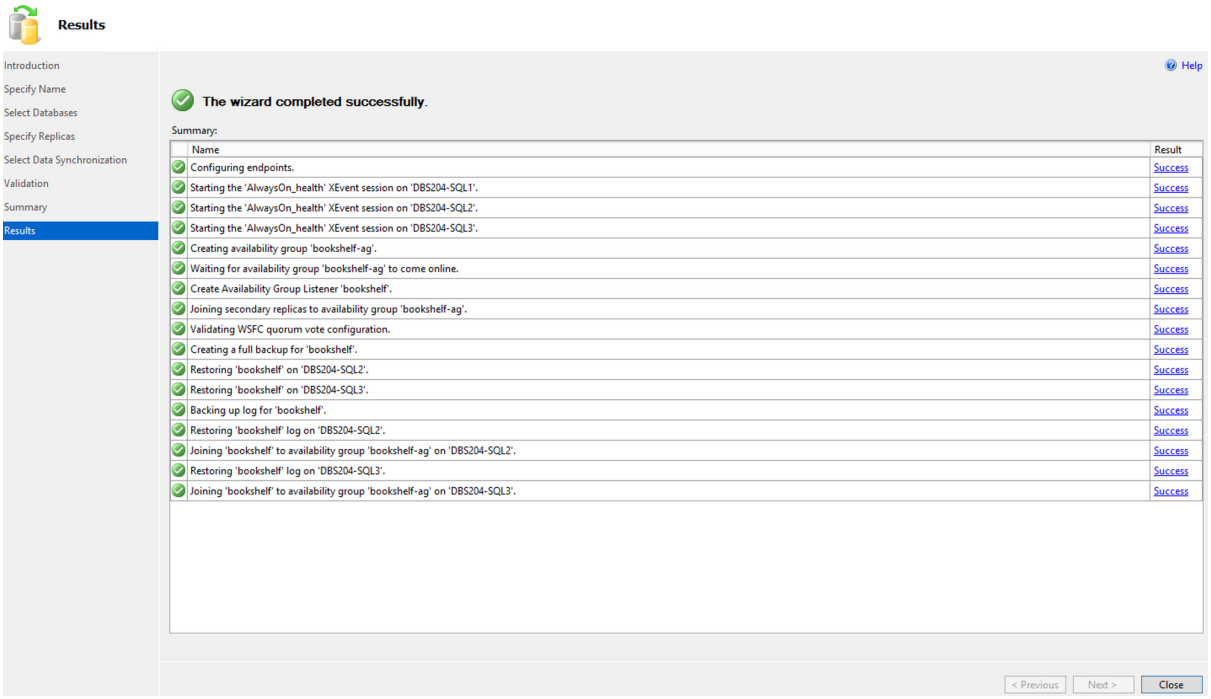
```
GO

GRANT CONNECT ON ENDPOINT::[Hadr_endpoint] TO  [AD\sql_server]
```

- 步驟7.在Summary頁面，選擇Finish

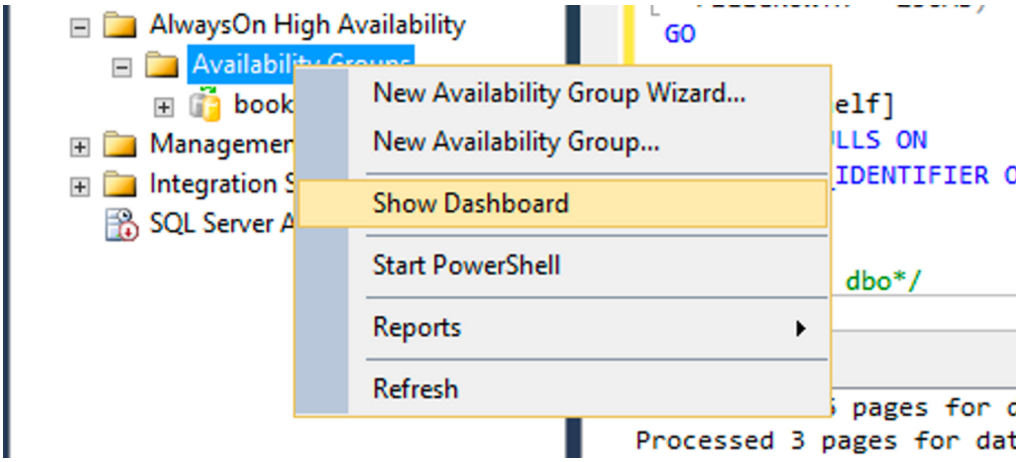


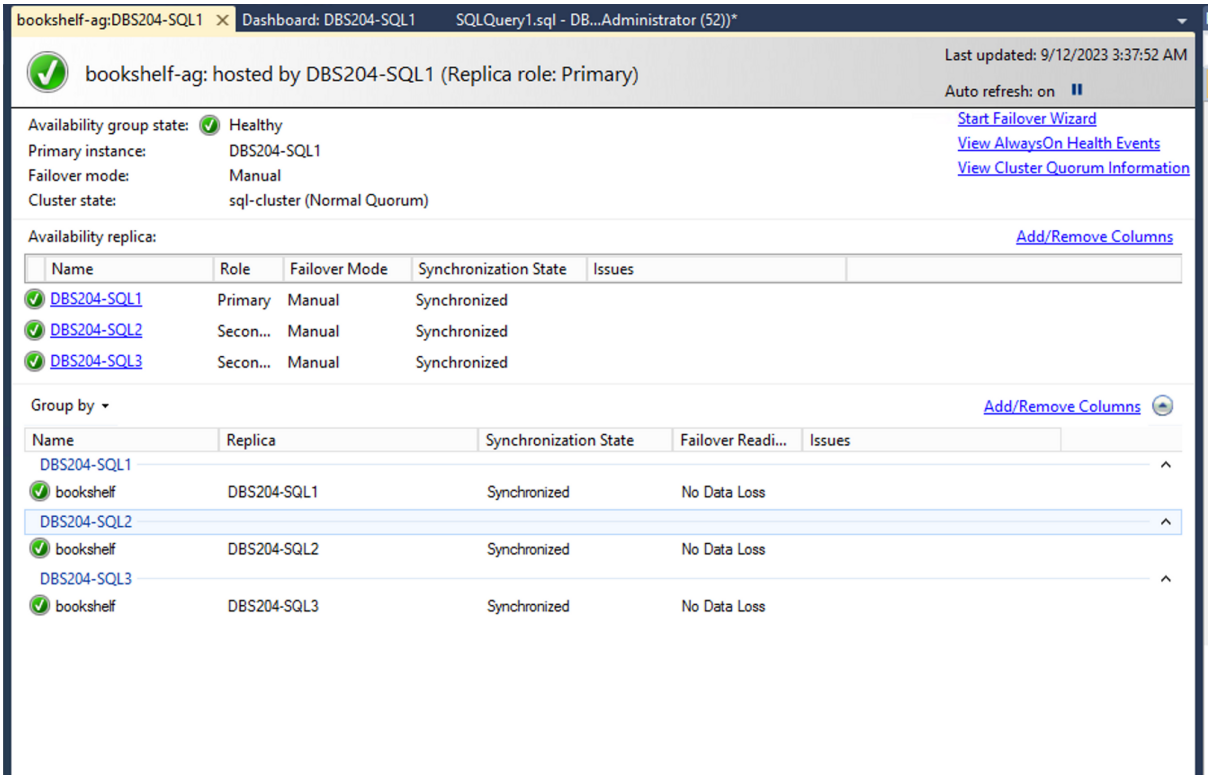
- 步驟8.在Results頁面等待執行完成，再選擇Close





- 開啟 Always-on group dashboard，檢查Cluster狀態。
  - 在“Object Explorer”窗口中，右鍵單擊“Always On High Availability”，然後選擇“Show Dashboard”。





DB中配置Health Check

- 操作步驟
  - 步驟1.返回到dbs204-sql1 上的 PowerShell
  - 步驟2.使用負載均衡器的 IP 地址初始化變量。

PowerShell>
$LoadBalancerIP = 'IP_ADDRESS'
將 IP_ADDRESS 替換為您之前保留的 wsfc 地址的 IP 地址10.10.20.10。

  - 步驟3.配置 Failover Cluster 響應健康檢查服務：

PowerShell>
```
$SqlIpAddress = Get-ClusterResource |
  Where-Object {$_.ResourceType -eq "IP Address"} |
  Where-Object {$_.Name.StartsWith("bookshelf")}

$SqlIpAddress | Set-ClusterParameter -Multiple @{
 'Address'= $LoadBalancerIP;
 'ProbePort'= 59997;
 'SubnetMask'='255.255.255.255';
 'Network'= (Get-ClusterNetwork).Name;
 'EnableDhcp'=0; }
```

  - 步驟4.重啟集群資源：

```
PowerShell>
$SqlIpAddress | Stop-ClusterResource
$SqlIpAddress | Start-ClusterResource
```

DB設定Read/Write Split(進階功能)

- 在SQL Server 2014是使用PowerShell建置路由表，但在SQL Server 2016以上可以使用GUI來做建制。
    - Specifies the address of the instance of SQL Server that is the host for an availability replica that is a readable secondary replica when running under the secondary role.
    - Use a comma-separated list to specify all of the server instances that might host a readable secondary replica. Read-only routing will follow the order in which server instances are specified in the list. If you include a replica's host server instance on the replica's read-only routing list, placing this server instance at the end of the list is typically a good practice, so that read-intent connections go to a secondary replica, if one is available.
    - Beginning with SQL Server 2016 (13.x), you can load-balance read-intent requests across readable secondary replicas. You specify this by placing the replicas in a nested set of parentheses within the read-only routing list. For more information and examples, see Configure load-balancing across read-only replicas.

- 在Primary節點開起SQL Console，並設定Read路由

```
SQL Console>
ALTER AVAILABILITY
GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL1'
WITH
(
  SECONDARY_ROLE

   (
     READ_ONLY_ROUTING_URL='TCP://dbs204-sql1.ad.local:1433'
   )
)

ALTER AVAILABILITY
GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL2'
WITH
(
  SECONDARY_ROLE

   (
     READ_ONLY_ROUTING_URL='TCP://dbs204-sql2.ad.local:1433'
```

```
    )
)

ALTER AVAILABILITY
GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL3'
WITH
(
  SECONDARY_ROLE

    (
      READ_ONLY_ROUTING_URL='TCP://dbs204-sql3.ad.local:1433'
    )
)
```

- 針對所有節點配置路由

```
SQL Console>
ALTER AVAILABILITY GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL1'
WITH
(
  PRIMARY_ROLE
  (
    READ_ONLY_ROUTING_LIST =('DBS204-SQL2', 'DBS204-SQL3')
  )
)

ALTER AVAILABILITY GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL2'
WITH
(
  PRIMARY_ROLE
  (
    READ_ONLY_ROUTING_LIST =('DBS204-SQL1', 'DBS204-SQL3')
  )
)

ALTER AVAILABILITY GROUP [bookshelf-ag]
MODIFY REPLICA
ON
'DBS204-SQL3'
WITH
(
  PRIMARY_ROLE
  (
    READ_ONLY_ROUTING_LIST =('DBS204-SQL1', 'DBS204-SQL2')
  )
)
```

- 以fsw1當作Client測試Read/Write Split是否成功運行
    - 撰寫Script測試一般的DB Connection

```
Power Shell>
while ($True){
  $Conn = New-Object System.Data.SqlClient.SqlConnection
  $Conn.ConnectionString = "Server=bookshelf;Integrated Security=true;Initial Catalog=bookshelf"
  $Conn.Open()

  $Cmd = New-Object System.Data.SqlClient.SqlCommand
  $Cmd.Connection = $Conn
  $Cmd.CommandText = "SELECT @@SERVERNAME"

  $Adapter = New-Object System.Data.SqlClient.SqlDataAdapter $Cmd
  $Data = New-Object System.Data.DataSet
  $Adapter.Fill($Data) | Out-Null
  $Data.Tables[0] + (Get-Date -Format "MM/dd/yyyy HH:mm:ss")

  Start-Sleep -Seconds 2
}
```

- 沒有設置Read-Only的Connection應該會連接到Primary

- 撰寫Script測試Read Only的DB Connection
  - 補上Connection String - ApplicationIntent=READONLY

```
PowerShell>
while ($True){
  $Conn = New-Object System.Data.SqlClient.SqlConnection
  $Conn.ConnectionString = "Server=bookshelf;Integrated Security=true;Initial
Catalog=bookshelf;ApplicationIntent=READONLY"
  $Conn.Open()

  $Cmd = New-Object System.Data.SqlClient.SqlCommand
  $Cmd.Connection = $Conn
  $Cmd.CommandText = "SELECT @@SERVERNAME"

  $Adapter = New-Object System.Data.SqlClient.SqlDataAdapter $Cmd
  $Data = New-Object System.Data.DataSet
  $Adapter.Fill($Data) | Out-Null
  $Data.Tables[0] + (Get-Date -Format "MM/dd/yyyy HH:mm:ss")

  Start-Sleep -Seconds 2
}
```

- 設置Read-Only的Connection應該會連接到Secondary

```
Column1
-------
DBS204-SQL2
09/12/2023 07:35:58
DBS204-SQL2
09/12/2023 07:36:00
DBS204-SQL2
09/12/2023 07:36:02
DBS204-SQL2
09/12/2023 07:36:04
DBS204-SQL2
09/12/2023 07:36:06
```

Internal Load Balancer

Primary和Secondary分別建立Unmanaged Instance Group

- 為了 SQL Server的Client端提供單一端點，需要部署一個內部負載平衡器 (Internal Load balancer)
- 負載均衡器使用Health Check來確保流量被定向到 WSFC 的活動節點。
    - 步驟1.返回到現在的 Cloud Shell 會話。
        - 重新登入Cloud Shell的時候，環境變數設定可能會不見，需要重新設定.

```
CloudShell>
/*設定region在asia-east1*/
export region=asia-east1

/*設定三個Zone，Zone1在asia-east-1a，Zone2在asia-east-1b，Zone3在asia-east-c*/
export zone_1=${region}-a
export zone_2=${region}-b
export zone_3=${region}-c

/*設定Project名稱*/
export project_id=lf-db-poc

/*設定VPC名稱*/
export vpc_name=dbs204-vpc

/*設定資料庫的子網域名稱*/
export sql_subnet_name=dbs204-asia-east1-database

/*設定預計使用的Region在Google Cloud CLI設定檔中*/
gcloud config set compute/region ${region}
```

    - 步驟2.建立兩個unmanaged instance groups，每個區域一個，並將兩個節點添加到組中：

```
CloudShell>
/*設定Region變數*/
REGION=$(gcloud config get-value compute/region)

/*建立Unmanaged Instance Group - wsfs-group1*/
gcloud compute instance-groups unmanaged create wsfc-group-1 --zone $REGION-a
```

```
/*分配wsfs-group1給dbs204-sql1*/
gcloud compute instance-groups unmanaged add-instances wsfc-group-1 --zone $REGION-a \
  --instances dbs204-sql1

/*建立Unmanaged Instance Group - wsfs-group2*/
gcloud compute instance-groups unmanaged create wsfc-group-2 --zone $REGION-b

/*分配wsfs-group2給dbs204-sql2*/
gcloud compute instance-groups unmanaged add-instances wsfc-group-2 --zone $REGION-b \
  --instances dbs204-sql2
```

TILB建立HealthCheck偵測節點是否Active

- 步驟1.建立負載均衡器可用於確定哪個是活動節點的運行狀況檢查。
  - –check-interval
  - –health-threshold
    - The number of consecutive successful health checks before an unhealthy instance is marked as healthy
    - 連續Check成功的次數，預設是2
  - –unhealth-threshold
    - The number of consecutive health check failures before a healthy instance is marked as unhealthy
    - 連續Check失敗的次數，預設是2
  - –timeout
    - If Google Compute Engine doesn't receive a healthy response from the instance by the time specified by the value of this flag, the health check request is considered a failure.
    - Check逾時的秒數，預設是5秒。

```
CloudShell>
gcloud compute health-checks create tcp wsfc-healthcheck \
  --check-interval="2s" \
  --healthy-threshold=1 \
  --unhealthy-threshold=2 \
  --port=59997 \
  --timeout="1s"
```

  - 運行狀況檢查探測端口 59997，這是您之前為可用性組偵聽器配置為 ProbePort 的端口。
- 步驟4.建立後端服務並添加兩個實例組：

```
CloudShell>
/*建立Backend - wsfc*/
gcloud compute backend-services create wsfc-backend \
```

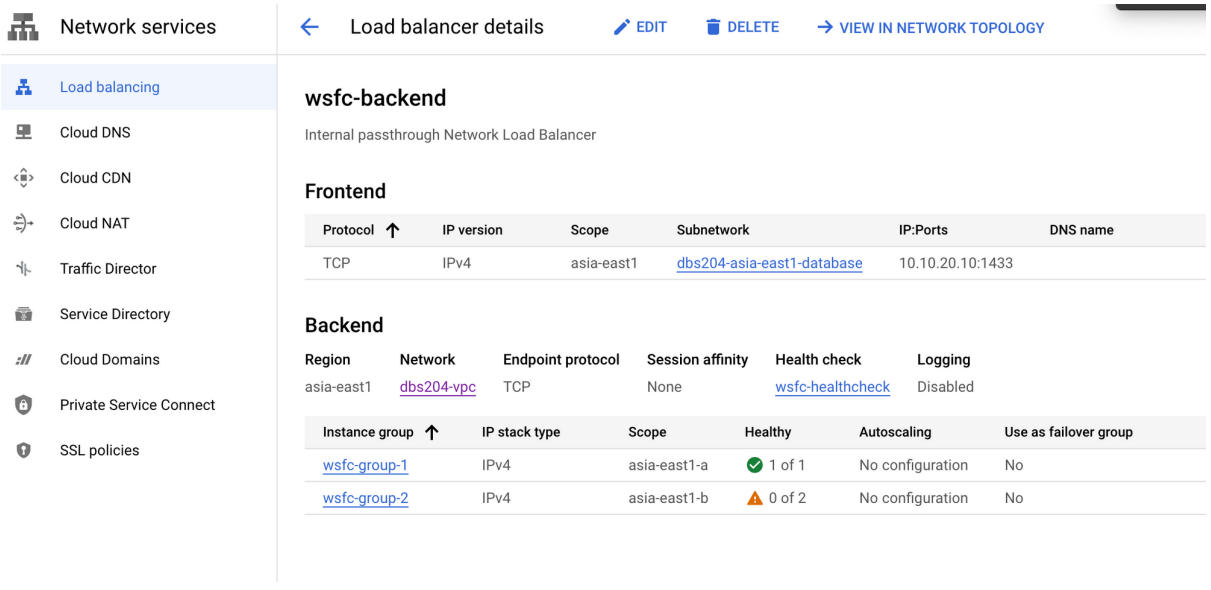
```
  --load-balancing-scheme internal \
  --region $(gcloud config get-value compute/region) \
  --health-checks wsfc-healthcheck \
  --protocol tcp

/*加入wsfc-group1到Backend - wsfc-backend*/
gcloud compute backend-services add-backend wsfc-backend \
  --instance-group wsfc-group-1 \
  --instance-group-zone $REGION-a \
  --region $REGION

/*加入wsfc-group2到Backend - wsfc-backend*/
gcloud compute backend-services add-backend wsfc-backend \
  --instance-group wsfc-group-2 \
  --instance-group-zone $REGION-b \
  --region $REGION
```

建立Internal Load Balancer

- 步驟1.建立內部負載均衡器:

```
CloudShell>
gcloud compute forwarding-rules create wsfc-sql \
  --load-balancing-scheme internal \
  --address $LOADBALANCER_ADDRESS \
  --ports 1433 \
  --network $vpc_name \
  --subnet $sql_subnet_name \
  --region $REGION \
  --backend-service wsfc-backend
```

- 從Load Balancer觀察DB節點的狀態



Terraform設定Instance/Network

編寫Terraform檔案

- 確認gcloud shell有安裝terraform
- 根據以下結構來編寫目錄和檔案
    - 建立以下目錄
        - mssql-cluster
        - mssql-cluster/instances
        - mssql-cluster/network

```
CloudShell>
mkdir mssql-cluster
mkdir mssql-cluster/instances
mkdir mssql-cluster/network
```

    - 建立以下檔案
        - mssql-cluster/Makefile
        - mssql-cluster/main.tf
        - mssql-cluster/variables.tf
        - mssql-cluster/instances/main.tf
        - mssql-cluster/instances/variables.tf
        - mssql-cluster/network/main.tf
        - mssql-cluster/network/variables.tf

```
CloudShell>
touch mssql-cluster/Makefile
touch mssql-cluster/main.tf
touch mssql-cluster/variables.tf
touch mssql-cluster/instances/main.tf
touch mssql-cluster/instances/variables.tf
touch mssql-cluster/network/main.tf
touch mssql-cluster/network/variables.tf
```

    - 確認檔案架構

- **mssql-cluster**
    - Makefile
    - main.tf
    - variables.tf
    - **instances**
        - main.tf
        - variables.tf
    - **network**
        - main.tf
        - variables.tf

撰寫Makefile

```
all:

start:
	terraform init
	terraform plan
	terraform apply

format:
	terraform fmt -recursive

init:
	terraform init

plan:
	terraform plan

destroy:
	terraform destroy
```

撰寫mssql-cluster/main.tf

- 設定兩個AD - ad1和ad2
- 設定一個Primary - db1
- 設定兩個Secondary - db2和db3
- 設定一個Witeness - fsw1
- 設定兩個Unmamaged Instance Group - wsfc-group-1和wsfc-group-2

```
provider "google" {
  project = var.project_id
  region  = var.region
}

module "network" {
  source          = "./network"
  project_id      = var.project_id
  vpc_name        = var.vpc_name
  region          = var.region
  sql_subnet_name = var.sql_subnet_name
  subnet_name     = var.subnet_name
}

module "vms" {
  source          = "./instances"
  project_id      = var.project_id
  vpc_name        = var.vpc_name
  region          = var.region
  subnet_name     = var.subnet_name
  sql_subnet_name = var.sql_subnet_name
  vm_ad1_name     = var.vm_ad1_name
  vm_ad2_name     = var.vm_ad2_name
  vm_zone1_name   = var.zone_1
  vm_zone2_name   = var.zone_2
```

```
  vm_zone3_name   = var.zone_3
  vm_db1_name     = var.vm_db1_name
  vm_db2_name     = var.vm_db2_name
  vm_db3_name     = var.vm_db3_name
  vm_fsw1         = var.vm_fsw1
  group1_name     = var.group_name1
  group2_name     = var.group_name2

  depends_on = [module.network]
}
```

撰寫mssql-cluster/variables.tf

- 修改project id為lf-db-poc

```
variable "project_id" {
  description = "The ID of the project in which to provision resources."
  type        = string
  default     = "lf-db-poc"
}
variable "vpc_name" {
  description = "The vpc_name of the project."
  type        = string
  default     = "dbs204-vpc"
}
variable "region" {
  description = "Name of the region."
  type        = string
  default     = "asia-east1"
}
variable "subnet_name" {
  description = "Name of the subnet_name have AD."
  type        = string
  default     = "dbs204-asia-east1-config"
}
variable "sql_subnet_name" {
  description = "Name of the subnet_name have db."
  type        = string
  default     = "dbs204-asia-east1-database"
}
variable "zone_1" {
  description = "Name of the zone_1."
  type        = string
  default     = "asia-east1-a"
}
variable "zone_2" {
  description = "Name of the zone_2."
  type        = string
  default     = "asia-east1-b"
}
variable "zone_3" {
  description = "Name of the zone_3."
  type        = string
```

```
  default     = "asia-east1-c"
}
variable "vm_ad1_name" {
  description = "Name of the ad1."
  type        = string
  default     = "dbs204-vm-ad1"
}
variable "vm_ad2_name" {
  description = "Name of the ad2."
  type        = string
  default     = "dbs204-vm-ad2"
}
variable "vm_db1_name" {
  description = "Name of the db1."
  type        = string
  default     = "dbs204-sql1"
}
variable "vm_db2_name" {
  description = "Name of the db2."
  type        = string
  default     = "dbs204-sql2"
}
variable "vm_db3_name" {
  description = "Name of the db3."
  type        = string
  default     = "dbs204-sql3"
}
variable "group_name1" {
  description = "Name of the instance group1."
  type        = string
  default     = "wsfc-group-1"
}
variable "group_name2" {
  description = "Name of the instance group2."
  type        = string
  default     = "wsfc-group-2"
}
variable "vm_fsw1" {
  description = "Name of the fsw1."
  type        = string
  default     = "dbs204-vm-fsw1"
}
```

撰寫mssql-cluster/instances/main.tf

```
resource "google_compute_instance" "ad1" {
  boot_disk {
    auto_delete = true
    device_name = var.vm_ad1_name

    initialize_params {
      image = "projects/windows-cloud/global/images/windows-server-2019-dc-v20230809"
```

```
    size  = 50
    type  = "pd-balanced"
   }

   mode = "READ_WRITE"
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false

  labels = {
   goog-ec-src = "vm_add-tf"
  }

  machine_type = "n1-standard-2"
  name         = var.vm_ad1_name

  network_interface {
   access_config {
    network_tier = "PREMIUM"
   }

   network_ip = "10.10.10.2"
   subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.subnet_name}"
  }

  scheduling {
   automatic_restart   = true
   on_host_maintenance = "MIGRATE"
   preemptible         = false
   provisioning_model  = "STANDARD"
  }

  shielded_instance_config {
   enable_integrity_monitoring = true
   enable_secure_boot          = true
   enable_vtpm                 = true
  }

  tags = ["ad", "rdp"]
  zone = var.vm_zone1_name
}

resource "google_compute_instance" "ad2" {
  boot_disk {
   auto_delete = true
   device_name = var.vm_ad2_name

   initialize_params {
    image = "projects/windows-cloud/global/images/windows-server-2019-dc-v20230809"
    size  = 50
    type  = "pd-balanced"
```

```
    }

    mode = "READ_WRITE"
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false

  labels = {
    goog-ec-src = "vm_add-tf"
  }

  machine_type = "n1-standard-2"
  name         = var.vm_ad2_name

  network_interface {
    access_config {
      network_tier = "PREMIUM"
    }

    network_ip = "10.10.10.3"
    subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.subnet_name}"
  }

  scheduling {
    automatic_restart   = true
    on_host_maintenance = "MIGRATE"
    preemptible         = false
    provisioning_model  = "STANDARD"
  }

  shielded_instance_config {
    enable_integrity_monitoring = true
    enable_secure_boot          = true
    enable_vtpm                 = true
  }

  tags = ["ad", "rdp"]
  zone = var.vm_zone2_name
}

resource "google_compute_disk" "data-disk1" {
  name = "${var.vm_db1_name}-datadisk"
  type = "pd-ssd"
  zone = var.vm_zone1_name
  size = 200

  physical_block_size_bytes = 4096
}

resource "google_compute_disk" "data-disk2" {
  name = "${var.vm_db2_name}-datadisk"
```

```
  type = "pd-ssd"
  zone = var.vm_zone2_name
  size = 200

  physical_block_size_bytes = 4096
}

resource "google_compute_disk" "data-disk3" {
  name = "${var.vm_db3_name}-datadisk"
  type = "pd-ssd"
  zone = var.vm_zone2_name
  size = 200

  physical_block_size_bytes = 4096
}

resource "google_compute_instance" "db1" {
  depends_on = [google_compute_disk.data-disk1]

  attached_disk {
    source = google_compute_disk.data-disk1.self_link
    mode   = "READ_WRITE"
  }

  metadata = {
    enable-wsfc                  = "true"
    sysprep-specialize-script-ps1 = "$ErrorActionPreference = \"stop\"\n\n# Install required
Windows features\nInstall-WindowsFeature Failover-Clustering
-IncludeManagementTools\nInstall-WindowsFeature RSAT-AD-PowerShell\n\n# Open
firewall for WSFC\nnetsh advfirewall firewall add rule name=\"Allow SQL Server health
check\" dir=in action=allow protocol=TCP localport=59997\n\n# Open firewall for SQL
Server\nnetsh advfirewall firewall add rule name=\"Allow SQL Server\" dir=in action=allow
protocol=TCP localport=1433\n\n# Open firewall for SQL Server replication\nnetsh
advfirewall firewall add rule name=\"Allow SQL Server replication\" dir=in action=allow
protocol=TCP localport=5022\n\n# Format data disk\nGet-Disk |\n Where partitionstyle -eq
'RAW' |\n Initialize-Disk -PartitionStyle MBR -PassThru |\n New-Partition
-AssignDriveLetter -UseMaximumSize |\n Format-Volume -FileSystem NTFS
-NewFileSystemLabel 'Data' -Confirm:$false\n\n# Create data and log folders for SQL
Server\nmd d:\\Data\nmd d:\\Logs"
  }

  boot_disk {
    auto_delete = true
    device_name = var.vm_db1_name

    initialize_params {
      image =
"projects/windows-sql-cloud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v2023
0809"
      size  = 50
      type  = "pd-balanced"
    }

    mode = "READ_WRITE"
```

```
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false

  labels = {
    goog-ec-src = "vm_add-tf"
  }

  machine_type = "n1-standard-1"
  name         = var.vm_db1_name

  network_interface {
    access_config {
      network_tier = "PREMIUM"
    }

    network_ip = "10.10.20.2"
    subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.sql_subnet_name}"
  }

  scheduling {
    automatic_restart   = true
    on_host_maintenance = "MIGRATE"
    preemptible         = false
    provisioning_model  = "STANDARD"
  }

  shielded_instance_config {
    enable_integrity_monitoring = true
    enable_secure_boot          = true
    enable_vtpm                 = true
  }

  tags = ["ad-member", "rdp", "sql", "wsfc", "wsfc-node"]
  zone = var.vm_zone1_name
}

resource "google_compute_instance" "db2" {
  depends_on = [google_compute_disk.data-disk2]

  attached_disk {
    source = google_compute_disk.data-disk2.self_link
    mode   = "READ_WRITE"
  }

  metadata = {
    enable-wsfc                  = "true"
    sysprep-specialize-script-ps1 = "$ErrorActionPreference = \"stop\"\n\n# Install required
Windows features\nInstall-WindowsFeature Failover-Clustering
-IncludeManagementTools\nInstall-WindowsFeature RSAT-AD-PowerShell\n\n# Open
firewall for WSFC\nnetsh advfirewall firewall add rule name=\"Allow SQL Server health
```

```
check\" dir=in action=allow protocol=TCP localport=59997\n\n# Open firewall for SQL
Server\nnetsh advfirewall firewall add rule name=\"Allow SQL Server\" dir=in action=allow
protocol=TCP localport=1433\n\n# Open firewall for SQL Server replication\nnetsh
advfirewall firewall add rule name=\"Allow SQL Server replication\" dir=in action=allow
protocol=TCP localport=5022\n\n# Format data disk\nGet-Disk |\n Where partitionstyle -eq
'RAW' |\n Initialize-Disk -PartitionStyle MBR -PassThru |\n New-Partition
-AssignDriveLetter -UseMaximumSize |\n Format-Volume -FileSystem NTFS
-NewFileSystemLabel 'Data' -Confirm:$false\n\n# Create data and log folders for SQL
Server\nmd d:\\Data\nmd d:\\Logs"
  }

  boot_disk {
    auto_delete = true
    device_name = var.vm_db2_name

    initialize_params {
      image =
"projects/windows-sql-cloud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v2023
0809"
      size  = 50
      type  = "pd-balanced"
    }

    mode = "READ_WRITE"
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false

  labels = {
    goog-ec-src = "vm_add-tf"
  }

  machine_type = "n1-standard-1"
  name         = var.vm_db2_name

  network_interface {
    access_config {
      network_tier = "PREMIUM"
    }

    network_ip = "10.10.20.3"
    subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.sql_subnet_name}"
  }

  scheduling {
    automatic_restart   = true
    on_host_maintenance = "MIGRATE"
    preemptible         = false
    provisioning_model  = "STANDARD"
  }
```

```
  shielded_instance_config {
    enable_integrity_monitoring = true
    enable_secure_boot          = true
    enable_vtpm                 = true
  }

  tags = ["ad-member", "rdp", "sql", "wsfc", "wsfc-node"]
  zone = var.vm_zone2_name
}

resource "google_compute_instance" "db3" {
  depends_on = [google_compute_disk.data-disk3]

  attached_disk {
    source = google_compute_disk.data-disk3.self_link
    mode   = "READ_WRITE"
  }

  metadata = {
    enable-wsfc                  = "true"
    sysprep-specialize-script-ps1 = "$ErrorActionPreference = \"stop\"\n\n# Install required
Windows features\nInstall-WindowsFeature Failover-Clustering
-IncludeManagementTools\nInstall-WindowsFeature RSAT-AD-PowerShell\n\n# Open
firewall for WSFC\nnetsh advfirewall firewall add rule name=\"Allow SQL Server health
check\" dir=in action=allow protocol=TCP localport=59997\n\n# Open firewall for SQL
Server\nnetsh advfirewall firewall add rule name=\"Allow SQL Server\" dir=in action=allow
protocol=TCP localport=1433\n\n# Open firewall for SQL Server replication\nnetsh
advfirewall firewall add rule name=\"Allow SQL Server replication\" dir=in action=allow
protocol=TCP localport=5022\n\n# Format data disk\nGet-Disk |\n Where partitionstyle -eq
'RAW' |\n Initialize-Disk -PartitionStyle MBR -PassThru |\n New-Partition
-AssignDriveLetter -UseMaximumSize |\n Format-Volume -FileSystem NTFS
-NewFileSystemLabel 'Data' -Confirm:$false\n\n# Create data and log folders for SQL
Server\nmd d:\\Data\nmd d:\\Logs"
  }

  boot_disk {
    auto_delete = true
    device_name = var.vm_db3_name

    initialize_params {
      image =
"projects/windows-sql-cloud/global/images/sql-2014-enterprise-windows-2012-r2-dc-v2023
0809"
      size  = 50
      type  = "pd-balanced"
    }

    mode = "READ_WRITE"
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false
```

```
  labels = {
    goog-ec-src = "vm_add-tf"
  }

  machine_type = "n1-standard-1"
  name         = var.vm_db3_name

  network_interface {
    access_config {
      network_tier = "PREMIUM"
    }

    network_ip = "10.10.20.5"
    subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.sql_subnet_name}"
  }

  scheduling {
    automatic_restart   = true
    on_host_maintenance = "MIGRATE"
    preemptible         = false
    provisioning_model  = "STANDARD"
  }

  shielded_instance_config {
    enable_integrity_monitoring = true
    enable_secure_boot          = true
    enable_vtpm                 = true
  }

  tags = ["ad-member", "rdp", "sql", "wsfc", "wsfc-node"]
  zone = var.vm_zone2_name
}

resource "google_compute_instance" "fsw1" {
  boot_disk {
    auto_delete = true
    device_name = var.vm_fsw1

    initialize_params {
      image = "projects/windows-cloud/global/images/windows-server-2019-dc-v20230809"
      size  = 50
      type  = "pd-ssd"
    }

    mode = "READ_WRITE"
  }

  can_ip_forward      = false
  deletion_protection = false
  enable_display      = false

  labels = {
    goog-ec-src = "vm_add-tf"
```

```
  }

  machine_type = "n1-standard-1"

  metadata = {
    sysprep-specialize-script-ps1 = "add-windowsfeature FS-FileServer"
  }

  name = var.vm_fsw1

  network_interface {
    access_config {
      network_tier = "PREMIUM"
    }

    network_ip = "10.10.20.4"
    subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.sql_subnet_name}"
  }

  scheduling {
    automatic_restart   = true
    on_host_maintenance = "MIGRATE"
    preemptible         = false
    provisioning_model  = "STANDARD"
  }

  shielded_instance_config {
    enable_integrity_monitoring = true
    enable_secure_boot          = false
    enable_vtpm                 = true
  }

  tags = ["ad-member", "fsw", "rdp", "wsfc"]
  zone = var.vm_zone3_name
}

resource "google_compute_instance_group" "group1" {
  depends_on = [google_compute_instance.db1]
  name       = var.group1_name
  zone       = var.vm_zone1_name
  instances  = [google_compute_instance.db1.self_link]

  lifecycle {
    create_before_destroy = true
  }
}

resource "google_compute_instance_group" "group2" {
  depends_on = [google_compute_instance.db2, google_compute_instance.db3]
  name       = var.group2_name
  zone       = var.vm_zone2_name
  instances  = [google_compute_instance.db2.self_link,
google_compute_instance.db3.self_link]
```

```
  lifecycle {
    create_before_destroy = true
  }
}

resource "google_compute_health_check" "tcp-health-check" {
  name = "wsfc-healthcheck"

  timeout_sec        = 1
  check_interval_sec  = 2
  healthy_threshold   = 1
  unhealthy_threshold = 2

  tcp_health_check {
    port = 59997
  }
}

# backend service
resource "google_compute_region_backend_service" "groups" {
  depends_on            = [google_compute_health_check.tcp-health-check]
  name                  = "wsfc-backend"
  region                = var.region
  protocol              = "TCP"
  load_balancing_scheme = "INTERNAL"
  health_checks         = [google_compute_health_check.tcp-health-check.id]
  backend {
    group = google_compute_instance_group.group1.id
  }
  backend {
    group = google_compute_instance_group.group2.id
  }
}

# frontend
resource "google_compute_forwarding_rule" "google_compute_forwarding_rule" {
  name                  = "forwarding-rule"
  region                = var.region
  ip_address            = "10.10.20.10"
  ip_protocol           = "TCP"
  load_balancing_scheme = "INTERNAL"
  ports                 = ["1433"]
  backend_service       = google_compute_region_backend_service.groups.id
  network               = var.vpc_name
  subnetwork            = var.sql_subnet_name

  depends_on = [google_compute_region_backend_service.groups]
}
```

撰寫mssql-cluster/instances/variables.tf

```
variable "project_id" {
  description = "The ID of the project in which to provision resources."
  type        = string
}
variable "vpc_name" {
  description = "The vpc_name of the project."
  type        = string
}
variable "region" {
  description = "Name of the region."
  type        = string
}
variable "subnet_name" {
  description = "Name of the subnet_name have AD."
  type        = string
}
variable "sql_subnet_name" {
  description = "Name of the subnet_name have db."
  type        = string
}
variable "vm_ad1_name" {
  description = "Name of the ad1."
  type        = string
}
variable "vm_ad2_name" {
  description = "Name of the ad2."
  type        = string
}
variable "vm_zone1_name" {
  description = "Name of the zone1."
  type        = string
}
variable "vm_zone2_name" {
  description = "Name of the zone2."
  type        = string
}
variable "vm_zone3_name" {
  description = "Name of the zone3."
  type        = string
}
variable "vm_db1_name" {
  description = "Name of the db1."
  type        = string
}
variable "vm_db2_name" {
  description = "Name of the db2."
  type        = string
}
variable "vm_db3_name" {
  description = "Name of the db3."
  type        = string
}
variable "vm_fsw1" {
  description = "Name of the fsw1."
```

```
  type        = string
}
variable "group1_name" {
  description = "Name of the group1."
  type        = string
}
variable "group2_name" {
  description = "Name of the group2."
  type        = string
}
```

撰寫mssql-cluster/network/main.tf

```
module "vpc" {
  source       = "terraform-google-modules/network/google"
  description  = "VPC network to deploy Active Directory"
  version      = "~> 7.0"
  project_id   = var.project_id
  network_name = var.vpc_name
  mtu          = 1460

  subnets = [
    {
      subnet_name           = var.subnet_name
      subnet_ip             = "10.10.10.0/24"
      subnet_region         = var.region
      subnet_private_access = "true"
    },
    {
      subnet_name           = var.sql_subnet_name
      subnet_ip             = "10.10.20.0/24"
      subnet_region         = var.region
      subnet_private_access = "true"
    }
  ]
}

module "firewall_rules" {
  source       = "terraform-google-modules/network/google//modules/firewall-rules"
  project_id   = var.project_id
  network_name = module.vpc.network_name
  depends_on   = [module.vpc]

  rules = [{
    name        = "dbs204-fw-ad-controller"
    description = "allow ad connect ad"
    direction   = "INGRESS"
    priority    = 1000
    source_tags = ["ad"]
    target_tags = ["ad"]
    allow = [{
      protocol = "all"
```

```
}]
}, {
name          = "dbs204-fw-ad-member"
description   = "allow others connect ad"
direction     = "INGRESS"
priority      = 1000
source_ranges = ["10.10.10.0/24", "10.10.20.0/24"]
target_tags   = ["ad", "ad-member"]
allow = [{
  protocol = "all"
}]
}, {
name        = "dbs204-fw-sql"
description = "allow sql connect sql"
direction   = "INGRESS"
priority    = 1000
source_tags = ["sql"]
target_tags = ["sql"]
allow = [{
  protocol = "all"
}]
}, {
name        = "dbs204-fw-fs"
description = "allow sql connect fsw"
direction   = "INGRESS"
priority    = 1000
source_tags = ["sql"]
target_tags = ["sql"]
allow = [{
  protocol = "tcp"
  ports    = ["445"]
}]
}, {
name          = "dbs204-fw-sql-client"
description   = "allow client connect sql"
direction     = "INGRESS"
priority      = 1000
source_ranges = ["0.0.0.0/0"] # should change to client ip in prod
target_tags   = ["sql"]
allow = [{
  protocol = "tcp"
  ports    = ["1433"]
}]
}, {
name          = "allow-rdp"
description   = "allow client connect sql"
direction     = "INGRESS"
priority      = 1000
source_ranges = ["0.0.0.0/0"] # should change to client ip in prod
target_tags   = ["rdp"]
allow = [{
  protocol = "tcp"
  ports    = ["3389"]
}] }, {
```

```
    name        = "dbs204-fw-allow-all-between-wsfc-nodes"
    description = "wsfc connect wsfc"
    direction   = "INGRESS"
    priority    = 10000
    source_tags = ["wsfc"]
    target_tags = ["wsfc"]
    allow = [{
      protocol = "tcp"
      }, {
      protocol = "udp"
      }, {
      protocol = "icmp"
    }]
    log_config = {
      metadata = "INCLUDE_ALL_METADATA"
    }
    }, {
    name          = "dbs204-fw-allow-sql-to-wsfc-nodes"
    description   = "db connect wsfc"
    direction     = "INGRESS"
    priority      = 10000
    source_ranges =
[module.vpc.subnets["${var.region}/${var.sql_subnet_name}"].ip_cidr_range]
    target_tags   = ["wsfc-node"]
    allow = [{
      protocol = "tcp"
      ports    = ["1433"]
    }]
    log_config = {
      metadata = "INCLUDE_ALL_METADATA"
    }
    }, {
    name          = "dbs204-fw-allow-health-check-to-wsfc-nodes"
    description   = "health check"
    direction     = "INGRESS"
    priority      = 10000
    source_ranges = ["130.211.0.0/22", "35.191.0.0/16"] // gcp health care ip
    target_tags   = ["wsfc-node"]
    allow = [{
      protocol = "tcp"
    }]
  }]
}

module "address" {
  source     = "terraform-google-modules/address/google"
  version    = "~> 3.1"
  project_id = var.project_id # Replace this with your project ID in quotes
  region     = var.region
  subnetwork =
"projects/${var.project_id}/regions/${var.region}/subnetworks/${var.sql_subnet_name}"
  names      = ["wsfc", "wsfc-cluster"]
  addresses  = ["10.10.20.10", "10.10.20.11"]
  depends_on = [module.vpc]
```

```
}
```

撰寫mssql-cluster/network/variables.tf

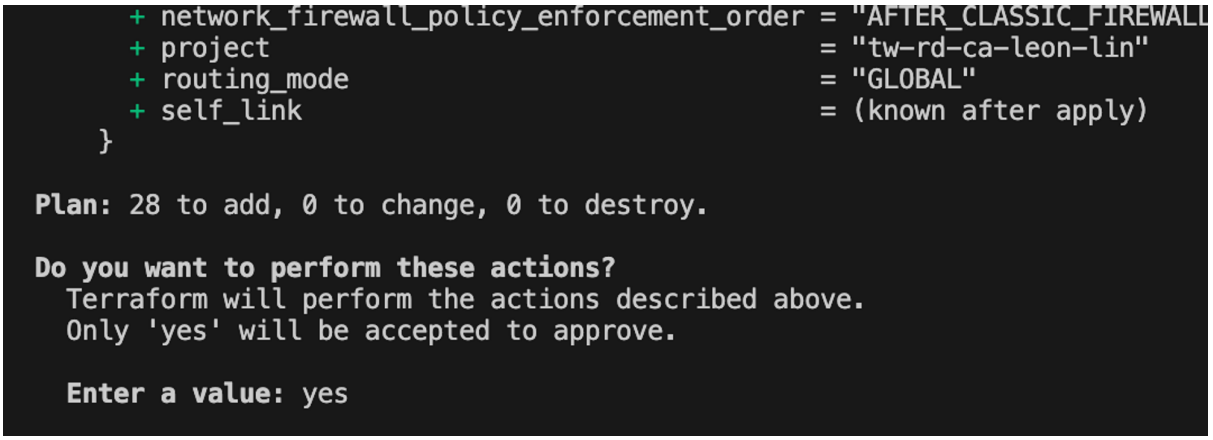
```
variable "project_id" {
  description = "The ID of the project in which to provision resources."
  type        = string
}
variable "vpc_name" {
  description = "The vpc_name of the project."
  type        = string
}
variable "region" {
  description = "Name of the region."
  type        = string
}
variable "subnet_name" {
  description = "Name of the subnet_name have AD."
  type        = string
}
variable "sql_subnet_name" {
  description = "Name of the subnet_name have db."
  type        = string
}
```

執行Terraform去建立Instance和Network

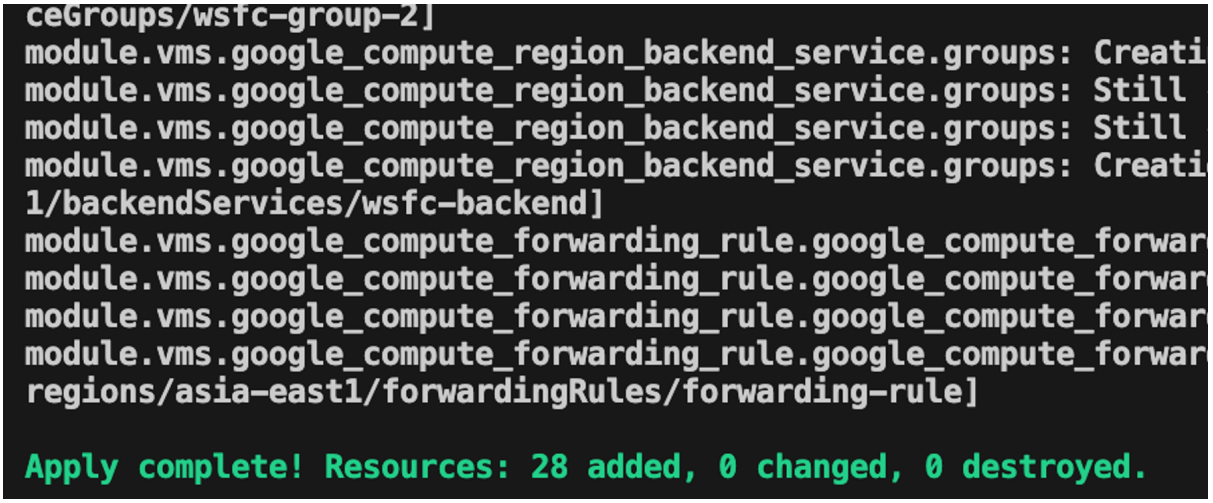
- 執行以下make指令來建立VM，設定防火牆和網路。

CloudShell>
make start

- Terraform會跳出詢問視窗，輸入yes。



- 結束時沒顯示錯誤，便表示創建成功。



- 如果顯示錯誤，則使用以下指令來移除VM和網路設定，並且根據錯誤顯示來除錯。

CloudShell>
make destroy

根據前述步驟設定Domain Controller和SQL AlwaysON

- VM，防火牆，Load Balancer皆已建立，參考P7-P14的**"設定Activity Directory來做高可用的Domain Controller"**步驟，設定AD1和AD2當作Domain Controller。
- 參考P18-P19的**"將VM加入AD的Domain管理"**將sql1，sql2，sql3和fsw1來加入AD的Domain管理。
- 參考P20-P32的**"SQL Server設定"**將sql1，sql2，sql3和fsw1設定SQL AlwaysOn。

FAQ

Q1. 這份文件的**SQL AlwaysOn Availabile Group**設定是針對**Asynchronous-commit mode**，**Synchronous-commit mode**，或**Configuration only mode**的哪一種模式？
答：
這邊是針對Synchronous-commit mode的方式做設定。

Q2. 為什麼**Listener**是設定**Internal Load Balancer**的**IP?**
答：
根據文件描述：
在地端，可以讓WSFC使用ARP去做health check。但在GCP用ILB取代這個機制。
In an on-premises environment, you can let WSFC perform ARP announcements if a failover occurs to notify network equipment about an IP address change. Google Cloud ignores ARP announcements and you must use an internal load balancer instead.
Listener所代表的是該可用性群組，因此在應用程式的連線字串，應該使用Listener的名稱作為資料庫伺服器名稱連線，當Availability Group的Failover發生時，應用程式才能對應連結到目前Activity的資料庫伺服器。

Q**3.** 這個**Internal Load Balancer**的功用為何？
答：

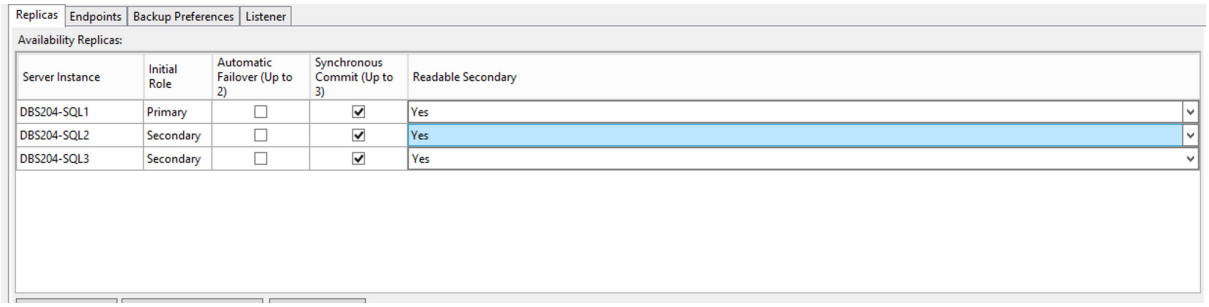
根據文件描述:
ILB就是做Health Check和當做流量的入口。
The load balancer does the following:

- Periodically performs a health check to identify the currently active WSFC node
- Clients connect to the IP address of the load balancer
- Forwards client traffic to the currently active WSFC node

Q4. 在SQL Server AlwaysOn with Single Subnet下要怎麼設定read/write split?
答:
在使用ILB的情況下要做到Read/Write Split的步驟如下:
(1) 要設定Readable的Secondary
(2) AG Listener要設定ILB的IP
(3) Client端要設定Application intent=readonly
PS.
傳統方式是需要在Listener設定多個IP來做故障切換，但在GCP上只是將改成這個故障切換的做法用ILB來處理

Q5. 在有ILB的設定下，對SQL Server AlwaysOn做Read/Write Split是否有影響？
答:
Read/Write Split是要在SQL Server中針對Read/Write封包路由，保證Write封包送到Primary，Read封包送到Secondary，和跟ILB不衝突。
其設定方式如下:

- 若希望可以對 secondary read (read/ write split)，在設置AlwaysOn Group時要開啟 readable secondary



- 設定Read/Write路由
  - 參見DB設定Read/Write Split
- 連接的Connection要補上Connection String - ApplicationIntent=READONLY

參考文件

- Run Highly Available Microsoft SQL Server in Compute Engine (Cloud Next '19)
- DBS204: Creating an AlwaysOn cluster in Google Cloud
- Deploy an Active Directory forest on Compute Engine
- Configuring SQL Server AlwaysOn availability groups with synchronous commit

- Deploying a Fault-Tolerant Microsoft Active Directory Environment
- SQL Server Always On Availability Group 可用性群組安裝筆記