Objectif : Créer des réseaux Vnet, configurer le « peering », créer des VMs dans ces réseaux et créer un compte de stockage avec un partage

PARTIE I : Réseaux virtuels (VNet) et Peering

1. Créer le VNet-BD avec sous-réseau SR-BD

1. Connecte-toi à https://portal.azure.com (ouvrir un abonnement **SANDBOX LEARN**)
2. Va dans **"Réseaux virtuels"** > **Créer**.
3. Onglet **Informations de base** :
    - Groupe de ressource : **learn-...**
    - Nom : **Vnet-BD**
    - Région : West US
4. Onglet **Adresses IP** :
    - Espace d'adressage : **10.10.0.0/16**
    - **Supprimer le sous-réseau par défaut.**
    - Ajouter un sous-réseau :
        - Nom : **SR-BD**
        - Plage : **10.10.1.0/24**
5. Laisser le reste par défaut > **Créer**

2. Créer le VNet-WEB avec sous-réseau SR-WEB

Répète les mêmes étapes mais avec :

- Nom VNet : **Vnet-WEB**
- Espace d'adressage : **10.20.0.0/16**
- Sous-réseau : **SR-WEB** avec **10.20.1.0/24**

3. Configurer le peering entre les deux VNet

Répète deux fois les étapes suivantes :

1. Va dans **"Réseaux virtuels"** > Vnet-BD > **Paramètres** > **Peering** > **Ajouter** :
    - Nom du peering : **Peering-BD-to-WEB**
    - Réseau distant : Vnet-WEB
    - Coche Autoriser le trafic vers le réseau distant : **Autoriser 'Vnet-WEB' à accéder à 'Vnet-BD'**
2. Puis fais la même chose dans l'autre sens :
    - Depuis Vnet-WEB
    - Vers Vnet-BD
    - Nom du peering : **Peering-WEB-to-BD**
    - Coche Autoriser le trafic vers le réseau distant : **Autoriser 'Vnet-BD' à accéder à 'Vnet-WEB'**

PARTIE II : Machines virtuelles (VM-BD et VM-WEB)

1. Créer la VM-BD (serveur MySQL)

2. Va dans **Machines virtuelles** > **Créer une VM** :
    - Groupe de ressource : **learn-...**
    - Nom : **serveur-bd-idosr00**
    - Image : **Ubuntu 22.04 LTS**
    - Type d'authentification : **mot de passe**
        - ✓ Nom d'utilisateur : **hassan**
        - ✓ Mot de passe : **P@ssw0rd123456**

- Dans l'onglet **Ports d'entrée** : **22 (SSH)** **(pas besoin d'ouvrir 3306 de MySQL)**
- Réseau : **Vnet-BD** / Sous-réseau : **SR-BD**
- Adresse IP publique : **oui (Nouveau)**

3. **Créer**

2. Créer la VM-WEB

Même processus avec :

- Nom : **serveur-web-idosr00**
- Login/Password : **hassan** / **P@ssw0rd123456**
- Ouvrir les ports **80** et **22**
- Réseau : **Vnet-WEB** / Sous-réseau : **SR-WEB**

PARTIE III : Compte de stockage + Partage Azure Files

1. Créer un compte de stockage

1. Va dans **Comptes de stockage** > **Créer** :
   - Groupe de ressources : **learn-...**
   - Nom : **storageaccountidosr00 (doit être unique dans tout AZURE)**
   - Service principal : **Azure Files**
   - Performance : **Premium**
   - Redondance : **LRS** (Stockage localement redondant)
   - Région : West US
2. **Créer**

2. Créer un partage de fichiers

1. Dans le compte créé **> Stockage des données > Partages de fichiers** > **+ Partage de fichiers**
   - Nom : **myshare00**
   - Quota : 100 Go
   - Activer la Sauvegarde : **NO** (décocher la case)
   - Niveau d'accès : Premium
   - **Créer**

3. Récupérer la clé de stockage

2. Dans le compte > **Sécurité + réseau** > **Clés d'accès** > **Copier la clé d'accès primaire**

PARTIE IV : Monter le partage dans Ubuntu

➔ Sur tes VMs (via SSH) :

```
sudo mkdir /mnt/azurefiles
sudo mount -t cifs //storageaccountidosr00.file.core.windows.net/myshare00 /mnt/azurefiles -o vers=3.0,username=storageaccountidosr00,password=<clé>,dir_mode=0777,file_mode=0777
```

Remplace <clé> par la vraie clé que tu as copiée.

PARTIE V : Afficher les IP publiques des VMs

1. Va dans **Machines virtuelles** > **serveur-bd-idosr00** > **Vue d'ensemble** > Copier l'**adresse IP publique**
2. Idem pour **serveur-web-idosr00**

PARTIE VI : Configurer les VMs

1. Dans VM-BD : installer et configurer MySQL

1. Accéder par SSH à la VM de base de données (**VM-BD**) :
   ssh hassan@<IP_VM-BD>
2. Créer un script **sql.sh**
   sudo vi sql.sh
3. Coller le script **(voir l'annexe ci-dessous)**
4. Donner le droit d'exécution et exécuter le script :
   sudo chmod +x sql.sh && sudo ./sql.sh

2. Dans VM-WEB : Apache + PHP + page de test

1. Accéder par SSH à la VM de base de données (**VM-WEB**) :
   ssh hassan@<IP_VM-WEB>
2. Créer le script **web.sh** avec le contenu du script PHP de test **(voir l'annexe)**
   sudo vi web.sh
3. Donner le droit d'exécution et exécuter le script :
   sudo chmod +x web.sh && sudo ./web.sh

➔ Accédez à **http://<IP_VM-WEB>/index.php** pour tester la connexion à MySQL (**VM-BD**) depuis le serveur WEB (**VM-WEB**).

3. Dans VM-BD : Se connecter au partage AZURE

1. Accéder par SSH à la VM de base de données (**VM-BD**) :
   ssh hassan@<IP_VM-BD>
2. Créer le script **connect.sh (voir l'annexe ci-dessous)**
3. Donner le droit d'exécution et exécuter le script :
   sudo chmod +x connect.sh && sudo ./connect.sh

4. Dans VM-BD : Sauvegarde automatique MySQL

1. Accéder par SSH à la VM de base de données (**VM-BD**) :
   ssh hassan@<IP_VM-BD>
2. Créer le script **backup.sh (voir l'annexe ci-dessous)**
3. Donner le droit d'exécution et exécuter le script :
   sudo chmod +x backup.sh && sudo ./backup.sh
4. Vérifier la création du premier fichier de sauvegarde sur VM-BD :
   ls /mnt/azurefiles

Supprimer l'IP publique de la VM-BD

1. Accédez à **la VM (VM-BD)** : **Mise en réseau > Paramètres réseau > cliquer votre** « Adresse IP publique » **> Dissocier > Supprimer** « serveur-bd-idosr00PublicIP »

NB : Pour vérifier accédez à **"Adresse IP publique"**

2. Dorénavant, on va accéder à la **VM-BD** depuis la **VM-WEB** en utilisant l'adresse IP privé

ANNEXES : Scripts

##########sudo vi **sql.sh**

```
#!/bin/bash
# Mettre à jour le système
sudo apt update && sudo apt upgrade -y

# Installer MySQL Server
sudo apt install -y mysql-server

# Activer et démarrer MySQL
sudo systemctl enable mysql
sudo systemctl start mysql

# Autoriser les connexions depuis l'extérieur
sudo sed -i "s/^bind-address.*/bind-address = 0.0.0.0/" /etc/mysql/mysql.conf.d/mysqld.cnf
#sudo sed -i "s/^bind-address.*/bind-address = 10.10.1.4/"
/etc/mysql/mysql.conf.d/mysqld.cnf

# Redémarrer MySQL pour appliquer la config
sudo systemctl restart mysql

# Créer un utilisateur admin 'hassan' et la base 'db_web'
sudo mysql <<EOF
CREATE DATABASE db_web;
CREATE USER 'hassan'@'%' IDENTIFIED BY 'P@ssw0rd123456';
GRANT ALL PRIVILEGES ON db_web.* TO 'hassan'@'%';
FLUSH PRIVILEGES;
EOF

# Ouvrir le port 3306 sur le pare-feu (si UFW est actif)
if sudo ufw status | grep -q active; then
  sudo ufw allow 3306/tcp
fi

echo "MySQL installé, configuré, et base de données 'db_web' prête !"
```

################ sudo chmod +x sql.sh
################ sudo ./sql.sh
##

##########sudo vi **web.sh**

```
#!/bin/bash
# Mettre à jour le système
sudo apt update

# Installer Apache2, PHP et les extensions PHP pour MySQL
sudo apt install -y apache2 php libapache2-mod-php php-mysql

# Créer la page PHP de test
sudo bash -c "cat > /var/www/html/index.php" <<EOF
<?php
\$conn = new mysqli('10.10.1.4', 'hassan', 'P@ssw0rd123456', 'db_web');

if (\$conn->connect_error) {
   die('Connexion échouée : ' . \$conn->connect_error);
}
echo 'Connexion réussie à la base de données MySQL distante ! <br>';
echo 'Je suis DRAOUI Hassan !';
\$conn->close();
?>
EOF

# Redémarrer Apache pour prendre en compte les changements
sudo systemctl restart apache2

echo "Installation terminée. Accédez à http://<IP_de_cette_VM-WEB>/index.php pour
tester."
```

################ sudo chmod +x web.sh
################ sudo ./web.sh
##

##########sudo vi **connect.sh**

```
#!/bin/bash
# ------------------------------------------------------------------------
# Script pour monter un partage Azure Files (SMB) sur Ubuntu 22.04
# Usage: ./connect.sh [--fstab] (ajoute le montage automatique au démarrage)
# ------------------------------------------------------------------------

# Variables à modifier par l'utilisateur
# Nom du compte de stockage Azure
STORAGE_ACCOUNT="storageaccountidosm00"
# Nom du partage Azure Files
SHARE_NAME="myshare00"
# Clé d'accès Azure (à modifier)
STORAGE_KEY="O0GewKv8C1AYCCbh+p7KPkySyOnKZJVWLRf+yzsTJOb3gnbrq
fNjLkUtX0+yuUmihn2JJ1Kq8fD3+AStpDD80g=="
# Dossier local de montage
MOUNT_POINT="/mnt/azurefiles"

# Options de montage SMB
MOUNT_OPTIONS="vers=3.0,username=$STORAGE_ACCOUNT,password=$STORAGE
_KEY,dir_mode=0777,file_mode=0777,serverino,nosharesock,actimeo=30"

# ------------------------------------------------------------------------
# Vérification des privilèges sudo
if [ "$(id -u)" -ne 0 ]; then
    echo "Ce script doit être exécuté avec sudo ou en tant que root."
    exit 1
fi

# Installation des dépendances
echo "Installation de cifs-utils..."
apt update -qq && apt install -y cifs-utils
if [ $? -ne 0 ]; then
    echo "Échec de l'installation de cifs-utils."
    exit 1
fi

# Création du dossier de montage
echo "Création du point de montage $MOUNT_POINT..."
mkdir -p "$MOUNT_POINT"
if [ $? -ne 0 ]; then
    echo "Échec de la création du dossier $MOUNT_POINT."
    exit 1
fi

# Montage du partage SMB
```

```
echo "Montage du partage Azure Files..."
mount -t cifs "//$STORAGE_ACCOUNT.file.core.windows.net/$SHARE_NAME"
"$MOUNT_POINT" -o "$MOUNT_OPTIONS"
if [ $? -ne 0 ]; then
   echo "Échec du montage. Vérifiez les paramètres (compte, clé, partage)."
   exit 1
fi

# Option: Ajouter à /etc/fstab pour le montage automatique
if [[ "$1" == "--fstab" ]]; then
   echo "Ajout à /etc/fstab pour le montage au démarrage..."
   FSTAB_ENTRY="//$STORAGE_ACCOUNT.file.core.windows.net/$SHARE_NAME
$MOUNT_POINT cifs $MOUNT_OPTIONS 0 0"
   if grep -q "$MOUNT_POINT" /etc/fstab; then
      echo "Une entrée existe déjà dans /etc/fstab pour $MOUNT_POINT."
   else
      echo "$FSTAB_ENTRY" | tee -a /etc/fstab > /dev/null
      echo "Entrée ajoutée à /etc/fstab. Test avec 'sudo mount -a'..."
      mount -a
   fi
fi

# Résumé
echo "------------------------------------------------------------------"
echo "Partage Azure Files monté avec succès:"
echo -e "Compte de stockage: \t$STORAGE_ACCOUNT"
echo -e "Partage: \t\t$SHARE_NAME"
echo -e "Point de montage: \t$MOUNT_POINT"
echo -e "Options: \t\t$MOUNT_OPTIONS"
if [[ "$1" == "--fstab" ]]; then
   echo "Configuré dans /etc/fstab pour le montage automatique."
fi
echo "------------------------------------------------------------------"

#Rendez-le exécutable :
########## chmod  +x  connect.sh
#Pour un montage manuel (temporaire) :
########## sudo  ./connect.sh
#Pour ajouter le montage automatique au démarrage (via /etc/fstab) :
########## sudo  ./connect  --fstab
#################################################################################
```

########## sudo vi **backup.sh**

```
#!/bin/bash
# --------------------------------------------------------------------------
# Script pour configurer des sauvegardes MySQL vers Azure Files
# - Configure des sauvegardes immédiates
# - Programme une tâche cron quotidienne à minuit
# --------------------------------------------------------------------------

# Variables à configurer
# Utilisateur MySQL (doit avoir les droits de sauvegarde)
MYSQL_USER="hassan"
# Mot de passe MySQL (vide si auth via socket)
MYSQL_PASSWORD="P@ssw0rd123456"
# Point de montage du partage Azure Files
AZURE_MOUNT_POINT="/mnt/azurefiles"
# Dossier de sauvegarde
BACKUP_DIR="$AZURE_MOUNT_POINT/mysql_backups"
# Nombre de jours à conserver les sauvegardes
DAYS_TO_KEEP=7
# --------------------------------------------------------------------------
# Vérification des privilèges
if [ "$(id -u)" -ne 0 ]; then
   echo "Ce script doit être exécuté avec sudo ou en tant que root."
   exit 1
fi

# Vérification que le partage Azure est monté
if ! mountpoint -q "$AZURE_MOUNT_POINT"; then
   echo "Erreur: Le partage Azure Files n'est pas monté sur $AZURE_MOUNT_POINT"
   exit 1
fi

# Création du dossier de sauvegarde
mkdir -p "$BACKUP_DIR"
chmod 777 "$BACKUP_DIR"

# Fonction pour effectuer une sauvegarde
perform_backup() {
   local BACKUP_FILE="$BACKUP_DIR/backup_$(date +%Y%m%d_%H%M%S).sql.gz"

   echo "Début de la sauvegarde MySQL vers $BACKUP_FILE..."

   # Commande de sauvegarde (avec auth par mot de passe ou socket)
   if [ -z "$MYSQL_PASSWORD" ]; then
      mysqldump -u "$MYSQL_USER" --all-databases | gzip > "$BACKUP_FILE"
   else
```

```
    mysqldump -u "$MYSQL_USER" -p"$MYSQL_PASSWORD" --all-databases | gzip >
"$BACKUP_FILE"
  fi

  # Vérification de la sauvegarde
  if [ $? -eq 0 ] && [ -s "$BACKUP_FILE" ]; then
    echo "Sauvegarde réussie. Taille: $(du -h "$BACKUP_FILE" | cut -f1)"
  else
    echo "Échec de la sauvegarde!"
    rm -f "$BACKUP_FILE"
    exit 1
  fi

  # Nettoyage des anciennes sauvegardes
  find "$BACKUP_DIR" -name "backup_*.sql.gz" -type f -mtime +$DAYS_TO_KEEP -
delete
  echo "Nettoyage des sauvegardes de plus de $DAYS_TO_KEEP jours effectué."
}

# ----------------------------------------------------------------------------
# A. Effectuer une sauvegarde immédiate
perform_backup

# ----------------------------------------------------------------------------
# B. Configurer la tâche cron pour minuit chaque jour
CRON_JOB="0 0 * * * $(which bash) -c '$(readlink -f "$0")'"

# Vérifier si la tâche existe déjà
if ! crontab -l | grep -qF "$(readlink -f "$0")"; then
  (crontab -l 2>/dev/null; echo "$CRON_JOB") | crontab -
  echo "Tâche cron configurée pour s'exécuter quotidiennement à minuit."
else
  echo "Une tâche cron existe déjà pour ce script."
fi

# ----------------------------------------------------------------------------
echo "Configuration terminée!"
echo " - Sauvegardes stockées dans: $BACKUP_DIR"
echo " - Tâche cron programmée:"
crontab -l | grep "$(readlink -f "$0")"
########## chmod +x backup.sh
########## sudo ./backup.sh
################################################################################
# 3. Pour vérifier
        ########## ls   /mnt/azurefiles/mysql_backups/
################################################################################
```