Lab 1: Create Linux VMs in a Secure Environment

In this Lab we are going to deploy two virtual machines and configure Azure networking for these VMs. Assume that the VMs are hosting a web application with a database backend, however an application is not deployed in the Lab. There are 6 Tasks in this Lab. You learn how to:

> - Create a virtual network and subnet
> - Create Bastion Host
> - Create a public IP address
> - Create a frontend(webapp) VM
> - Enable AADSSH for VMs.
> - Secure network traffic with NSGs
> - Block default outound connection and use NAT Gateway for explicit outbound connections
> - Create a backend(DB) VM

VM networking overview

In this Lab We'll create 3 subnets for traffic segmentation, one for frontend VM, one for backend-vm and third one is for Azure Bastion host.
At the end of the Lab the following virtual network resources are created:

- *myVNet-${SUFFIX}* - The virtual network that the VMs use to communicate with each other and the internet.
- *myFrontendSubnet* - The subnet in *myVNet* used by the frontend resources.
- *myPublicIPAddress* - The public IP address used to access *myFrontendVM* from the internet.
- *myFrontendNic* - The network interface used by *myFrontendVM* to communicate with *myBackendVM*.
- *myFrontendVM* - The VM used to communicate between the internet and *myBackendVM*.
- *myBackendNSG* - The network security group that controls communication between the *myFrontendVM* and *myBackendVM*.
- *myBackendSubnet* - The subnet associated with *myBackendNSG* and used by the backend resources.
- *myBackendNic* - The network interface used by *myBackendVM* to communicate with *myFrontendVM*.
- *myBackendVM* - The VM that uses 3306 to communicate with *myFrontendVM*.
- *bastion-levelup-${SUFFIX}* - The bastion host that used for connecting to *myBackendVM*
- *nat-level-up-${SUFFIX}* - The nat gateway for outbound traffic communication in *myBackendSubnet*

TASK 1 : Create a virtual network and subnets

For this tutorial, a single virtual network is created with three subnets. A frontend subnet for hosting a web application, and a backend subnet for hosting a database server and a AzureBastionSubnet for hosting bastion host.

Before you can create a virtual network, create a resource group with *az group create*. The following example creates a resource group named *rg-levelup-${SUFFIX}* in the `swedencentral` location.

Step 1: Variables

Set the following variables to create the Azure resources.

```
export SUFFIX=$(cat /dev/urandom | LC_ALL=C tr -dc 'a-z0-9' | fold -w 8 | head -n
1)
export RESOURCE_GROUP_NAME="rg-levelup-${SUFFIX}"
export VNET_NAME="myVNet-${SUFFIX}"
export VM_ADMIN_USER="azureuser"
export VM_IMAGE="Canonical:ubuntu-24_04-lts:server:latest"
export REGION="swedencentral"
```

```
az group create --name $RESOURCE_GROUP_NAME --location $REGION
```

Step 2: Create virtual network and Frontend Subnet

Use the *az network vnet create* command to create a virtual network. In this example the network is named *mvVNet-${SUFFIX}* and is given an address prefix of *10.0.0.0/16*. A subnet is also created with a name of *myFrontendSubnet* and a prefix of *10.0.1.0/24*. Later in this lab a frontend VM is connected to this subnet.

See the below important features that we used while creating the VNET:

- For *myBackendSubnet* the Outbound Internet connection is disabled with *--default-outbound false* flag. By default outbound internet connection is open for the VNETs. A breaking change will be happen to disable default outbound internet connection on Sep 30th 2025. You can find more information here : https://azure.microsoft.com/en-us/updates/default-outbound-access-for-vms-in-azure-will-be-retired-transition-to-a-new-method-of-internet-access/
- VNET Encrytion is set with *--enable-encryption true* . This means the VM traffic within the VNET will be encrypted. Virtual Network encryption is supported on general-purpose and memory optimized virtual machine instance sizes. The *--encryption-enforcement-policy* flag is for controlling if the VM without encryption is allowed in encrypted Virtual Network or not.

```
az network vnet create \
  --resource-group $RESOURCE_GROUP_NAME \
  --name $VNET_NAME \
  --address-prefix 10.0.0.0/16 \
  --enable-encryption true \
  --encryption-enforcement-policy allowUnencrypted \
  --location $REGION
```

```
az network vnet subnet create \
  --resource-group $RESOURCE_GROUP_NAME \
  --vnet-name $VNET_NAME \
  --name myFrontendSubnet \
  --address-prefix 10.0.1.0/24
```

Step 3: Create backend and bastion subnets

A new subnet is added to the virtual network using the *az network vnet subnet create* command. In this example, the subnet is named *myBackendSubnet* and is given an address prefix of *10.0.2.0/24*. This subnet is used with all backend services.

```
az network vnet subnet create \
  --resource-group $RESOURCE_GROUP_NAME \
  --vnet-name $VNET_NAME \
  --name myBackendSubnet \
  --default-outbound false \
  --address-prefix 10.0.2.0/24
```

This will create a subnet for Azure Bastion. The subnet name had to be *AzureBastionSubnet* for bastion host.

```
export BASTION_SUBNET_NAME="AzureBastionSubnet"
export BASTION_SUBNET_CIDR="10.0.3.0/26"

az network vnet subnet create \
    --name $BASTION_SUBNET_NAME \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --vnet-name "$VNET_NAME" \
    --address-prefix $BASTION_SUBNET_CIDR \
    --query 'properties.provisioningState' \
    --output tsv
```

At this point, a network has been created and segmented into three subnets, one for frontend services, one for backend services and the last one is for bastion host. In the next section, virtual machines are created and connected to these subnets. We will not assign an publicIP for *myBackendVM* and we'll use bastion host for connecting *myBackendVM*

TASK 2: Create and Configure Frontend VM

Step 1: Extra settings

We're going to use EncryptionAtHost feature while creating our VMs. This settings is for using the Host Based Encryption to encrypt the disks on your VM.
You must enable the feature for your subscription before you use the EncryptionAtHost property for your VM/VMSS. Use the following steps to enable the feature for your subscription:

Execute the following command to register the feature for your subscription

```
az feature register --namespace Microsoft.Compute --name EncryptionAtHost
```

Once the feature 'EncryptionAtHost' is registered, invoking 'az provider register -n Microsoft.Compute' is required to get the change propagated

Step 2: Create cloud-init file

We're going to use the frontend VM at our next Lab. Installing some prerequisetes like firewall(ufw) , apparmor* etc.

```
cat << EOF > cloud-init.txt
#cloud-config
# Install, update, and upgrade packages
package_upgrade: true
package_update: true
package_reboot_if_require: true
# Install packages
packages:
  - vim
  - ufw
  - curl
  - bash-completion
  - nginx
  - pwgen
  - libpam-pwquality
  - haveged rng-tools5
  - wget
  - apt-transport-https
  - gnupg
  - apparmor
  - apparmor-utils
  - apparmor-profiles
EOF
```

Step 3: Generate SSH key pair using ED25519 encryption

ED25519 ssh keys are in *Public Preview*. You can use ED25519 or RSA keys. ED25519 ssh keys provides better security and performance. The following command creates an SSH key pair using ED25519 encryption with a fixed length of 256 bits:

```
ssh-keygen -m PEM -t ed25519 -f $HOME/id_ed25519_levelup_key.pem -C "LevelUp Linux VM SSH Key"
```

Step 4: Create a public IP address for frontend VM

```
az network public-ip create --resource-group $RESOURCE_GROUP_NAME --name myPublicIPAddress
```

Step 5: Create a frontend VM

Use the *az vm create* command to create the VM named *myFrontendVM* using *myPublicIPAddress*.

```
az vm create \
  --resource-group $RESOURCE_GROUP_NAME \
  --name myFrontendVM \
  --vnet-name $VNET_NAME \
  --subnet myFrontendSubnet \
  --nsg myFrontendNSG \
  --public-ip-address myPublicIPAddress \
  --image $VM_IMAGE \
  --assign-identity \
  --accelerated-networking true \
  --storage-sku os=Premium_LRS \
  --encryption-at-host true \
  --os-disk-caching ReadWrite \
  --os-disk-delete-option Delete \
  --os-disk-size-gb 30 \
  --admin-username $VM_ADMIN_USER \
  --authentication-type ssh \
  --ssh-key-value "$HOME/id_ed25519_levelup_key.pem.pub" \
  --custom-data cloud-init.txt \
  --output tsv
```

TASK 3: Secure network traffic

Step 1: Create network security groups

A network security group can be created at the same time as a VM using the *az vm create* command. When doing so, the NSG is associated with the VMs network interface and an NSG rule is auto created to allow traffic on port *22* from any source. Earlier in this lab the frontend NSG was auto-created with the frontend VM. An NSG rule was also auto created for port 22.

In some cases, it may be helpful to pre-create an NSG, such as when default SSH rules should not be created, or when the NSG should be attached to a subnet. We would like prevent default ssh rule to our backend VM, so we'll create the NSG before hand.

Use the *az network nsg create* command to create a network security group.

```
az network nsg create --resource-group $RESOURCE_GROUP_NAME --name myBackendNSG
```

Instead of associating the NSG to a network interface, it is associated with a subnet. In this configuration, any VM that is attached to the subnet inherits the NSG rules.

Update the existing subnet named *myBackendSubnet* with the new NSG.

```
az network vnet subnet update \
  --resource-group $RESOURCE_GROUP_NAME \
  --vnet-name $VNET_NAME \
  --name myBackendSubnet \
  --network-security-group myBackendNSG
```

Step 2: Secure incoming traffic

When the frontend VM was created, an NSG rule *default-allow-ssh* named was created to allow incoming traffic on port 22. This default rule allows SSH connections to the VM. We'll delete the *default-allow-ssh* rule since we do not want to keep port *22* open publicly. We'll configure Just-in Time Access (JIT) for frontend VM ssh connection on upcoming steps(Task 6).
For this example, traffic should also be allowed on ports *80* and *443*. This configuration allows a web application to be accessed on the VM.

Use the *az network nsg rule create* command to create a rule for port *80* and *443*.

Delete default port 22 permission with *az network nsg rule delete* command

```
az network nsg rule delete  --resource-group $RESOURCE_GROUP_NAME   --nsg-name
myFrontendNSG   --name default-allow-ssh
```

Create NSG rule for incoming web traffic on port 80 and 443

```
az network nsg rule create \
  --resource-group $RESOURCE_GROUP_NAME \
  --nsg-name myFrontendNSG \
  --name httpandhttps \
  --access allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 200 \
  --source-address-prefix "*" \
  --source-port-range "*" \
  --destination-address-prefix "*" \
  --destination-port-ranges 80 443
```

The frontend VM is only accessible on port port *80* and port *443*. All other incoming traffic is blocked at the network security group. It may be helpful to visualize the NSG rule configurations. Return the NSG rule configuration with the *az network rule list* command.

```
az network nsg rule list --resource-group $RESOURCE_GROUP_NAME --nsg-name
myFrontendNSG --output table
```

Step 3: Configure role assignments for the VM

The following example uses az role assignment create to assign the Virtual Machine Administrator Login role to the VM for your current Azure user. You obtain the username of your current Azure account by using az account show, and you set the scope to the VM created in a previous step by using az vm show. This role is needed for using AADSSH with VM.

You can also assign the scope at a resource group or subscription level.We'll use resource group for scope.

```
USERNAME=$(az account show --query user.name --output tsv)
SCOPE=$(az group show -n  $RESOURCE_GROUP_NAME  -o tsv --query id)

az role assignment create --role "Virtual Machine Administrator Login" --assignee
$USERNAME --scope $SCOPE
```

Step 4: Enable Azure AD Login for a Linux virtual machine in Azure

The following code example installs the extension to enable an Azure AD Login for a Linux VM. VM extensions are small applications that provide post-deployment configuration and automation tasks on Azure virtual machines.

```
az vm extension set \
    --publisher Microsoft.Azure.ActiveDirectory \
    --name AADSSHLoginForLinux \
    --resource-group $RESOURCE_GROUP_NAME \
    --vm-name myFrontendVM \
    --output tsv
```

Step 5: Secure VM to VM traffic

Network security group rules can also apply between VMs. For this example, the frontend VM needs to communicate with the backend VM on port *3306* and Bastion host needs to reach out to backend VM on port *22*. This configuration allows SSH connections from the Bastion Host, and also allow an application on the frontend VM to communicate with a backend MySQL database. All other traffic should be blocked between the frontend and backend virtual machines.

Use the *az network nsg rule create* command to create a rule for port 22. Notice that the `--source-address-prefix` argument specifies a value of *10.0.1.0/24* for forntend VM subnet and *10.0.3.0/26 * for Bastion subnet. This configuration ensures that only traffic from the frontend subnet is allowed for port *3306* and bastion subnet for port *22* through the NSG.

Add a NSG rule to allow SSH(22) access from AzureBastionSubnet(10.0.3.0/26) to backend subnet

```
az network nsg rule create \
  --resource-group $RESOURCE_GROUP_NAME \
  --nsg-name myBackendNSG \
  --name SSH \
  --access Allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 100 \
  --source-address-prefix 10.0.3.0/26 \
  --source-port-range "*" \
  --destination-address-prefix "*" \
  --destination-port-range "22"
```

Now add a rule for MySQL traffic on port 3306.

```
az network nsg rule create \
  --resource-group $RESOURCE_GROUP_NAME \
  --nsg-name myBackendNSG \
  --name MySQL \
  --access Allow \
  --protocol Tcp \
  --direction Inbound \
  --priority 200 \
  --source-address-prefix 10.0.1.0/24 \
  --source-port-range "*" \
  --destination-address-prefix "*" \
  --destination-port-range "3306"
```

Finally, because NSGs have a default rule allowing all traffic between VMs in the same VNet, a rule can be created for the backend NSGs to block all traffic. Notice here that the `--priority` is given a value of *300*, which is lower that both the NSG and MySQL rules. This configuration ensures that SSH and MySQL traffic is still allowed through the NSG.

```
az network nsg rule create \
  --resource-group $RESOURCE_GROUP_NAME \
  --nsg-name myBackendNSG \
  --name denyAll \
  --access Deny \
  --protocol Tcp \
  --direction Inbound \
  --priority 300 \
  --source-address-prefix "*" \
  --source-port-range "*" \
  --destination-address-prefix "*" \
  --destination-port-range "*"
```

Step 6: Create public IP address for Azure NAT Gateway

The backend VM does not need a public IP, we are going to add a NAT Gateway for Outbound Internet connection. We're adding this just for LAB purposes to show how you can use NAT GW to allow outbound internet connection explicitly.

```
export NAT_PUBLIC_IP_NAME="nat-ip-levelup-${SUFFIX}"
```

```
az network public-ip create \
    --name "$NAT_PUBLIC_IP_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --location $REGION \
    --allocation-method Static \
    --sku Standard \
    --version IPv4 \
    --zone 1 2 3 \
    --output tsv
```

Step 7: Create NAT gateway resource

```
export NAT_GW_NAME="nat-levelup-${SUFFIX}"

az network nat gateway create \
    --resource-group $RESOURCE_GROUP_NAME \
    --name $NAT_GW_NAME \
    --public-ip-addresses $NAT_PUBLIC_IP_NAME \
    --idle-timeout 10 \
    --location $REGION \
    --output tsv
```

Step 8 : Configure NAT gateway for the backend subnet

```
az network vnet subnet update \
    --name myBackendSubnet \
    --resource-group $RESOURCE_GROUP_NAME \
    --vnet-name $VNET_NAME \
    --nat-gateway $NAT_GW_NAME
```

TASK 4 : Create Azure Bastion Host

Step 1: Create Azure Public IP for Bastion Host

```
export BASTION_NAME="bastion-levelup-${SUFFIX}"
export BASTION_PUBLIC_IP_NAME="bastion-ip-levelup-${SUFFIX}"

az network public-ip create \
    --name "$BASTION_PUBLIC_IP_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --location $REGION \
    --allocation-method Static \
    --sku Standard \
    --dns-name "$BASTION_NAME" \
    --sku Standard \
    --version IPv4 \
    --zone 1 2 3 \
    --allocation-method Static \
    --output tsv
```

Step 2: Create Azure Bastion Host with native client support and IP-based connections

```
az network bastion create \
    --name "$BASTION_NAME" \
    --resource-group "$RESOURCE_GROUP_NAME" \
    --vnet-name "$VNET_NAME" \
    --location "$REGION" \
    --public-ip-address "$BASTION_PUBLIC_IP_NAME" \
    --enable-ip-connect true \
    --enable-tunneling true \
    --sku Standard \
    --query 'properties.provisioningState' \
    --output tsv
```

TASK 5: Create backend VM and Connect via SSH

Step 1: Create backend VM

Now create a virtual machine, which is attached to the *myBackendSubnet*. Notice that the `--nsg` argument has a value of empty double quotes. An NSG does not need to be created with the VM. The VM is attached to the backend subnet, which is protected with the pre-created backend NSG. This NSG applies to the VM. Also, notice here that the `--public-ip-address` argument has a value of empty double quotes. This configuration creates a VM without a public IP address.

```
az vm create \
  --resource-group $RESOURCE_GROUP_NAME \
  --name myBackendVM \
  --vnet-name $VNET_NAME \
  --subnet myBackendSubnet \
  --public-ip-address "" \
  --assign-identity \
  --nsg "" \
  --image Ubuntu2204 \
  --assign-identity \
  --accelerated-networking true \
  --storage-sku os=Premium_LRS \
  --encryption-at-host true \
  --os-disk-caching ReadWrite \
  --os-disk-delete-option Delete \
  --os-disk-size-gb 30 \
  --admin-username $VM_ADMIN_USER \
  --authentication-type ssh \
  --ssh-key-value "$HOME/id_ed25519_levelup_key.pem.pub"
```

The backend VM is only accessible on port *22* from bastion subnet and on port *3306* from the frontend subnet. All other incoming traffic is blocked at the network security group. It may be helpful to visualize the NSG rule configurations. Return the NSG rule configuration with the *az network rule list* command.

```
az network nsg rule list --resource-group $RESOURCE_GROUP_NAME --nsg-name
myBackendNSG --output table
```

Step 2: Enable Azure AD Login for Backend and Frontend VMs

```
az vm extension set \
    --publisher Microsoft.Azure.ActiveDirectory \
    --name AADSSHLoginForLinux \
    --resource-group $RESOURCE_GROUP_NAME \
    --vm-name myBackendVM \
    --output tsv
```

Step 3: Connect to backend VM through bastion

Connect to myBackendVM with your AD account through the Bastion Host we created above .

You need to get the Resource ID for the VM to which you want to connect. The Resource ID can be easily located in the Azure portal or with the *az vm show* command as we use below.

```
export VM_ID=$(az vm show -n myBackendVM --resource-group $RESOURCE_GROUP_NAME -o
tsv --query id)

az network bastion ssh --name $BASTION_NAME --resource-group $RESOURCE_GROUP_NAME
--target-resource-id $VM_ID --auth-type "AAD"
```

TASK 6: Enable and configure Just-in-Time (JIT) access for frontend VM SSH access

Step 1: Enable *Defender for Servers Plan 2* on Azure subscription

```
az security pricing create --name VirtualMachines --tier Standard --subplan P2
```

Step 2: Enable JIT on your frontend Virtual Machine

In this lab we will use JIT access to secure public Internet access to port 22, by locking-down the NSG automatically, and then creating automatic temporary NSG rules to allow specific access for a chosen duration.

Note: it is currently only possible to enable JIT access and create a JIT request by using the Azure Portal

1. Access your VM's **Configuration** blade from within **Settings** on the Azure Portal.
2. Click **Enable just-in-time**.

Step 3: Create a JIT access request for your public IP

Note: it is currently only possible to enable JIT access and create a JIT request by using the Azure Portal

1. Find your VM within the Azure portal and access the **Connect** blade within section **Connect**.
2. Click the link **Request access** and then the button **Request access** and select **Local machine IP** to grant your public IP address temporary JIT access to the VM's SSH port TCP/22.

You should now be able to connect to the VM using the SSH client on your local machine.

Step 4: Connect to your frontend VM via JIT using Entra ID

Run the following command to add the SSH extension for the Azure CLI:

```
az extension add --name ssh
```

ssh to frontend VM through *az ssh*

```
az ssh vm --name myFrontendVM --resource-group $RESOURCE_GROUP_NAME
```

Step 5: Disable *Defender for Servers Plan 2* on Azure subscription (after completion of this lab)

Defender for Servers Plan 2 introduces an additional cost per server, therefore it is advisable to disable the feature once you have completed this lab.

```
az security pricing create --name VirtualMachines --tier Free
```