This module will describe how obtaining performance metrics from a Linux system and test network performance using iperf3

It will cover the following LAB topics:

1. Obtaining performance metrics from a Linux system
2. Test network performance using iperf3

> [!IMPORTANT] All of the commands below are meant to be run on a Ubuntu 24.04 LTS machine previously installed in the Azure-Infra Lab, using sudo or root privileges. The below commands have been tested on WSL2 installation and the latest azure-cli version.

LAB 1: Obtaining performance metrics from a Linux system

This lab was created and inspired by the content of our amazing support colleagues Diego Vargas - @divargas, Esteban Flores - @esflores and Marco Bicca - @mabicca.

Intro: There are several commands that can be used to obtain performance counters on Linux. Commands such as vmstat and uptime, provide general system metrics such as CPU usage, System Memory, and System load. Most of the commands are already installed by default with others being readily available in default repositories. The commands can be separated into:

- CPU
- Memory
- Disk I/O
- Networking
- Processes

TASK:

1. Run the following commands to obtain performance metrics from a Linux system
    - CPU
        - mpstat
        - vmstat
        - uptime
    - Memory
        - free
    - I/O
        - iostat
    - Networking
        - ping
        - qperf
    - Processes
        - pidstat
        - ps

- top

Step 1: CPU

sysstat

Some commands are part of the sysstat package which might not be installed by default. The package can be easily installed with:

```
sudo apt install -y sysstat
```

The mpstat utility is part of the sysstat package. It displays per CPU utilization and averages, which is helpful to quickly identify CPU usage. mpstat provides an overview of CPU utilization across the available CPUs, helping identify usage balance and if a single CPU is heavily loaded.

```
mpstat -P ALL 1
```

The options and arguments are:

- -P: Indicates the processor to display statistics, the ALL argument indicates to display statistics for all the online CPUs in the system.
- 1: The first numeric argument indicates how often to refresh the display in seconds.
- 2: The second numeric argument indicates how many times the data refreshes.

The number of times the mpstat command displays data can be changed by increasing the second numeric argument to accommodate for longer data collection times. Ideally 3 or 5 seconds should suffice, for systems with increased core counts 2 seconds can be used to reduce the amount of data displayed. From the output:

```
Linux 6.8.0-1013-azure (linux001)       08/29/24        _x86_64_        (1 CPU)

17:00:08     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest
%gnice   %idle
17:00:09     all    1.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00   99.00
17:00:09       0    1.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00   99.00

17:00:09     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest
%gnice   %idle
17:00:10     all    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00  100.00
17:00:10       0    0.00    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00  100.00

Average:     CPU    %usr   %nice    %sys %iowait    %irq   %soft  %steal  %guest
%gnice   %idle
Average:     all    0.50    0.00    0.00    0.00    0.00    0.00    0.00    0.00
```

```
0.00   99.50
Average:       0    0.50    0.00    0.00    0.00    0.00    0.00    0.00    0.00
0.00   99.50
```

Things to look out for Some details to keep in mind when reviewing the output for mpstat:

Verify that all CPUs are properly loaded and not a single CPU is serving all the load. This information could indicate a single threaded application. Look for a healthy balance between %usr and %sys as the opposite would indicate more time spent on the actual workload than serving kernel processes. Look for %iowait percentages as high values could indicate a system that is constantly waiting for I/O requests. High %soft usage could indicate high network traffic.

vmstat

The vmstat utility is widely available in most Linux distributions, it provides high level overview for CPU, Memory, and Disk I/O utilization in a single pane. The command for vmstat is:

```
vmstat -w 1 5
```

The options and arguments are:

- -w: Use wide printing to keep consistent columns.
- 1: The first numeric argument indicates how often to refresh the display in seconds.
- 5: The second numeric argument indicates how many times the data refreshes.

The output:

```
--procs-- -----------------------memory---------------------- ---swap-- -----io---
- -system-- ----------cpu----------
   r    b         swpd         free         buff        cache   si   so    bi
bo   in   cs  us  sy  id  wa  st  gu
   2    0            0       642036        85644      2287696    0    0    69
584  101    1   2   0  97   1   0   0
   0    0            0       642036        85644      2287712    0    0     0
0  137  175   0   0 100   0   0   0
   0    0            0       642036        85644      2287712    0    0     0
0   53   79   0   0 100   0   0   0
   0    0            0       642036        85644      2287712    0    0     0
0   29   57   0   0 100   0   0   0
   0    0            0       642036        85644      2287712    0    0     0
0   23   46   0   0 100   0   0   0
```

Things to look out for Some details to keep in mind when reviewing the output for vmstat:

- The r column indicates the number of processes waiting for CPU time, a high value could indicate a CPU bottleneck.

- The b column indicates the number of processes in uninterruptible sleep, a high value could indicate a disk I/O bottleneck.
- The wa column indicates the percentage of time the CPU is waiting for I/O operations to complete, a high value could indicate a disk I/O bottleneck.
- The si and so columns indicate the amount of data swapped in and out of memory, a high value could indicate a memory bottleneck.
- The us and sy columns indicate the percentage of time the CPU is spending on user and system processes, respectively.
- The id column indicates the percentage of time the CPU is idle.
- The gu column indicates the percentage of time the CPU is guest time.
- The st column indicates the percentage of time the CPU is stolen time.
- The in and cs columns indicate the number of interrupts and context switches per second, respectively.
- The bi and bo columns indicate the number of blocks received and sent to block devices per second, respectively.
- The free column indicates the amount of free memory available.
- The buff column indicates the amount of memory used as buffers.
- The cache column indicates the amount of memory used as cache.
- The swpd column indicates the amount of memory swapped to disk.

uptime

For CPU related metrics, the uptime utility provides a broad overview of the system load with the load average values.

```
uptime
```

The output:

```
 17:00:08 up  1:00,  1 user,  load average: 0.00, 0.00, 0.00
```

The load average displays three numbers. These numbers are for 1, 5 and 15 minute intervals of system load.

To interpret these values, it's important to know the number of available CPUs in the system, obtained from the mpstat output before. The value depends on the total CPUs, so as an example of the mpstat output the system has 8 CPUs, a load average of 8 would mean that ALL cores are loaded to a 100%.

A value of 4 would mean that half of the CPUs were loaded at 100% (or a total of 50% load on ALL CPUs). In the previous output, the load average is 9.26, which means the CPU is loaded at about 115%. The 1m, 5m, 15m intervals help identify if load is increasing or decreasing over time.

Step 2: Memory

free

The free utility provides a high level overview of the system memory usage.

```
free -m
```

The -m option displays the output in megabytes.

The output:

```
              total        used        free      shared  buff/cache   available
Mem:           7826        1044        5864         104         917        6544
Swap:          2047           0        2047
```

Things to look out for Some details to keep in mind when reviewing the output for free:

- The total column indicates the total amount of memory available.
- The used column indicates the amount of memory used.
- The free4 column indicates the amount of memory available for use.
- The shared column indicates the amount of memory shared between processes.
- The buff/cache column indicates the amount of memory used as buffers and cache.
- The available column indicates the amount of memory available for use by applications.
- The swap column indicates the amount of swap space available.

Step 3: I/O

iostat

The `iostat` utility provides an overview of disk I/O utilization and can help identify bottlenecks that are related to the disk subsystem.

```
iostat -dxtm 1 5
```

The options and arguments are:

- -d: Per device usage report.
- -x: Extended statistics.
- -t: Display the timestamp for each report.
- -m: Display in MB/s.
- 1: The first numeric argument indicates how often to refresh the display in seconds.
- 2: The second numeric argument indicates how many times the data refreshes.

If you include the extra parameters, the output resembles the following text:

```
    [host@rhel76 ~]$ iostat -dxctm 1
    Linux 3.10.0-957.21.3.el7.x86_64 (rhel76)       08/05/2019      _x86_64_
(1 CPU)
        08/05/2019 07:03:36 PM
```

```
    avg-cpu:  %user   %nice %system %iowait  %steal   %idle
               3.09    0.00    2.28    1.50    0.00   93.14

    Device:         rrqm/s   wrqm/s     r/s     w/s    rMB/s    wMB/s avgrq-sz
avgqu-sz   await r_await w_await  svctm  %util
    sda               0.02     0.52    9.66    2.46     0.31     0.10    70.79
0.27   23.97    6.68   91.77   2.94   3.56
    sdd               0.00     0.00    0.12    0.00     0.00     0.00    64.20
0.00    6.15    6.01   12.50   4.44   0.05
    sdb               0.00    22.90    0.47    0.45     0.01     5.74 12775.08
0.17  183.10    8.57  367.68   8.01   0.74
    sdc               0.00     0.00    0.15    0.00     0.00     0.00    54.06
0.00    6.46    6.24   14.67   5.64   0.09
    md0               0.00     0.00    0.15    0.01     0.00     0.00    89.55
0.00    0.00    0.00    0.00   0.00   0.00
```

Things to look out for:

- Look for `r/s` and `w/s` (IOPS) and `rMB/s` and `wMB/s` and verify that these values are within the limits of the given disk. If the values are close or higher the limits, the disk are going to be throttled, leading to high latency. This information can also be corroborated with the `%iowait` metric from `mpstat`.
- The latency is an excellent metric to verify if the disk is performing as expected. Normally, less than `9ms` is the expected latency for PremiumSSD, other offerings have different latency targets.
- The queue size is a great indicator of saturation. Normally, requests would be served near real time and the number remains close to one (as the queue never grows). A higher number could indicate disk saturation (that is, requests queuing up). There's no good or bad number for this metric. Understanding that anything higher than one means that requests are queuing up helps determine if there's disk saturation.

Step 4: Networking

Fore latency and throughput metrics, the `ping` and `qperf` utilities can be used to measure network performance.

ping

```
ping -c 5 xbox.com

PING xbox.com (20.76.201.171) 56(84) bytes of data.
64 bytes from 20.76.201.171: icmp_seq=1 ttl=105 time=38.8 ms
64 bytes from 20.76.201.171: icmp_seq=2 ttl=105 time=40.0 ms
64 bytes from 20.76.201.171: icmp_seq=3 ttl=105 time=39.0 ms
64 bytes from 20.76.201.171: icmp_seq=4 ttl=105 time=39.9 ms
64 bytes from 20.76.201.171: icmp_seq=5 ttl=105 time=39.2 ms

--- xbox.com ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4005ms
rtt min/avg/max/mdev = 38.833/39.392/40.048/0.482 ms
```

qperf

Install qperf with the following command:

```
sudo apt install -y qperf
```

To test network performance between two systems, the `qperf` utility can be used. The `qperf` utility is a network bandwidth and latency measurement tool that can be used to measure network performance between two systems.

```
qperf -vvs <server_hostname_or_ip_address> tcp_lat
```

For network bandwidth measurements, the `tcp_bw` test can be used.

```
qperf -vvs <server_hostname_or_ip_address> tcp_bw
```

Step 5: Processes

Gathering details on a per process basis helps understand where the load of the system is coming from. The main utility to gather process statics is `pidstat` as it provides details per process for CPU, Memory, and I/O statistics. Lastly, a simple `ps` to sort process by top CPU, and memory usage complete the metrics.

pidstat

To gather process CPU statistics, run pidstat without any options:

The following commands can be used if you want to execute it from Azure CLI:

```
 pidstat 1 2
Linux 5.15.153.1-microsoft-standard-WSL2 (DESKTOP-52L2ATU)      09/09/24
_x86_64_        (4 CPU)

18:25:31      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
18:25:32     1000       381    1.00    2.00    0.00    0.00    3.00     0  node

18:25:32      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
18:25:33     1000       381    1.00    0.00    0.00    0.00    1.00     0  node
18:25:33     1000      2722    0.00    1.00    0.00    0.00    1.00     3  pidstat
18:25:33     1000      9826    1.00    4.00    0.00    0.00    5.00     0  node

Average:      UID       PID    %usr %system  %guest   %wait    %CPU   CPU  Command
Average:     1000       381    1.00    1.00    0.00    0.00    2.00     -  node
Average:     1000      2722    0.00    0.50    0.00    0.00    0.50     -  pidstat
Average:     1000      9826    0.50    2.00    0.00    0.00    2.50     -  node
```

Things to look out for:

- Look for processes with high %wait (iowait) percentage as it might indicate processes that are blocked waiting for I/O, which might also indicate disk saturation.
- Verify that no single process consumes 100% of the CPU as it might indicate a single threaded application.

ps

Lastly ps command displays system processes, and can be either sorted by CPU or Memory.

To sort by CPU and obtain the top 10 processes:

```
ps aux --sort=-%cpu | head -10

      PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root        2200 57.0 43.1 14248092 14175632 pts/1 R+ 16:55   0:08 stress-ng --cpu
12 --vm 2 --vm-bytes 120% --iomix 4 --timeout 240
root        2199 43.0 33.0 14248092 10871144 pts/1 R+ 16:55   0:06 stress-ng --cpu
12 --vm 2 --vm-bytes 120% --iomix 4 --timeout 240
root        1231  0.2  0.1 336308 33764 ?        Sl   16:46   0:01
/usr/bin/python3 -u bin/WALinuxAgent-2.9.1.1-py3.8.egg -run-exthandlers
root         835  0.0  0.0 127076 24860 ?        Ssl  16:46   0:00
/usr/bin/python3 -s /usr/sbin/firewalld --nofork --nopid
root        1199  0.0  0.0  30164 15600 ?        Ss   16:46   0:00
/usr/bin/python3 -u /usr/sbin/waagent -daemon
root           1  0.2  0.0 173208 12356 ?        Ss   16:46   0:01
/usr/lib/systemd/systemd --switched-root --system --deserialize 31
root         966  0.0  0.0 3102460 10936 ?       Sl   16:46   0:00
/var/lib/waagent/Microsoft.GuestConfiguration.ConfigurationforLinux-
1.26.60/GCAgent/GC/gc_linux_service
panzer      1803  0.0  0.0  22360  8220 ?        Ss   16:49   0:00
/usr/lib/systemd/systemd --user
root        2180  0.0  0.0  73524  6968 pts/1    SL+  16:55   0:00 stress-ng --cpu
12 --vm 2 --vm-bytes 120% --iomix 4 --timeout 240
```

top

The `top` command displays real-time system metrics, including CPU, memory, and I/O usage. It can be used to identify processes that are consuming the most resources.

```
top
```

The output:

```
top - 13:51:19 up 2 days,  6:09,  0 user,  load average: 0.53, 0.38, 0.29
Tasks:  56 total,   1 running,  55 sleeping,   0 stopped,   0 zombie
```

```
%Cpu(s):  1.0 us,  0.6 sy,  0.0 ni, 98.3 id,  0.0 wa,  0.0 hi,  0.2 si,  0.0 st
MiB Mem :   7946.9 total,   3948.1 free,   2966.7 used,   1338.8 buff/cache
MiB Swap:   2048.0 total,   2048.0 free,      0.0 used.   4980.1 avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 9119 admn      20   0   23.5g   1.2g  55416 S   3.0  15.5   1:17.01
/home/admn/.vscode-server/bin/4849ca9bdf9666755eb463db297b69e5385090e3/node --dns-
result-order=ipv4first /home/admn/.vscode-server/bin/4849+
    1 root      20   0    2476   1712   1600 S   0.0   0.0   0:01.72 /init
    5 root      20   0    2540   1132   1132 S   0.0   0.0   0:00.58 plan9 --
control-socket 5 --log-level 4 --server-fd 6 --pipe-fd 8 --log-truncate
   22 root      20   0    2496    116      0 S   0.0   0.0   0:00.56 /init
   55 admn      20   0    8544   3916   3000 S   0.0   0.0   0:03.21 ssh-agent
  104 root      20   0   35868  23172   5180 S   0.0   0.3   0:43.33
/usr/bin/python3 /usr/bin/supervisord -c /etc/supervisor/supervisord.conf
  107 nobody    20   0   12964   5552   3948 S   0.0   0.1   0:03.74
/usr/sbin/dnsmasq -k
  108 root      20   0 2327412  82836  53284 S   0.0   1.0   0:21.23
/usr/bin/dockerd -p /var/run/docker.pid
  113 root      20   0   29328   5344   4652 S   0.0   0.1   0:00.00
/usr/sbin/pcscd --foreground
  114 root      20   0   24320   6468   4920 S   0.0   0.1   0:09.72
/lib/systemd/systemd-udevd
```

Things to look out for:

- The `PID` column indicates the process ID.
- The `USER` column indicates the user that owns the process.
- The `%CPU` column indicates the percentage of CPU usage.
- The `%MEM` column indicates the percentage of memory usage.
- The `TIME+` column indicates the total CPU time the process has used.
- The `COMMAND` column indicates the command that is being executed.

The load average values are displayed at the top of the output. The load average values represent the average number of processes that are waiting to run on the CPU over the last 1, 5, and 15 minutes.

For the Cpu(s) section, the `%us` column indicates the percentage of CPU time spent running user processes, the `%sy` column indicates the percentage of CPU time spent running system processes, and the `%id` column indicates the percentage of CPU time spent idle. If the system is presenting a high value for wa, it might indicate that the system is waiting for I/O operations to complete. This could be due to a disk bottleneck. If you observe a high value for si and hi it indicates that software interrupts and hardware interrupts are consuming a lot of CPU time. This could be due to a network or disk bottleneck. Finally if you observe a high value for st it indicates that the system is spending a lot of time in a hypervisor. This could be due to a CPU bottleneck.

LAB 2: Using two Azure VMs to test network performance using iperf3

Intro: iperf3 is a tool used to measure network performance. It can be used to test the maximum achievable bandwidth on IP networks. It is particularly useful when troubleshooting network issues or when comparing network performance between different VMs cross Azure Regions.

TASK: Create two Azure VMs in different regions, peer the VNets, and test network performance using iperf3.

Step 1: Create iperf3 server Azure VM

To set up an Azure VM for an iperf3 server in the East US region, create a resource group named "$RESOURCE_GROUP_NAME" and a VNet named "iperf3-server-vnet" with a subnet "iperf3-server-subnet." Configure a network security group "iperf3-server-nsg" with a rule to allow SSH traffic. Create a public IP "iperf3-server-pip" and a NIC "iperf3-server-nic." Deploy a VM named "iperf3-server-vm" using the Ubuntu 24.04 LTS image and Standard D2s v5 size. Bastion Host is also created for secure access to the VM.

1. Set the following variables to create the Azure resources.

```
export SUFFIX=$(cat /dev/urandom | LC_ALL=C tr -dc 'a-z0-9' | fold -w 8 | head -n
1)
export LOCAL_NAME="iperf3-server"
export TAGS="Environment=iperf3"
export IPERF_SERVER_RESOURCE_GROUP_NAME="rg-${LOCAL_NAME}-${SUFFIX}"
export REGION="eastus2"
```

2. Create a resource group in the primary region.

```
az group create --name $IPERF_SERVER_RESOURCE_GROUP_NAME --location $REGION --tags
$TAGS --output tsv
```

3. Create a VNET and subnet in the primary region.

```
export IPERF_SERVER_VNET_NAME="iperf3-server-vnet"
export VNET_CIDR="10.230.0.0/23"
export IPERF_SERVER_SUBNET_NAME="iperf3-server-subnet"
export IPERF_SERVER_SUBNET_CIDR="10.230.0.0/24"

az network vnet create \
   --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
   --name $IPERF_SERVER_VNET_NAME \
   --location $REGION \
   --tags $TAGS \
   --address-prefix $VNET_CIDR \
   --output tsv

az network vnet subnet create \
   --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
   --vnet-name $IPERF_SERVER_VNET_NAME \
   --name $IPERF_SERVER_SUBNET_NAME \
   --address-prefix $IPERF_SERVER_SUBNET_CIDR \
   --output tsv
```

4. Create a subnet for Azure Bastion.

```
export BASTION_SUBNET_NAME="AzureBastionSubnet"
export BASTION_SUBNET_CIDR="10.230.1.0/26"

az network vnet subnet create \
   --name $BASTION_SUBNET_NAME \
   --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
   --vnet-name $IPERF_SERVER_VNET_NAME \
   --address-prefix $BASTION_SUBNET_CIDR \
   --output tsv
```

5. Create Azure Public IP for Bastion Host

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

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

6. Create Azure Bastion Host with native client support and IP-based connections

```
az network bastion create \
   --name "$BASTION_NAME" \
   --resource-group "$IPERF_SERVER_RESOURCE_GROUP_NAME" \
   --vnet-name "$IPERF_SERVER_VNET_NAME" \
   --location "$REGION" \
   --public-ip-address "$BASTION_PUBLIC_IP_NAME" \
   --enable-ip-connect true \
   --enable-tunneling true \
   --sku Standard \
   --tags $TAGS \
   --output tsv \
   --no-wait
```

7. Create Azure VM in the VM Subnet using Ubuntu 24.04 and without a Public IP

```
cat << EOF > cloud-init-os-perf.txt
#cloud-config
# Install, update, and upgrade packages
package_upgrade: true
package_update: true
package_reboot_if_require: false
# Install packages
packages:
  - iperf3
  - sysstat
  - qperf
EOF
```

8. 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"
```

9. Create a network security group (NSG) to allow SSH and iperf3 traffic.

```
export NSG_NAME="NSG-${SUFFIX}"
export NSG_RULE_NAME="Allow-Access-${SUFFIX}"
export VM_NIC_SERVER_NAME="VMNic-${SUFFIX}"

az network nsg create \
   --name $NSG_NAME \
   --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
   --location $REGION \
   --tags $TAGS \
   --output tsv
```

10. Create a rule to allow connections to the virtual machine on port 22 for SSH and ports 9000 for iperf3.

```
az network nsg rule create \
   --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
   --nsg-name $NSG_NAME \
   --name $NSG_RULE_NAME \
   --access Allow \
   --protocol '*' \
   --direction Inbound \
   --priority 100 \
   --source-address-prefix '*' \
   --source-port-range '*' \
   --destination-address-prefix '*' \
```

```
    --destination-port-range 22 9000 \
    --output tsv
```

11. Use az network nic create to create the network interface for the virtual machine.

```
az network nic create \
    --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
    --name $VM_NIC_SERVER_NAME \
    --location $REGION \
    --accelerated-networking true \
    --ip-forwarding false \
    --subnet $IPERF_SERVER_SUBNET_NAME \
    --vnet-name $IPERF_SERVER_VNET_NAME \
    --network-security-group $NSG_NAME \
    --tags $TAGS \
    --output tsv
```

12. Create the Azure VM using the Ubuntu 24.04 LTS image and the Standard_D2s_v5 size.

```
export IPERF_SERVER_VM_NAME="vm-${LOCAL_NAME}-${SUFFIX}"
export VM_SIZE="Standard_D2s_v5"
export VM_IMAGE="Canonical:ubuntu-24_04-lts:server:latest"
export VM_ADMIN_USER="azureuser"

az vm create \
    --name "$IPERF_SERVER_VM_NAME" \
    --resource-group "$IPERF_SERVER_RESOURCE_GROUP_NAME" \
    --location $REGION \
    --size $VM_SIZE \
    --image $VM_IMAGE \
    --nics $VM_NIC_SERVER_NAME \
    --nic-delete-option Delete \
    --storage-sku os=Premium_LRS \
    --os-disk-caching ReadWrite \
    --os-disk-delete-option Delete \
    --os-disk-size-gb 30 \
    --admin-username $VM_ADMIN_USER \
    --authentication-type ssh \
    --ssh-key-values "$HOME/id_ed25519_levelup_key.pem.pub" \
    --custom-data cloud-init-os-perf.txt \
    --tags $TAGS \
    --output tsv
```

Step 2: Create iperf3 client Azure VM

To set up an Azure VM for an iperf3 client in the West US region, create a resource group named "$RESOURCE_GROUP_NAME" and a VNet named "iperf3-client-vnet" with a subnet "iperf3-client-subnet." Configure a network security group "iperf3-client-nsg" with a rule to allow SSH traffic. Create a public IP

"iperf3-client-pip" and a NIC "iperf3-client-nic." Deploy a VM named "iperf3-client-vm" using the Ubuntu 24.04 LTS image and Standard D2s v5 size.

1. Set the following variables to create the Azure resources.

```
export SUFFIX=$(cat /dev/urandom | LC_ALL=C tr -dc 'a-z0-9' | fold -w 8 | head -n
1)
export LOCAL_NAME="iperf3-client"
export TAGS="Environment=iperf3"
export IPERF_CLIENT_RESOURCE_GROUP_NAME="rg-${LOCAL_NAME}-${SUFFIX}"
export IPERF_CLIENT_REGION="swedencentral"
```

2. Create a resource group in the primary region.

```
az group create --name $IPERF_CLIENT_RESOURCE_GROUP_NAME --location
$IPERF_CLIENT_REGION --tags $TAGS --output tsv
```

3. Create a VNET and subnet in the primary region.

```
export IPERF_CLIENT_VNET_NAME="iperf3-client-vnet"
export VNET_CIDR="10.220.0.0/23"
export IPERF_CLIENT_SUBNET_NAME="iperf3-client-subnet"
export IPERF_CLIENT_SUBNET_CIDR="10.220.0.0/24"

az network vnet create \
   --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   --name $IPERF_CLIENT_VNET_NAME \
   --location $IPERF_CLIENT_REGION \
   --tags $TAGS \
   --address-prefix $VNET_CIDR \
   --output tsv

az network vnet subnet create \
   --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   --vnet-name $IPERF_CLIENT_VNET_NAME \
   --name $IPERF_CLIENT_SUBNET_NAME \
   --address-prefix $IPERF_CLIENT_SUBNET_CIDR \
   --output tsv
```

4. Create a network security group (NSG) to allow SSH and iperf3 traffic.

```
export NSG_NAME="NSG-${SUFFIX}"
export NSG_RULE_NAME="Allow-Access-${SUFFIX}"
export VM_NIC_CLIENT_NAME="VMNic-${SUFFIX}"

az network nsg create \
```

```
    --name $NSG_NAME \
    --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
    --location $IPERF_CLIENT_REGION \
    --tags $TAGS \
    --output tsv
```

5. Create a rule to allow connections to the virtual machine on port 22 for SSH and ports 9000 for iperf3.

```
az network nsg rule create \
    --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
    --nsg-name $NSG_NAME \
    --name $NSG_RULE_NAME \
    --access Allow \
    --protocol '*' \
    --direction Inbound \
    --priority 100 \
    --source-address-prefix '*' \
    --source-port-range '*' \
    --destination-address-prefix '*' \
    --destination-port-range 22 9000 \
    --output tsv
```

6. Use az network nic create to create the network interface for the virtual machine.

```
az network nic create \
    --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
    --name $VM_NIC_CLIENT_NAME \
    --location $IPERF_CLIENT_REGION \
    --accelerated-networking true \
    --ip-forwarding false \
    --subnet $IPERF_CLIENT_SUBNET_NAME \
    --vnet-name $IPERF_CLIENT_VNET_NAME \
    --network-security-group $NSG_NAME \
    --tags $TAGS \
    --output tsv
```

7. Create the Azure VM using the Ubuntu 24.04 LTS image and the Standard_D2s_v5 size.

```
export IPERF_CLIENT_VM_NAME="vm-${LOCAL_NAME}-${SUFFIX}"
export VM_SIZE="Standard_D2s_v5"
export VM_IMAGE="Canonical:ubuntu-24_04-lts:server:latest"
export VM_ADMIN_USER="azureuser"

az vm create \
    --name "$IPERF_CLIENT_VM_NAME" \
    --resource-group "$IPERF_CLIENT_RESOURCE_GROUP_NAME" \
    --location $IPERF_CLIENT_REGION \
```

```
    --size $VM_SIZE \
    --image $VM_IMAGE \
    --nics $VM_NIC_CLIENT_NAME \
    --nic-delete-option Delete \
    --storage-sku os=Premium_LRS \
    --os-disk-caching ReadWrite \
    --os-disk-delete-option Delete \
    --os-disk-size-gb 30 \
    --admin-username $VM_ADMIN_USER \
    --authentication-type ssh \
    --ssh-key-values "$HOME/id_ed25519_levelup_key.pem.pub" \
    --custom-data cloud-init-os-perf.txt \
    --tags $TAGS \
    --output tsv
```

Step 3: Configure Azure VNet Peering

To allow communication between the two Azure VMs, configure VNet peering between the two VNets.

1. Set the following variables to configure VNet peering.

```
export IPERF_SERVER_VNET_ID=$(az network vnet show --resource-group
$IPERF_SERVER_RESOURCE_GROUP_NAME --name $IPERF_SERVER_VNET_NAME --query id --
output tsv)
export IPERF_CLIENT_VNET_ID=$(az network vnet show --resource-group
$IPERF_CLIENT_RESOURCE_GROUP_NAME --name $IPERF_CLIENT_VNET_NAME --query id --
output tsv)
```

2. Create a VNet peering connection from the iperf3-server-vnet to the iperf3-client-vnet.

```
az network vnet peering create \
    --name iperf3-server-to-client \
    --resource-group $IPERF_SERVER_RESOURCE_GROUP_NAME \
    --vnet-name $IPERF_SERVER_VNET_NAME \
    --remote-vnet $IPERF_CLIENT_VNET_ID \
    --allow-vnet-access \
    --allow-forwarded-traffic \
    --output tsv
```

3. Create a VNet peering connection from the iperf3-client-vnet to the iperf3-server-vnet.

```
az network vnet peering create \
    --name iperf3-client-to-server \
    --resource-group $IPERF_CLIENT_RESOURCE_GROUP_NAME \
    --vnet-name $IPERF_CLIENT_VNET_NAME \
    --remote-vnet $IPERF_SERVER_VNET_ID \
    --allow-vnet-access \
```

```
    --allow-forwarded-traffic \
    --output tsv
```

Step 3: Test network performance using iperf3

1. SSH into the iperf3-server-vm via the Azure Bastion Host

> [!CAUTION] Working with Bastion tunnels please run the following commands in a separate terminal window to establish an SSH tunnel to the Azure VM. For those running in Azure Shell please use any terminal mulitplexor like tmux or screen to run the following commands. You could use the CTRL+Z to pause the current process and then run bg 1 to run the process in the background for the current shell session and the same CTRL+Z and bg 2 for the second process.

```
export IPERF_SERVER_RESOURCE_GROUP_NAME="rg-iperf3-server-<SUFFIX>" #replace
<SUFFIX> with the actual value
export IPERF_SERVER_VM_NAME="vm-iperf3-server-<SUFFIX>" #replace <SUFFIX> with the
actual value
export BASTION_NAME="bastion-<SUFFIX>" #replace <SUFFIX> with the actual value

export IPERF_SERVER_VM_ID="$(az vm show -g "$IPERF_SERVER_RESOURCE_GROUP_NAME" -n
"$IPERF_SERVER_VM_NAME" --query id -o tsv)"

az network bastion tunnel -n $BASTION_NAME -g "$IPERF_SERVER_RESOURCE_GROUP_NAME"
\
   --target-resource-id "$IPERF_SERVER_VM_ID" --resource-port 22 --port 2022 #
CTRL+Z ; bg 1
```

```
ssh azureuser@localhost -p 2022 -i $HOME/id_ed25519_levelup_key.pem
```

Start the iperf3 server on the iperf3-server-vm.

```
iperf3 --server --port 9000 --format m
```

> [!TIP] An alternative way of running the iperf3 server is to run it in the background using the nohup command via the Azure CLI run-command extension.

```
az vm run-command invoke \
-g $IPERF_SERVER_RESOURCE_GROUP_NAME \
-n $IPERF_SERVER_VM_NAME \
--command-id RunShellScript \
--scripts "iperf3 --server --port 9000 --format m &" \
-o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

In order to check if the port is listening on the iperf3-server-vm, run the following command:

```
az vm run-command invoke \
   -g $IPERF_SERVER_RESOURCE_GROUP_NAME \
   -n $IPERF_SERVER_VM_NAME \
   --command-id RunShellScript \
   --scripts "ss -ntulp | grep 9000" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

2. Start the iperf3 test on the client VM iperf3-client-vm.

```
export IPERF_SERVER_VM_IP=$(az network nic show -g
$IPERF_SERVER_RESOURCE_GROUP_NAME -n $VM_NIC_SERVER_NAME --query
"ipConfigurations[0].privateIPAddress" -o tsv)

az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "iperf3 --client $IPERF_SERVER_VM_IP --port 9000 --format m --time 60
--parallel 1 --omit 1" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

```
-----------------------------------------------------------
Server listening on 9000 (test #1)
-----------------------------------------------------------
Accepted connection from 10.220.0.4, port 57340
[  5] local 10.230.0.4 port 9000 connected to 10.220.0.4 port 57344
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-1.00   sec  6.25 MBytes  52.4 Mbits/sec                  (omitted)
[  5]   0.00-1.00   sec  27.5 MBytes   230 Mbits/sec
[  5]   1.00-2.00   sec  30.1 MBytes   253 Mbits/sec
[  5]   2.00-3.00   sec  27.1 MBytes   228 Mbits/sec
[  5]   3.00-4.00   sec  26.4 MBytes   221 Mbits/sec
[  5]   4.00-5.00   sec  27.8 MBytes   233 Mbits/sec
[  5]   5.00-6.00   sec  30.6 MBytes   257 Mbits/sec
[  5]   6.00-7.00   sec  31.6 MBytes   265 Mbits/sec
[  5]   7.00-8.00   sec  31.0 MBytes   260 Mbits/sec
[  5]   8.00-9.00   sec  33.2 MBytes   279 Mbits/sec
...
[  5]  59.00-60.00  sec  30.0 MBytes   252 Mbits/sec
[  5]  60.00-60.11  sec  3.12 MBytes   249 Mbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate
[  5]   0.00-60.11  sec  1.70 GBytes   243 Mbits/sec                  receiver
```

> [!IMPORTANT] As we can observe with the default settings the network performance is not optimal and we are not close to the advised line speed of the Azure VMs.

Step 4: Tuning the sysctl parameters

By increasing the kernel buffers and changing the congestion control algorithm to BBR we can improve the network performance.

Modify the sysctl parameters on **BOTH** Azure VMs to improve network performance.

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "cat << EOF > /etc/sysctl.d/99-azure-network-buffers.conf
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.ipv4.tcp_rmem = 4096 67108864 1073741824
net.ipv4.tcp_wmem = 4096 67108864 1073741824
EOF" -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed
'/\[stdout\]/d' | sed '/\[stderr\]/d'

az vm run-command invoke \
   -g $IPERF_SERVER_RESOURCE_GROUP_NAME \
   -n $IPERF_SERVER_VM_NAME \
   --command-id RunShellScript \
   --scripts "cat << EOF > /etc/sysctl.d/99-azure-network-buffers.conf
net.core.rmem_max = 2147483647
net.core.wmem_max = 2147483647
net.ipv4.tcp_rmem = 4096 67108864 1073741824
net.ipv4.tcp_wmem = 4096 67108864 1073741824
EOF" -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed
'/\[stdout\]/d' | sed '/\[stderr\]/d'
```

Apply the changes to the sysctl parameters on **BOTH** Azure VMs.

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "sysctl -p /etc/sysctl.d/99-azure-network-buffers.conf" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'

az vm run-command invoke \
   -g $IPERF_SERVER_RESOURCE_GROUP_NAME \
   -n $IPERF_SERVER_VM_NAME \
   --command-id RunShellScript \
   --scripts "sysctl -p /etc/sysctl.d/99-azure-network-buffers.conf" \
```

```
    -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

Step 5: Repeat the iperf3 test

> [!TIP] If not already loaded please use the modprobe command to enable the TCP BBR congestion control algorithm.

On the client side Azure VM please repeat the iperf3 test to measure the network performance after tuning the sysctl parameters.

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "modprobe tcp_bbr" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'

az vm run-command invoke \
   -g $IPERF_SERVER_RESOURCE_GROUP_NAME \
   -n $IPERF_SERVER_VM_NAME \
   --command-id RunShellScript \
   --scripts "modprobe tcp_bbr" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

Repeat the iperf3 test to measure the network performance after tuning the sysctl parameters.

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "nohup iperf3 --client $IPERF_SERVER_VM_IP --port 9000 --format m --
time 60 --parallel 1 --omit 1 --congestion bbr" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

```
[  5]  12.00-13.00  sec  62.9 MBytes   527 Mbits/sec    0   18.0 MBytes
[  5]  13.00-14.00  sec  63.0 MBytes   529 Mbits/sec    0   18.4 MBytes
[  5]  14.00-15.00  sec   126 MBytes  1056 Mbits/sec    0   18.3 MBytes
...
[  5]  53.00-54.00  sec  63.0 MBytes   529 Mbits/sec    0   18.0 MBytes
[  5]  54.00-55.00  sec   126 MBytes  1057 Mbits/sec    0   18.1 MBytes
[  5]  55.00-56.00  sec  63.2 MBytes   531 Mbits/sec    0   18.4 MBytes
[  5]  56.00-57.00  sec   126 MBytes  1054 Mbits/sec    0   18.3 MBytes
...
```

```
[  5]  59.00-60.00  sec   126 MBytes  1057 Mbits/sec    0   18.1 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bitrate         Retr
[  5]   0.00-60.00  sec  5.03 GBytes   720 Mbits/sec    0             sender
[  5]   0.00-60.10  sec  4.92 GBytes   703 Mbits/sec                  receiver

iperf Done.
```

Step 4: qdisc and tc commands

The `tc` command is used to configure traffic control in Linux. It is used to configure the Linux kernel packet scheduler. The `qdisc` command is used to configure queuing disciplines in Linux. It is used to configure the queuing discipline for a network interface.

To show the queuing discipline on the iperf3-client-vm please run the following command:

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "tc qdisc show" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```

```
qdisc noqueue 0: dev lo root refcnt 2
qdisc mq 0: dev eth0 root
qdisc pfifo_fast 0: dev eth0 parent :1 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1 1 1 1
1 1
qdisc mq 0: dev enP1331s1 root
qdisc pfifo_fast 0: dev enP1331s1 parent :1 bands 3 priomap 1 2 2 2 1 2 0 0 1 1 1
1 1 1 1 1
```

To change the queuing discipline to FQ-CoDel use the command below:

```
az vm run-command invoke \
   -g $IPERF_CLIENT_RESOURCE_GROUP_NAME \
   -n $IPERF_CLIENT_VM_NAME \
   --command-id RunShellScript \
   --scripts "tc qdisc replace dev eth0 root fq_codel" \
   -o JSON | jq -r '.value[0].message' | awk '/\[stdout\]/,/\[stderr\]/' | sed '/\
[stdout\]/d' | sed '/\[stderr\]/d'
```