v1.0 | March 2023 | TN-2172

TECH NOTE

Nutanix Cloud Clusters in Microsoft Azure for Database Workloads

NUTANIX
YOUR ENTERPRISE CLOUD

Copyright

Contents

1. Executive Summary

Nutanix Cloud Clusters (NC2) is a hybrid multicloud platform that enables organizations to run applications in private clouds or multiple public clouds all operated as a single cloud. NC2 empowers IT operators to seamlessly manage and migrate VMs, containers, and applications without expensive retooling or reworking, delivering freedom from cloud lock-in.

NC2 extends the Nutanix platform into the public cloud, providing a seamless private-to-public cloud experience. Prism Central ties your private cloud to the public cloud to create a true hybrid cloud infrastructure.

NC2 on Microsoft Azure reduces the operational complexity of extending, bursting, or migrating applications between clouds. Because NC2 runs Nutanix AOS and AHV, existing IT processes or third-party integrations that work on-premises continue to work no matter where they're running. By deploying NC2 resources through your cloud provider account, you can use your existing cloud provider relationship, credits, commitments, and discounts.

This tech note provides a real-world example of how to use NC2 to run Microsoft SQL Server, Oracle, or PostgreSQL databases. It includes instructions for configuring the Azure environment, deploying an NC2 instance, and running the databases.

2. Introduction

Audience

This tech note is part of the Nutanix Solutions Library. We wrote it for individuals responsible for deploying NC2 in Azure for use with databases. Readers should already be familiar with Nutanix AOS and databases.

Purpose

In this document, we cover the following topics:

- How to create and configure an NC2 cluster
- Performance results for SQL Server, Oracle, and PostgreSQL on Azure Nutanix-Ready nodes

Document Version History

Version Number	Published	Notes
1.0	March 2023	Original publication.

3. Deploying an NC2 Cluster

Networking

Before deploying an NC2 instance using the NC2 deployment tool, you must first identify the appropriate subnets. To create a seamless extension to a private Nutanix cloud, extend your on-premises IP address space to the NC2 instance in Azure using a site-to-site VPN. Start by selecting an address space. For our scenario, the IT networking team assigned us 10.1.0.0/20, so we used this address space to carve out the subnets needed for the NC2 deployment. NC2 requires two subnets—one for the cluster and another for a highly available instance of Prism Central.

The NC2 deployment tool requires a /22 for the two required subnets, so we broke the 10.1.0.0/20 address space into two /22 subnets and two additional /24 subnets for the VPN and user VM (UVM) subnets.

Table: Assigned Subnets

Use	Configuration	CIDR	Netmask	Useable IP Addresses
Cluster	Azure	10.1.0.0/22	255.255.252.0	10.1.0.1–10.1.3.254
Prism Central	Azure	10.1.4.0/22	255.255.252.0	10.1.4.1–10.1.7.254
VPN Virtual Network (VNET)	Azure	10.1.8.0/24	255.255.255.0	10.1.8.1–10.1.8.254
UVM Subnet	Prism Central	10.1.9.0/24	255.255.255.0	10.1.9.1–10.1.9.254

NC2 Deployment Using Console

Once you've identified the subnets, the next step is to deploy the NC2 instance. Select Create Cluster from the NC2 console to create a new VNET in Azure for the cluster and for Prism Central. On the third step of the deployment wizard,

select Create New VNET and enter the CIDR (Classless Inter-Domain Routing) for that VNET. For our setup, we used 10.1.0.0/22 for the cluster.

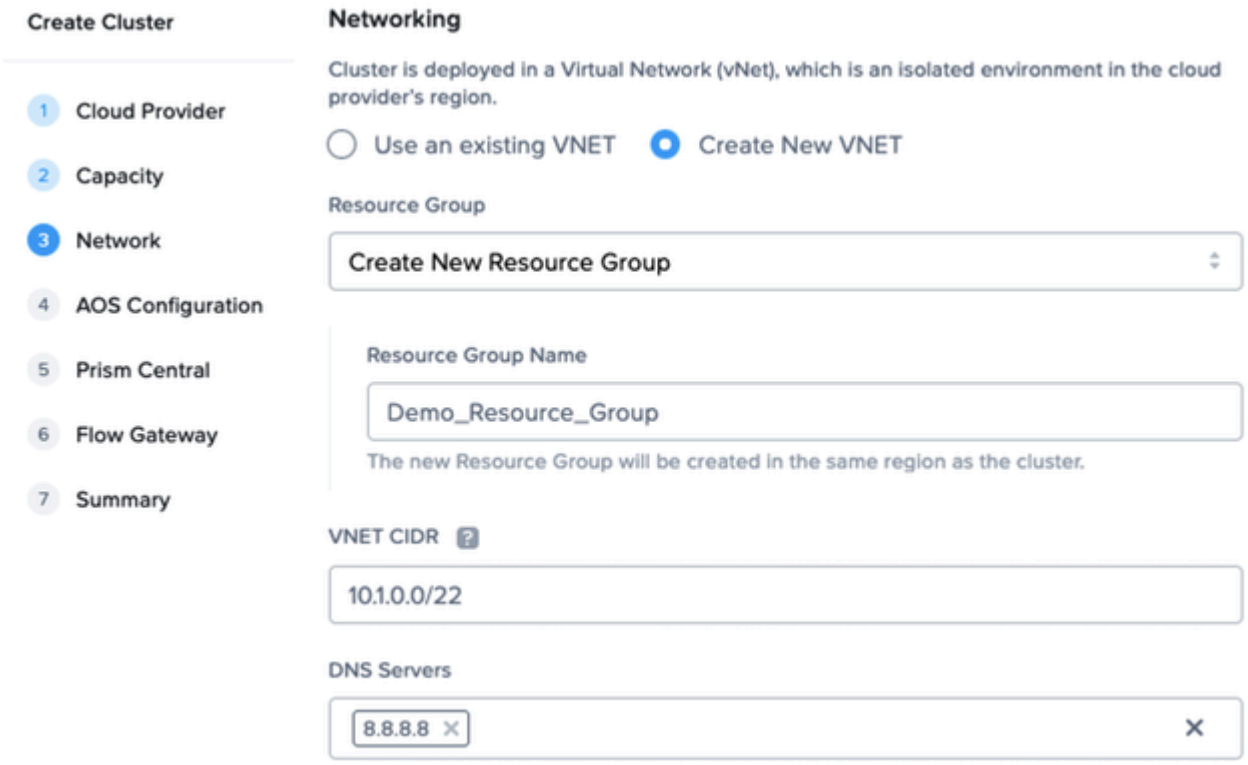

Figure 1: Define the Cluster Network Configuration

On step 5, enter the CIDR (in our case, 10.1.4.0/22) for the Prism Central VNET.

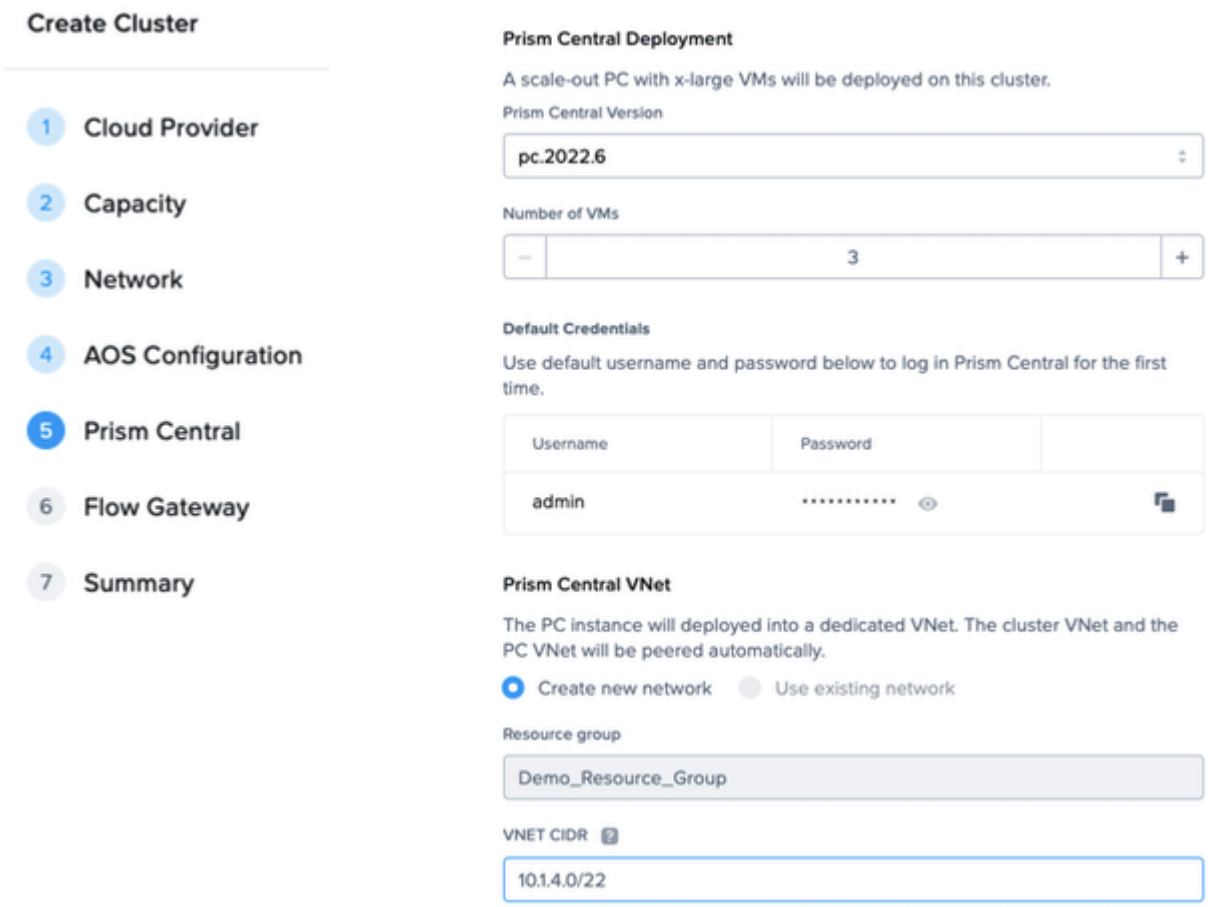

Figure 2: Define the Prism Central Network Configuration

The cluster deployment creates a VNET for the cluster (`Nutanix_Cluster_XXXXXXXXXXX_vnet - 10.1.0.0/22`) and a VNET for Prism Central (`Nutanix_Cluster_XXXXXXXXXXX_pc_vnet_c31 - 10.1.4.0/22`), where xxxx is the serial number of the cluster. Place the cluster on a separate VNET from Prism Central so the Prism Central instance can more easily manage multiple clusters in the future.

Following NC2 deployment, we set up the rest of the environment in Azure to make the cluster accessible from our on-premises systems.

VPN Corporate Firewall Configuration

The next step is to configure firewall access between an on-premises Nutanix cluster and the NC2 instance. This section provides an example firewall configuration to allow communication between two Nutanix clusters.

Group Members

- Administrator Systems (Admin): [List of administrative systems]
- On-Prem-CVM: [List of Controller VM (CVM) IP addresses in the on-premises cluster including the virtual IP addresses]
- On-Prem-UVM: [IP address space for UVMs]
- Azure-CVM: [List of CVMs in Azure including the virtual IP addresses]
- Azure-UVM: [IP address space for UVMs]

Table: Firewall Rules for Access to NC2 Cluster from On-Premises Nutanix Cluster

From	To	Direction	Protocol	Port	Description
Admin	Azure-CVM	To	TCP	9440	UI Access
Admin	Azure-CVM	To	TCP	2009	UI Access
Admin	Azure-CVM	To	TCP	2010	UI Access
Admin	Azure-UVM	To	TCP	22	SSH Access
Admin	Azure-UVM	To	TCP	3389	RDP Access
On-Prem-CVM	Azure-CVM	To, from	TCP	2020	Replication
On-Prem-CVM	Azure-CVM	To, from	TCP	2009	Replication
On-Prem-UVM	Azure-UVM	To, from	TCP	5432	PostgreSQL

VPN Azure Configuration

Once your network team has configured the endpoint of the VPN, the next step is to configure the site-to-site VPN.

Using the Azure portal, create a VNET for the Azure VPN components, including an instance of the local network gateway service and a virtual network gateway service. The local network gateway defines the connection properties between Azure and your onsite VPN endpoint. The virtual network gateway contains routing tables and runs Azure gateway services (see What is Azure VPN Gateway? for more information).

We created the VNET (VPN-VNET) using subnet 10.1.8.0/24, a virtual network gateway (VIRT-VPN-GW), and a local network gateway (LCL-VPN-Gateway).

The virtual network gateway defines the incoming connection from your on-premises VPN endpoint. Provide the public IP address associated with the virtual network gateway to your IT networking team.

The local network gateway defines the outgoing connection to your on-premises VPN endpoint. Gather a shared key and the public IP address of the on-premises gateway (local network gateway IP Address) from your IT networking team.

Tie the created VPN-VNET to your cluster and Prism Central virtual networks by defining subnet peerings. When defining the peerings with the subnet hosting the Azure virtual network gateway, select the following:

- For the VPN-VNET: Use this virtual network's gateway or Route Server
- For the other VNET: Use the remote virtual network's gateway or Route Server

You must select both settings for incoming routing to work.

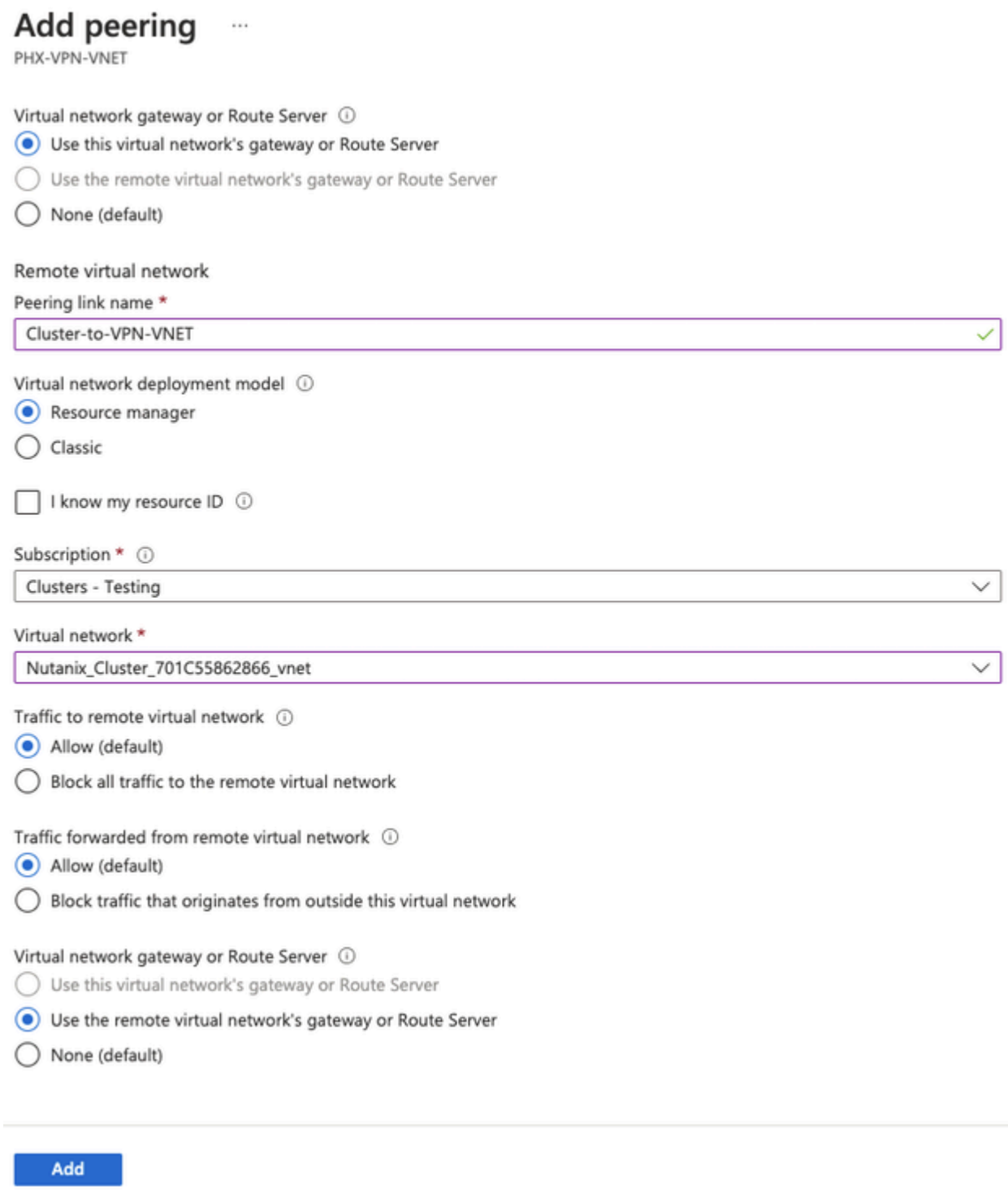

Figure 3: Peering Uses the Remote Virtual Networks Gateway

This configuration enables traffic flow between the corporate network and the NC2 instance in Azure.

The last step is to create a route for incoming traffic to reach the UVMs in the NC2 instance. The UVMs are accessed through the flow gateway VM that you created in Azure during the NC2 deployment process. To route the traffic to the UVM, start by creating a route table in Azure. The route table should route all traffic to the UVM CIDR of 10.1.9.0/24. When creating the route table, define the next hop as a virtual appliance and provide the IP address of the flow VM. Add the route table and any other subnets that need to send traffic to a UVM in the NC2 instance to the VPN subnet.

Networking for User VMs in the NC2 Cluster

During the NC2 cluster installation process, the system creates a virtual private cloud (VPC) by default. You can't delete this special VPC (transit-vpc). The transit VPC contains the subnets used for network access to IP addresses outside the NC2 instance. By default the transit VPC contains a single subnet that provides Network Address Translation (NAT) services to UVMs running in NC2. The NAT-enabled subnet is called overlay-external-subnet-nat. Optionally you can add a second subnet without NAT.

In this scenario we extended the IP address space from our corporate network to Azure. To do so, we create a subnet without NAT enabled using an IP address space that's part of the on-premises address space. Using Prism Central, create a subnet with external connectivity without NAT enabled and then create a new VPC with a subnet that defines the address space to use for the VMs created in NC2.

To create the subnet without NAT enabled in the transit VPC, we use the atlas_cli in the Prism Central VM. When creating this subnet, use an address space that doesn't conflict with your address space. In this example we used 100.64.1.0/24. This address is only used for internal routing and doesn't appear in any route from a UVM. Define the VM address space when you create a new VPC.

Example atlast_cli command to create the no-NAT subnet in the transit-vpc:

```
atlas_cli subnet.create overlay-external-subnet-nonat \
```



```
cidr=100.64.1.0/24 gateway=100.64.1.1 enable_snat=false \
ip_pool_start=100.64.1.100 ip_pool_end=100.64.1.200 \
is_external=true subnet_type=kOverlay network=transit-vpc
```

The next step is to create a new VPC. This VPC, visible to the VMs in NC2, is used to define the subnets. When creating the new VPC, select the NAT and no-NAT subnets. The NAT-enabled subnet is used for accessing the internet, and the no-NAT subnet is used to access the on-premises network.

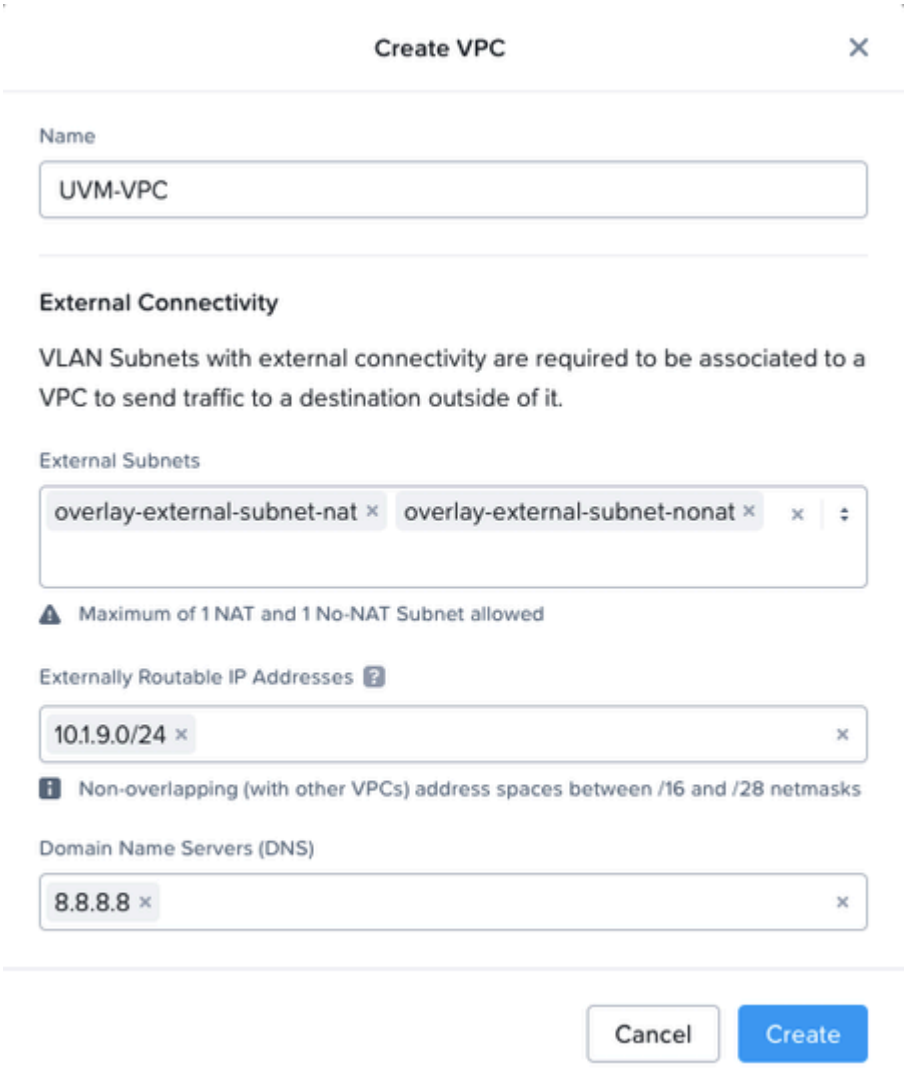

Figure 4: Example Process for Creating a VPC Used for Defining Subnets

Next we add a subnet to the new VPC using the address space that matches our on-premises network.

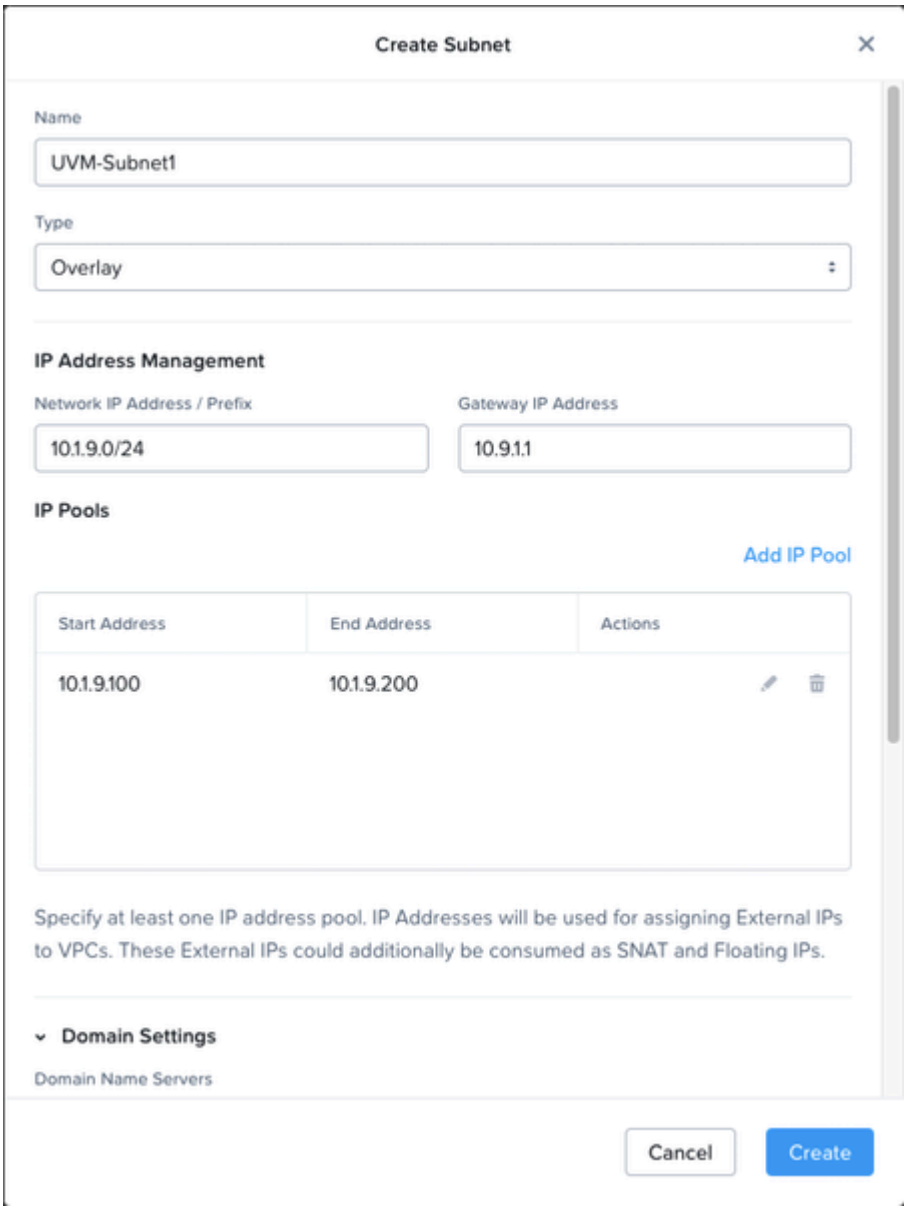

Figure 5: Add a Subnet to the New VPC

Once you've created the new VPC, you need to define routes:

Note: We added two additional routes because the test VMs need to access Prism Central and Prism Element. The Prism Central and cluster subnets are handled differently from other subnets in Azure.

1. A route to your internal network 10.0.0.0/8 using the no-NAT next hop
2. A route to the Prism Element subnet 10.1.0.0/22 through the NAT-enabled next hop
3. A route to the Prism Central subnet 10.1.4.0/22 through the NAT-enabled next hop
4. A static route, for example, 0.0.0.0/0 Default Route, through the NAT-enabled next hop

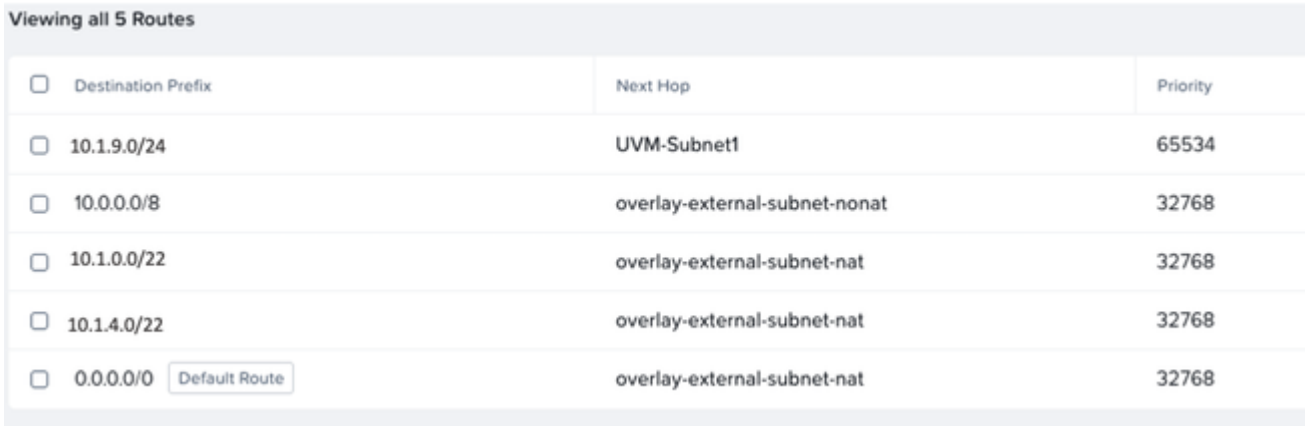

Figure 6: Example of Internal and Default Routes Defined for the New VPC

The next step is to configure VMs to use the new VPC. To use the new VPC, choose UVM-Subnet1 in the UVM-VPC VPC when creating a VM using Prism Central.

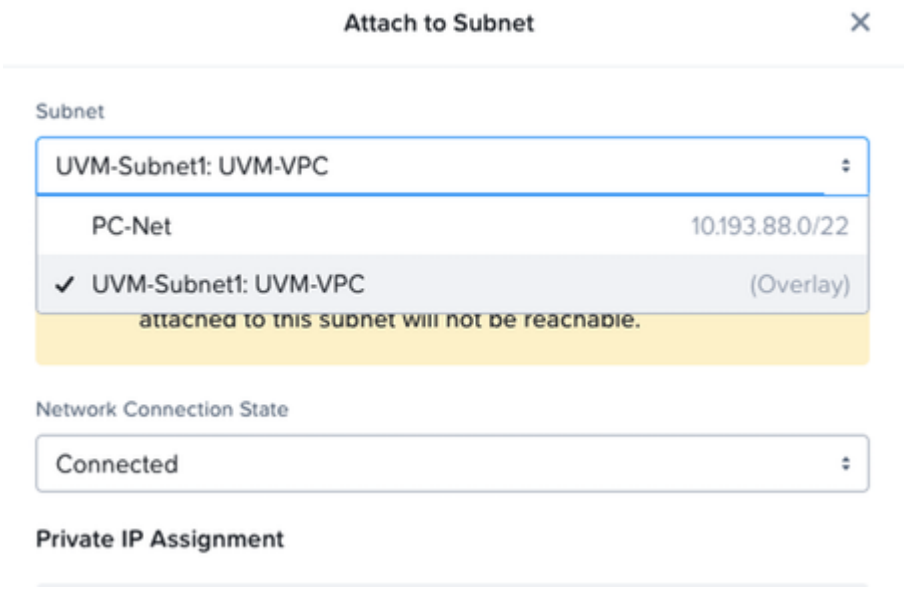

Figure 7: Select the No-NAT Subnet When Creating the VM

The cluster is now operational and ready to run tests.

4. Performance Tests

We tested to collect baseline performance results for database workloads running on NC2 in Azure. We used HammerDB TPC-C/TPROC-C with 5,000 warehouses for SQL Server and PostgreSQL and the Silly Little Oracle Benchmark (SLOB) for Oracle. The tests included a single database instance and a high availability configuration with the benchmark driver VM in the same cluster. We measured database performance on NC2 Azure for the AN36P Nutanix-Ready node type.

Table: NC2 Azure Nutanix-Ready Node and Cluster Details

Component	Ready Node for Nutanix AN36P
Core	Intel 6240, 36 cores, 2.6 GHz
vCPUs	72
RAM	768 GB
Storage	19.95 TB (2 × 375 GB Optane, 6 × 3.2 TB NVMe)
Network	25 Gbps
Size	4 nodes
AHV	5.4.109-2.el7.nutanix.20201105.30398.x86_64
AOS	6.5.0.1

Benchmark Software

HammerDB

We used the HammerDB TPC-C/TPROC-C workload to validate SQL Server and PostgreSQL on NC2. HammerDB TPC-C/TPROC-C transactions run with zero delay, driving the system to the maximum number of IOPS. This approach lets us observe the performance of the system under a most demanding workload.

SLOB

To validate the Oracle solution, we used SLOB to run an online transaction processing (OLTP) workload against the cluster and then monitored the behavior of the underlying infrastructure under load.

For a more CPU-focused benchmark, we turned to Swingbench, using the Order Entry workload for a TPC-C-like load profile.

We used the following tools to run the workload and measure the performance:

- SLOB: SLOB is specifically designed to drive as much storage I/O as possible with the lowest possible CPU usage on the database VM by placing only one data row in each database block. With this approach, the smallest number of rows transfers the largest possible number of database blocks from the disks.
- Swingbench: Swingbench simulates application workloads, and we picked the Order Entry type that's based on Oracle sample oe schema. It causes heavy contention on a small number of tables and highlights CPU capabilities.
- Oracle Automatic Workload Repository (AWR): AWR collects, processes, and maintains performance statistics for database AWR reports to show the differences between two snapshots.
- Iostat: The Linux iostat utility monitors database server VM I/O performance metrics such as read and write latency, IOPS, throughput, I/O request size, and queue length.

SQL Server Performance

The following table contains the parameters we used for the HammerDB workload benchmark configuration.

Table: SQL Server TPC-C Configuration

Parameter	Value
HammerDB version	3.3
Number of virtual users	400
Think time or delay time	1 ms

Parameter	Value
Number of transactions per user	1,000,000
Number of warehouses	5,000
All warehouse option	TRUE
Ramp up time	5 min
Steady runtime	15 min
Number of runs	5
Results collected	Removed outliers (lowest and highest) and averaged the remaining three runs

We ran the tests on a four-node AN36P cluster.

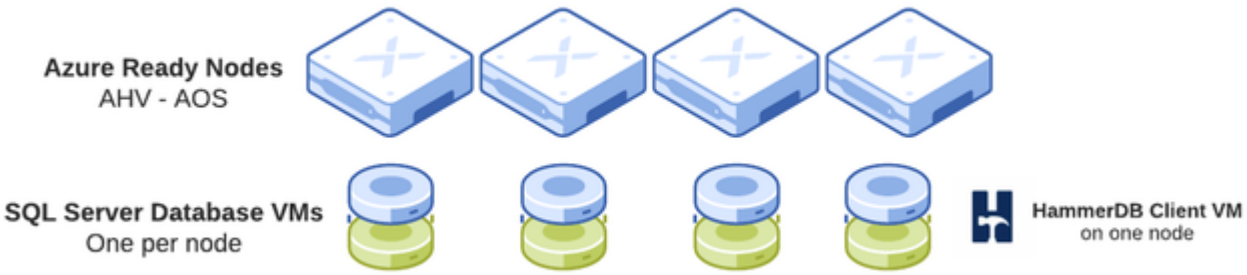

Figure 8: SQL Server Single Instance Benchmark Design

During the tests, we collected the TPROC-C metrics TPM and NOPM:

- TPM (transactions per minute): Number of database transactions completed each minute
- NOPM (new orders per minute): Number of new orders completed each minute

SQL Server Single Instance

We ran HammerDB TPC-C against a single SQL Server instance on the Azure AN36P-based NC2 instance. No other workloads were running on the cluster during the test time.

Table: SQL Server Single Instance Performance on NC2 Azure VM

Metric	NC2 Azure AN36P
TPM	861,341
NOPM	374,455

Scale-Up Tests

To validate scale-up, we ran HammerDB against a SQL Server VM with 16 vCPU and increased the amount of memory allocated to the database server VM in each test: 64 GB, 128 GB, and 256 GB. In each test, we reserved 4 GB of memory for the OS, so the SQL Server instance had 60 GB, 124 GB, and 252 GB as the actual maximum memory.

Table: SQL Server Single Instance Performance on VM with 16 vCPU and Different Amounts of Memory

Metric	64 GB	128 GB	256 GB
TPM	861,341	924,627	1,085,601
NOPM	374,455	401,997	471,948

The TPC-C TPM increased as the memory size increased, which is expected as more data is cached in memory. For the SQL Server VM with 64 GB memory, the performance increase was observed to be around 10 and 24 percent when we increased VM memory to 128 GB and 256 GB, respectively.

We then ran HammerDB against a SQL Server VM with 64 GB of memory and used different quantities of vCPUs: 8 vCPU, 16 vCPU, and 32 vCPU.

Table: SQL Server Single Instance Performance on VM with 64 GB Memory and Different Quantities of vCPUs

Metric	8 vCPU	16 vCPU	32 vCPU
TPM	481,856	861,341	962,962
NOPM	209,368	374,455	418,778
% VM CPU	100	96	77

The results show that this workload can't efficiently use more than 16 vCPU. When run with 8 vCPU, this TPC-C workload was CPU bound and the

performance scaled well to 16 vCPU. Increasing the vCPUs from 16 to 32 showed a reduction in the percentage of total VM CPU utilization.

Scale-Out Tests

For this test, we started with a single SQL Server VM and added one VM at a time for a total of four concurrently running database server VMs, each on their own AN36P Azure Nutanix-Ready node. For each iteration we measured the cluster-wide TPM and NOPM.

Table: SQL Server Concurrent Database Test Results on AN36P

Number of VMs	Cumulative TPM	Cumulative NOPM	Avg. TPM per VM	Avg. NOPM per VM
1	861,341	374,455	861,341	374,455
2	1,578,599	686,813	789,300	343,406
3	2,376,557	1,033,347	792,186	344,449
4	2,521,778	1,095,976	630,445	273,994

The aggregate throughput of the VMs when added to the cluster show near linear scalability in both TPM and NOPM metrics.

SQL Server Always On Availability Group Performance

For this test we deployed an Always On availability group (AG) with one primary replica and one secondary replica on a two-node Windows cluster. These Windows nodes ran in a single NC2 on Azure instance. The VM and database configuration were the same as the single instance test setup described earlier.

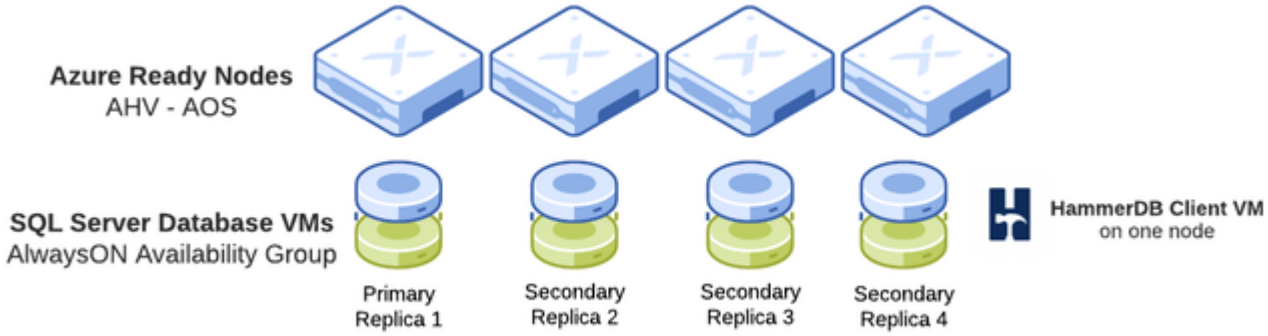

Figure 9: SQL Server AlwaysOn Benchmark Design

Replica 2 and Replica 3 were added for AG scalability testing on the cluster.

We carried out separate test iterations for synchronous and asynchronous replica performance validations and ran the HammerDB TPC-C workload on a VM hosted on the same cluster.

SQL Server Always On Test Configuration:

- Number of replicas: 4 replicas (1 primary and 3 secondary) on NC2 Azure AN36P
- AUTOMATED_BACKUP_PREFERENCE: SECONDARY
- REQUIRED_SYNCHRONIZED_SECONDARIES_TO_COMMIT: 0 for asynchronous; 1 for synchronous
- AVAILABILITY_MODE: ASYNCHRONOUS_COMMIT / SYNCHRONOUS_COMMIT
- SEEDING_MODE: MANUAL
- SECONDARY_ROLE(ALLOW_CONNECTIONS): NO

Two-Node Tests

We ran the TPC-C workload against the primary replica with asynchronous replication enabled, then we configured the replicas for synchronous replication and repeated the same set of tests.

For these tests, we used the same HammerDB configuration as for the single-instance SQL Server tests. We conducted the tests with no other workloads running on the cluster.

Table: TPM and NOPM for SQL Server Always On Validation

Metric	Asynchronous	Synchronous
TPM	829,939	728,957
NOPM	361,187	316,880

The performance result is the average of three TPC-C test iterations. The synchronous configuration delivered around 12 percent lower performance than the asynchronous setup on the NC2 Azure AN36P instance. In the case of synchronous replicas, the database transactions need to be committed on both the primary and secondary replicas before sending acknowledgement, so we expected a performance impact compared to the asynchronous replicas.

Scale-Up Tests

We conducted memory and CPU scale-up tests for the SQL Server Always On configuration. All the scale-up tests ran on a two-node SQL Server cluster configuration, with the TPC-C workload running against the primary replica.

To validate the scale-up topology, we ran HammerDB against a two-node SQL Server Always On cluster. We configured each of the cluster's nodes with 16 vCPU, and we ran separate test iterations for different memory sizes: 64 GB, 128 GB, and 256 GB.

Table: 16 vCPU SQL Server Always On Memory Scale-Up Tests

Metric	64 GB	128 GB	256 GB
TPM	829,939	884,881	997,588
NOPM	361,187	384,754	433,554

TPM increased as the memory size increased; this scaling is expected as there's more room in memory for data cache.

We also ran tests with TPC-C against a two-node SQL Server Always On cluster with 8 vCPU, 16 vCPU, and 32 vCPU. During all of these tests, the replica VM's memory was 64 GB.

Table: 64 GB of VM Memory SQL Server Always On vCPU Scale-Up Tests

Metric	8 vCPU	16 vCPU	32 vCPU
TPM	394,745	829,939	819,112
NOPM	171,608	361,187	356,247

For this workload, 16 vCPU yielded better performance than 32 vCPU, which demonstrates that scale-up performance depends on the workload. Increasing the vCPUs on a SQL Server might not always improve performance for a workload.

Scale-Out Tests

We scaled out the Always On cluster to four replicas (one primary and three secondary replicas) and ran the HammerDB workload against the primary replica to evaluate the performance. The scale-out test results are the result of using asynchronous replication.

Table: TPM and NOPM for SQL Server Always On Scale-Out Tests

Metric	2 Replicas	3 Replicas	4 Replicas
TPM	829,939	715,171	689,697
NOPM	361,187	310,891	299,699

On the NC2 Azure AN36P instance, the performance dropped 13.8 percent when moving from two to three replicas and 16.9 percent when moving from two to four replicas. We recommended configuring only the replicas meant for high availability on the same cluster. Host the rest of the replicas on different clusters to account for disaster recovery and performance implications.

Oracle Performance

Oracle Workload Test Configuration: SLOB

The benchmark setup consisted of four single-instance Oracle 19.3 databases that each ran on their own 32 GB, 8 vCPU database VM, with one VM per node on the four-node cluster. The first test used only database server VM 1, the second used database server VMs 1 and 2, the third used database server VMs 1, 2, and 3, and the fourth test used all four database server VMs. This test demonstrates how the workload scales and provides consistently low read latency as you add databases to the cluster.

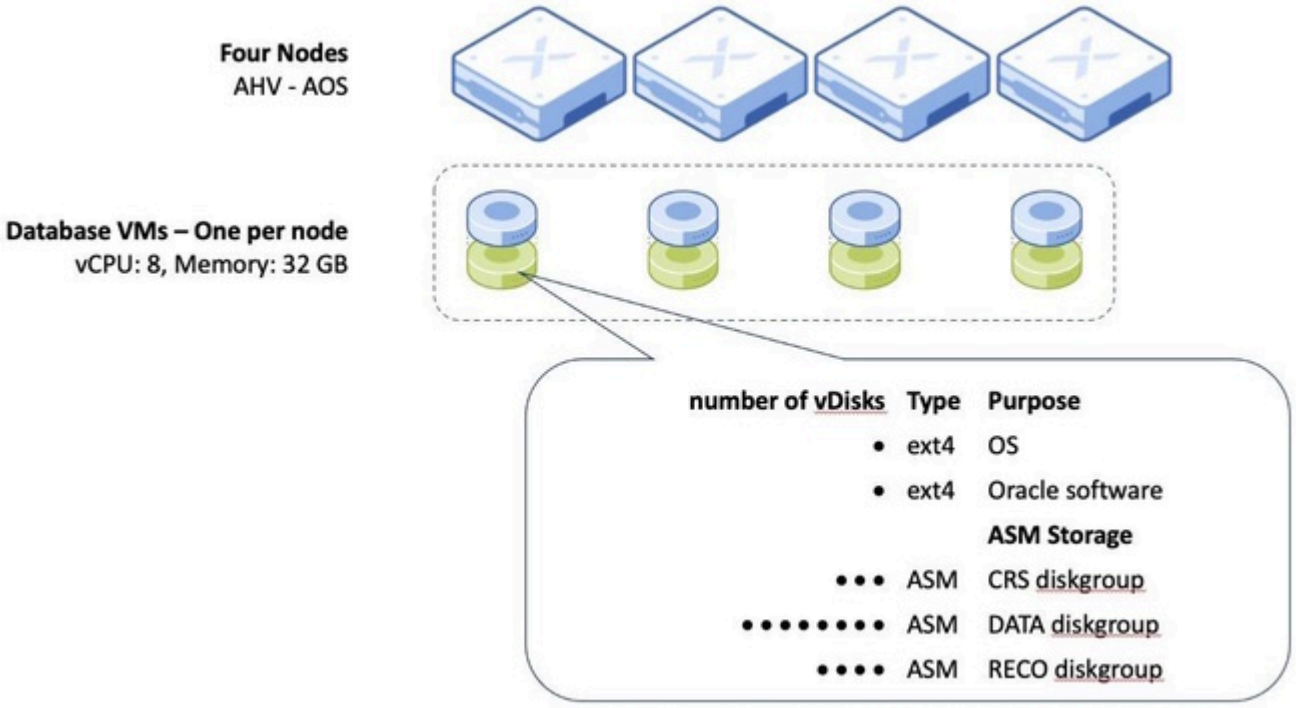

Figure 10: Oracle NC2 Azure AN36P Four-Node Test Configuration

For the benchmark, we used SLOB 2.5.3.1 to run an OLTP-like workload (read and update). We left the SLOB parameters at their default settings, apart from those shown in the following table.

Table: SLOB Benchmark Parameters

Parameter	Value
RUN_TIME	1800
SCALE	32 GB
Number of schemas	32

The SLOB database configuration used 32 database schemas per database, with a per-schema size of around 32 GB (SLOB parameter SCALE=32G), for a total benchmark-relevant database size of around 1 TB. We used the entire database as the working set.

We used a separate VM as the benchmark coordinator to initiate the benchmark runs on the database VMs. SLOB ran on the database VMs.

Single Oracle VM Test: SLOB

For the first Oracle benchmark, we ran SLOB against one Oracle database VM and measured the performance.

Table: Oracle SLOB Results on AN36P Node

Metric	1 Database VM
Read IOPS	42,374
Write IOPS	15,786
Total IOPS	58,160
Read latency (ms)	0.92
Write latency (ms)	1.25

The results of the single-VM test demonstrate the performance you can achieve when you don't run any other workloads in the cluster.

Oracle Scale-Out Test: SLOB

For this benchmark setup, we started with one Oracle database and gradually scaled out to one Oracle database VM per node.

Table: Oracle SLOB Scale-up Test Results on AN36P Cluster

Metric	1 Database VM	2 Database VMs	3 Database VMs	4 Database VMs
Read IOPS	42,374	79,632	112,598	145,727
Write IOPS	15,786	29,606	41,743	53,768
Total IOPS	58,160	109,238	154,340	199,496
Read latency (ms)	0.92	0.95	1.00	1.02
Write latency (ms)	1.25	1.36	1.51	1.62

The SLOB results show that cumulative performance increases as more database VMs become active. The total IOPS topped out at almost 200,000 IOPS with four database VMs running concurrently.

As we added database VMs, read latency remained stable and write latency increased a few percent each time we added a VM, as expected.

Oracle Workload Test Configuration: Swingbench

The benchmark setup consisted of four single-instance Oracle 19.3 databases that each ran on their own 256 GB, 16 vCPU database VM, with one VM per node on the four-node cluster.

For the benchmark, we used Swingbench 2.6, build 1149, to run the Order Entry workload. We left the Swingbench parameters at their default settings, apart from those shown in the following table.

Table: Swingbench Benchmark Parameters

Parameter	Value
Ramp-up time	1 minute
Benchmark duration	60 minutes
Ramp-down time	1 minute
Number of sessions per database	300
Size per Swingbench schema	~ 128 GB
Benchmark configuration (unmodified)	SOE_Client_Side.xml

We sized the Swingbench schema to fit into the database cache, as in a well-tuned OLTP system. This configuration highlights CPU performance.

The Swingbench workloads were driven from separate load driver VMs and placed onto the same cluster (one per node). These driver VMs had 8 vCPU and 32 GB memory each.

Oracle Scale-Out Test: Swingbench

The first test used only database server VM 1, the second used database server VMs 1 and 2, the third used database server VMs 1, 2, and 3, and the fourth test used the database server VMs on all four nodes. This test demonstrates how the workload scales and provides consistently high TPM as you add databases to the cluster.

Table: Swingbench Multi-Database Scale-Out Results

Number of Databases	TPM	Relative Comparison
1	228,057	100%
2	452,209	198%
3	682,946	299%
4	913,732	401%

The benchmark results show near-perfect linear scalability.

PostgreSQL Performance

We used HammerDB TPROC-C to measure PostgreSQL performance. For this test, we used a single-instance database and a high availability database configuration using PostgreSQL streaming replication with both instances located in the same Nutanix cluster.

PostgreSQL Workload Test Configuration

The test setup consisted of PostgreSQL14 databases that each ran on their own 128 GB, 8 vCPU database VM, with one VM per node on the Azure NC2 instances.

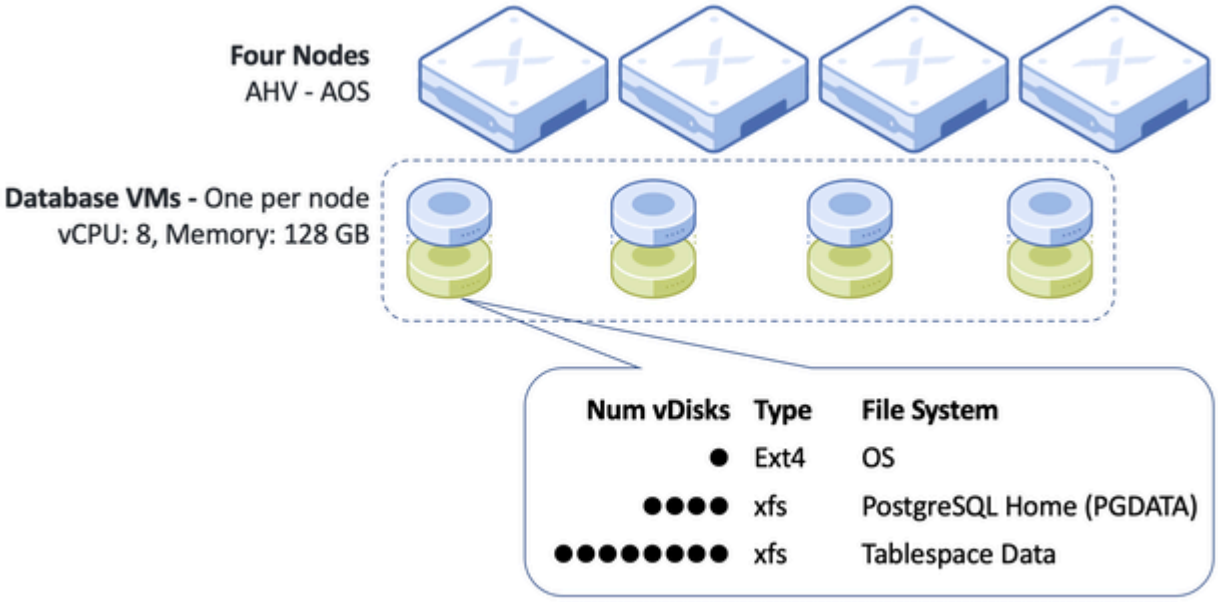

Figure 11: PostgreSQL on NC2 Azure Four-Node AN26P Test Configuration

The majority of PostgreSQL parameters are stored in the /opt/postgresql.conf file. The following table contains the output of an optimized postgresql.conf file, which serves as a guide for following our best practices.

Table: Nondefault PostgreSQL Configuration Parameters

Parameter	Value	Notes
max_wal_size	32 GB	Increased to match the benchmark high transaction rate
work_mem	8 MB	Reduce for OLTP
shared_buffers	32 GB	25% of system memory

We collected the TPROC-C metrics TPM and NOPM.

PostgreSQL Single-Instance and High Availability Tests

We ran HammerDB TPC-C against deployment on the Azure AN36P-based NC2 instance. No other workloads were running on the cluster during the test time.

We validated two types of deployments on Azure.

- PostgreSQL with single-instance database
- PostgreSQL with high availability database

Table: PostgreSQL TPROC-C Results

Metric	AN36P	AN36P High Availability
TPM	149,857	144,504
NOPM	65,141	62,819

The results of each deployment test demonstrate the performance you can achieve when you don't run any other workloads in the cluster.

5. Conclusion

NC2 on Azure provides the same Nutanix AOS interface on-premises and in the public cloud so you can easily expand your database platform into the public cloud for cloud migration, capacity expansion, or disaster recovery.

About Nutanix

Nutanix is a global leader in cloud software and a pioneer in hyperconverged infrastructure solutions, making clouds invisible and freeing customers to focus on their business outcomes. Organizations around the world use Nutanix software to leverage a single platform to manage any app at any location for their hybrid multicloud environments. Learn more at www.nutanix.com or follow us on Twitter @nutanix.

List of Figures