NUTANIX

YOUR ENTERPRISE CLOUD

AHV

Nutanix Best Practices

Version 4.6 • February 2020 • BP-2029

Copyright

Contents

1. Executive Summary

The Nutanix Enterprise Cloud is compute agnostic, providing the flexibility to choose the hypervisor and cloud service combination that suits your needs today, as well as the freedom to move workloads on demand. The native hypervisor for the Nutanix solution is AHV, which lets administrators unbox a Nutanix system and immediately start loading VMs with enterprise-class virtualization capabilities at no extra cost.

This document presents an overview of AHV features and offers best practices for integrating these features into a production datacenter. We address such virtualization topics as VM deployment, CPU configuration, oversubscription, high availability, data protection, and live migration. We also make recommendations for such networking topics as VLANs and segmentation, load balancing, and IP address management. Reading this document prepares administrators to architect and deploy a VM environment using AHV.

2. Introduction

With AHV, Nutanix adapted proven open source technology and wrapped it in the easy-to-use and efficient Prism interface. Underneath Prism, Nutanix configured the hypervisor to take advantage of the seamless scale and reliability of the Acropolis Distributed Storage Fabric (DSF). As with all design prioritizing simplicity, we removed unnecessary decision making from the user experience. The remaining decisions include points like host networking and VM-level features that sometimes require the administrator to make a choice. This document is the guide to those choices.

We describe and define AHV's features and offer best practices for configuring and operating the hypervisor on Acropolis. After a brief introduction to the Acropolis architecture, we turn to the hypervisor itself. This best practices guide provides the information you need to connect to the network and start running VMs. It also prepares you to customize AHV's features for optimal performance and to make full use of its extensive capabilities.

Table 1: Document Version History

Version Number	Published	Notes
1.0	November 2015	Original publication.
2.0	August 2016	Updated to include changes for the AOS 4.6 release.
3.0	January 2017	Updated to include changes for the AOS 5.0 release.
3.1	August 2017	Updated to include changes for the AOS 5.1 release.
4.0	December 2017	Updated to include changes for the AOS 5.5 release.
4.1	May 2018	Updated the NUMA and vNUMA sections.
4.2	October 2018	Updated product naming and the Disks section.
4.3	December 2018	Updated the VM Data Protection section.
4.4	September 2019	Updated the AHV Turbo Technology section.
4.5	October 2019	Updated the Virtual Machine High Availability and VM Data Protection sections.
4.6	February 2020	Updated CPU recommendations.

3. Nutanix Enterprise Cloud Overview

Nutanix delivers a web-scale, hyperconverged infrastructure solution purpose-built for virtualization and cloud environments. This solution brings the scale, resilience, and economic benefits of web-scale architecture to the enterprise through the Nutanix Enterprise Cloud Platform, which combines three product families—Nutanix Acropolis, Nutanix Prism, and Nutanix Calm.

Attributes of this Enterprise Cloud OS include:

- Optimized for storage and compute resources.
- Machine learning to plan for and adapt to changing conditions automatically.
- Self-healing to tolerate and adjust to component failures.
- API-based automation and rich analytics.
- Simplified one-click upgrade.
- Native file services for user and application data.
- Native backup and disaster recovery solutions.
- Powerful and feature-rich virtualization.
- Flexible software-defined networking for visualization, automation, and security.
- Cloud automation and life cycle management.

Nutanix Acropolis provides data services and can be broken down into three foundational components: the Distributed Storage Fabric (DSF), the App Mobility Fabric (AMF), and AHV. Prism furnishes one-click infrastructure management for virtual environments running on Acropolis. Acropolis is hypervisor agnostic, supporting two third-party hypervisors—ESXi and Hyper-V—in addition to the native Nutanix hypervisor, AHV.

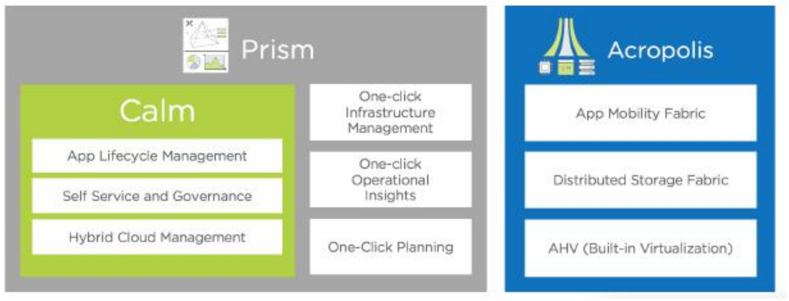

Prism
Calm
App Lifecycle Management
Self Service and Governance
Hybrid Cloud Management
One-click Infrastructure Management
One-click Operational Insights
One-Click Planning
Acropolis
App Mobility Fabric
Distributed Storage Fabric
AHV (Built-in Virtualization)

Figure 1: Nutanix Enterprise Cloud

3.1. Nutanix Acropolis Architecture

Acropolis does not rely on traditional SAN or NAS storage or expensive storage network interconnects. It combines highly dense storage and server compute (CPU and RAM) into a single platform building block. Each building block delivers a unified, scale-out, shared-nothing architecture with no single points of failure.

The Nutanix solution requires no SAN constructs, such as LUNs, RAID groups, or expensive storage switches. All storage management is VM-centric, and I/O is optimized at the VM virtual disk level. The software solution runs on nodes from a variety of manufacturers that are either all-flash for optimal performance, or a hybrid combination of SSD and HDD that provides a combination of performance and additional capacity. The DSF automatically tiers data across the cluster to different classes of storage devices using intelligent data placement algorithms. For best performance, algorithms make sure the most frequently used data is available in memory or in flash on the node local to the VM.

To learn more about the Nutanix Enterprise Cloud, please visit the Nutanix Bible and Nutanix.com.

3.2. AHV Networking Overview

AHV uses Open vSwitch (OVS) to connect the CVM, the hypervisor, and guest VMs to each other and to the physical network. The OVS service, which starts automatically, runs on each AHV node.

Open vSwitch

OVS is an open source software switch implemented in the Linux kernel and designed to work in a multiserver virtualization environment. By default, OVS behaves like a layer-2 learning switch that maintains a MAC address table. The hypervisor host and VMs connect to virtual ports on the switch.

OVS supports many popular switch features, including VLAN tagging, Link Aggregation Control Protocol (LACP), port mirroring, and quality of service (QoS), to name a few. Each AHV server maintains an OVS instance, and all OVS instances combine to form a single logical switch. Constructs called bridges manage the switch instances residing on the AHV hosts.

Bridges

Bridges act as virtual switches to manage network traffic between physical and virtual network interfaces. The default AHV configuration includes an OVS bridge called br0 and a native Linux bridge called virbr0. The virbr0 Linux bridge carries management and storage communication between the CVM and AHV host. All other storage, host, and VM network traffic flows through the br0 OVS bridge. The AHV host, VMs, and physical interfaces use “ports” for connectivity to the bridge.

Ports

Ports are logical constructs created in a bridge that represent connectivity to the virtual switch. Nutanix uses several port types, including internal, tap, VXLAN, and bond:

- An internal port—with the same name as the default bridge (br0)—provides access for the AHV host.
- Tap ports act as bridge connections for virtual NICs presented to VMs.
- VXLAN ports are used for the IP address management functionality provided by Acropolis.
- Bonded ports provide NIC teaming for the physical interfaces of the AHV host.

Bonds

Bonded ports aggregate the physical interfaces on the AHV host. By default, a bond named br0-up is created in bridge br0. After the node imaging process, all interfaces are placed within a single bond, which is a requirement for the Foundation imaging process. Changes to the default bond, br0-up, often rename this to bond0. Nutanix recommends using the name br0-up to quickly identify the interface as the bridge br0 uplink.

OVS bonds allow for several load-balancing modes, including active-backup, balance-slb, and balance-tcp. LACP can also be activated for a bond. The “bond_mode” setting is not specified during installation and therefore defaults to active-backup, which is the recommended configuration. You can find additional information on bonds in the AHV Networking guide.

The following diagram illustrates the networking configuration of a single host immediately after imaging. This is the configuration we recommend if you are using at least two NICs of the same speed (for example, 10 Gb) and the remaining NICs are disconnected.

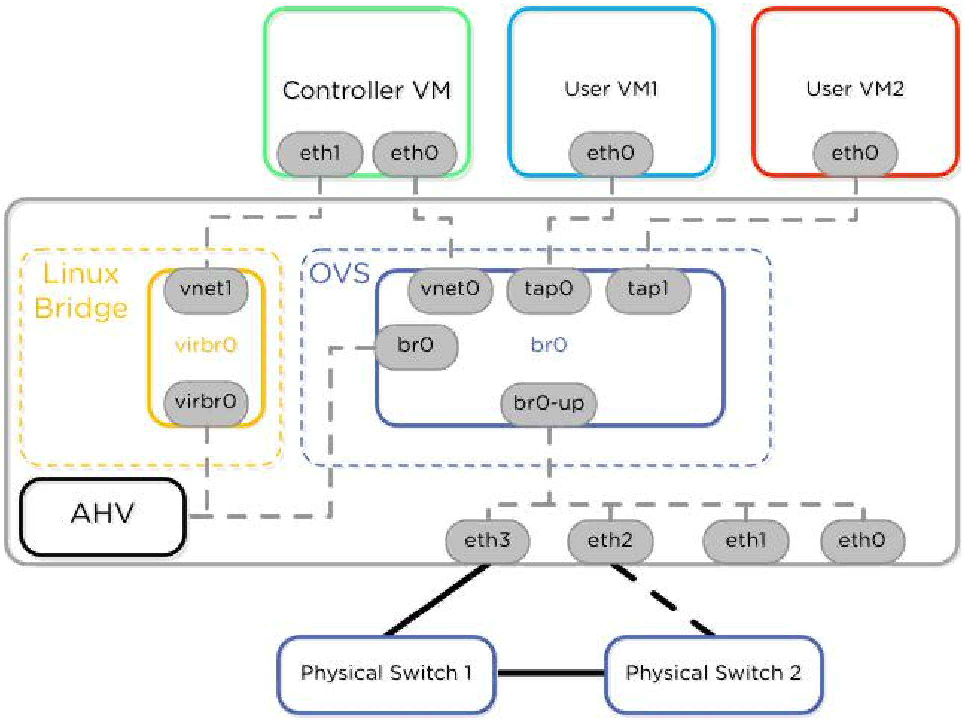


Figure 2: Post-Foundation Network State

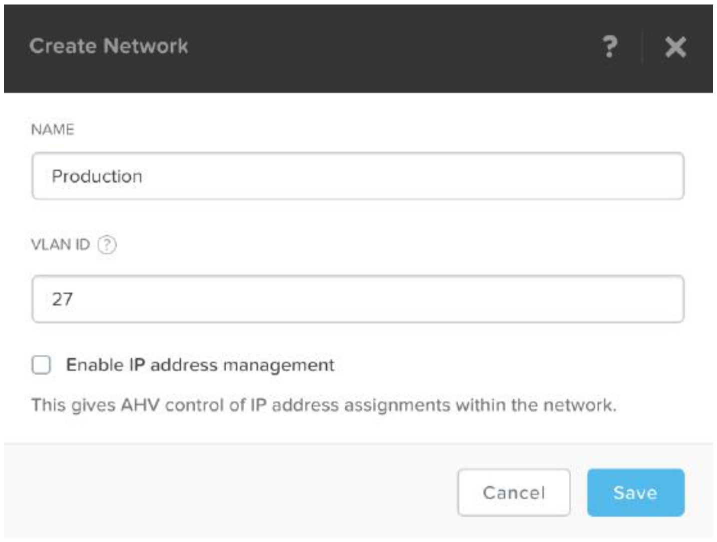
Note: Only utilize NICs of the same speed within the same bond.

We recommend using only the 10 Gb NICs and leaving the 1 Gb NICs unplugged if you do not need them.

Virtual Local Area Networks (VLANs)

AHV supports the use of VLANs for the CVM, AHV host, and user VMs. We discuss the steps for assigning VLANs to the AHV host and CVM in the AHV Networking guide. For user VMs, you can create and manage the networks that a virtual NIC uses in the Prism GUI, Acropolis CLI (aCLI), or by using REST.

Each virtual network that Acropolis creates is bound to a single VLAN. The following figure displays the process of creating a network using Prism, where we assign a friendly network name and VLAN ID for a user VM network called Production.



Production

27

☐ Enable IP address management

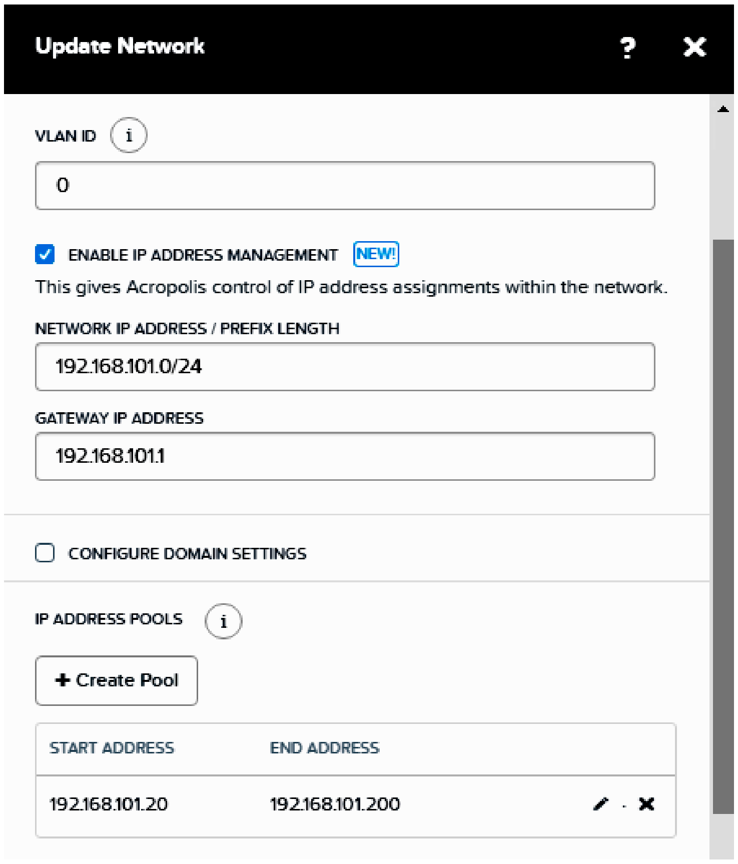
This gives AHV control of IP address assignments within the network.

Figure 3: Prism UI Network Creation

Although a virtual network is configured for a specific VLAN, it is possible to configure a virtual NIC in either “access” or “trunked” mode. By default all virtual NICs are created in access mode, which allows a single configured VLAN based on the virtual network. For information on configuring virtual NICs in trunked mode, please see the AHV Networking guide.

IP Address Management (IPAM)

In addition to network creation and VLAN management, AHV also supports IP address management (IPAM), as shown in the figure below. IPAM enables AHV to assign IP addresses automatically to VMs using the Dynamic Host Configuration Protocol (DHCP). Each virtual network and associated VLAN can be configured with a specific IP subnet, associated domain settings, and group of IP address pools available for assignment. Acropolis uses VXLAN and OpenFlow rules in OVS to intercept DHCP requests from user VMs so that the configured IP address pools and settings are used.



Update Network

VLAN ID: 0

[✓] ENABLE IP ADDRESS MANAGEMENT NEW!

This gives Acropolis control of IP address assignments within the network.

NETWORK IP ADDRESS / PREFIX LENGTH: 192.168.101.0/24

GATEWAY IP ADDRESS: 192.168.101.1

[ ] CONFIGURE DOMAIN SETTINGS

IP ADDRESS POOLS

START ADDRESS	END ADDRESS
192.168.101.20	192.168.101.200

Figure 4: IPAM

Administrators can use Acropolis with IPAM to deliver a complete virtualization deployment, including network management, from the unified Prism interface. This capability radically simplifies the traditionally complex network management associated with provisioning VMs and assigning network addresses. To avoid address overlap, be sure to work with your network team to reserve a range of addresses for VMs before enabling the IPAM feature.

The Acropolis master assigns an IP address from the address pool when creating a managed VM NIC; the address releases back to the pool when the VM NIC or VM is deleted.

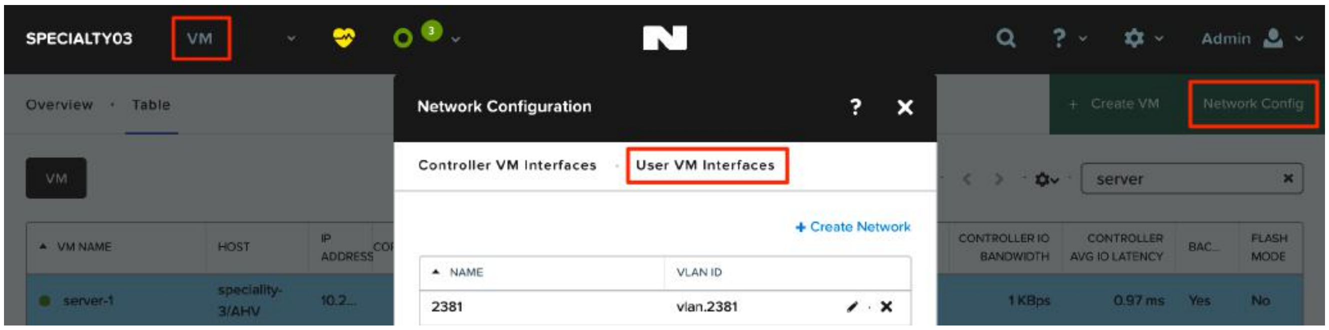
4. AHV Best Practices

4.1. Networking

The overarching best practice is to keep things simple. The default networking configuration we described above provides a highly available environment that performs well.

Viewing Network Configuration for VMs in Prism

Click **Network Configuration**, then **User VM Interfaces** to view VM virtual networks from the VM page, as shown in the figure below.



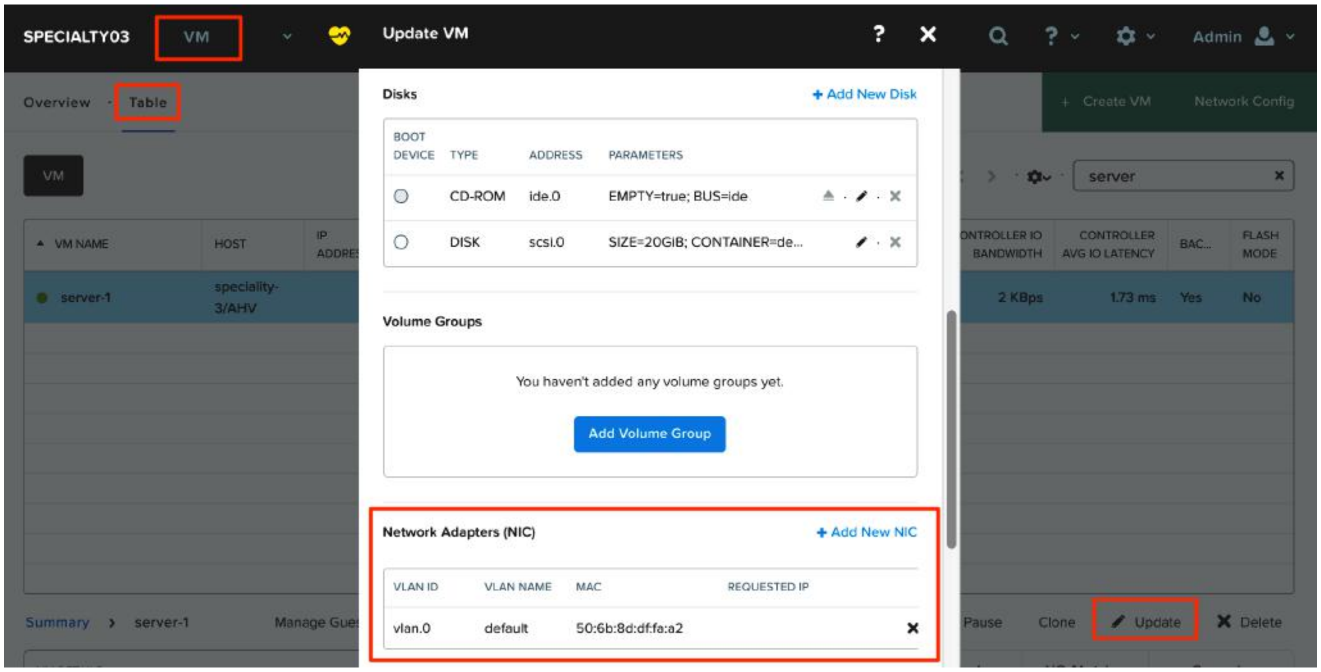
VM NAME	HOST	IP ADDRESS
server-1	speciality-3/AHV	10.2...

NAME	VLAN ID
2381	vlan.2381

CONTROLLER IO BANDWIDTH	CONTROLLER AVG IO LATENCY	BAC...	FLASH MODE
1 KBps	0.97 ms	Yes	No

Figure 5: Prism UI Network List

You can see individual VM network details under the Table view on the VM page by selecting the desired VM and choosing **Update**, as shown in the figure below.



Disks

BOOT DEVICE	TYPE	ADDRESS	PARAMETERS
○	CD-ROM	ide.0	EMPTY=true; BUS=ide
○	DISK	scsi.0	SIZE=20GiB; CONTAINER=de...

Volume Groups

You haven't added any volume groups yet.

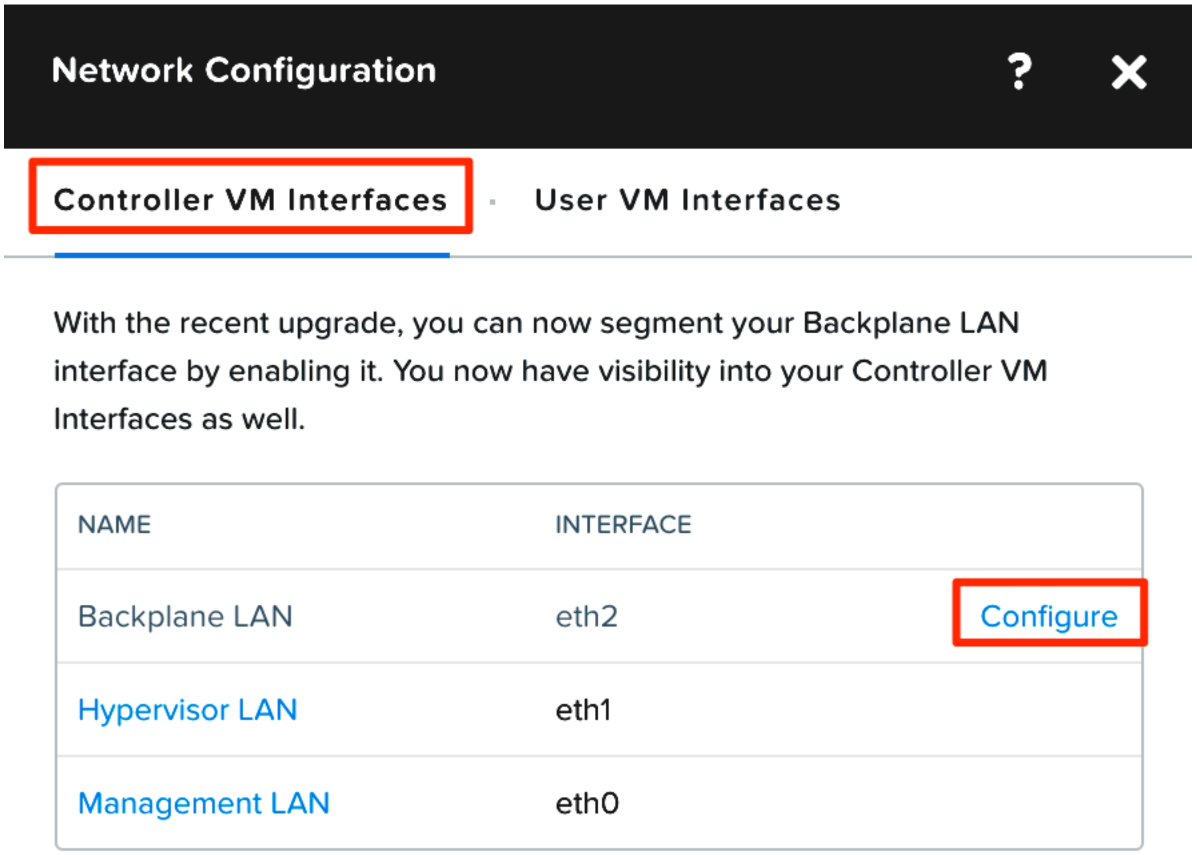
Network Adapters (NIC)

VLAN ID	VLAN NAME	MAC	REQUESTED IP
vlan.0	default	50:6b:8d:df:fa:a2	

VM NAME	HOST	IP ADDRE	ONTROLLER IO BANDWIDTH	CONTROLLER AVG IO LATENCY	BAC...	FLASH MODE
● server-1	speciality-3/AHV		2 KBps	1.73 ms	Yes	No

Figure 6: Prism UI VM Network Details

Use the **Controller VM Interfaces** menu to create a private CVM backplane network if desired. The optional backplane LAN creates a dedicated interface in a separate VLAN on all CVMs in the cluster for exchanging storage replication traffic. Use this backplane network only if there is a requirement to separate CVM management traffic from storage replication traffic.



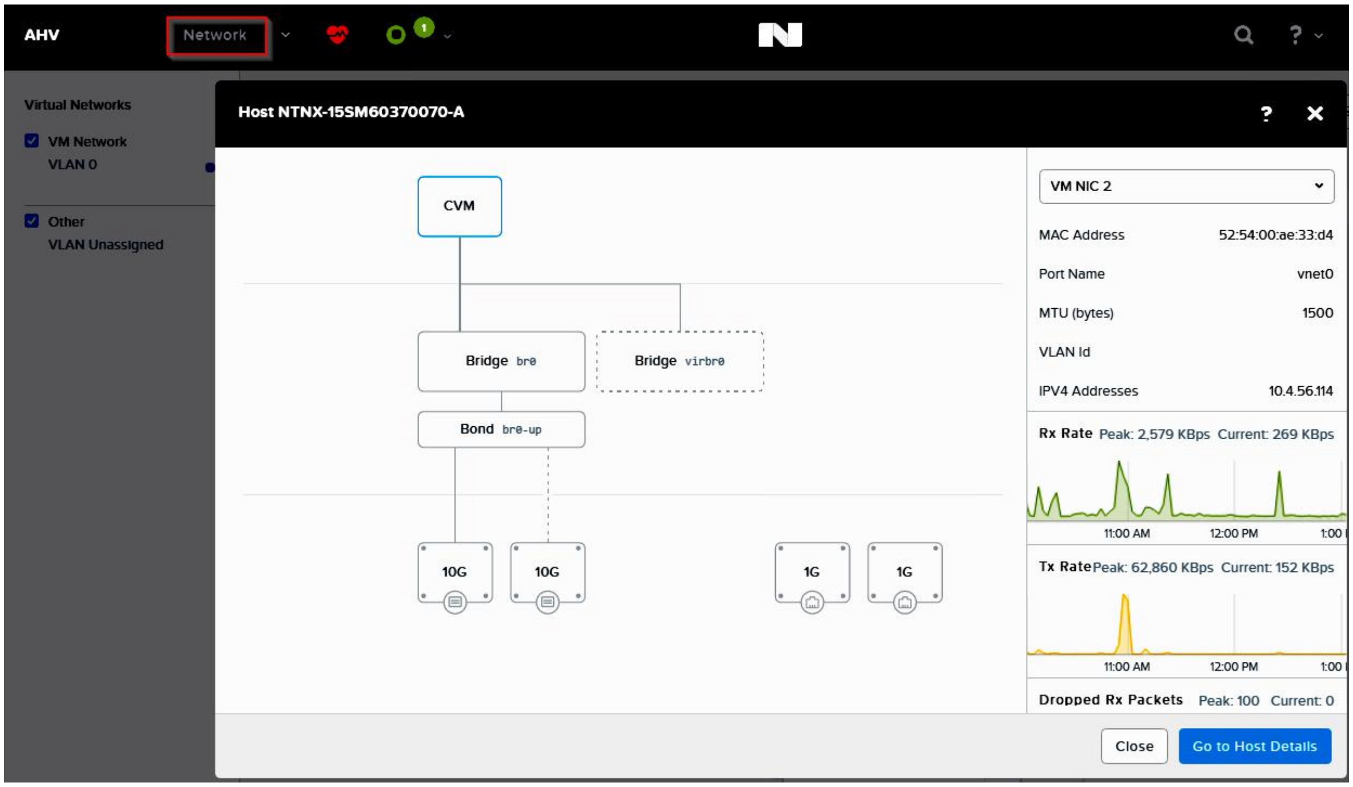
With the recent upgrade, you can now segment your Backplane LAN interface by enabling it. You now have visibility into your Controller VM Interfaces as well.

NAME	INTERFACE	
Backplane LAN	eth2	Configure
Hypervisor LAN	eth1	
Management LAN	eth0	

Figure 7: Prism UI CVM Network Interfaces

Viewing Network Configuration for Hosts in Prism

Select the Network page to view VM and host-specific networking details. Selecting a specific AHV host provides the network configuration as shown in the figure below.



MAC Address	52:54:00:ae:33:d4
Port Name	vnet0
MTU (bytes)	1500
VLAN Id	
IPV4 Addresses	10.4.56.114

Figure 8: Prism UI Host Network Details

VLANs for AHV Hosts and CVMs

The recommended VLAN configuration is to place the CVM and AHV in the default "untagged" (or native) VLAN, as shown below.

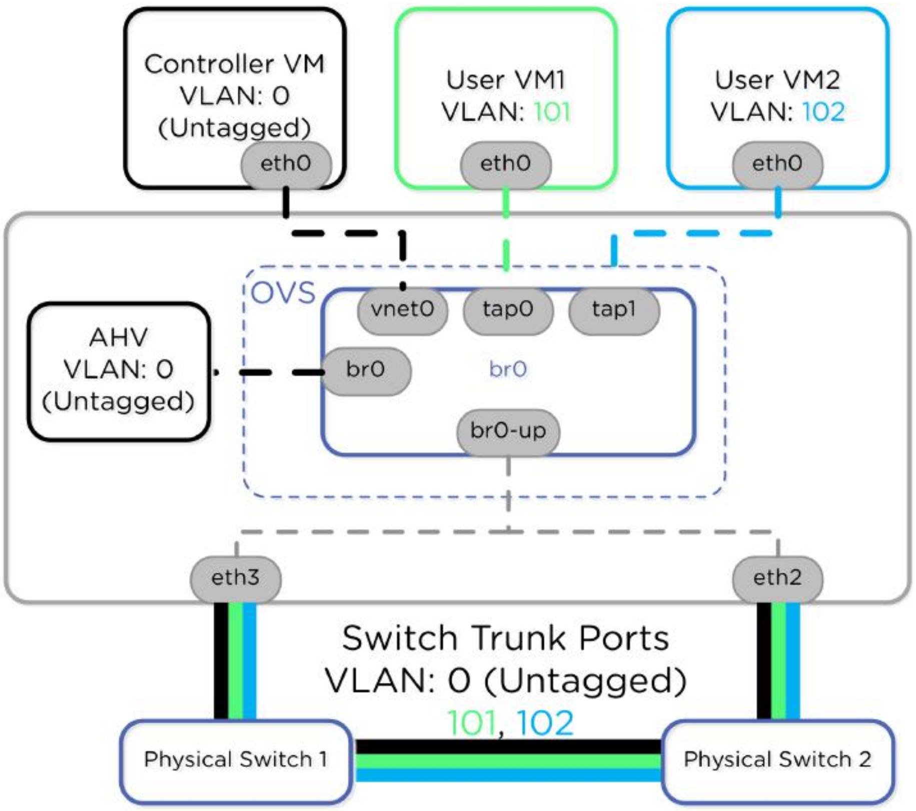


Figure 9: Default Untagged VLAN for CVM and AHV Host

If you do not want to send untagged traffic to the AHV and CVM, or the security policy doesn't allow this, please refer to the AHV Networking guide.

Jumbo Frames

Because of the additional network configuration required, Nutanix does not recommend configuring jumbo frames on AHV for most deployments. However, some scenarios, such as database workloads using Nutanix Volumes with iSCSI network connections, can benefit from the larger frame sizes and reduced number of frames. To enable jumbo frames, refer to the AHV Networking best practices guide. If you choose to use jumbo frames, be sure to enable them end to end in the desired network and consider both the physical and virtual network infrastructure impacted by the change.

4.2. Virtual Machine High Availability

Virtual machine high availability (VMHA) ensures that VMs restart on another AHV host in the cluster if a host fails. VMHA considers RAM when calculating available resources throughout the cluster for starting VMs.

VMHA respects affinity and antiaffinity rules. For example, with VM-host affinity rules, VMHA does not start a VM pinned to AHV host 1 and host 2 on another host when those two are down unless the affinity rule specifies an alternate host.

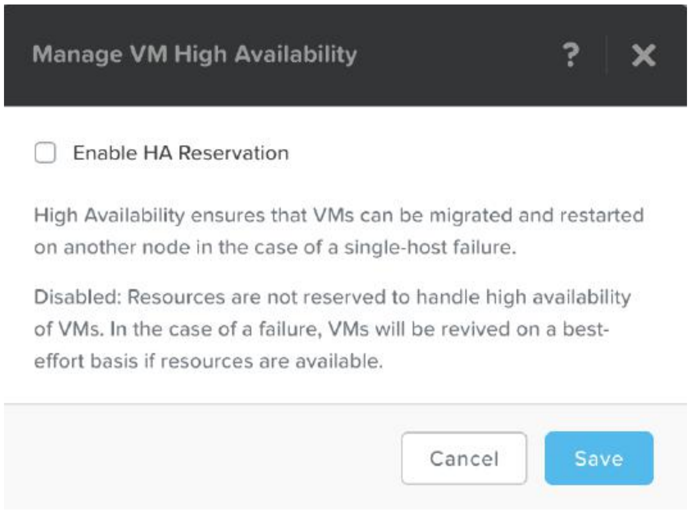
There are two VM high availability modes:

- Default

  This mode requires no configuration and is included by default when installing an AHV-based Nutanix cluster. When an AHV host becomes unavailable, the VMs that were running on the failed AHV host restart on the remaining hosts, depending on the available resources. If the remaining hosts do not have sufficient resources, some of the failed VMs may not restart.

- Guarantee

  This nondefault configuration reserves space throughout the AHV hosts in the cluster to guarantee that all VMs can restart on other hosts in the AHV cluster during a host failure. To enable Guarantee mode, select the **Enable HA** check box, as shown in the figure below. A message then displays the amount of memory reserved and how many AHV host failures the system can tolerate.


Manage VM High Availability

Enable HA Reservation

High Availability ensures that VMs can be migrated and restarted on another node in the case of a single-host failure.

Disabled: Resources are not reserved to handle high availability of VMs. In the case of a failure, VMs will be revived on a best-effort basis if resources are available.

Figure 10: VM High Availability Reservation Configuration

The VMHA configuration reserves resources to protect against:

- One AHV host failure, if all Nutanix containers are configured with a replication factor of 2.
- Two AHV host failures, if any Nutanix container is configured with a replication factor of 3.

Admins can use the aCLI to manage protection against two AHV host failures when using replication factor 2. The command for designating the maximum number of tolerable AHV host failures is:

```
nutanix@CVM$ acli ha.update num_host_failures_to_tolerate=X
```

When an unavailable AHV host comes back online after a VMHA event, VMs that were previously running there migrate back to their original host to maintain data locality.

To disable VMHA per VM, set a negative value (-1) when creating or updating the VM. This configuration removes the VM from the VMHA resource calculation.

```
nutanix@CVM$ acli vm.update <VM Name> ha_priority=-1
nutanix@CVM$ acli vm.create <VM Name> ha_priority=-1
```

This configuration does not start the VM on a new AHV host when the host where the VM was running fails. The VM starts again when the failed AHV host comes back online.

VMHA Recommendations

- Use the nondefault VMHA Guarantee mode when you need to ensure that all VMs can restart in case of an AHV host failure.
- When using Guarantee mode, keep the default reservation type of kAcropolisHAReserveSegments; this setting must not be altered.

Note: The VMHA reservation type kAcropolisHAReserveHosts is deprecated. Never change the VMHA reservation type to kAcropolisHAReserveHosts.

- Consider storage availability requirements when using VMHA Guarantee mode. The parameter num_host_failures_to_tolerate should not be higher than the configured storage availability. If there are only two copies of the VM data, the VM data could be unavailable if two hosts are down at the same time, even though the CPU and RAM resources to run the VMs are available.
- Disable VMHA for VMs pinned to one AHV host via VM-host affinity, an Acropolis Dynamic Scheduler (ADS) feature. This step must be completed before a VM can be pinned to one AHV host via ADS. This is not a recommended configuration, as we discuss in the following section.

4.3. Acropolis Dynamic Scheduler

The Acropolis Dynamic Scheduler (ADS) ensures that compute (CPU and RAM) and storage resources are available for VMs and volume groups (VGs) in the Nutanix cluster. ADS, enabled by default, uses real-time statistics to determine:

- Initial placement of VMs and VGs, specifically which AHV host runs a particular VM at power-on or a particular VG after creation.
- Required runtime optimizations, including moving particular VMs and VGs to other AHV hosts to give all workloads the best possible access to resources.

Affinity Policies

Affinity policies are rules that you define one of two ways—manually, by a Nutanix Enterprise Cloud administrator, or via a VM provisioning workflow. The two affinity policies available are:

- VM-host affinity

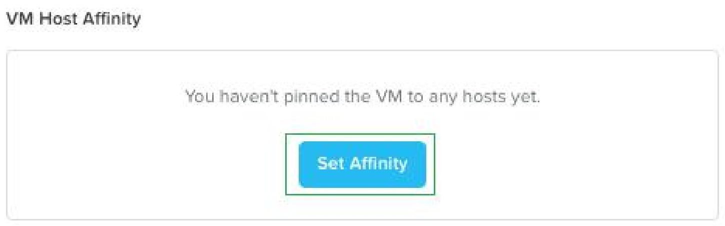
  This configuration keeps a VM on a specific set of AHV hosts. It is useful when VMs must be limited to a subset of available AHV hosts due to application licensing, AHV host resources (such as available CPU cores or CPU GHz speed), available RAM or RAM speed, or local SSD capacity. Host affinity is a "must" rule; AHV always honors the specified rule.

- VM-VM antiaffinity

  This policy ensures that two or more VMs do not run on the same AHV host. It is useful when an application provides high availability and an AHV host must not be an application's single point of failure. Antiaffinity is a "should" rule that is honored only when there are enough resources available to run VMs on separate hosts.

VM-host affinity is configured per VM during VM creation or via VM update operation using the aCLI or Prism.

- During VM creation or via VM update operation, click **Set Affinity** in the VM-Host Affinity section.


VM Host Affinity

You haven't pinned the VM to any hosts yet.

Figure 11: VM-Host Affinity Configuration Step One

- Select the AHV host where you want the VM to run.

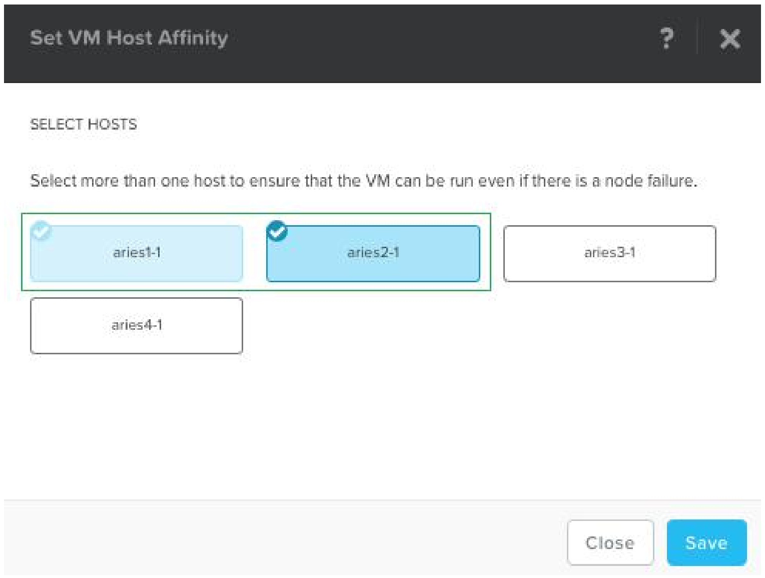

Set VM Host Affinity
?
SELECT HOSTS
Select more than one host to ensure that the VM can be run even if there is a node failure.
aries1-1
aries2-1
aries3-1
aries4-1
Close
Save

Figure 12: VM-Host Affinity Configuration Step Two

- The VM properties update, as shown below.

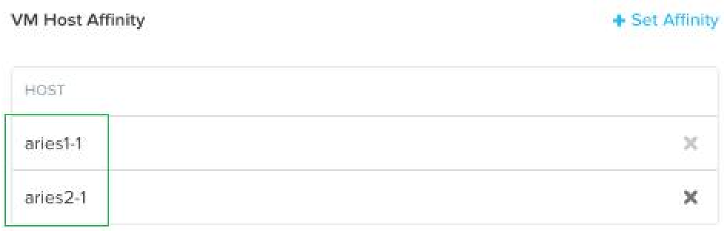


Figure 13: VM-Host Affinity Verification

Configure VM-VM antiaffinity using the following aCLI commands:

- Create VM antiaffinity group:

```
nutanix@CVM$ acli vm_group.create VM_AA_Group-Name
```

- Add VMs to the VM antiaffinity group:

```
nutanix@CVM$ acli vm_group.add_vms VM_AA_Group-Name vm_list=VM1,VM2
```

- Enable the VM antiaffinity group:

```
nutanix@CVM$ acli vm_group.antiaffinity_set VM_AA_Group-Name
```

- Disable the VM antiaffinity group:

```
nutanix@CVM$ acli vm_group.antiaffinity_unset VM_AA_Group-Name
```

- List VMs in a VM-VM antiaffinity group:

```
nutanix@CVM$ acli vm_group.list_vms VM_AA_Group-Name
```

- List VM antiaffinity groups:

```
nutanix@CVM$ acli vm_group.list
```

ADS does not migrate VMs with CPU passthrough enabled, VMs with GPU or vGPU enabled, or VMs classified as agent VMs, which are typically appliances providing a specific feature such as network or backup and restore capabilities. All these VM types bind to a specific AHV host. They power off during a host upgrade and automatically power on when the AHV host boots up.

Although it is possible to pin a VM to a specific AHV host using the VM-host affinity rule, we do not recommend doing so; this configuration introduces additional operational procedures and potential problems, including:

- The AHV upgrade process could fail and require manual actions.
- If you pin the VM when VMHA is configured for default mode, the VMHA Guarantee mode cannot be set unless you:
    - Disable VMHA for the VMs pinned to one AHV host.
    - Remove the VM-host affinity rule.
    - Add another AHV host to the VM-host affinity rule.

Use the following procedure to ensure that a VM runs exclusively on one specific AHV host using VM-host affinity rules:

- Disable VMHA for the VM using the following command:

```
nutanix@CVM$ acli vm.update <VMname> ha_priority=-1
```

- Configure VM-host affinity to one AHV host.

Note: This configuration does not start the VM on another AHV host when the original host fails. However, when the failed host comes back online, the VM starts again.

Verify the ADS status using:

```
nutanix@CVM$ acli ads.get
```

The return code for ads.get is “enabled: True” for enabled and “enabled: false” for disabled.

Admins can disable ADS rebalancing cluster-wide using the aCLI, but not on a per-VM basis. ADS affinity policies cannot be disabled.

Note: When ADS is disabled, automatic hotspot remediation and antiaffinity violation correction does not occur.

The command to disable ADS is:

```
nutanix@CVM$ acli ads.update enable=false
```

ADS Recommendations

- To ensure VMHA functionality on a VM, use VM-host affinity to pin the VM to a minimum of two AHV hosts if VMHA tolerates one AHV host failure, or to a minimum of three AHV hosts if VMHA tolerates two AHV host failures.
- Do not use VM-host affinity rules to pin a VM to only one AHV host.
- To guarantee that all VMs in an ADS VM-VM antiaffinity group run on different AHV hosts, limit the maximum number of VMs in the group to the number of AHV hosts in the cluster. If VMs must not run on the same hosts during a VMHA event, the number of VMs per VM-VM antiaffinity group must equal the number of AHV hosts minus one or two, depending on the VMHA failure configuration value num_host_failures_to_tolerate. This configuration ensures that all VMs run on different AHV hosts at all times, even during failure scenarios.
- Disable ADS only if you require strict control and do not want to allow for automatic ADS rebalancing.

4.4. VM Deployment

There are five different options for creating VMs in AHV:

1. Create an empty VM and install the operating system and needed applications.
2. Create a VM from an existing VM using the clone feature.
3. Create a VM from an existing VM snapshot using the clone feature.
4. Import a VM from a VM disk image using the image service or network share.
5. Import a VM using cross-hypervisor disaster recovery or hypervisor conversion.

We cover the best practices for each of these VM creation scenarios below.

Software Driver Requirements

This table below summarizes the driver requirements for the different VM deployment options.

Table 2: VM Driver Requirements

AHV VM Operation	VirtIO Required	NGT Required	Additional Information
Create New VM: Windows	Y	N	VirtIO driver installation is required. The Nutanix Guest Tools (NGT) software is optional.

AHV VM Operation	VirtIO Required	NGT Required	Additional Information
Create New VM: Linux	Y	N	VirtIO support in the Linux kernel is required and enabled for all AHV-supported Linux distributions by default. The NGT software is optional.
External Import via Image Service or Copied to ADFS: Windows	Y	N	VirtIO is required in the image (for example, VMDK, qcow) uploaded to the image service or copied to ADFS. The NGT software is optional.
External Import via Image Service or Copied to ADFS: Linux	Y	N	VirtIO support in the Linux kernel is required for the image uploaded to image service or copied to ADFS. All AHV-supported Linux distributions have VirtIO enabled by default. The NGT software is optional.
Cross-Hypervisor DR or Conversion: Windows	Y	Y	VirtIO and NGT are required.
Cross-Hypervisor DR or Conversion: Linux	Y	Y	VirtIO support in the VM Linux kernel and NGT are required.

Create New VM

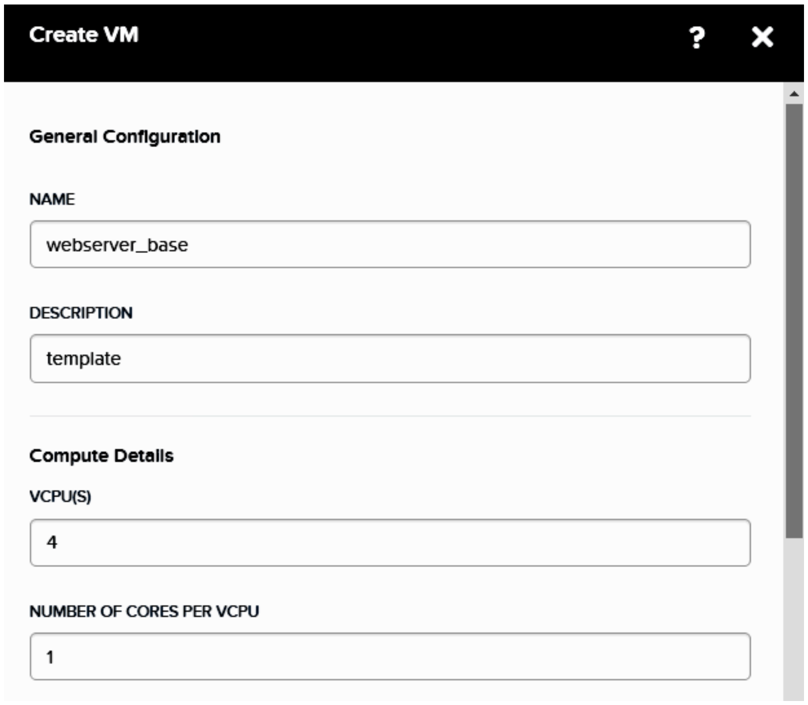
New VM with New OS Installation

When creating a new Windows VM on AHV, use the Nutanix VirtIO installation ISO to install the correct network and disk drivers in the guest VM. Windows VMs require these drivers. The Nutanix Support Portal maintains complete installation instructions. The NGT software is optional for newly installed Windows VMs; install NGT only if you want to use the features discussed in the NGT section below.

Ensure that new Linux guest VMs contain the virtio_scsi kernel driver when installed on AHV. Each of the supported Linux guest VMs listed on the Nutanix Support Portal contains these drivers. Without the virtio_scsi disk drivers, Linux guests can only use the IDE or PCI disk formats. As with Windows VMs, only install NGT if you need it.

New VM Cloned from Existing AHV VM

When cloning VMs, Nutanix recommends starting with an existing VM that is powered off. When cloning from powered-off VMs, the only limit is the amount of available resources. Select a name for the powered-off VM that is easy to find in a search, such as “base,” “original,” or “template.” For maximum performance when making more than 10 clones of a powered-on VM, use the snapshot method described in the next section instead of cloning from the VM directly.


Create VM

General Configuration

NAME: webserver_base

DESCRIPTION: template

Compute Details

VCPU(S): 4

NUMBER OF CORES PER VCPU: 1

Figure 14: Creating a VM Base for Clones

When you need optimal storage performance and minimal latency, Nutanix recommends either performing a new VM installation or cloning from a base that includes the operating system and applications, but no application data. Optimize the template VM to include only the features and applications that are required in the resulting cloned VMs before performing the clone operation. Use custom scripts to automate deployment tasks, such as network addressing and naming, with Sysprep for Windows and CloudInit for Linux environments.

For clones of Windows guest VMs, ensure that you have installed the Nutanix VirtIO drivers in the original VM before performing the clone operation.

To clone a VM from an existing VM in the Prism web console, select the desired VM and click **Clone**. To clone a VM from the aCLI use:

```
vm.clone <new_vm> clone_from_vm=<vm_name>
```

New VM Cloned from Existing AHV VM Snapshot

Cloning from a snapshot of an existing VM is similar to cloning from the VM itself—the base is simply from an earlier point in time. Administrators can create snapshots of VMs from the Prism web console or through the aCLI. Use a recognizable base name for the snapshot and follow the recommendations above for optimizing your base image.

To clone a VM from an existing snapshot in the Prism web console, navigate to the **VM Snapshots** tab of the desired VM and choose a snapshot. To clone from a snapshot using the aCLI, enter:

```
vm.clone <new_vm> clone_from_snapshot=<snapshot_name>
```

Nutanix recommends cloning from a VM snapshot instead of from a VM when cloning more than 10 VMs at a time, because cloning from a VM snapshot is more efficient. The following example creates 20 clones from a single snapshot, numbered 01 through 20.

```
vm.clone clone-vm-[01..20] clone_from_snapshot=base-vm
```

Import VM from Disk Image Using Image Service or Network Share

To create a VM from a disk image that has been imported via the image service or network share transfer, follow these rules before exporting the disk image from the source system:

- Windows VMs should have the Nutanix VirtIO package installed. You can install the optional NGT package, which contains VirtIO drivers, if you need NGT features.
- Linux VMs should have the NGT package installed. The NGT package ensures that the appropriate kernel drivers are enabled for the first boot into AHV using dracut.conf.

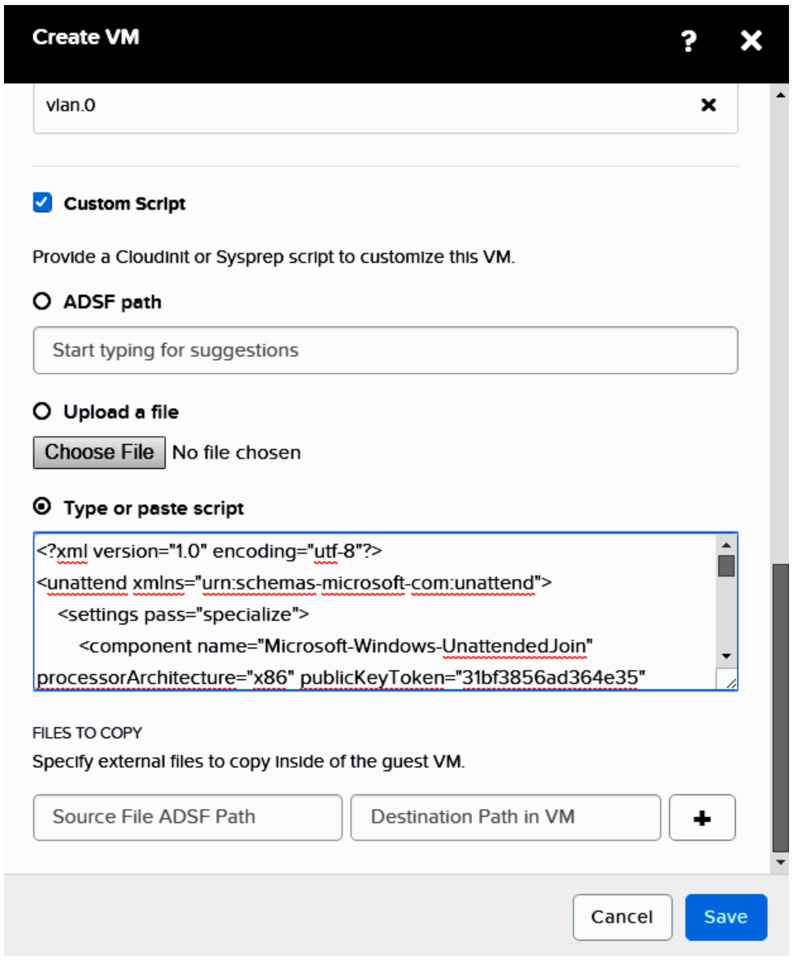
Note: Beginning with AOS 5.5, if Prism Central is deployed, you must use it for image management.

Import VM from Cross-Hypervisor Disaster Recovery or Hypervisor Conversion

Before importing Windows or Linux VMs from a non-AHV cluster using cross-hypervisor disaster recovery or hypervisor conversion, install the NGT package using the instructions from the Nutanix Support Portal. The NGT package provides support for cross-hypervisor disaster recovery and contains the VirtIO drivers that are required for the VM to function properly in AHV.

Guest Customization

It is also possible to customize the operating system deployment in conjunction with Sysprep for Windows environments or CloudInit for Linux environments. With Prism or aCLI custom scripts, you can perform functions such as adding a user, renaming a host, joining a domain, or injecting a driver. Specify the scripts by using an existing ADSF file, uploading a new file, or pasting the script directly into Prism (see the figure below). You can insert additional customization files into the VM during deployment using Prism or the REST API.



Provide a CloudInit or Sysprep script to customize this VM.

```
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
  <settings pass="specialize">
    <component name="Microsoft-Windows-UnattendedJoin"
processorArchitecture="x86" publicKeyToken="31bf3856ad364e35"
```

Specify external files to copy inside of the guest VM.

Figure 15: VM Deployment Customization

In Windows environments using Sysprep, admins can add customization when cloning a VM, creating a new VM from an existing file, or creating a new VM from installation media. Sysprep uses an XML-based answer file to manage customization. When cloning a VM from an existing file, specify an Unattend.xml file and format. When deploying a VM during a new installation, use the Autounattend.xml file and format.

In Linux environments using CloudInit, admins can customize when cloning a VM or when creating a new VM from an existing file. CloudInit does not support customization when creating VMs from installation media. Acropolis, with CloudInit, can support several script formats. Those formats are available in the Cloud-Init documentation.

4.5. Nutanix Guest Tools

Nutanix Guest Tools (NGT) is a set of software features installed within a VM and a Nutanix CVM. A Nutanix guest agent (NGA) publishes information about the VM to the Nutanix cluster, such as guest OS type, VM mobility status, and VSS services. The NGA is installed in both Windows and Linux guests.

- Nutanix Volume Shadow Copy Service (VSS) hardware provider

  The Nutanix VSS hardware provider enables integration with native Nutanix data protection. The provider allows for application-consistent, VSS-based snapshots when using Nutanix protection domains.

  Supported with Windows guests.

- Self-Service Restore

  The Self-Service Restore (SSR) feature enables Nutanix administrators or VM owners to mount snapshots directly to the VM from which they were taken. This capability allows end users to select the specific files they wish to use from a given snapshot.

  The Self-Service Restore GUI and CLI are installed in Windows guests. The Nutanix Guest Tools CLI (ngtcli) is used to execute the SSR CLI. Access the SSR GUI by opening a browser, navigating to http://localhost:5000, and logging on with machine administrator rights.

- Nutanix VirtIO Drivers

  NGT installs the Nutanix VirtIO drivers and provides the registry configuration necessary to support the migration or conversion of VMs bidirectionally between AHV and ESXi. These drivers are used when performing Nutanix VM Mobility or Nutanix cluster conversion.

  The required drivers are installed in Windows guests.

NGT requires network connectivity between the Nutanix cluster and the user VMs. NGT also requires an empty IDE CD-ROM slot in the guest for attaching the ISO image.

For NGT communication with the Nutanix cluster, ensure that the Nutanix cluster is configured with a virtual IP address (VIP). Also ensure that port 2074 is open between the required VMs and the CVMs.

NGT can be enabled from either Prism (VM page, table view) or the Nutanix CLI (nCLI) on a VM-by-VM basis. When first enabled, NGT mounts an ISO to the specified VM. An administrator can then run the NGT installer from inside the VM. After installation, the installer package ejects the CD-ROM and enables NGT for the VM.

VSS snapshot functionality is enabled by default, while self-service restore, previously called file-level restore, is disabled by default. To view or modify services, or control ISO mount state, use Prism by selecting the target VM and choosing **Manage Guest Tools**. nCLI can also be used, specifically the **ngt** command. The following example shows how to query the NGT state for a specific VM.

```
nutanix@CVM$ ncli

ncli> ngt get vm-id=89bca6ab-e995-4de7-b094-a7c9d6f62b59

VM Id                      : 000524ac-81af-5cb8-0000-000000005240::89bca6ab-e995-4de7-b094-
a7c9d6f62b59

VM Name                    : VMCC

NGT Enabled                : true

Tools ISO Mounted          : false

Vss Snapshot               : true

File Level Restore         : false

Communication Link Active : true
```

Nutanix Guest Tools Recommendation

Only install NGT within guests as required for the following use cases:

- For VMs that require self-service restore.
- For VMs that require VSS.
- Prior to performing a cluster conversion.
- When utilizing Nutanix VM Mobility.

If none of these conditions apply, use the Nutanix VirtIO driver standalone installation package and not NGT.

4.6. VM Data Protection

There are two built-in methods available in AHV for quickly bringing back VM data:

- On-demand VM instant recovery option.
- Protection domain VM instant recovery option.

Both options use the Nutanix cluster built-in snapshot functionality.

On-demand VM instant recovery offers a way to create local snapshots or checkpoints on demand local to the Nutanix cluster.

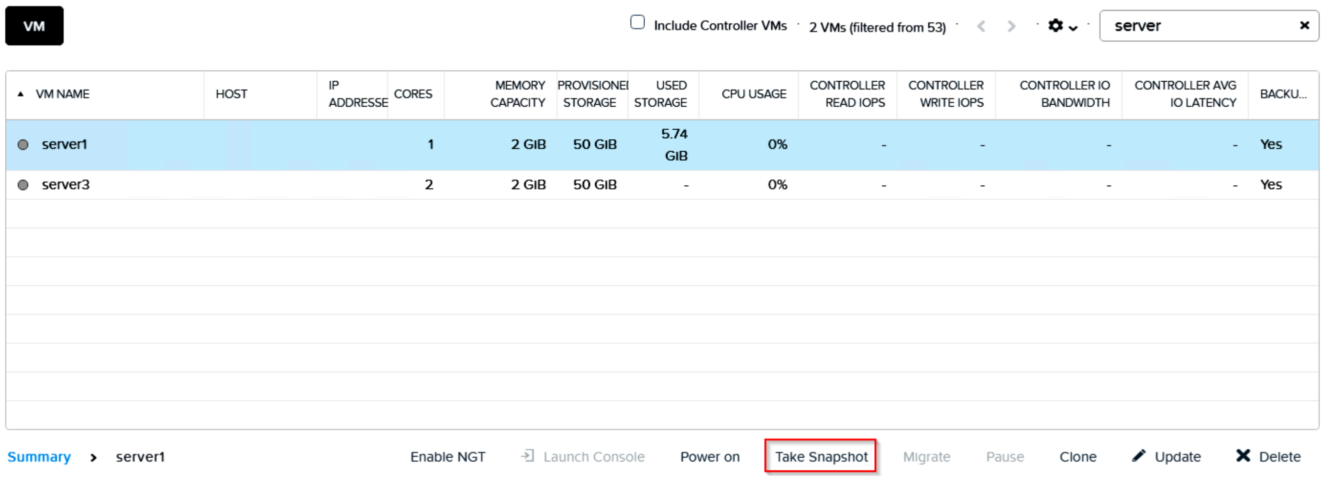
A protection domain is an administrator-defined set of VMs (or VGs) scheduled for local Nutanix cluster instant recovery. There is an option to provide a true backup and restore solution with the protection domain option, which requires the local Nutanix snapshots to be sent to a remote site. A VM can be part of only one protection domain and this protection domain can be replicated to one or more remote sites. Because a protection domain can be replicated to remote Nutanix clusters, its name must be unique across the system.

Both options provide a way to return rapidly to an exact time and state for both VMs and Nutanix VGs.

Snapshots are currently available for restore only via the same method used for snapshot creation. In addition to the built-in Nutanix features, admins can use third-party backup and restore solutions that include options to back up VMs either via AHV or via a VM-installed client to export VM data from the Nutanix cluster.

VM Snapshots: On-Demand VM Data Protection

The on-demand option is in the VM section in Prism or via the aCLI.



VM

Include Controller VMs · 2 VMs (filtered from 53) · server

VM NAME	HOST	IP ADDRESSE	CORES	MEMORY CAPACITY	PROVISIONEI STORAGE	USED STORAGE	CPU USAGE	CONTROLLER READ IOPS	CONTROLLER WRITE IOPS	CONTROLLER IO BANDWIDTH	CONTROLLER AVG IO LATENCY	BACKU...
server1			1	2 GIB	50 GIB	5.74 GIB	0%	-	-	-	-	Yes
server3			2	2 GIB	50 GIB	-	0%	-	-	-	-	Yes

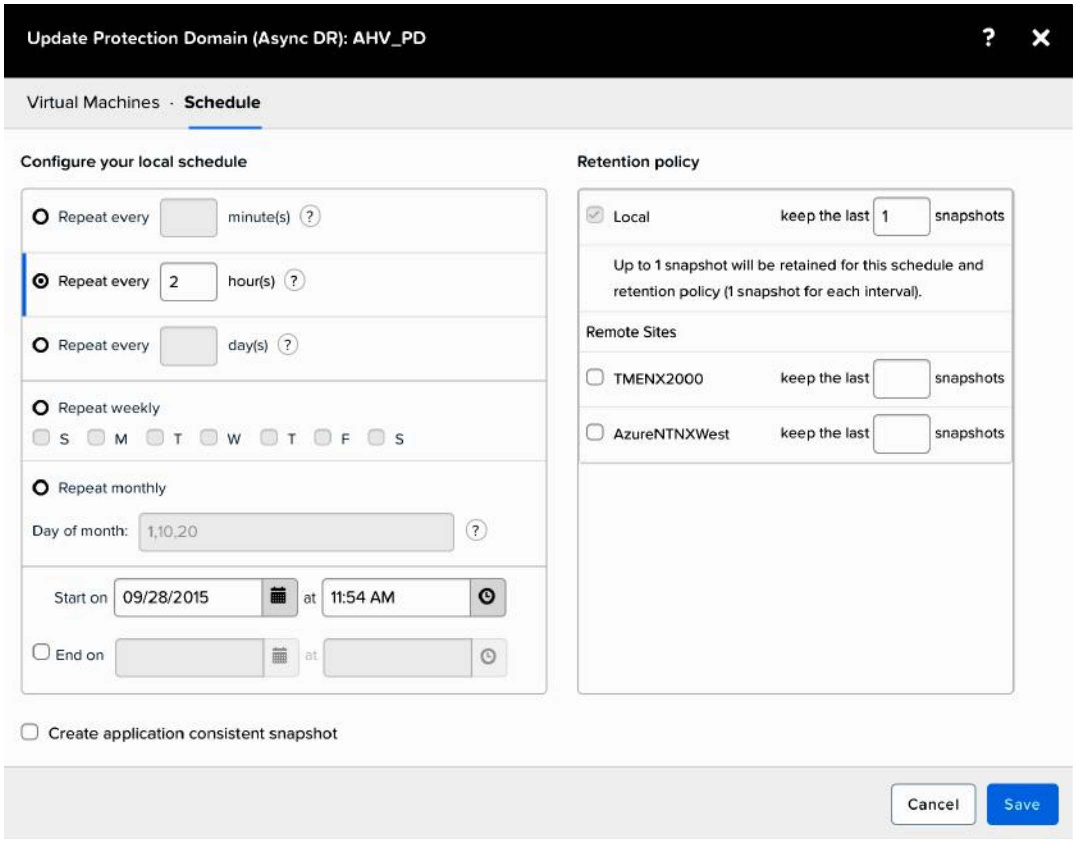
Summary > server1 · Enable NGT · Launch Console · Power on · Take Snapshot · Migrate · Pause · Clone · Update · Delete

Figure 16: On-Demand VM Snapshots

This option lets users take VM snapshots, restore snapshots, and create and clone new VMs. This process is helpful when taking an on-demand snapshot before starting potentially sensitive administrative tasks.

Protection Domain VM Snapshots: Scheduled VM Data Protection

The protection domain option allows you to create VM or volume group snapshots on a schedule and can be stored locally or in a remote cluster.


Update Protection Domain (Async DR): AHV_PD

Virtual Machines · **Schedule**

Configure your local schedule

- Repeat every [ ] minute(s)
- [✓] Repeat every 2 hour(s)
- Repeat every [ ] day(s)
- Repeat weekly: S M T W T F S
- Repeat monthly — Day of month: 1,10,20

Start on 09/28/2015 at 11:54 AM

End on

Create application consistent snapshot

Retention policy

Local	keep the last	1	snapshots

Up to 1 snapshot will be retained for this schedule and retention policy (1 snapshot for each interval).

Remote Sites

TMENX2000	keep the last		snapshots
AzureNTNXWest	keep the last		snapshots

Figure 17: Protection Domain Scheduled VM and Volume Groups Snapshots

Local Data Protection

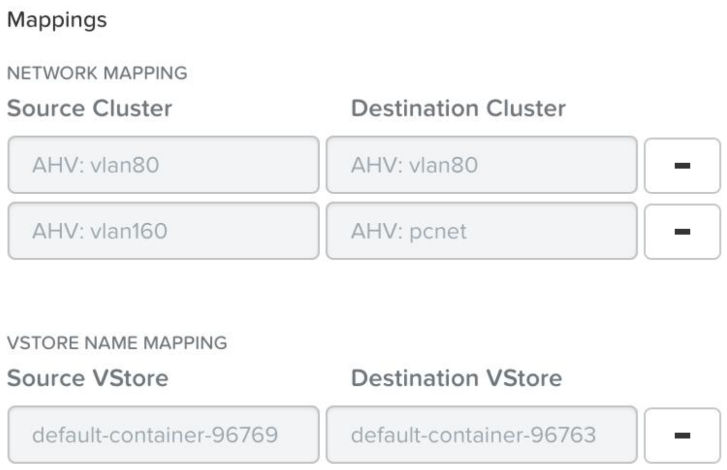
The protection domain asynchronous disaster recovery option includes the ability to schedule VM and volume group snapshots in the Nutanix cluster where the VM runs.

Remote Data Protection

Protection domain remote sites let you send VM snapshots to a remote site, such as a separate Nutanix cluster or a public cloud target, including AWS and Azure.

When network-naming conventions at the local and remote sites differ, Acropolis allows you to change the VM network name and VLAN ID so that your local VM can restart on the remote site in a valid network. For example, a VM running on AHV network VLAN-X at the local site can be configured to start on network VLAN-Y at the remote site. For IPAM-enabled networks, remote sites do not share DHCP reservations, so the administrator should plan for VMs to receive new addresses.

The mapping section includes both vstore or container names and network names to make sure that the system replicates VMs to the correct containers on the remote Nutanix cluster.


Mappings

NETWORK MAPPING

Source Cluster	Destination Cluster
AHV: vlan80	AHV: vlan80
AHV: vlan160	AHV: pcnet

VSTORE NAME MAPPING

Source VStore	Destination VStore
default-container-96769	default-container-96763

Figure 18: Protection Domain Remote Site Mappings

VM Data Protection Recommendations

- Snapshot or on-demand data protection for day-to-day checkpointing and snapshots (such as before making VM changes). This snapshot restore operation is simpler and faster than the scheduled snapshot option.
- Protection domain or scheduled data protection for local snapshots that need to run on a schedule or for offsite protection.
- When replicating data to a remote site, use the following naming convention for easier protection domain identification:
    - <Local-Site>_<Remote-Site>_PD# (or function)—for example, SiteB_SiteA_PD1.

4.7. Hypervisor Mobility and Conversion

The Acropolis operating system (AOS) includes core capabilities that enhance VM mobility and cluster conversion between different hypervisors—specifically, the bidirectional movement of VMs between Nutanix clusters running AHV and ESXi, called Nutanix VM Mobility, or the conversion of a Nutanix cluster from ESXi to AHV or from AHV to ESXi.

Nutanix VM Mobility

Nutanix VM Mobility provides a simplified means for replicating Nutanix-based VMs between different hypervisors. VM Mobility provides support for replicating Windows and Linux-based VMs bidirectionally between a Nutanix ESXi cluster and a Nutanix AHV cluster.

Nutanix VM Mobility uses native Nutanix snapshots to replicate data between clusters. Nutanix provides NGT to install the appropriate drivers and communicate with the cluster to confirm mobility support. The mobility process is very straightforward and includes the following high-level steps:

1. Enabling and installing NGT for the VMs to be replicated.
2. Creating a remote site in each Nutanix cluster. This step includes network mapping between the AHV and ESXi networks.
3. Creating an asynchronous protection domain and schedules to replicate the required VMs.
4. Moving or creating VMs using one of several workflow options:
    a. For planned mobility, all VMs within a protection domain are moved using the “migrate” option. This unregisters the VMs from the source cluster, replicates any incremental changes since the last snapshot, and registers the VMs in the target cluster.
    b. For unplanned mobility, where a source cluster is offline, the “activate” option moves all VMs within a protection domain. Activating registers the VMs within a protection domain in the target cluster using the most recently available local snapshot.
    c. The snapshot “restore” option clones individual VMs. A snapshot restore to a new location operates as a clone and registers the VM in the target cluster. This enables test and development scenarios and allows for targeting individual VMs within a protection domain.

Requirements and limitations:

- Install NGT so the appropriate drivers are available within the VM. This one-time operation allows for communication between the CVM and Nutanix cluster to validate driver installation.
- ESXi delta disks are not supported.
- VMs with SATA or PCI devices are not supported.

Prism sends an alert if specific VMs cannot be converted because of delta disks or virtual hardware. These “Recovery Details” are under the VM Recovery column of a given local or remote snapshot. Also, because Nutanix VirtIO drivers must be installed in a VM targeted for conversion, Prism gives a warning if the installation status is unknown.

Nutanix Cluster Conversion

A Nutanix cluster can be converted from ESXi to AHV. A cluster can also be converted from AHV to ESXi if it was previously converted from ESXi. Existing data, hypervisor networking settings, and VM settings are preserved during the cluster conversion process.

Requirements and limitations:

- As with Nutanix VM Mobility, install and enable NGT to ensure that the proper drivers are available.
- ESXi HA and DRS are enabled.

- An ESXi cluster supports only one external virtual switch.
- All uplinks must be homogenous (that is, have the same adapter speed—all 10 Gb or all 1 Gb).
- LACP-based virtual switch load balancing is not supported.
- VMs with delta disks cannot be converted from ESXi to AHV.

The general conversion process is as follows:

1. Administrator selects **Convert Cluster** from Prism Element.
2. Administrator selects the target hypervisor and VM boot options.
    a. You can choose to keep the original power state of the VMs after conversion, or power off VMs before conversion.
3. Cluster validation determines if requirements are met or if limitations exist.
4. If there are no blocking limitations, the conversion process may proceed. Prism displays warnings when applicable, including:
    a. If an existing active-active network team would become active-passive on conversion.
    b. If NGT are not enabled for specific VMs, preventing confirmation of Nutanix VirtIO driver installation.

Once the conversion begins, the following high-level steps occur:

1. The Acropolis conversion process collects and saves hypervisor information. The cluster remains manageable during the conversion process.
2. User VMs live migrate from the node targeted for conversion to other nodes in the cluster.
3. Acropolis converts the node evacuated in the previous step to the targeted hypervisor.
4. The process restores user VMs to the newly converted node one at a time, then also converts each VM. Running VMs experience downtime similar to the duration of one power-off and one power-on cycle.
5. Once the targeted node and all original VMs have been converted, the process moves onto the next node.
6. When all nodes are using the targeted hypervisor, the conversion is complete.

Hypervisor Mobility and Conversion Recommendations

- Use Nutanix VM Mobility for simplified migration between Nutanix clusters using different hypervisors.
- Use snapshot restores to new locations when using VM Mobility to create VMs without affecting protection domain replication schedules.
- Use Nutanix cluster conversion to utilize the same hardware while converting to a different hypervisor.

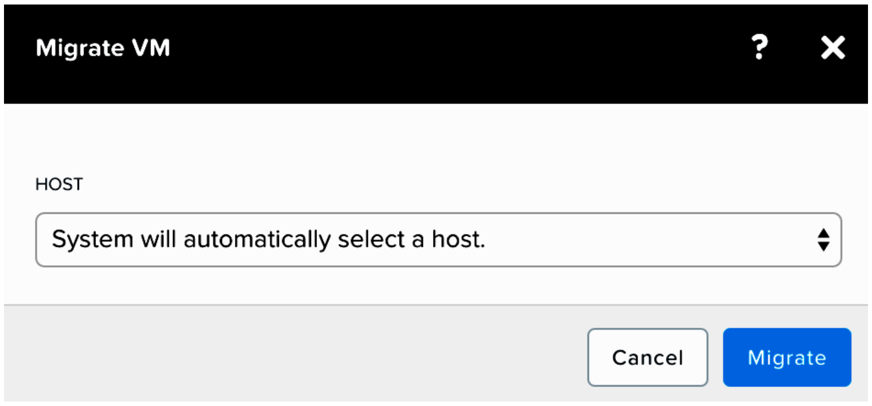
- Utilize DHCP where possible to simplify network connectivity for converted VMs.
- See the Migrating VMs to Nutanix AHV tech note and the Prism Web Console Guide for additional details.

4.8. Live Migration

Live migration lets admins move a user VM from one AHV host to another while the VM is powered on. Live migration can be initiated as long as the target AHV host has available resources. Acropolis selects a target host automatically, but you can specify a target if required.

Start live migration with any of the following methods:

- Put the AHV host in maintenance mode (VM evacuation).
- PRISM UI (automatic or targeted).



System will automatically select a host.

Figure 19: Prism UI Automatic Live Migration

- aCLI (automatic, targeted, or maintenance mode).
- REST API (automatic, targeted, or maintenance mode).

CPU types are abstracted from the VMs because AHV uses virtual CPUs that are compatible between all nodes in the AHV cluster. As long as enough CPU cycles are available in the destination AHV host, you can migrate a VM.

By default, live migration uses as much available bandwidth as required over the AHV host management interface, br0 and br0-up. The bandwidth allocated for migration can be restricted via the aCLI and the REST API using bandwidth_mbps=X during each migration.

The following aCLI example enforces a 100 Mbps limit when migrating a VM, slow-lane-VM1, to AHV host 10.10.10.11:

```
nutanix@CVM$ acli vm.migrate slow-lane-VM1 bandwidth_mbps=100

host=10.10.10.11
```

To prevent resource contention, AHV limits the number of simultaneous automatic live migration events to two. Nutanix does not recommend any specific live migration method; use the live migration method suited to the current administrative task.

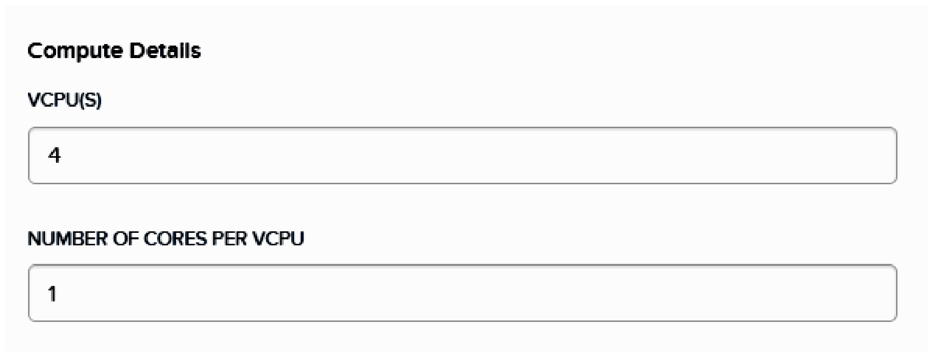
Live Migration Recommendations

Note that there is no limit to the number of simultaneous user-initiated live migrations, so be sure to consider the following to avoid compute and network resource contention:

- Do not exceed two simultaneously initiated live migrations that share the same destination or source hosts.
- Limit the total number of simultaneously initiated live migrations in an AHV cluster to a maximum of one per destination and source AHV host pair. When migrating VMs with lots of active memory, when cluster I/O is high, or in large AHV clusters, this number may be reduced depending on available network throughput. For smaller AHV clusters, it may be possible to run more than one live migration per destination and source AHV host pair, depending on available network capacity.

4.9. CPU Configuration

AHV lets you configure vCPUs (equivalent to CPU sockets) and cores per vCPU (equivalent to CPU cores) for each VM. Nutanix recommends first increasing the number of vCPUs when more than one CPU is needed in a VM, rather than increasing the cores per vCPU. If a user VM requires four CPUs, Nutanix recommends the configuration shown below.



Compute Details

VCPU(S): 4

NUMBER OF CORES PER VCPU: 1

Figure 20: VM vCPUs and Cores per vCPU Configuration

The following situations may require increasing cores per vCPU instead of vCPUs:

- VM guest operating system cannot use more than a limited number of vCPUs (sockets).
- VM application licensing is based upon number of vCPUs (sockets).

The maximum number of vCPUs and cores for a single VM is equal to the total number of hyperthreaded cores available in the AHV host.

It is possible to enable CPU passthrough for a VM using the aCLI. This means the VM would have access to all the CPU specifications provided by the physical CPU (pCPU), such as VT-x for nested virtualization. It is not possible to use live migration when using CPU passthrough.

CPU Recommendations

- Use vCPUs instead of cores to increase the number of vCPUs available for a VM.
  - Using vCPUs to add additional CPU power also makes sure that hot adding CPU works. It is not possible to hot add cores.
- Use only as many vCPUs as the VM requires to limit resource waste. If the application performance is not affected, it is preferable from an AHV resource scheduling and usage perspective to have two vCPUs running at 50 percent utilization each, rather than four vCPUs running at 25 percent utilization each. Two vCPUs are easier to schedule than four.
- Use the physical core count instead of the hyperthreaded count for maximum single VM sizing. Do not configure a single VM with more vCPU cores than pCPU cores available on the AHV host. This configuration can cause significant performance problems for the VM.

4.10. Memory

All physical AHV host memory can be used for VMs, aside from the memory used by the CVM and the AHV host itself. There is no AHV host memory overcommit.

Nonuniform Memory Access (NUMA) and User VMs

CPUs have memory controllers that are integrated on the processor socket. Nonuniform Memory Access (NUMA) is a computer memory design in which memory access times depend on the memory’s location relative to a processor. Simply put, because accessing nonlocal memory requires going through an interconnect, a system can access the memory local to a processor a lot faster than it can access nonlocal memory. As a result, the best practice is to create VMs that fit within the memory available to the CPU where the VM is running whenever possible.

An AHV host containing two CPU sockets with 10 cores each and 256 GB of memory has two NUMA nodes (one per physical socket), each of the following size:

- 1 socket
- 10 CPU cores (20 with hyperthreading)

- 128 GB memory

Some hosts have a different NUMA architecture than this per-socket setup; for example, a 12-core processor can have two NUMA nodes with 6 cores each. Conversely, it is also possible (but quite unusual) to have an architecture in which a NUMA node can span two physical CPU sockets. In most cases, an AHV host has one NUMA node per socket.

For optimal resource utilization, use both CPU and memory from the same NUMA node. Running CPU on NUMA node 0 and accessing memory from NUMA node 1 introduces memory latency. To take advantage of the lowest memory latency, create a virtual hardware topology for VMs that matches the physical hardware topology.

Use the following CVM command to determine the NUMA topology of all AHV hosts. Note that “cpus num” includes hyperthreaded CPU cores.

```
nutanix@cvm$ allssh "virsh capabilities | egrep 'cell id|cpus num|memory unit'"
Executing virsh capabilities | egrep 'cell id|cpus num|memory unit' on the cluster
================= 10.4.56.82 =================
        <cell id='0'>
          <memory unit='KiB'>134181480</memory>
          <cpus num='20'>
        <cell id='1'>
          <memory unit='KiB'>134217728</memory>
          <cpus num='20'>
```

We use the following VM definitions throughout this document:

- vNUMA or wide VMs

  These are VMs requiring more CPU or memory capacity than is available on a single NUMA node.

- vUMA or narrow VMs

  The capacity of a single NUMA node can fulfill the CPU and memory requirements of these VMs. Thus, you only need one NUMA node to provide these resources.

- Regular VMs

  The total capacity of the AHV host can fulfill the CPU and memory requirements of these VMs; thus, one or more NUMA nodes may have to provide the resources. All NUMA nodes can provide the compute resources, even if the VM’s requirements fit within a single NUMA node.

Virtual Nonuniform Memory Access (vNUMA) and User VMs

The primary purpose of vNUMA is to give large virtual machines (or wide VMs) the best possible performance. vNUMA helps the wide VMs create multiple vNUMA nodes. Each vNUMA node has virtual CPUs and virtual RAM. Pinning a vNUMA node to a physical NUMA node ensures that virtual CPUs accessing virtual memory can get the expected NUMA behavior. Low latency memory access expectation in virtual hardware (within vNUMA) can now match low latency in physical hardware (within physical NUMA), and high latency expectation in virtual hardware (cross vNUMA) can match high latency on physical hardware (cross physical NUMA).

In many situations that require vNUMA configuration, each AHV host is running only one wide VM. There are circumstances where multiple VM types (including vNUMA, vUMA, and regular VMs) must run on the same AHV host, but AHV is not optimized for such cases. Administrators can configure vNUMA for a VM via the aCLI or REST API; this configuration is VM-specific and defines the number of vNUMA nodes. Memory and compute are divided in equal parts across each vNUMA node.

```
nutanix@CVM
$ vm.create <VMname> num_vcpus=<X> num_cores_per_vcpu=<X> memory=<X>G num_vnuma_nodes=<X>
```

vNUMA and vUMA VM configurations require strict fit. With strict fit, for each VM virtual node (configured via **num_vnuma_nodes**), memory must fit inside a physical NUMA node. Each physical NUMA node can provide memory for any number of vNUMA nodes. If there is not enough memory within a NUMA node, the VM does not power on. Strict fit is not required for CPU.

The VM-to-AHV host configuration only implements strict fit from a memory perspective—not from a CPU perspective—as per the below examples, which use four physical NUMA nodes, each with one CPU socket, 10 CPU cores, and 128 GB of memory:

- vUMA configuration:
  - VM configured with 8 cores and 96 GB of memory fits within the NUMA node from a memory perspective, so the VM can be powered on.
  - VM configured with 8 cores and 256 GB of memory does not fit within the NUMA node from a memory perspective, so the VM cannot be powered on.
- vNUMA configuration:
  - VM configured with 16 cores, 192 GB of memory, and two vNUMA nodes fits in two NUMA nodes, so the VM can be powered on.
  - VM configured with 20 cores, 384 GB of memory, and two vNUMA nodes does not fit in two NUMA nodes, so the VM cannot be powered on.

If a VM's vUMA or all vNUMA nodes fit in an AHV host, the virtual resources are pinned to physical resources. If a VM does not fit any AHV host, the VM power-on action either succeeds or fails, depending on the initiator of the power-on action. Administrator-initiated VM power-on and VM live migration operations are always strictly pinned, which means that you cannot perform them unless the target AHV host's node can fit the vNUMA and vUMA configuration. vNUMA and vUMA VMs may be moved to an AHV host without sufficient memory in a single NUMA node under the following circumstances:

- AHV maintenance mode.
- Host HA failover or restore.

Any VM configured for vNUMA or vUMA is excluded from ADS.

Wide VMs

VMs with memory requirements that cannot be fulfilled by one NUMA node are wide VMs; with wide VMs, the administrator defines the number (and, implicitly, the size) of vNUMA nodes. The compute and memory for the CVM are pinned to a physical NUMA node, so when you determine the number of vNUMA nodes for a wide VM, you can choose from one of the following approaches:

- Ignore all compute resources used by the CVM and allow for CPU overcommit on the CVM's physical NUMA node.
- Set aside an AHV host core for each CVM vCPU; this reservation prevents CPU overcommit.
- Ignore some fraction of the CVM compute resources out of the available AHV host compute resources. Allow CPU overcommit to the extent of the ignored fraction.

If the vNUMA VM does not drive both compute and storage heavily at the same time, overcommit may not be a problem in practice depending on your workload, meaning that applications that are not affected by the CVM resource utilization can allow for CPU resource overcommit, while other applications see negative performance impact with any CPU overcommit.

vUMA VMs

If a VM's compute and memory requirements can fit into one NUMA node, an administrator can enable virtual Uniform Memory Access (vUMA), guaranteeing that the VM uses both CPU and memory from the same NUMA node.

vUMA VMs (also known as narrow VMs) deliver the lowest memory latency possible for VMs that fit within one NUMA node. Enabling vUMA uses the same command syntax used to enable vNUMA; the only difference is that the **num_vnuma_nodes** parameter is set to 1.

```
nutanix@CVM
$ vm.create <VMname> num_vcpus=<X> num_cores_per_vcpu=<X> memory=<X>G num_vnuma_nodes=1
```

You cannot use vUMA when the VM requires more memory than is available in a NUMA node based on strict fit requirements. Although it is technically possible to use vUMA if the VM requires more vCPU than there are cores available in the NUMA node, Nutanix strongly recommends that you do not take this approach.

Memory Recommendations

- If possible, keep regular VMs within the size of one AHV host NUMA node.
- Avoid running vNUMA, vUMA, and regular VMs on the same AHV host since migrations of other VMs can cause a vNUMA or vUMA VM to not power on.
- vNUMA:
  - Use a vNUMA configuration for large VMs requiring more CPU or memory than available in one physical NUMA node.
  - Follow application-specific **num_vnuma_nodes** configuration recommendations when provided by Nutanix.
  - If there is no application-specific **num_vnuma_nodes** configuration recommendation available, ignore the CVM compute resources and allow for CPU overcommit. Set **num_vnuma_nodes** equal to the number of AHV host NUMA nodes, and set the number of VM CPUs equal to the number of available AHV host pCPUs.
  - vNUMA VMs should run alone on the AHV host, separate from the CVM.
- vUMA:
  - Size vUMA VMs proportionally with respect to compute and memory. For example, if vRAM is half of physical NUMA RAM, then set the number of vCPUs to be half the pCPUs in the physical socket.
  - Do not use vUMA for VMs requiring more vCPU than there are available cores in a NUMA node.
  - Do not configure vUMA unless there is a solid use case for doing so, proven via Nutanix official recommendation or internal testing.
  - Use vUMA when predictable performance with the lowest memory latency possible is critical for the application.
- Use homogenous AHV hosts when taking advantage of vNUMA and vUMA features, or determine the NUMA boundaries of all Nutanix nodes where you are placing vNUMA- and vUMA-enabled VMs.
- Use VM-host affinity rules, with a minimum of two AHV hosts to cover for VMHA and maintenance activities, together with vNUMA to make sure that the VM does not move between nodes.
  - This aspect of the configuration is extremely important if there are heterogeneous AHV hosts in the cluster.

Controller VM Memory Sizing

The CVM memory requirements depend on the features that the cluster uses, specifically deduplication. If performance-tier deduplication is enabled, the CVM memory should be at least 24 GB. For capacity-tier deduplication, configure at least 32 GB of memory on each CVM. Use the steps detailed on the Nutanix Support Portal to modify CVM memory on all AHV hosts.

4.11. Hotplug CPU and Memory

It is possible to increase the compute (both CPU and memory) capacity of a running VM. The hotplug feature allows you to add vCPUs (sockets) to increase the CPU capacity. You cannot increase cores per vCPU while a VM is running. Because no memory overcommit is allowed, you can increase VM memory capacity within the limit of the available AHV host's physical RAM.

Hotplug Recommendations

Size VMs correctly in terms of compute resources when they are created or when they are powered off.

If you must increase compute resources while the VM is running, be sure to consider the following:

- CPU: The multiqueue virtio-scsi PCI storage controller presented to the VM uses one request queue per vCPU. Because adding a vCPU to a running VM does not create new queues, the addition could limit performance.
- Memory: The number of memory hotplug operations is limited to two or three, depending on the VM's configuration when it was powered on. This limitation is due to the number of memory regions available for a VM. This counter resets on VM power cycles (VM off and VM on).

4.12. AHV Turbo Technology

To maximize storage performance, Nutanix has developed AHV Turbo technology, which benefits:

- The VMs, by presenting a multiqueue virtio-scsi PCI controller.
- The hypervisor, by processing different queues in parallel using multiple threads.
- The interface between the hypervisor and the storage layer, by increasing efficiency when receiving the requests from the hypervisor and passing them to the CVM.

The figure below explains the logical implementation of Turbo technology.

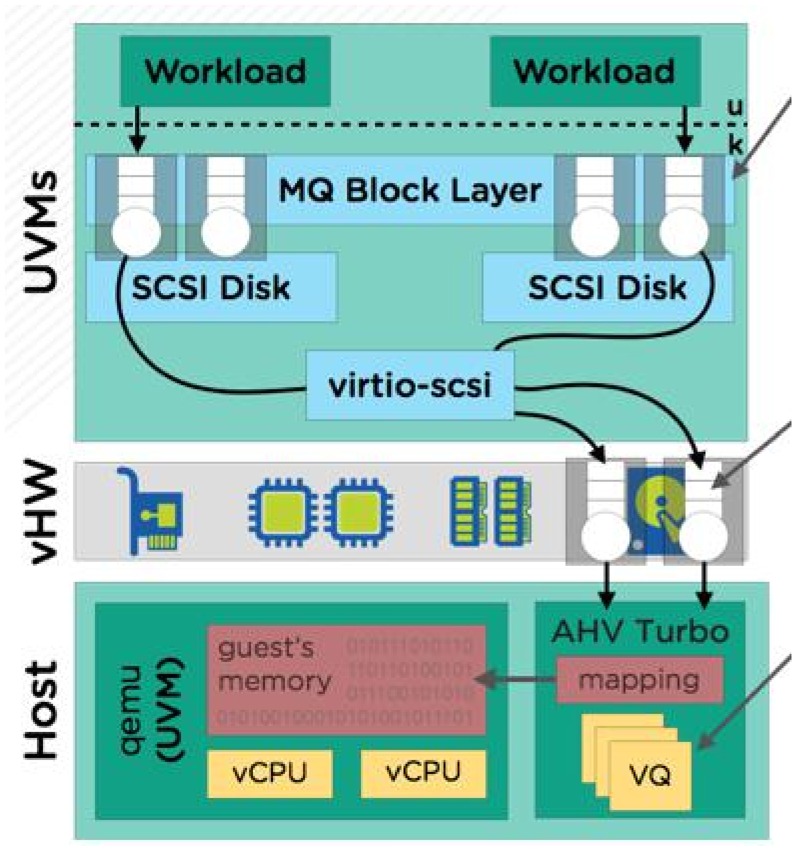


Figure 21: AHV Turbo Technology Logical Implementation

AHV Turbo Technology Recommendations

AHV Turbo technology is transparent to the VMs, but you can achieve even greater performance if:

- The VM has multiqueue enabled. Consult your Linux distribution documentation to make sure that the guest operating system fully supports multiqueue before you enable it, as well as for instructions on how to enable it. One common method for enabling multiqueue is to add one of the following lines to the kernel command line:

```
"scsi_mod.use_blk_mq=y"
```

```
"scsi_mod.use_blk_mq=1"
```

- You have installed virtIO for Windows-based VMs. No additional configuration is required.
- The VM has more than one vCPU.
- The workloads are multithreaded.

4.13. Disks

AHV-based VMs can use SCSI, PCI, SATA, or IDE disks; SCSI is the default disk type when creating AHV-based VMs. You can determine the appropriate number of disks for a given VM by referring to Nutanix reference architectures, best practices guides, and technical notes, as well as by consulting vendor recommendations. The maximum number of SCSI disks attached to a VM is 256, and the maximum number of PCI disks attached to a VM is 7.

Thin Provisioning

Thin provisioning provides efficient storage space utilization by allowing the system to present storage devices without allocating space until data is written. Nutanix storage pools and containers enable thin provisioning by default as a core feature of the Acropolis DSF. vDisks created within AHV VMs or VGs are thinly provisioned.

Over time, as the system writes data, it allocates storage within the vDisks. At some point, data deleted from within a vDisk could remain allocated within the Nutanix storage pool. To maintain storage efficiency, when you delete a vDisk from an AHV VM or VG, the DSF reclaims the space consumed by the deleted file as a background process.

SCSI UNMAP

From the AOS 4.7 release onward, AHV VM vDisks and VGs support the SCSI UNMAP command as defined in the SCSI T10 specification. The SCSI UNMAP command allows a host application or operating system to specify that a given range of storage is no longer in use and can be reclaimed. This capability is useful when data is deleted from within a vDisk that AHV presents. Both the Windows and Linux operating systems provide native support for issuing UNMAP commands to storage.

When the guests send UNMAP commands to the Nutanix storage layer, Prism accurately reflects the amount of available storage as DSF background scans complete. If the guest does not send UNMAP commands, freed space is not made available for other guests and does not display as available in Prism.

Windows

Windows Server 2012 or later can issue industry-standard UNMAP commands to tell the Nutanix cluster to free storage associated with unused space. Windows supports reclaim operations against both NTFS and ReFS formatted volumes; these operations are enabled by default.

With Windows Server 2012 or later, UNMAP commands are issued under the following conditions:

- When you delete files from a file system, Windows automatically issues reclaim commands for the area of the file system that you freed.

- When you format a volume residing on a thin-provisioned drive with the quick option, Windows reclaims the entire size of the volume.
- When a regularly scheduled operation selects the Optimize option for a volume, or when you manually select this option, either from the Optimize Drives console or when using the optimize-volume PowerShell command with the Retrim option.

To check the current Windows configuration, which is a global setting for the host, you can use the **fsutil** command.

```
fsutil behavior query DisableDeleteNotify
DisableDeleteNotify=0    <---- enabled (default)
DisableDeleteNotify=1    <---- disabled
```

To disable the feature:

```
fsutil behavior set DisableDeleteNotify 1
```

To enable the feature:

```
fsutil behavior set DisableDeleteNotify 0
```

Nutanix recommends using the default Windows configuration, which allows the guest file system to report freed space back to the storage layer as files are deleted.

Windows Disk Formatting with UNMAP

Windows guests send UNMAP commands by default during disk format operations, which can increase the amount of time required to complete the format. Set DisableDeleteNotify to 1, temporarily disabling the feature during format to finish the format operation faster. Enable the feature once the format operation is completed.

Linux

Most Linux guests do not send UNMAP operations by default for mounted disks. Administrators can enable UNMAP operations in the Linux guest if they wish, either by mounting the disk with the discard option or by performing scheduled fstrim operations. Nutanix recommends following RHEL documentation and using a periodic fstrim operation instead of the discard mount option if you need guest free space reporting in Linux.

To check the current Linux configuration, which is a setting for each disk, you can view the value of **discard_zeroes_data**.

```
cat /sys/block/<device>/queue/discard_zeroes_data
0 => TRIM (UNMAP) disabled
1 => TRIM (UNMAP) enabled
```

Some Linux distributions, such as Ubuntu and SUSE, perform periodic fstrim operations for you in a weekly cron task. Nutanix recommends using the following configuration in Linux guests. If possible, randomize the start time among guests.

```
cat /etc/cron.weekly/fstrim
/sbin/fstrim --all || true
```

Linux Disk Formatting with UNMAP

Linux file systems such as Ext4 send UNMAP commands during the format operation, which can increase the amount of time required to finish formatting. To finish the format operation faster, use the nodiscard option.

```
mkfs.ext4 -E nodiscard /dev/sdXY
```

Disk Recommendations

- Nutanix recommends using SCSI-based disks wherever possible for maximum performance.
- Use the default Windows settings for UNMAP.
- Configure Linux guest VMs with a weekly scheduled fstrim if desired.
- Use the nodiscard option when formatting Linux disks for maximum format speed.
- Use SATA or PCI-based disks for older Linux VMs (such as RHEL5) that don't include the SCSI paravirtual drivers used by AHV.
- Use the IDE disk type for CD-ROM tasks such as booting a VM and installing the paravirtual drivers from an ISO. Due to the lower performance of the IDE bus, use the other bus types for applications and drives that require high storage performance.

4.14. Resource Oversubscription

You must take into account a slight memory overhead per VM in the AHV host when calculating available memory for VMs. This overhead is used for tasks such as management, the emulation layer, and I/O buffers. You can use the following overhead values when calculating memory requirements:

```
2.6 GB per AHV host
60 MB per VM
```

CPU Oversubscription

Virtualization allows the number of vCPU cores in the VM to be decoupled from the number of pCPU cores in the hypervisor. Nutanix recommends following industry-standard vCPU-to-pCPU oversubscription ratios to provide the best possible performance.

The following examples of vCPU-to-pCPU oversubscription ratios are general guidance; adjust them as appropriate for individual deployments. Latency-sensitive server virtualization applications have lower ratio requirements than applications like VDI end-user computing that can tolerate higher vCPU-to-pCPU oversubscription.

Table 3: CPU Oversubscription Ratios

pCPU to vCPU Ratio	Application Class
1:1 or 2:1	Tier 1: Business-critical or latency-sensitive applications
2:1 or 4:1	Tier 2: Noncritical applications, not highly latency sensitive

During VM boot, the DSF enforces a maximum 20:1 oversubscription ratio to ensure that adequate resources are available to running VMs. During VM steady-state operation, the DSF allows oversubscription based on available host CPU cycles.

Memory Oversubscription

It is not possible to oversubscribe memory in AHV with active VMs. You can create and configure VMs to oversubscribe memory, but you cannot power them on unless memory is available.

Resource Oversubscription Recommendations

Use the figures for CPU oversubscription ratios provided in the table above as a starting point for determining CPU oversubscription. The most critical inputs for determining oversubscription are the environment's business and technical requirements.

4.15. Hugepages

x86-based CPUs address memory in 4 KB pages by default, but they can use larger, 2 MB pages, which are called "large pages" or "hugepages." AHV uses these hugepages to provide the best possible performance.

Certain applications running specific types of workloads see performance gains when using hugepages at the VM level. However, if the VM is configured to use hugepages, but the application and VM guest operating system do not fully utilize them, memory fragmentation and wasted memory may occur. Some database applications, such as MongoDB, recommend against using Transparent Huge Pages (THP; a method to abstract the use of hugepages) in the guest VM.

Hugepages Recommendations

- Enable hugepages within the guest VMs only when the application and VM operating system require them.
- Do not turn off hugepages in the AHV host.
- Do not enable THP in the AHV host.

5. Conclusion

You now have the information you need to install and configure AHV, add it to the physical network, and create virtual networks—all in a few hours rather than a few weeks. AHV eliminates the virtualization tax that comes from additional licensing fees, bundled shelfware, and unnecessary complexity. Tightly integrated with the Nutanix Enterprise Cloud OS, AHV takes full advantage of features like compression, deduplication, and erasure coding, as well as high availability and live migration. And, unlike other hypervisors, AHV is front-ended by the highly available and easy-to-use Prism management interface. AHV is one more way that Nutanix lets you focus on your applications instead of your infrastructure.

For more information about Acropolis or to review other Nutanix reference architectures, best practices guides, solution notes, and tech notes, please visit the Nutanix website.

Appendix

AHV Networking Best Practices Checklist

- VLANs
  - # Keep the CVM and AHV in the same VLAN. By default, the CVM and the hypervisor are assigned to VLAN 0, which effectively places them on the native, untagged VLAN configured on the upstream physical switch.
  - # Configure switch ports connected to AHV as VLAN trunk ports.
- Open vSwitch
  - # Do not modify the OpenFlow tables associated with the default OVS bridge br0; the AHV host, CVM, and IPAM rely on this bridge.
  - # While it is possible to set QoS policies and other network configuration on the VM tap interfaces manually (using the ovs-vsctl command), we do not recommend it. Policies are ephemeral and do not persist across VM power cycles or migrations between hosts.
- Virtual bridges
  - # Do not delete or rename OVS bridge br0.
  - # Do not modify the native Linux bridge virbr0.
- OVS bonded port (br0-up)
  - # Aggregate the 10 Gb interfaces on the physical host to an OVS bond on the default OVS bridge br0 and trunk these interfaces on the physical switch.
  - # Do not include connected 1 Gb interfaces in the same bond or bridge as the 10 Gb interfaces. Leave the 1 Gb interfaces unplugged, or create a separate bond and bridge for the connected 1 Gb interfaces as outlined in the AHV Networking guide.
  - # If required, you can connect the 1 Gb interfaces to physical switches other than the 10 Gb to provide physical network separation for user VMs.
  - # LACP configurations are the preferred method if link aggregation is required.
- Upstream physical switch
  - # Connect the 10 Gb uplink ports on the Acropolis node to switch ports that provide line-rate traffic throughput and are nonblocking.
  - # Use an Ethernet switch that has a low-latency, cut-through design and that provides predictable, consistent traffic latency regardless of packet size, traffic pattern, or the

features enabled on the 10 Gb interfaces. Port-to-port latency should be no higher than 2 microseconds.

  # Use fast-convergence technologies (such as Cisco PortFast) on switch ports that are connected to the AHV host.

- Jumbo frames
  # Nutanix does not currently recommend jumbo frames when using AHV for most deployments. If your system uses Nutanix Volumes and network-heavy iSCSI traffic, refer to the AHV Networking best practices guide to enable jumbo frames.
- IP address management
  # Coordinate the configuration of IP address pools to avoid address overlap with existing network DHCP pools.
- CVM
  # Do not remove the CVM from either the OVS bridge br0 or the native Linux bridge virbr0.
  # If you need network segmentation between storage backplane and management traffic within the CVM to meet security requirements, please see the CVM Network Segmentation feature documentation.
- IPMI ports
  # Do not trunk switch ports that connect to the IPMI interface. Configure the IPMI switch ports as access ports for management simplicity.
- Physical network layout
  # Use redundant top-of-rack switches in a leaf-spine architecture. This simple, flat network design is well suited for a highly distributed, shared-nothing compute and storage architecture.
  # Add all the nodes that belong to a given cluster to the same layer-2 network segment.
- Guest network drivers
  # Use Nutanix VirtIO drivers for Windows-based VMs.
  # Use the NGT installation package to prepare guests for migration to and from AHV or to use VSS and SSR.

AHV Best Practices Checklist

- VM deployment
  # Create VM clones from an optimized base image, cleared of all unneeded applications, data, and instance-specific information.

    - Create clones from a powered-off VM or from a VM snapshot for best performance when creating more than 10 clones.
    - Create new VMs without clones to maximize storage performance for highly latency-sensitive applications.
- VM data protection
    - Use on-demand snapshots for day-to-day VM management tasks.
    - Use scheduled snapshots with protection domains and remote sites for disaster recovery.
    - Ensure that layer-2 VLAN mappings are configured when enabling a remote site protection domain.
- Acropolis Dynamic Scheduler
    - Leave ADS enabled unless you require strict control—specifically, if no live migrations are allowed.
    - Using VM-host affinity, pin each VM to a minimum of two AHV hosts to ensure VMHA functionality for the VM during a VMHA event.
    - Limit the maximum number of VMs in an ADS VM-VM antiaffinity group to the number of AHV hosts in the cluster to ensure that all VMs in the group run on different AHV hosts.
- VM high availability
    - Always use the segments reservation type.
- Live migration
    - Restrict migration bandwidth using the aCLI if required.
    - There is no limit on the number of simultaneous manually initiated migrations, so exercise caution when manually initiating migration of a large number of VMs.
- CPU configuration
    - Increase the vCPU count in VMs rather than cores per vCPU, unless specific VM licensing requirements call for a minimum number of CPU sockets.
    - Use the physical core count instead of the hyperthreaded count for maximum single VM sizing. Do not configure a single VM with more vCPU cores than pCPU cores available on the AHV host.
- Memory configuration
    - Increase the CVM memory in the cluster if deduplication and compression are enabled. 32 GB of RAM for the CVM is a good starting point if using both deduplication and compression.

    # Size VMs to fit within the smallest potential NUMA boundary of the cluster for maximum VM performance on all cluster nodes.

- VM disk configuration
    # Use SCSI disks wherever possible for maximum performance. In order of preference use SATA, PCI, and IDE only where required.
    # Use the default Windows guest configuration to send UNMAP commands to the Nutanix cluster. You can temporarily disable UNMAP temporarily to improve disk format speed.
    # Configure Linux guests with a weekly scheduled fstrim task to return free space to the Nutanix cluster. Format disks with the nodiscard option to improve disk format speed.
    # Install the required VirtIO drivers in the Windows guest VMs using the packaged Nutanix installer.
    # Use NGT when VM mobility, cross-hypervisor DR, VSS, or self-service restore are required. Otherwise, use the standalone Nutanix VirtIO driver package.
- Resource oversubscription
    # Follow industry-standard vCPU oversubscription practices and do not oversubscribe more than 4:1 when booting VMs.
    # Take memory overhead per vCPU into consideration when calculating available memory.
    # Memory cannot be oversubscribed.
- Hugepages
    # Leave hugepages enabled in the AHV host and allow the VM OS or VM administrator to enable huge pages as required in the guest.
- General management
    # Manage the AHV environment through Prism or use SSH to access the CVM when required.
    # Avoid connecting directly to the AHV host.
    # Use one of these options to make sure the configuration is consistent across all CVMs or AHV hosts in the cluster:

```
allssh

hostssh

for i in `svmips`

for i in `hostips`
```

Command Examples

- Network view commands

```
nutanix@CVM$ manage_ovs --bridge_name br0 show_uplinks
nutanix@CVM$ ssh root@192.168.5.1 "ovs-appctl bond/show br0-up"
nutanix@CVM$ ssh root@192.168.5.1 "ovs-vsctl show"
nutanix@CVM$ acli
<acropolis> net.list
<acropolis> net.list_vms vlan.0
nutanix@CVM$ allssh "manage_ovs show_interfaces"
nutanix@CVM$ allssh "manage_ovs --bridge_name <bridge> show_uplinks"
```

- Load balance view command

```
nutanix@CVM$ ssh root@192.168.5.1 "ovs-appctl bond/show"
```

- Load balance active-backup configuration

```
nutanix@CVM$ ssh root@192.168.5.1 "ovs-vsctl set port br0-up bond_mode=active-backup"
```

- High availability options

```
nutanix@CVM$ acli ha.update num_reserved_hosts=X
nutanix@CVM$ acli ha.update reservation_type=kAcropolisHAReserveSegments
nutanix@CVM$ acli vm.update <VM Name> ha_priority=-1
nutanix@CVM$ acli vm.create <VM Name> ha_priority=-1
```

- Live migration bandwidth limits

```
nutanix@CVM$ acli vm.migrate slow-lane-VM1 bandwidth_mbps=100 host=10.10.10.11 live=yes
```

- Viewing NUMA topology

```
nutanix@cvm$ allssh "virsh capabilities | egrep 'cell id|cpus num|memory unit'"
```

- Verifying transparent huge pages

```
nutanix@CVM$ ssh root@192.168.5.1 "cat /sys/kernel/mm/transparent_hugepage/enabled"
[always] madvise never
```

References

1. AHV Administration Guide: Supported Guest VM Types for AHV
2. AHV Administration Guide: Windows VM Provisioning
3. AHV Administration Guide: Changing CVM Memory Configuration (AHV)

4. Prism Web Console Guide: Nutanix Cross-Hypervisor Disaster Recovery

About the Authors

Jason Burns is an NPP-certified staff solutions architect at Nutanix and CCIE Collaboration #20707. He designs, tests, and documents virtual workloads on the Nutanix platform, creating solutions that solve critical business problems. Jason has designed and supported Unified Communications infrastructure in the enterprise for the past decade, deploying UC to connect hundreds of thousands of end users. Outside of his day job, he has an unusual passion for certificates, security, and motorcycles. Follow Jason on Twitter @bbbburns.

Mike McGhee has 17 years of experience in the IT industry and is currently focused on helping customers and partners better understand the core functionality of the Nutanix platform. Mike specializes in stretched clustering solutions and the Microsoft ecosystem of products, including Hyper-V. Prior to Nutanix, Mike worked for EMC as a Systems Engineer with a focus on storage technologies and their intersection with hybrid clouds. He has several certifications, including MCP, MCDBA, and NPP. Follow Mike on Twitter @mcghem.

Magnus Andersson is a senior staff solutions architect at Nutanix. In his role, Magnus develops methods for successfully implementing applications on the Nutanix platform. In addition, he also delivers customer projects, including defining architectural, business, and technical requirements, creating designs, and implementing the Nutanix solution. Magnus holds the Nutanix Platform Expert (NPX) title and two VMware Certified Design Experts (VCDX) titles, the VCDX-DCV and VCDX-Cloud. Prior to joining Nutanix, Magnus worked as a consultant focusing on defining, designing, and implementing public and internal cloud-based solutions for the enterprise and service providers. Follow Magnus on Twitter @magander3.

About Nutanix

Nutanix makes infrastructure invisible, elevating IT to focus on the applications and services that power their business. The Nutanix Enterprise Cloud OS leverages web-scale engineering and consumer-grade design to natively converge compute, virtualization, and storage into a resilient, software-defined solution with rich machine intelligence. The result is predictable performance, cloud-like infrastructure consumption, robust security, and seamless application mobility for a broad range of enterprise applications. Learn more at www.nutanix.com or follow us on Twitter @nutanix.

List of Figures

List of Tables