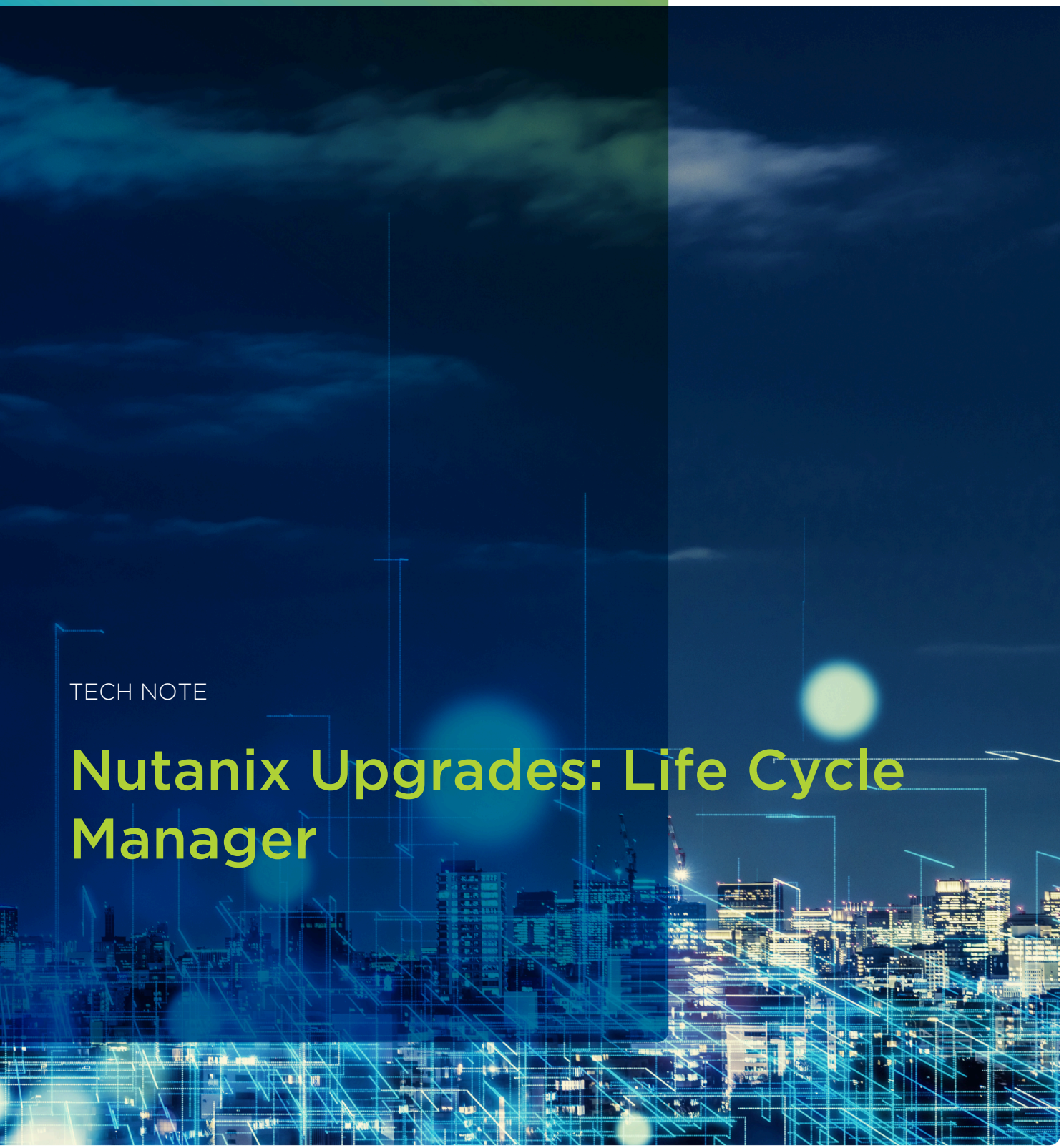
v2.0 | May 2022 | TN-2122



TECH NOTE

Nutanix Upgrades: Life Cycle Manager

NUTANIX
YOUR ENTERPRISE CLOUD

Copyright

Contents

1. Executive Summary....4
Document Version History....4

2. Why LCM Is Critical....5

3. LCM Design....7

4. Taking Inventory....8

5. Software Upgrades....10
AOS Upgrades....11
AHV Upgrades....14
Third-Party Hypervisor Upgrades....20

6. Firmware Upgrades....24
Available Upgrades and Dependency Checking....24
Orchestration Engine....26
Redfish Protocol....29
In-VM Updates (IVU)....30

7. Conclusion....31

8. Appendix....32
References....32

About Nutanix....33

List of Figures....34

1. Executive Summary

Nutanix designed the upgrade process from the ground up to provide the best customer experience and reduce risk during upgrades, which historically are risky and time consuming for datacenters. One-click upgrades apply to all the software products in the Nutanix platform.

To maintain simplicity, we developed the second generation of one-click upgrade and released it as Life Cycle Manager (LCM), which allows simple upgrades of software layers that have multiple dependencies.

The first goal of LCM was to expand server firmware upgrades beyond NX-branded appliances. LCM can now provide one-click firmware upgrades for all Nutanix appliances, including OEM appliances. LCM is the only tool that can upgrade firmware on multiple server platforms.

Document Version History

Version Number	Published	Notes
1.0	May 2019	Original publication.
2.0	May 2022	Updated screenshots throughout and the Software Upgrades and Firmware Upgrades sections. Added LCM Design, Taking Inventory, Redfish Protocol, and In-VM Updates (IVU) sections.

2. Why LCM Is Critical

Lost productivity and higher maintenance costs due to old software and firmware aren't the only mission-critical concerns for IT admins. There are also concerns surrounding system vulnerabilities.

Intruders have had time to learn how to break past the security in older hardware. Older software versions often lack the improvements to features such as asset management that newer software versions have. These features enable better analysis and improved workload and performance tracking and are critical for enterprises today. Barry Angell's blog post contains further discussion of the hidden costs of aging infrastructure. Meanwhile, even as they apply the latest firmware and updates, users have concerns about compatibility and security compliance with other version and tools. All these factors make the process of applying upgrades cumbersome and time intensive.

LCM greatly simplifies infrastructure upgrades and version compliance. Data collected on the Dell XC Series 630-10 for a four-node cluster shows that the LCM 2.0 update is 70 percent faster than a traditional update cycle.

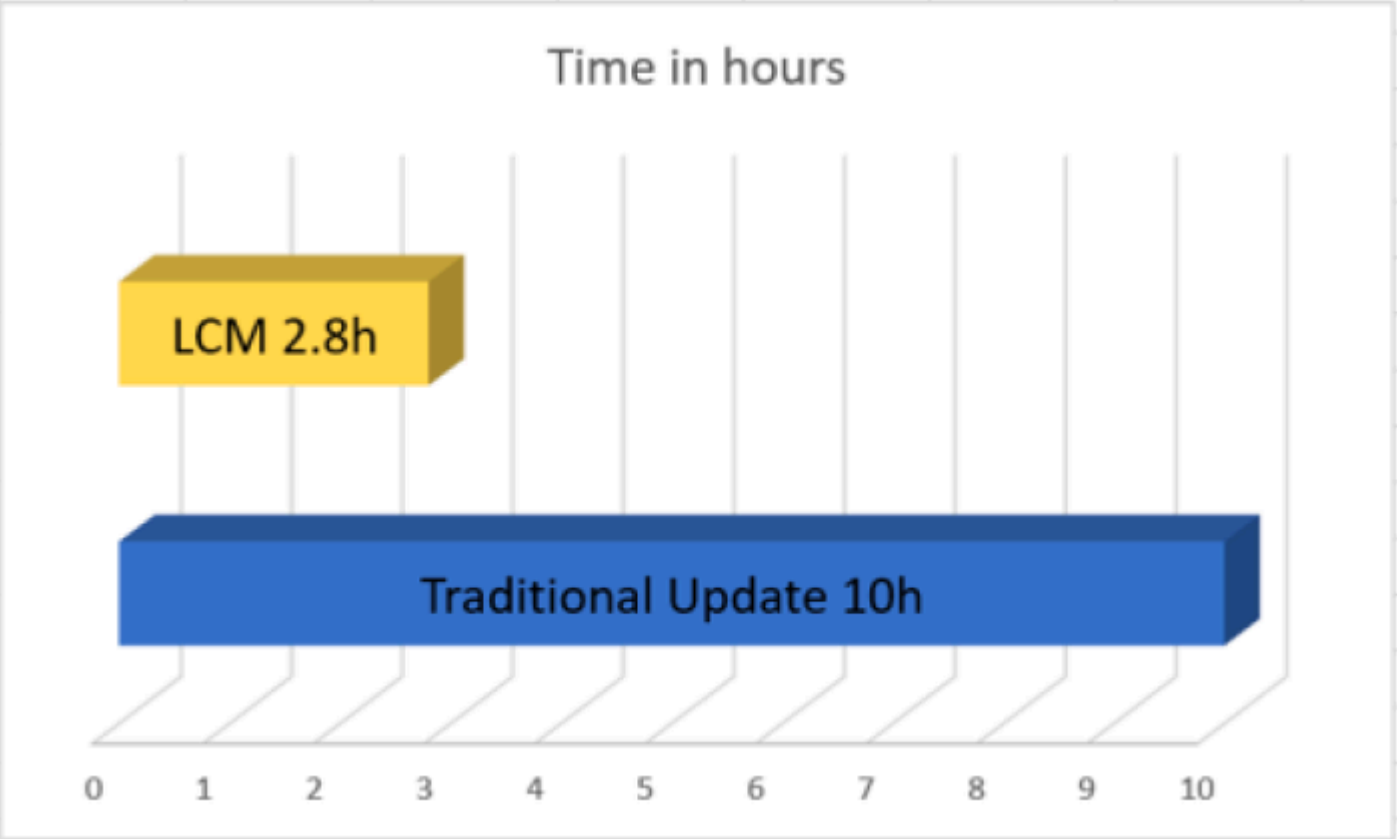


Figure 1: Update Time: LCM vs. Traditional

3. LCM Design

LCM is a pluggable framework so that it can be kept up to date in every customer environment (automatically on portal-connected clusters), and so that software and firmware modules can be released and updated independently and instantly consumed.

As of LCM 2.4.5, you can choose to auto-update Nutanix Cluster Check (NCC), which automatically updates NCC without manual intervention. We plan to eventually expand this option to other software modules to mirror a cloud-like experience where all software is updated automatically.

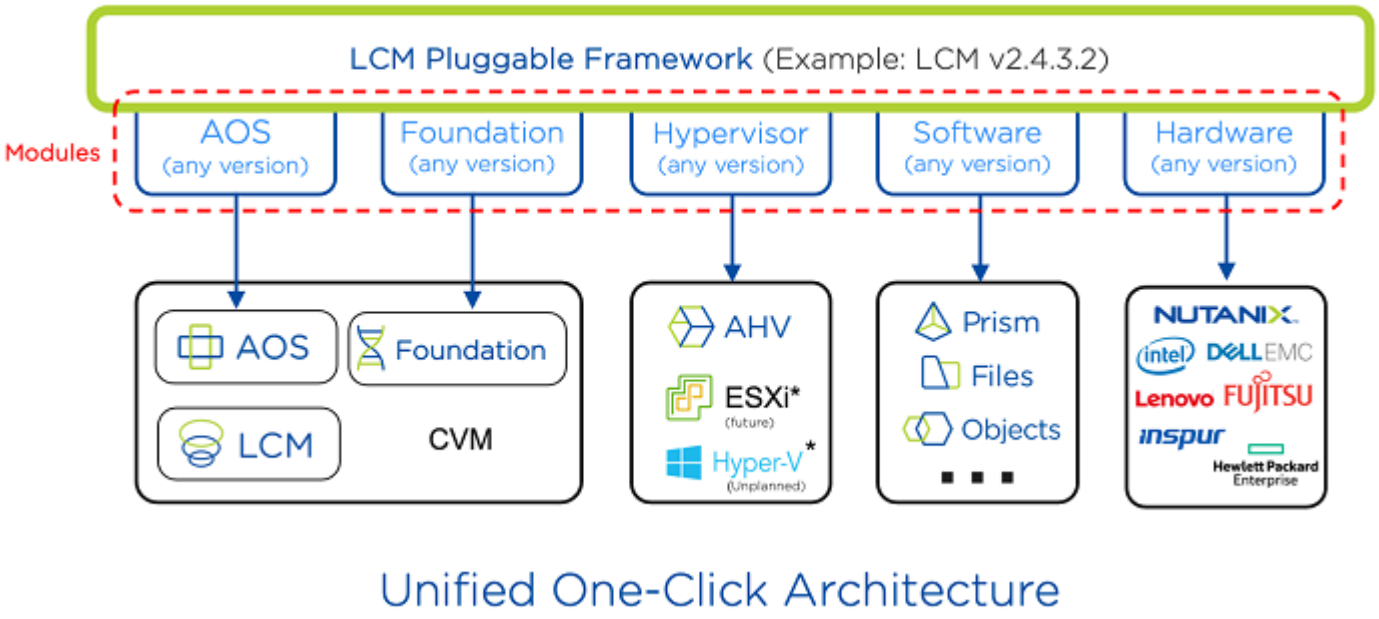


Figure 2: LCM Design

4. Taking Inventory

Before you perform any upgrade operation, you must take inventory so that you know what version of software is running on the system and what firmware each node contains. The inventory also updates the LCM framework if required. The LCM page in Prism displays this inventory for each cluster. Once you start the initial inventory, you can set it up to run automatically at a specified interval (typically every 24 hours).

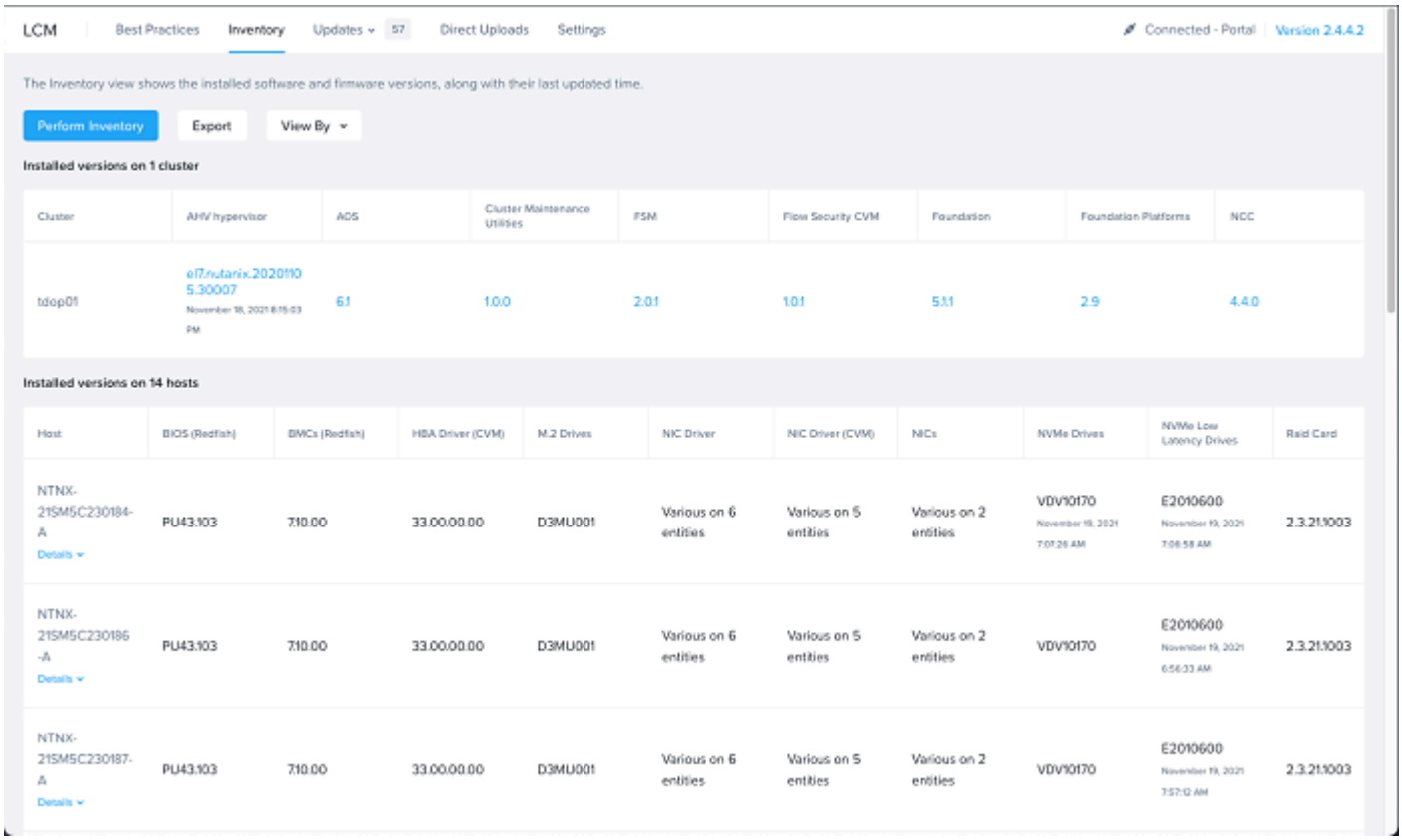


Figure 3: LCM Inventory

LCM initially builds the inventories by downloading all the upgrade modules from the Nutanix portal or dark site bundle and storing them in the Catalog and the Insights Data Fabric (IDF). The Catalog is a service running on the CVM that

stores the images, and the IDF is the distributed database the Nutanix platform uses to store configuration data, alert data, and performance data. LCM then determines which modules it needs to send to which endpoints; for example, every node gets the CVM and host modules. LCM sends the modules to each endpoint using SSH.

Once the modules are in place, the inventory process asks each one to run its detect function, which interrogates its related entities and discovers the current versions on all endpoints. LCM uses this data to build the inventory summary and stores it in the IDF for future lookups. LCM can now calculate the available upgrade versions by comparing the discovered inventory data to options on the portal or dark site bundle.

The inventory updates when you use LCM to perform future upgrades. When new content is released, you're prompted to run inventory if you haven't run it yet. You can also request a manual inventory through Prism to confirm the current state of the endpoints at any time.

5. Software Upgrades

Updating software is unavoidable. As datacenters and the entire world become increasingly software driven, updates shouldn't be a point of concern. Establishing a process and reputation that builds confidence in regular updates is central to our mission. To deliver an excellent upgrade experience, we needed to remove the complexity encountered with legacy hypervisor solutions.

In legacy scenarios, the upgrade process is as adversely affected as the deployment process. Users must upgrade each individual piece of a traditional solution separately and in a specific order to maintain functionality and availability. Complex traditional upgrade processes generate massive upgrade manuals that are time-consuming to create and maintain. Creators also must review them with every version upgrade because the architectures change over time, which affects the process and number of pieces involved in an upgrade. The result is that organizations don't upgrade as often as they should, and many elect to only upgrade to major releases or even leapfrog major releases, applying only critical security fixes in between. Many organizations are also hesitant to perform the upgrades themselves and elect to pay for professional services or a partner to perform the upgrades.

Nutanix offers simple upgrades via LCM for the different software components in the platform. After you run an inventory, you can view and select available software updates from the LCM menu and create an update plan. For sites without internet access, you can upload software bundles to LCM using the Direct Upload feature (with LCM 2.4.2 and later versions) or set up a local web server to host the LCM repository.

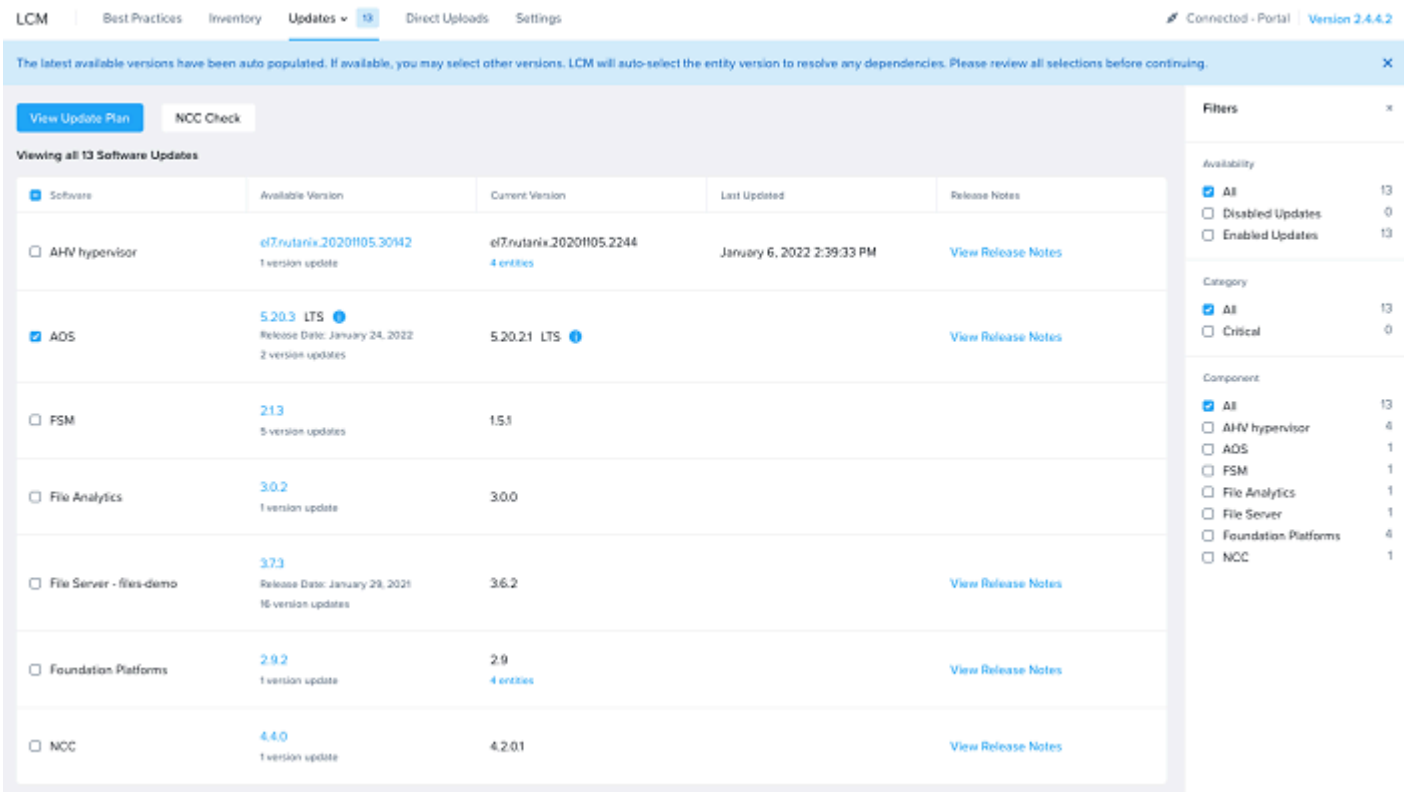


Figure 4: Available Software Updates

Once you have selected the software to upgrade, you can review the update plan and start the upgrade, which runs in the background.

AOS Upgrades

The control plane (Prism) and data plane (AOS storage) are both updated as part of an AOS upgrade. AOS upgrades typically offer a wealth of benefits, such as bug fixes, security updates, new features, and performance updates, which means AOS is typically the most upgraded layer in the Nutanix platform. You can perform these upgrades without moving VMs around or upgrading the underlying hypervisor.

LCM shows the latest long-term support (LTS) version by default, but you can select a different version by clicking the version number. If you have any

questions about whether you can upgrade from one version of AOS to another, check the Upgrade Paths page on the Support portal.

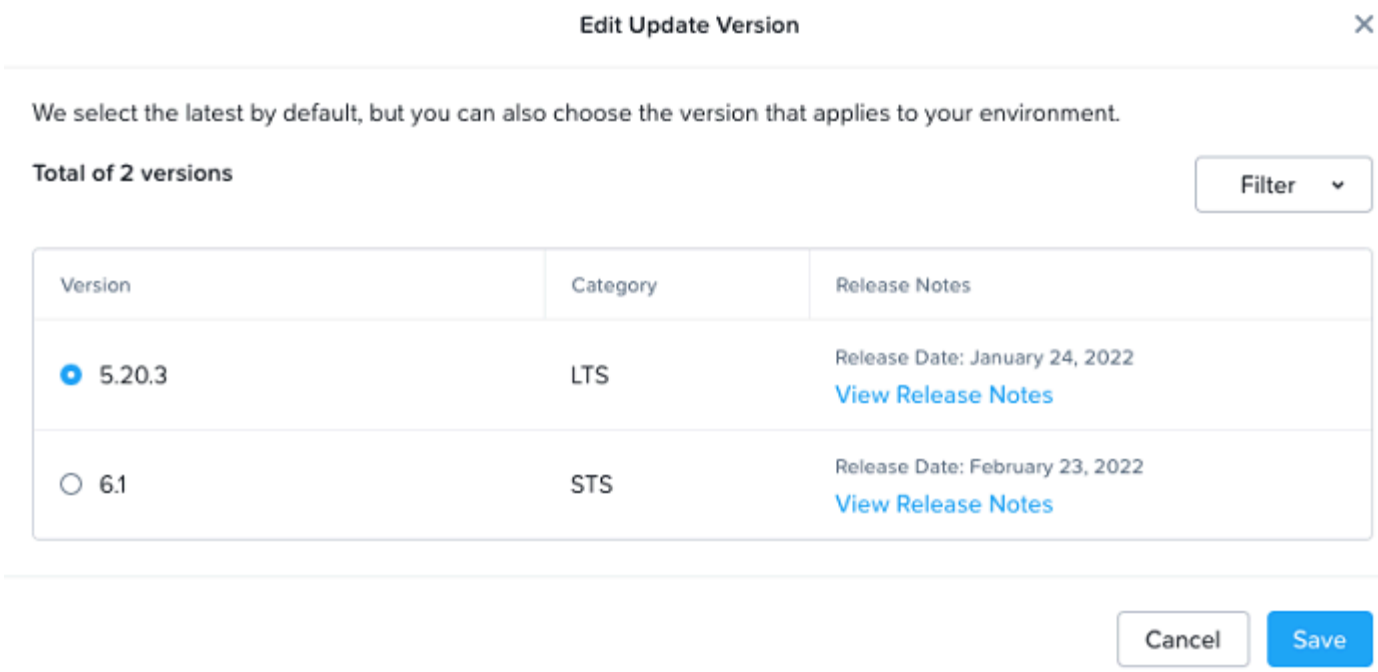


Figure 5: Available Update Versions

To begin the upgrade, select the checkbox for the AOS upgrade, click View Update Plan, review the plan, then click Apply 1 Update. LCM pulls the package down directly from the Support portal. For clusters without internet connectivity, the administrator can download the AOS LCM bundle from the Support portal and either directly upload the bundle to LCM (with LCM 2.4.2 and later versions) or set up a local web server. The software binaries are copied to two nodes in the cluster to ensure that a copy is always available during the process.

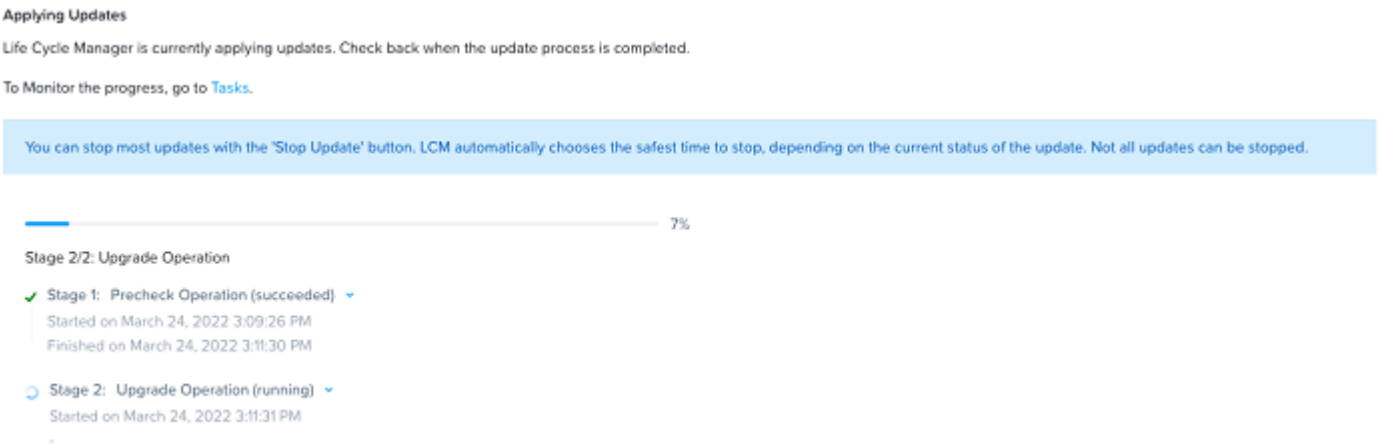


Figure 6: Upgrade in Progress

The first phase of any upgrade is running preupgrade checks. These prechecks ensure that an upgrade can be successful if it proceeds. The first check is for version compatibility, ensuring that the desired AOS version can upgrade from the existing AOS version and support the installed hypervisor version along with any other features running in the cluster. Next, the process ensures that there is network connectivity between all CVMs and hypervisor nodes. There is a check for free space and cluster status, along with Cassandra, Zookeeper, and Stargate health checks. After passing all the prechecks, the upgrade process can proceed. If an error or blocker occurs, Prism displays a message and logs the error in the alerts section for review.

The upgrade process installs the new AOS version to all nodes in the cluster in parallel, but the new version isn't active until the CVM restarts. Once it installs the new AOS version on all the CVMs, the process issues the upgrade token to the first node identified. The upgrade token can only be held by one CVM in a cluster, which means it must be on the only node that can restart. This first node restarts just the CVM to activate the new AOS version.

During AOS upgrades, you don't need to migrate the VMs running on each node off the node. While the node holding the upgrade token performs the CVM reboot, I/O from the local VMs is temporarily redirected to other CVMs in the cluster. This I/O redirection eliminates the need to move VMs around and reduces the time required to perform an upgrade.

Different hypervisors redirect I/O differently. ESXi and Hyper-V use a process called CVM Autopathing, which uses HA.py to forward traffic from the local internal address (192.168.5.2) to the external IP addresses of other CVMs throughout the cluster. This process keeps the datastore intact; the CVM responsible for serving the I/O is simply remote. Once the local CVM comes back and is stable, the redirection ends and the local CVM takes over all new I/O again.

AHV uses iSCSI multipathing, where the primary path is the local CVM and the two other paths are remote. If the primary path fails, one of the other paths becomes active. As with Autopathing, when the local CVM comes back online, it takes over again as the primary path.

After the node holding the token restarts, it ensures that all local services on the CVM are running and safely rejoin the cluster after passing an integrity check. The node then releases the upgrade token and the upgrade process issues it to the next node in the cluster. The process repeats for each node in the cluster until the upgrade finishes. All these steps and checks together produce a rolling, nondisruptive upgrade process where you don't need to migrate any VMs.

AHV Upgrades

You also perform AHV upgrades through LCM. On clusters that have internet connectivity, select the checkbox for the AHV upgrade, click View Update Plan, review the plan, then click Apply Update(s). LCM pulls the package directly from the Support portal. For clusters without internet connectivity, the administrator can download the AHV LCM bundle from the Support portal and either directly upload the bundle to LCM (with LCM 2.4.2 and later versions) or set up a local web server.

The upgrade process first evaluates the cluster to ensure that it's in a healthy state before allowing an upgrade to begin. It uses the JSON metadata file included with each version to validate that the AHV version uploaded can update from the AHV version currently running and that it's compatible with the AOS version running on the cluster. After passing these checks, the process

copies the AHV .tar package to the hypervisor host on each node in the cluster so it's accessible during its phase of the rolling upgrade.

The upgrade process selects the first node to upgrade, issues it the upgrade token, then puts that node in maintenance mode. Any VMs running on that node live-migrate to other nodes in the cluster. Some VM types can't be live-migrated:

- Agent VMs
- VMs using GPU passthrough
- Nested VMs with CPU passthrough

In these cases, these VMs receive an ACPI shutdown request for a graceful shutdown. If the VM doesn't shut down in the allotted two minutes, the process issues a shutdown command for the VMs still running.

> Note: Agent VMs are always the last to shut down and the first to turn back on following the CVM.

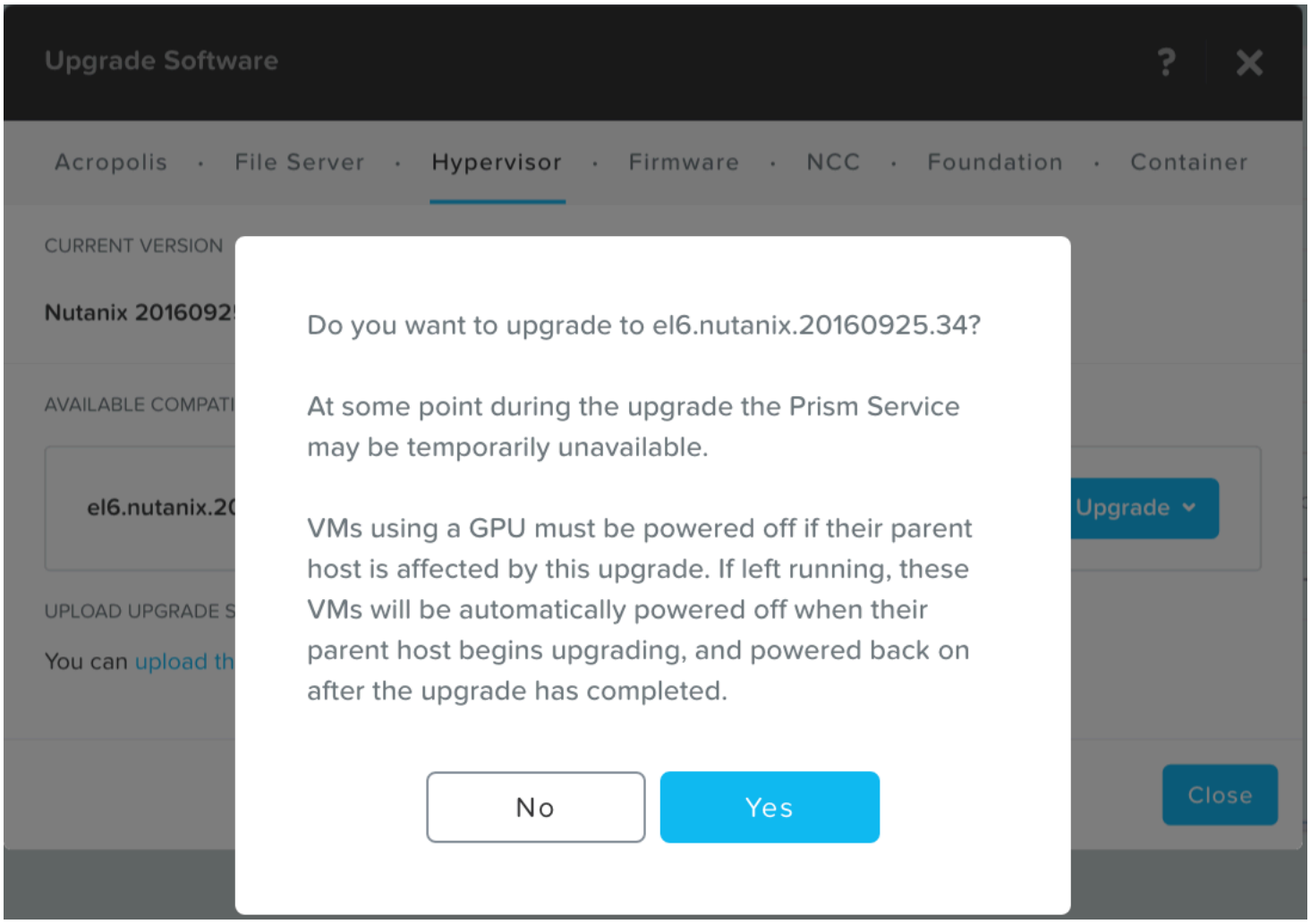


Figure 7: VM Using a GPU Error Message

The VMs you couldn't migrate because of their physical constraints automatically turn on again once the host has been upgraded and restarted to complete the process.

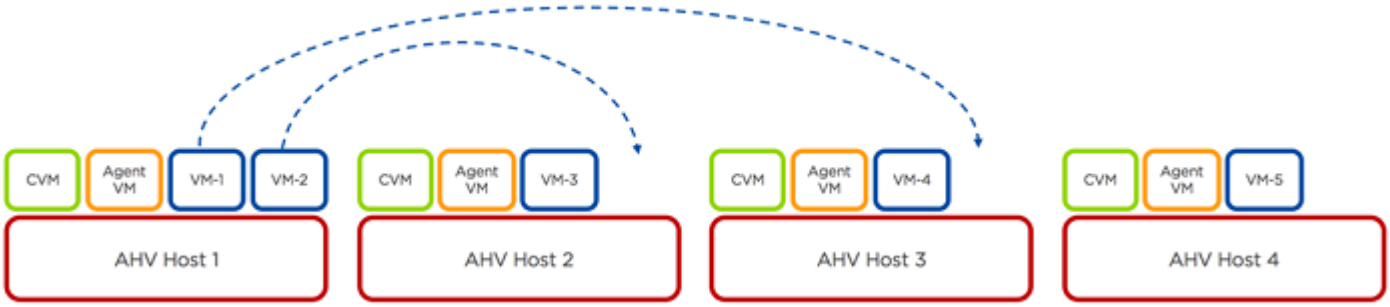


Figure 8: VM Evacuation

Once all the VMs have moved or shut down, the process runs the upgrade commands. Because AHV is based on CentOS, it uses a standard yum upgrade process and references the repository on the local CVM (Nutanix repo) where the process placed the .tar file. Yum compares the versions of all the RPM packages in the local upgrade repo against the installed versions and only upgrades the RPM packages that have an updated version available. The infrastructure process on the local CVM controls this workflow using SSH commands issued to the host.

Some of the upgraded RPMs have new configuration settings. In this case, the upgrade workflow uses the configuration management tools Puppet and Salt to apply the configuration changes immediately, keeping the node in compliance with the cluster's configuration.

When the node finishes upgrading, the automated process issues a time-delayed shutdown command to the AHV host that gracefully shuts down the local CVM and restarts the node.

Figure 9: CVM Shutdown

After restarting the host, the CVM automatically starts and must pass upgrade checks that ensure the host upgrade was successful. The Genesis service verifies that it's running the new kernel version, then the CVM runs an abbreviated storage integrity check before rejoining the storage cluster.

Because the cluster understands that the CVM was safely shut down as part of the upgrade process, it doesn't require the full integrity check normally required when an unexpected failure occurs.

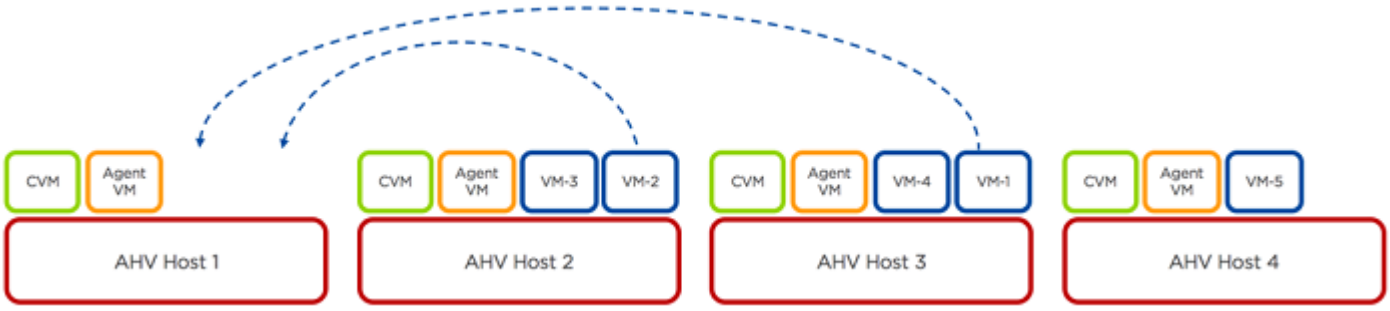


Figure 10: VM Locality Restoration

Once the integrity checks finish, the host exits maintenance mode. The unmigrated VMs that were shut down turn on, and the VMs that live-migrated to other hosts move back to the newly upgraded node to restore data locality.

Figure 11: Host Upgrade Complete

During the rolling upgrade process for the cluster, when you live-migrate VMs between hosts to allow a node to upgrade, the infrastructure service prevents any compatibility issues. A cluster going through the upgrade process has two versions of AHV deployed: some hosts have the existing version (which we call AHV.old) and others that have already been upgraded have AHV.new. When you live-migrate VMs between nodes, they can only move between hosts that have both AHV.old and AHV.new or from hosts with AHV.old to hosts with AHV.new. The process blocks VMs from moving from a host with AHV.new to one with AHV.old to prevent any loss of new capabilities.

The upgrade process selects the next host to upgrade in the cluster, passes it the upgrade token, and repeats the process until the entire AHV cluster has been upgraded. You can monitor the upgrade progress from LCM or the Tasks

view in Prism to see which host is being upgraded currently as well as the overall progress.

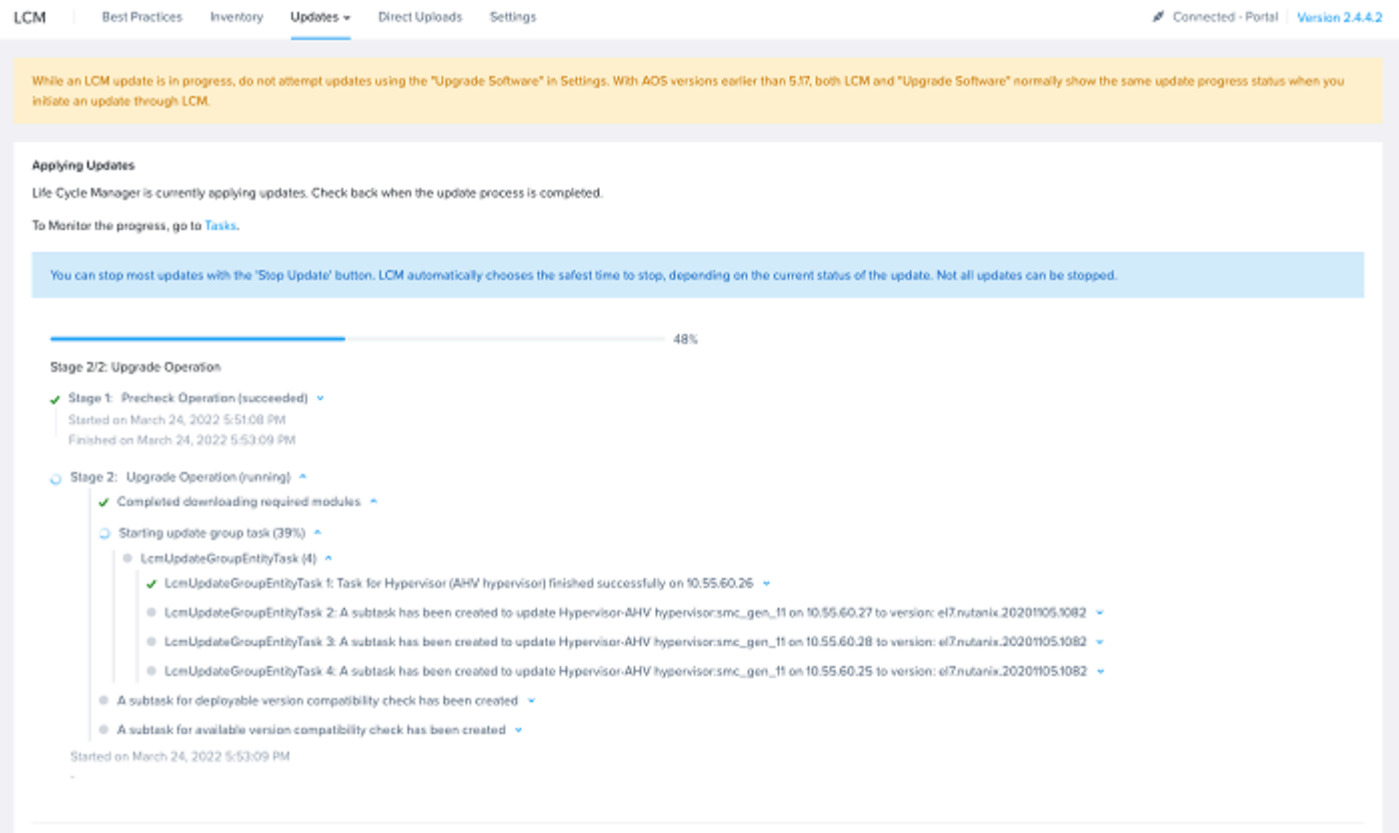


Figure 12: AHV Upgrade in Progress

Although it isn't a typical approach, you can also use a command-line interface (CLI) to run AOS and AHV upgrades if desired or instructed by Nutanix Support. The CLI method provides additional flexibility not available in Prism, such as the ability to change pinned VM migration behaviors or specify a different repository for upgrade packages.

If any of the upgrade commands issue a nonzero response code, it's considered a failure and Genesis stops the upgrade. In these situations, engage with Nutanix Support to quickly resolve the issue. To retry the upgrade on the host that had the failure, restart the Genesis service on the local CVM to force it to retry the same upgrade. If it fails again, you can refer to the log files to identify which command is failing to understand how to proceed.

There are several logs you can look at if you need more details than the Prism event and task lists provide. The following are logs with data written to them as part of the upgrade process:

- Genesis logs on CVMs:
  - `/home/nutanix/data/logs/genesis.out`
  - `/home/nutanix/data/logs/host_preupgrade.out` (on Genesis leader, node that triggers upgrade)
  - `/home/nutanix/data/logs/host_upgrade.out`
  - `/home/nutanix/data/logs/lcm_ops.out`
- AHV logs on hosts:
  - `/var/log/yum.log` (explains what the upgrade ran)
  - `/var/log/upgrade_config.log` (primary log for upgrade details)
  - `/var/log/upgrade_config-puppet.log` (Puppet-related details; examine if mentioned in `upgrade_config.log`)
  - `/var/log/upgrade_config-salt.log` (Salt-related details; examine if mentioned in `upgrade_config.log`)

Third-Party Hypervisor Upgrades

Upgrading a third-party hypervisor such as ESXi or Hyper-V is a one-click upgrade in Prism Element, under Settings > Upgrade Software > Hypervisor. You must first download the hypervisor upgrade bits from the vendor's portal, upload them to Prism as part of the one-click process, then download a JSON metadata file from the Nutanix Support portal for your target hypervisor version. Once these are downloaded, upload them to Prism Element, click the Upgrade button, and let the automated process take care of everything.

First, the upgrade process evaluates the cluster to ensure that it's in a healthy state before allowing an upgrade to begin. It uses the JSON metadata file to validate that the hypervisor version uploaded can update from the running version and is compatible with the AOS version currently running on the cluster.

In addition to the JSON checks, the upgrade process also performs several prechecks before allowing the upgrade to proceed. There is a precheck for cluster status and free space, along with Cassandra, Zookeeper, and Stargate health checks. After passing the prechecks, the process copies the hypervisor package to the CVM on each node in the cluster so it's accessible during its phase of the rolling upgrade.

The upgrade process selects the first node to upgrade, issues it the upgrade token, then asks to put that node in maintenance mode. Any VMs running on that node are live-migrated to other nodes in the cluster. Some VM types can't be live-migrated:

- Agent VMs
- VMs using hardware or GPU passthrough
- Nested VMs with CPU passthrough

You need to remediate these VMs, either before you begin the upgrade process or manually during the upgrade. Identifying all VMs in the cluster that can't live-migrate and shutting them down before starting upgrade is the simplest method but results in longer downtime for these VMs, as they must turn back on once the upgrade finishes. The other option is to monitor the upgrade process using the hypervisor management console and, when a node enters maintenance mode, shut down these VMs on just that node, then turn them back on once the node restarts. Repeat as each node is upgraded in the cluster.

> Note: If a node fails to enter maintenance mode, this process and the one-click upgrade process eventually time out, causing the upgrade to fail. You can remediate the problem VMs and restart the upgrade process to complete the remaining nodes in the cluster.

Once all the VMs have moved or shut down, the process runs the upgrade commands. With third-party hypervisors, the upgrades are performed using the CLI from a remote CVM in the cluster and the hypervisor upgrade bits are sourced from another remote CVM in the cluster.

When the node finishes upgrading, the automated process issues a time-delayed shutdown command to the hypervisor host that restarts the node.

After restarting the host, the CVM automatically starts and must pass upgrade checks that ensure the host upgrade was successful. Then the CVM runs

an abbreviated storage integrity check before rejoining the storage cluster. Because the cluster understands that the CVM was safely shut down as part of the upgrade process, it doesn't require the full integrity check normally required when an unexpected failure occurs.

Once the integrity checks finish, the host exits maintenance mode. The VMs that couldn't live-migrate and shut down either restart now or wait until the entire upgrade completes. Depending on the hypervisor and the settings, some of the VMs that live-migrated to other hosts move back to the newly upgraded node.

The upgrade process selects the next host to upgrade in the cluster, passes it the upgrade token, and repeats the process until the entire hypervisor cluster has been upgraded. You can monitor the upgrade progress from the Tasks view in Prism to see which host is being upgraded currently as well as the overall progress.

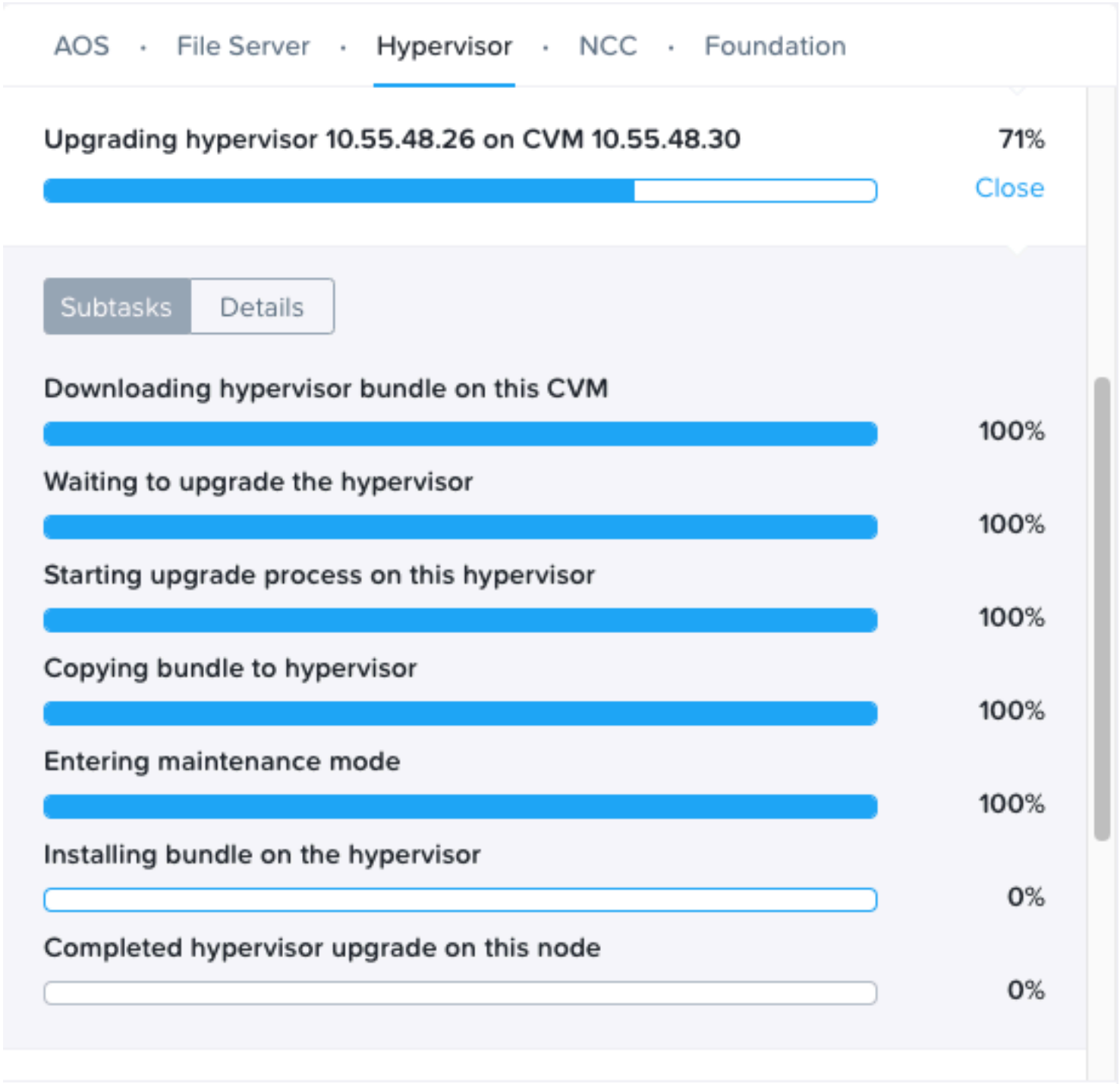


Figure 13: Third-Party Hypervisor Upgrade in Progress

6. Firmware Upgrades

Firmware upgrades are the most avoided upgrades in every organization's datacenter. The firmware upgrade process typically involves time-consuming manual research to check whether an upgrade is available, needed, and beneficial before moving forward. The process of running the actual upgrades also varies greatly between vendors and, depending on your configuration, a given vendor may even have multiple available options.

LCM simplifies this problem for Nutanix environments. LCM provides a single process in Prism that identifies and qualifies upgrade paths, then uses a nondisruptive one-click process to complete the upgrade. LCM can perform firmware upgrades on the following server platforms:

- Nutanix NX
- Dell XC and XC Core
- Lenovo HX and HX Core
- HPE DX
- HPE DL (G10)
- Fujitsu XF
- Intel DCB
- Inspur InMerge

When you upgrade firmware, we recommend that you use the latest versions of AOS, Foundation, and LCM.

Available Upgrades and Dependency Checking

Most IT organizations become frustrated trying to determine if and when an upgrade is available and which one they should use. Mobile devices have set the bar for upgrades, and Prism meets this expectation for a consumer upgrade

experience by comparing the available inventory to metadata available on the portal or dark site bundle. This comparison provides a convenient view of available upgrades for the nodes and cluster.

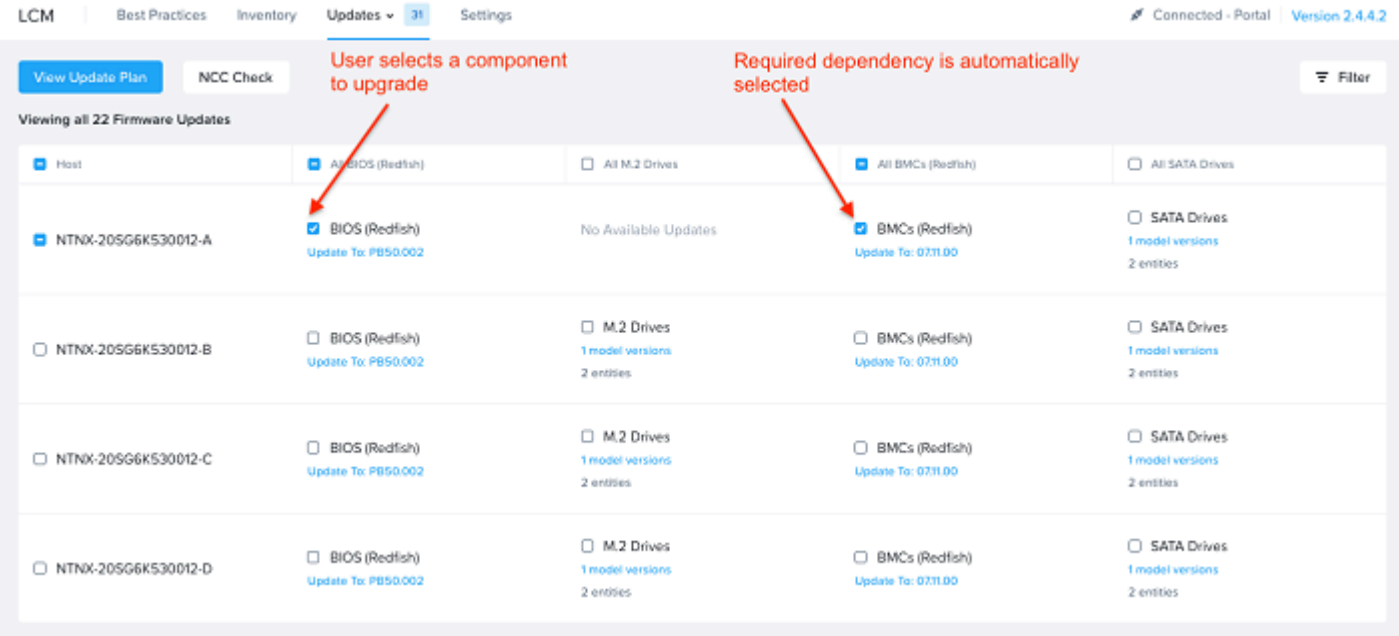


Figure 14: LCM Dependencies

As our supported hardware base expands and additional software products are released, managing the growing list of dependencies requires a tremendous amount of invisible logic and automation. The logic understands dependencies and provides a recommended upgrade version. The recommended version takes into consideration firmware's dependencies on other items in the node as well as the versions for software like AOS running on the cluster. Sometimes, before one upgrade can complete, the system must remediate the dependencies, which typically occurs during the upgrade process because the orchestration engine understands package order.

There is often more than one upgrade version available for a particular entity. The recommended version is generally the best choice because it's based on dependencies and the maturity of the firmware version. For example, a firmware version that's been available for several months and widely deployed is preferable to one that's only weeks old. If you choose a version other than the

recommended one, Prism allows you to easily change the version you want to apply.

Finally, the logic knows if a server vendor requires all firmware updates to be bundled and upgraded together.

Orchestration Engine

The orchestration engine behind LCM upgrades takes the inputs from the inventory, available versions, and dependencies and runs the nondisruptive upgrades. By this point, users have already looked at the inventory and selected the entities and versions they wish to upgrade (downloaded from the portal or dark site bundle) and can now click the upgrade button.

When the process starts, it evaluates the cluster to ensure that it's in a healthy state before it allows an upgrade to begin. The engine polls all hosts in the cluster; the order of the hosts that the poll returns becomes their upgrade order. The process discovers any VMs that can't migrate and presents a list of them for the administrator to remediate before the upgrade starts. Clusters running AHV automatically shut down these VMs on a host as it enters maintenance mode. Other hypervisors, such as ESXi, require the administrator to shut down the VMs until the upgrade finishes. Most often, you can't live-migrate these VMs to another host because of a passthrough hardware device or an affinity rule that only allows the VM to run on a declared host.

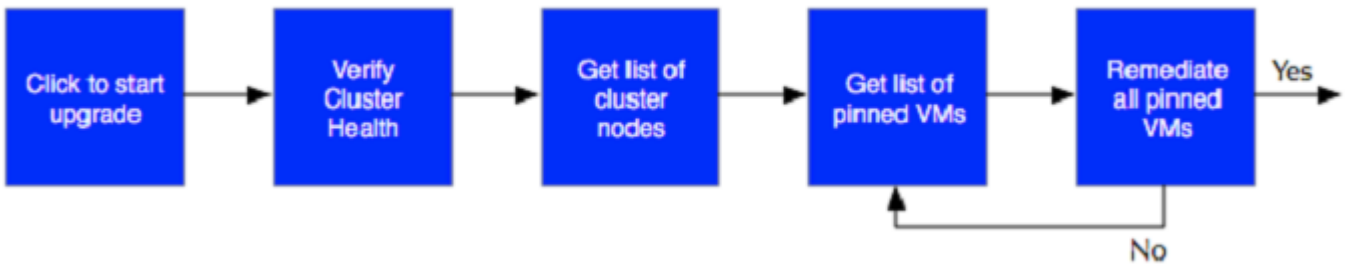


Figure 15: LCM Upgrade Flow Part 1

Once the pinned VMs have been remediated, AOS issues the shutdown token to the first host on the list. Only one host in a cluster can possess the shutdown token at a time; the token allows it to enter maintenance mode and restart.

The host with the shutdown token issues a request to the hypervisor to enter maintenance mode, which evacuates all remaining VMs from the host with the CVM. Once all VMs have evacuated, the CVM receives a maintenance mode command.

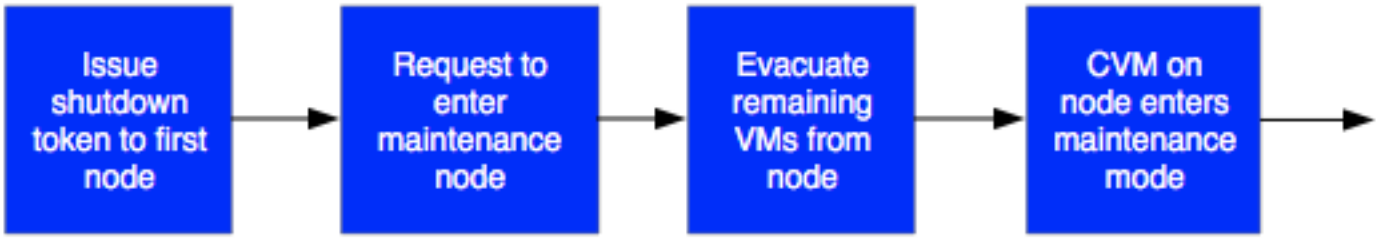


Figure 16: LCM Upgrade Flow Part 2

With the CVM in maintenance mode, the upgrade process takes one of two paths to upgrade the firmware based on the capabilities of the underlying server infrastructure. A few server vendors offer server tools installed in the hypervisor layer and allow you to upgrade firmware using these tools. In this case the firmware upgrades are batched together and run in a single serial upgrade process. This batched process applies the upgrades in the order required by the understood dependencies. Rare cases may require an additional restart before applying a specific firmware upgrade; in these cases, the node restarts and the server tools apply any remaining upgrades after the restart.

For server platforms without server tools, the upgrade process reboots the node into Nutanix Phoenix. Phoenix is a Linux-based ISO that allows a node to boot into an environment and provides the tools necessary to upgrade firmware. The Phoenix ISO is presented from another CVM in the cluster and mounts using the IPMI interface on the node being upgraded. Once in Phoenix, the firmware upgrades are batched together and run in a single serial upgrade process, just as with server tools.

Once the upgrade process finishes on the node, the node receives a restart command.

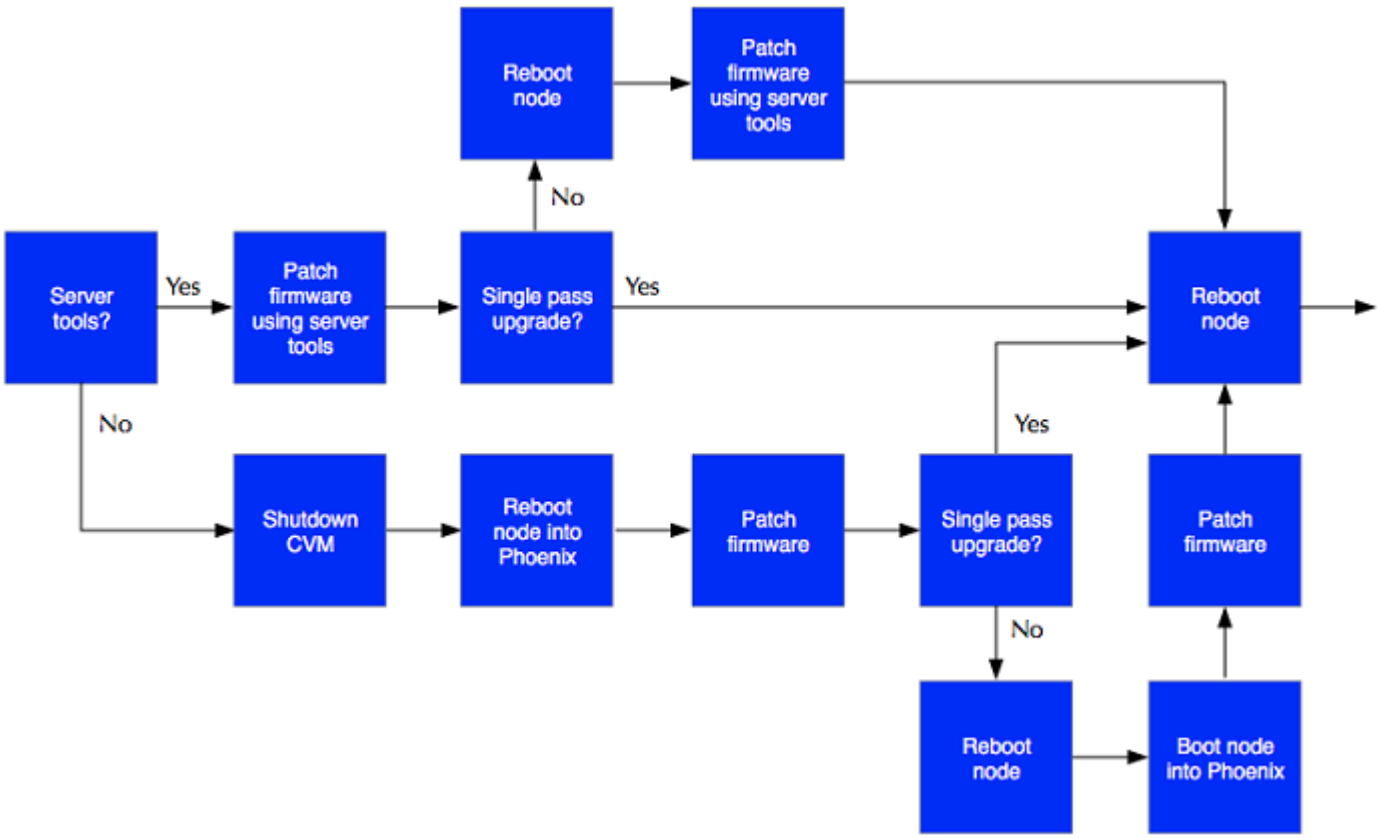


Figure 17: LCM Upgrade Flow Part 3

The node starts back into the installed hypervisor and starts the CVM. The process requests that the node exit hypervisor maintenance mode. The local CVM processes health checks before exiting maintenance mode and rejoining the cluster from a storage perspective. The LCM inventory updates to reflect the new firmware versions for the node along with the upgrade date. The process then selects the second node from the list and repeats the same process, working through each node in the cluster until every node has upgraded. Then the upgrade task shows as completed in the Tasks view in Prism.

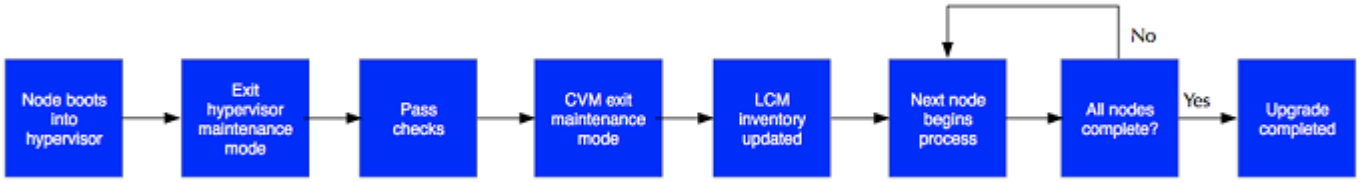


Figure 18: LCM Upgrade Flow Part 4

Redfish Protocol

As of LCM 2.4, LCM uses the Redfish protocol to update firmware through a virtual USB NIC on NX G6, NX G7, Dell XC 14G, and Dell XC 15G platforms. Redfish uses a RESTful API interface for server management. On Nutanix NX platforms, Redfish is enabled for the BMC and BIOS. On Dell XC platforms, Redfish is enabled for all XC components.

The most visible effect of Redfish is that LCM performs BMC and BIOS updates about twice as quickly as the procedure that uses the Phoenix ISO. With Redfish, LCM doesn't need to start into Phoenix and performs fewer system restarts. Using the older procedure, LCM restarted the system twice for BMC updates and three times for BIOS updates. With Redfish, LCM doesn't need to restart the system for BMC updates and only needs one restart for BIOS updates.

LCM detects and shows Redfish availability in the UI. For the full list of system requirements, see the Life Cycle Manager Guide.

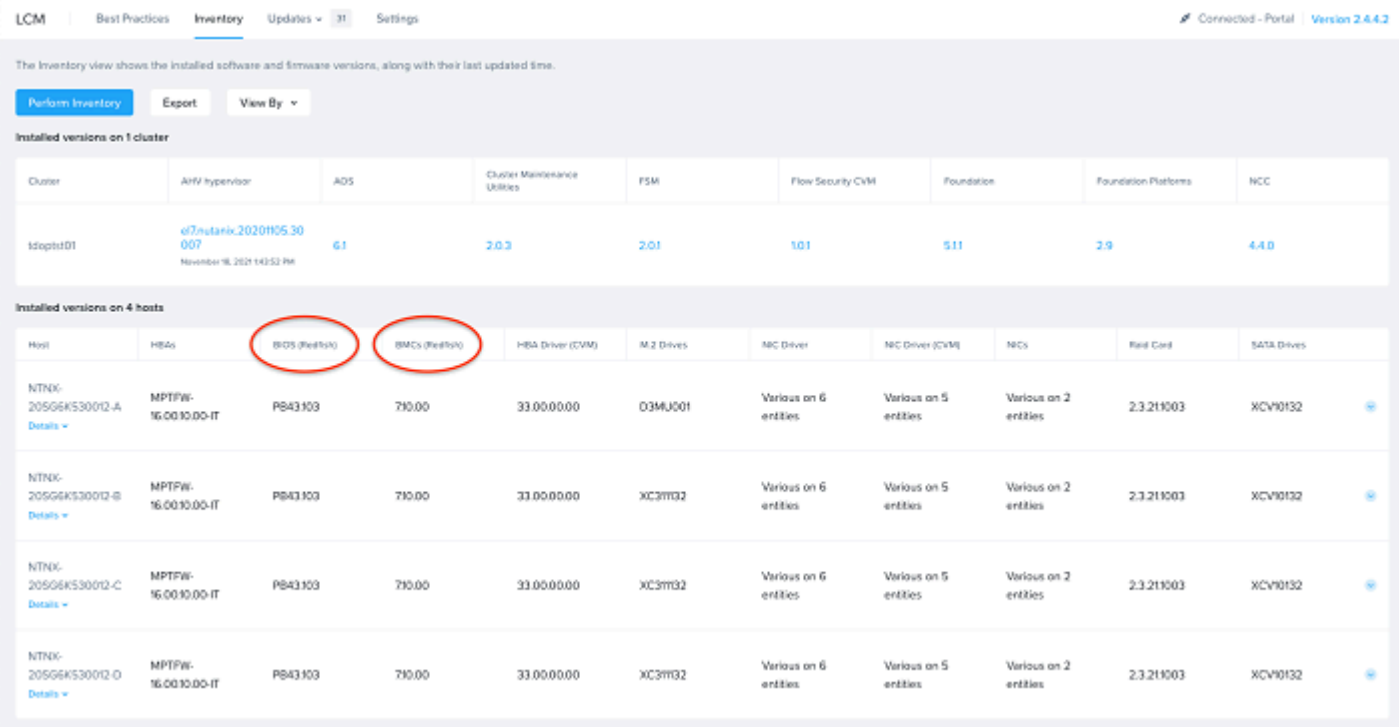


Figure 19: Redfish Availability

In-VM Updates (IVU)

On NX platforms, LCM can perform disk firmware updates through the CVM without requiring a host restart. Since the data disks and host bus adapter (HBA) controller are passed through to the CVM, LCM can start the CVM into a live CD that performs the updates on the disks and HBA controller instead of needing to start the node into Phoenix. Using this method, you don't have to migrate the VMs on the host, which results in much faster updates.

7. Conclusion

Nutanix offers a comprehensive set of upgrade solutions across the stack to manage software and firmware life cycles and ensure that environments provide the highest availability and best performance and are free from known security vulnerabilities.

8. Appendix

References

1. The Hidden Costs of Your Aging IT Infrastructure
2. Life Cycle Manager Guide

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

Figure 1: Update Time: LCM vs. Traditional....6
Figure 2: LCM Design....7
Figure 3: LCM Inventory....8
Figure 4: Available Software Updates....11
Figure 5: Available Update Versions....12
Figure 6: Upgrade in Progress....13
Figure 7: VM Using a GPU Error Message....16
Figure 8: VM Evacuation....17
Figure 9: CVM Shutdown....17
Figure 10: VM Locality Restoration....18
Figure 11: Host Upgrade Complete....18
Figure 12: AHV Upgrade in Progress....19
Figure 13: Third-Party Hypervisor Upgrade in Progress....23
Figure 14: LCM Dependencies....25
Figure 15: LCM Upgrade Flow Part 1....26
Figure 16: LCM Upgrade Flow Part 2....27
Figure 17: LCM Upgrade Flow Part 3....28
Figure 18: LCM Upgrade Flow Part 4....29
Figure 19: Redfish Availability....30