WHITE PAPER

THE ULTIMATE GUIDE TO **smart**Infrastructure

nebulon®

Table of Contents

Introduction

Shared enterprise storage systems have been with us for decades now. Each new generation brings new features, better performance and scalability, and increased capacity density. Enterprise storage systems have been especially suitable workhorses for modern applications and virtualized workloads over the years. The ability to attach many hosts to shared storage systems has been critical for enabling transformative technologies such as server virtualization.

Despite their utility, these same systems have become an operational challenge as they proliferate. While organizations fell in love with their first shared storage system, the increased management overhead have made them hate their twentieth. Each system is an island unto itself, requiring dedicated cycles from an operations group to manage, update, and triage each system individually. As organizations scale their storage systems to meet their growing needs this quickly becomes problematic from a time, cost, uptime, and management perspective.

How can enterprises re-define this established paradigm? Should architects continue to accept the status quo and proceed down the path of undifferentiated heavy lifting? Nebulon recognizes this growing problem and has created a solution which addresses all the major pain points of managing a growing fleet of storage systems — **smart**Infrastructure.

This paper is intended for those in technology roles such as: Chief Technology Officers (CTOs), architects, developers, support engineers, and members of operations teams. After reading this paper, you will understand best practices on how to design, build and operate an environment with Nebulon **smart**Infrastructure. This paper includes product details, implementation methods as well as architectural patterns.

What is **smart**Infrastructure?

Nebulon's **smart**Infrastructure is server-embedded, infrastructure software delivered as-a-service, which offers the benefits of the public cloud on-premises, from core to edge for any application—containerized, virtualized or bare-metal.

Nebulon **smart**Infrastructure, provides self-service infrastructure provisioning, infrastructure management-as-a-service and enterprise-class shared and local block data services. The solution eliminates the need for external storage or a SAN and eliminates the density and workload restrictions imposed by HCI. IT organizations can benefit from a single-API across all of their on-prem enterprise data with two solutions which make up the Nebulon **smart**Infrastructure portfolio: the Nebulon **smart**Edge solution & the Nebulon **smart**Core solution.

Nebulon **smart**Infrastructure is made up of two components: The cloud control plane, nebulon ON, and the data plane made up of cloud-managed IoT endpoints embedded in your application server. The solution delivers easily accessible AIOps features, behind-the-scenes updates and powerful programmability at any scale.

Architecture Overview

The principal architectural components of **smart**Infrastructure are the Services Processing Unit (SPU), Nebulon nPods, and the Nebulon Cloud (Nebulon ON).

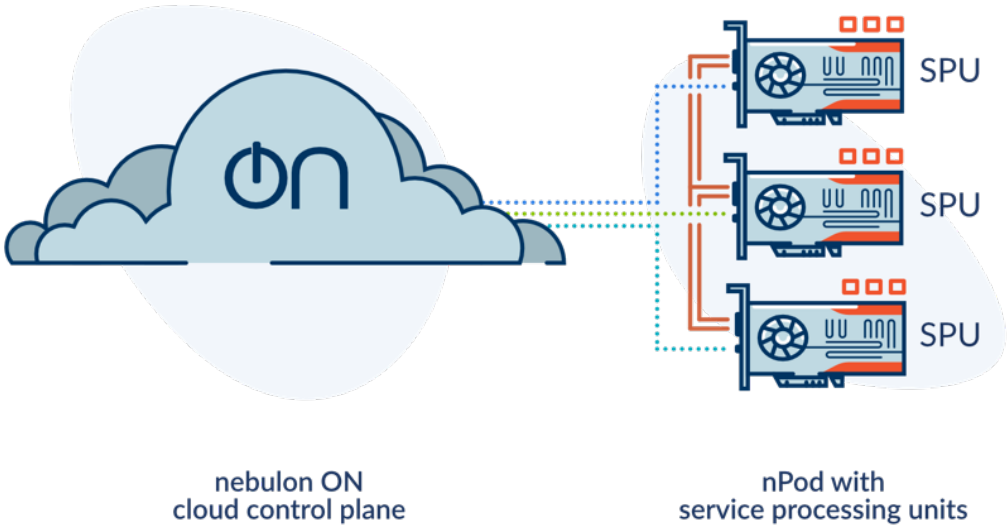


Figure: Nebulon ON is the cloud-based control plane which integrates with on-premises application servers.

Component	Description
SPU	The Services Processing Unit (SPU) is the physical data storage controller that is part of standard application servers in the data center. The SPU utilizes the internal storage of application servers to provide distributed access to logical data volumes.
nPod	A nPod is a collection of network-connected application servers with SPUs installed that form an application cluster. Together, the SPUs in a nPod serve shared or local storage to the servers in the application cluster, e.g. a hypervisor cluster, container platform, or clustered bare metal application.
Nebulon ON	Nebulon ON is the cloud control plane in **smart**Infrastructure. It is composed of a collection of microservices that collectively analyze, manage, and suggest optimizations for your infrastructure. It is accessible to users through a single management endpoint using a web browser for interactive management and through a comprehensive API for programmatic access and automation. It is

	here that users manage and monitor all aspects of their storage infrastructure, including SPUs and nPods.

The following sections describe each of the architectural components of the Nebulon platform in detail.

nebulon nPod

Nebulon nPods are the basic unit of deployment and configuration within Nebulon. nPods are created in Nebulon ON via the user interface, API or SDK by applying a Configuration Template to a group of SPU-equipped servers. A typical nPod deployment maps directly to a single application cluster. For example, one VMware cluster is built on a single nPod. A nPod provides application servers with necessary shared storage to operate; the minimum number of SPUs in a single nPod which requires mirrored data is three, otherwise you may provision single node nPods as well.

Depending on the server manufacturer and model, each SPU-equipped server can contain between 6 and 12, or 6 and 24 drives. Nebulon supports SAS connected drive types with capacities of 1TB, 2TB or 4TB. Application server configurations containing a Nebulon SPU with no physical drives is a supported design pattern. This enables an application server to access shared nPod storage resources without providing additional storage capacity. When a server contains only a SPU and no physical drives, and only accesses storage over the nPod data network is operating in passthrough mode.

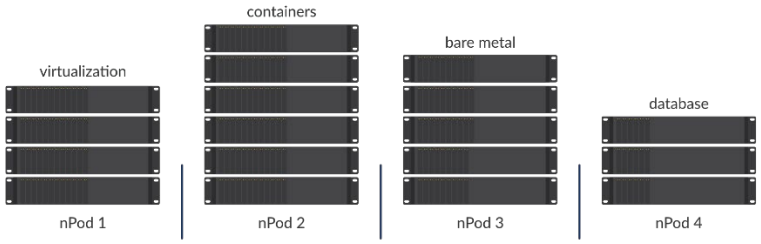


Figure: nPods are the basic unit of deployment and configuration

For the purposes of providing management isolation, nPods can be logically organized into nPod groups. This allows an application administrator to manage all nPods dedicated without giving wide-ranging permissions to all the organization's nPods. This can also allow for arbitrary groupings of nPods and provides control based on business needs such as geographic location, line of business (LoB), or application use.

The following figure illustrates an example of how nPod groups can be used across an enterprise’s organizations.

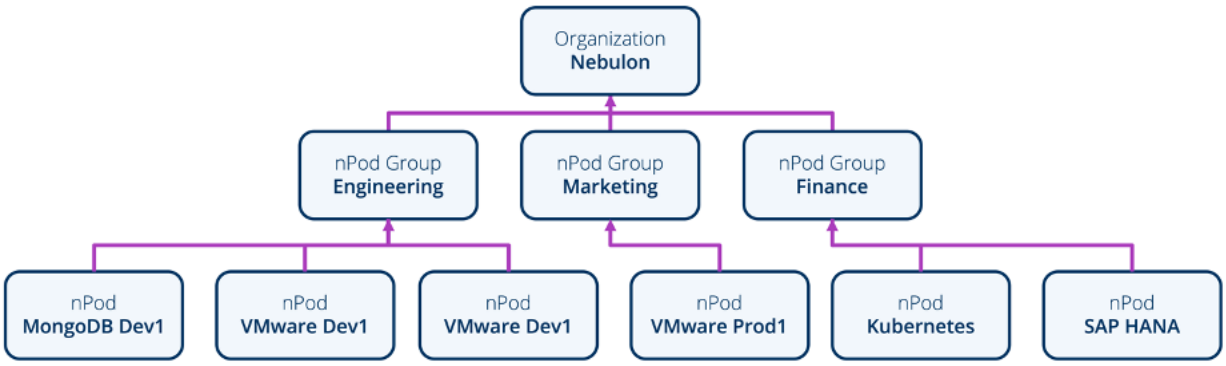


Figure: nPods can be logically organized into nPod groups which align to lines of business

Nebulon Medusa 100 Services Processing Unit

The Nebulon Medusa 100 Services Processing Unit (SPU) is the physical data storage controller which is integrated by the server vendor and included as part of application servers within the data center. The Nebulon Medusa 100 SPU is a full-length, full-height, double-wide PCIe 3.0 compliant card which utilizes the internal storage of application servers to provide shared and local access to storage volumes.

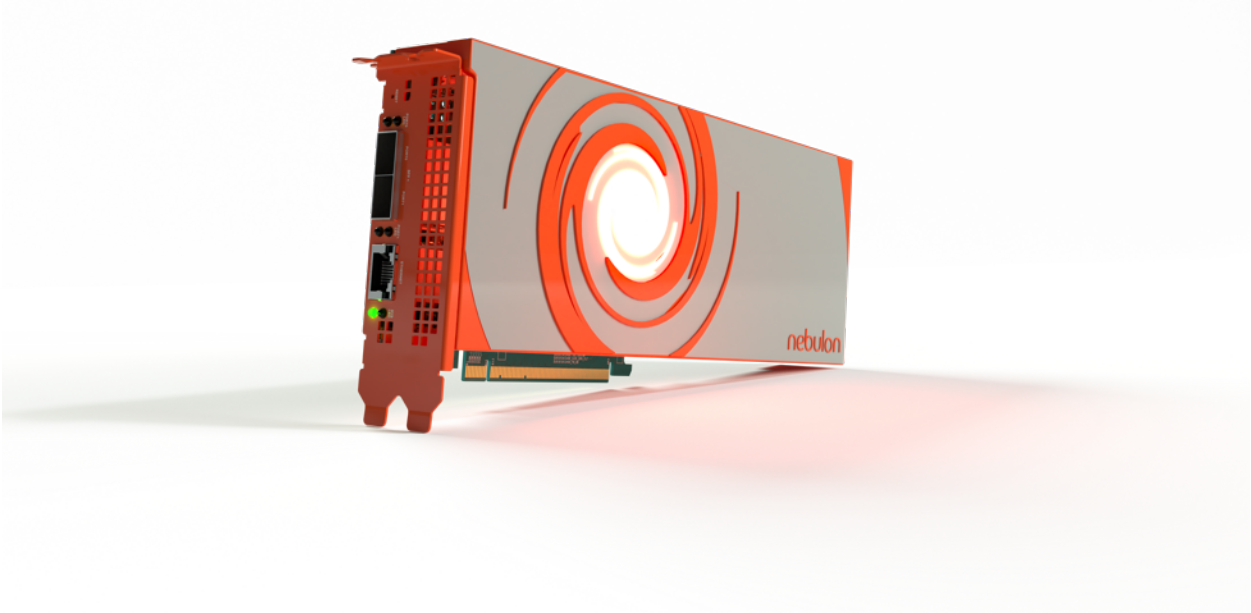


Figure : Nebulon Medusa 100 SPU PCIe card

The Nebulon Medusa 100 SPU runs nebOS and features enterprise data storage services, including automatic storage failover, data reduction (compression, deduplication, compaction), thin provisioning, snapshots, encryption, and more. The Nebulon Medusa 100 SPU and nebOS are equivalent to a traditional storage controller. These enterprise data services are automatically activated and tuned for application-optimized volumes.

Multiple SPUs are grouped together for redundancy and collectively serve shared or local storage for an application environment. This collection of SPUs is called an nPod. Apart from the ability to tailor data

layouts for specific application workloads, this data sharing domain ensures that only application servers with SPUs in the same nPod (tenant) can access application data.

Moreover, depending on the application and Nebulon Configuration Template, storage access can be further restricted to just individual application servers in a nPod. As an example, for cloud-native databases, data can be compartmentalized for individual servers in the same nPod. This is made possible as user data in **smart**Infrastructure is only accessible through the SPU in a server. This also means there is no shared Fibre Channel (FC) Storage Area Network (SAN) that requires careful masking of volumes to individual hosts.

Nebulon Hardware

Nebulon Medusa 100 SPU Security

The Nebulon SPU is equipped with a cryptographic co-processor with ultra-secure hardware-based cryptographic key storage and cryptographic countermeasures which strengthens protection against potential backdoors linked to software weaknesses. This provides the foundation for secure authentication, encryption key management, and secure boot in nebOS.

Nebulon SPUs use secure boot capabilities to ensure that the lowest levels of nebOS are not tampered with and that only trusted nebOS firmware from Nebulon loads at startup. In nebOS, security begins in immutable code that is part of its CPU and laid down during card fabrication. This ensures the SPU only boots using firmware which is authentic and properly signed by Nebulon.

Moreover, all SPUs include a dedicated AES hardware engine to power line-speed encryption as data is accessed. As **smart**Infrastructure uses a scale-out storage architecture, each SPU in a nPod participates in data encryption and decryption using their own AES hardware engine. The SPU in the application server encrypts data as it is written. Similarly, the SPU in the application server decrypts data as it is read. This ensures that data is protected as it enters or leaves the SPU without exposing unencrypted data to the network. The scale-out architecture of nPods ensures encryption is well balanced between all hardware engines in all SPUs.

Nebulon SPU Networking

Each Nebulon SPU is equipped with three (3) dedicated network ports, 1 x 1 Gbps control port (management port) and 2 x 10 or 25 Gbps data ports (auto negotiated). These network ports are not visible or usable by the server operating system for network communication. The below matrix details the type of traffic which traverses each port.

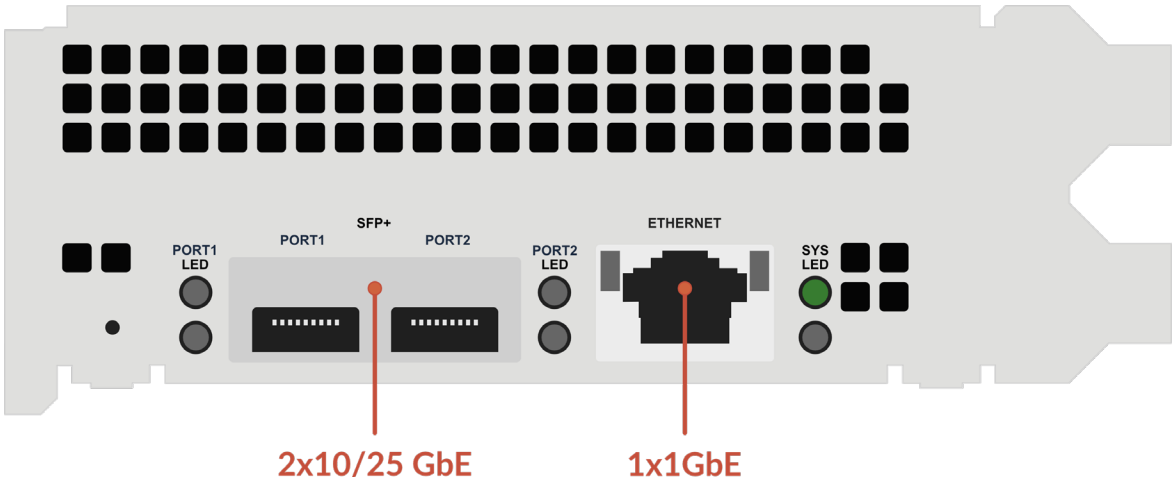


Figure: Nebulon Medusa 100 SPU is equipped with 2 data network interfaces and 1 control interface

Port	Count	Description
1GbE RJ45	1	SPU control network: A routed outbound network which connects the SPU to Nebulon ON
10GbE SFP+ or 25GbE SFP28	2	SPU data net: A private, network that is used by SPUs to transfer data for data services within a single nPod

Table : SPU networking interface details

Physical network connectivity

Nebulon supports several network configurations for the ports on the SPU as shown in the following table. The preferred configuration leverages two networks, one for control and one for data. In this configuration, the control ports can be placed into a tightly controlled network segment that allows outbound communication to Nebulon ON while the data ports are placed in a private network.

When connecting the SPU to your on-premises network environment, follow these networking guidelines.

Ports	Network	Guidelines for uplink ports on the switch
1GbE RJ45 Port	Control network	• Configure as access port • Allow connectivity to Nebulon ON over TCP on port 443. Nebulon ON resolves to the following IPv4 addresses: 3.20.202.129, 18.220.123.25, 18.223.202.114, 3.20.214.24, 3.133.173.31, 3.14.207.106 • Spanning tree should be disabled for these ports (PortFast or equivalent)

		• DHCP should be available on this port for the easiest startup configuration
10GbE SFP+ or 25GbE SFP28 Ports	Data network	• Configured as access ports • Port speed hard set to either 10GbE or 25GbE depending on the switch and DAC cable capability. Fiber connections are not qualified • Both interfaces configured in a LACP bond. Set LACP mode 4 (802.3ad compatible – mode active) and LACP transmit rate to "fast" •

Table : SPU networking options and configuration details

Nebulon recommends implementing LACP (in fast mode) for data ports. This reduces the number of IP addresses required, simplifies network redundancy, and optimizes network bandwidth.

Option	Control Port	Data 0 Port	Data 1 Port
1*	Net A	Net B (LACP)	
2	Net A	Net B	Net C

** denotes the preferred configuration.*

The following table describes some of the pros and cons for each port VLAN configuration option listed above.

Option	Pros	Cons
1*	• Separation of control and data traffic • Single IP address for data • Fast failover for data ports • Improved bandwidth	• Requires LACP configuration on switch
2	• Separation of control and data traffic • No LACP switch configuration	• Requires double the number of IP addresses vs LACP • Requires double the number of VLANs

** denotes the preferred configuration.*

The following diagram (Figure) illustrates the connections required when connecting application servers and their SPUs to the physical network infrastructure. Note that the VPC/MLAG configuration between the 10/25GbE switches is required if implementing LACP on the Medusa data ports.

Note: Nebulon SPUs do not expose their network ports to the host operating system on the application server and they are only used by the SPUs. The application server will require their own network interfaces, which have been omitted in this illustration for simplicity. Customers can choose to run application networking and SPU networking on the same switches, provided they are VLAN separated. If VLANs are in use, ensure that the VLAN is untagged at the SPU access port.

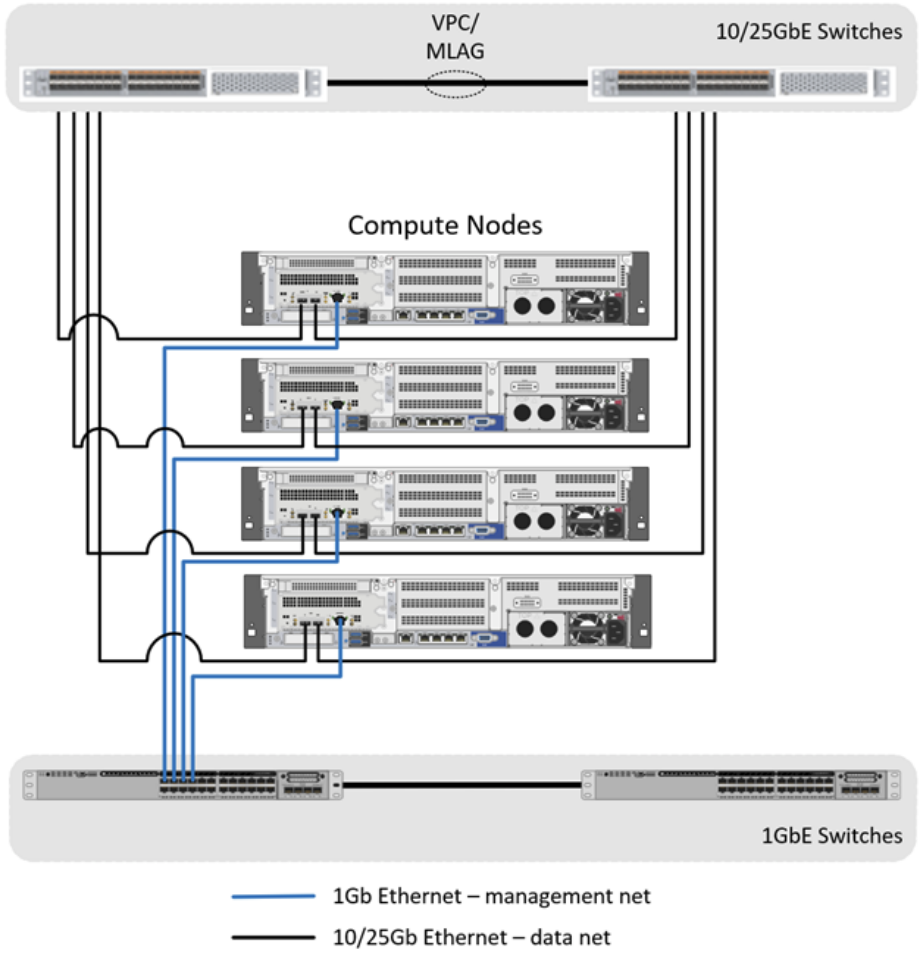


Figure : Connections required when connecting application servers and their SPUs to the physical network infrastructure.

Control (Management) Network – Control Plane

The 1GbE control network allows the Nebulon Services Processing Unit (SPU) to communicate with Nebulon ON. To communicate with Nebulon ON, the control port must have a routable IP address which has Internet access available over TCP port 443. Network administrators should ensure that the following IP addresses are whitelisted in their firewall configuration:

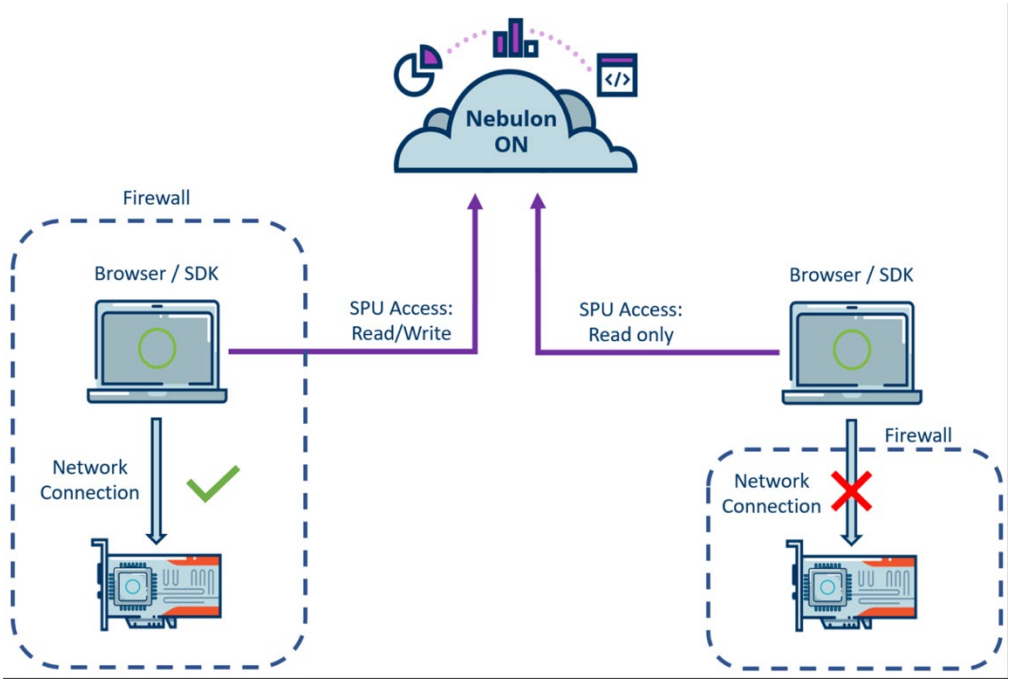
IP Address	Port	Protocol
3.14.207.106	443	TCP
3.20.202.129	443	TCP
3.20.214.24	443	TCP
3.133.173.31	443	TCP
18.220.123.25	443	TCP
18.223.202.114	443	TCP

Table : Nebulon ON IP address table

For the easiest configuration, DHCP may be used to configure the control port with a valid IPv4 address, NTP server, and DNS server that can resolve public domain names.

Alternatively, users can use the server's UEFI user interface to configure the SPUs control network manually. This includes configuration of a network proxy and static IP addresses.

The 1GbE control-plane network needs to be accessible by users administrating **smart**Infrastructure. Only



management tools, i.e. web browser, or SDKs, that can establish a TCP connection on port 443 with the control IP of an SPU will be able to configure the SPU. Refer to the secure command execution section in this paper for more details. The ability to manage or access Nebulon ON resources like users, metrics, and reporting does not depend on local SPU access.

Figure : Connection from an on-premises environment to Nebulon ON

Control (Management) Network Configuration

The 1GbE control network allows the Nebulon Services Processing Unit (SPU) to communicate with Nebulon ON. Administrators may configure the static IP address of the 1GbE control network via your server UEFI. The look and feel of the UEFI will vary dependent upon server vendor however the steps will remain the same.

Administrators may follow these steps to configure a static IP address for the 1GbE control network.

1. Enter the UEFI of the Nebulon SPU equipped server. Navigate to 'Nebulon Configuration':

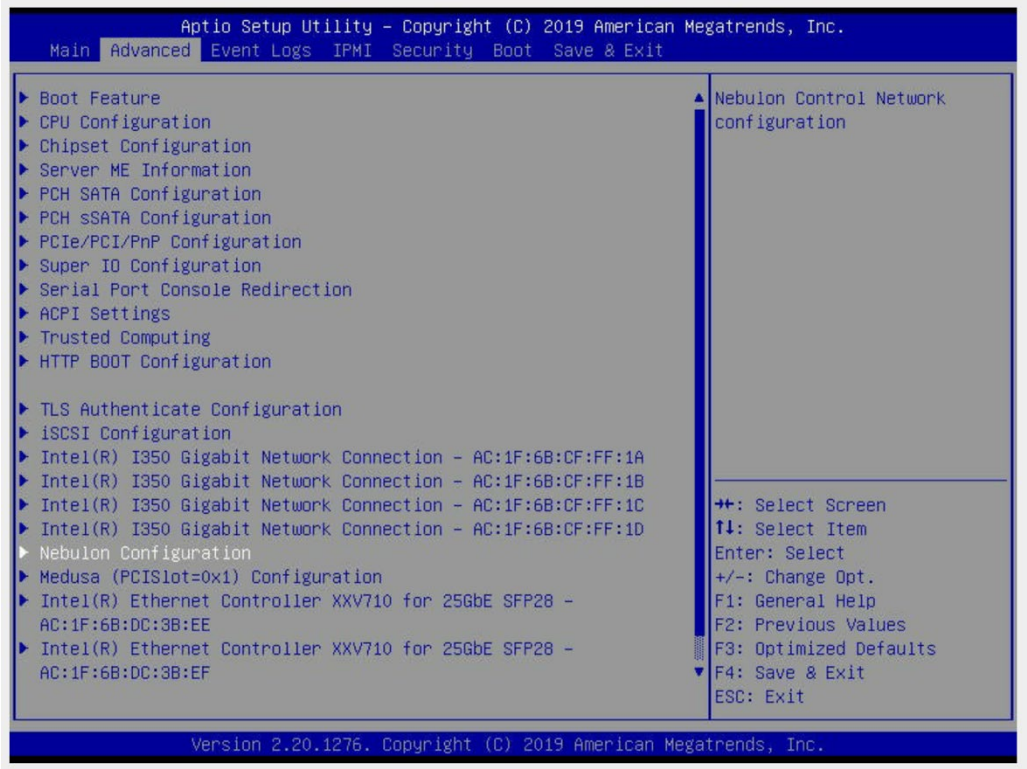


Figure : Configure Nebulon control IP address via server UEFI

Control_UEFI_1.png

2. The control interface supports both IPv4 and IPv6. Choose the appropriate protocol:

Figure : Nebulon control interface supports IPv4 and IPv6 protocols

Control_UEFI_2.png

3. By default, the 1GbE Nebulon control interface is set to dynamic IP address assignment using DHCP. You may also statically assign an IP address to the control interface:

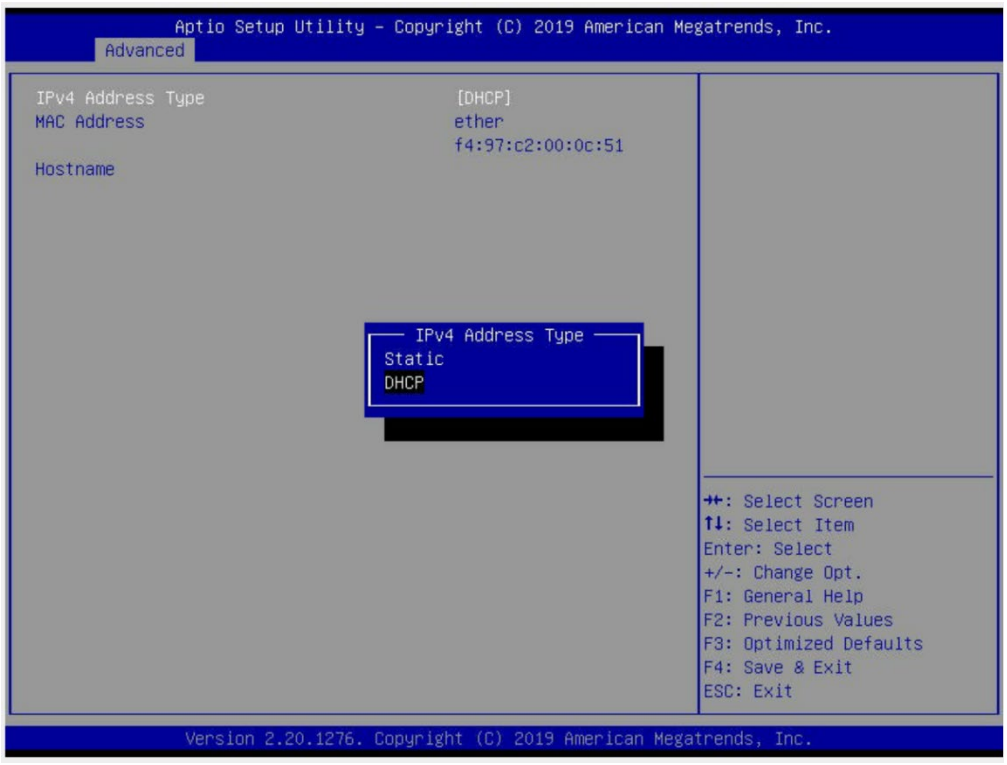


Figure : Configure static or dynamic IP address allocation on the control interface

Control_UEFI_3.png

4. To assign a static IP address enter the required and appropriate IP address, subnet mask, default gateway, MTU size and DNS:

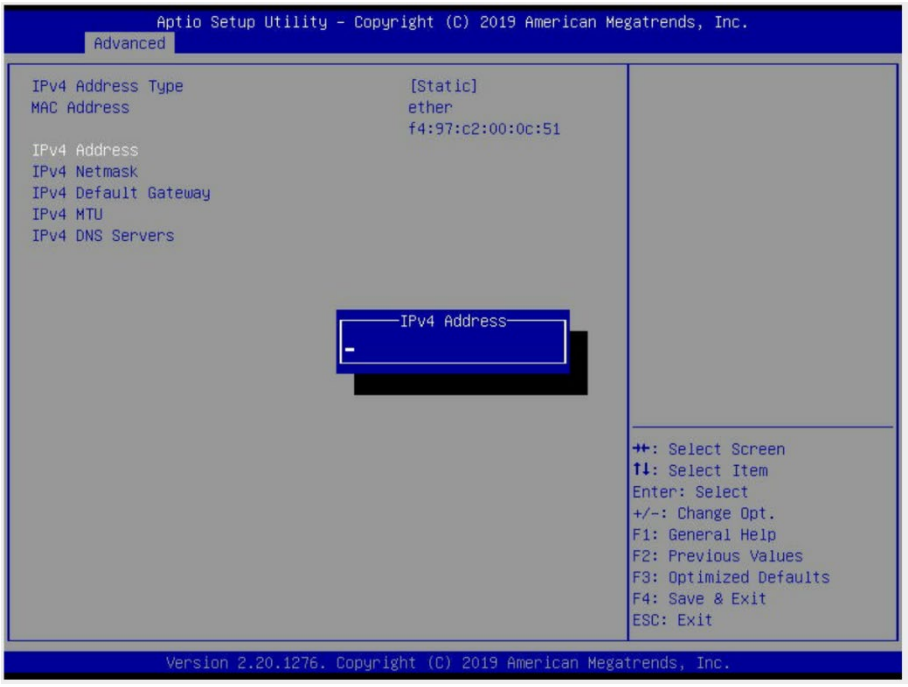


Figure : Type in a IPv4 address when using a statically assigned IP address

Control_UEFI_4.png

5. Within the server UEFI, Administrators may easily view connectivity to nebulon ON. This is useful for initial interface (server and switch) configuration and checking connectivity status.

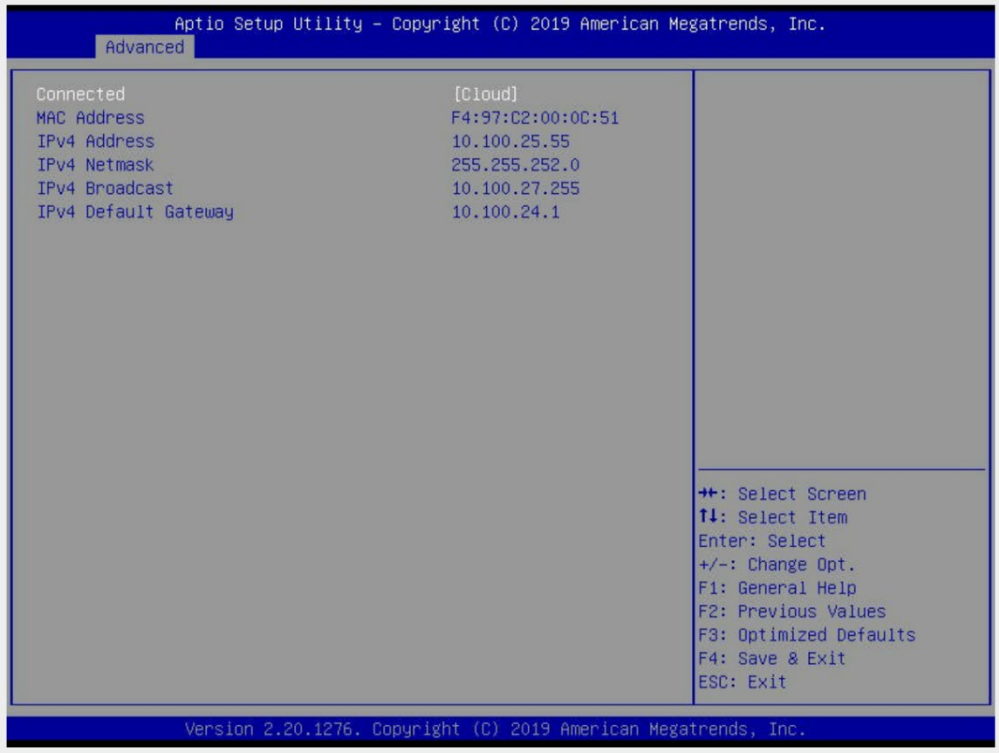


Figure : Users may verify cloud connectivity via UEFI

Control_UEFI_5.png

Data Network – Data Plane

The Nebulon Medusa 100 Services Processing Unit (SPU) contains redundant data network ports which are used internally by SPUs to communicate with other SPUs in the same nPod. No host network traffic from application server(s), applications, virtual machines, or containers traverse these links.

The SPU data ports are configured during the nPod creation process. Users may create an nPod for their applications to run on via the Nebulon ON web user interface or via SDK. The SPU data ports support both DHCP and static IP address assignment. Nebulon recommends that customers statically assign IP addresses to their data interfaces.

LACP is not required for the data network, except when using a single layer 2 segment for both data ports. LACP is recommended to customers for a resilient data network and to save on IP addresses. If LACP is used, the switch should be configured for Etherchannel / Virtual Port Channel (VPC)/MLAG – terminology will vary depending on your switch vendor. LACP includes TCP port numbers in the hash which will improve distribution of traffic across the links. LACP should also be in “fast” mode to avoid possible I/O errors during link failure events.

The Nebulon Medusa 100 Services Processing Unit supports two modes of link aggregation configuration:

1.) **None** – With bond mode none Nebulon SPU data interfaces operate independent of each other and no additional switch configuration is required. With none, administrators configure a single IP address per interface and nebOS will internally load balance in a round-robin manner across the interfaces. In the event of link failure, nebOS will retry communication with SPUs in the nPod by re-transmitting the packets over the alternate link. Bond mode none supports both static IP address assignment and assignment via DHCP. For this mode to operate reliably, the two networks must be separate Layer 2 networks.
2.) **Link Aggregation Control Protocol (LACP) 802.3ad** – LACP is an IEEE specification in which allows the bundling of physical network switch ports and physical network interfaces to form a single channel. In the event of disruption, LACP allows for seamless failover between the interfaces in the bond. Administrators should configure the network switch ports as access ports and configure LACP rate fast. LACP requires a single IP address for a pair of Nebulon SPU data interfaces. When choosing bond mode LACP, ONLY static IP address assignment is permitted. With bond mode LACP, DHCP is NOT supported.

Nebulon SPU Storage Domain

The storage SoC (system on a chip) is comprised of a 3GHz 8 core 64-bit ARM CPU with 32GB DDR4 2400 MHz memory, integrated offload engines, and dual 25Gbps Ethernet ports which provide data network connectivity. The SoC boots off embedded eMMC media while a M.2 NVMe SSD provides the persistence layer for the data cache in the event of complete server power loss. With Nebulon, storage processing (power, CPU, memory, and network connections) is done on the SPU, which is on a separate, battery backed power domain from the application server. This unique configuration allows the Nebulon SPU to operate and have data persisted across server reboots and power cycles.

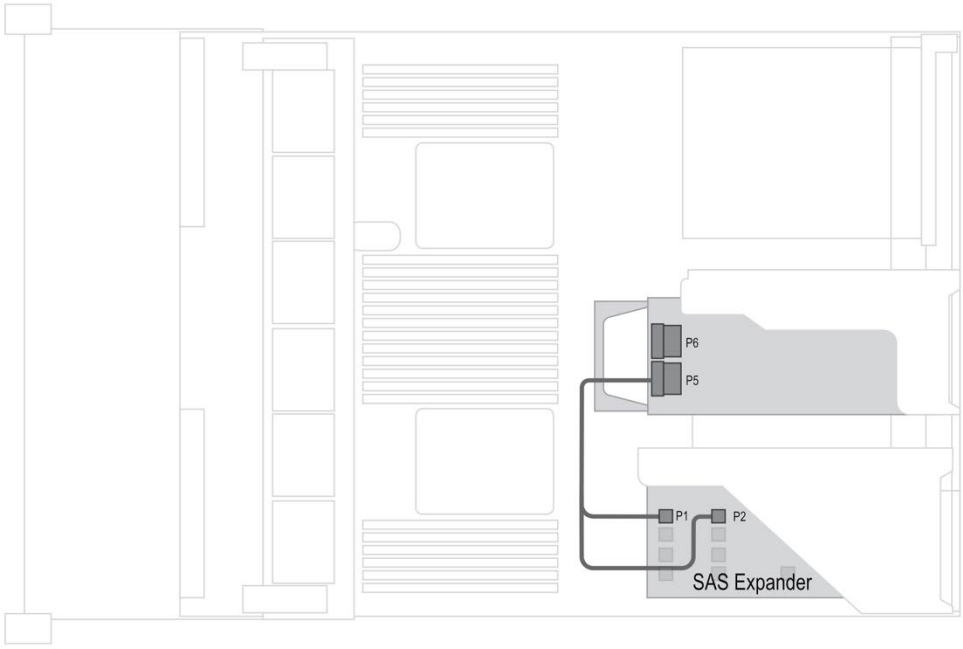
The Nebulon SPU contains a SAS/SATA/NVMe (Tri-Mode) I/O controller (IOC) which is connected to the SoC's PCI bus. The IOC provides connectivity to both the storage media within the application server and to the Nebulon SPU Host I/O controller. There are two (2) internal x 8 SlimSAS connectors which cable either directly to the server backplane, or through a SAS expander card, providing the IOC access to the server's storage. This configuration replaces an application server's typical onboard storage controller.

The Nebulon SPU manages all aspects of its attached SSDs, including I/O, health monitoring, LED control and firmware upgrades. Standard 2.5" (Small Form Factor - SFF) SATA or SAS-based server SSDs are used with Nebulon SPUs. For detailed compatibility information and an updated list of compatible SSDs and capabilities, please review the latest compatibility matrix from your server vendor.

Lastly, all SPU-to-SPU communication including cluster membership, heartbeat and volume mirroring is done via dedicated dual 25GbE data network connections which terminate at a top-of-rack switch. Additionally, a 1GbE network connection is required for cloud-based control and management via Nebulon ON. The control connection should also terminate at a top-of-rack switch.

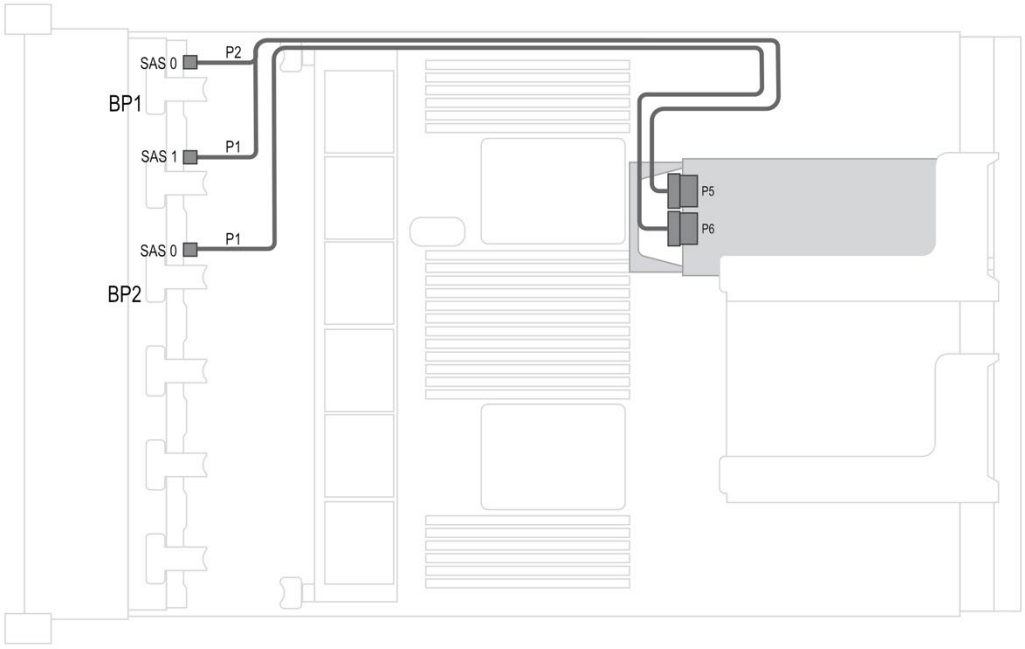
Storage Media Cabling

Storage Media Cabling HPE ProLiant DL380 Gen10



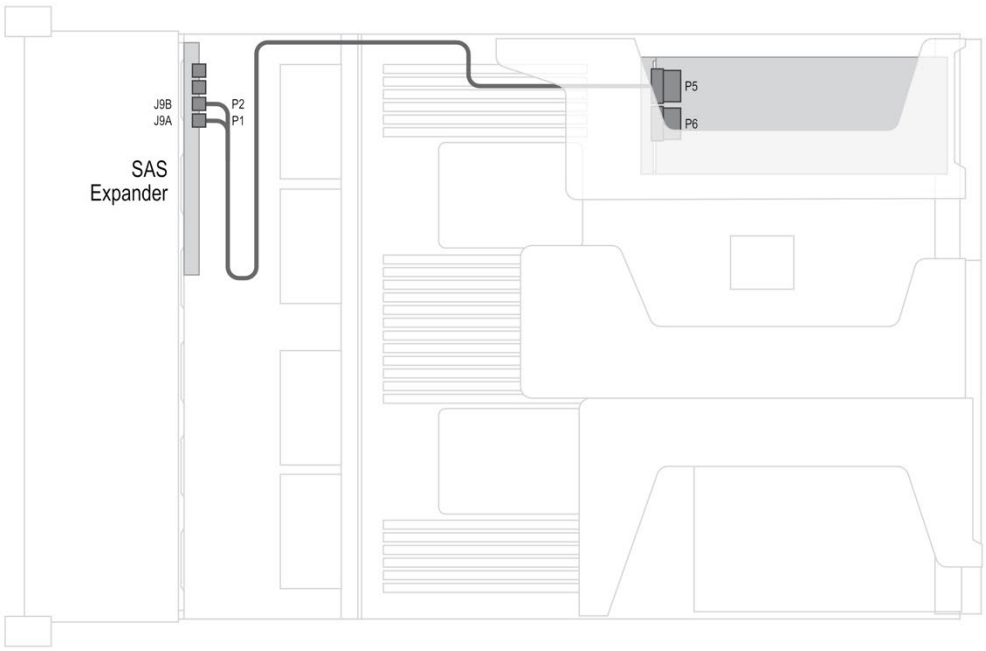
SAS Expander Port	Cable Port	Medusa 100 SPU Port
Port 1 Port 2	Cable P1 Cable P2	Port 5

Storage Media Cabling Lenovo ThinkSystem SR650



Backplane Port	Cable Port	Medusa 100 SPU Port
BP1 SAS0, BP1 SAS1	Cable P2 Cable P1	Port 5
BP2 SAS0, BP2 SAS1 - not connected	Cable P1 N/A	Port 6

Storage Media Cabling Supermicro Computer SYS-2029U-E1CRT



SAS Expander Port	Cable Port	Medusa 100 SPU Port
Port A - J9A Port A - J9B	Cable P1 Cable P2	Port 5

Nebulon Host Domain

The Nebulon SPU appears to the server as a standard host bus adapter (HBA) and the application server operating system will automatically discover the Nebulon Medusa 100 SPU as a “Broadcom SAS3408” controller. The Nebulon SPU leverages standard, out-of-the-box drivers which are included with popular operating system (OS) distributions, including VMware® vSphere®, Microsoft® Windows® Server, and RedHat® Enterprise Linux.

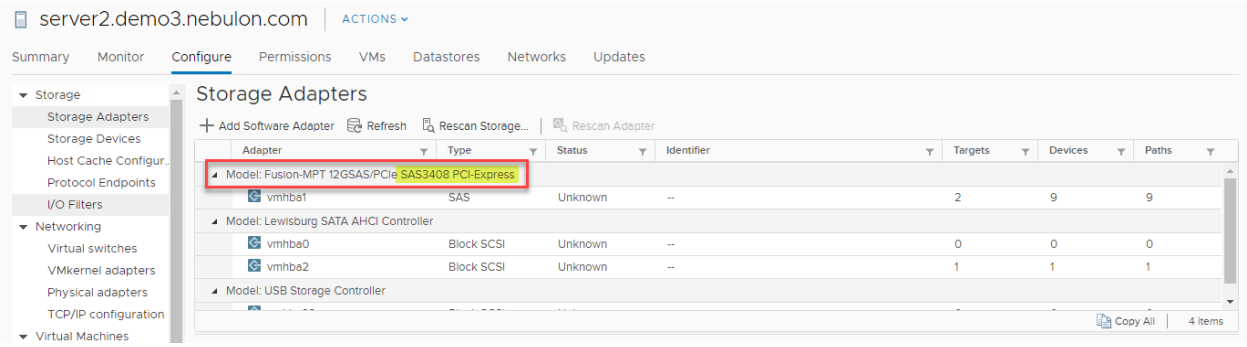


Figure : Nebulon SPU appears to the application server as a vmhba within vCenter

vmwareadapters.png

Nebulon SPU LED Indicators and Troubleshooting

Two System LEDs are provided on the SPU displaying the status of the SPU's health and activity. Located to the right of the control Ethernet port, the System LEDs (top to bottom, LED 1 and LED 2) can be illuminated either green or yellow.

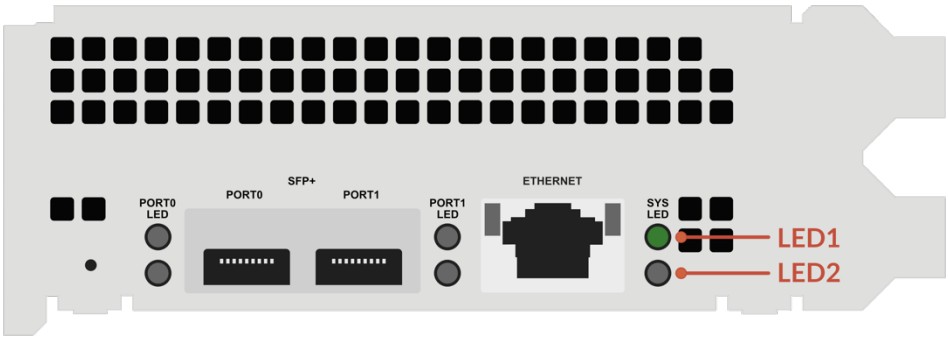


Figure : Rear view of the SPU and LED indicators

The LEDs and their behaviors are described in the following table.

SYS LED 1	SYS LED 2	Description
Off	Off	No power. Check power connections and supply.
Blinking green at 1 Hz	Blinking yellow at 1 Hz	NebOS is booting.
Blinking green at 1 Hz	Off	NebOS is up.
Blinking green at 2 Hz	Off	NebOS control IP address is assigned.
Solid green	Off	NebOS is connected to Nebulon ON.
Alternating yellow and green at 1 Hz	Alternating yellow and green at 1 Hz	NebOS is in locate mode. This can be triggered using the GUI or the Nebulon automation SDKs to easily locate a SPU in your datacenter.
Any color/any state	Solid yellow	Failed. Please contact Nebulon Support.

nebOS

nebOS is Nebulon's lightweight storage operating system which runs on the Nebulon Services Processing Unit (SPU). The function of this storage operating system is to provide resilient enterprise data storage services for application clusters running on commodity application servers. nebOS is also responsible for establishing and maintaining a secure connection to the cloud-based control plane, Nebulon ON.

Unlike traditional storage controllers, nebOS is not cluttered with a complex, heavyweight management stack requiring regular maintenance activities. Instead, this traditionally overhead-heavy management stack is elevated to Nebulon ON and delivered as a service in which you subscribe to. The cloud-based control plane allows for simplified fleet management at scale, faster firmware updates and overall minimizes day 2 administrative overhead. As a result, applications will achieve increased reliability and decreased workload interruptions.

The core function of nebOS is storage I/O processing. This includes storage virtualization, erasure coding for data availability, data mirroring, snapshots, encryption, compression, and deduplication. Management aspects of these data services are provided through Nebulon ON, where Configuration Templates which relate to storage layout, presentation, provisioning, volume distribution, and snapshot scheduling are created and enforced by nebOS.

The following section aims to provide technical details regarding nebOS as well as many of the native data services provided by **smart**Infrastructure.

Storage High Availability & Data I/O Path

Volume protection, reliability and high availability is configured at the nPod level and done so by employing erasure coding and optional data mirroring. The server's physical drives are organized into Erasure Coding Sets which provides data redundancy by utilizing Reed Solomon erasure coding algorithms. **smart**Infrastructure requires a minimum of 6 drives within a server while the maximum supported drive configuration is either 12 or 24 drives [1].

[1] *Note: The maximum number of drives possible in a server chassis is determined by the server OEM.*

Within **smart**Infrastructure, Erasure coding is implemented over segments which reside on multiple drives. Unused segments are used when data reconstruction occurs. nebOS stores information about the system including information regarding the physical drives, their UUIDs and server configuration metadata. This ensures that if an SPU failure occurs the device can be replaced without the worry of any data loss on the drives in the system. Secondly, this allows the decoupling of the physical drive from an individual drive slot. In the event of a SPU failure a field replaceable unit (FRU) will be dispatched using your server vendor replacement process, while the remaining drives in the system maintain application data and current system state. The entity in which data is stored on and subsequently runs applications, is a volume. Data blocks written by a host are initially captured in a battery backed RAM module on the SPU, compressed using LZ4 compression algorithms, fingerprinted, hashed (deduplicated) and encrypted. For local volumes, while the data is in the SPU RAM, the acknowledgement is sent back to the application, the I/Os are de-staged in 6MB chunks from RAM to disk. For mirrored volumes, the I/O is initially captured in local SPU RAM and mirrored to a backup server in the nPod. Once the I/O is received by the backup SPU RAM and I/O acknowledgement is provided back to the application.

For local volumes, where the server running the application also maintains the only copy of the volume data, all read and write I/Os are served locally. When read I/Os occur the required data blocks reside in SPU RAM and the I/Os are served directly from RAM without looking to the drives in the system. In scenarios where data does not reside in SPU RAM, read I/Os require drive access.

As application data is written, read, and deleted from a volume, some strips and segments will inevitably become garbage and require reclamation. nebOS employs a background garbage collection daemon which regularly scans for changes. Garbage collection is responsible for maintaining storage extents, cleaning up garbage space as well as further compression. Garbage collection will run continuously, however, the service concedes priority to write intensive application data operations.

A SPU can operate in passthrough mode when it provides access to a volume for its server which is not stored on that SPU. All I/O requests pass through that SPU and are directed to the appropriate owner SPU for that volume. All I/Os will route to the volume owner as this is the authoritative truth for all volumes.

nPod Redundancy and Availability

To protect from temporary network connectivity issues to Nebulon ON and to enable enterprise storage availability, SPUs are equipped with all capabilities and intelligence to continue providing data services and maintain defined storage policies in various failure and cloud disconnected scenarios.

Connecting the server's solid-state drives (SSDs) directly to the SPU creates two separate operating domains: one for the application server and one for the Nebulon SPU. As the server has power, the SPU storage resources are available to the servers in the nPod. This provides several advantages compared to Software-Defined Storage offerings which require software on the host operating system:

- SPU storage resources are managed independently of the host OS. This allows configuration of storage resources when there is no host OS installed and the use of Nebulon volumes for the OS boot device.
- The host operating system can be rebooted or crash without affecting availability, performance, or data redundancy. This means maintenance complexity is tremendously reduced as data does not require special care prior to OS reboot. For example, rebooting a VMware host for maintenance does not require draining the storage from that node.
- Shared volumes can be advertised to servers in a nPod that do not contribute any storage capacity to the nPod by using a SPU with no drives attached. This can be desirable for backup hosts, or to independently scale compute resources in a nPod.

Apart from being a separate failure domain, the SPU frees up application server resources (CPU, memory, networking) as it shoulders all the work necessary to store, optimize, and protect data. The resource savings on the application server can be significant compared to Hyper-Converged infrastructure (HCI) or Software-Defined Storage (SDS) solutions that make use of the servers' CPU, server memory, and server network resources for I/O processing, data reduction, and encryption. With Nebulon **smart**Infrastructure there are no 3rd party drivers, VIBs or controller virtual machines required and thus application servers' applications benefit from all compute and networking resources of the server. Nebulon allows customers to obtain higher density and lower licensing costs.

Garbage Collection & Deduplication

As new application data is written to a nPod volume, it is first captured in NVRAM (cache) and data is subsequently flushed. When flushed, data is always first written to new data segments and Garbage Collection (GC) will consolidate and reclaim the old location.

nebOS utilizes a relocate on write process for space allocation, as such, any re-write of data can create garbage. When an overwrite occurs and the previous version becomes garbage, unused space is now present. . A fundamental design element, as it relates to space reclamation, is to give priority to write performance. The space reclamation domain is conducted and maintained locally at a SPU level.

Garbage Collection is responsible for generating space consumption reporting, which is delivered via Nebulon ON, Space Reclamation processes and Offline Deduplication. To guarantee the highest levels of performance and to not negatively affect application I/O, as data is overwritten or updated, inline deduplication will automatically throttle based on available resources. Deduplication will then occur as part of the offline deduplication process.

nebOS deduplication normalizes I/Os to 8 KB. For example, an application generates 4 KB I/O – nebOS will deduplicate the single 4 KB I/O. Any I/O smaller than 8 KB is deduplicated as the application has generated the I/O. Whereas if an application generates a 32 KB I/O this will create 4 normalized 8 KB I/Os in which are deduplicated.

Data Encryption

Applications which run on Nebulon benefit from always-on encryption of data at rest as well as data in flight. A secondary benefit to applications is that the encryption is achieved with no performance degradation or loss of space savings – deduplication and compression.

Data at rest is always encrypted using AES 128-bit encryption algorithm with cipher block chaining (CBC) to ensure that customer and application data is always secure – even if drives are physically removed from the system(s). Nebulon has designed a streamlined mechanism which eliminates the need for users to manage their own encryption keys. Encryption keys are always stored within a customer's on-premises nPod. At no point in time are encryption keys shared with Nebulon ON.

Data is encrypted at rest using Data Encryption Key (DEK) and a Key Encryption Keys (KEK). The KEK is used to encrypt the keys in a nPod and is automatically generated at the time a nPod is created and is securely transferred between SPUs using a mutual TLS 1.3 connection and stored in the hardware key vault. The KEK is unable to be read, it can only be overwritten. When an SPU is added or replaced, a new KEK is generated and distributed amongst the other SPUs in the nPod. At this point the previous KEK is used to decrypt any previously encrypted keys and re-encrypt using the new KEK. The previous KEK is overwritten.

Data written to drives in a nPod is encrypted using the Data Encryption Key. Like the KEK, the DEK is generated at the time a nPod is created and securely transferred amongst the SPUs in the nPod. The DEK is encrypted using the cryptographic co-processor on the SPU and the KEK before it is persisted to the physical drive media. The DEK is persisted across many drives in the nPod, making it safe from individual drive failures.

Data in flight is always encrypted as well and user data stored in a nPod is always encrypted at the time the application server writes it to its SPU. Data is only unencrypted when read back by the application server. Decryption of the data is handled by the SPU of the application server issuing the I/O request.

Nebulon makes use of two distinct methods of encrypting communication between components and distinguishes between control data and user data. All control or management communication between Nebulon components is encrypted using secure TLS 1.3 encryption. The SPU cryptographic co-processor is used to securely store network certificates for SPUs. When SPUs communicate with other SPUs in the nPod or for network communication with Nebulon ON, these certificates are used to authenticate them reliably.

For more detailed information about Secure Management and the Security Triangle review the Nebulon Security White Paper on www.nebulon.com.

Nebulon Volume

A volume is associated with a nPod and is the application's visible entity in which data is stored. The SPU within a nPod utilizes local server-based storage to implement volumes which are exported and presented as block storage LUNs to application server(s) within a nPod. Volumes within Nebulon can be used as storage for application data and volumes may also be used as bare metal server operating system (OS) boot device(s). Users and application owners may automatically provision volume(s) based on pre-determined sizes and snapshot schedules at the time of the nPod creation by leveraging Nebulon Configuration Templates. Alternatively, users may create standalone volumes via the Nebulon ON graphical user-interface, API or SDK. Volumes may be exported to any number of hosts within the nPod. Additionally, for data high availability, volumes may be mirrored to a 'backup' server within a nPod. SPUs can have the following roles for any volume:

- **Owner** – A SPU with in a nPod which owns a volume.
- **Backup** – A SPU that stores a mirrored copy of volume data, installed in a separate server of the nPod, that does not contain the owner SPU.
- **Passthrough** – A SPU that provides host access to a volume that is not stored locally on the SPU but owned by another SPU in the same nPod. Example 1: Application A runs on Server A however, the volume is owned by Server B. The SPU in Server A is operating in passthrough mode for this volume. Example 2: A server does not contain any disk and is still able to participate in the nPod - the local SPU is passthrough and passes all I/O requests to the volume owner.

The cloud-based control plane, Nebulon ON, is responsible for determining optimal placement of the volume owner and volume mirror (backup). Volume placement is mainly determined based on 2 criteria:

1.) Using the two (2) SPUs within the nPod with the highest amount of available storage capacity.

2.) Nebulon ON will sort the list of SPUs in the nPod based on the number of currently owned volumes. The SPU with the lowest set of owned volumes will become the volume owner and the SPU with the second lowest number of volumes will become the volume backup.

Volume Access Methods

Volumes within a nPod can be presented to a single application server or to many application servers as either mirrored or unmirrored storage. Enterprise architects should evaluate storage requirements and considerations based upon the application which is being deployed on **smart**Infrastructure. Application servers are provided flexibility with **smart**Infrastructure and have two access methods in which an application data can be written and read to disk.

- Single-Host Access
- Multi-Host Access

<u>Single-Host Access</u>

Single-host access volumes create a 1:1:1 relationship between an application, SPU, and volume. In this scenario, the application on a single server has exclusive access to a volume that is stored on any of the SPUs in the nPod. No other application server within the nPod has awareness of the volume. The data on the volume may be, but is typically not, mirrored to backup the SPU within the nPod and all data protection is provided locally via erasure coding where parity is used to re-calculate blocks of data in the event of failure. As with all application deployments, users should carefully consider storage access methods prior to creation of their Nebulon Configuration Template. An example scenario where single host access is applicable would be MongoDB. A second use case would be Nebulon boot volumes as only

the boot volume owning SPU can mount the boot device. Users should also carefully consider their backup and disaster recovery requirements to ensure that the necessary recovery time and recovery point objectives (RTO / RPO) are met.

Multi-Host Access

Multiple application servers within a nPod can access a Nebulon volume simultaneously; this is referred to as multi-host access. With multi-host access, several application servers can both read and write to the volume. For example, a VMware VMFS datastore where multiple hosts within the vSphere Cluster can simultaneously mount the datastore as well as run live virtual machines. A second example scenario would be a clustered database environment where failover from one cluster node to another is done at the application level and the volume requires multi-host access. The expectation is that volume access coordination is done on the application layer through clustering or a clustered filesystem, e.g. VMFS in VMware.

Volume Types

A volume within Nebulon is associated with a nPod and exported to the application servers in the nPod. A nPod volume may be used to store application data as well as the OS boot device. There are 3 types of volumes within Nebulon:

1.) **Base Volume** is a read / writable storage volume which can be presented to a server or group of servers within a nPod. A base volume may be utilized as a storage device for application data (e.g. /etc/mnt/app1) as well as the OS boot device.

2.) **Snapshot** is a point-in-time recovery point of an entire volume. The snapshot is immutable (read only) and is not chain dependent. Snapshot ownership follows the ownership of the base volume.

3.) **Clone** is a read / write copy of a volume snapshot. A clone can be mounted to an application server or a group of application servers within a nPod and be used to recover data.

Read Operations

Read operations in nebOS are independent of the volume access method and the data protection setting (mirrored / unmirrored) and is always serviced from the current volume owner.

When an application operates local to the SPU, which is the current owner of a volume, the read I/O process happens locally. In this scenario, the required data blocks may reside within the local battery backed SPU RAM module. When the data blocks do reside in the local RAM module, the I/O is served directly from memory. When data no longer resides in local RAM, the data is fetched from the SPU's drives, decompressed, decrypted, and returned to the application.

When an application operates on a server in which the SPU is not the volume owner, the I/O is passed through (forwarded) to the current volume owner SPU over the private TCP/IP data network. The data is read from the volume and returned over the private data network in a compressed and encrypted format. Once the forwarded request is received, the passthrough SPU then decompresses and decrypts the data. Decryption is done by utilizing the nPod disk encryption key (DEK) and the I/O is returned to the application.

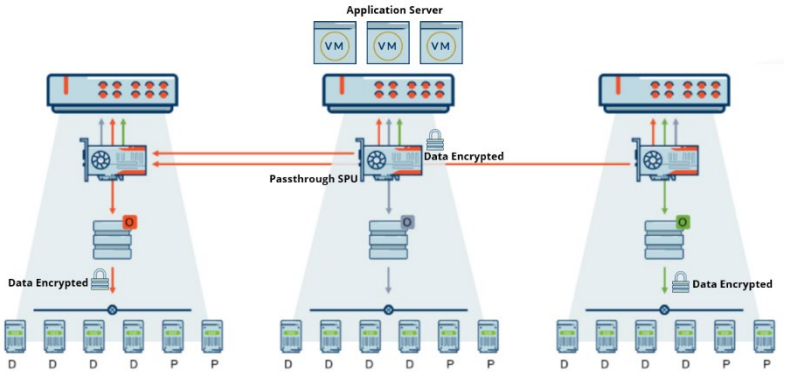


Figure : SPU which provides host access to a volume that is not stored locally on the SPU but owned by another SPU in the same nPod

Figure Passthrough

Write Operations for Unmirrored Volumes

Unmirrored volumes in nebOS are stored on a single owner SPU where the volume can be mounted by one or more application server(s) in the nPod. The owner SPU protects volume data locally using erasure coding where parity is used to re-calculate blocks of data in the event of drive failure. With unmirrored volumes' application data is not mirrored to a backup SPU in the nPod.

When an application operates local to a SPU that is the owner of a volume, write I/Os are processed locally. Data blocks are initially captured in a battery backed RAM module on the SPU, compressed using LZ4 compression algorithms, fingerprinted, hashed (deduplicated) and encrypted. At this point, an acknowledgement back to the application occurs and the I/Os are de-staged in 6MB chucks from memory to physical drives.

When an application operates on a server where the SPU is not the owner of the volume (passthrough), data is initially captured in the SPU RAM module on the passthrough SPU. Here, data is compressed using LZ4 compression algorithms, fingerprinted, hashed (for deduplication) and encrypted. The encrypted data is then passed through (forwarded) to the volume owner SPU over the private TCP/IP data network, where it is captured in RAM and acknowledged to the passthrough SPU and the application. The owner SPU continues to de-stage the data in 6MB chunks from memory to physical drives if the received data is unique. The owner SPU determines if data is unique using the hash which was calculated by the passthrough SPU.

Write Operations for Mirrored Volumes

Mirrored volumes in nebOS are stored on a volume owner SPU and the volume data is mirrored to a backup SPU. Mirrored volumes can be mounted by one or more application servers in the nPod. With mirrored volumes, both the volume owner and volume backup SPU protect data locally using erasure coding where parity is used to re-calculate blocks of data in the event of drive failure.

The volume owner SPU is responsible for de-staging I/Os to disk. The volume owner SPU synchronously forwards data modifications (writes) to the backup SPU, then acknowledges write completion to the application. When a write I/O occurs, data blocks are ingested and captured in a battery backed RAM module on the current owner SPU, compressed using LZ4 compression algorithms, fingerprinted, hashed (deduplicated) and encrypted.

The current volume owner SPU forwards (mirrors) the encrypted data I/Os to the current backup SPU and waits for an acknowledgement by the current backup SPU. This acknowledgement is provided when the current backup SPU has stored the data in its SPU RAM module. At this point, the current owner SPU responds to the application issuing the writes with an acknowledgement.

With mirrored volumes, when the backup volume SPU is unavailable, nebOS will stop forwarding IOs, raise an alert and acknowledge the write back to the application after data blocks are captured in the SPU RAM module. Once the volume backup SPU becomes available again delta data will automatically resynch via the TCP/IP data network. Volume resynch will continue alongside new IO until the backup volume is in synch with the volume owner.

When a volume owner SPU becomes unavailable, the volume backup SPU will assume ownership. Scenarios may include network issues, server power loss or complete SPU failure. When the volume owner SPU becomes available again it will become the current backup SPU and resynch changes alongside new I/O until the volume is in synch on both SPUs. Once the volume is in synch the natural owner SPU will re-assume current volume ownership from the natural backup.

In scenarios where current volume owner and current volume backup SPU are not in sync for a volume, failover will not occur to prevent data loss. In this scenario of a double failure, volume data will become unavailable to applications and self-heal once both SPUs are available again.

nebulon ON

A key component of **smart**Infrastructure is management as-a-service through a secure cloud-based control plane – Nebulon ON. Simple, secure, and scalable tooling and instrumentation for operating a large-scale IT infrastructure should not be a burden for application owners or infrastructure administrators but an inherent feature that doesn't require servicing or special attention, provides rich automation capabilities, and detailed infrastructure insights.

Nebulon ON allows organizations to provide self-service infrastructure management for application owners. In addition to providing insights for Artificial Intelligence for IT Operations (AIOps), Nebulon ON provides centralized server, storage, and application management that enables the necessary tooling of real-time detection of issues across the IT infrastructure stack and the facilities for corrective measures.

A single, secure API endpoint is provided for application owners with direct access to the AIOps functions and to fully automate their entire IT infrastructure needs. With global insights, Nebulon ON unlocks unprecedented opportunities for DevOps teams to control and query their servers, storage, and applications.

Since the control plane is provided as-a-service, it is always up to date. Lengthy and complex update procedures are an artifact of the past. Users of **smart**Infrastructure can make use of new features and enhancements instantly through cloud-delivered updates – a comfort that has yet to become mainstream in on-prem enterprise applications.

Users may login to Nebulon ON via an internet browser and navigate to:

https://on.nebulon.com

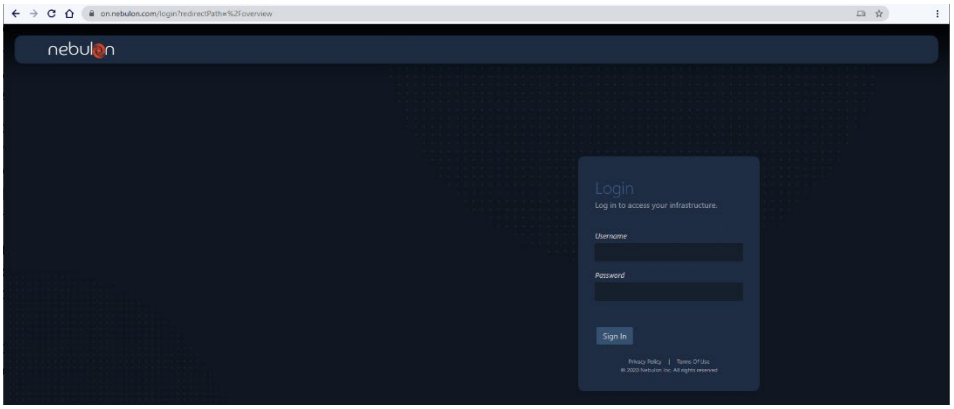


Figure : Nebulon ON is the secure cloud-based control and management plane

login.png

Recommended browsers include Gecko-based (Firefox), WebKit-based (Safari), and Blink-based (Google Chrome, Microsoft Edge) desktop browsers. While some functionality is provided for mobile browsers, the user experience is optimized for desktop versions.

nebulon ON Organizations

In a Nebulon environment, a Nebulon ON Organization represents a customer environment and is the cloud-based tenant structure. Organizations provide customers with the ability to centrally govern, control and scale their resources. Nebulon ON provides the mechanism to authenticate, communicate, automate, and manage a Nebulon powered on-prem IT infrastructure. On-premises SPUs are registered to a single Nebulon ON Organization and are unable to communicate outside of this security boundary.

Initial login details will be provided to the Organizational Admin and, upon the first login, the user will be prompted to accept the Nebulon End User License Agreement (EULA) (Figure) along with Terms and Conditions. By accepting the EULA your company accepts these terms and conditions supplied within, so it is recommended for users to go through the appropriate internal mechanisms to ensure this is done in accordance with your company policies. Do not forget to download a copy of the EULA for your own records. Alternatively, a copy of the EULA can be requested by opening a support case through the nebulon ON web user interface.

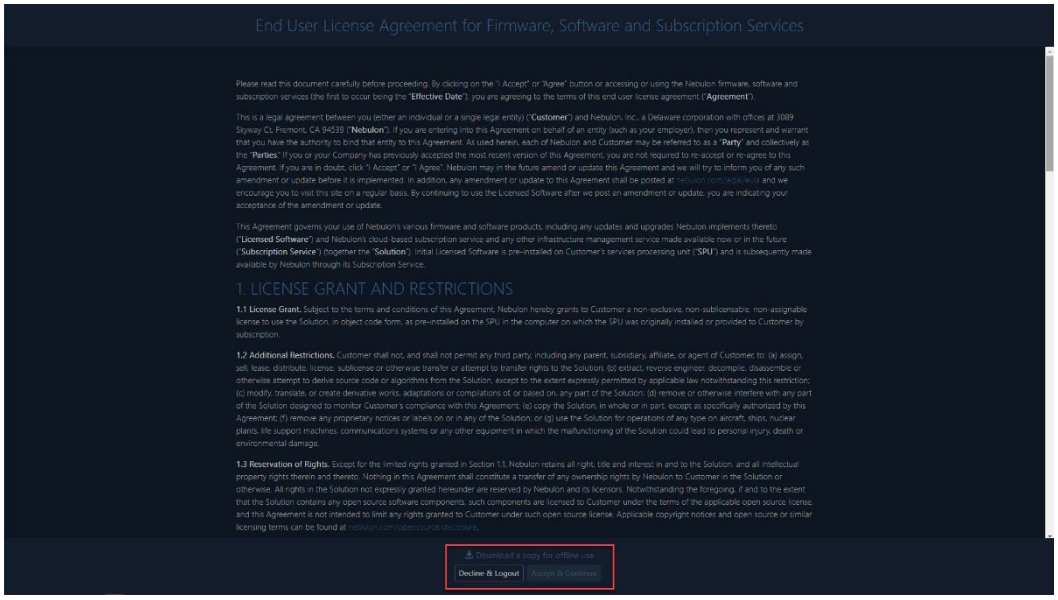


Figure : Upon initial login to Nebulon ON, authorized company personnel must accept the Nebulon EULA

Secure Management and Security Triangle

Nebulon ON, is the cloud-based control plane which manages **smart**Infrastructure and is accessible from anywhere with a working Internet connection. Stringent security measures are needed to ensure that data and applications are always secure. With this, data security is (and will always be) a key principle, a fundamental element, and building block of **smart**Infrastructure.

Nebulon differentiates between user and application data, system metrics and diagnostics data (telemetry), and control data. User data and application data never leaves the customer's on-premises infrastructure and is securely stored with data-at-rest encryption. This ensures the user is in full control over where their data is stored, eliminating the worry about secure erasing of physical storage media, and minimizing the risk of data leakage. This also aligns well with data sovereignty requirements for GDPR, as an example. Nebulon also encrypts all control-plane traffic occurring between users, Nebulon ON, and SPUs.

System telemetry and continuous streaming of diagnostics data is leveraged for analytics and provides predictive recommendations for optimizing customer environments.

Control data encompasses instructions issued by users or automation systems to SPUs and nPods. To secure control data and ensure authenticity of commands, Nebulon implements a unique authentication mechanism between the user, SPUs, and Nebulon ON, which form a "secure message exchange triangle" for control data. This layer of security prevents unauthorized changes to the on-premises storage assets (nPods and SPUs) from the cloud.

Once authenticated, users have access to management and monitoring functions provided through Nebulon ON in accordance with the set authorization policies. However, management actions that involve changes to on-premises infrastructure require an additional level of authentication. Users need to be inside their organization's firewall and have network access to SPUs. This prevents users external to a customer's network, including Nebulon employees, from altering an organization's on-premises infrastructure. To ensure that a user is inside the corporate firewall and has access to an SPU, management commands issued in to Nebulon ON are routed through the SPU and back to the cloud

through a “secure message triangle.” This secure message exchange between a user, cloud, and the SPUs can be separated into 5 steps.

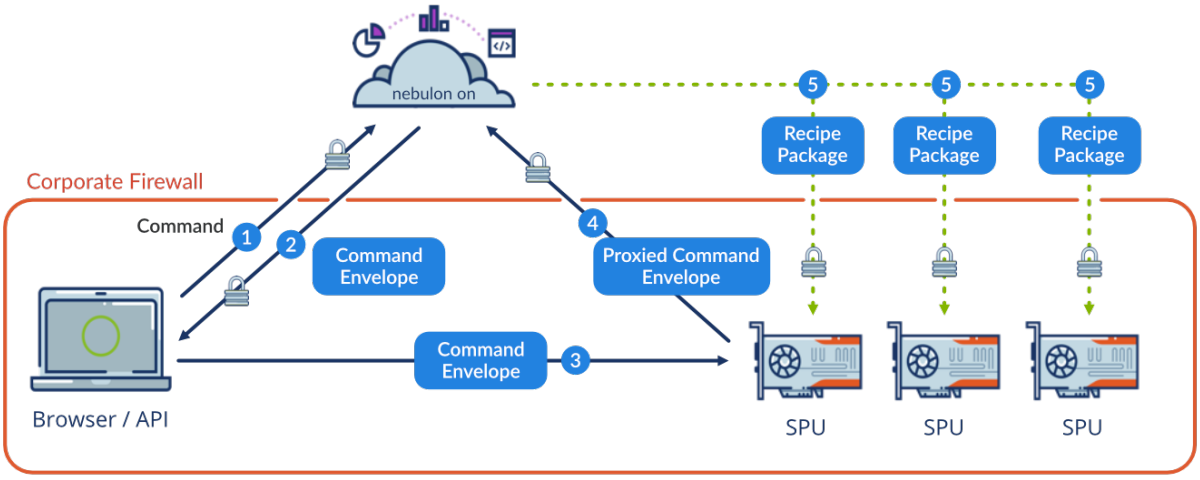


Figure : Nebulon Security Triangle

triangle.png

After the authenticated user issues a command (1) to Nebulon ON, using the graphical user interface, API, or SDK it determines if the user has appropriate permissions. If the user is authorized to execute the command, Nebulon ON generates a signed Command Envelope (2).

Nebulon ON returns the signed Command Envelope with additional network address information for all SPUs that are targeted in the original command. The browser or API use the network address information and pass the Command Envelope to one of the SPUs involved in the operation using a secure HTTPS connection (3). This SPU is called Proxy SPU.

The Proxy SPU will accept the Command Envelope and relay it back to Nebulon ON (4), attaching additional information to create a signed message. After receiving a message from the Proxy SPU, Nebulon ON verifies its authenticity and unpacks the Command Envelope. It verifies the Command Envelope’s contents as they were originally set by Nebulon ON. If the Command Envelope is valid, Nebulon ON generates a recipe of instructions. It places it with additional data on the message queue for each affected SPU.

Each SPU processes Recipe Packages in their message queues individually (5). Each SPU first fetches the Recipe Package from the queue and validates its signature. From the Recipe Package, it can extract the Proxied Command Envelope and verify its signature. It also verifies that the Proxy SPU is in the same nPod as the SPU itself.

From the extracted Command Envelope, the SPU verifies the attached signature, that its timestamp is sufficiently recent, and that the command was not executed recently. This is done to ensure that any recipe is only executed once and guards against unauthorized attackers inside the firewall replaying messages.

After logging the original command, the recipe, and other data, the SPU will execute the requested command and record the one-time code in a local cache.

Nebulon SPU Lifecycle Management

When deploying **smart**Infrastructure, SPUs are registered with individual nebulon ON organizations. Secondly, SPUs may only be registered with a single Nebulon ON organization. Before an nPod can be provisioned and applications deployed on **smart**Infrastructure, SPUs must first be registered (or discovered) with nebulon ON.

Adding SPU to a Nebulon Organization

To register an SPU, the SPU requires a valid IP address on the control interface. The first SPU in an organization must be added either via API, SDK or via nebulon ON GUI. Upon registration (or after auto discovery), Nebulon begins to collect system and telemetry data from SPUs and nPods can be provisioned.

To manually register an SPU with nebulon ON navigate to **Admin** tab and choose **Register New SPUs**.

1. Open the register SPU dialog from within the nebulon ON GUI:

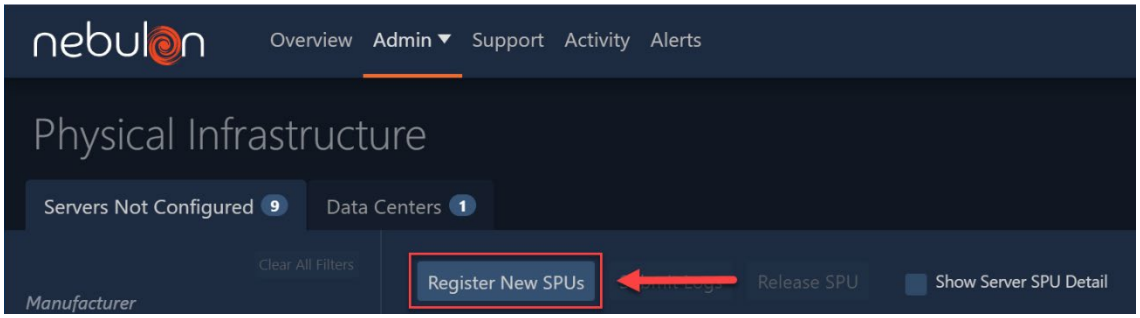


Figure : Register an SPU with a nebulon ON organization

register-spu-1.png

2. Enter the SPU serial number. The serial number of an SPU can be obtained from server UEFI or the label on the side of the SPU.

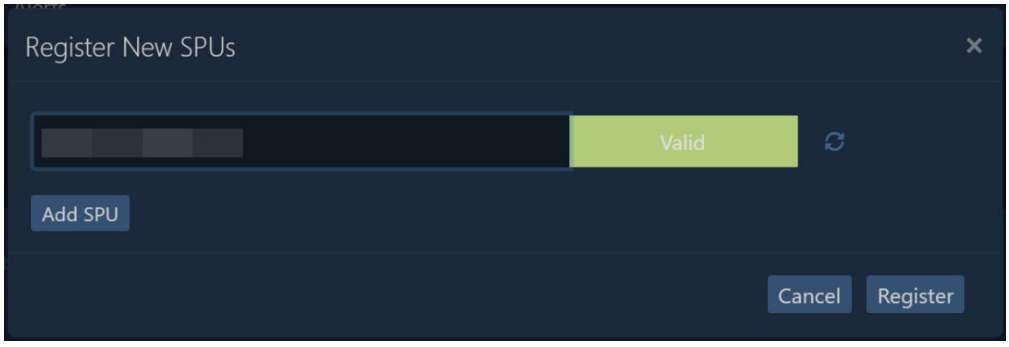


Figure : Register SPU with nebulon ON using serial number

register-spu-2.png

While the first SPU in an organization is added via API, SDK or via the nebulon ON GUI, subsequent SPUs on the same LAN are auto discovered. Servers with auto discovered SPUs can be viewed via **Admin ->**

Physical Infrastructure (Figure --)**.** Once SPUs are registered with a nebulon ON organization, they are available to be utilized in nPod creation.

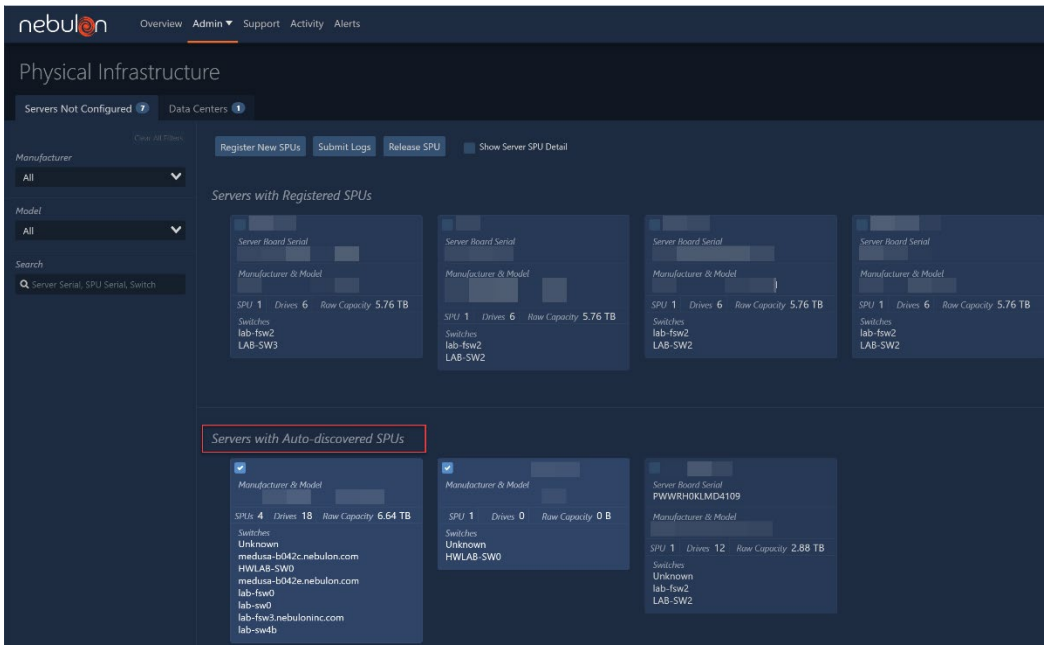


Figure : Application servers with auto discovered and manually registered SPUs

physical-infra.png

Release an SPU from nebulon ON

When servers and SPUs are decommissioned, they can be removed and disassociated with nebulon ON. When an SPU is released from nebulon ON it is no longer able to participate in an nPod and needs to be manually registered or auto discovered once more. The release of an SPU from an organization can only occur when the SPU is not already a member of an nPod. The release of an SPU should only be done whenever you wish to decommission the SPU from nebulon ON.

To release an SPU from nebulon ON GUI navigate to the **Admin** tab, choose **Physical Infrastructure,** and select the SPU(s) which are to be released from the organization (Figure xxx). To complete the release, choose **Release SPU** and provide confirmation (Figure yyy).

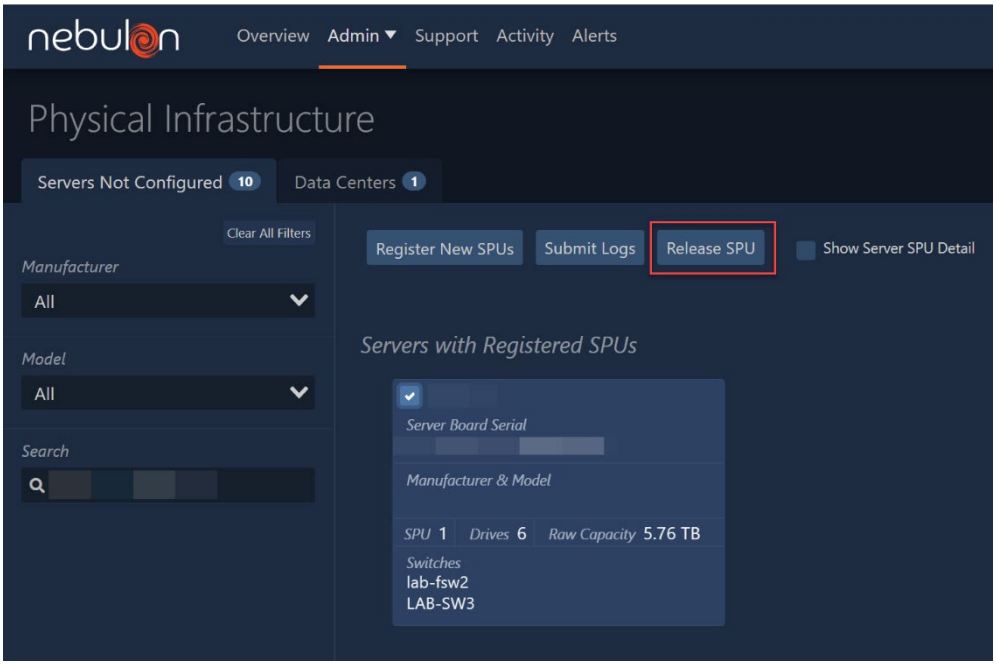


Figure : Release SPU(s) from nebulon ON

spu-release-1.png

Figure : Confirmation that the entered SPU(s) will be released

spu-release-2.png

Role Based Access Control

The cloud-based control plane, Nebulon ON, provides security access control to your Nebulon powered infrastructure. Role Based Access Control (RBAC) utilizes Users, Groups, Roles and Policies to provide fine-grained control over who can authenticate (sign in) and is authorized (appropriate permissions) to utilize Nebulon resources in the organization.

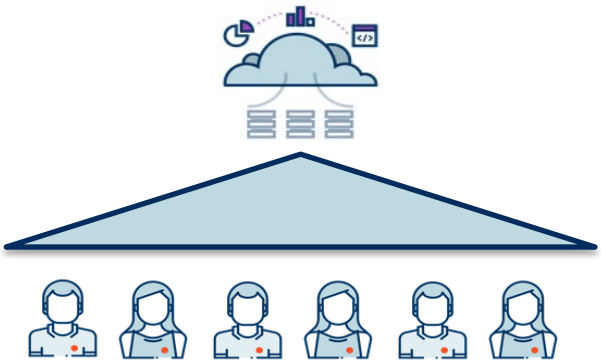
Upon initial login to Nebulon ON, users will begin with the Organizational Master Account. This user account is accessed by signing in with the email address and password in which you registered with Nebulon Support. Nebulon strongly recommends that you do not use the organizational master account for everyday tasks. Instead create additional user accounts and lock the credentials for organization

master away. When allocating permissions, it is recommended to follow the principle of least privilege – only assign enough access to perform the required job.

Nebulon ON RBAC is administered using Nebulon ON web portal or the Nebulon SDKs.

Users

A user account is the principal entity which you create within Nebulon ON and represents a service account or an individual person that interacts with Nebulon resources (Figure XX). A user account is used to authenticate with Nebulon ON when accessing the graphical user interface, API and SDK's. Whenever a user is created the user is assigned to a group and you may also assign a policy directly to an individual

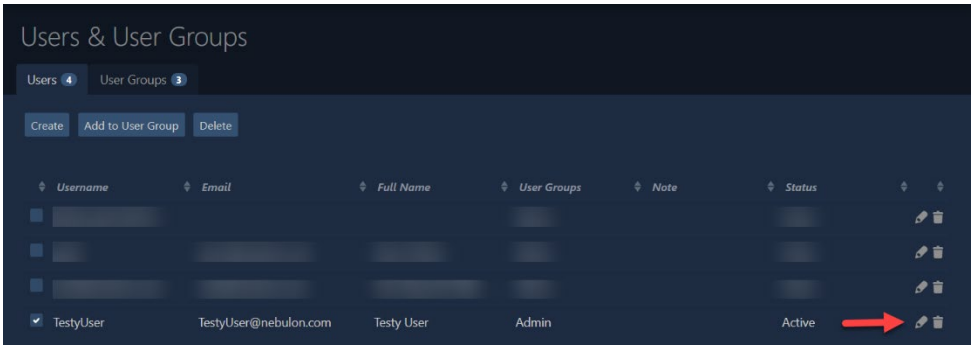


user a.

users.png

Figure XX: A user account represents an individual or a service account which interacts with Nebulon resources.

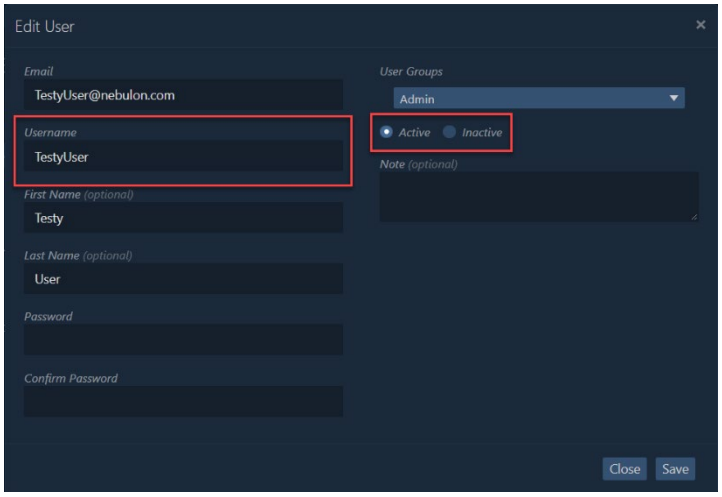
User accounts are identified by a unique username. It is recommended that administrators construct usernames to match individual user e-mail addresses. Administrators may also edit an individual username after creation via the User Management modal within Nebulon ON or via SDK (Figure xxx). A single username may only be associated with one Nebulon Organization.



editUser1.png

Figure xxx: User management is done via the User modal.

User accounts may also be marked as either active or inactive. Figure xxxx.



EditUser-2.png

Figure xxxx: Usernames may be altered after creation and marked as either active or inactive.

User Groups

A user group is a collection of users within a Nebulon Organization. Groups are typically created for specific sets of users or lines of business and provide an easy mechanism of ensuring that the assigned set of users all have the same permissions applied. For example, a new user is created for the Marketing team – simply add them to the Marketing group and they will instantly have the permissions which are scoped to affect only the Marketing resources (Figure xx). There are a few considerations users should consider:

- You may not nest groups; a group may not be a member of a group, only users.
- Users may be a member of more than one group.
- When a user belongs to more than one group, the most permissive right is in effect.
- When a right is not explicitly allowed, there is an implicit deny applied.

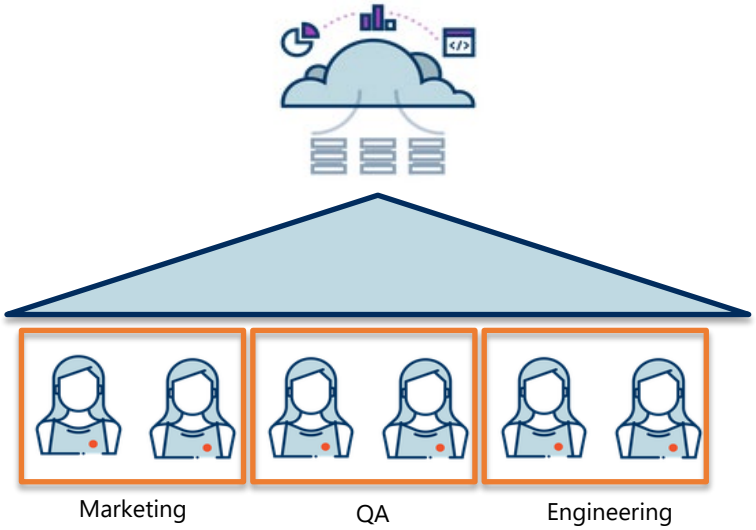


Figure XX : Groups may contain users and assign permissions to Nebulon resources and actions.

usergroups.png

Built-In Groups

Users may create their own custom groups or utilize the three (3) built-in groups to which users can be added to:

Group name	Description
Admin	The built-in Admin group has the Admin role assigned and grants users the ability to perform *create, update, delete and read* actions on all resources
Operator	The built-in Operator group has the Operator role assigned and grants users the ability to perform all create, update, delete and read actions apart from user management and FirmwareUpdate actions
Monitor	The built-in Monitor group has the Monitor role assigned and grants users the ability to read all organizational resources

Roles

Nebulon RBAC provides users the ability to apply a set of rights to an individual user or to a group of users via a role. A role defines a collection of rights (permissions) that are associated with a specific function of a user or a user group inside a customer organization. The actions that a user or user group may perform are expressed with the Rights Definitions of a role. For example, there may be a function in an organization that is permitted to create and modify Configuration Templates for the entire organization.

Built-In Roles

Nebulon ON provides three (3) built-in roles in which users may utilize. If the built-in roles do not meet an organization's specific requirements, users can create their own custom roles as well. The built-in roles are:

Name	Description	Right Definition
Admin	Provides full access to all resources in the organization	• /*/*
Operator	Provides full access to all resources in the organization, except user management and firmware management	• /Lab/* • /Alert/* … • /User/read • FirmwareUpdate/read
Monitor	Provides read-only access to all resources in the organization	• /*/read

Permissions

Nebulon ON permissions can be one of the following:

- ***** - everything is permitted.
- **Read** – Read operations are permitted.
- **Create** – Create operations are permitted.
- **Update** – Update operations are permitted.
- **Delete** – Delete operation are permitted.
- **Execute** – Execute operations are permitted. Execute is used for operations which do not fall into the previous categories.

Rights Definitions

Nebulon ON Role Based Access and Control (RBAC) allows users to define least privileges required to perform the required job. The Rights Definition below lists out which resources Nebulon RBAC controls may be applied.

The following are Nebulon resources:

```
"Datacenter"
"Lab"
"Audit"
"Alert"
"FirmwareUpdate"
"UserGroup"
"nPodGroup"
"Volume"
"PhysicalDrive"
"User"
"nPod"
"SnapshotScheduleTemplate"
"SPU"
"Row"
"Rack"
"nPodTemplate"
"SnapshotSchedule"
"Host"
"LUN"
"Webhook"
```

In some definitions, users may want to reference all resource types without listing them explicitly. In this case they can use “*” as a wildcard that includes all resource types.

Multiple Role Allocations

Often, users may wind up in multiple groups which have overlapping roles. For this case, Nebulon ON employs an additive permission model, meaning that the permissions applied are the most permissive in the given situation. For example, Role A includes read and Role B includes create for a nPod – the effective permissions for a user with these roles are both create and read.

Policies

A Policy is the association of a user or group with a combination of permissions via a role and a scope of resources. For example, a user "Joe" in the engineering team may have the role of Operator for a nPod with the name Kubernetes. In this example, Operator is the role in the policy, the scope is the Kubernetes nPod and the user is "Joe".

Scope

Scope is the set of specific Nebulon resources which a policy applies to. A scope is defined only at the policy level and explicitly grants permissions to the set of resources that are defined in it. It is important to understand that by limiting scope, the risk and potential security vulnerability in cases of a compromised user account is significantly lower.

Built-In Policies

Nebulon ON provides three (3) built-in policies where users are unable to edit or make any changes to:

Policy name	Role	Scopes	User groups
Admin	Admin	/nPodGroup/*	Admin
Operator	Operator	/nPodGroup/*	Operator
Monitor	Monitor	/nPodGroup/*	Monitor

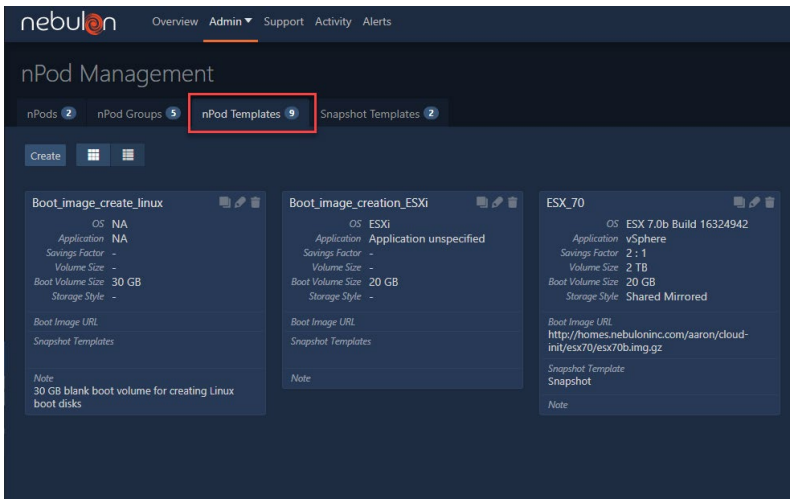
Template-Driven Application Deployments

Infrastructure as Code (IAC) allows application owners, developers, and infrastructure administrators a method of automating the deployment of applications in a reliable and consistent manner along with at scale. Nebulon ON provides users with a template driven approach to deploying and configuring Nebulon resources. These resources include nPods (application clusters) and volumes with associated policies for data reduction, volume access, and snapshot schedules. These decisions are critical, and it is important that resources follow organizational best practices. Nebulon's template driven approach permits the consistent and reliable deployment of applications while fitting very well into an existing deployment pipeline.

Nebulon nPod Configuration Templates

To enable self-service infrastructure provisioning for product teams, nPod Configuration Templates are used to help users automatically provision storage artifacts, based upon organizational and application best practices.

nPod Configuration Templates define, for applications, the ideal storage configuration for a set of SPU-equipped application servers which will participate in the nPod. Templates are used to repeatedly and consistently build multiple nPods which conform to established best practices. Template configurations are flexible in that they specify the characteristics of a desired configuration. In this way, a template configuration for a MongoDB nPod can be used on 3 servers or 32 servers, 1 nPod or 50 nPods. Configuration Templates ensure consistent application storage settings which align to organizational best practices across an environment.



configtemplates.png

Figure A: nPod Configuration Templates define ideal storage configurations for a set of SPU-equipped application servers

Configuration Templates dictate how storage resources are configured and presented to the application servers within the nPod. Templates define the expected savings factor (deduplication and compression), volume size, volume access method (single host access or multi-host access), and optionally, the boot device configuration. Nebulon ON collects telemetry data from customers and runs machine learning against the data to help drive intelligent decisions for customers. As such, Nebulon provides curated, pre-configured and application-optimized Configuration Templates which align with optimal storage settings. Customers may clone and further customize the curated Configuration Templates to meet their specific needs and requirements.

Configuration Templates allow users the ability to define the application which will run within the nPod along with the required Storage Volumes (see Figure B below). Nebulon Configuration Templates allow users to define their storage requirements. Users can define the number of volumes required (Set Volume Count) and/or the size of the volumes required (Set Volume Size). Pre-provisioned volumes will export storage capacity in the nPod and are based on a combination of the specified volume size, the size and number of physical drives in the server, and the specified savings factor.

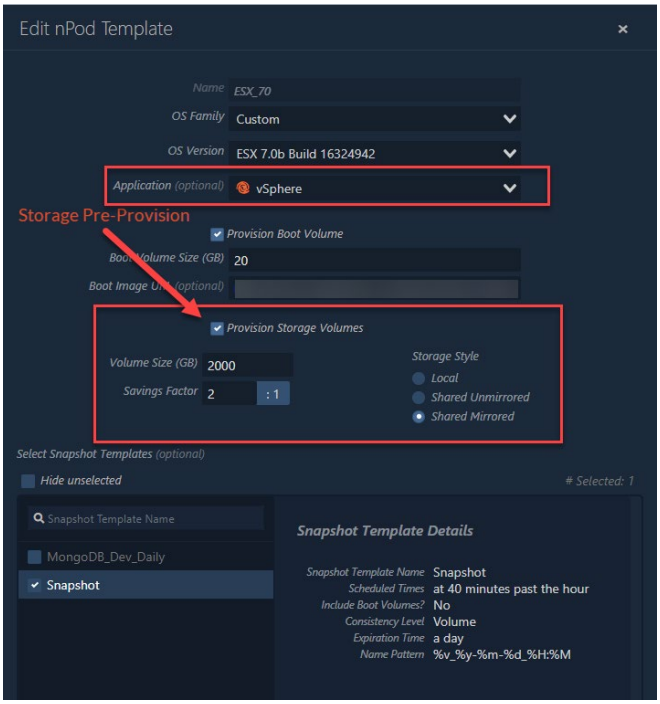


Figure B: nPod Template creation wizard within Nebulon ON

Some applications are best served by allowing dynamic or on-demand provisioning of data volumes. As an example, a Kubernetes cluster can make use of a Container Storage Interface (CSI) Driver to dynamically provision storage volumes for containers. Nebulon nPod Configuration Templates allow you to also serve these use cases.

Note: For detailed information on running Kubernetes with Nebulon, refer to the Nebulon Kubernetes Infrastructure Guide which can be obtained from www.nebulon.com.

nPod Template Creation

Nebulon provides a set of curated nPod Configuration Templates for customers to apply. If customers have a specific need as to how storage is provisioned to servers in a nPod, they can create their own or clone and customize one of the curated Configuration Templates. Templates are created through the **Admin nPod management à nPod templates** menu in Nebulon ON. When creating a template, the following information should be specified:

Property	Description
Name	The nPod template name is a unique, human-readable identifier in the customer's organization. It should be a descriptive name based on the function of the template.
Operating System	This field describes the server operating system (OS) that any nPod using the template will run. Major OS version and patch information can be selected here.

Application	The name of the application that is targeted by the nPod Configuration Template. Note: A precise selection for this field will yield in better recommendations from the AI recommendation engine in nebulon ON.
Provision Boot Volume	If selected, this option allows provisioning of a local, non-shared volume for the host operating system from the SPU. If not selected, there must be another boot device present in the server. There are two ways Nebulon boot volumes can be leveraged. ● Installing to an empty Nebulon boot volume. In this scenario, the *Boot Image URL* field is left unconfigured in the template and a blank boot volume is presented to the nPod server. The OS must be installed to this volume. ● Booting from a pre-configured OS image specified as part of the nPod Configuration Template. In this scenario, a URL that points to a valid OS image, optionally compressed with any one of the gzip, bzip2, or xz algorithms must be supplied as part of the nPod template in the *Boot Image URL* field. This enables servers to boot into a pre-installed server environment that can then be customized (either manually or via automation) before installing the application. When using a Nebulon boot device the servers must be configured for UEFI boot mode. The Nebulon device will also need to be configured as the first boot operation.
Boot Volume Size (GB)	This specifies how large to create the boot volume for the nPod server. There are two items to consider when specifying the size. ● Capacity specified for the boot volume is unavailable for nPod data volumes. Therefore, this volume should be sized correctly for the server OS. ● If specifying a boot volume in the Boot Image URL field, the boot volume must be as large or larger than the size of the volume from which the boot image was captured. For example, if the boot image was captured from a 50GB volume, the boot volume must be at least that large.
Boot Image URL	This is a valid HTTP or HTTPS URL containing a boot image to be used for the nPod servers. If using HTTPS, the server must have a valid certificate. If using a DNS name for the server, the SPUs must have a valid DNS server specified in their management network configuration. It is recommended to use IP addresses for the boot image URL. Currently gzip, bzip2, or xz images are supported as boot volumes. The format for the URL is http://<server>/<path>/<image_name>

	For example, http://10.100.4.70/bootimages/centos.gz When a nPod is first created, the server will check for an image at the specified URL. Note that there can be only one boot volume per server. If an image is found, the server will download and decompress the boot image into LUN 0. Once the boot image has been fully decompressed to the boot volume, a "Nebulon NebStore 0001" device will be presented to the server.
Provision Storage Volumes	If selected, this option tells Nebulon ON to pre-provision data volumes as per the definition in the template. If not enabled, volumes should be created dynamically using an application plugin (e.g. Kubernetes CSI-Driver) or the Nebulon provided SDKs.
Volume Size	The volume size field controls the size of volumes to create. The nPod will create as many volumes of this size as can fit on a server. The specified volume size, the size and number of physical drives in the server, storage style, and the savings factor dictate how many volumes will be created.
Savings Factor	The savings factor is the expected reduction ratio for a workload and is the product of the deduplication ratio and the compression ratio. The savings factor is used in conjunction with the volume size and volume storage style when determining how many volumes to create on each SPU.
Storage Style	This toggle controls the availability and protection model for volumes created on the nPod servers. ● **Local** – SPU volumes are presented to the local server only. ● **Shared Unmirrored** – All SPU data volumes in the nPod are available to all hosts in the nPod, but they are not mirrored to secondary SPUs for extra protection. ● **Shared Mirrored** – All volumes in the nPod are available to all hosts in the nPod and a secondary copy of each volume is available through another SPU in the nPod for increased availability.
Snapshot Templates	Allows the selection of a previously created snapshot schedules that will be automatically executed after the nPod creation.
Note	Allows the specification of additional information to describe the template

nPod Storage and Configuration Templates

An nPod must reference a Configuration Template. A Configuration Template may optionally define storage requirements to be followed whenever a nPod is provisioned. Users can define both the number and/or size of volumes required for their application. For example, a user defines a Configuration Template to provision 10 multi-host access, mirrored volumes with a capacity of 1TB each. When the nPod is provisioned, Nebulon will validate the configuration is valid (i.e. is it even possible to provision quantity 10 mirrored volumes of size 1TB), if validation is successful, volume provisioning will occur with the nPod creation and the volumes will be exported and presented to the servers within the nPod. A second example highlights behavior when a user only specifies desired capacity. In this case a user creates a Configuration Template and defines that the required volume size is 3TB. Nebulon will pre-provision as many 3TB as possible and will consume all capacity within the nPod. A final example highlights only defining the number of volumes required within the nPod. Here, a user creates a Configuration Template and defines 5 volumes to be provisioned with the Storage Style "local". Nebulon will create 5 equally sized volumes on each SPU which will consume all capacity within the nPod. Diving into this example a bit further, a SPU in the nPod has a possible 10TB and the user requested 5 volumes, each volume will be 2TB in size on that SPU.

nPod Creation

nPods are created by applying a Configuration Template to a collection of available application servers. In the GUI, users create a new nPod from the **Admin** -> **nPod management -> nPods -> New nPod** menu. Users simply select from a list of automatically discovered application servers, select the application or nPod template, name the nPod and assign it to a nPod Group for access control. As servers are selected, the GUI displays a dynamic nPod preview detailing the capacity metrics, servers and number of SPUs and volumes to be provisioned.

When creating a nPod, the following information is required (See Table __ below)

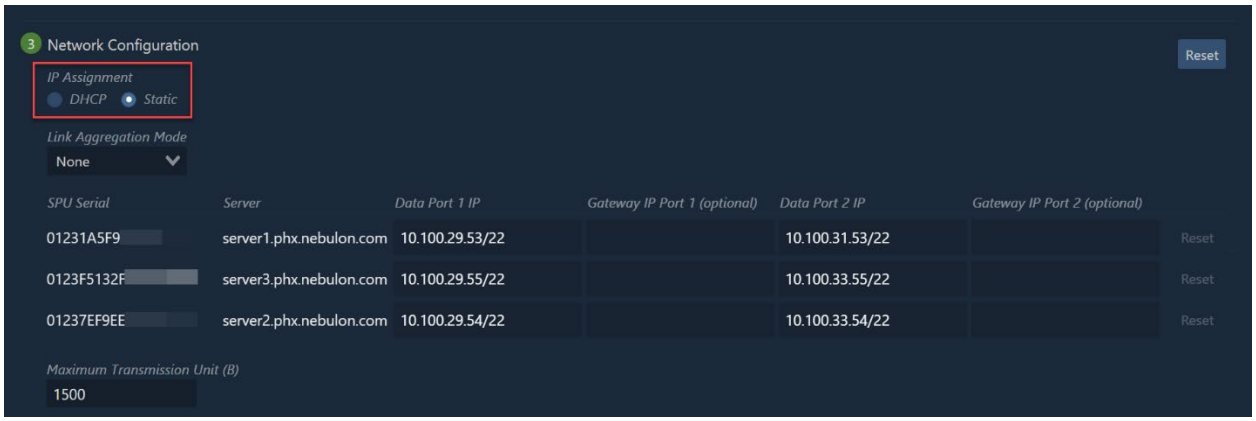
Property	Description
Name	A meaningful name for the nPod
Configuration Template	The nPod or application template to use for provisioning
Data Network Information	The network configuration information for the data network, e.g. IP address information and link aggregation information
nPod Group	Which group to assign the new nPod to. A nPod group may contain multiple nPods.
Note (optional)	Allows the specification of additional information to describe the nPod

Table

Once the minimum number of servers are selected, a nPod may be created. The high-level steps for creating a nPod are listed below:

- The SPU data network ports are configured
- Connectivity between nPod member SPUs is verified
- SPUs are joined to form a cluster
- Volumes are created on each nPod member based on specifications within the applied Configuration Template
- Boot volumes are pre-populated
- Volumes are exported to each nPod member server. If volumes are shared, SPU primary and backup relationships are defined

The nPod creation wizard, SDKs and APIs include the ability to configure the SPU data network (Figure). When choosing the Link Aggregation mode of None, Administrators may dynamically (DHCP) or statically assign IP address information. When utilizing DHCP it is recommended that Administrators create DHCP reservations. This will ensure that the data interface IP address will not change when a DHCP lease expires. Whenever the Link Aggregation Mode is set to LACP, the option for DHCP is not supported. Lastly, you may configure the maximum transmission unit size (MTU) – 1500 bytes is the default MTU. When configuring MTU size larger than 1500, otherwise known as jumbo frames, ensure the upstream switch ports are configured for the appropriate MTU size. The maximum MTU size supported is 9216.



updated_npod_networking.png

Figure : nPod Data Network Configuration and Link Aggregation Mode

VMware vCenter Server Configuration

Upon creation, Nebulon Configuration Templates will apply the specified settings to the nPod. Specific to VMware based environments, administrators may optionally connect their vCenter Server to Nebulon ON. This integration allows users to view information related to their VMware environment through Nebulon ON. Nebulon ON gathers system telemetry and performance data to allow system recommendations using ML / AI. This will help application owners and administrators troubleshoot and resolve issues faster and, in some cases, be notified of issues before they occur.

From the nPod **Management** detail pane within Nebulon ON users can optionally set their vCenter Server Credentials. Nebulon ON requires read only permissions to the vCenter Cluster in where the nPod is

deployed. It is recommended that administrators create a service account for Nebulon ON to authenticate to the vCenter server.

Users set **vCenter Config** from within the nPod detail.

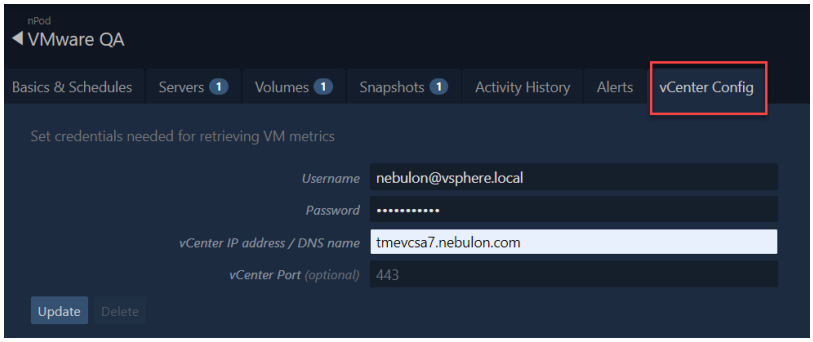


Figure : Set vCenter Credentials within Nebulon ON

setvcenter.png

From the **Overview** tab within Nebulon ON, users can change the card type to **Application** to view information related to the aggregate. The image (Figure) below illustrates a nPod Group named Production where 100% of the VMs are powered on.

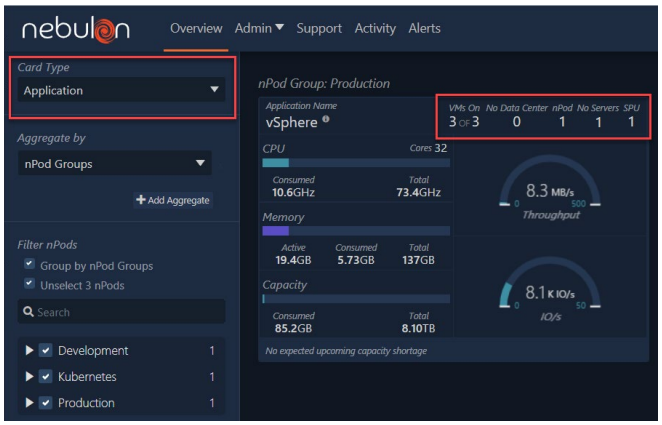


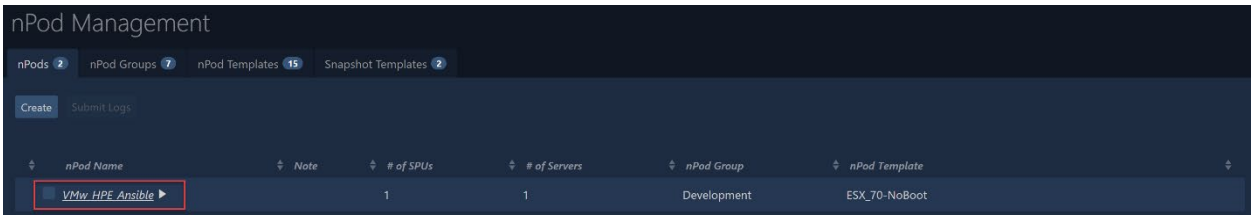
Figure : Application Overview view within Nebulon ON

Pod Deletion

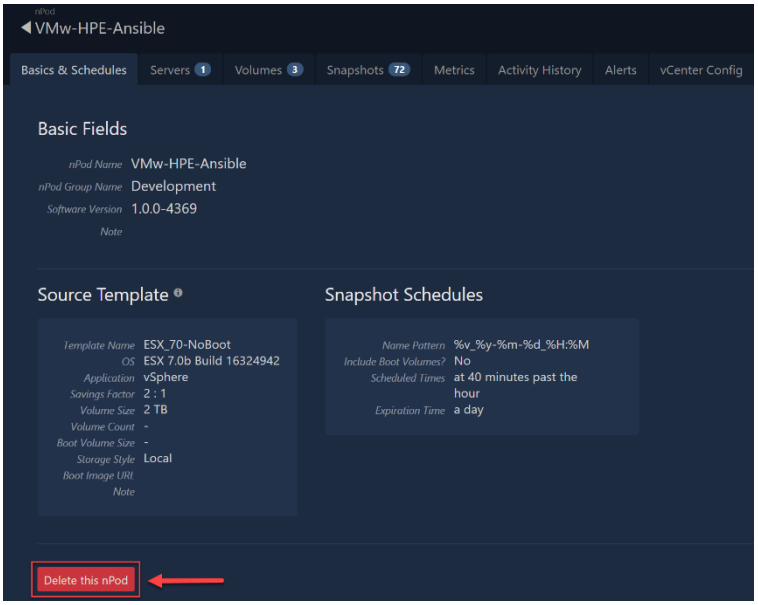
A benefit which **smart**Infrastructure provides is the composable nature and the speed in which application clusters can be deployed or re-deployed. As part of an application lifecycle, an application cluster may need to be deleted. The deletion of an nPod is an irreversible action. The nPod key encryption key is stored within the SPU cryptographic co-processor, at the time of deletion the key encryption key is irrecoverably overwritten with random data. This renders the encrypted data encryption key unusable and it is no longer able to decrypt data. In the end, the application data is not recoverable from the server-based drives.

nPod deletion is an irreversible and destructive action and must complete the Security Triangle to ensure the validity and authenticity of the request.

From the **Admin** tab within Nebulon ON, users can manage and administer their nPods. To delete an nPod users select an nPod in which requires deletion by clicking the nPod hyperlink. The image below (Figure --) illustrates an nPod named VMw-HPE-Ansible.

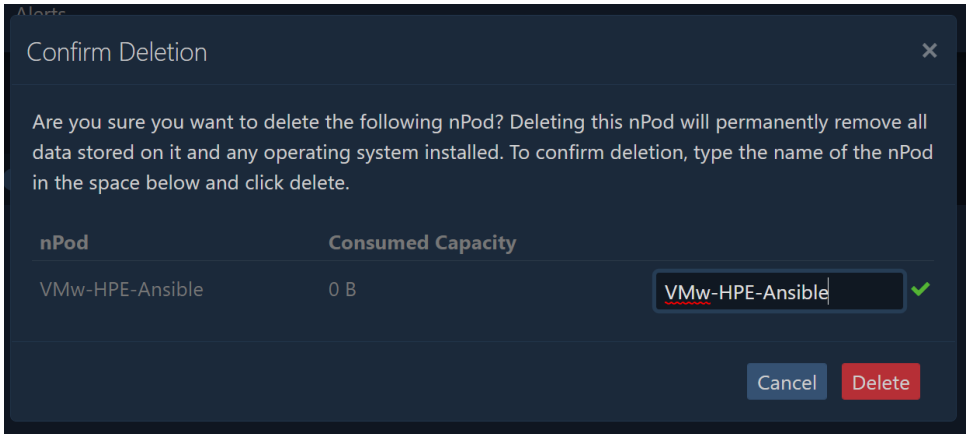


npod-delete-1.png



npod-delete-2.png

The deletion of an nPod requires confirmation. To confirm nPod deletion enter the nPod name.



npod-delete-3.png

Upon the deletion of an nPod, SPUs are not immediately available for new nPods as their configuration is wiped. After successful wipe, within a few moments, the SPUs are made available again and able to form a new nPod.

nPod Volume Creation

In addition to template-based provisioning of storage, users have the option to manually create volumes using the GUI or by use of automation frameworks and the Nebulon SDKs. During volume provisioning, users define the appropriate data access and required distribution method. For example, users may provision 2 TB volumes which are shared and mirrored. nebOS will handle the export, mounting, and assigning access to the volumes automatically. A volume can be a base volume or users may clone a point-in-time snapshot to become a volume. At the time of creation, volumes are assigned unique UUIDs as well as a user visible Worldwide Name (WWN).

For provisioning storage volumes, Nebulon maintains secure management of on-premises data by utilizing a patent-pending security process referred to as the Security Triangle, as discussed previously. In the case of volume creation, a control-plane change requires authentication and validation using the Security Triangle. Upon successful authentication via Security Triangle, Nebulon ON will send the required recipe containing the instructions for the SPU to execute.

Volume Data Access and Distribution

There are three (3) methods of data access and distribution available when self-service provisioning a Nebulon nPod and provisioning data storage volumes for application and operating system access using nPod Configuration Templates. The storage distribution method ultimately comes down to the use-case for the volumes and application requirements.

There are three (3) types of data access and distribution models selectable in an nPod Configuration Template:

- Local
- Shared unmirrored
- Shared mirrored

The table X4 below summarizes the storage styles available and example use cases within a Nebulon nPod:

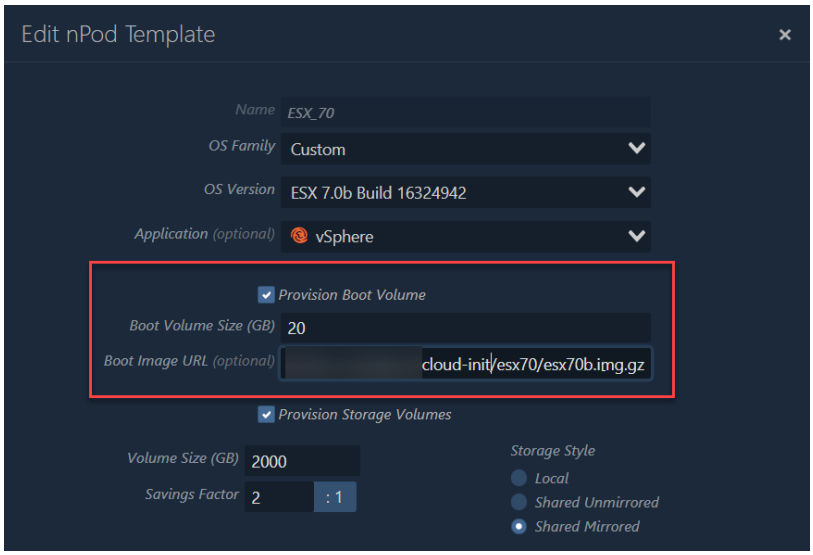
Storage Style	Resistant to SPU Failure	Multi-Host Access	Example
Local	No	No	MongoDB or other standalone application servers.
Shared unmirrored	No	Yes	VMware Dev
Shared mirrored	Yes	Yes	VMware Production

Table X4

***Details about storage configuration options are discussed in the section* Nebulon Volume.**

Provisioning Boot Volumes

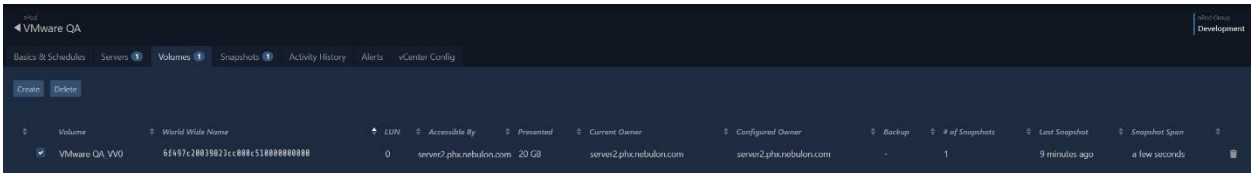
Optionally, users may create volumes from which application servers' OS will boot from. Boot volumes are provisioned at the time of nPod creation and will lay down the necessary OS based off a golden image(s) which is pre-built and conforms to organizational requirements and best practices. For example, users would install the OS of their choice with the necessary patches and anti-virus software. This information is configured in Configuration Templates. The screenshot below, Figure XX, illustrates a nPod Configuration Template which will provision a 20GB boot volume based off the esx70b.img.gz image.



npodboot.png

Figure XX

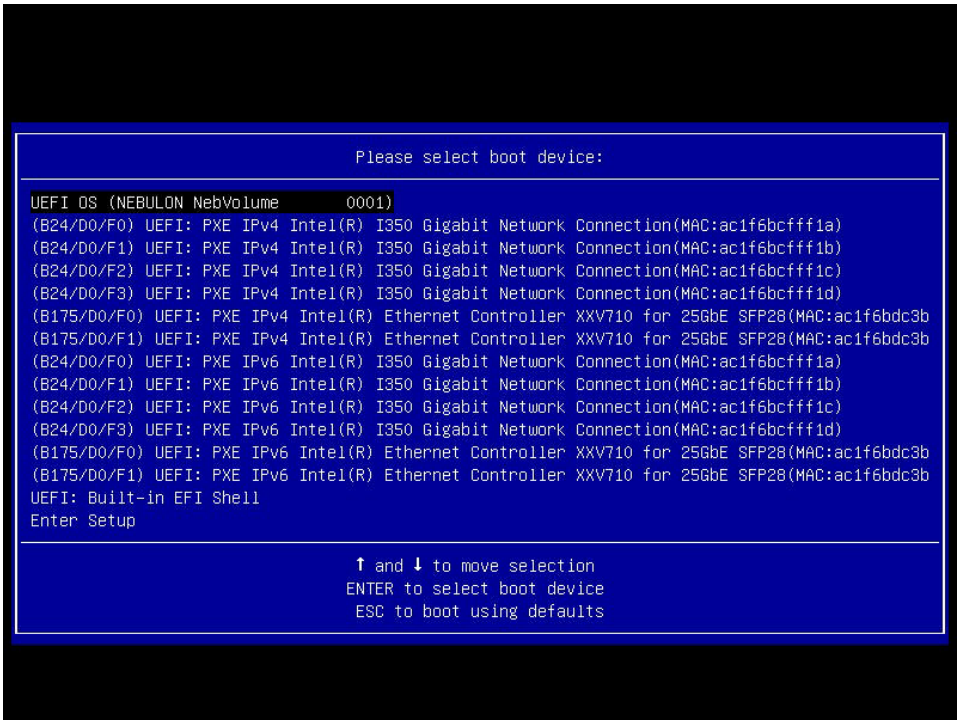
Boot volumes can be placed on a web server of your choice within the environment and be in a compressed format (gzip, bzip2). Once a nPod is created, users can view volume details within the nPod detail page. The screenshot below (Figure XX) illustrates a 20GB virtual volume – VMware QA_VV0 – which is only visible to server2.phx.nebulon.com.



bootvolume.png

Figure XX

Setting boot order is dependent upon your server vendor and the process may vary slightly depending on which server platform is utilized within an environment. In the screenshot below (Figure XX) we will choose NEBULON NebVolume 0001 to boot.



console.png

Figure XX

Users will also need to ensure that the NEBULON device is set as the top device within the boot order. Depending on server vendor and management licensing, you may benefit by automating server workflows, e.g. by using the DMTF (Distributed Management Task Force) RedFish API to assist with remote server management and configuration.

Cloud-Init

Cloud-Init is the standard for Linux image customization in the cloud and in "cloud-like" environments which allows users to perform advanced image customizations. Nebulon advises using Cloud-Init to prepare boot images for Linux operating systems. If you have deployed a server instance using a public cloud provider, you have likely heard the term: userdata – think Cloud-Init. Cloud-Init supports both virtual machine (VM) and bare metal server customizations using YAML-based configuration files (cloud.cfg). For example, with Cloud-Init users can configure basic server startup tasks like setting the hostname, importing SSH keys, and package installation. Users may also pair Cloud-Init with Ansible to customize and automate even further. For example, run an Ansible module which creates a nPod pointing at the Cloud-Init enabled boot image which would lay down OS' and the appropriate settings. Upon completion, Ansible would take over to create a Kubernetes Cluster, install the Nebulon CSI driver and deploy a containerized application. Cloud-Init provides a platform agnostic method to deploying systems across public cloud and/or private cloud providers.

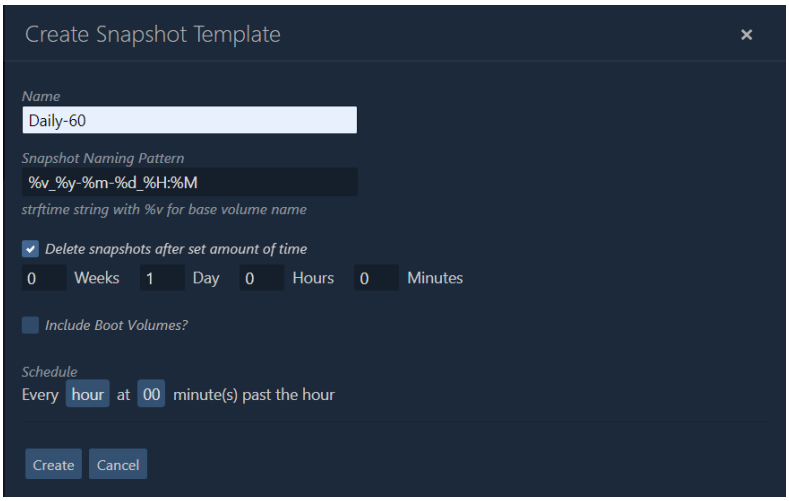
Nebulon nPod Snapshot Templates

Snapshots offer application owners a fast, easy, and complimentary recovery option for an existing backup strategy. Nebulon volume-based snapshots provide users with point-in-time recovery of application volumes and allow users to restore application data. Snapshots are not meant to completely

replace your backup software, but snapshots do offer an additional recovery option for both data and boot volumes which reside within a nPod. A use case would be when deploying a pre-application upgrade, you would initiate a snapshot and if things go awry, you may revert to that point-in-time.

Snapshot Templates offer users the ability to set common schedules, retention policies, and naming conventions for all applications running within a nPod and they can be referenced in nPod Configuration Templates.

Snapshot Templates are created and managed via API or via Nebulon ON graphical user interface. Navigate to **Admin -> nPod Management -> Snapshot Templates**. Figure z



snapshottemplate.png

Figure z: Create Snapshot Template

Nebulon Snapshot Export & Snapshot Clone

When an application operates within a nPod, the application has its data stored on a base volume. When the snapshot of a volume is initiated, either ad-hoc or via a Snapshot Template schedule, the snapshot data is stored on the same SPU as the base volume. Nebulon volume snapshots are compressed, deduplicated, and encrypted. In the end this means that snapshots are secure and have very little impact on overall volume capacity. The effect on capacity also works in reverse if you delete snapshots from Nebulon this does not have a direct correlation to reclaimed capacity. Nebulon volume snapshots are immutable restore points – meaning they are read only, even when they are exported to an application server within a nPod. To restore data, users may clone a volume snapshot and recover the needed data from the snapshot. The clone is a read-writeable version of a snapshot.

Nebulon Snapshot Naming Pattern

Snapshots are automatically named according to the format pattern specified. The format specification follows the familiar Linux strftime() function. It formats time according to the format specification and may contain special character sequences, each of which is introduced by a '%' character and terminated by some other character. All other character sequences are ordinary character sequences. Special characters and their replacement are shown in the list below.

Character	Replacement

%a	The abbreviated name of the day of the week according to the current locale
%A	The full name of the day of the week according to the current locale
%b	The abbreviated month name according to the current locale
%B	The full month name according to the current locale
%c	The preferred date and time representation for the current locale
%C	The century number (year/100) as a 2-digit integer
%d	The day of the month as a decimal number (range 01 to 31)
%D	Equivalent to %m/%d/%y
%e	Like %d, the day of the month as a decimal number, but a leading zero is replaced by a space
%E	Modifier: use alternative format (see below)
%F	Equivalent to %Y-%m-%d (the ISO 8601 date format)
%G	The ISO 8601 week-based year (see NOTES) with century as a decimal number. The 4-digit year corresponding to the ISO week number (see %V). This has the same format and value as %Y, except that if the ISO week number belongs to the previous or next year, that year is used instead.
%g	Like %G, but without century, that is, with a 2-digit year (00–99)
%h	Equivalent to %b
%H	The hour as a decimal number using a 24-hour clock (range 00 to 23)
%I	The hour as a decimal number using a 12-hour clock (range 01 to 12)
%j	The day of the year as a decimal number (range 001 to 366)
%k	The hour (24-hour clock) as a decimal number (range 0 to 23); single digits are preceded by a blank (See also %H)
%l	The hour (12-hour clock) as a decimal number (range 1 to 12); single digits are preceded by a blank. (See also %I)
%m	The month as a decimal number (range 01 to 12)
%M	The minute as a decimal number (range 00 to 59)

%n	A newline character
%O	Modifier: use alternative format (see below)
%p	Either "AM" or "PM" according to the given time value, or the corresponding strings for the current locale. Noon is treated as "PM" and midnight as "AM"
%P	Like %p but in lowercase: "am" or "pm" or a corresponding string for the current locale
%r	The time in am or pm notation. In the POSIX locale this is equivalent to %I:%M:%S %p
%R	The time in 24-hour notation (%H:%M). For a version including the seconds, see %T below.
%s	The number of seconds since the Epoch, 1970-01-01 00:00:00 +0000 (UTC)
%S	The second as a decimal number (range 00 to 60). The range is up to 60 to allow for occasional leap seconds
%t	A tab character
%T	The time in 24-hour notation (%H:%M:%S)
%u	The day of the week as a decimal, range 1 to 7, Monday being 1. See alsop %w
%U	The week number of the current year as a decimal number, range 00 to 53, starting with the first Sunday as the first day of week 01. See also %V and %W.
%v	**The name of the Nebulon base volume**
%V	The ISO 8601-week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the new year. See also %U and %W
%w	The day of the week as a decimal, range 0 to 6, Sunday being 0. See also %u
%W	The week number of the current year as a decimal number, range 00 to 53, starting with the first Monday as the first day of week
%x	The preferred date representation for the current locale without the time
%X	The preferred time representation for the current locale without the date
%y	The year as a decimal number without a century (range 00 to 99)
%Y	The year as a decimal number including the century
%z	The +hhmm or -hhmm numeric time zone (that is, the hour and minute offset from UTC)

%Z	The time zone name or abbreviation
%+	The date and time in date(1) format
%%	A literal '%' character

Monitoring

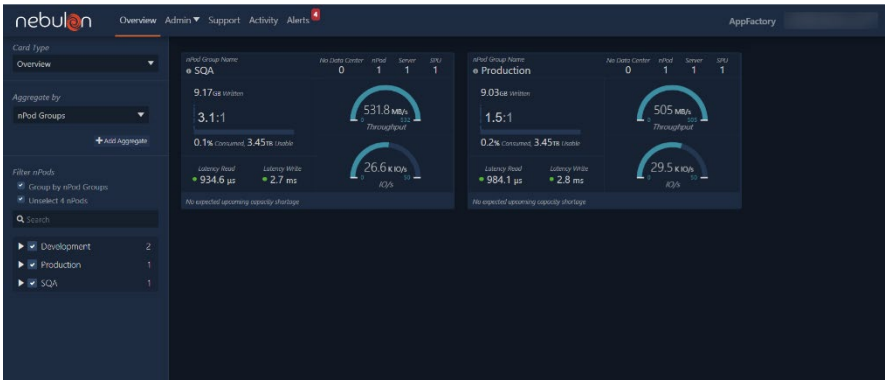
Reporting Metrics

Nebulon ON, the cloud-based control-plane, continuously receives information from on-premises Nebulon Services Processing Units (SPUs) via the 1GbE management interface. Data points, metadata, performance information and environment telemetry data are fed into the Nebulon ON cloud where AI / ML is applied. Resulting information is displayed in multiple places in the Nebulon ON GUI and is accessible via the Nebulon SDKs.

The Overview tab within the Nebulon ON GUI displays performance views in the form of cards. There are many card options available:

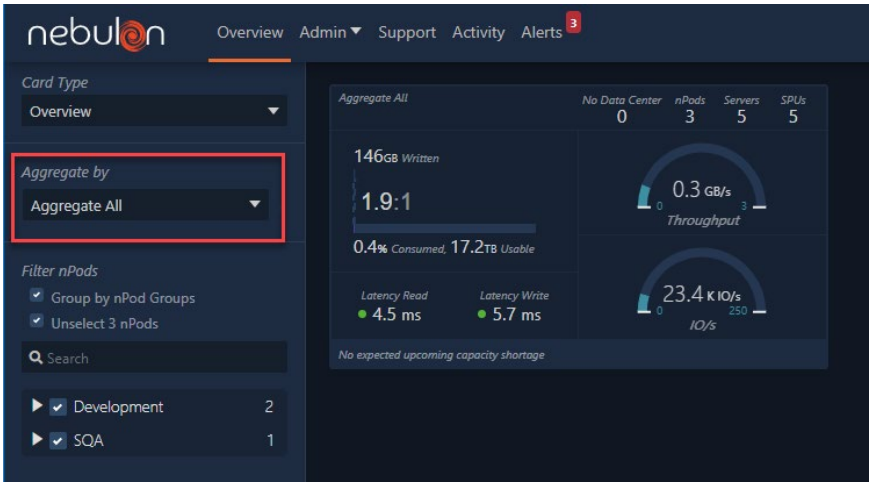
- Overview
- Performance
- Capacity
- Application

Depending on selection, the cards will display either Overview (Performance and Capacity), Performance, Capacity, or Application details about the Aggregate in which you have selected. Once a nPod is in place, each SPU reports storage, application server (host), application performance, and capacity metrics to Nebulon ON. This allows a consolidated view of all SPU metrics in one single location. For example, users may display Overview information about an Aggregate of all Datacenters or all nPods or a combination of Datacenters and nPods to further distill the information.



overview.png

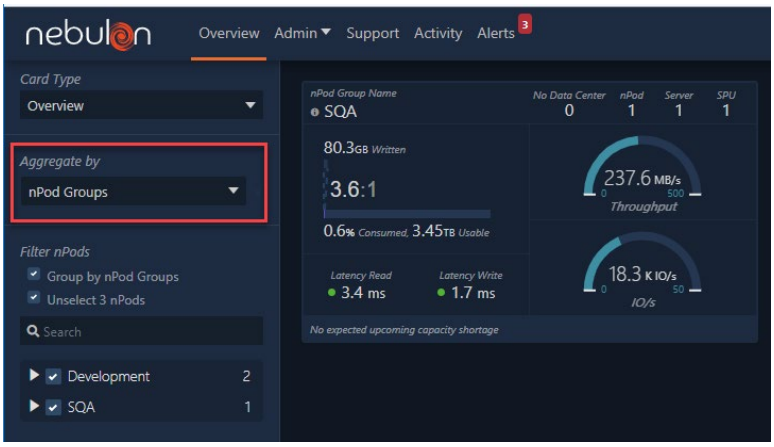
A singular view of storage metrics for the entire enterprise enables Nebulon to provide reporting at any level of aggregation required. Rollups of the storage usage for the entire enterprise down to the stats for an individual card can be surfaced in Nebulon ON.



aggregateall.png

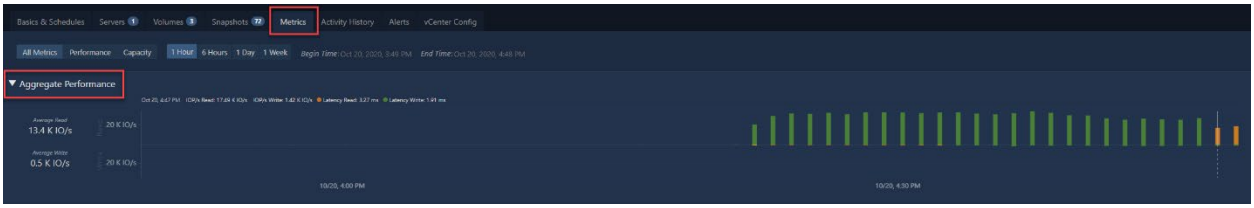
Organization or enterprise-wide metrics are helpful in determining overall allocation and consumption of capacity and performance. Capacity metrics are fed into an analytics engine to help determine macro-scale utilization of storage resources. This information can be fed into buying cycles for the entire business.

Metrics can also be aggregated by individual nPod groups. In this view, metrics from each nPod group, or a subset of nPod groups are visible.



npodgroup.png

Performance and Capacity metrics are available from an individual nPod with further details and the option to explore historical trends of metrics which can be useful for troubleshooting or evaluating the load of a nPod over time. The illustration below shows the Aggregate Performance which rolls storage IOPS and latency into a single chart. The top portion of the chart details read performance and the bottom portion of the chart details write performance.



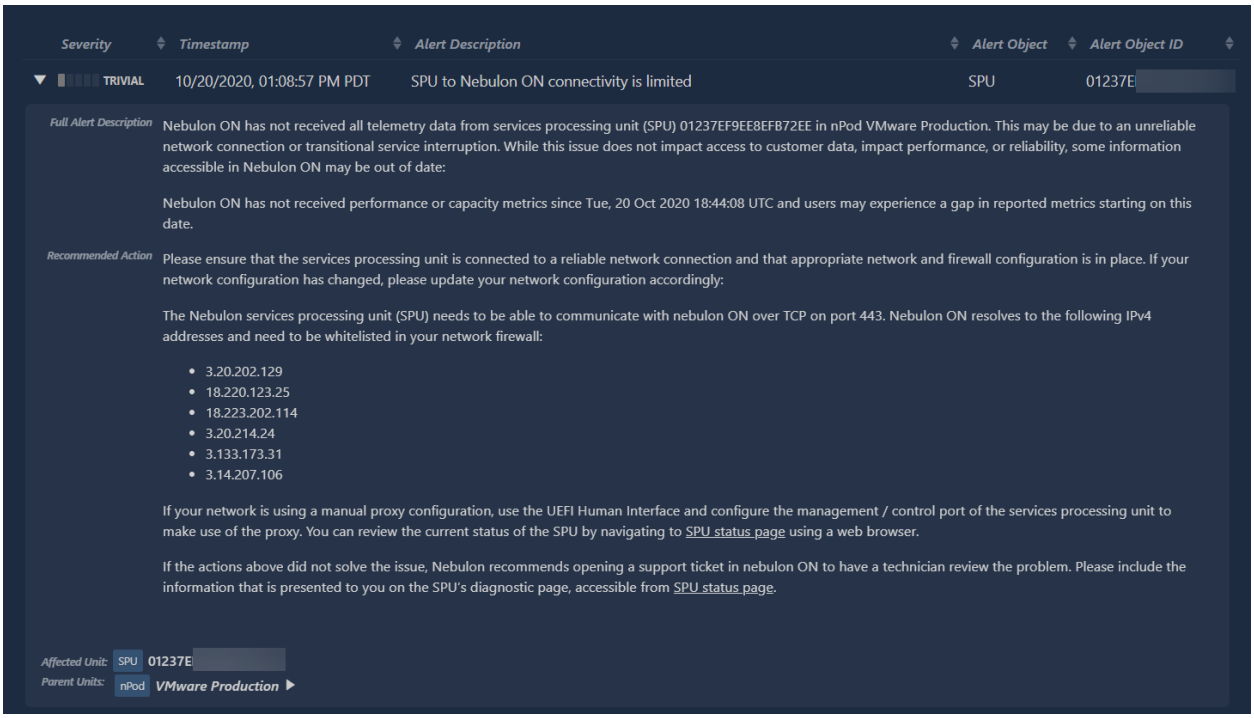
aggregatemetrics.png

Also, from within the nPod Metrics view users can dive into Throughput as depicted in the illustration below. The top portion of the graph shows read throughput where the bottom portion of the graph shows write throughput for the given period.

Alerts & Notifications

Alerting and monitoring when issues occur is crucial to ensuring that applications are available 24x7. With Nebulon ON, users can customize the way they receive their alerts and how they are notified of problems. For example, notifications are configured on a user-by-user basis and the delivery methods include e-mail as a daily summary digest or you may be notified by e-mail as soon as the alert is raised. Users may also define custom Webhooks to implement their own notification logic. For example, they can use Webhooks to integrate with common messaging and support platforms which support like PagerDuty, Slack and Microsoft Teams for example.

Alerts through Nebulon ON are viewed via the Alerts tab. Alerts are structured with a description and recommended actions to take for remediation. Alerts also contain the affected and parent resource. Below is a sample Alert for 'SPU to Nebulon ON connectivity is limited.' Figure blah



alert.png

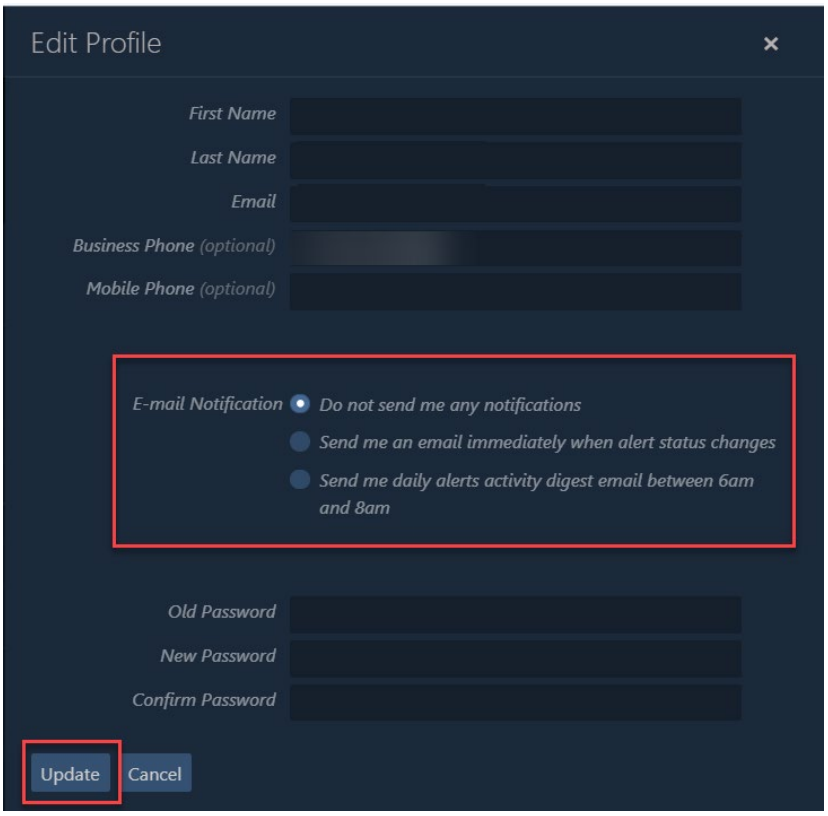
Figure blah

E-mail

E-mail notifications are configured at a user profile level within Nebulon ON. Notifications are driven from the cloud which means that users do not need to add any on-premises SMTP information as the e-mail send originates from Nebulon ON. Users simply need to make sure they have an e-mail address specified in their account and have opted in for notifications as shown. Figure A & B below shows where users can update their profile settings.

editprofile.png

Figure A

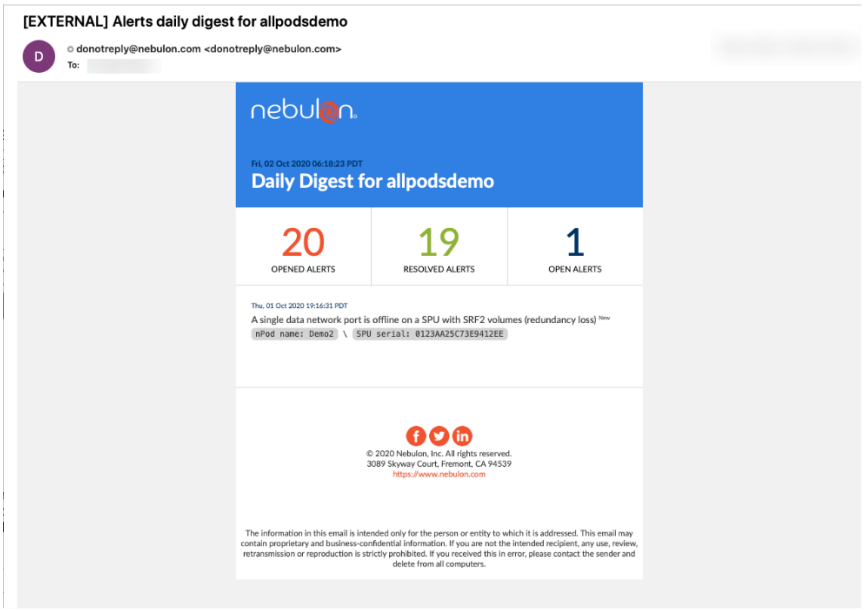


notifications.png

Figure B

By default, e-mail notifications are disabled. Alternatively, users may choose to receive email notifications immediately when alert status changes or they may choose to receive a daily email digest between 6am and 8am, local time.

A sample e-mail digest is included below (Figure C). Users will want to whitelist donotreply@nebulon.com to ensure that e-mail notifications are delivered successfully.



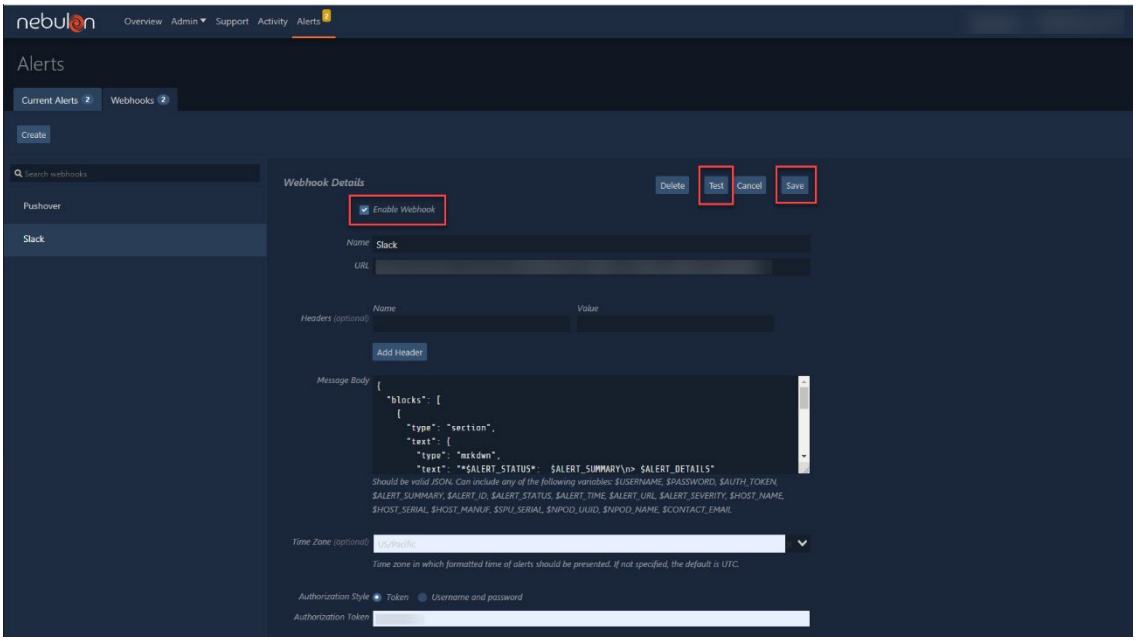
digest.png

Figure C

Webhooks

Webhooks are a user-defined, event driven method of delivering a message to an HTTP endpoint. Webhooks offer a modern approach to delivering system notifications. Common messaging and support platforms which support Webhooks are PagerDuty, Slack and Microsoft Teams.

The illustration below (Figure D) walks a user through the Nebulon ON Webhooks tab where the creation, editing, testing, and customization of Webhooks is completed. When configuring Webhooks the target application – ex: PagerDuty, Slack, Microsoft Teams – will provide users with the HTTP URL as well as the authorization token or Username and Password to be used.

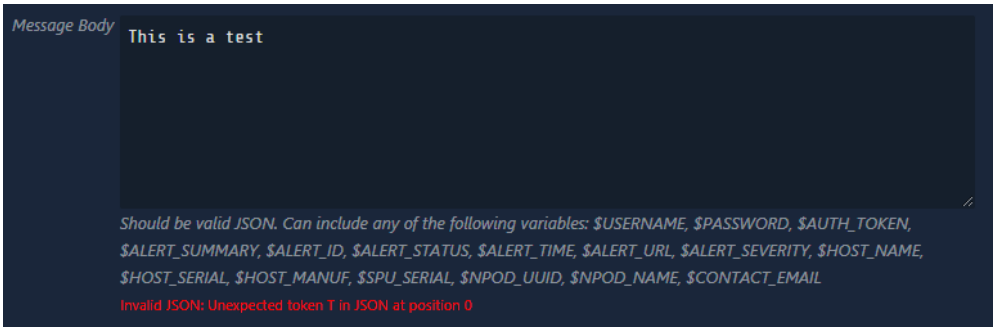


webhook.png

Figure D

Upon completion and prior to saving, users may 'Test' (Step 5) their Webhook configuration. Testing ensures that Webhook notifications are configured correctly and operating as expected prior to saving. The benefit is that users can ensure that their JSON is providing the correct and desired output.

When the Message Body of the Webhook contains invalid JSON, Nebulon ON will provide a notification. Figure E.



webookerror.png

Figure E

Upon completion, users may go back to their application to view the Webhook notifications.

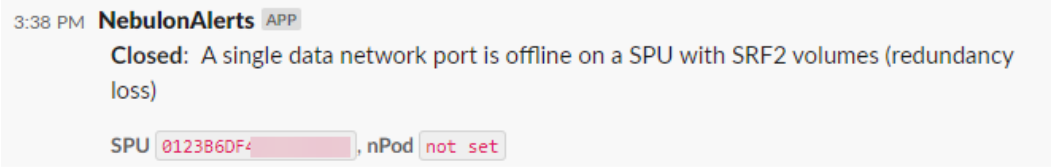
JSON Tags

Below are supported JSON tags which users may include within the Webhook message body.

$ALERT_SUMMARY	Summary of the alert status *(e.g.: physical drive 12345 failed)*
$ALERT_ID	ID of alert *(e.g.: 1234)*.
$ALERT_DETAILS	Full details of the alert
$ALERT_TIME	Time in number of seconds since epoch of when alert was created
$ALERT_URL	URL of the alert *(e.g.: https://on.nebulon.com/alerts/?alert_id=123456)*.
$ALERT_SEVERITY	Severity of the event (e.g. trivial, minor, major, etc.)
$HOST_NAME	Host name (if known) of the server in question
$HOST_SERIAL	Host serial (if known) of the server in question
$HOST_MANUF	Host manufacturer of the server in question
$SPU_SERIAL	SPU serial number related to the described alert
$NPOD_UUID	nPod UUID related to the described alert (if present)
$NPOD_NAME	nPod name related to the described alert (if present)
$ORG_UUID	Org UUID related to the alert (optional, but could be useful if the customer decides to integrate with a 3rd party system, e.g. an OEM would then have this information to do more)
$ORG_NAME	Organization name related to the alert
$CONTACT_EMAIL	Email address of the person registered to be the contact for the datacenter

Sample JSON for Slack notifications

Below, Figure xxxxx, is a sample in which Nebulon ON is configured to send Webhook notifications to a Slack channel.



slack.png

Figure xxxxx

The sample JSON body below will deliver a notification containing the $ALERT_STATUS, $ALERT_SUMMARY and $ALERT_DETAILS to a custom Slack integration. The full details on composing a

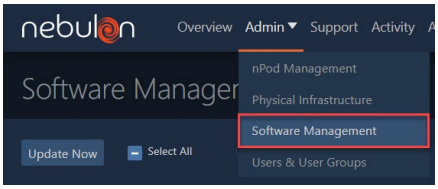
message for Slack can be retrieved from the Slack documentation at https://api.slack.com/messaging/managing.

```
{
  "blocks": [
    {
      "type": "section",
      "text": {
        "type": "mrkdwn",
        "text": "*$ALERT_STATUS*:  $ALERT_SUMMARY\n> $ALERT_DETAILS"
      }
    },
    {
      "type": "context",
      "elements": [
        {
          "type": "mrkdwn",
          "text": "*SPU* `$SPU_SERIAL`, *nPod* `$NPOD_NAME`"
        }
      ]
    }
  ]
}
```

Software Management

smartInfrastructure improves application server fleet management that users may be used to through the existing experience. Management and GUI based updates are provided as a service to customers, much like Gmail or Microsoft 365. This means that with Nebulon ON, the cloud-based control plane is always up to date with the latest software and Nebulon provides customers with frequent, but transparent updates. Example updates to the control plane include API enhancements, workflow improvements within Nebulon ON, and additional alerting mechanisms, just to name a few.

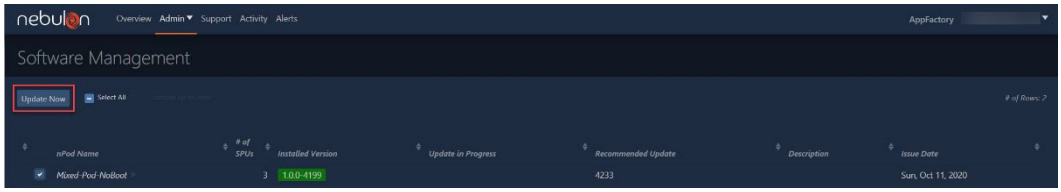
Data plane, nebOS, and updates to the Nebulon Services Processing Unit (SPU) can be deployed at scale, in a simple manner through Nebulon ON Software Management (Figure xx) interface. Whether users manage an application portfolio with 1 nPod or 50 nPods, updates are deployed and managed in the same manner.



softwaremanagement.png

Figure xx

Security of on-premises customer data and control actions initiated through Nebulon ON Security Triangle are a top priority. The image below illustrates (Figure xxx) an environment which has a nPod with an applicable update to apply. The user who is authenticated with Nebulon ON and has the appropriate role can complete the Security Triangle and recommended updates are shown within Nebulon ON. When a user is not within the corporate firewall and unable to complete the Security Triangle, updates will not display within Nebulon ON.



updatenow.png

Figure xxx

When updating a nPod with nebOS, all SPUs within a nPod must be able to communicate with Nebulon ON. When attempting to update nebOS with an SPU that is not able to communicate with Nebulon ON, the update will fail with an error message (Figure xxxx).

updateerror.png

Figure xxxx

To ensure compliance with the latest APIs within Nebulon ON the cloud will automatically, after 24 hours, keep SPUs that are not in a nPod updated to the latest software versions of nebOS. This auto-upgrade feature will only function whenever SPU-equipped servers are connected to power, network and powered on. Otherwise, if servers and SPUs are left powered-off for extended periods of time they will require updating whenever being brought online.

API and Automation Use-Cases

Nebulon provides a fully featured API-based architecture with SDKs in Python, Go and C# and a PowerShell module. Automation is a key capability of the Nebulon platform which allows the ability to plug Nebulon into your existing automation workflows or DevOps practices. Leveraging the Nebulon

Python SDK and PowerShell module as well as Ansible for automation is simple. Below are a few nPod specific examples to help get started.

Python SDK

The following examples will help you get started quickly using the Nebulon Python SDK. Nebulon has published example Python scripts which focus on day 1 and day 2 operations. For example, nPod creation, Volume creation and full-stack environment creation. The Nebulon Python SDK provides full support for Python version 3.

Below are a few code snippets which will help you get up and running with the Nebulon Python SDK. For full examples visit the Nebulon GitHub repositories at https://github.com/nebulon.

For the full SDK documentation visit the Nebulon Python SDK documentation at:

https://nebulon.github.io/nebpyclient/index.html

Installation

The Nebulon Python SDKclient is available through the Python Package Index with the name nebpyclient.

```
pip install nebpyclient
```

or

```
python3 -m pip install nebpyclient
```

Examples

The following code snippet will import the required Nebulon Python SDK client class:

```
#Import Python nebulon module
from nebpyclient import NebPyClient
```

With the client imported, users will need to establish a connection with Nebulon ON and provide login credentials:

```
# Customize username and password for your organization
client = NebPyClient(username = "username",password="password")
```

All mutations and queries are accessible through the "client" object. As an example, you can use the SDK to create a new nPod. First, query for a Configuration Template for the new nPod. This example uses the Configuration Template with the name "vSphere Default":

```
# Get the existing vSphere Default Template
npodtemplate = client.get_npod_template('vSphere Default')
if npodtemplate is None:
  print("Configuration Template does not exist")
  exit
```

Next, the list of available application servers is retrieved, and their SPU information is recorded.
Get list of hosts from Nebulon ON and get the list of nPod Groups:

```
# Store outputs in variable
```

```
hostoutput = client.get_hosts()
spu_serials = []
for host in hostoutput:
  for spu in host.spus:
    spu_serials.append(spu)
```

When creating an nPod, the nPod needs to be placed in a nPod Group. The relevant information is retrieved in the next step. In this example, the script is extracting the UUID of the nPod Group with the name “Engineering”.

```
npodgroups = client.get_npod_groups()
npodgroup_uuid = None
for npodgroup in npodgroups:
  if 'Engineering' == npodgroup.name:
    npodgroup_uuid = npodgroup.uuid
    break
if npodgroup_uuid == None:
  print(“nPod Group with name Engineering not found”)
  exit
```

Now the new nPod with the name “new nPod” can be created:

```
new_npod = client.new_npod(
  name = “new nPod”,
  npod_group_uuid = npodgroup_uuid,
  spu_serials = spu_serials,
  template_uuid = npodtemplate.uuid,
  ignore_warnings = True
)
print("nPod SUCCESSFULLY CREATED!")
```

NebPowerAutomation - PowerShell Module

Nebulon provides support for PowerShell via the NebPowerAutomation module. The following sections will help you get started quickly using the Nebulon PowerShell module.

Getting Started with NebPowerAutomation Module – Installation & Examples

Users may download the NebPowerAutomation installation package from here…

Examples

1.) Import NebPowerAutomation PowerShell Module

```
Import-Module ./NebPowerAutomation.dll
```

2.) List NebPowerAutomation Commands

```
Get-Command -Module NebPowerAutomation

CommandType     Name                                Version    Source
```

Cmdlet	Get-NebAlert	1.0.0.0	NebPowerAutomationpython
Cmdlet	Get-NebDataCenter	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebHost	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebKeyValues	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebLab	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebLun	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebNPod	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebNPodGroup	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebNPodGroupCount	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebNPodTemplate	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebNPodTemplateCount	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebPhysicalDrives	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebRow	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebSpu	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebulonConnectionState	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebUser	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebUserCount	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebUserGroup	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebUserGroupCount	1.0.0.0	NebPowerAutomation
Cmdlet	Get-NebVolume	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebAddress	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebContact	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebDataCenter	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebLab	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebLun	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebNPod	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebNPodGroup	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebRow	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebulonConnection	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebUser	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebUserGroup	1.0.0.0	NebPowerAutomation
Cmdlet	New-NebVolume	1.0.0.0	NebPowerAutomation
Cmdlet	Ping-NebSpu	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebDataCenter	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebKeyValue	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebLab	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebLun	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebNPod	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebNPodGroup	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebRow	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebulonConnection	1.0.0.0	NebPowerAutomation

Cmdlet	Remove-NebUser	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebUserGroup	1.0.0.0	NebPowerAutomation
Cmdlet	Remove-NebVolume	1.0.0.0	NebPowerAutomation
Cmdlet	Rename-NebDataCenter	1.0.0.0	NebPowerAutomation
Cmdlet	Rename-NebLab	1.0.0.0	NebPowerAutomation
Cmdlet	Rename-NebRow	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebDataCenter	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebHost	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebKeyValue	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebLab	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebRow	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebUser	1.0.0.0	NebPowerAutomation
Cmdlet	Set-NebUserGroup	1.0.0.0	NebPowerAutomation

3.) Connect to Nebulon ON

```
New-NebulonConnection -Username xyz -Password SuperSecurePassword
```

4.) Get list of SPUs and Select-Object based upon HostSerial, Version and PhysicalDriveCount:

```
Get-NebSPU | Select-Object HostSerial, Version, PhysicalDriveCount | Format-Table
```

5.) Get list of nPod named VMware Production:

```
Get-NebNPod | Where-Object Name -eq "VMware Production" | Format-List
```

6.) Get list of available SPUs:

```
Get-NebSpu -Unused | Select-Object Serial
```

7.) Create an nPod named PowerShellRocks based upon the Configuration Template ESX_70:

```
New-NebNPod -Name PowerShellRocks -NPodGroupGuid 16d71d7b-691e-4582-917d-e760194d0870 -TemplateGuid c9f74de2-fe95-4245-a992-b53996b9e61d -SpuSerials 0123667B10041AFFEE -IgnoreWarnings
```

References

Beach, B. (n.d.). *Backblaze Open Sources Reed-Solomon Erasure Coding Source Code*. Retrieved from https://www.backblaze.com/blog/reed-solomon/