Lightbits Installation Guide
Lightbits Version: v3.3.2

Lightbits Labs

August 2023
Contents

Lightbits™ v3.x Installation and Configuration Guide 4

About the Installation Guide 5

Lightbits Cluster Overview 6
Lightbits Cluster Topology 6
Planning for the Lightbits Cluster Software Installation 8
Installation Files Backup 8
Lightbits Cluster Architecture 8
Nodes 9
Failure Domains 9

Lightbits Cluster Software Installation Process 11
Installation Preparation 11
General System Requirements 12
Lightbits Server Prerequisites 12
Required Ports for Installation and Operation 14
Testing Ports 14
Relevant Lightbits Support Documentation 15

Lightbits Cluster Software Installation 15
Before You Begin 16
Connecting to the Lightbits Software Repository 17
Verifying Network Connectivity for the Servers in the Cluster 18
Configuring the Ansible Environment 19
Prepare Installation Workstation (Ansible Controller) 20
Ansible Method 20
Docker Method 21
Copying the Ansible Environment Tarball 22
Inventory Structure and Adding the Ansible Hosts File 23
Multi-Tenancy 26
Configuring Global Variables in Ansible 28
NTP vs Chrony 29
Use Lightos Kernel 30
Single IP Dual NUMA Configuration 30
Verifying Hosts Connection 30
Testing Connectivity via Ping 30
Testing Connectivity via SSH 31
Defining Configuration Files for Each “Ansible Host” (Server) in the Cluster 32
Defining Failure Domains 35
Setting the SSD Configuration 37
Confirming the Required Directory Structure 38
Running the Ansible Installation Playbook to Install Lightbits Cluster Software 39
Lightbits Cluster Installation Process 39
Ansible Installation Method 39
Prebuilt Ansible Docker Installation Method 40
Verify Successful Installation 41
Post Installation Steps 42
Back Up Important Content 42
Check Cluster Health 42

Linux Cluster Client Software Installation 45
Connecting to the Cluster Client DEB Repository 45
Connecting to the Cluster Client RPM Repository 46
Installing the Latest Kernel on the Client 46
Installing the Latest Kernel on AlmaLinux/Red Hat 8 47
Installing the Latest Kernel on Centos 7.9 47
Configuring the Client to Boot from the New Kernel 47
Lightbits™ v3.x Installation and Configuration Guide

The Lightbits™ cluster storage solution distributes services and replicates data across different Lightbits servers to guarantee service and data availability when one or more Lightbits servers experience transient or permanent failures. A cluster of Lightbits servers replicates data internally and keeps it fully consistent and available in the presence of failures. From the perspective of clients accessing the data, data replication is transparent, and server failover is seamless.

Lightbits also protects the storage cluster from additional failures not related to the SSDs (e.g., CPU, memory, NICs) failures, software failures, network failures, or rack power failures. It provides additional data security through in-server Erasure Coding (EC) that protects servers from SSD failures and enables non-disruptive maintenance routines that temporarily disable access to storage servers (e.g., TOR firmware upgrades).
About the Installation Guide

This installation guide is for system administrators who are installing the Lightbits storage server software. It includes instructions for installing the Lightbits cluster software, installing the cluster client software, and connecting clients to the Lightbits Cluster.

Use the information in this installation guide to:

- Plan for the Lightbits cluster software installation in your environment.
- Successfully install the software so that a cluster of Lightbits servers is ready for use.

Lightbits Labs™ recommends that you follow the installation instructions in the order that they are written to ensure a successful installation.
Lightbits Cluster Overview

This section provides you with information about the major components of the Lightbits cluster software solution and how they work together.

It also contains recommended best practice tips for collecting information required to use the automated installation script. For the installation script to download and install the Lightbits software onto your system’s storage nodes, you must have details about your specific environment—such as your specific networking details.

Lightbits Cluster Topology

The following is a basic diagram that shows the components and resources required to automatically install Lightbits software onto your servers in your data center.

**Note:** There are two types of installation methods: the online installation method, which connects to online repositories to download the Lightbits software; and the offline installation method, which grabs the Lightbits software files locally from the Ansible host. Note that for the offline method, Lightbits will provide the software files in advance. This guide mainly covers the online installation method, but notes are provided on what differs with the offline installation method as well.

![Lightbits Cluster Resources](image-url)

**Figure 1: Lightbits Cluster Resources**

Based on the numbers next to each component or resource in the diagram, see the following table for a description of the components and resources in the Lightbits cluster topology diagram.

**Lightbits Cluster Topology Components Table**

<table>
<thead>
<tr>
<th>#</th>
<th>Component or Resource</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>dl.lightbitslabs.com</td>
<td>Lightbits supplied configuration files and installation tools via remote repository. The Lightbits software is maintained in a password-protected software repository, referred to as “The Lightbits Repo”.</td>
</tr>
<tr>
<td>2</td>
<td>dl.lightbitslabs.com</td>
<td></td>
</tr>
<tr>
<td>#</td>
<td>Component or Resource</td>
<td>Description</td>
</tr>
<tr>
<td>----</td>
<td>---------------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>3</td>
<td>Preferred Linux repo.org</td>
<td>There are many publicly available Linux repositories that are already configured in your environment. Some standard tools might need to be downloaded from these repos or updated Linux kernel files to allow your clients to perform NVMe/TCP functions. These standard files are not available from Lightbits but are part of the core operating system.</td>
</tr>
<tr>
<td>4</td>
<td>Public Network Time Protocol Server</td>
<td>Lightbits cluster nodes remain in sync using NTP or Chrony. This is automatically configured by the Ansible installation script. However, a custom time service configuration is possible.</td>
</tr>
<tr>
<td>5</td>
<td>Management Network</td>
<td>To connect and configure a given server, the standard Secure Shell (SSH) protocol is used. Each server must be reachable and is configured over this management network. This network is separate from the network that Lightbits will send and receive application data over. It is also possible to use the same network for both management and data networks.</td>
</tr>
<tr>
<td>6</td>
<td>Data Network</td>
<td>Acts as the interconnection between the enterprise’s “Clients” or “Application Servers”. This is a separate network from the management network network and carries all of the NVMe/TCP data traffic.</td>
</tr>
<tr>
<td>7</td>
<td>Cluster Installation Workstation</td>
<td>This server is where you download the Lightbits installation software. This is composed of an Ansible script. This server must be outside of the planned cluster. The server will automatically download the Lightbits cluster software files from the <a href="http://dl.lightbitslabs.com">dl.lightbitslabs.com</a> repo to each of the Lightbits cluster servers. <strong>Note:</strong> For information about the system requirements for the installation workstation, see System Requirements.</td>
</tr>
<tr>
<td>8</td>
<td>Lightbits Cluster Servers</td>
<td>The cluster servers store the application data. Each node is what essentially makes up the data storage portion of a Lightbits cluster. All of the Lightbits software from the “The Lightbits Repo” are installed on these servers.</td>
</tr>
</tbody>
</table>
Planning for the Lightbits Cluster Software Installation

At a very high level, the following command automatically installs a Lightbits cluster:

```bash
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/deploy-lightos.yml
```

When this command completes, you will have a Lightbits cluster.

To use this Ansible command successfully, you will need to provide the Ansible software with information about your data center’s specific environment. This means you should gather some details and enter them into text files that Ansible uses during the Lightbits installation operations.

However, to use this Ansible command successfully, you will need to prepare the environment and provide the Ansible software with information about your data center’s specific environment into specific configuration files. This includes the `hosts` file seen above, and other Ansible yaml files. This means that you should gather some details and enter them into text files that Ansible uses during the Lightbits installation operations.

The remainder of this Installation Guide is organized into the following flow: * Section 3 covers preparing the environment. * Section 4 covers setting up, configuring, and running the Ansible installation - which deploys installs the Lightbits cluster. * Section 5 and 6 cover connecting a client to a live cluster. * Section 7 covers additional troubleshooting. * Section 8 covers additional information that can be useful for other sections.

The installation process generally follows the path in the following diagram. When Ansible runs, it reads the text files you configured (in step 3), connects to the Lightbits software repository (in step 1), and downloads Lightbits software onto each storage server that will exist in the cluster (in step 4).

### Installation Files Backup

During the installation process, Ansible generates certification files required by etcd used by Lightbits, the API service, and Admin. These files are not critical but are very important in case of changes required in the cluster (adding/replacing/recovering a server).

Lightbits’ recommendation is to back up this directory: `lightos-certificates` (using Ansible), or `lightos-certificates` (using lb-docker) - for future use, if required.

If the certificate files are lost, Lightbits can help work through a procedure of regenerating those files.

Additionally, we advise backing up the Ansible configuration files in the `ansible` directory and the created jwt files: `lightos-system-jwt` and `lightos-default-admin-jwt`.

### Lightbits Cluster Architecture

The Lightbits cluster storage solution distributes services and replicates data across different Lightbits servers. This guarantees service and data availability when one or more Lightbits servers experience transient or permanent failures. A cluster of Lightbits servers replicates data internally and keeps it fully consistent and available in the presence of failures. From the perspective of clients accessing the data, data replication is transparent, and server failover is seamless.
Lightbits also protects the storage cluster from additional failures not related to the SSDs (e.g., CPU, memory, NICs), software failures, network failures, or rack power failures. It provides additional data security through in-server Erasure Coding (EC) that protects servers from SSD failures and enables non-disruptive maintenance routines that temporarily disable access to storage servers (e.g., TOR firmware upgrades).

The following sections describe the failure domain and volume components used in the Lightbits cluster architecture.

**Note:** For more information about Lightbits cluster architecture, see the Deploying Reliable High-Performance Storage with Lightbits Whitepaper.

### Nodes

Each server can be split into multiple logical nodes. Each logical node owns a specific set of SSDs and CPUs, and a portion of the RAM and NVRAM. The physical network can be shared or exclusive per node.

Nodes can be across NUMAs or per NUMA. There is no relation or limitation between a logical node and the NUMA of the resources used by the logical node.

Each storage server runs a single Node Manager service. The service controls all the logical nodes of the storage server.

**Note:** The current Lightbits release only supports up to two logical nodes per server. The single logical node deployment is commonly referred to as “single instance, node or NUMA deployment”. Dual logical node deployment is referred to as “dual instance, node or NUMA deployment”.

### Failure Domains

Users define FDs based on data center topology and the level of protection that they strive to achieve. Each server in the cluster can be assigned to a set of FDs.

An example of an FD definition is separating racks of servers by FD labels. In this case, all servers in the same rack are assigned the same FD label, while servers in different racks are assigned distinct labels (e.g., FD label = rack ID). Two replicas of the same volume will not be located on two nodes in the same rack.

The system stores different replicas of the data on separate FDs to keep data protected from failures.

The definition of an FD is expressed by assigning FD labels to the storage nodes. Single or multiple FD labels can be assigned to every node.
Another example of an FD definition is grid topology, in which every node is assigned a label of a row and a label of a column. In this case, the volume is not stored on two servers that are placed on the same row or on the same column.

**Note:** Per the previous section, servers can be configured using a single or dual instance. The same Failure Domain rules apply to dual instance, in addition to the fact that volumes will never be placed on a different node of the same server. This is because any server failure will usually affect both nodes.

For more information on Failure Domain configuration, see the Lightbits Administration Guide.
Lightbits Cluster Software Installation Process

The process of installing Lightbits products includes the installation of Lightbits software on the storage server. It can also include the installation of a new kernel on the client if the client’s kernel version is less than v5.3.5.

The following chart summarizes the steps for completing the Lightbits cluster software installation and required actions on the clients.

![Diagram of Lightbits Cluster Software Installation Process]

Figure 3: Lightbits Clustering Multipath Replication Design

Lightbits recommends you complete each of these steps in the order that they are written to ensure a successful software installation and connection between the Lightbits Storage Server and the clients.

Note: To complete the installation process, you must have the **Lightbits Installation - Customer Addendum** that was sent to you by Lightbits. The customer addendum contains customer-specific information and is referred to throughout the installation procedure.

**Installation Preparation**

Before you begin the installation, Lightbits recommends that you create a reference table to list the networking and server names you will use for your Lightbits cluster. The following is an example of a table you can use with the Configuring the Ansible Environment section.

**Installation Planning Table**

Note: The following represents a cluster with three Lightbits servers with a single client.

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Role</th>
<th>Management Network IP</th>
<th>Data NIC Interface Name</th>
<th>Data NIC IP</th>
<th>NVMe Drives</th>
</tr>
</thead>
<tbody>
<tr>
<td>server00</td>
<td>Lightbits Storage Server 1</td>
<td>192.168.16.22</td>
<td>ens1</td>
<td>10.10.10.100</td>
<td>6</td>
</tr>
<tr>
<td>server01</td>
<td>Lightbits Storage Server 2</td>
<td>192.168.16.92</td>
<td>ens1</td>
<td>10.10.10.101</td>
<td>6</td>
</tr>
<tr>
<td>server02</td>
<td>Lightbits Storage Server 3</td>
<td>192.168.16.32</td>
<td>ens1</td>
<td>10.10.10.102</td>
<td>6</td>
</tr>
<tr>
<td>client00</td>
<td>client</td>
<td>192.168.16.45</td>
<td>ens1</td>
<td>10.10.10.103</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note that during this installation, client00 will function as the Ansible installation host. However, any server will work that has connectivity over SSH to the management IP of each server.

2023 Lightbits Labs Proprietary And Confidential Under NDA Only
This table appears throughout this installation guide to help you follow the Lightbits installation process, to show the progress you have made to complete the installation, and to successfully configure a cluster of servers.

Additional relevant information about this cluster: * The servers are running AlmaLinux 8.7 with kernel 4.18.0-425.19.2.el8; however, other OSs are supported with different build releases. For additional information, see General System Requirements. * The Lightbits RHEL release will be installed. Our Red Hat build supports Red Hat and AlmaLinux. For additional information, see General System Requirements. * We will install in Single Instance/NUMA/node mode; however, if the servers have dual or more NUMAs, we could also do a dual instance/NUMA/node installation. Examples for this will be provided.

**Lightbits Cluster Installation Process**

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

**General System Requirements**

Before you begin installing the Lightbits product, you should be aware of the following installation considerations:

- The system administrator performing this installation must have the following permissions:
  - SSH accessibility (needed packages/permissions)
  - Root user permissions are required to complete the installation (can use normal user with sudo access).

- The Linux distribution that your clients use must have the NVMe/TCP client-side drivers. These drivers are included starting with Linux kernel v5.3.5 and above.
  - If your system’s Linux distribution does not include this kernel version or a later version, download back-ported NVMe/TCP client side drivers for specific kernels and distributions from the Lightbits drivers webpage.

**Lightbits Server Prerequisites**

Consider the following prerequisites for the storage servers that will host the Lightbits software.

- Lightbits recommends that you plan to use two networking interfaces on the Lightbits servers: one for management and another as a data interface. This is not required, as the data interface can function as both management and data. For dual instance/node/NUMA configurations, an additional network interface can be used.
- We support the following server-based OS installations: AlmaLinux/Red Hat. Note that the CentOS releases are end of life and will be deprecated.
- If persistent memory (Intel Optane or NVDIMM) is used, configure the pmem properly per different vendors’ servers. Please consult with the server vendor or with Lightbits for any additional questions. Please note: you should enable the memory interleaving in the BIOS/ueFI, and for Intel Optane, you should use the App Direct Mode.
- You must have Python v3.6 (or higher) installed on the Lightbits servers. Additionally, it is advised to have network-scripts, yum-utils and net-tools installed.

**Note:** In December 2020, the CentOS community and Red Hat announced the sunset of CentOS.

Example command to install:
$ yum install -y python3
$ yum install -y network-scripts
$ yum install -y yum-utils
$ yum install -y net-tools

- The Lightbits software kernel requires a boot partition with at least 512 MB available.
- To complete the installation process, you will need information from your version of the Lightbits Installation-Customer Addendum. If you do not have the customer addendum, contact a Lightbits representative to receive a copy.
- For more information about which Python version supports Ansible, see the Ansible Installation Guide.
- Lightbits releases are tested on Red Hat based distributions, including Rocky Linux and AlmaLinux. See the current distribution's release notes for full details on tested distributions.

The following table details the supported Lightbits distributions and kernels for older releases of Lightbits. Starting with Lightbits release v3.4.1, there is no longer any kernel dependency, although Lightbits recommends using the latest stable kernel available for your distribution.

<table>
<thead>
<tr>
<th>Lightbits Release</th>
<th>Release Type</th>
<th>Kernel Version</th>
<th>Supported OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3.2~b1363</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.3.2~b1362</td>
<td>RHEL</td>
<td>4.18.0-477.13</td>
<td>Alma, Red Hat 8.7</td>
</tr>
<tr>
<td>3.3.1~b1335</td>
<td>RHEL</td>
<td>4.18.0-477.13</td>
<td>Alma, Red Hat 8.7</td>
</tr>
<tr>
<td>3.3.1~b1336</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.3.1~b1334</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.2.1~b1253</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.2.1~b1252</td>
<td>RHEL</td>
<td>4.18.0-425.19.2</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.2.1~b1251</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.1.2~b1127</td>
<td>RHEL</td>
<td>4.18.0-425.3.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.1.2~b1130</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.1.2~b1125</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.1.1~b1119</td>
<td>RHEL</td>
<td>4.18.0-425.3.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.1.1~b1118</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.1.1~b1116</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.0.5~b1107</td>
<td>RHEL</td>
<td>4.18.0-372.32.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.5~b1105</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.5~b1102</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>3.0.4~b1085</td>
<td>RHEL</td>
<td>4.18.0-372.32.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.3~b1062</td>
<td>RHEL</td>
<td>4.18.0-372.26.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.3~b1061</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.3~b1059</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.22~b1031</td>
<td>RHEL</td>
<td>4.18.0-372.26.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.1~b1007</td>
<td>RHEL</td>
<td>4.18.0-372.19.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>3.0.1~b1004</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.20~b988</td>
<td>RHEL</td>
<td>4.18.0-372.19.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>2.3.19~b962</td>
<td>RHEL</td>
<td>4.18.0-372.16.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>2.3.18~b951</td>
<td>RHEL</td>
<td>4.18.0-372.13.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>2.3.17~b930</td>
<td>RHEL</td>
<td>4.18.0-305.12.1</td>
<td>Red Hat 8.4</td>
</tr>
<tr>
<td>2.3.17~b927</td>
<td>RHEL</td>
<td>4.18.0-372.9.1</td>
<td>Alma, Red Hat 8.6</td>
</tr>
<tr>
<td>2.3.17~b923</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.16~b887</td>
<td>RHEL</td>
<td>4.18.0-305.12.1</td>
<td>Red Hat 8.4</td>
</tr>
<tr>
<td>2.3.16~b886</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.14~b806</td>
<td>RHEL</td>
<td>4.18.0-305.12.1</td>
<td>Red Hat 8.4</td>
</tr>
<tr>
<td>2.3.14~b805</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.12~b793</td>
<td>GA</td>
<td>4.14.252...</td>
<td>CentOS 7.9</td>
</tr>
<tr>
<td>2.3.8~b664</td>
<td>GA</td>
<td>4.14.216...</td>
<td>CentOS 7.9</td>
</tr>
</tbody>
</table>
Note: For Lightbits RHEL releases, the kernel version shown must be pre-installed on the servers for the Ansible installation of Lightbits in order to work. For (soon to be sunsetting) Lightbits GA releases, the kernel version shown is installed on the servers by the Ansible installation.

Note: Due to the known kernel issue detailed here - and in order to avoid potential issues - for RHEL below 9.2 and/or upstream kernel below 6.2, Lightbits recommends using cgroup v1.

Required Ports for Installation and Operation

The Lightbits cluster software requires access to several ports to complete its installation and operate successfully.

Required Ports for Installation

The following table lists the required ports for the Ansible installation host to communicate with the storage servers, in order to successfully install the Lightbits cluster software:

<table>
<thead>
<tr>
<th>Component</th>
<th>Management/Data NIC</th>
<th>Port (TCP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ansible operation over SSH</td>
<td>Management</td>
<td>22</td>
</tr>
<tr>
<td>Installing RPM dependencies. Note that this should be installed with yum or dnf.</td>
<td>Management</td>
<td>443</td>
</tr>
</tbody>
</table>

Note: The Ansible installation host at a minimum requires its SSH port to be open.

Required Ports for Operation

The following table lists the default ports used by the Lightbits components:

<table>
<thead>
<tr>
<th>Component</th>
<th>Management/Data NIC</th>
<th>Port (TCP)</th>
<th>Default location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management CLI</td>
<td>Management</td>
<td>443</td>
<td>roles/etcd/defaults/main.yml</td>
</tr>
<tr>
<td>etcd peer port</td>
<td>Data</td>
<td>2380</td>
<td></td>
</tr>
<tr>
<td>Exporter port</td>
<td>Management</td>
<td>8090</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
<tr>
<td>Duroslight port Note: NVMe client connects to Duroslight via this port.</td>
<td>Data</td>
<td>4420, 8009</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
<tr>
<td>Replicator port Note: Other nodes connect for replication to the node via this port.</td>
<td>Data</td>
<td>22226</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
</tbody>
</table>

Note: If using Single IP Dual Numa configuration (see Single-IP-Dual-NUMA Configuration), open the above ports and two additional ports: 4421 and 22227. Duroslight will use 4420 and the additional 4421 port. Replicator will use 22226 and the additional 22227 port.

Testing Ports

See Open TCP Ports and Verify for more information on how to open and verify ports.

Port accessibility can be tested with the open-source Nmap program.

1. Install Nmap with the following command: `bash $ yum install -y nmap`  
   Note: If testing port accessibility from a non-rpm/yum based operating system, the installation will differ, but the commands below should still work, as nmap installs and relies on nc (netcat). >
2. You must have the netcat program running in listen mode on the server you are testing with the following command:

```
$ nc -l -p <port>
```

3. From another server, check the port’s accessibility with either of the following commands:

```
$ nc -v -z <ip> <start port>-<end port>
```

or

```
bash $ nc -v -u <ip> <start port>-<end port>
```

### Relevant Lightbits Support Documentation

This installation and configuration guide is part of a documentation set that provides complete information about using Lightbits products.

This document set includes the following Lightbits Support documentation.

<table>
<thead>
<tr>
<th>Document</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightbits Installation Guide (this document)</td>
<td>Contains the instructions to install the Lightbits cluster software, installs the Linux cluster client software, and then connects the cluster client to Lightbits.</td>
</tr>
<tr>
<td>Installation Guide - Customer Addendum</td>
<td>Includes customer-specific passwords to access installation files.</td>
</tr>
<tr>
<td>Lightbits Administration Guide</td>
<td>Provides detailed information about the operations you can perform using the Lightbits lbcli CLI command and REST API. <strong>Note:</strong> After you complete the installation process in this document, you should refer to the Administrator’s Guide for important management and automation instructions.</td>
</tr>
<tr>
<td>User’s Manual: Lightbits REST and CLI API</td>
<td>Lists the low level details for the REST API and CLI command usage. This document is typically used as a reference manual when building and administering the system. <strong>Note:</strong> See the Administrator’s Guide for detailed examples for using the REST API and CLI commands.</td>
</tr>
</tbody>
</table>

The following diagram shows how to use the documents to install, test, and maintain Lightbits products, and how the above referenced documents can be used to support the typical user experience.

![Figure 4: Lightbits Documentation Set](image)

### Lightbits Cluster Software Installation

**Installation Planning Table**

2023 Lightbits Labs

Proprietary And Confidential

Under NDA Only
The table below details a Lightbits cluster installed on three servers, with a connected client.

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Role</th>
<th>Management Network IP</th>
<th>Data NIC Interface Name</th>
<th>Data NIC IP</th>
<th>NVMe Drives</th>
</tr>
</thead>
<tbody>
<tr>
<td>server00</td>
<td>Lightbits Storage</td>
<td>192.168.16.22</td>
<td>ens1</td>
<td>10.10.10.100</td>
<td>6</td>
</tr>
<tr>
<td>server01</td>
<td>Lightbits Storage</td>
<td>192.168.16.92</td>
<td>ens1</td>
<td>10.10.10.101</td>
<td>6</td>
</tr>
<tr>
<td>server02</td>
<td>Lightbits Storage</td>
<td>192.168.16.32</td>
<td>ens1</td>
<td>10.10.10.102</td>
<td>6</td>
</tr>
<tr>
<td>client00</td>
<td>client</td>
<td>192.168.16.45</td>
<td>ens1</td>
<td>10.10.10.103</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note that:
* The servers are running AlmaLinux 8.7 with kernel 4.18.0-425.19.2.el8; however, other OSs are supported with different build releases. For additional information, see General System Requirements.
* The Lightbits RHEL release will be installed. Our Red Hat build supports Red Hat and AlmaLinux. For additional information, see General System Requirements.
* We will install in Single Instance/NUMA/node mode; however, if the servers have dual or more NUMAs, we could also do a dual instance/NUMA/node installation. Examples for this will be provided.

This section includes:

Before You Begin
Connecting to the Lightbits Software Repository
Verifying Network Connectivity for the Servers in the Cluster
Configuring the Ansible Environment
Running the Ansible Installation Playbook to Install Lightbits Cluster Software

Before You Begin

Lightbits recommends that you plan to use two networking interfaces for the Lightbits cluster installation: one for control and another as a data storage node.

Before you begin, it is recommended to review the General System Requirements and Required Ports sections of this guide.

Note that:
- The data interfaces must be on the same subnet (in pre-configured interfaces or as an input for Ansible).
- To install the cluster software, you need an Ansible module, and Ansible application-deployment tool v4.2.0 or later.
- The Python netaddr module, which is used to represent and manipulate network addresses.
- There is support for multiple Ansible tags (for cleanup for example), by using comma-separated tags.
- Based on the placement of SSDs in the server, check if you need to allow cross-NUMA devices in the profile.

Also, review the data networking and NVMe drive placement of the servers. This will be important during the installation configuration phase.

The online installation requires an internet connection and the need to configure several files on your system. The file repository URL is accessible and the RPMs are updated. The data interfaces of each server must be pre-configured on the same subnet. Our subnet is 10.0.10.0/24 (if the data interfaces are not pre-configured, they can be configured later using Ansible).

Additionally, check the NUMA placement of the NVMe drives. The command below shows which NUMA each NVMe drive belongs to (you can update your table with this information). Note that the main example of this installation section assumes that each server has six NVMe drives in NUMA 0.

```
lspci -mm | grep -Ei "nvme|SSD|Non-Volatile memory controller" \
  | awk '{print $1}' | xargs -I{} bash -c \n  'D=/sys/bus/pci/devices/0000:{};/ echo -n "$D: "; \
  echo $(cat $D/numa_node 2> /dev/null), $(cat $D/label 2> /dev/null)' | nl
```
The example output below is unrelated to the cluster we are demonstrating for installation. However, its output is useful in demonstrating how to interpret the command output. This shows 8 drives in NUMA0 and 8 drives in NUMA1. The column after numa_node shows the NUMA ID.

```plaintext
1 /sys/bus/pci/devices/0000:62:00.0/numa_node 0
2 /sys/bus/pci/devices/0000:63:00.0/numa_node 0
3 /sys/bus/pci/devices/0000:64:00.0/numa_node 0
4 /sys/bus/pci/devices/0000:65:00.0/numa_node 0
5 /sys/bus/pci/devices/0000:66:00.0/numa_node 0
6 /sys/bus/pci/devices/0000:67:00.0/numa_node 0
7 /sys/bus/pci/devices/0000:68:00.0/numa_node 0
8 /sys/bus/pci/devices/0000:69:00.0/numa_node 0
9 /sys/bus/pci/devices/0000:b3:00.0/numa_node 1
10 /sys/bus/pci/devices/0000:b4:00.0/numa_node 1
11 /sys/bus/pci/devices/0000:b5:00.0/numa_node 1
12 /sys/bus/pci/devices/0000:b6:00.0/numa_node 1
13 /sys/bus/pci/devices/0000:b7:00.0/numa_node 1
14 /sys/bus/pci/devices/0000:b8:00.0/numa_node 1
15 /sys/bus/pci/devices/0000:b9:00.0/numa_node 1
16 /sys/bus/pci/devices/0000:ba:00.0/numa_node 1
```

The online installation requires an internet connection on the Lightbits servers.

**Note:** An offline installation method is available that does not require an internet connection to access the file repository URL. For more information, see Performing an Offline Installation.

---

### Connecting to the Lightbits Software Repository

#### Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

**Notes:** - To proceed, see the Linux Repo File Customer TOKEN section in your Lightbits Installation Customer Addendum, for the token that is required to access the yum repository. Access to this repository is required to install the Lightbits cluster software.
- Contact Lightbits Support if you do not have this addendum document.
- If you are using the offline installation method, you can skip this step and proceed to Verifying Network Connectivity for the Servers in the Cluster.

Verify that you have the TOKEN & baseurl for the Lightbits RPM Repository. Log in to any of the future Lightbits servers and test the connection to the repository.
Verifying Network Connectivity for the Servers in the Cluster

Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td><strong>Verifying the network connectivity of the servers used in the cluster</strong></td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

Lightbits recommends that you verify the network connectivity for the servers you plan to use in the Lightbits cluster before you run the Ansible playbook. To simply confirm the connectivity status, use a ping command for each of the management NIC IPs and data NIC IPs in the servers.
Referring back to the Installation Planning Table, the example uses three Lightbits servers. Each server has a management IP.

Before proceeding with the installation, enter the following ping command from the Ansible installation host, to confirm that each Lightbits server is accessible via the Management Network IP.

```bash
$ ping -c 4 192.168.16.22
PING 192.168.16.22 (192.168.16.22) 56(84) bytes of data.
64 bytes from 192.168.16.22: icmp_seq=1 ttl=64 time=0.208 ms
--- 192.168.16.22 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.208/0.208/0.208/0.000 ms
```

Continuing with the example, a ping command is sent to each of the management network IPs and data network IPs.

<table>
<thead>
<tr>
<th>Server</th>
<th>Management Network IP</th>
<th>Data Network IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightbits server00</td>
<td>ping -c 4 192.168.16.22</td>
<td>ping -c 4 10.10.10.100</td>
</tr>
<tr>
<td>Lightbits server01</td>
<td>ping -c 4 192.168.16.92</td>
<td>ping -c 4 10.10.10.101</td>
</tr>
<tr>
<td>Lightbits server02</td>
<td>ping -c 4 192.168.16.32</td>
<td>ping -c 4 10.10.10.102</td>
</tr>
</tbody>
</table>

Before continuing, confirm the following connections:

- The Ansible installation host has ping connectivity to each Lightbits storage server’s management network IP. It does not need to have data network connectivity.
- Each Lightbits storage server has connectivity between all of the management network IPs.
- Each Lightbits storage server has connectivity between all of the data network IPs.
- Additionally, review the section on Required Ports, and make sure all of those ports are open and accessible on the Lightbits storage servers.

### Configuring the Ansible Environment

#### Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td><strong>Setting up an Ansible environment on your installation workstation</strong></td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

**This section includes:**

- Prepare Installation Workstation (Ansible Controller)
- Copying the Ansible Environment Tarball
- Creating the Inventory Structure and Adding the Ansible Hosts File
- Multi-Tenancy
- Configuring Global Variables in Ansible
- Verifying Hosts Connection
- Defining Configuration Files for Each “Ansible Host” (Server) in the Cluster
Prepare Installation Workstation (Ansible Controller)

The Ansible Installation host or Ansible Controller is the host running the Ansible playbook to install the Lightbits cluster.

We support two ways to set up the Ansible Controller:

- Ansible and dependencies installed on the Ansible Controller.
- Using a prebuilt Ansible Docker image.

Choose one of the methods and follow the steps - for either the Ansible or Docker method.

Ansible Method

For the Ansible method, follow these instructions to install the dependencies.

Install Ansible And Dependencies

The following tools are required to complete the Lightbits cluster software installation:

- sshpass or ssh-key authentication
- Python v3.6 or higher
- Python Modules: ansible, netaddr, python_jwt, six

If you have validated the networking environment as described in the previous section, log in to your installation workstation and begin downloading and installing the required tools on your workstation.

Install sshpass or Use ssh-key Authentication

The Python tool is essential for running commands remotely on each of the servers used in the cluster. To run these commands, you must install a Secure Shell (SSH) authentication software package. There are two ways to install this package.

- Use the Linux sshpass utility.
  
  To install “sshpass”, enter the following command at the CLI:

  ```
  $ yum install -y sshpass
  ```

- Use ssh-key authentication.
  
  To use ssh-key authentication, see Using SSH-Key Authentication.

- Package required for Multi-Tenancy

  ```
  $ yum install -y libselinux-python3
  ```

Install the Required Python Version

The Ansible installer is a module installed with Python. Lightbits recommends that you have Python v3.x or above installed on your system.

If Linux reports that Python 3.x is not installed, use the following command:
$ yum install python36-
Complete!

Install Ansible Module Using PyPI
Check if Ansible is installed, as well as its version; for example:

```
ansible --version
Command 'ansible' not found
```

If not found, you can also install Ansible using pip for Python3:

```
$ pip3 install ansible
```

Verify that Ansible is installed, as well as its version, by entering:

```
$ ansible --version
ansible [core 2.11.2]
  config file = None
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/local/lib/python3.9/site-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections:
  executable location = /usr/local/bin/ansible
  python version = 3.9.2 (default, Feb 19 2021, 17:33:48) [GCC 10.2.1 20201203]
  jinja version = 3.0.1
  libyaml = False
```

**Note:** If the installation fails due to a UnicodeEncodeError, it is because the locale is not fully configured on the Ansible host. Set the local LC_ALL environment variable and run the “pip3 install ansible” command again. For example for UTF8 systems, set the local to: `export LC_ALL=en_US.UTF-8`

Install Additional Python Modules
The Python netaddr module and python_jwt are also required.
These modules are used to manipulate network addresses, and generate JWT tokens as part of the installation.

At your workstation CLI enter:

```
$ pip3 install netaddr python_jwt six
```

Docker Method
Rather than installing and preparing Ansible and its dependencies, we also provide a custom Ansible image to deploy the Lightbits cluster image that contains all dependencies.
Copying the Ansible Environment Tarball

Lightbits Support provided you with an installation tarball along with your Installation Addendum that contains all of the configuration files that the Ansible playbook requires.

Copy the tarball file to your installation workstation home directory and unpack the tar file using the instructions below.

Create a light-app directory and extract the contents into the new directory.

```
$ mkdir ~/light-app
$ tar -xvzf light-app-install-environment-v<Version>.tgz -C ~/light-app
```

Unpacking this tarball creates the following Ansible directory structure inside of the light-app directory, which contains the Ansible environment where the “ansible-playbook” command runs.
Inventory Structure and Adding the Ansible Hosts File

The Ansible playbook installer requires configuration files to drive it.

In Ansible terminology, each Lightbits storage server is referred to as a “host”. Details about the Lightbits storage servers must be entered into the Ansible “hosts” file that is stored in an “inventory” temporary directory structure.

Complete the following steps to configure the Ansible “hosts” file, which describes all of the Lightbits storage server names and their management IPs.

**Notes:**
- The servers’ hostnames and Ansible names do not have to match. We usually refer to the servers’ Ansible names as server00, server01, server02, etc. These will become the servers’ identifying names with the Lightbits software. Therefore this will give the servers a name of serverXX going forward.
- For the ansible_host field, provide the management IP. However if the servers are only configured with data IPs and no management IP, then provide the data IPs of the server (in this case the data IP doubles as the management and data IP).

1. Open a text editor and edit the copied `hosts` example file, which is now found in the new `~/light-app/ansible/inventories/cluster_example/hosts` path. Replace the `ansible_host`, `ansible_ssh_pass`, and `ansible_become_user` values with your environment’s relevant values. This is for each server that will be in your cluster. Refer to the following example for reference.
You can replace the `ansible_host` flag's value with the interface DNS name or IP address. In this example, the management network IP addresses from the cluster details table are used, not the data network IPs.

- Also in this example hosts file, there is a "local_repo_base_url" entry that includes . This information was provided to you in the Customer Addendum. You will need to enter this value here before proceeding.

2. Remove the client00 line in the top section and “[initiators]” sections.

**Note:** It is possible to set up the Ansible files to install and configure clients. However, this section only describes how to install the Lightbits storage servers. The next section details how to configure and connect clients.
3. Take into account the following information when filling out the “hosts” file.

More information about the hosts file:

- The top section of the “hosts” describes how Ansible will connect to other servers to install Lightbits. It also provides a friendly name for each Lightbits server; for example, “server00”. These names will be used going forward by the Lightbits software as the identifying name for the servers.
- The “duros_nodes” section describes where Lightbits will be installed.
- The “duros_nodes:vars” section describes from where Lightbits will be installed. In this case/example the repo URL is provided, as this uses the online method. However, for the offline installation method this section is different. For more, see Single-IP-Dual-NUMA Configuration.
- The “local_repo_base_url” field must be filled in with the and remainder of the URI to properly direct the Ansible installation to the correct Lightbits repository. Different release versions will have slightly different paths. The TOKEN is provided in the Customer Installation Addendum. The “local_repo_base_url” must be correct or else the installation will fail. The “local_repo_base_url” value should be the same as the “baseurl” value in Connecting to the Lightbits Software Repository. If that worked successfully, then it is ok.
- The “auto_reboot” field instructs the servers to reboot during the installation. This is an important part of the installation.
- The “cluster_identifier” can be left as is, as it is unused. Note that the cluster will end up getting an autogenerated ID called the “clusterName” key, which can be changed after installation.
- Lightbits uses “etcd” for the key/value database of the cluster. The “etcd” section describes where etcd will be installed and which servers will become members of etcd. Every server appearing in “duros_nodes” must appear in the “etcd” section.

Host File Server Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>local_repo_base_url</td>
<td>Yes</td>
<td>Mandatory unless offline installation is used. This is the same value entered for the “baseurl” you configured in the Connecting to the Lightbits Software Repository section.</td>
</tr>
</tbody>
</table>
### Multi-Tenancy

Lightbits v2.2.1 and above enforces tenant isolation on the control plane (“multi-tenancy”). With multi-tenancy, multiple tenants can share a Lightbits cluster without being able to see or affect each other’s resources when accessing the Lightbits API or using the Lightbits command line tools.

Command line tools and all other API users must use the v2 Lightbits API. The v2 API includes provisions for authentication and authorization via standard JSON Web Tokens (“JWTs”), as well as transport security for all API operations.

The following three predefined roles are created by default:

- cluster-admin (system scope)
- admin (project scope)
- viewer (project scope)

Currently, roles cannot be added.

At installation, the user can provide their own certificate and CA to be used by the peers. If these files are not provided, the installation will generate self-signed certificates.

### Certificates Directory

By default, certificates are stored at `certificates_directory=~/lightos-certificates` on the Ansible controller machine.

`certificates_directory` can be overridden via cmd-line:

```
ansible-playbook playbooks/deploy-lightos.yml \
-e 'certificates_directory=/path/to/certs' ...
```

Or via `group_vars/all`:

```
yaml
certificates_directory=/path/to/certs
```

### Certificate Types

Implementing multi-tenancy involves three sets of certificates:

---

**Table:**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>auto_reboot</td>
<td>No</td>
<td>A False value means that the installation will wait for user instructions to either reboot or not after installation. A True value means that the installation will reboot in case of a kernel change without user instructions. The default value is False.</td>
</tr>
<tr>
<td>cluster_identifier</td>
<td>No</td>
<td>An identifier of the cluster that is used to filter the logs of a specific cluster.</td>
</tr>
</tbody>
</table>

---

**Note:** To use `ansible_ssh_private_key_file` instead of `ansible_ssh_pass`, see Using SSH-Key Authentication.

---

1. The final “hosts” file will look similar to the above output. Save and exit out of the `~/light-app/ansible/inventories/cluster_example/hosts` file.
• Multi-Tenancy
  – Certificates Directory
    * etcd Certificates for mTLS Peer Communication
    * API Service Certificates For TLS
    * System Scope Cluster Admin Certificates
  – Generating Self-Signed Certificates
  – Bring Your Own Certificates

**etcd Certificates for mTLS Peer Communication**
All etcd services serve client APIs only on localhost. This minimizes the exposure of etcd to outside malicious activity. Peer communication must be encrypted at all times, since etcd passes sensitive traffic between its peers.

The installation script expects the following files to be present at *certificates_directory* on the Ansible controller machine:

```
etcd-ca-key.pem
etcd-ca.pem
{ansible_hostname}-cert-etcd-peer-key.pem
{ansible_hostname}-cert-etcd-peer.pem
```

- **etcd-ca**: Certificate authority (CA) parameters for etcd certificates. This CA is used to sign certificates used by etcd (such as peer and server certificates).
- **{ansible_hostname}-cert-etcd-peer**: The peer certificate is used by etcd for peer communication.

These files are passed to the following etcd parameters: `--peer-cert-file` and `--peer-key-file`.

**Note:** `{ansible_hostname}` is the name we gave the etcd node in the hosts file.

**Example**
A 3-node cluster with server00-02 will result in:

```
etcd-ca-key.pem
etcd-ca.pem
server00-cert-etcd-peer-key.pem
server00-cert-etcd-peer.pem
server01-cert-etcd-peer-key.pem
server01-cert-etcd-peer.pem
server02-cert-etcd-peer-key.pem
server02-cert-etcd-peer.pem
```

**Notes:** - These names are hard-coded in the installation script. Only the source directory can change.
- If these files are not provided, the installation will generate self-signed certificates and place them at *certificates_directory* on the Ansible controller machine.

**API Service Certificates For TLS**
All API endpoints are TLS-enabled by default.

The user can provide their own SSL certificates, or the installation process will generate a self-signed certificate.

These are the files used by *api-service* to set up TLS communication.
System Scope Cluster Admin Certificates

These files will be stored in etcd and used to authenticate a system-scope project.

These are the files used to generate system scope credentials:

- cert-lb-admin-key.pem
- cert-lb-admin.pem

Generating Self-Signed Certificates

The Lightbits installation playbook checks for `certificates_directory` existence. If it does not exist, a folder will be created and populated with self-signed certificates. If it exists, the playbook will verify that all expected certificates files are present. In case one is missing the installation will fail.

**Notes:** Certificate file names are hard-coded in the installation script. Only the source directory can change. These are pairs and go together.

**File name format:**
- `<name>.pem`: Certificate.
- `<name>-key.pem`: RSA private key that matches the certificate.

- In case we want to regenerate the self-signed certificates, we should delete the `certificates_directory` and all of its content.

Bring Your Own Certificates

You can provide your own certificates for each of the components.

You can override part or all of the files before running the `install-lightos.yaml` playbook.

Configuring Global Variables in Ansible

Review the configuration of the global variables in the Ansible file `all.yml`, located in the following location:

```
~/light-app/ansible/inventories/cluster_example/group_vars/all.yml
```

If all of the machines in the cluster have PMEM (NVDIMM or Intel Optane) installed, the `persistent_memory` flag must be set as follows:

```
persistent_memory: true
```

If there are machines in the cluster that do not have PMEM installed, then set this flag to `false`.

**Note:** Since the `persistent_memory` flag is a global property for all of the clusters, it is important to declare this flag only once under the `all.yml` file and not in `host_vars` files with different values.
NTP vs Chrony

Check if your OS prefers NTP or Chrony, and proceed using that option.

NTP:

For NTP configurations, the default settings in the `all.yml` file can be used (note that the defaults are greyed out). You can uncomment and edit the preferred NTP servers, NTP version, and its dependencies packages. For more information on these parameters, see Network Time Protocol Configuration.

```yaml
#nvme_subsystem_nqn_suffix: "some_suffix"
#nntp_enabled: true
#chrony_enabled: false
#nntp_manage_config: true

#nntp_servers:
#  - "0{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "1{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "2{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "3{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"

#nntp_version: "nntp-4.2.6p5-29.el7.centos.x86_64"

#nntp_packages:
#  - "autogen-libopts*.rpm"
#  - "ntpd*.rpm"
#  - "nntp*.rpm"
```

**Note:** When configuring to use NTP, make sure `chrony_enabled` is either commented out like above, or set to `false`.

Chrony:

For Chrony configurations - which are the default for Red Hat 8 based releases and onward - configure the `all.yml` settings as follows:

Disable NTP and enable Chrony. Additionally, make sure that the Chrony service is configured on your servers.

```yaml
ntp_enabled: false
chrony_enabled: true
#nntp_manage_config: false
use_lightos_kernel: false

#nntp_servers:
#  - "0{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "1{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "2{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"
#  - "3{'.' + nntp_area if nntp_area else ''}..pool.ntp.org iburst"

#nntp_version: "nntp-4.2.6p5-29.el7.centos.x86_64"

#nntp_packages:
#  - "autogen-libopts*.rpm"
#  - "ntpd*.rpm"
#  - "nntp*.rpm"
```

**Note:** Verify that the date and time are in sync on the Lightbits storage servers and the Ansible installation host. You can use `date` and run it simultaneously on all servers.
Use Lightos Kernel

If a Red Hat based release is used, add the following to \texttt{all.yml}:

\begin{verbatim}
use_lightos_kernel: false
\end{verbatim}

Setting \texttt{use\_lightos\_kernel} to false ensures that the kernel that is on the servers remains as is.

If a GA based release is used, remove that line or set it true.

\begin{verbatim}
use_lightos_kernel: true
\end{verbatim}

Setting \texttt{use\_lightos\_kernel} to true will install the Lightbits supplied kernel, which is a requirement for the Lightbits GA releases.

Single IP Dual NUMA Configuration

The Single IP dual NUMA configuration (see Single-IP-Dual-NUMA Configuration), requires changes to the \texttt{all.yml} file. Follow the directions in that section to properly configure this setup.

Verifying Hosts Connection

Verify connectivity from the Ansible Installation Host to all of the machines where Lightbits will be installed. First verify that ping works. Then verify that SSH connectivity works using the normal user and then using a privileged (become) user.

Testing Connectivity via Ping

Use the ping command to verify that all machines in the cluster respond. Enter the following Ansible shell command:

\begin{verbatim}
$ cd ~/light-app
$ ansible -i ansible/inventories/cluster_example/hosts all -m ping
\end{verbatim}

A successful response from this Ansible ping is as follows:
Note: If you see an error related to ssh-key authentication, see Troubleshooting.

Testing Connectivity via SSH

This command tests a connection using the user configured with the `ansible_ssh_user` field in the “hosts” file.

```
$ cd ~/light-app
$ ansible all -i ansible/inventories/cluster_example/hosts -m command -a id
```

The expected output for each machine is its “id” output, which should return the username and groups.

The following is an example of a good output:

```
server00 | CHANGED | rc=0 >>
uid=0(root) gid=0(root) groups=0(root)
server01 | CHANGED | rc=0 >>
uid=0(root) gid=0(root) groups=0(root)
server02 | CHANGED | rc=0 >>
uid=0(root) gid=0(root) groups=0(root)
```

Any output other than an error is a good output. If there are connection issues, verify that SSH is properly set up and make sure that the `ansible_ssh_user` and `ansible_ssh_pass` are properly configured.

After the above command is successful, test that the privileged user can access the machines over SSH. Note that if the `ansible_ssh_user` is root, you can skip this final verification.

```
$ cd ~/light-app
$ ansible all -i ansible/inventories/cluster_example/hosts -m command -a id -b
```
Defining Configuration Files for Each “Ansible Host” (Server) in the Cluster

Return to the `~/light-app/ansible/inventories/cluster_example` directory you created in Inventory Structure and Adding the Ansible Hosts File.

```bash
~/light-app/ansible/inventories/cluster_example
|-- cluster_example
    |-- group_vars
    |    |-- all.yml
    |-- hosts
    |-- host_vars
          |-- client00.yml <- This file can be ignored or deleted.
          |-- server00.yml
          |-- server01.yml
          |-- server02.yml
```

From this path we will edit each of the yml files found in the `~/light-app/ansible/inventories/cluster_example/host_vars` subdirectory. In our example cluster, we have three Lightbits storage nodes that are defined by the files:

- `host_vars/server00.yml`
- `host_vars/server01.yml`
- `host_vars/server02.yml`

1. In each of the host variable files, update the following required variables:

   **Required Variables for the Host Variable File**

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The cluster server’s name. Example: serverXX. Must match the filename (without the extension) and server names configured in the “hosts” file.</td>
</tr>
<tr>
<td>instanceID</td>
<td>The configuration parameters for the logical node in this server. Currently, Lightbits supports up to two logical nodes per server.</td>
</tr>
<tr>
<td>ec_enabled</td>
<td>(per logical node) Enables Erasure Coding (EC) protects against SSD failure within the storage server by preventing IO interruption. Normal operation continues during reconstruction when a drive is removed.</td>
</tr>
<tr>
<td>failure domains</td>
<td>(per logical node) The servers sharing a network, power supply, or physical location that are negatively affected together when there are network, power, cooling, or other critical service experience problems. Different copies of the data are stored in different FDs to keep data protected from various failures. To specify the servers in the FD, you must add the server names. For further information, see Defining Failure Domains.</td>
</tr>
<tr>
<td>data_ip</td>
<td>(per logical node) The data IP used to connect to other servers. Can be IPv4 or IPv6.</td>
</tr>
</tbody>
</table>
Defining Configuration Files for Each “Ansible Host” (Server) in Lightbits Cluster Software Installation

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>storageDeviceLayout</td>
<td>(per logical node) Sets the SSD configuration for a node. This includes the number of initial SSD devices, the maximum number of SSDs allowed, allowance for NUMA across devices, and memory partitioning and total capacity. For further information, see Setting the SSD Configuration.</td>
</tr>
<tr>
<td>initialDeviceCount</td>
<td>The number of NVMe drives accounted for this instance to use.</td>
</tr>
<tr>
<td>maxDeviceCount</td>
<td>The maximum number of NVMe drives current instances can support. Commonly configured equal to initialDeviceCount or higher.</td>
</tr>
<tr>
<td>allowCrossNumaDevices</td>
<td>Leave this setting set as “false” if all of the accounted NVMe drives for this instance are in the same NUMA. Set it to “true” if to access the NVME drives this instanceID will need to do cross-NUMA communication.</td>
</tr>
<tr>
<td>deviceMatcers</td>
<td>This determines which NVMe drives will be considered for data and which will be ignored. For example, if the OS drive is an NVME drive, it can be ignored using the name option. The default settings do a good job by only counting NVMe drives greater than 300 GiB and without partitions to be part of the data.</td>
</tr>
</tbody>
</table>

To update these parameters, the cluster details table is useful.

**Installation Planning Table Sample**

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Role</th>
<th>Management Network IP</th>
<th>Data NIC Interface Name</th>
<th>Data NIC IP</th>
<th>NVMe Drives</th>
</tr>
</thead>
<tbody>
<tr>
<td>server00</td>
<td>Lightbits Storage Server 1</td>
<td>192.168.16.22</td>
<td>ens1</td>
<td>10.10.10.100</td>
<td>6</td>
</tr>
<tr>
<td>server01</td>
<td>Lightbits Storage Server 2</td>
<td>192.168.16.92</td>
<td>ens1</td>
<td>10.10.10.101</td>
<td>6</td>
</tr>
<tr>
<td>server02</td>
<td>Lightbits Storage Server 3</td>
<td>192.168.16.32</td>
<td>ens1</td>
<td>10.10.10.102</td>
<td>6</td>
</tr>
<tr>
<td>client00</td>
<td>client</td>
<td>192.168.16.45</td>
<td>ens1</td>
<td>10.10.10.103</td>
<td>N/A</td>
</tr>
</tbody>
</table>

Note: The following is an example for three Lightbits servers in a cluster with a single client.

Examples for the three host variable files follow.

**server00.yml**
name: server00
nodes:
  - instanceID: 0
data_ip: 10.10.10.100
failure_domains:
  - server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition == false
  - size >= gib(300)
  # - name =~ "nvme0n1"

server01.yml

name: server01
nodes:
  - instanceID: 0
data_ip: 10.10.10.101
failure_domains:
  - server01
instanceID: 0
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition == false
  - size >= gib(300)
  # - name =~ "nvme0n1"

server02.yml
Defining Failure Domains

A Failure Domain (FD) encompasses a section of a network, power supply, or physical location negatively affected when network, power, cooling, or other critical service experiences problems occur. Different copies of the data are stored in different FDs to keep data protected from various failures.

To specify the servers in the FD, you can configure it with items in the failure_domains array of the server configuration files. Take into consideration the server00.yml and server01.yml configuration below:

Server00 failure_domains array is configured with its own server name and the rack it is placed in, “rack00”.

Notes: - See Host Configuration File Variables for the entire list of variables available for the host variable files.
- You can also reference additional host configuration file examples.
- Typically the servers should already be configured with the data_ip. However, the Ansible playbook can configure the data NIC IP; for that you will need to add a section data_ifaces with the data interface name. For further information, see Configuring the Data Network. Also section 4.4.9 shows an example of this configuration.
- If you need to create a separate partition for etcd data on the boot device, see etcd Partitioning.
- Based on the placement of SSDs in the server, check if you need to make a change in the client profile to permit cross-NUMA devices.
- Starting from Version 3.1.1, data IP can be IPv6. For example: data_ip: 2600:80b:210:440:ac0:ebff:fe8b:ebc0
Server01 failureDomains array is configured with its own server name and the rack it is placed in “rack00”.

Make a note of the items in both server00 and server01 failureDomains arrays.

Since both servers share the same “rack00”, volumes replicas will not be shared between these two servers (and their nodes).

If the lists were default, then volume replicas would be shared between the servers. Default means server00 failure_domain
array only had “server00”, and server01 failure_domain array only had “server01”.

Notes: - At a minimum or good default configuration, configure the failure_domains with the server names. Add other items as above with “rack00”, to help control the flow of volume replication.
- In a dual instance/node setup, volume replicas will not land on other nodes of same server.
- See Host Configuration File Variables for the entire list of variables available for the host variable files.

Notes: - The configurations above have a “data_ifaces” section for each server configuration. Typically this section is not included as the servers should be preconfigured with their data IPs; however, we can instruct Ansible to configure the data IPs during the Lightbits installation, so that the “data_ifaces” section tells Ansible to configure the IP and subnet on said interface.
- Note that for ipv6 addresses, you will use ‘ip6: ip/prefix’ format. For example: ip6: 2001:0db8:0:f101::1/64.
- The addresses used for ip4 or ip6 fields must match the address used in data_ip. The only difference is that ip4 and ip6 show the subnet or prefix as well. However, note that data_ip only shows the address without the subnet or prefix.

Setting the SSD Configuration

To allow for future storage expansion, you will need to set the Maximum Device Count to the total number of drive slots physically available in the Lightbits node during the initial Lightbits configuration process.

Setting the Maximum Device Count to the maximum number of drive slots allows you to start the Lightbits node with empty drive slots in the server chassis. This is because you only need a small amount of storage and plan to add more SSDs into the empty drive slots as demand increases.

For example, your storage server chassis has 12 SSD slots, but initially, you only want to configure Lightbits to use eight drives. So in this case, you need to:

- Set your Maximum Device Count to 12.
- Physically install only eight drives.
- Leave four drive slots empty for later use.

Note: If Erasure Coding is enabled (ec_enabled: true), you must have a minimum of six SSDs installed in that node.

To specify the SSD configuration for a node, you must enter a value for the total drive slots available for your Lightbits node to the host configuration file as follows:
Confirming the Required Directory Structure

Before proceeding, change to the `light-app` directory. It is important to be in this directory when running the ansible-playbook in the next section. Ansible depends on some files and directories to be in certain places.

```bash
$ cd ~/light-app
```

Run `ls` and make sure you see the following files and folders.

- `ansible`
- `ansible.cfg`
- `playbooks`
- `plugins`
- `roles`

Additionally confirm the structure of your `ansible` directory to be similar to this:

```yaml
name: server00
data_ifaces:
  - bootproto: static
    conn_name: ens1
    ifname: ens1
    ip4: 10.10.10.100/24

nodes:
  - instanceID: 0
    data_ip: 10.10.10.100
    failure_domains:
      - server00
    ec_enabled: true
    data_ip: 10.10.10.100
    lightfieldMode: SW_LF
    storageDeviceLayout:
      initialDeviceCount: 6
      maxDeviceCount: 12
      allowCrossNumaDevices: false
    deviceMatchers:
      # - model =~ ".*"
      - partition == false
      - size >= gib(300)
      # - name =~ "nvme0n1"
```
Running the Ansible Installation Playbook to Install Lightbits Cluster Software

Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td><strong>Installing a Lightbits cluster by running the Ansible installation playbook</strong></td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

As discussed in Prepare Installation Workstation (Ansible Controller), we support installing using Ansible or using a prebuilt Docker image that contains Ansible. Pick the method that applies to your installation environment and follow the commands to install the Lightbits cluster software on the storage servers. Afterwards, go to the bottom of the section to confirm a successful installation.

**Note:** For both methods, we provide the simple default installation methods. However, we provide other more advanced installation configuration examples in the Ansible Docker section. Note, however, that these same examples can be adopted into the Ansible method; for that you will just skip the Docker commands and just refer to the ansible-playbook commands as the template.

Ansible Installation Method

Running the Ansible Controller

**Note:** The Ansible playbook operations below can take several minutes. The output will report the status of all the tasks that succeeded/failed on the nodes.

To install the cluster software and configure the cluster, change into light-app directory with `cd ~/light-app`, and enter the following command to run the playbook:

```
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/deploy-lightos.yml -vvv
```
Running the Ansible Installation Playbook to Install Lightbits

**Notes:**
- This command must be run from the directory where light-app was extracted to. Then all of the paths will work as displayed.
- The inventory file points to a “hosts” file, which instructs Ansible where to deploy Lightbits.
- The selected playbook, “deploy-lightos.yml”, instructs Ansible on how to install and configure the Lightbits cluster on the servers mentioned in the “hosts” file.
- Ansible will log to its default path as specified by ansible.cfg. By default that is /var/log/ansible.log. The log path can be changed by prefixing ANSIBLE_LOG_PATH=/var/log/ansible.log ansible-playbook ...
- The following files will be created into the home directory: lightos-system-jwt & lightos-default-admin-jwt.
- Certificates used by the cluster will be saved into a new directory, lightos-certificates. This directory will be created in the home directory.
- It is recommended to make a secure backup of this content, or at a minimum, the jwt files and lightos-certificates directory.
- Debug level verbosity is enabled with -vvv. It helps diagnose any issues if they happen.

When the installation is done, the cluster will be bootstrapped with a system-scope project. You will need access to the JWT. By default the cluster-admin JWT is placed in ~/lightos-system-jwt of the Ansible host. This path can be changed by editing group_vars/all.yml before running the ansible-playbook, and appending this variable system_jwt_path: "{{ '-/lightos-system-jwt' | expanduser }}"

**Prebuilt Ansible Docker Installation Method**

If using the prebuilt Docker image with Ansible, ensure that Docker is logged in. Refer to the bottom of Prepare Installation Workstation (Ansible Controller) for instructions on how to log in using Docker.

This first subsection shows instructions on launching a simple or default Lightbits installation. The subsections afterwards show other variations to the installation that might apply to more complex configurations.

Additionally, each Docker example requires the correct Docker URL docker.lightbitslabs.com/lightos-3-(Minor Ver)-(Rev)-rhl-8/lb-ansible:4.2.0. Note that the path bit is incomplete and requires substitution. Refer to the Lightbits Installation Customer Addendum for the correct Docker image URL.

**Running Using the lb-ansible Docker Image**

```
mkdir -p /opt/lightos-certificates
cd -/light-app
docker run -it --rm --net=host \
  -v /opt/lightos-certificates:/lightos-certificates \
  -v `pwd`:/ansible \
  -w /ansible \
  -e ANSIBLE_LOG_PATH=/ansible/ansible.log \
  docker.lightbitslabs.com/lightos-3-(Minor Ver)-(Rev)-rhl-8/lb-ansible:4.2.0 \
  sh -c 'ansible-playbook \
   -e system_jwt_path=/ansible/lightos_jwt \
   -e lightos_default_admin_jwt=/ansible/lightos_default_admin_jwt \
   -e certificates_directory=/lightos-certificates \
   -i ansible/inventories/cluster_example/hosts \
   playbooks/deploy-lightos.yml -vvv'
```

We will pre-create the /opt/lightos-certificates directory, so that our certificates get saved outside of the container.

Command breakdown:
- Mount host’s /opt/lightos-certificates to docker’s /lightos-certificates to store generated certificates on the host. Docker will create the /opt/lightos-certificates directory on the host if it is missing.
- Mount the current working directory or $PWD to /ansible inside the container, to have access to the playbook and roles. The current working directory at this point will be where light-app was extracted.
- Set the WORKDIR to /ansible inside the container. This sets the current working directory within docker to /ansible.
• Configure Ansible to write logs to /ansible/ansible.log.
• Run the playbook with specified hosts from an inventory folder in the ansible/inventories/cluster_example.
• Set system_jwt_path to be placed at $PWD/lightos_jwt after the container is closed.
• Set lightos_default_admin_jwt to be placed at $PWD/lightos_default_admin_jwt after the container is closed.

Custom Inventory Folder
If the inventory folder is placed in a different location, such as: /path/to/inventory, you can mount this path as well and use it:

```bash
mkdir -p /opt/lightos-certificates
cd ~/light-app
docker run -it --rm --net=host \
  -v /opt/lightos-certificates:/lightos-certificates \
  -v `pwd`:/ansible \
  -v /path/to/inventory:/inventory \
  -w /ansible \
  -e ANSIBLE_LOG_PATH=/ansible/ansible.log \
docker.lightbitslabs.com/lightos-3-(Minor Ver)-(Rev)-rhl-8/lb-ansible:4.2.0 \
sh -c 'ansible-playbook \
  -e system_jwt_path=/ansible/lightos_jwt \
  -e lightos_default_admin_jwt=/ansible/lightos_default_admin_jwt \
  -e certificates_directory=/lightos-certificates \
  -i /inventory/hosts \
  playbooks/deploy-lightos.yml -vvv'
```

Using SSH-Keys Present On Ansible-Controller Host
If you use SSH-Keys present on your ansible-controller machine and you copied these keys to the authorized_keys on target hosts you will want to use this key inside the container.

The following example shows how to mount the ~/.ssh folder so that Ansible running inside the container will use it.

```bash
mkdir -p /opt/lightos-certificates
cd light-app
docker run -it --rm --net=host \
  -v /opt/lightos-certificates:/lightos-certificates \
  -v `pwd`:/ansible \
  -v ${HOME}/.ssh:${HOME}/.ssh \
  -w /ansible \
  -e ANSIBLE_LOG_PATH=/ansible/ansible.log \
docker.lightbitslabs.com/lightos-3-(Minor Ver)-(Rev)-rhl-8/lb-ansible:4.2.0 \
sh -c 'ansible-playbook \
  -e system_jwt_path=/ansible/lightos_jwt \
  -e lightos_default_admin_jwt=/ansible/lightos_default_admin_jwt \
  -e certificates_directory=/lightos-certificates \
  -i ${HOME}/.ssh/authorized_keys \
  playbooks/deploy-lightos.yml -vvv'
```

Verify Successful Installation
When the installation completes with no errors, you will see an output similar to the following:
PLAY RECAP********************************************************************************

server00 : ok=68  changed=19  unreachable=0  failed=0  skipped=34  rescued=0  ignored=0
server01 : ok=67  changed=18  unreachable=0  failed=0  skipped=33  rescued=0  ignored=0
server02 : ok=67  changed=18  unreachable=0  failed=0  skipped=33  rescued=0  ignored=0

Notes: - The “failed=0” indicates that the installation finished without errors.
    - If the installation process failed, see Recovering from Cluster Installation Failure.

The installation flow is now complete, and you can move on to the client configuration sections of the Installation Guide.

Note: You should also make sure you back up your installation files properly. For more, see Lightbits Software Installation Planning.

Post Installation Steps

After a successful Lightbits cluster installation, perform the following steps.

Back Up Important Content

Back up the following contents to a secure location. This will be useful if another node is added in the future or other troubleshooting is required.

1. Back up the ~/light-app directory and all of its contents. The contents of the Ansible directory will be helpful in the future if there is a need to add servers or check over how a previous installation was done.

2. Back up the generated JWTs. Back up the system-jwt and the default project jwt. By default they are placed in the home directory: ~/lightos-system-jwt and ~/lightos-default-admin-jwt.

3. Back up the generated certificates. By default these are placed in ~/lightos-certificates.

Check Cluster Health

1. Copy the contents of the system jwt file ~/lightos-system-jwt into the clipboard.

From the Ansible host, run cat ~/lightos-system-jwt; echo; Note that we add the last echo to generate a new line at the end, so that it is easy to determine where the JWT ends.

The contents will be similar to this:

```bash
export LIGHTOS_JWT=eyJhbGciOiJSUzI1Ni<...CONTENTS OF JWT...>5PcYPBRBaFEuMsT9gQNQA
```

Note: The contents of a JWT are long and all are on a single line.

2. Log in to any Lightbits server and paste the contents into the shell. The JWT will now be available via the $LIGHTOS_JWT environment variable.

3. Check the state of the servers, nodes, and cluster.

The servers look healthy as they all state “NoRiskOfServiceLoss”:
Post Installation Steps

server00:~# lbcli -J $LIGHTOS_JWT list servers
NAME   UUID                  State         RiskOfServiceLoss State
LightOSVersion
server00  5c2b7375-64fa-583e-8ebe-e82b8e1e1a53 Enabled  NoRiskOfServiceLoss
3.1.2-b1125
server01  bb5433d2-9740-5130-a5eb-47f623c19b4d Enabled  NoRiskOfServiceLoss
3.1.2-b1125
server02  a9097bf0-6005-5dca-8f07-eeaca114ec70 Enabled  NoRiskOfServiceLoss
3.1.2-b1125

The nodes also look healthy, as they all state to be “Active”:

server00:~# lbcli -J $LIGHTOS_JWT list nodes
Name   UUID                  State         NVMe endpoint  Failure
domains Local rebuild progress
server00-0 d2c336f2-c9e5-5a4c-951e-ede739d10774 Active 10.10.10.100:4420 [server00] None
server01-0 810ed593-a97b-5495-b08a-be5e37a65f82 Active 10.10.10.101:4420 [server01] None
server02-0 2cf6ce67-fe5e-5a98-86bf-98a5792a8916 Active 10.10.10.102:4420 [server02] None

Also check that the cluster health state is ok:
At this point the cluster's health has been confirmed at the node, server, and cluster level.
Linux Cluster Client Software Installation

To connect to the Lightbits storage server, the cluster client software requires the appropriate NVMe over TCP kernel module and application support. The client should support NVMe/TCP with an ANA (Asymmetric Namespace Access) enabled kernel. NVME multipath has to be enabled on the clients as well.

Connecting to the Cluster Client DEB Repository

Prepare your Debian/Ubuntu based client with the Lightbits repository, using these steps.

1. Run the following commands:

   ```bash
   apt-get install -y debian-keyring
   apt-get install -y debian-archive-keyring
   apt-get install -y apt-transport-https
   apt-get install curl
   ```

2. Add the apt key:

   ```bash
   curl -1sLf 'https://dl.lightbitslabs.com/<TOKEN>/lightos-3-(Minor Ver)-(Rev)-rhl-8/cfg/gpg/gpg.<KEY>.key' | apt-key add -
   ```

   **Note:** All of the required parameters for the curl command above will be in your Lightbits Installation Customer Addendum. This includes the TOKEN, path, and GPG KEY fingerprint.

3. Create the lightos repo:

   ```bash
   curl -1sLf 'https://dl.lightbitslabs.com/<TOKEN>/lightos-3-(Minor Ver)-(Rev)-rhl-8/cfg/setup/config.deb.txt?distro=ubuntu&codename=xenial' > /etc/apt/sources.list.d/lightos.list
   ```

   **Notes:** - Token and path are provided via the Customer Addendum.
   - Replace “xenial” in the URL with the correct codename of your Ubuntu OS.
   - Run `lsb_release -a` to verify your codename.

4. Editing the repository file to point at the correct GPG key or force trust.

**Correct GPG Key:**

By default, the created `lightos.list` repo file points to an incorrect path for the GPG key, so running `apt-get update` at this point will fail.

First, confirm the correct GPG key path by running `apt-key list`.

Locate the Lightbits key. It should sit in the `/etc/apt/trusted.gpg` file.

Edit the repo file `/etc/apt/sources.list.d/lightos.list` and replace `[signed-by=/path/to/key]` to the correct path `[signed-by=/etc/apt/trusted.gpg]`.
$ cat /etc/apt/sources.list.d/lightos.list

deb [signed-by=/etc/apt/trusted.gpg] https://dl.lightbitslabs.com/<TOKEN>/
lightos-3-(Minor Ver)-(Rev)-rhl-8/deb/ubuntu xenial main

deb-src [signed-by=/etc/apt/trusted.gpg] https://dl.lightbitslabs.com/<TOKEN>/
lightos-3-(Minor Ver)-(Rev)-rhl-8/deb/ubuntu xenial main

**Force Trust:**

In case you want to bypass the GPG verification, edit the `/etc/apt/sources.list.d/lightos.list` file and replace `[signed-by=/path/to/key]` with `[signed-by=/etc/apt/trusted.gpg]` after the `deb` and `deb-src` parts:

$ cat /etc/apt/sources.list.d/lightos.list

deb [trusted=yes] https://dl.lightbitslabs.com/<TOKEN>/lightos-3-(Minor Ver)-(Rev)-rhl-8/deb/ubuntu xenial main


5. Run the apt-update command:

```shell
apt-get update
```

**Connecting to the Cluster Client RPM Repository**

Follow the steps in Connecting to the Lightbits Software Repository to prepare the Lightbits repository on your RPM-based client. Make sure to have the TOKEN ready, which is provided in your Lightbits Installation Customer Addendum. This is required to install the Lightbits NVME-Client-RPMs.

**Note:** You can copy the repo file content from an installed Lightbits server’s file, `/etc/yum.repos.d/lightos.repo`. It will have the correct token and baseurl.

**Note:** An optional Ansible playbook is available to you that performs the following:
- Installs kernel v5.x, which includes the nvme-tcp upstream driver.
- Creates a small 4GB volume with a replication factor of 2.
- Runs the `nvme connect` command to connect the client machine to the cluster volume.
- Runs an `fio` read/write workload for 30 seconds.
- Performs a cleanup that disconnects the nvme client and removes the volume.

For more information about using this optional playbook, see Automated Client Connectivity Verification.

**Installing the Latest Kernel on the Client**

**Note:** You can use any kernel version v5.3.5 or above, which is written in the following instructions.

It is recommended to run the latest kernel on the clients connecting to the Lightbits servers. This section describes how to update the kernel on a Red Hat based OS (AlmaLinux), and CentOS.

**Note:** The instructions below are only for Centos 7.9 and Red Hat 8 based OSes (AlmaLinux included). Updating the kernel will vary for different OSs. Please verify with the official OS documentation for how to upgrade the kernel.
Installing the Latest Kernel on AlmaLinux/Red Hat 8

To install the latest kernel on AlmaLinux/Red Hat 8, update all of the system packages with the following command:

```
yum update
```

This will update all of the packages, including the kernel. Before updating the package, yum will list all of the changes that it will make. Please review the list of changes before accepting. Kernel updates will be listed in the output.

Installing the Latest Kernel on Centos 7.9

To install the latest kernel on CentOS 7.9, perform the following steps.

1. Update the system packages:

   ```
yum update
   ```

2. Install elrepo for CentOS 7.9:

   ```
yum install https://www.elrepo.org/elrepo-release-7.el7.elrepo.noarch.rpm
rpm --import https://www.elrepo.org/RPM-GPG-KEY-elrepo.org
   ```

3. Install the kernel:

   ```
yum --enablerepo=elrepo-kernel -y install kernel-ml
   ```

Configuring the Client to Boot from the New Kernel

You must configure the client to boot from the new kernel that you just installed. On Red Hat and CentOS based operating systems, this can be done with `grubby`. For distributions that do not have `grubby`, refer to the official OS documentation to ensure that the latest kernel is set for boot.

1. Identify the installed kernel from Installing the Latest Kernel on the Client. In this example we will assume it is `kernel-ml-5.4.11-1.el7.elrepo.x86_64`. In your case, the kernel will be a newer and higher number.

2. Find the new kernel grub entry with the following command.

   ```
   $ grubby --info=ALL
   ```

3. Identify the new kernel index in the output list of the command above. In the following example, the new kernel has an index value of 0 because it is at the top of the list of available kernels.
4. Use the following command to set the default kernel index value.

In this example, the new kernel grub entry index value number is 0. So we set the default index to 0. This will make the OS boot off of this kernel on the next boot.

```
$ grubby --set-default-index 0
```

5. Verify the correct kernel version is set.

```
$ grubby --default-kernel
/boot/kernel-m1-5.4.11-1.el7.elrepo.x86_64
```

6. Reboot the system to load the Lightbits kernel.

```
$ shutdown -r now
```

7. After the client reboots, you must log in and verify that the client is now running from the new kernel using the Linux command `uname -r`.

For example:

```
$ uname -r
kernel-m1-5.4.11-1.el7.elrepo.x86_64
```
Installing the Lightbits NVMe Command Line Interface

The NVMe command line interface (CLI) is a standard command line interface to run NVMe over fabrics commands from the client. Lightbits provides a customized NVMe CLI for Lightbits that will be available in future versions of the public/upstream NVMe CLI version.

The `nvme` CLI program is provided by the `nvme-cli` package. Typically this is already installed; however, if it is missing or you are unsure, you can install using `apt-get install nvme-cli`.

**How To Replace With Latest Version From Lightbits**

1. Make sure the Lightbits repository is configured on the client. Refer to Connecting to the Cluster Client DEB Repository.

2. (Optional) If a public NVMe CLI version is installed on your system, you can replace it with the NVMe CLI version supplied by Lightbits. Before installing the supplied NVMe CLI from the Lightbits repository, you'll need to remove the public NVMe CLI from your system.

   To check if you have an NVMe CLI package installed, enter the following in the system’s command shell:

   ```
   $ apt list --installed | grep nvme-cli
   ```

   WARNING: apt does not have a stable CLI interface. Use it with caution in scripts.

3. (Optional) If the command returns this value, you need to delete the NVMe CLI package from your system with the following command:

   ```
   $ apt-get remove nvme-cli
   ```

4. With the public NVMe CLI version deleted from the system, you can install the NVMe CLI from the Lightbits RPM repository by entering the following in the system’s command shell:

   ```
   $ apt-get install nvme-cli
   ```

5. Enter the following command to verify that the NVMe CLI version is v1.9.1.

   ```
   $ apt list --installed | grep nvme-cli
   ```

   WARNING: apt does not have a stable CLI interface. Use it with caution in scripts.

Installing the Lightbits NVMe Command Line Interface (Ubuntu)

The NVMe command line interface (CLI) is a standard command line interface to run NVMe over fabrics commands from the client. Lightbits provides a customized NVMe CLI for Lightbits that will be available in future versions of the public/upstream NVMe CLI version.

**Note:** These instructions will work on any Lightbits client’s side deb that you want to install on your client.

1. (Optional) If a public NVMe CLI version is installed on your system, you can replace it with the NVMe CLI version supplied by Lightbits. Before installing the supplied NVMe CLI from the Lightbits repository you’ll need to remove the public NVMe cli from your system.

   To check if you have an NVMe CLI package installed, enter the following in the system’s command shell:
$ apt list --installed |grep nvme-cli

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

nvme-cli/bionic-updates,now 1.5-1ubuntu1 amd64 [installed]

2. (Optional) If the command returns this value, you need to delete the NVMe CLI package from your system with the following command:

$ apt-get remove nvme-cli

3. With the public NVMe CLI version deleted from the system, you can install the NVMe CLI from the Lightbits RPM repository by entering the following in the system’s command shell:

$ apt-get install nvme-cli

4. Enter the following command to verify that the NVMe CLI version is v1.9-1.

$ apt list --installed |grep nvme-cli

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

nvme-cli/xenial,now 1.9~2.3.4-1~bionic amd64 [installed]

The output for this command can include additional package names with the nvme string.

Loading the NVMe/TCP Host Software and Enabling Multipath

For a client to connect to an NVMe over TCP volume on a Lightbits cluster, the nvme-tcp modules must be loaded. To fully utilize the multipathing features of Lightbits, multipath must then be enabled on the nvme-core module.

Load the NVME TCP module

To use NVMe/TCP, you must load the NVMe host modules by entering the following command in the system’s command shell.

$ modprobe nvme-tcp
$ lsmod | grep nvme

The output is similar to the following example:

<table>
<thead>
<tr>
<th>Module</th>
<th>Size</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvme_tcp</td>
<td>24576</td>
<td>0</td>
</tr>
<tr>
<td>nvme_fabrics</td>
<td>20480</td>
<td>1</td>
</tr>
<tr>
<td>nvme_core</td>
<td>49152</td>
<td>4</td>
</tr>
</tbody>
</table>

Make the setting boot persistent by loading the module on boot with this setting:
$ echo nvme_tcp > /etc/modules-load.d/nvme_tcp.conf

**Multipath**

By default, multipath should be enabled with the nvme_core module.

However, you can run the following command to check:

$ grep -r "" /sys/module/nvme_core/parameters

If you see /sys/modules/nvme_core/parameters/multipath:N, then multipath is not enabled.

Enable multipath using one of the following methods.

**Enable Multipath Using Grubby**

Figure out your current loaded kernel with grubby:

$ grubby --default=kernel

The output should show the full path of the kernel in the format of /boot/vmlinuz-...

Now configure the kernel boot arguments to load enable multipath. Make sure to put the full path of the default kernel into the command below:

$ grubby --args=nvme_core.multipath=Y --update-kernel /boot/vmlinuz-...

*Note:* grubby is available out of the box on AlmaLinux, Red Hat, and Centos based flavors. For distributions that do not have grubby, use the next method.

**Enable Multipath Using a Configuration File**

First, create a configuration file that enables multipath:

```
> echo "options nvme_core multipath=Y" > /etc/modprobe.d/50-nvme_core.conf
```

Then, update the initramfs, which OSes use to load and configure modules on boot. Use the appropriate tool for the OS: * On Red Hat/Centos (including AlmaLinux), run dracut -f. * On Debian/Ubuntu systems, run update-initramfs -u.

**Reboot**

It is recommended to reboot the client to make sure that all of the settings are loaded properly. Make sure that the nvme_tcp modules are loaded on boot and that multipath is enabled.
Provisioning Storage and Connecting the Cluster Client to Lightbits

Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Step</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

With the Lightbits software installed and the Lightbits management service running, you can create a volume and connect that volume to your application clients.

This section includes:

- Creating a Volume on the Lightbits Storage Server
- Connecting the Cluster Client to Lightbits
- Client Connection Verification

Creating a Volume on the Lightbits Storage Server

To create a volume on the cluster, log into any of the Lightbits cluster servers and enter the lbcli create volume command.

Sample Command

```
$ lbcli -J $LIGHTOS_JWT create volume --size="2 GiB" --name=vol1 --acl="acl3" --replica-count=2 --project-name=default
```

**Note:** By default, the LIGHTOS_JWT is generated during the Lightbits installation on the Ansible installation host, and is saved to `~/lightos-system-jwt`. See Post Installation Steps for an example of how to get LIGHTOS_JWT.

Sample Output
This example command creates a volume with 2 GiB of capacity, an Access Control List (ACL) string “acl3”, and a replication factor of 3.

**Note:** Only clients that mention the ACL value of “acl3” during connect can connect to this volume. This is detailed in Connecting the Cluster Client to Lightbits.

### Connecting the Cluster Client to Lightbits

After creating a volume on the Lightbits storage server, log in to one or more of your application clients and use the Lightbits NVMe CLI utility to make a connection to the Lightbits cluster.

Before you begin, enter a Linux ping command to check the TCP/IP connectivity between your application client and the Lightbits storage servers. In the example below, the client has a data NIC connected to reach the 10.10.10.x network. The IP 10.10.10.100 is data NIC of one of the Lightbits storage servers.

**Sample Command**

```
$ ping -c 1 10.10.10.100
```

**Sample Ouptut**

```
PING 10.10.10.100 (10.10.10.100) 56(84) bytes of data.
64 bytes from 10.10.10.100: icmp_seq=1 ttl=255 time=0.032 ms
--- 10.20.20.10 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.032/0.032/0.032/0.000 ms
```

This output indicates that this application client has a connection to the data NIC IP address on the Lightbits storage server where volumes were created.

Repeat this ping check for the other Lightbits cluster servers: 10.10.10.101 and 10.10.10.102.

After you have checked the TCP/IP connectivity between your application client and the Lightbits storage servers, use the nvme CLI utility to connect the application client via NVMe/TCP to the Lightbits storage server.

To use the nvme CLI utility on your application client, you will need the following details.

### Required Lightbits Storage Cluster Connection Details

<table>
<thead>
<tr>
<th>Required Data</th>
<th>Description</th>
<th>Connect Command Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightbits Data NIC IP address</td>
<td>The data NIC IP address of each Lightbits cluster node. These values can be retrieved from the Lightbits management server using the lbcli list nodes command. You used this ACL string when you created the volume on the Lightbits storage server node.</td>
<td>-a</td>
</tr>
<tr>
<td>ACL string</td>
<td></td>
<td>-q</td>
</tr>
</tbody>
</table>
Connecting the Cluster Client to Lightbits

**PROVISIONING STORAGE AND CONNECTING THE CLUSTER CLIENT TO LIGHTBITS**

<table>
<thead>
<tr>
<th>Required Data</th>
<th>Description</th>
<th>Connect Command Parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Subsystem NQN</td>
<td>The Lightbits cluster subsystem NQN value can be retrieved from the Lightbits management server using the <code>lbcli get cluster</code> command.</td>
<td>-n</td>
</tr>
<tr>
<td>TCP port</td>
<td>The data TCP port for each of the Lightbits cluster nodes can be retrieved from the Lightbits management server using the <code>lbcli list nodes</code> command.</td>
<td>-s</td>
</tr>
</tbody>
</table>

Enter the `lbcli get cluster` command on any Lightbits storage server to identify the subsystem NQN.

**Sample Command**

```
$ lbcli -J $LIGHTOSJWT get cluster -o yaml
```

**Note:** By default the LIGHTOS_JWT is generated during the Lightbits installation on the ansible installation host and is saved to `~/lightos-system-jwt`. Post Installation Steps shows one way to get LIGHTOS_JWT.

**Sample Output**

```
UUID: 95a251b6-0885-4f5b-a0eb-90e90a2009a3
currentMaxReplicas: 3
... 
subsystemNQN: nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31
  <--- subsystem NQN
supportedMaxReplicas: 3
```

Enter the `lbcli list nodes` command to identify the NIC IP address and TCP port.

**Sample Command**

```
$ lbcli -J $LIGHTOSJWT list nodes
```

**Sample Output**

```
Name     UUID                        State  NVMe endpoint    Failure
domains  Local rebuild progress     Active 10.10.10.100:4420 [server00]
server00-0 08fdb3bd-925a-5e73-adde-8daf881969d3 [server00]
server01-0 112a555f-8168-5f07-a4e0-bf8f5b59c740 [server01]
server02-0 bc759c13-856d-5521-9ba2-752259abf8f0 [server02]
```

With the IP, port, subsystem NQN and ACL values for the volume, you can execute the `nvme connect` command to connect to all of the nodes in the cluster.
Sample NVMe Connect Commands

```bash
$ nvme connect -t tcp -a 10.10.10.100 --ctrl-loss-tmo -1 -n \
 nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31 -s 4420 -q acl3
$ nvme connect -t tcp -a 10.10.10.101 --ctrl-loss-tmo -1 -n \
 nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31 -s 4420 -q acl3
$ nvme connect -t tcp -a 10.10.10.102 --ctrl-loss-tmo -1 -n \
 nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31 -s 4420 -q acl3
```

Notes: - We are using an ACL value/hostnqn of “acl3”, so that we can connect to the volume created, as detailed in Creating a Volume on the Lightbits Storage Server.
- Use the client procedure for each node in the cluster. Remember to use the correct NVME-Endpoint for each node.
- Using the `--ctrl-loss-tmo -1` flag allows for infinite attempts to reconnect nodes, and prevents a timeout from occurring when attempting to connect with a node in a failure state.
- Starting from Version 3.1.1, data IP can be IPv6.
- See the discovery-client documentation of the Lightbits Administration Guide. Like `nvme connect`, this can connect to NVMe over TCP volumes. However, it can also monitor the nodes and if new nodes/paths are created or removed, it will properly maintain those.

Confirming the Cluster Client Connection to Lightbits

Lightbits Cluster Installation Process

<table>
<thead>
<tr>
<th>#</th>
<th>Installation Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Connecting your installation workstation to Lightbits’ software repository</td>
</tr>
<tr>
<td>2</td>
<td>Verifying the network connectivity of the servers used in the cluster</td>
</tr>
<tr>
<td>3</td>
<td>Setting up an Ansible environment on your installation workstation</td>
</tr>
<tr>
<td>4</td>
<td>Installing a Lightbits cluster by running the Ansible installation playbook</td>
</tr>
<tr>
<td>5</td>
<td>Updating clients (if required)</td>
</tr>
<tr>
<td>6</td>
<td>Provisioning storage, connecting clients, and performing IO tests</td>
</tr>
</tbody>
</table>

Each `/dev/nvmeX` is a successful NVMe over TCP connection to a server in the cluster. When the optimized path is connected, a block device is created with the name `/dev/nvmeXnY`, which can then be used as any block device (create fs on top of it and mount it).

If you see a multipath error (with the `nvme` block devices showing up as 0 byte, or each replica/nvme connection showing up as a separate nvme block device), refer to the Lightbits Troubleshooting Guide, or contact Lightbits Support.

After you have entered the `nvme connect` command, you can confirm the client’s connection to Lightbits by entering the `nvme list` command. This will list all of the NVMe block devices. For more information on each connection’s multipathing, you can use `nvme list-subsys`, which will list all of the NVMe character devices.
Confirming the Cluster Client Connection to Lightbits

**Note:** The `nvme list` and `lsblk` command will show the NVMe block device that is created upon a successful connection. It will be of the format `nvme0n1`. The `nvme list-subsys` will list all of the paths that make up these block devices; these paths appear as character devices. So from the output below we can conclude that block device `nvme0n1` is made of 3 character devices: `nvme0`, `nvme1`, and `nvme2`. When we need to interact with the block device - for example to create a filesystem and mount it - we will interact with the block device, `nvme0n1`, and not the character devices (nvme0, nvme1, and nvme2).

### Sample Command

```
$ nvme list-subsys
```

### Sample Output

```
nvme-subsys0 - NQN=qqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d3
  \        
  +- nvme0 tcp traddr=10.10.10.100 trsvcid=4420 live
  +- nvme1 tcp traddr=10.10.10.101 trsvcid=4420 live
  +- nvme2 tcp traddr=10.10.10.102 trsvcid=4420 live
```

Next, review your connected block devices to see the newly connected NVMe/TCP block device using the Linux `lsblk` command.

### Sample Command

```
$ lsblk
```

### Sample Output

```
NAME     MAJ:MIN RM  SIZE   RO TYPE MOUNTPOINT
nvme0n1   259:1   0  2G   0  disk         
sdb       8:16   0 223.6G 0  disk         |
        |-sdb2  8:18   0 222.6G 0  part        
        |    `-CentOS00-swap  253:1  0  22.4G 0  lvm  [SWAP]
        |    `-CentOS00-home  253:2  0  150.2G 0  lvm  /home
        |    `-CentOS00-root  253:0  0  50G   0  lvm  /
        |    `-sdb1  8:17   0  1G   0  part  /boot
sda       8:0    0 111.8G 0  disk
```

A new `nvme0n1` block device with 2GB of storage is identified and available.

To determine which node in the cluster is the primary and which is secondary for this block device, enter the `nvme list-subsys` command with the block device name.

### Sample Command

```
$ nvme list-subsys /dev/nvme0n1
```
Sample Output

```
nvme-subsys0  - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31
  +- nvme0 tcp traddr=10.10.10.100 trsvcid=4420 live optimized
  +- nvme1 tcp traddr=10.10.10.101 trsvcid=4420 live inaccessible
  +- nvme2 tcp traddr=10.10.10.102 trsvcid=4420 live
```

In the output, the optimized status identifies the primary node, and an inaccessible status for the secondary node. In this case we can see that server 10.10.10.100 is the primary node with the optimized path. All of the IO from the client will go to 10.10.10.100. The cluster will then replicate the data between the other nodes.

**Troubleshooting**

**Note:** For additional troubleshooting-related information, see the Lightbits Troubleshooting Guide, or contact Lightbits Support.

**Ansible Role Errors**

Confirm that the durosight ports are synchronized in the Ansible default yml file, which can be overridden in inventory ymls, and the node-manager configuration Ansible default yml:

```
~/light-app/ansible/roles/install-lightos/defaults/main.yml
```

**SSH Strict Key Errors When Using sshpass**

If you use the sshpass utility method in your hosts file, you can receive an error related to SSH keys in the Known Hosts file, such as:

```
$ ansible -i ansible/inventories/cluster_example/hosts all -m ping
node02 | FAILED! => {
    "msg": "Using a SSH password instead of a key is not possible because Host Key checking is enabled and sshpass does not support this. Please add this host's fingerprint to your known_hosts file to manage this host."
}
```

To avoid this error, you need to disable `StrictHostKeyChecking` in the `/etc/ssh/ssh_config`, or log into each node from your installation workstation at least once.

By default, `StrictHostKeyChecking` is enabled in the file `/etc/ssh/ssh_config`. You can disable this by un-remarking it in ssh_config and setting it to:

```
StrictHostKeyChecking no
```

Or, you can leave `StrictHostKeyChecking` enabled and log into each node from the installation workstation and “answer yes” to permanently add the host to the Known Hosts files.

The first time you SSH from one server to another the following SSH exchange occurs:
So, by logging into all the servers at least once from your installation workstation before you run the Ansible playbook, there will be no issues using the sshpass method.

Free Space in Linux OS for etcd Logical Volume Manager Use

If your Linux operating system has volume groups that were created for the home, root, and swap file systems and are utilizing 100% of the storage, you must reduce one of these volume groups. The Lightbits installer requires at least 10GB of space to create an LVM for use with etcd.

For example, review the Linux OS logical volumes. The Linux Virtual Server (LVS) software is used in this example.

```bash
$ lvs
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync
  Convert
home CentOS_lightos-c3 -wi-ao---- <64.24g
root CentOS_lightos-c3 -wi-ao---- 50.00g
swap CentOS_lightos-c3 -wi-ao---- 4.00g

$ lvs
LV VG Attr LSize Pool Origin Data% Meta% Move Log Cpy%Sync
  Convert
home CentOS_lightos-c3 -wi-ao---- <64.24g
root CentOS_lightos-c3 -wi-ao---- 50.00g
swap CentOS_lightos-c3 -wi-ao---- 4.00g
```

**Note:** If the Linux Virtual Server (LVS) software reports anything but “CentOS” for the Volume Group name used for the LinuxOS file system, you will need to specify the exact name in the ~/light-app/ansible/inventories/cluster_example/host_vars file for that node. For more information, see the `etcd_vg_name` variable description in the Host Configuration File Variables list.

In this example, the LinuxOS was installed onto a 118 GB drive and the entire amount is allocated. You can resize the home LVM by 20 GB to free up some space.

To resize this file system, you need to:

1. Move any files you have in the `/home` file system to a safe location.
2. Unmount, resize, and recreate the file system.
3. Remount the file system.

To identify how much space is available to free up, use `lsblk` as follows:
Recovering from Cluster Installation Failure

In this example, the LinuxOS is installed on device “sda” and on partition sda1 with 119.2 GB of space available. It is possible to take 20 GB away from home to free up some space and still have over 44 GB remaining.

1. Mount and record the current mount path for home.

   $ mount
   /dev/mapper/CentOS_lightos--c3-home on /home type xfs (rw,relatime,attr2,inode64, noquota)

2. Unmount home and then resize it.

   $ umount /home
   $ lvresize -L -20G CentOS_lightos-c3/home

3. Remake the home file system.

   $ mkfs.xfs -f /dev/mapper/CentOS_lightos--c3-home

4. Remount home.

   $ mount /dev/mapper/CentOS_lightos--c3-home /home

Recovering from Cluster Installation Failure

At times during cluster deployment, errors occur and the configuration step must be retried. To do that, a playbook is provided to stop all services and delete the data-plane and control-plane data and configuration.

Cleanup command for a full cluster:

```bash
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/cleanup-lightos-playbook.yml -t cleanup
```

Cleanup for one Lightbits server:

```bash
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/cleanup-lightos-playbook.yml -t cleanup --limit <server_name>
```
Log Artifacts Collection

Reconfigure command:

```bash
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/configure-lightos-playbook.yml
```

The following is important for better understanding cleanup and configure:

- When a Lightbits installation is done via a deploy-lightos playbook - as described in Running the Ansible Installation Playbook to Install Lightbits Cluster Software - it runs two playbooks in order. First it runs an install playbook, which installs all of the Lightbits dependencies and packages and does a reboot. Then it runs the configure playbook, which sets up all of the Lightbits services.
- The cleanup playbook removes all of the Lightbits configurations. It does not uninstall any of the packages that were installed.
- The configure playbook does not install any Lightbits packages; it simply reconfigures all of them. Only run this if you are certain that the deploy-lightos playbook ran through the install playbook on all servers; otherwise, use the deploy-lightos playbook as described in Running the Ansible Installation Playbook to Install Lightbits Cluster Software.

Log Artifacts Collection

Logs can be gathered from each Lightbits server configured in the `hosts` file.

```bash
$ ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/logs.yml
```

The output of each Lightbits server is saved into a dated directory inside `/tmp/` on the Ansible host.

**Notes:**
- This can be run against servers that have Lightbits installed, as well as servers that do not have Lightbits installed.
- To properly gather logs from Lightbits servers, the playbook will depend on the jwt being in the `~/lightos-system-jwt` file.
- If a node is inactive, unattached, or services seem unresponsive, collect logs using `lb-support.sh -s 7 -l $ (hostname -s)`. The `-s 7` parameter specifies a seven-day log capture duration; you can adjust this as needed, based on when the issue began. The `-l $(hostname -s)` option appends the server’s hostname to the log tarball, aiding in file differentiation. Forward the issue and tarball to Lightbits Support for evaluation, and to determine if intervention or a restart of the NM, CM, or server is needed.

Fully Clean Lightbits From Servers or Cluster

To fully clean Lightbits packages and configuration from a server, run the following steps.

**Note/Caution:** Do not run this on servers that are active or show up on a Lightbits cluster. Only run this against servers that need Lightbits removed; otherwise, the cluster state will be in danger. All of the commands run from the servers must be run as highest privilege (root).

1. From the Ansible host, run the cleanup playbook to unconfigure the server.

```bash
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/cleanup-lightos-playbook.yml -t cleanup --limit <server_name>
```
Note: Make sure matches the name of the server from where Lightbits must be removed. Typically, it will be of the form “server00” or “server01”, etc.

2. Erase ETCD from the Lightbits server.

```
rm -vrf /usr/bin/etcd /usr/bin/etcdctl /etc/etcd /opt/etcd /var/lib/etcd
rm -vf /etc/systemd/system/etcd.service
rm -vf /etc/systemd/system/multi-user.target.wants/etcd.service
```

3. Uninstall Lightbits packages with these steps from the Lightbits server.

First, find out the version of Lightbits that is installed.

```
lbcli version | awk '{print $NF}'
```

The output will be like “3.0.1~b1004”. This is the format used for the next steps.

You can also see the latest installed rpms using `rpm -qa --last`, which will list the latest installed packages at the top of the list.

Extract the version number in a format similar to this: “A.B.C~bD”. In this example, it is 3.0.1~b1004. But each case could be different.

```
rpm -qa | grep 3.0.1~b1004
```

Now you can manually remove each package with `yum remove PKG -y` or `rpm -e PKG`. However, if all of the packages look Lightbits-related, then run the next command and it will uninstall them.

Important: Make sure to replace 3.0.1~b1004 with your version.

```
bash <(echo "(" set -xeu"; rpm -qa | grep 3.0.1~b1004 | xargs -I {} echo yum remove {} -y; echo ")")")
```

4. For GA (sunsetted releases), Lightbits installed a custom kernel. To fully uninstall Lightbits from a server, it is recommended to uninstall the Lightbits kernel and set another kernel as the default. Refer to your OS documentation on how to uninstall a kernel and set another kernel as default.

Note: The Lightbits Red Hat releases do not need to run this step. Instead, edit `/etc/yum.conf` and remove or comment out the line that shows `exclude=redhat-release* kernel* kmod-kvdo*`.

5. Clean up the repo from any Lightbits artifacts.

```
rm -f /etc/yum.repos.d/lightos.repo
yum clean all
```

Appendixes

The following sections provide additional information to help you complete the Lightbits installation.

This section includes:
Host Configuration File Variables

Each host configuration file includes some basic configuration variables.

See Host Configuration File Examples for instances of how these variables are used in a host configuration file.

```
data_ifaces:
  bootproto:
  ifname:
  ip4:
  ip6:
instances:
  instanceID:
  data_ip:
  failure_domains:
  ec_enabled:
  storageDeviceLayout:
  initialDeviceCount:
  maxDeviceCount:
allowCrossNumaDevices:
  deviceMatchers:
    - model =~
    - partition ==
    - size >=
    - path =
    - name =~
use_lvm_for_etcd:
  etcd_lv_name:
  etcd_settings_user:
  etcd_lv_size:
  etcd_vg_name:
  auto_reboot:
  datapath_config:
  listen-client-urls:
```

Host Configuration File Variable Notes
<table>
<thead>
<tr>
<th>Variable</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data_ifaces</td>
<td>No</td>
<td>If provided, the Ansible playbook configures the interface. The configuration is permanent, and results in a new <code>ifcfg-&lt;iface-name&gt;</code> file. If this variable is not provided, no action is taken, the playbook assumes that the interfaces are valid, and the link is up and configured. If data-ifaces is used, you must also use the bootproto, conn_name, ifname, and ip4 variables.</td>
</tr>
<tr>
<td>bootproto</td>
<td>No</td>
<td>IP allocation type (dynamic or static). Only static is supported in Lightbits 2. Default value: Static.</td>
</tr>
<tr>
<td>ifname</td>
<td>No</td>
<td>The interface name, such as eth0 or enp0s2, that the data path in the ip4 variable is dedicated to.</td>
</tr>
<tr>
<td>ip4</td>
<td>No</td>
<td>The data IPv4 address and subnet to set for the interface mentioned in the ifname field. The format is “IP/subnet”. Example: “10.10.10.100/24”</td>
</tr>
<tr>
<td>ip6</td>
<td>No</td>
<td>The data IPv6 address and prefix to set for the interface mentioned in the ifname field. The format is “IP/prefix”. Example: “2001:0db8:0:f101::1/64”</td>
</tr>
<tr>
<td>data_ip</td>
<td>Yes</td>
<td>The data/etcd IP used to connect to other nodes. The subnet or prefix is not required, only the address.</td>
</tr>
<tr>
<td>instances</td>
<td>Yes</td>
<td>A list of instance IDs, one for each logical data-path instance.</td>
</tr>
<tr>
<td>failure_domains</td>
<td>No</td>
<td>The servers sharing a network, power supply, or physical location that are negatively affected together when network, power, cooling, or other critical service experiences problems. For more information, see the Defining Failure Domains procedure. Default value: Empty list.</td>
</tr>
<tr>
<td>instanceID</td>
<td>Yes</td>
<td>A unique number assigned to this logical node. Only two logical nodes per server are supported in Lightbits. This means that the value is “0” and/or “1”.</td>
</tr>
<tr>
<td>storageDeviceLayout</td>
<td>Yes</td>
<td>The storageDeviceLayout key, under the node-specific settings, groups the information required to detect the initial storage configuration of the node.</td>
</tr>
<tr>
<td>initialDeviceCount</td>
<td>Yes</td>
<td>A setting specifying the initial count of physical drives the system will start with on the first startup.</td>
</tr>
<tr>
<td>Variable</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------</td>
<td>----------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>maxDeviceCount</td>
<td>Yes</td>
<td>The pre-determined, maximum number of physical nvme drives that this node can contain; i.e. the number of NVME drive slots available for this node.</td>
</tr>
<tr>
<td>allowCrossNumaDevices</td>
<td>No</td>
<td>An optional setting, specifying whether block devices can be used by system-nodes that are affiliated with a different Numa ID, or instance IDs that the block device is attached to. Default: false. <strong>Note:</strong> Do not allow devices attached on different NUMAs to be used by this node.</td>
</tr>
<tr>
<td>deviceMatchers</td>
<td>No</td>
<td>This section contains a list of matching conditions for locating the wanted physical drives to be used by the system.</td>
</tr>
<tr>
<td>partition</td>
<td>No</td>
<td>Whether or not a device is a partition (true/false).</td>
</tr>
<tr>
<td>model</td>
<td>No</td>
<td>The vendor/model of the device. For example: name =~ “nvme5n.*” or name == “nvme1n1”.</td>
</tr>
<tr>
<td>name</td>
<td>No</td>
<td>The file name of the device.</td>
</tr>
<tr>
<td>path</td>
<td>No</td>
<td>The full path of the device.</td>
</tr>
<tr>
<td>size</td>
<td>No</td>
<td>The capacity of the physical device. size &gt;= mib(1000), size == gib(20), size &lt;= tib(50).</td>
</tr>
<tr>
<td>ec_enabled</td>
<td>Yes</td>
<td>Enables Erasure Coding (EC) for protecting against SSD failure within the storage server. Normal operation continues during reconstruction when a drive is removed. At least six NVMe devices must be in the node for erasure coding to be enabled.</td>
</tr>
<tr>
<td>name</td>
<td>Yes</td>
<td>A unique, user-friendly name for the node.</td>
</tr>
<tr>
<td>use_lvm_for_etcd</td>
<td>No</td>
<td>Use the Linux Volume Manager (LVM) partition for etcd data. Default value: false. <strong>Note:</strong> If this variable is not used in the host configuration file, the system uses the default fault value. The following etcd variables are only relevant if the use_lvm_for_etcd variable value is true.</td>
</tr>
<tr>
<td>etcd_lv_name</td>
<td>No</td>
<td>Logical volume name for etcd data local volume management.</td>
</tr>
<tr>
<td>etcd_settings_user</td>
<td>No</td>
<td>Key-value map for overriding the etcd service settings.</td>
</tr>
<tr>
<td>etcd_lv_size</td>
<td>No</td>
<td>Logical volume size for etcd data local volume management.</td>
</tr>
<tr>
<td>etcd_vg_name</td>
<td>No</td>
<td>Volume group name for etcd data local volume management.</td>
</tr>
<tr>
<td>datapath_config</td>
<td>Yes</td>
<td>The path to the system-profile yml file.</td>
</tr>
</tbody>
</table>
### Variable Required Description

<table>
<thead>
<tr>
<th>Variable</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>etcd_settings_user</td>
<td>No</td>
<td>User etcd settings.</td>
</tr>
<tr>
<td>listen-client-urls</td>
<td>No</td>
<td><a href="http://127.0.0.1:2379">http://127.0.0.1:2379</a></td>
</tr>
<tr>
<td>profile_generator_overrides_dir</td>
<td>No</td>
<td>Directory path containing &lt;system-profile&gt;.yaml file to override the profile-generator generated one.</td>
</tr>
<tr>
<td>auto_reboot</td>
<td>No</td>
<td>If set to false the system will not automatically reboot after installation.</td>
</tr>
</tbody>
</table>

**Notes:**
- The user must provide the etcd volume group name in the `etcd_vg_name` variable, and confirm that there is enough server space to create a new logical volume. The default logical volume name (`etcd_lv_name`) is “etcd” and the default volume size (`etcd_lv_size`) is 10GB.
- If there is not enough space in the server, the user must reduce the other logical volume sizes before the cluster software installation to allocate the required space. For more details, see [https://www.rootusers.com/lvm-resize-how-to-decrease-an-lvm-partition](https://www.rootusers.com/lvm-resize-how-to-decrease-an-lvm-partition).

### Host Configuration File Examples

This section covers various configuration examples and how they affect the host configuration files. The first host configuration file will be shown server00.yml, and the host configuration files for the remaining servers will follow the same pattern. For more complex examples, two host configuration files will be shown to help better understand the configuration.

**Example 1: Data Network Interface Manually Configured**

Host configuration with no data interfaces (data_ifaces) provided. The playbook does not need to configure the IP on the data interface. The user configured the interfaces prior to running the playbook.

```yaml
---
nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
- server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition == false
  - size >= gig(300)
  # - name =~ "nvme0n1"
```

**Example 2: Data Network Interface Automatically Configured**

Host configuration with a single data interface. The playbook configured the interface. The user did not configure the interface with the data IP.
---
data_ifaces:
- bootproto: static
  conn_name: ens1
  ifname: ens1
  ip4: 10.10.10.100/24

nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
- server00
  ec_enabled: true
  lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
  deviceMatchers:
  # - model =~ ".*"
  # - partition == false
  # - size >= gib(300)
  # - name =~ "nvme0n1"

---

Note: The example above shows the format for setting IPv4 addresses using `ip4: ip/subnet`. IPv6 addresses can be set using `ip6: ip/prefix`.

Example 3: Override the Lightbits Configurations

Host configuration with Lightbits override. The provided value overrides the key `listen-client-urls`.

---
nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
- server00
  ec_enabled: true
  lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
  deviceMatchers:
  # - model =~ ".*"
  # - partition == false
  # - size >= gib(300)
  # - name =~ "nvme0n1"
etcd_settings_user:

Example 4: Provide Custom Datapath Configuration

Host configuration with custom datapath configuration provided.
By default, the playbook inspects the remote machine and determines the directory containing the specific configuration for Duroslight and backend services (datapath configuration). The excluding node-manager configuration uses the following logic:

```
<system_vendor>-<processor_count>-processor-<processor_cores>-cores
```

---

```yaml
nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
  - server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:

#  - model =~ ".*"
#  - partition == false
#  - size >= gib(300)
#  - name =~ "nvme0n1"

datatpath_config: custom-datapath-config
```

---

**Example 5: Use the Linux Volume Manager (LVM) Partition for etcd Data**

Host configuration with custom lvm partition for etcd data.

```yaml
nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
  - server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 6
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:

#  - model =~ ".*"
#  - partition == false
#  - size >= gib(300)
#  - name =~ "nvme0n1"

use_lvm_for_etcd: true
etcd_lv_name: etcd
#etcd_settings_user:
etcd_lv_size: 15GiB
etcd_vg_name: centos
**Example 6: Profile-Generator Overrides**

Enable humans to override profile-generator output and provide for each server a custom file that will be taken by profile-generator as the system-profile.

Each host may be different so each host can specify its own override file.

```
nodes:
- instanceID: 0
  data_ip: 10.10.10.100
  failure_domains:
    - server00
  ec_enabled: true
  storageDeviceLayout:
    initialDeviceCount: 6
    maxDeviceCount: 12
    allowCrossNumaDevices: false
    deviceMatchers:
      - partition == false
      - size >= gib(300)
  profile_generator_overrides_dir: /tmp/overrides.d/server00
```

In case the cluster is homogeneous and we want to apply the same override to all nodes, we can provide a single setting in the `groups/all.yml` file or via the cmd with:

```
ansible-playbook -i ansible/inventories/cluster_example/hosts playbooks/deploy-lightos.yml -e profile_generator_overrides_dir=/tmp/overrides.d
```

In the above example, we specify `profile_generator_overrides_dir` which is a directory on the Ansible Controller that will be copied to the target machine.

**Example 7: Dual Instance Configuration**

In a dual instance configuration, Lightbits is run from both NUMAs of the CPU.

Below is an example configuration for `server00.yml` and `server01.yml`.

In this configuration, both servers NUMA0 and 1 have 12 NVME drives, for a total of 24 NVME drives in the server.

Each server has a single management IP configured (not shown below) and two data IPs (one for each NUMA). Each instance has a unique data IP set for it.

```
server00.yml
```
---
name: server00
nodes:
- instanceID: 0
data_ip: 172.16.10.10
failure_domains:
- server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 12
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  # - partition == false
  # - size >= gib(300)
  # - name =~ "nvme0n1"
- instanceID: 1
data_ip: 172.16.20.10
failure_domains:
- server00
ec_enabled: true
lightfieldMode: SW_LF
storageDeviceLayout:
  initialDeviceCount: 12
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  # - partition == false
  # - size >= gib(300)
  # - name =~ "nvme0n1"

server01.yml
Example 8: Single IP Dual NUMA Configuration

This example is for Single IP Dual Instance configuration. For more information, see Single-IP-Dual-NUMA Configuration.

The below is an example of a full server00.yml, with a similar physical configuration as the above example. However, instead of using two data IPs, it only uses one data IP for both of its instances.

Note: Unlike dual instance configuration from the example above, which had a unique data IP per instance, each instance on a server has the same IP.
Performing an Offline Installation

The offline installation scenario is used when there is no internet access to download the required Lightbits RPMs and their dependencies. In such a case, the machine being used for installation should include the Lightbits cluster software RPM files and their dependencies.

During the offline installation, the software packages are copied to the target machine and installed locally.

To complete the offline installation:

1. Copy the packages to the installation server.
2. Enter the following commands on the installation machine under hosts. See Creating the Inventory Structure and Adding the Ansible Hosts File.

```
source_type=offline
source_etcd_binary=<path to etcd binary zip>
source_rpms_dir=<path to rpms>
source_dependencies_rpms_dir=<path to dependencies rpms>
dest_dir=<path>
```

For example:
Offline Ansible Controller Installation and Self-Signed Certificates

The Lightbits cluster installation requires SSL certificates. You can provide these certificates, or the playbook will automatically generate self-signed certificates. To create these certificates Ansible downloads some binaries from the internet at runtime.

In case of an offline Ansible controller, the installation script requires that certificates_directory will be present and contain all needed certificates before running the playbook.

This directory and its content can come from two sources:

- If you bring your own organization certificates.
- Use the self-signed certificates generated by the initial cluster installation process.

You will need to copy the folder credentials_directory to the Ansible Controller machine, before running the installation script.

Configuring the Data Network

Nodes in the Lightbits server clusters communicate via a high-speed data network interface.

All nodes in the cluster must be configured with an IP address from the same accessible network before running the Ansible playbook.

You can configure the network using an automatic (recommended) or manual method.

**Note:** For full examples, see Host Configuration File Examples.

Automatic Data Network Configuration (Recommended)

The Ansible playbook can help you automatically set the data interface IP when some optional network host variables are transferred.

This means that to make deployment easier the playbook configures the data network interface persistently for you by specifying for each host the list variable. For example:

```yaml
data_ifaces:
  - bootproto: static
    conn_name: ens1
    ifname: ens1
    ip4: 10.20.20.10/27
```

In this example, we have set the playbook to permanently configure interface ens1 with static IP 10.20.20.10.

Manual Data Network Configuration

In this method, you assign the data IPs on the data interfaces for each node on the cluster.

To set the data IPs:

1. Log into the machine with the following command:


etcd Partitioning

Based on your boot device's write latency performance, you might need to create a separate partition for etcd data on the boot device. If you have questions about the need to use etcd partitioning, contact Lightbits.

To use etcd partitioning:

1. Confirm that a partition pre-allocated for etcd exists on the node and has at least 10 GB of space.
2. If it does not already exist on the node, configure an LVM group.
3. Enter the LVM group name for the `etcd_vg_name` variable in the host configuration file.

**Using SSH-Key Authentication**

To use key authentication, you must provide the SSH key file used in all the cluster servers to the `hosts` file, which is usually located from the `light-app` directory in this path: `ansible/inventories/cluster_example/hosts`.

To use SSH-key authentication instead of a plain text password, see the knowledge base article How To Configure SSH Key-Based Authentication on a Linux Server.

After you have configured the SSH key for authentication, you can connect from the installation server to the target with the `ansible_ssh_private_key_file` instead of `ansible_ssh_pass`, in the following format:

```plaintext
ansible_ssh_private_key_file=<private RSA key file path>
```

With the default configuration, the top section of the `hosts` file lines are configured as below:

```
server00 ansible_host=rack11-server92 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=light ansible_become_user=root ansible_become_pass=light
```

As an example, assume that the SSH key for the servers is located at `/root/mykey.txt`. If so, change the configuration line to this:

```
server00 ansible_host=rack11-server92 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_private_key_file=/root/mykey.txt ansible_become_user=root ansible_become_pass=light
```

**Network Time Protocol Configuration**

The Network Time Protocol (NTP) must be installed and configured on the cluster nodes to keep the cluster nodes in sync with each other.

**Note:** For operating systems (AlmaLinux and Red Hat) that use chrony, skip these steps and refer to the chrony documentation regarding the time syncing. Also refer to Configuring Global Variables in Ansible to properly configure the Ansible files for chrony.

You can use one of the following methods to install the required NTP packages.

**Method 1**

The latest RPMs are retrieved from the OS repository and installed on the cluster nodes.

**Method 2**

The specific NTP version required by the customer is installed on the cluster nodes. To use this method:

1. Edit the `all.yml` file: `~/light-app/ansible/inventories/cluster_example/group_vars/all.yml`.
2. Edit or append the following to the `all.yml` file, using the specific version that you want to install. For example:

```
ntp_version: ntp-4.2.6p5-29.el7.CentOS.x86_64
```
Method 3

The NTP is installed using an offline method.

1. Edit the all.yml file: 

   `~/light-app/ansible/inventories/cluster_example/group_vars/all.yml`

2. Under the new `group_vars` directory, create a new all.yml file.

3. Edit or append the following section to the all.yml file with the desired packages to install. This section's order first lists the prerequisites and then the desired package. For example:

   ```
   ntp_packages:
   - "autogen-libopts*.rpm"
   - "ntpdate*.rpm"
   - "ntp*.rpm"
   ```

4. The desired NTP packages must be copied to the `dest_dir`. For more, see Performing an Offline Installation.

Configuring the NTP Server

After you have installed NTP on all of the cluster nodes, you must configure the NTP service to sync with a global NTP server that is inside or outside the enterprise.

The default NTP configuration is implemented during the cluster software installation and configuration process run by the Ansible tool, which uses the defaults provided in the NTP package (Global Server Pool).

To overwrite the defaults provided in the NTP package and provide these overrides to other NTP servers, complete the following steps:

1. Edit the all.yml file: 

   `~/light-app/ansible/inventories/cluster_example/group_vars/all.yml`

2. Edit or append the following sections to the all.yml, using the relevant NTP servers for your system.

   ```
   ntp_enabled: true
   ntp_manage_config: true
   ntp_servers:
   - "0{{ '. ' + ntp_area if ntp_area else '' }}.pool.ntp.org iburst"
   - "1{{ '. ' + ntp_area if ntp_area else '' }}.pool.ntp.org iburst"
   - "2{{ '. ' + ntp_area if ntp_area else '' }}.pool.ntp.org iburst"
   - "3{{ '. ' + ntp_area if ntp_area else '' }}.pool.ntp.org iburst"
   ```

Additional Note

In order to ensure NTP client consistency and synchronization with the NTP servers, it is highly recommended to eliminate the NetworkManager from updating /etc/resolv.conf. Incorrect configuration of the file could cause the NTP client to communicate with the NTP server, and therefore create time drifting between the cluster nodes.

This can be done by:

As the root user, create the /etc/NetworkManager/conf.d/90-dns-none.conf file with the following content - by using a text editor:
Automated Client Connectivity Verification

After you finish installing Lightbits and configuring the nodes for a cluster, you can use an optional Ansible playbook that verifies the success of the cluster installation and runs basic IO operations to verify the client connectivity.

To use this optional Ansible playbook, the following must be configured:

1. In the host files, the Ansible host file must have the “initiators” section to declare another client server. For more, see Creating the Inventory Structure and Adding the Ansible Host File.

2. If you want the Ansible script to configure the P, you must add the host variables file that includes the `data_ifaces` section. For more, see Configuring the Data Network.

**Note:** It is important that the inventory folder is shared with the cluster inventory folder so that you can fetch all cluster IPs.

Enter the following command to start the Ansible playbook:

```bash
$ ansible-playbook -i <hosts file> playbooks/deploy-nvme-tcp-initiator.yml
```

Configuring Grafana and Prometheus

Prometheus gathers statistics from the Lightbits cluster. Grafana in turn represents everything in graphs on dashboards. This monitoring package can monitor several clusters at once, and multiple clusters can be configured.

**Prerequisite**

- docker-ce

**Installing Grafana and Prometheus**

**Note:** These monitoring packages should be installed on host machines, not on the Lightbits target servers.

```bash
sudo yum install lightos-monitoring-images lightos-monitoring-clustering
```

**Note:** See Connecting to the Lightbits Software Repository for additional information.

**Usage**

After lightos monitor rpms (lightos-monitoring-clustering, lightos-monitoring-images), run the following:
Edit the following file:

```
/var/lib/monitoring-clustering/configure_grafana/configure_grafana.yml
```

In the Clusters section, change the instance names for your cluster hosts (remove the extra lines in case of a single cluster).

```
clusters:
  cluster_1:
    - rack01-server01
    - rack02-server02
    - rack03-server03
  cluster_2:
    - rack04-server04
    - rack05-server05
```

Then run:

```
/var/lib/monitoring-images/deploy.sh configure-monitor
```

**Outcome**

Running the following:

```
docker ps
```

- We should see two Dockers running Prometheus and Grafana.
- They are running on port http://localhost:9090 (Prometheus) and http://localhost:3000 (Grafana).
- The Grafana user/password is:
  - user: admin
  - password: foobar
- Inside Grafana we should have two dashboards:
  - cluster_tab - showing information about the cluster.
  - nodes_tab - showing information per node.

**Lightbits Monitoring Integration with Existing Grafana and Prometheus**

You can also integrate Lightbits’ reference monitoring metrics with your existing Prometheus and Grafana platform.

**Getting the Metrics**

After installing Lightbits monitoring, you can get Lightbits’ reference monitoring metrics in the `/var/lib/monitoring-clustering/` folder, as illustrated in the example below.
Integrating Grafana

There are two options for integrating the Grafana reference metrics: * Manually create the data source for Lightbits Prometheus with the Grafana GUI, and then manually create a dashboard by importing reference metrics. * Integrate the reference files directly, as shown in the example below:

Merge the data source configuration in `monitoring-clustering/configure_grafana/roles/grafana/defaults/main.yml` with the existing data source.

Note that a different version of Grafana may have a different format for the configuration.

You can also easily create a data source manually with the GUI.
Configuring Grafana and Prometheus

APPENDIXES

[ ]

```
[root@localhost ~]# vim /usr/share/grafana/conf/provisioning/datasources/sample.yaml
...
data sources:
  - name: Prometheus
type: prometheus
  url: http://localhost:9090
```

Copy the Dashboard metrics files to the Grafana configuration folder.
```
[root@localhost ~]# tree monitoring-clustering/grafana/
dashboards
  * cluster_tab.json
  * nodes_tab.json
  * performance_tab.json
```

```
[root@localhost ~]# vim /usr/share/grafana/conf/provisioning/dashboards/sample.yaml
...
providers:
  - name: 'default'
    orgId: 1
    folder: '
    folderUid: '
    type: file
    options:
      path: /var/lib/grafana/dashboards
```

```
[root@localhost ~]# mkdir /var/lib/grafana/dashboards
[root@localhost ~]# cp monitoring-clustering/grafana/dashboards/* /usr/share/grafana/
  conf/provisioning/dashboards/ -a
# restart the Grafana service
```

```
[root@localhost ~]# /usr/sbin/grafana-server -homepath /usr/share/grafana/
```

Use the GUI to verify the result. Access the Prometheus GUI using the instructions above. For example:
http://localhost:9090/ or http://monitoring-server:9090/. Note that when using the installation above, the Grafana and Prometheus are the same host.

Integrating Prometheus

To integrate Prometheus, merge the configuration inside the Lightbits reference Prometheus configuration files and Lightbits reference configure files - as shown below:

```
[root@localhost ~]# tree monitoring-clustering/prometheus/
  monitoring-clustering/prometheus/
    alert.rules.yaml
    prometheus.yaml
    record.rules.yaml
```

You will need to manually merge contents inside of prometheus.yaml with your existing prometheus.yaml.
[root@localhost prometheus]# vim /usr/local/prometheus/prometheus.yml

... 
rule_files:
- "alert.rules.yaml"
- "record.rules.yaml"

A scrape configuration containing exactly one endpoint to scrape. Here, it is Prometheus itself:
scrape_configs:
- job_name: lightos
  scheme: http
  scrape_timeout: 25s
  scrape_interval: 30s
  metrics_path: /metrics
  honor_timestamps: True
  params:
    collect[]:
    - clustering
    - datapath
    - meminfo
    - textfile
    - lightfield
    - netstat
    - netdev
    - cpufreq
  file_sd_configs:
    - refresh_interval: 10s
  files:
    - 'file_sd_configs/lightbox-exporter/*.yaml'

- job_name: lightos-smart
  scheme: http
  scrape_timeout: 10s
  scrape_interval: 5m
  metrics_path: /metrics
  honor_timestamps: True
  params:
    collect[]:
    - smart
  file_sd_configs:
    - refresh_interval: 10s
  files:
    - 'file_sd_configs/lightbox-exporter/*.yaml'

- job_name: api-service
  scheme: https
  tls_config:
    insecure_skip_verify: True
  scrape_timeout: 10s
  scrape_interval: 15s
  metrics_path: /metrics
  honor_timestamps: True
  file_sd_configs:
    - refresh_interval: 10s
  files:
    - 'file_sd_configs/api/*.yaml'

And copy the other two associated rule configuration files to the Prometheus configuration file folder.

[root@localhost prometheus]# cp alert.rules.yaml record.rules.yaml /usr/local/prometheus/
Copy the Lightbits reference target files to the Prometheus configuration file folder, and update the IP address of the Lightbits cluster.

```
[root@localhost ~]# tree monitoring-clustering/file_sd_configs
monitoring-clustering/file_sd_configs
  api
    targets.yaml
  lightbox-exporter
    targets.yaml
[root@localhost ~]# cp monitoring-clustering/file_sd_configs /usr/local/prometheus/ -a
```

Update the prometheus.yml with the new location of the target files.

```
[root@localhost prometheus]# vim /usr/local/prometheus/prometheus.yml
... files:
  - 'file_sd_configs/lightbox-exporter/*.yaml'
... 
```

Restart the Prometheus service.

Use the GUI to check whether Prometheus is working properly. Access the Prometheus GUI using the instructions above. For example: http://localhost:9090/ or http://monitoring-server:9090/. Note that when using the installation above, the Grafana and Prometheus are the same host.

Using Grafana and Prometheus

Using Grafana

Log in to Grafana using the access instructions in Configuring Grafana and Prometheus.
From the welcome dashboard, click **Dashboards** and then **Manage**.

![Grafana Welcome Dashboard](image)

**Figure 5: Grafana Welcome Dashboard**

Three dashboards will be visible:

- **cluster_tab**: stats and monitoring for the full cluster
- **nodes_tab**: stats and monitoring for each node
- **performance_tab**: performance stats and monitoring
Below is a screenshot of the **cluster_tab** dashboard. This is composed of multiple sections of graphs, statistics, and tables.

![cluster_tab Dashboard](image)

**Figure 6: cluster_tab Dashboard**
Hovering over each section reveals an arrow with additional options.

Figure 7: cluster_tab Arrows
Click on each artifact’s arrow button. Clicking the **View** option will expand the window to full screen.

![cluster_tab View](image_url)

**Figure 8: cluster_tab View**

**Using Prometheus**

Log in to Prometheus using the access instructions in Configuring Grafana and Prometheus.

Prometheus can be used to query any of the time series metrics received from a Lightbits cluster. The metrics come in at the cluster level and node level. This means that most metrics can be viewed for each node and also for the cluster as a whole. Prometheus is also helpful in figuring out the full names of metrics, which then can be used for creating dashboards in Grafana.

As an example, let’s look at the write bandwidth for the whole cluster. The values will be shown in their raw format. We can assume that this will be in “bytes/seconds”; however, if this is not the case, we could compare with other known values.
Step 1
Make sure **Use Local Time** and **Enable Autocomplete** are enabled. Local time will help in lining up the times to your timezone, regardless of the server’s timezone. Autocomplete will help explore all of the different metrics.

![Enable query history](use local time) ![Enable autocomplete](enable autocomplete)

Figure 9: Prometheus Autocomplete
Step 2
Start by writing “instance:cluster” into the expression field. As characters are entered, it will show available metrics in the drop-down. As more characters are entered, the drop-down menu converges on specific metrics.

![Expression input field](image_url)

Figure 10: Prometheus Metrics
With **Enable Autocomplete**, as text is typed into the expression field, Prometheus will then show metrics that have matching text as a drop-down.

![Prometheus Enable Autocomplete](image)

Figure 11: Prometheus Enable Autocomplete
As you enter more text, you will see less metrics that are more specific.

Figure 12: Prometheus Specific Metrics
Scroll to the bottom of the drop-down metric names:

![Prometheus Metrics Drop-Down](image)

Figure 13: Prometheus Metrics Drop-Down

Here we can see that we have “write_iops” and “write_throughput” as options. Since we want to know about write bandwidth, the suitable metric would be “instance:cluster:write_throughput”.

**Tip**

One good way to know what to type into the **Expression** field is to study the drop-down. Another is to simply view all of the available metrics.

To view all possible Prometheus metrics, curl, wget or open your browser to Prometheus Metrics.

The output will be large, but it will have all of the metrics. Here are example snippets of the output (searching for the word “throughput”):

```
```

```
```
Step 3
Finish typing “instance:cluster:write_throughput” into the **Expression** field, or select it from the drop-down menu, and enter **Execute**.

![Prometheus Expression Field](image)

Figure 14: Prometheus Expression Field

Here we can see the raw value of the cluster write_throughput expressed in bytes. We can see that the current write throughput is 98709485 bytes per second. This matches the fio job running in the background.

The following is the fio command that was launched from the same client.

```
root@rack02-server65 [client_0]:~ # fio --direct=1 --rw=write --numjobs=8 --iodepth=1 --ioengine=posixaio --bs=4k --group_reporting=1 --filesize=1G --directory=/test/ --time_based=1 --runtime=3600s --name=test
```
The fio output also shows 93 MiB/s:

```
Jobs: 8 (f=8): [W(8)] [83.9%] [r=0KiB/s, w=94.0MiB/s] [r=0, w=24.1k IOPS]
Jobs: 8 (f=8): [W(8)] [83.9%] [r=0KiB/s, w=97.5MiB/s] [r=0, w=24.0k IOPS]
Jobs: 8 (f=8): [W(8)] [84.0%] [r=0KiB/s, w=96.4MiB/s] [r=0, w=24.7k IOPS]
Jobs: 8 (f=8): [W(8)] [84.0%] [r=0KiB/s, w=98.5MiB/s] [r=0, w=25.2k IOPS]
Jobs: 8 (f=8): [W(8)] [84.0%] [r=0KiB/s, w=94.4MiB/s] [r=0, w=24.2k IOPS]
Jobs: 8 (f=8): [W(8)] [84.1%] [r=0KiB/s, w=93.6MiB/s] [r=0, w=23.9k IOPS]
[eta 09m:34s]
```

Figure 15: Prometheus fio Output
The following is the output of iostat -tmx 3, also showing 93 MiB/s:

![Figure 16: Prometheus iostat Output](image_url)

<table>
<thead>
<tr>
<th>Device</th>
<th>r/s</th>
<th>w/s</th>
<th>rMB/s</th>
<th>wMB/s</th>
<th>rrqm/s</th>
<th>wrqm/s</th>
<th>%rrqn</th>
<th>%wrqn</th>
<th>r+wa/s</th>
<th>avgrq-sz</th>
<th>svctm</th>
<th>%util</th>
</tr>
</thead>
<tbody>
<tr>
<td>sda</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>1.00</td>
<td>0.00</td>
<td>50.00</td>
<td>0.00</td>
<td>6.67</td>
<td>1.33</td>
</tr>
<tr>
<td>dm-0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dm-1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>2.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.01</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>nvme0n1</td>
<td>0.00</td>
<td>23943.00</td>
<td>0.00</td>
<td>93.53</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.27</td>
<td>0.00</td>
<td>4.00</td>
<td>0.67</td>
</tr>
<tr>
<td>nvme0n2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>sda</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dm-0</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dm-1</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>dm-2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>nvme0n1</td>
<td>0.00</td>
<td>24571.33</td>
<td>0.00</td>
<td>95.98</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.26</td>
<td>0.00</td>
<td>4.00</td>
<td>0.67</td>
</tr>
<tr>
<td>nvme0n2</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
</tbody>
</table>
**Step 4**

Click **Graph** to view the graph output. The duration of the graph and end time and shading of the graphs are adjustable with the buttons.

Here the graph shows the last 1 hour's worth of data. However, any time period can be viewed by adjusting the values in the boxes.

Note that there was a period of no throughput when the the fio job was cancelled temporarily.

---

**Figure 17:** Prometheus Cancelled fio Job
Step 5
In Prometheus, you can also:

- Create alerts (this can also be done in Grafana).
- Stack other metrics to compare. Click **Add Panel** and then follow the same steps above to add another expression. As an example, in the screenshot below, another panel was added to the bottom showing the write IOPs metric of the entire cluster, by using the expression "instance:cluster:write_iops".

![Prometheus Add Panel](image)

**Figure 18: Prometheus Add Panel**

Open TCP Ports and Verify
TCP ports in CentOS 7 and AlmaLinux/Red Hat 8 can be blocked either using the IPTABLES service or using the firewall service. The following is an example of how to use the IPTABLES service to open a TCP port and then test it using the netcat utility.

1. Check if a port is blocked.
   In this example, we can check if port 80 is accepting traffic by entering the **iptables** command with grep:

   ```bash
   $ iptables -nL | grep 80
   ```

   If the **iptables** command returns no data, the port needs to be opened.

2. To open TCP Port 80, enter the **iptables** command as follows:
Open TCP Port and Verify

3. Re-enter the `iptables -nL` command to see if the port is now open.

```
$ iptables -nL | grep 80
ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 tcp dpt:80
```

4. Install the netcat utility.

```
$ yum install nc
```

5. Run netcat as a server listing on port 80.

```
$ nc -l -p 80
```

6. From another server, install the netcat utility.

```
$ yum install nc
```

7. Run netcat to the server you are running iperf3 to verify that port 80 is accepting commands.

```
$ nc -z -v 192.168.16.7 80
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.16.7:80.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
nc
```

Open TCP Port Example

Open TCP Port 80:

```
# nc -z -v 192.168.40.41 80
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connected to 192.168.40.41:80.
Ncat: 0 bytes sent, 0 bytes received in 0.01 seconds.
```

Closed TCP Port Example

Closed TCP Port 80:

```
# nc -z -v 192.168.40.31 80
Ncat: Version 7.50 ( https://nmap.org/ncat )
Ncat: Connection refused.
```
Installation Behind HTTP-Proxy

In order to install behind http_proxy, you will need to update group_vars/all.yml with the following:

```yaml
proxy_env:
  http_proxy: http://<proxy-host>
  https_proxy: https://<proxy-host>
```

These settings will be passed to all tasks accessing the web for the installation of RPMs and other binaries, through the proxy settings provided.

**Note:** You will need to ensure that the formatting is correct (yaml formatting). This can be in a separate page.

Single-IP-Dual-NUMA Configuration

Starting from version 3.1.2, Lightbits supports dual NUMA configuration with single data IP (network interface) for the server.

The typical installation uses one data IP for instance ID 0, and another data IP for instance ID 1, but with this feature both instance IDs use the same IP.

Therefore, instead of using two network interfaces in a dual NUMA server, only one network interface will be utilized for the data network in both NUMAs.

To configure this single IP for both NUMA, duroslight and replicator use different ports for the different NUMAs.

**Example**

Configure this by appending the following lines into all.yml:

```yaml
duroslight_ports:
  0: "4420"
  1: "4421"
replicator_ports:
  0: "22226"
  1: "22227"
```

The above settings allow Duroslight and replication to run off of the same IP (single IP), but with different ports for each instance. This therefore allows two different instances of Lightbits to run off of the same IP, by using different ports.
Adding a JWT Token To a Configuration File

When running lbcli commands, the jwt token must be provided via the -J variable, like this: `lbcli -J $LIGHTOS_JWT get cluster`.

There is another way also, which is to configure the system jwt into a configuration file on the Lightbits server. In this way lbcli commands can be run from that server without the -J variable.

**Note:** Having the system JWT preconfigured introduces security concerns, because any lbcli command can now be run. Therefore it’s important to ensure that the server is secured.

1. After deploying the cluster, grab the system jwt. From the Ansible installation host, the file will be in located in `~/.lightos-system-jwt`. Show the content of the file with cat:

   ```
cat ~/.lightos-system-jwt
   ```

   The output should show the token, as below. Note that the token has been cut for brevity.

   ```
   export LIGHTOS_JWT=eyJhbGciOi<remaining jwt content>BaFEuMsT9gQNQA
   ```
Adding a JWT Token To a Configuration File

APPENDIXES

Copy the jwt token portion (everything after “LIGHTOS.JWT=”). Note that its long output will span multiple lines of terminal output; however, it should only take up one line in a file.

2. On a Lightbits server, edit /etc/lbcli/lbcli.yaml and append the jwt to the bottom.

```yaml
jwt: <jwt>
```

The full content of /etc/lbcli/lbcli.yaml will be similar to this:

```yaml
output-format: human-readable
dial-timeout: 5s
command-timeout: 60s
insecure-skip-tls-verify: true
debug: false
api-version: 2
insecure-transport: false
endpoint: https://127.0.0.1:443
jwt: eyJhbGciOi</remaining jwt content>BAFEuMsT9gQNQA
```

Running lbcli From a Non-Lightbits Server

Lightbits supports running lbcli remotely against a cluster from a non-Lightbits server. The limitation is that it has to be a x86_64 Linux server. Make sure that the remote server has IP connectivity to the management or data IP.

1. Copy the lbcli binary to /usr/bin/ on the remote server.
2. On the remote server, create the /etc/lbcli directory.
3. Create the /etc/lbcli/lbcli.yaml file, specifying the management or data IP and the jwt.

```yaml
output-format: human-readable
dial-timeout: 5s
command-timeout: 60s
insecure-skip-tls-verify: true
debug: false
api-version: 2
insecure-transport: false
endpoint: https://10.10.100:443
jwt: eyJhbGciOi</remaining jwt content>BAFEuMsT9gQNQA
```

**Note:** You can only provide one endpoint.
About - Legal

Lightbits Labs (Lightbits) is leading the digital data center transformation by making high-performance elastic block storage available to any cloud. Creators of the NVMe® over TCP (NVMe/TCP) protocol, Lightbits software-defined storage is easy to deploy at scale and delivers performance equivalent to local flash to accelerate cloud-native applications in bare metal, virtual, or containerized environments. Backed by leading enterprise investors including Cisco Investments, Dell Technologies Capital, Intel Capital, JP Morgan Chase, Lenovo, and Micron, Lightbits is on a mission to make high-performance elastic block storage simple, scalable and cost-efficient for any cloud.

www.lightbitslabs.com
info@lightbitslabs.com

US Offices
1830 The Alameda, San Jose, CA 95126
1412 Broadway 21st Floor, New York, NY 10018

Israel Office
17 Atir Yeda Street, Kfar Saba, Israel 4464313

The information in this document and any document referenced herein is provided for informational purposes only, is provided as is and with all faults and cannot be understood as substituting for customized service and information that might be developed by Lightbits Labs ltd for a particular user based upon that user’s particular environment. Reliance upon this document and any document referenced herein is at the user’s own risk.

The software is provided “As is”, without warranty of any kind, express or implied, including but not limited to the warranties of merchantability, fitness for a particular purpose and non-infringement. In no event shall the contributors or copyright holders be liable for any claim, damages or other liability, whether in an action of contract, tort or otherwise, arising from, out of or in connection with the software or the use or other dealings with the software.

Unauthorized copying or distributing of included software files, via any medium is strictly prohibited.

COPYRIGHT (C) 2023 LIGHTBITS LABS LTD. - ALL RIGHTS RESERVED