Lightbits Installation Guide
Lightbits Version: v3.0.5

Lightbits Labs

November 2022
Contents

Lightbits™ v3.x Installation and Configuration Guide

About the Installation Guide

Lightbits Cluster Overview

Lightbits Cluster Topology
Planning for the Lightbits Cluster Software Installation
Installation Files Backup
Lightbits Cluster Architecture
Nodes
Volume Assignments
NVMe/TCP MultiPath
Failure Domains

Lightbits Cluster Software Installation Process

Installation Preparation
System Requirements
Required Ports for Installation
Relevant Lightbits Support Documentation

Lightbits Cluster Software Installation

Before You Begin
Connecting to the Lightbits Software Repository
Connecting to the Cluster Client RPM Repository
Connecting to the Cluster Client DEB Repository
Verify Successful Installation

Lightbits Cluster Installation Process

Running the Ansible Installation Playbook to Install Lightbits Cluster Software
Confirming the Required Directory Structure
Setting the SSD Configuration
Defining Failure Domains
Defining Configuration Files for Each “Ansible Host” (Server) in the Cluster
Verifying Hosts Connection

Red Hat Linux Installation

Creating the Inventory Structure and Adding the Ansible Hosts File
Multi-Tenancy

Configuring Global Variables in Ansible
Verifying Hosts Connection

Installing New Kernel on CentOS

Install Additional Python Modules
Install sshpass or Use ssh-key Authentication
Install Additional Python Modules
Use Prebuilt Ansible Docker Image
Copying the Ansible Environment Tarball

Verifying Network Connectivity for the Servers in the Cluster
Connecting to the Lightbits Software Repository

Before You Begin

Relevant Lightbits Support Documentation
Required Ports for Installation
Installation Preparation

Linux Cluster Client Software Installation

Connecting to the Cluster Client DEB Repository
Connecting to the Cluster Client RPM Repository
Installing the New Kernel on CentOS
Configuring the Client to Boot from the New Kernel
Installing the Lightbits NVMe Command Line Interface
Installing the Lightbits NVMe Command Line Interface (Ubuntu)
Loading the NVMe/TCP Host Software ........................................... 38

Provisioning Storage and Connecting the Cluster Client to Lightbits ............ 38
  Creating a Volume on the Lightbits Storage Server ................................ 39
  Connecting the Cluster Client to Lightbits ......................................... 39
  Confirming the Cluster Client Connection to Lightbits ............................ 42

Troubleshooting .................................................................................. 43
  Ansible Role Errors ............................................................................... 43
  SSH Strict Key Errors When Using sshpass ........................................... 43
  Free Space in Linux OS for etcd Logical Volume Manager Use ...................... 44
  Recovering from Cluster Installation Failure ........................................ 45
  Log Artifacts Collection ..................................................................... 46

Appendixes ......................................................................................... 46
  Host Configuration File Variables ......................................................... 47
  Host Configuration File Examples ......................................................... 49
    Example 1: Data Network Interface Manually Configured ....................... 49
    Example 2: Data Network Interface Automatically Configured .................. 50
    Example 3: Override the Lightbits Configurations .................................. 50
    Example 4: Provide Custom Datapath Configuration .................................. 51
    Example 5: Use the Linux Volume Manager (LVM) Partition for etcd Data .... 51
    Example #6: Profile-Generator Overrides ............................................ 52
  Performing an Offline Installation ......................................................... 53
    Offline Ansible Controller Installation and Self-Signed Certificates .......... 53
  Configuring the Data Network .............................................................. 53
  Automatic Data Network Configuration (Recommended) ............................. 54
  Manual Data Network Configuration ..................................................... 54
  etcd Partitioning .............................................................................. 55
  Using SSH-Key Authentication .............................................................. 55
  Network Time Protocol Configuration .................................................. 55
  Automated Client Connectivity Verification .......................................... 57
  Configuring Grafana and Prometheus .................................................. 57
    Prerequisite ..................................................................................... 57
    Installing Grafana and Prometheus .................................................... 57
    Usage ............................................................................................. 58
    Outcome ....................................................................................... 58
    Lightbits Monitoring Integration with Existing Grafana and Prometheus ... 58
  Using Grafana and Prometheus ............................................................. 62
  Open TCP Ports and Verify ................................................................. 76

About - Legal ....................................................................................... 78
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:

- Learn about the Lightbits cluster software solution.
- 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.

![Lightbits Cluster Resources Diagram](image)

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.</td>
</tr>
<tr>
<td>2</td>
<td>dl.lightbitslabs.com</td>
<td>The Lightbits software is maintained in a password-protected software repository, referred to as “The Lightbits Repo”.</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. This is automatically configured by the Ansible installation script. However a custom NTP 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 <code>dl.lightbitslabs.com</code> 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 RPM files (from <code>dl.lightbitslabs.com</code>) 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:

```
ansible-playbook -i /tmp/inventory/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.

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 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 LightOS 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.

Volume Assignments

Lightbits provides the following levels of protection for the volumes in a cluster:

- **Double-replica:** Volumes are stored on two storage nodes.
- **Triple-replica:** Volumes are stored on three storage nodes.

In double and triple replica protection, one of the storage nodes behaves as a primary (P) node for this volume, and the other volume’s storage nodes behave as secondary (S) nodes.

Each storage node that stores data of multiple volumes can act as a primary node of one volume or as a secondary node of another volume. A primary node appears in the accessible path of the client, handles all user IO requests, and replicates data to the secondary nodes. If a primary node fails, the NVMe/TCP multipath feature changes the accessible path and reassigned the primary replica to another node.

When a user creates a volume, Lightbits transparently selects the nodes that hold the volume’s data and configures the primary and secondary roles. The node selection logic balances the volumes between nodes upon volume creation.

NVMe/TCP MultiPath

NVMe multipath I/O refers to two or more independent paths between a single host and a namespace. Each path uses its controller, although multiple controllers can share a subsystem port. Multipath I/O like namespace sharing requires that the NVM subsystem contains two or more controllers.

Multipath is part of NVMe specification and is used by the Lightbits cluster software as follows:
1. The Primary node exposes the path to the volume.
2. Clients send read and write requests to the Primary node.
3. The Primary node replicates to the Secondary nodes.
4. If the Primary node fails, the Secondary node exposes a path to the client so the client can continue working with the Secondary node.

Lightbits uses a proprietary protocol on top of TCP to replicate data between Primary and Secondary nodes, without requiring any changes to the client.

**Failure Domains**

Users define the Failure Domains (FD) based on data center topology and the level of protection that it strives 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.
Lightbits Cluster Software Installation Process

The process of installing Lightbits products includes the installation of the 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.

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 use 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.

**Note:** This is for a cluster with three Lightbits servers with a single client.

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Role</th>
<th>Access Network IP</th>
<th>Data NIC Interface Name</th>
<th>Data NIC IP</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>
</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>
</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>
</tr>
<tr>
<td>client00</td>
<td>client</td>
<td>192.168.16.45</td>
<td>ens1</td>
<td>10.10.10.103</td>
</tr>
</tbody>
</table>

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.

**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>

**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

- The Linux distribution (CentOS, Red Hat or Ubuntu) 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.

- Installing the Lightbits 3 on the cluster storage server includes:
  - Lightbits installation files
  - Lightbits was verified with kernel version 4.14. If support for another kernel version is required, contact Lightbits support at support@lightbitslabs.com.

Additionally:

- We recommend using CentOS 7.9 for storage servers, and selecting “Compute Node” + “Administration tools” for installation.
- In case of persistent memory (Intel Optane or NVDIMM) used, you should configure them 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 Storage Server where you are installing Lightbits.
• 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.

**Required Ports for Installation**

The Lightbits cluster software requires access to several ports to complete its installation process.

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

<table>
<thead>
<tr>
<th>Component</th>
<th>Access/Data NIC</th>
<th>Port (TCP)</th>
<th>Default location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Management CLI</td>
<td>Access</td>
<td>443</td>
<td>None</td>
</tr>
<tr>
<td>etcd peer port</td>
<td>Data</td>
<td>2380</td>
<td>roles/etcd/defaults/main.yml</td>
</tr>
<tr>
<td>Exporter port</td>
<td>Access</td>
<td>8090</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
<tr>
<td>Duroslight port</td>
<td>Data</td>
<td>4420,8009</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
<tr>
<td>Replicator port</td>
<td>Data</td>
<td>22226</td>
<td>roles/install-lightos/defaults/main.yml</td>
</tr>
</tbody>
</table>

Note: NVMe client connects to duroslight via this port.

Note: Other nodes connect for replication to the node via this port.

See the Open TCP ports and verify according to the examples below of how to open and test TCP ports.

You can use the following procedure with the open-source nmap program if you need to check a port’s accessibility:

1. Install the open-source nmap program with the following command: `bash $ yum install -y nmap`
2. Check a port’s accessibility with either of the following commands: `bash $ nc -v -z <ip> <start port>-<end port>` or `bash $ nc -v -u <ip> <start port>-<end port>`
3. You must have the netcat program running in listen mode on the server you are testing with the following command: `bash $ nc -l -p <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>Document</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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.

![Diagram](Image)

Figure 6: Lightbits Documentation Set

**Lightbits Cluster Software Installation**

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.

Also:

- 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 v2.10 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.

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.

**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
# Installation Steps

1. Connecting your installation workstation to Lightbits’ software repository
2. Verifying the network connectivity of the servers used in the cluster
3. Setting up an Ansible environment on your installation workstation
4. Installing a Lightbits cluster by running the Ansible installation playbook
5. Updating clients (if required)
6. Provisioning storage, connecting clients, and performing IO tests

**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 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. - For information on installing Red Hat, see Red Hat Linux Installation.

Verify that you have the TOKEN for the Lightbits RPM Repository 3X and log in to your installation workstation.

1. In your preferred text editor, open a new file in the workstation’s following CentOS directory:

   ```bash
   $ /etc/yum.repos.d/lightos.repo
   ```

2. Copy the following template into the file.

   ```bash
   # Lightbits repository
   [lightbits-lightos-3-<Minor Ver>-x-ga]
   name=lightbits-lightos-3-<Minor Ver>-x-ga
   baseurl=https://dl.lightbitslabs.com/<YOUR_TOKEN>/lightos-3-<Minor Ver>-x-ga/rpm/el7/$basearch
   repo_gpgcheck=0
   enabled=1
   gpgcheck=0
   autorefresh=1
   type=rpm-md
   ```

   For the `<YOUR_TOKEN>`, enter the Lightbits token that was included in your copy of the Lightbits Installation Customer Addendum.

3. Save the `lightos.repo` file.

4. Verify your system’s connectivity to the repository by entering the `yum repolist` command. This command displays the enabled software repositories; for example:

   ```bash
   $ yum repolist
   lightbits-lightos-2-<Minor Ver>-x-ga
   ```

   **Note:** For information on installing Red Hat, see Red Hat Linux Installation.
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 access 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 an access IP. Before proceeding with the installation, enter the following ping command at the installation workstation to confirm that each Lightbits server is accessible, and that the client via the “192.xxx.xxx.xxx” network IP that was set up on each server’s GbE NIC ports is connected.

```
$ ping -c 1 192.168.16.22
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 access network IPs and data network IPs.

<table>
<thead>
<tr>
<th>Server</th>
<th>Access Network IP</th>
<th>Data Network IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lightbits server00</td>
<td>ping -c 1 192.168.16.22</td>
<td>ping -c 1 10.10.10.100</td>
</tr>
<tr>
<td>Lightbits server01</td>
<td>ping -c 1 192.168.16.92</td>
<td>ping -c 1 10.10.10.101</td>
</tr>
<tr>
<td>Lightbits server02</td>
<td>ping -c 1 192.168.16.32</td>
<td>ping -c 1 10.10.10.102</td>
</tr>
<tr>
<td>Lightbits client00</td>
<td>ping -c 1 192.168.16.45</td>
<td>ping -c 1 10.10.10.103</td>
</tr>
</tbody>
</table>

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><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>
</tbody>
</table>
**Prepare Installation Workstation (Ansible Controller)**

**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 these steps:

**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

```bash
yum install -y libselinux-python3
```

**Install the Required Python Version from CentOS Repo**

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:

```bash
yum install python36...
```

Complete!

**Install Ansible Module Using PyPI**

Check if Ansible is installed, as well as its version; for example:

```bash
ansible --version
```

Command 'ansible' not found

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

```bash
pip3 install ansible
```

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

```bash
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
```

**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:

```bash
$ pip3 install netaddr python_jwt six
```
Use Prebuilt Ansible Docker Image

Usually it will be some machine outside the cluster that has SSH access to all cluster servers.

There are also some other dependencies that the installation process requires besides Ansible.

We provide a custom Ansible image to deploy the Lightbits cluster image that contains all dependencies.

The only prerequisites to use this image are:

- Having Docker installed.
- Access to Lightbits public registry or a private registry to fetch the `lb-ansible` image.

**NOTE:** `docker.lightbitslabs.com` requires a Docker login. Credentials are provided in the Customer Addendum.

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 and unpack the tar file using a command similar to the following:

```
$ tar -xvzf light-app-install-environment-v<Version>.tgz
```

The installation folder name should be under the “light-app” folder.

For example:

```
$ tar -xvzf light-app-install-environment-v2.x.1.tgz
```

Unpacking this tarball creates the following Ansible directory structure, which contains the Ansible environment where the “ansible-playbook” command runs.

```
|-- ansible
   |-- ansible.cfg
   |-- inventories
   |-- playbooks
   |-- roles
```

Creating the 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 create the inventory directory structure for the Ansible host file:

1. From the installation workstation, create the inventory directory structure. **bash** `mkdir -p /tmp/inventory`

2. Copy all of the files and subdirectories in the `cluster_example` directory. An example host file is included in the Ansible subdirectory that was created when you unpacked the tarball from Lightbits that was described in the previous section.

   Enter the following command to copy the example hosts file:

   ```
   $ cp -r <base-dir>/ansible/inventories/cluster_example /tmp/inventory
   ```
Creating the Inventory Structure and Adding the Ansible Hosts

The source directory path depends on the location of the unpacked file from Lightbits.

When this recursive copy command finishes, you will have the following directory structure:

```
/tmp/inventory/
|-- cluster_example
    |-- group_vars
    |    |-- all.yml
    |-- hosts
    |    |-- client00.yml
    |    |-- server00.yml
    |    |-- server01.yml
    |    |-- server02.yml
```

3. Open a text editor and edit the copied `hosts` example file, now found in the new `/tmp/inventory/cluster_example/hosts` path. Replace the `ansible_host`, `ansible_ssh_pass`, and `ansible_become_user` values with your environment’s relevant values, for each server that will be in your cluster. Refer to the following example for reference.

```
server00 ansible_host=192.168.16.22 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=light ansible_become_user=root ansible_become_pass=light
server01 ansible_host=192.168.16.92 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=light ansible_become_user=root ansible_become_pass=light
server02 ansible_host=192.168.16.32 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=light ansible_become_user=root ansible_become_pass=light
client00 ansible_host=192.168.16.45 ansible_connection=ssh ansible_ssh_user=root ansible_ssh_pass=light ansible_become_user=root ansible_become_pass=light

[duros_nodes]
server00
server01
server02

[duros_nodes:vars]
local_repo_base_url=https://dl.lightbitslabs.com/\<YOUR_TOKEN\>/lightos-3\-0-Minor Ver\-x-ga.rpm/el/7/\$basearch
auto_install=true
cluster_identifier=ae7bdef-897e-4c5b-4abef-20234abf21bf

# For cluster identifier use any human readable string identifying the cluster

[etcd]
server00
server01
server02

[initiators]
client00
```

- You can replace the `ansible_host` flag’s value with the interface DNS name or IP address. In this example, the access 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.
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:

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

Or via `group_vars/all`:

---

**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. The same value entered for the “baseurl” you configured in the Connecting to the Lightbits Software Repository section.</td>
</tr>
<tr>
<td>auto_install</td>
<td>No</td>
<td>A <strong>False</strong> value means the installation will wait for user instructions to reboot or not after installation. A <strong>True</strong> value means the installation will reboot in case of a kernel change without user instructions. Default value: 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. Save the new Ansible host file under the `/tmp/inventory/cluster_example` directory.

**Note:** For information on installing Red Hat, see Red Hat Linux Installation.
MULTI-TENANCY LIGHTBITS CLUSTER SOFTWARE INSTALLATION

yaml

certificates_directory=/path/to/certs

Certificate Types
Implementing multi-tenancy involves three sets of certificates:

- etcd Certificates for mTLS Peer Communication
- API Service Certificates for TLS
- System Scope Cluster Admin 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:

```yaml
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:

```yaml
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.

<table>
<thead>
<tr>
<th>cert-lb-api-service-key.pem</th>
</tr>
</thead>
<tbody>
<tr>
<td>cert-lb-api-service.pem</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>cert-lb-admin-key.pem</th>
</tr>
</thead>
<tbody>
<tr>
<td>cert-lb-admin.pem</td>
</tr>
</tbody>
</table>

**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.

**Red Hat Linux Installation**

The following summarizes the key points of distribution-specific information for Red Hat 8.

Note that the default “GA” Lightbits installation is based on Centos and the “RHEL” builds are based on Red Hat. The “GA” builds install a Lightbits modified kernel; however, the “RHEL” builds are installed on top of the Red Hat kernel.

Make sure that all of the Lightbits servers are on the same Red Hat distribution and kernel. Ensure also that the specific kernel is set as the default kernel via: `grubby --default-kernel`.

Install the Lightbits release that matches the kernel and distribution of the servers. See the official mapping below to select the proper Lightbits release.

The following is the official Lightbits release to Red Hat version mapping:

- Red Hat RHEL 8.4, with kernel 4.18.0-305.12.1.el8_4 for Lightbits RHL release 2.3.16-b887
- Red Hat RHEL 8.6, with kernel 4.18.0-372.9.1.el8_6 for Lightbits RHL release 2.3.17-b927
- Red Hat RHEL 8.6, with kernel 4.18.0-372.13.1.el8_6 for Lightbits RHL release 2.3.18-b951
• Red Hat RHEL 8.6, with kernel 4.18.0-372.16.1.el8_6 for Lightbits RHL release 2.3.19-b962
• Red Hat RHEL 8.6, with kernel 4.18.0-372.19.1.el8_6 for Lightbits RHL release 2.3.20-b988
• Red Hat RHEL 8.6, with kernel 4.18.0-372.26.1.el8_6 for Lightbits RHL release 2.3.22-b1031
• Red Hat RHEL 8.6, with kernel 4.18.0-372.19.1.el8_6 for Lightbits RHL release 3.0.1-b1007
• Red Hat RHEL 8.6, with kernel 4.18.0-372.9.1.el8_6 for Lightbits RHL release 3.0.3-b1061
• Red Hat RHEL 8.6, with kernel 4.18.0-372.26.1.el8_6 for Lightbits RHL release 3.0.3-b1062
• Red Hat RHEL 8.6, with kernel 4.18.0-372.32.1.el8_6 for Lightbits RHL release 3.0.4-b1085
• Red Hat RHEL 8.6, with kernel 4.18.0-372.9.1.el8_6 for Lightbits RHL release 3.0.5-b1105
• Red Hat RHEL 8.6, with kernel 4.18.0-372.32.1.el8_6 for Lightbits RHL release 3.0.5-b1107

Before the Ansible installation:
1. Make sure Red Hat subscription manager is registered and attached.
2. To avoid missing packages, install the following:
   ```
   # yum install -y python3
   # yum install -y network-scripts
   # yum install -y yum-utils
   # yum install -y net-tools
   ```
3. Edit the `hosts` file with the required target details. Consult with Lightbits Support for the Red Hat repository `baseurl` value.
4. To ensure that the kernel does not get overwritten by another kernel, add to `group_vars/all.yml`. Add `use_lightos_kernel: false`.
5. From Red Hat 8 and onward, chrony took over NTP as the default network time protocol. Edit `all.yml` to ensure that NTP is not installed and chrony is configured. Comment out the NTP sections and set the following NTP variables to false.

```
ntp_enabled: false
chrony_enabled: true
ntp_manage_config: false
use_lightos_kernel: false
# 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"
# ntp_version: "ntp-4.2.6p5-29.el7.centos.x86_64"
# ntp_packages:
#   - "autogen-libopts*.rpm"
#   - "ntpdate*.rpm"
#   - "ntpd*.rpm"
```
6. Install the Lightbits software as described in the Lightbits Cluster Software Installation Process section.

**Configuring Global Variables in Ansible**

To configure global variables in Ansible, you can use a special `all.yml` file that is normally located in the following location:

```
/tmp/inventory/cluster_example/group_vars/all.yml
```

If all the machines in the cluster have nvdimm or Intel Optane installed, the Persistent Memory flag must be set as follows:
Verifying Hosts Connection Lightbits Cluster Software Installation

**persistent_memory**: `true`

If there are machines in the cluster that do not have nvdimm, 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.

IP ACL allows support for restricted/non-restricted access to a cluster. This feature must be enabled during installation, by setting the `enable_iptables` flag; otherwise it cannot be used.

When the `enable_iptables` flag is set to `true`, access to the cluster nodes is allowed only from client IPs that are defined per volume using the `ip_acl` setting of each volume. By default, it is set to `false`. In order to use this mode, add the following to `all.yaml`:

```yaml
enable_iptables: true
```

Verifying Hosts Connection

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

```bash
$ cd light-app/
ansible -i /tmp/inventory/cluster_example/hosts all -m ping
```

A successful response from this Ansible ping is as follows:

```json
server02 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
Server00 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
server01 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

**Note**: If you see an error related to ssh-key authentication, see Troubleshooting.
Defining Configuration Files for Each “Ansible Host” (Server) in the Cluster

Return to the `/tmp/inventory/cluster_example` directory you created in Creating the Inventory Structure and Adding the Ansible Host File.

```
/tmp/inventory/
|-- cluster_example
 |  |-- group_vars
 |     |  |-- all.yml
 |  |-- hosts
 |  |-- host_vars
 |     |-- client00.yml
 |     |-- server00.yml
 |     |-- server01.yml
 |     |-- server02.yml
```

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

- server00.yml
- server01.yml
- 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.</td>
</tr>
<tr>
<td>datapath_config</td>
<td>The path is the customer profile directory information provided by Lightbits.</td>
</tr>
<tr>
<td>instances</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 (per logical node)</td>
<td>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) (per logical node)</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. 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 (per logical node)</td>
<td>The data/etcd IP used to connect to other servers.</td>
</tr>
<tr>
<td>storageDeviceLayout (per logical node)</td>
<td>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>
</tbody>
</table>

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

**Installation Planning Table Sample**
### Defining Configuration Files for Each “Ansible Host” (Server) in LIGHTBITS CLUSTER SOFTWARE INSTALLATION

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

<table>
<thead>
<tr>
<th>Server Name</th>
<th>Role</th>
<th>Access Network IP</th>
<th>Data NIC Interface Name</th>
<th>Data NIC IP</th>
</tr>
</thead>
<tbody>
<tr>
<td>server00</td>
<td>Lightbits Storage Server</td>
<td>192.168.16.22</td>
<td>ens1</td>
<td>10.10.10.100</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>
</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>
</tr>
<tr>
<td>client00</td>
<td>client</td>
<td>192.168.16.45</td>
<td>ens1</td>
<td>10.10.10.103</td>
</tr>
</tbody>
</table>

Examples for the three host variable files follow.

**server00.yml**

```yaml
name: server00
nodes:
- instanceID: 0
data_ip: 10.10.10.100
failure_domains:
- server00
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition == false
  - size >= gib(300)
  # - name =~ "nvme0n1"
```

**server01.yml**

```yaml
name: server01
nodes:
- instanceID: 0
data_ip: 10.10.10.101
failure_domains:
- server01
  instanceID: 0
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition == false
  - size >= gib(300)
  # - name =~ "nvme0n1"
```

**server02.yml**

```yaml
name: server02
```
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. 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 enter the server names to the host configuration file as follows:

```
name: server02
nodes:
  - instanceID: 0
data_ip: 10.10.10.102
failure_domains:
  - server02
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  #
  - model =~ ".*"
  - partition == false
  - size >= gib(300)
  #
  - name =~ "nvme0n1"
```

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.
- If you want the Ansible playbook to configure the data NIC IP, you must add a section Data_ifaces with the data interface name. For further information, see Configuring the Data Network.
- 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.

Defining Failure Domains

```
name: server00
data_ifaces:
  - bootproto: static
    conn_name: ens1
    ifname: ens1
    ip4: 10.16.205.14/27
nodes:
  - instanceID: 0
data_ip: 10.16.205.14
failure_domains:
  - server01
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  #
  - model =~ ".*"
  - partition == false
  - size >= gib(300)
  #
  - name =~ "nvme0n1"
```
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 eight SSDs installed.

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:

```yaml
name: server00
Data_ifaces:
  - bootproto: static
    conn_name: ens1
    iface: ens1
    ip4: 10.10.10.100/27
nodes:
  - instanceID: 0
    failure_domains:
      - server00
    ec_enabled: true
    data_ip: 10.10.10.100
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  # - model =~ ".*"
  - partition != false
  - size >= 300GiB
  # - name =~ "nvme0n1"
```

Confirming the Required Directory Structure

Before proceeding, confirm that a directory structure similar to the following example exists on your system for the inventory directories and host variable files.
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>

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, enter the following command to run the playbook:

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

**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.
Running Using the lb-ansible Docker Image

To save the Lightbits certificates at `/opt/lightos-certificates`, or to provide a certificate directory that already exists, place it under `/opt/lightos-certificates`.

```bash
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 \
docker.lightbitslabs.com/lightos-3-<Minor Ver>-x-ga/lb-ansible:4.2.0 \
sh -c '
  `ANSIBLE_LOG_PATH=ansible/ansible.log 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'
```

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.

**Note:** For information on installing Red Hat, see Red Hat Linux Installation.

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 \
docker.lightbitslabs.com/lightos-3-<Minor Ver>-x-ga/lb-ansible:4.2.0 \
sh -c '
  `ANSIBLE_LOG_PATH=ansible/ansible.log 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.

```
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 \
  docker.lightbitslabs.com/lightos-3-<Minor Ver>-x-ga/lb-ansible:4.2.0 \
sh -c 'ANSIBLE_LOG_PATH=/ansible/ansible.log 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'
```

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.

System-Scope JWT

Once the installation process is done, the system is bootstrapped with a system-scope project.

To act as cluster-admin you will need a JWT. The installation process will create a JWT and should be placed at:

```
yaml
system_jwt_path: "{{ '~/lightos-system-jwt' | expanduser }}"
```
This path can be overwritten at the command-line, or via a variable in the `group_vars/all.yml`

**Note:** This JWT gives admin access to the cluster. Make sure to securely back up the jwt files and certificate directory before deleting them. Do not share the files with unauthorized users.
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.

Connecting to the Cluster Client DEB Repository

To start the installation process, you must verify that you have the TOKEN for the Lightbits NVME-Client-DEBs.

**Note:** The Linux Repo File Customer TOKEN section in your Lightbits Installation-Customer Addendum has the TOKEN that is required to install the Lightbits NVME-Client-RPMs.

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>-x-ga/cfg/gpg/gpg.<KEY>.key' | apt-key add -
   ```

   **Note:** GPG KEY and Token should be provided via the customer addendum.

3. Create the lightos repo:

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

   **Note:** Token should be provided via the customer addendum.

In case you want to bypass the GPG verification - edit the `/etc/apt/sources.list.d/lightbits-lightos-3-x-ga.list` file and add `[trusted=yes]` after the `deb`:

```bash
cat /etc/apt/sources.list.d/lightbits-lightos-3-<Minor Ver>-x-ga.list
# Source: Lightbits
# Repository: Lightbits / lightos-2-<Minor Ver>-x-ga
# Description: A certifiably-awesome private package repository curated by
# Lightbits, hosted by Cloudsmith.

deb [trusted=yes] https://dl.lightbitslabs.com/<TOKEN>/lightos-3-<Minor Ver>-x-ga/deb/ubuntu xenial main

deb-src [trusted=yes] https://dl.lightbitslabs.com/<TOKEN>/lightos-3-<Minor Ver>-x-ga/deb/ubuntu xenial main
```
4. Run the `apt-get update` command:

```
apt-get update
```

---

**Connecting to the Cluster Client RPM Repository**

To start the installation process, you must verify that you have the TOKEN for the Lightbits NVME-Client-RPMs.

**Note:** The Linux Repo File Customer TOKEN section in your Lightbits Installation-Customer Addendum has the TOKEN that is required to install the Lightbits NVME-Client-RPMs.

1. In your preferred text editor, create the following new file: `bash /etc/yum.repos.d/lightos.repo`
2. Copy the following template into the file.

```
bash
# LightOS-Client-RPM-repository - Lightbits repository
[lightbits-lightos-3-<Minor Ver>-x-ga] name=lightbits-lightos-3-<Minor ver>-x-ga baseurl=https://dl.lightbitslabs.com/<YOUR_TOKEN>/lightos-3-<Minor Ver>-x-ga.rpm/el7/$basearch repo_gpgcheck=0 enabled=1 gpgcheck=0 autorefresh=1 type=rpm-md
```
3. Enter the Lightbits token that was included in your copy of the Lightbits Installation - Customer Addendum.
4. Save the `lightos.repo` file.
5. Verify your system’s connectivity to the repository by entering the `yum repolist` command. This command displays the enabled software repositories. For example: `bash $ yum repolist lightbits-lightos-3-<Minor Ver>-x-ga`

**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 New Kernel on CentOS**

**Notes:** - Before proceeding with the installation, you must have the GNU Wget software installed. You can download the software at https://www.gnu.org/software/wget/.
- You can use any kernel version v5.3.5 or above, which is written in the following instructions.

To install the Client on CentOS:
1. Go to https://elrepo.org/linux/kernel/el7/x86_64/RPMS/ and search for any kernel-ml-5.3.x, or above, kernel file.
2. Use Wget to download the following kernel:

   `https://elrepo.org/linux/kernel/el7/x86_64/RPMS/kernel-ml-5.X.Y-1.el7.elrepo.x86_64.rpm`

   where X.Y is the latest version
3. To install the kernel, run the `yum install` command as follows:

   `$ yum install -y kernel-ml-5.X.* #where X is the latest ver`
Configuring the Client to Boot from the New Kernel

You must configure the client to boot from the new kernel that you just installed.

1. Find the Lightbits kernel grub entry with the following command.

   $ grubby --info=ALL

2. Identify the Lightbits kernel index in the output list. In the following example, the Lightbits kernel has an index value of 0 because it is at the top of the list of available kernels.

   ```
   index=0
   kernel=/boot/kernel-ml-5.4.11-1.el7.elrepo.x86_64
   args="\n   ro crashkernel=auto rd.lvm.lv=CentOS_rack05-server67/root rd.lvm.lv=
   CentOS_rack05-server67/swap rhgb quiet LANG=en_US.UTF-8"
   root=/dev/mapper/CentOS_rack05--server67-root
   initrd=/boot/initramfs-kernel-ml-5.4.11-1.el7.elrepo.x86_64.img
   title=CentOS Linux (kernel-ml-5.4.11-1.el7.elrepo.x86_64) 7 (Core)
   index=1
   kernel=/boot/vmlinuz-3.10.0-957.el7.x86_64
   args="\n   ro crashkernel=auto rd.lvm.lv=CentOS_rack05-server67/root rd.lvm.lv=
   CentOS_rack05-server67/swap rhgb quiet LANG=en_US.UTF-8"
   root=/dev/mapper/CentOS_rack05--server67-root
   initrd=/boot/initramfs-3.10.0-957.el7.x86_64.img
   title=CentOS Linux (3.10.0-957.el7.x86_64) 7 (Core)
   index=2
   kernel=/boot/vmlinux-0-rescue-9758554168974f5dbe0d6dac5a6ac621
   args="\n   ro crashkernel=auto rd.lvm.lv=CentOS_rack05-server67/root rd.lvm.lv=
   CentOS_rack05-server67/swap rhgb quiet"
   root=/dev/mapper/CentOS_rack05--server67-root
   initrd=/boot/initramfs-0-rescue-9758554168974f5dbe0d6dac5a6ac621.img
   title=CentOS Linux (0-rescue-9758554168974f5dbe0d6dac5a6ac621) 7 (Core)
   index=3
   non linux entry
   ```

3. Use the following command to set the default kernel index value.

   In this example, the Lightbits kernel grub entry index value number is 0.

   $ grubby --set-default-index 0

4. Verify the correct kernel version is set.

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

5. Reboot the system to load the Lightbits kernel.

   $ shutdown -r now

6. 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-ml-5.3.6-1.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.

**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:

   ```
   $ rpm -qa | grep nvme-cli
   nvme-cli-1.9-2.3.4-1.x86_64
   ```

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

   ```
   $ rpm -e 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:

   ```
   $ yum install -y nvme-cli-1.9-2.3.4-1.x86_64
   ```

4. Enter the following command to verify that the NVMe CLI version is v1.9-2.2.1-1.

   ```
   $ rpm -qa | grep nvme-cli
   nvme-cli-1.9-2.3.4-1.x86_64
   ```

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

### 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:

```bash
$ 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:

```bash
$ 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:

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

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

```bash
$ 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

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

```bash
$ 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>
<th>Flags</th>
</tr>
</thead>
<tbody>
<tr>
<td>nvme_ttcp</td>
<td>24576</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>nvme_fabrics</td>
<td>20480</td>
<td>1</td>
<td>nvme_tcp</td>
</tr>
<tr>
<td>nvme_core</td>
<td>49152</td>
<td>4</td>
<td>nvme_fabrics, nvme_tcp</td>
</tr>
</tbody>
</table>

---

### Provisioning Storage and Connecting the Cluster Client to Lightbits

**Lightbits Cluster Installation Process**
Creating a Volume on the Lightbits Storage Server

<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><strong>Provisioning storage</strong>, 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

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
```

**Sample Output**

```
Name   UUID                        State   Size     Replicas   ACL
vol1   76c3eae8-7ade-4394-82e5-056d05a92b5e creating 2.0 GiB 2 values:"acl3"
```

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

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 below example `rack02-server70` is an application client and the IP address of the data NIC on one of the Lightbits storage server nodes is `10.10.10.100`.

**Sample Command**

```
$ ping -c 1 10.10.10.100
```
Sample Output

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.

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 <code>lbcli list nodes</code> command.</td>
<td>-a</td>
</tr>
<tr>
<td>ACL string</td>
<td>You used this ACL string when you created the volume on the Lightbits storage server node.</td>
<td>-q</td>
</tr>
<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 the Lightbits storage server to identify the subsystem NQN.

**Sample Command**

```
$ lbcli get cluster
```

**Sample Output**
Connecting the Cluster to Lightbits

Provisioning Storage and Connecting the Cluster Client to Lightbits

---

**UUID:** 95a251b6-0885-4f5b-a0eb-90e90a2009a3

**currentMaxReplicas:** 3

**Statistics:**
- **effectivePhysicalStorage:** "42638469827787"
- **estimatedFreeLogicalStorage:** "42638469827787"
- **freePhysicalStorage:** "42638469827787"
- **installedPhysicalStorage:** "75014889578496"
- **logicalStorage:** "4294967296"
- **logicalUsedStorage:** "0"
- **managedPhysicalStorage:** "56011126800384"
- **physicalUsedStorage:** "0"
- **physicalUsedStorageIncludingParity:** "0"

**subsystemNQN:** nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31

**supportedMaxReplicas:** 3

---

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

**Sample Command**

```bash
$ lbcli list nodes
```

**Sample Output**

<table>
<thead>
<tr>
<th>Name</th>
<th>UUID</th>
<th>State</th>
<th>NVMe endpoint</th>
<th>Failure</th>
</tr>
</thead>
<tbody>
<tr>
<td>domains</td>
<td>Local rebuild progress</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>server00-0</td>
<td>08fdb3bd-925a-5e73-adde-8daf881969d3</td>
<td>Active</td>
<td>10.10.10.100:4420</td>
<td>[server00]</td>
</tr>
<tr>
<td>server01-0</td>
<td>112a555f-8168-5f07-a4e0-bf8f5b59c740</td>
<td>Active</td>
<td>10.10.10.101:4420</td>
<td>[server01]</td>
</tr>
<tr>
<td>server02-0</td>
<td>bc759c13-856d-5521-9ba2-752259abf8f0</td>
<td>Active</td>
<td>10.10.10.102:4420</td>
<td>[server02]</td>
</tr>
</tbody>
</table>

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.nvmeexpress: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.nvmeexpress: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.nvmeexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d31 -s 4420 -q acl3
```
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, <strong>connecting clients</strong>, and performing IO tests</td>
</tr>
</tbody>
</table>

After you have entered the `nvme connect` command, you can confirm the client’s connection to Lightbits by entering the `nvme list-subsys` command.

**Sample Command**

```
$ nvme list-subsys
```

**Sample Output**

```
nvme-subsys0 - NQN=nqn.2014-08.org.nvmexpress:NVMf:uuid:b5fe744a-b919-465a-953a-a8a0df7b9d3
\  
  +-- nvme0 tcp traddr=10.10.10.100 trsvcid=4420 live
  +-- nvme1 tcp traddr=10.10.101 trsvcid=4420 live
  +-- nvme2 tcp traddr=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**

```
```

Notes: - 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.
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=qqn.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.

**Troubleshooting**

**Ansible Role Errors**

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

```yaml
/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 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:

```
$ ssh root@192.168.16.22
The authenticity of host '192.168.16.22 (192.168.16.22)' can't be established.
ECDSA key fingerprint is SHA256:zouTZEZF2oUXfIGpnvWutrOR4/fBnd5ARqXNJj0iqD0.
Are you sure you want to continue connecting (yes/no)? Yes
Warning: Permanently added '192.168.16.22' (ECDSA) to the list of known hosts.
root@192.168.16.22's password:
Last login: Wed Nov 13 19:06:13 2019 from cluster-manager
[root@node00 ~]#
```

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.

```
$ 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 `/tmp/inventory/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:

```
$ lsblk
NAME    MAJ:MIN    RM  SIZE RO TYPE MOUNTPOINT
sda      8:0       0 119.2G 0 disk
|--sda1   8:1       0    1G 0 part /boot
|--sda2   8:2       0 118.2G 0 part
   |--CentOS-root 253:0       0 50G 0 lvm /
   |--CentOS-swap 253:1       0  4G 0 lvm [SWAP]
   |--CentOS-home 253:2       0 64.2G 0 lvm /home
```

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
```

**Note:** For information on installing Red Hat, see Red Hat Linux Installation.

### Recovering from Cluster Installation Failure

At times during 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:
Reconfigure command:

```bash
ansible-playbook -i /tmp/inventory/hosts playbooks/configure-lightos-playbook.yml
```

**Log Artifacts Collection**

In addition to the td-agent facility, use a more basic or fundamental Ansible tool to collect logs artifacts like journal files, system health, and CLI outputs by entering the following Ansible command:

```bash
$ ansible-playbook -i <hosts file> playbooks/logs.yml
```

**Sample Output**

```
logs-2019-11-24-17:05:41
|-- server00
 | |-- tmp
 | | |-- lb-cluster-logs
 | | | | |-- lvm.txt
 | | | | |-- pip.txt
 | | | |-- lb-support
 | | | | |-- lb-support_rack02-server68_20191124-170544.tgz
 | |-- server01
 | | |-- tmp
 | | | |-- lb-cluster-logs
 | | | | |-- lvm.txt
 | | | | |-- pip.txt
 | | | |-- lb-support
 | | | | |-- lb-support_rack08-server93_20191124-170544.tgz
```

See the following default file for further customizations:

```bash
roles/logs-collector/defaults/main.yml
```

**Appendixes**

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

**This section includes:**

- Performing an Offline Installation
- Configuring the Data Network
- etcd Partitioning
- Using SSH-Key Authentication
- Network Time Protocol Configuration
- Fluent Bit Logging Configuration
- Log Artifacts Collection
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.

```yaml
data_ifaces:
  bootproto:
  ifname:
  ip4:
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 ifcfg-&lt;iface-name&gt; 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>Variable</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>------------------</td>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>bootproto</td>
<td>No</td>
<td>IP allocation type (dynamic or static). Only static is supported in Lightbits. 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 path IP address.</td>
</tr>
<tr>
<td>data_ip</td>
<td>Yes</td>
<td>The data/etcd IP used to connect to other nodes.</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>No</td>
<td>An optional setting specifying the initial count of physical drive the system will start with on the first startup.</td>
</tr>
<tr>
<td>maxDeviceCount</td>
<td>Yes</td>
<td>The pre-determined, maximum number of physical nvme drives that this node can contain.</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 than 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 =~ &quot;nvme5n.*&quot; or name == &quot;nvme1n1&quot;.</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>Variable</td>
<td>Required</td>
<td>Description</td>
</tr>
<tr>
<td>---------------------------</td>
<td>----------</td>
<td>---------------------------------------------------------------------------------------------------------------------------------------------</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.</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. Mandatory if use_lvm_for_etcd is used.</td>
</tr>
<tr>
<td>datapath_config</td>
<td>Yes</td>
<td>The path to the system-profile yml file.</td>
</tr>
<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:**
- By default, the etcd is installed on a dedicated logical volume and the default value for the use_lvm_for_etcd variable is set to false.
- 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.

**Host Configuration File Examples**

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

Host configuration with no data interfaces provided. The user configured the interfaces prior to running the playbook.
Host Configuration File Examples

Example 1:

```yaml
---
nodes:
- instanceID: 0
data_ip: 10.10.10.10
failure_domains:
- server00
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  - model =~ ".*"
  - partition == false
  - size >= gib(300)
# - name =~ "nvme0n1"
```

Example 2: Data Network Interface Automatically Configured

Host configuration with a single data interface. The playbook configured the interface.

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

nodes:
- instanceID: 0
data_ip: 10.10.10.10
failure_domains:
- server00
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  maxDeviceCount: 12
  allowCrossNumaDevices: false
deviceMatchers:
  - model =~ ".*"
  - partition == false
  - size >= gib(300)
# - name =~ "nvme0n1"
```

Example 3: Override the Lightbits Configurations

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

```yaml
---
```
yaml
---
nodes:
  - instanceID: 0
    data_ip: 10.10.10.10
    failure_domains:
    - server00
    ec_enabled: true
    storageDeviceLayout:
      initialDeviceCount: 4
      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:

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

---
nodes:
  - instanceID: 0
    data_ip: 10.10.10.10
    failure_domains:
    - server00
    ec_enabled: true
    storageDeviceLayout:
      initialDeviceCount: 4
      maxDeviceCount: 12
      allowCrossNumaDevices: false
    deviceMatchers:
      # - model =~ ".*"
      - partition == false
      - size >= gib(300)
      # - name =~ "nvme0n1"

datapath_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.

---

2022 Lightbits Labs Proprietary And Confidential Under NDA Only
yaml
---
nodes:
- instanceID: 0
data_ip: 10.10.10.10
failure_domains:
  - server00
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  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

Note: For information on installing Red Hat, see Red Hat Linux Installation.

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.

yaml
nodes:
- instanceID: 0
data_ip: 10.10.10.10
failure_domains:
  - server00
ec_enabled: true
storageDeviceLayout:
  initialDeviceCount: 4
  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:
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.

```yaml
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:

```yaml
[duros_nodes:vars]
source_type=offline
source_etcd_binary="/root/lightos_release/deps/etcd-v3.4.1-linux-amd64.tar.gz"
source_rpms_dir="/root/lightos_release/target_rpms"
source_dependencies_rpms_dir="/root/lightos_release/deps"
dest_dir="/tmp/rpms"
```

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.

**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
    Netmask: 255.255.255.240
```

In this example, we have set the playbook to permanently configure interface `ens1` with static IP `10.20.20.10` and netmask `255.255.255.240`.

**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:
   ```
   $ ssh root@rack03-server70
   ```

2. Detect the data interface with the `businfo` command.
   ```
   $ lshw -c net -businfo
   ```

<table>
<thead>
<tr>
<th>Bus info</th>
<th>Device</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>pci@0000:01:00.0 eno1 network</td>
<td>I350 Gigabit Network Connection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pci@0000:01:00.1 eno2 network</td>
<td>I350 Gigabit Network Connection</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pci@0000:02:00.0 ens1f0 network</td>
<td>MT27710 Family [ConnectX-4 Lx]</td>
<td></td>
<td></td>
</tr>
<tr>
<td>pci@0000:02:00.1 ens1f1 network</td>
<td>MT27710 Family [ConnectX-4 Lx]</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
   ```

3. Set a new data interface IP and net mask IP for the data NIC. In the following example, the card is ens1f0:
   ```
   $ cat >/etc/sysconfig/network-scripts/ifcfg-ens1f0
   DEVICE=ens1f0
   NM_CONTROLLED=no
   IPADDR=10.20.20.10
   NETMASK=255.255.255.224
   ONBOOT=yes
   BOOTPROTO=static
   ```
4. Toggle the NIC down and then up again by entering the `ifdown` command, waiting at least 30 seconds, and then entering the `ifup` command.

```bash
$ ifdown ens1f0
$ ifup ens1f0
```

5. Verify that the data interface's IP is updated.

```bash
$ ip -4 a...
4: ens1f0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP group default qdisc mq
          state UP group default qlen 1000
          inet 10.20.20.10/27 brd 10.20.20.31 scope global ens1f0
          valid_lft forever preferred_lft forever
```

### 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 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_password` in the following format:

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

For example:

```bash
ansible_ssh_private_key_file=/root/mykey.txt
```

### 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.

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

#### Method 1

The latest RPMs are retrieved from the CentOS 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. Create a new `group_vars` directory under your inventory directory. For more, see Creating the Inventory File (Hosts).
2. Under the new `group_vars` directory, create a new `all.yml` file.
3. Add the following section to the new `all.yml` file using the specific version that you want to install. For example:

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

Method 3

The NTP is installed using an offline method.

1. Create a new `group_vars` directory under your inventory directory. For more, see Creating the Inventory File (Hosts).
2. Under the new `group_vars` directory, create a new `all.yml` file.
3. Add the following section to the new `all.yml` file with the desired packages to install. This section’s order first lists the prerequisites and then the desired package. For example:

   ```yaml
   ntp_packages:
   - "autogen-libopts*.rpm"
   - "ntpd*.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. Under the inventory directory (`/tmp/inventory`) described in Creating the Inventory File (Hosts), create a new directory named `group_vars`.
2. Under the new `group_vars` directory, create a new `all.yml` file.
   You can skip this step if the `all.yml` file was created in a previous step.
3. Add to the `all.yml` file the following section, using the relevant NTP servers for your system.

   ```yaml
   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:

```
[main]
dns=none
```

Reload the NetworkManager service:

```
# systemctl reload NetworkManager
```

Note: After you reload the service, NetworkManager no longer updates the /etc/resolv.conf file. However, the last contents of the file are preserved.

Optionally, remove the Generated by NetworkManager comment from /etc/resolv.conf to avoid confusion.

Note: For information on installing Red Hat, see Red Hat Linux Installation.

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:

```
$ 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.
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:

```
/var/lib/monitoring-images/deploy.sh deploy-clustering
```

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.

Please 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.

```
[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:
You will need to manually merge contents inside of prometheus.yml with your existing prometheus.yml.

```
[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:
```

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**.

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

![Grafana Welcome Dashboard](image)
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 8: cluster_tab Dashboard
Hovering over each section reveals an arrow with additional options.

Figure 9: 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)

**Figure 10: 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]

Figure 11: 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 field with instance:cluster text](image)

Figure 12: 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 13: Prometheus Enable Autocomplete
As you enter more text, you will see less metrics that are more specific.

Figure 14: Prometheus Specific Metrics
Scroll to the bottom of the drop-down metric names:

![Prometheus Metrics Drop-Down](image)

Figure 15: 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”):

```
"instance:cluster:read_throughput","instance:cluster:total_read_bytes","instance:cluster:
:total_reads","instance:cluster:total_write_bytes","instance:cluster:total_writes","in
stance:cluster:write_iops","instance:cluster:write_throughput"

"instance:node:read_throughput","instance:node:receive_bytes_total","instance:node:
receive_drop_total","instance:node:receive_errs_total","instance:node:receive_packets_to
tal","instance:node:replication_write_iops_rx","instance:node:replication_write_iops_t
x","instance:node:replications_write_throughput_rx","in
stance:node:replications_write_throughput_tx"
```
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)

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=93.0MiB/s][r=0,w=23.9k IOPS]
[eta 09m:34s]
```

Figure 17: Prometheus fio Output
The following is the output of `iostat -tmx 3`, also showing 93 MiB/s:

![Prometheus iostat Output](image)

**Figure 18: Prometheus iostat Output**
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 fio job was cancelled temporarily.

![Graph Output]

Figure 19: 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 20: Prometheus Add Panel**

**Open TCP Ports and Verify**

TCP ports in CentOS 7 for example 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:

   ```
   $ 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:
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.
```
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 © 2022 LIGHTBITS LABS LTD. - ALL RIGHTS RESERVED

COPYRIGHT (C) 2022 LIGHTBITS LABS LTD. - ALL RIGHTS RESERVED