Cluster Federation Overview

Nov 27, 2025

Table of Contents

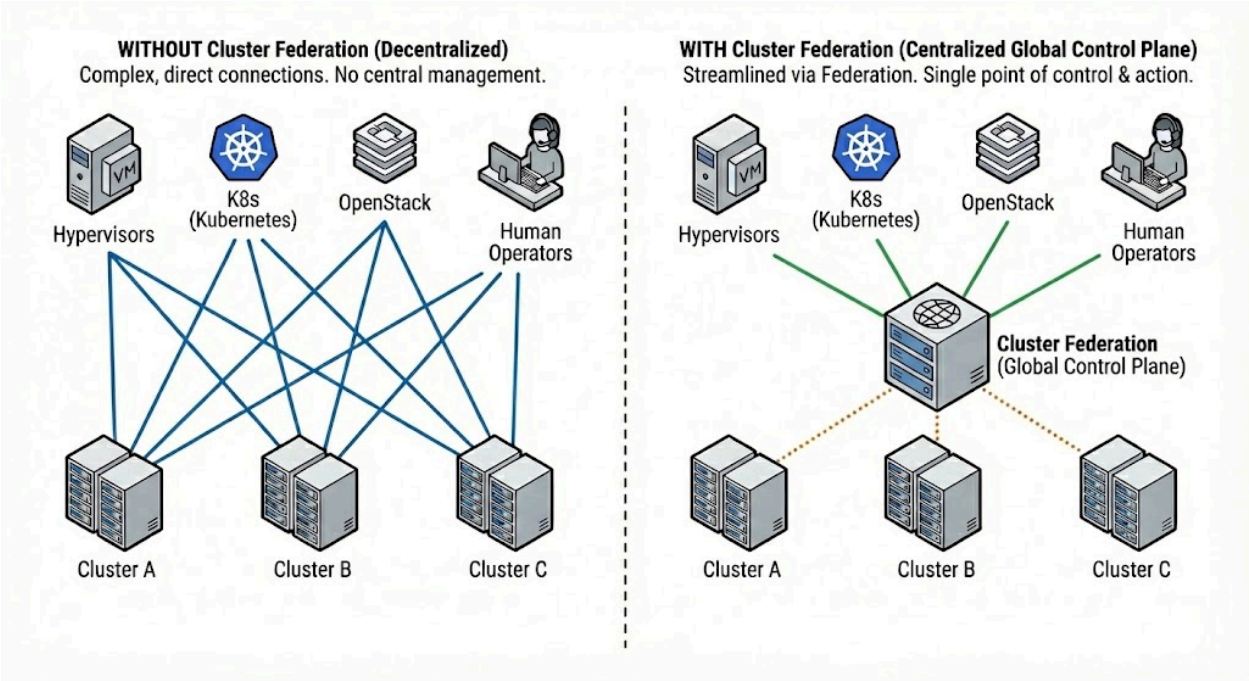
Overview

Cluster Federation (CF) is a multi-cluster management control plane designed to overcome the inherent scale limitations of single Lightbits storage clusters. CF acts as a provisioning broker across multiple attached Lightbits clusters, enabling users to scale their storage infrastructure quickly and simply. The primary value of this centralized system is to provide a unified interface for centralized fleet-wide management, which is intended to ensure seamless integration with existing deployments and the goal of requiring no or minimal changes to current API usage.

For its initial release (V1), Cluster Federation supports complete **CRUD (Create, Read, Update, Delete)** operations for managing volumes and snapshots across the federation.

The current tech preview version is currently delivered as a set of container images and is deployed using a dedicated script that simplifies the installation.

This initial setup provides a single, fault-tolerant service that ensures persistent configurations, enabling the service to retain data following restarts and upgrades. High Availability (HA) via Kubernetes (K8s) and Helm for true resilience and streamlined deployment (using StatefulSets, Services, and Valkey Sentinel) is planned to be added in the future, targeted for subsequent phases (Version 1.1 and beyond). CF is also the strategic foundation for future intelligent cluster management, which will include extended policy-driven provisioning based on capabilities, geography, and operational status in later versions.



Key Concepts

Intelligent Placement Policy

CF employs intelligent policy-driven provisioning to automate storage allocation based on predefined rules. For V1.0, CF uses a simple placement policy: volumes are provisioned on the cluster according to the provisioned ratio of existing clusters. CF will always attempt to place a new volume on the cluster with lowest provisioned ration (Provisioned Capacity/Available Capacity).

Data Consistency

CF maintains an in-memory cache (IMT) of volumes, snapshots, and cluster information. This data is regularly synced (polled) from the managed clusters. In addition, CF persistently stores information about the managed clusters. This information is saved to a file so that when CF is restarted, it will connect to these clusters, poll the data, and resume operation (fault-tolerant behavior).

Functionality and Limitations of CF V1.0

The CF V1 release is limited in scope and functionality to deliver essential multi-cluster scaling capabilities quickly. These limitations and architectural assumptions are summarized below.

- Deployment package via Ansible playbook for a standalone service.
- V1 supports **a single project** (defined by the customer). Multi-tenancy will be introduced in a later version.
- Full CRUD operations exclusively for **volumes and snapshots**.
- **QoS/Labels:** In V1, volume creation and update operations will not support flags for QoS Policy UUID/Name or volume labels.
- **Lists Filtering:** List commands for volumes and snapshots will only support filtering options of snapshot/volume name or uuid.
- Default placement policy: volumes are provisioned on the cluster with the lowest provisioned usage ratio (provisioned capacity/available capacity).
- **Volume Naming**: CF assumes volume and snapshot names to be globally unique across all clusters (within the context of the supported single project in V1).
- Monitoring is limited to basic health probing (as is in DMS). Future versions will include more observability capabilities.
- CF V1 supports managing three-node clusters and above.
- Up to 2 clusters are supported for v1.0.
- Placement logic is eventually consistent and non blocking, such that placement decision is based on the last info polled from the cluster.

Installing CF

CF is installed via a deployment script that automates the necessary configuration steps on the local server (see **Prerequisites**).

System Requirements

- 64-bit x86_64 Linux Compatible
- RockyLinux 9.1 and above.
- Minimum: 4 vCPUs
- Minimum 16GB RAM
- 20GB available disk space

Network Requirements

- HTTPS access to each cluster's management/API endpoints (TCP port 443).

Prerequisites

Before running the script, ensure that you have the following:

- **Cloudsmith API Key:** Provided by Lightbits for accessing the Cloudsmith repository.
- **Deployment Script:** Download the script from https://github.com/LightBitsLabs/lightbits-community/tree/main/intelligent-cluster-management

Installation Steps

1. **Run the script:**

   Execute the deployment script as an elevated user (root/sudo):

   ```
   sudo bash ./deploy_cf_unified.sh
   ```

2. **Follow the on-screen prompts:**

   The script will guide you through the configuration process, requesting the following inputs:

   - **CloudSmith API Key:** Enter your provided key.
   - **CF Version:** Select the desired version from the list of available releases (currently v1.0.0 is only available).
   - **Project Name:** Configure the project name. *Note: For the Tech Preview version, only a single project is supported. You can use the name 'default' or specify a custom project name.*

3. **Complete the installation:**

   Once the deployment is complete, reload your shell to begin using the 'cfcli' tool:

   ```
   exec $SHELL
   ```

Continue the initial setup steps as explained in the next section.

Working with CF

The following sections describe how to use CF with the included cfcli utility.

For REST API usage, refer to Working with CF Using REST API.

Working with CF Using the cfcli Utility

The `cfcli` is the command-line interface tool used to manage resources within the Lightbits CF service.

The following options are global options that can be used with the main `cfcli` command:

OPTION	SHORT	TYPE	DEFAULT	DESCRIPTION
--auth.jwt	-j	string		JWT token for authentication.
--base-url		string	https://localhost	Endpoint of the CF service (used with login).
--config		string	$HOME/.local/cluster-federation/cf-cli.yaml	Not available. Path to the configuration file.
--output-format	-o	string	"human"	Output format (json, yaml, human).

`--command-timeout`		string		Timeout for the command (default is `60s`).
`--dial-timeout`		string		Timeout for the dial operation (e.g., `10s`).

Login to the CF Service

Running commands using the `cfcli` tool requires a login. Use the following command to log in (connect) to the CF service running on `baseUrl`:

Bash

```
None
cfcli login --username=admin --password=light
--base-url=https://<CF-Server>:443
```

Note: The username and password are currently configured as username=admin and password=light. These are saved in: /etc/cf/.htpasswd.

The result of this command will sign you in to the CF service and update the `~/.local/cluster-federation/cf-cli.yaml` file with a valid `idToken` to be consumed by the CF API.

Example file:

YAML

```
None
baseUrl: https://<CF-Server>:443
auth:
  tokenType: Bearer
```

```
  expiresIn: 2592000
  idToken: <returned-jwt-token>
```

Attaching/Detaching Lightbits Clusters

Once the CF service is running and a CF client is configured to work with it, you can attach the Lightbits clusters to the CF service.

The CF service can manage multiple Lightbits clusters. A cluster attach operation will update the CF service with the Lightbits cluster's information.

Attaching a Lightbits Cluster

Attaching a Lightbits cluster is a three-step procedure:

1. Retrieve CF service credentials.
2. Update these credentials on the Lightbits cluster.
3. Invoke the attach cluster operation on the CF service.

Step 1: Retrieve the CF service credentials.

The information from this step will be needed in the next step. Run the following command on the CF host:

Bash

```
None
cfcli get credential -o json
```

Output example:

JSON

```
None
A file containing the public-key will be generated and stored at:
~/.local/cluster-federation/<pubKeyId>.pem

{
```

```
  "pubKeyId":
"cluster-federation-4c26b680-7bf8-11ef-98e2-e338f5c94dea",
  "pubKey": "<base64 encoded Public Key>"
}
```

The information in this file will be needed in the next step.

Step 2: Update the credentials on a Lightbits cluster.

To authorize the CF service to interact with the Lightbits cluster you want to use, you will need to upload the publicKey/credentials to the Lightbits cluster.

1. Upload the CF decoded public key to the Lightbits server. Using the pubkey file path from the previous step (~/.local/cluster-federation/<pubKeyId>.pem), upload this file to the Lightbits server:

   Bash

```
None
scp ~/.local/cluster-federation/<pubKeyId>.pem
<user>@<LightbitsServer>:/tmp/
```

2. Run the lbcli create credential command on the Lightbits server:
   The pubKeyId (specified in the file name from the previous step) will be used here as the id of the credential. SSH to the Lightbits server and execute the following (requires system-admin privileges):

   Bash

```
None
lbcli create credential --project-name=system --type=rsa256pubkey
--id=<pubKeyId> /tmp/<pubKeyId>.pem
```

Once the credentials have been added, the CF service that generated the credentials is authorized to operate with this Lightbits cluster.

Invoke the Attach Cluster Operation

It is possible to attach a Lightbits cluster using any of the Cluster's API endpoints.

Note: It is mandatory to have a healthy connection between the CF host and the specified Lightbits API endpoint. Otherwise, the attach operation will fail.

To attach a Lightbits cluster, run the following command on the CF service:

Bash

```
None
cfcli attach cluster --api-endpoint=<LightbitsServer>:443
```

The output for this action will be a workflowId, describing the attach-cluster workflow state, enabling monitoring of the attach operation. The result of this action on the CF service side would be adding a cluster entry to the /etc/cf/clusters.yaml file. For example:

YAML

```
None
clusters:
- uuid: 4cb2e0bf-20fa-5720-8cc2-1a11fe3e9850
  mgmtEps:
  - rack01-server01:443
  - rack01-server02:443
  - rack01-server03:443
```

Listing Lightbits Clusters

Run the following command to list the attached Lightbits clusters:

Bash

```
None
cfcli list clusters
```

Example of the list cluster response:

Bash

None


```
{
  "clusters": [
    {
      "id": "733840bf-30dc-54ae-99d5-da43d4e2f802",
      "name": "733840bf-30dc-54ae-99d5-da43d4e2f802",
      "version": "3.12.4~b15783733640",
      "subsystemNqn":
"nqn.2016-01.com.lightbitslabs:uuid:0b57d0d5-a789-4ecc-830c-814fea25
4767",
      "apiEndpoints": [
        "10.17.179.25:443",
        "192.168.20.28:443"
      ],
      "discoveryEndpoints": [
        "10.17.179.25:8009"
      ],
      "clusterConnectionStatus": {
        "accessConnectionStatus": "Connected"
      },
      "provisionedToEffectiveStorageRatio": "2.8738"
    },
    {
      "id": "07d2d333-f01f-5918-8784-0bc837f56e10",
      "name": "07d2d333-f01f-5918-8784-0bc837f56e10",
      "version": "3.12.4~b15783733640",
      "subsystemNqn":
"nqn.2016-01.com.lightbitslabs:uuid:cb87cbe0-6d6c-4c54-b8f5-90563aa1
f92f",
      "apiEndpoints": [
        "10.17.179.3:443",
        "192.168.20.67:443"
      ],
      "discoveryEndpoints": [
        "10.17.179.3:8009"
      ],
      "clusterConnectionStatus": {
        "accessConnectionStatus": "Connected"
      },
      "provisionedToEffectiveStorageRatio": "13.3823"
    }
```

```
  ]
}
```

Detaching a Cluster

Detaching a cluster can be done by specifying its ID. Run the following command to detach a Lightbits cluster:

Bash

```
None
cfcli detach cluster --cluster-id
4cb2e0bf-20fa-5720-8cc2-1a11fe3e9850
```

Volume Commands

Note1: Cluster ID can be retrieved by running cfcli list clusters command.

Note2: volume commands are intended to be as similar as possible to lbcli commands, with the exception of CF specific functionalities such as specifying a specific cluster.

cfcli create volume

Creates a new volume with specified properties: size, replica count, and placement affinity.

Command Syntax	Example
cfcli create volume [flags]	cfcli create volume --project-name my-project --name vol --size 10gib --replica-count 3 --acl client_1

OPTION	TYPE	DESCRIPTION
--name	string	Volume name.
--project-name	string	The project the volume belongs to. **Note**: This must specify the single project that is configured in CF.
--size	string	Volume maximal capacity (e.g., 10gib, 1024mib).
--acl	stringSlice	
--replica-count	uint32	Number of replicas (values: {1, 2, 3}).
--placement-affinity	string	Specifies placement affinities for the volume. Must be in the format: "key:value|key:value|key:value|... " (up to 25 pairs, with each 'value' up to 100 characters). The supported keys are: 'FD', 'PrimaryFD' and 'clusterid' - where 'clusterid' can be used in cluster-federation to specify the ID of the cluster where the volume must be created.

--source-snapshot-name / --source-snapshot-uuid	string	Create a volume from an existing snapshot.

cfcli update volume

Updates properties of an existing volume (e.g., size, ACL). Specify the volume using its name or UUID.

Command Syntax	Example
cfcli update volume [flags]	cfcli update volume --project-name proj --name vol --size 20Gib

OPTION	TYPE	DESCRIPTION
--size	string	Volume maximal capacity. **Note**: Use the --force flag to intentionally shrink volume size.
--acl / --ip-acl	stringSlice	ACL/IP Address ACL assigned to the volume.
--compression	string	Compression mode ('true'/'false').

cfcli get volume

Retrieves information about an existing volume. Specify the volume using its name or UUID.

Command Syntax	Example
cfcli get volume [flags]	cfcli get volume --project-name proj --name vol

OPTION	TYPE	DESCRIPTION
--project-name	bool	The project the volume belongs to.
--uuid	string	Volume uuid: cannot be used together with the name flag.
--name	string	Volume name: cannot be used together with the uuid flag.

cfcli delete volume

Deletes an existing volume. Specify the volume using its name or UUID.

Command Syntax	Example
cfcli delete volume [flags]	cfcli delete volume --project-name my-project --uuid 68268540-d175-5a95-abbe-429e1940ef8b

OPTION	TYPE	DESCRIPTION

--project-name	bool	The project the volume belongs to.
--uuid	string	Volume uuid of volume to delete (specify either UUID or name of volume).
--name	string	Volume name of volume to delete (specify either UUID or name of volume)

cfcli list volumes

Lists all volumes, optionally filtered by project.

Command Syntax
cfcli list volumes [flags]

OPTION	TYPE	DESCRIPTION
--show-all	bool	Show all volumes, currently Lightbits recommends to set this to true.

Snapshots Commands

Note: Cluster ID can be retrieved by running the cfcli list clusters command.

Commands for creating, deleting, getting, and listing snapshots.

cfcli create snapshot

Creates a new snapshot from a specified source volume.

Command Syntax	Example
cfcli create snapshot [flags]	cfcli create snapshot --project-name proj --name snap --source-volume-uuid {uuid} --retention-time 7200s

OPTION	TYPE	DESCRIPTION
--name	string	Snapshot name.
--project-name	string	The project the snapshot belongs to.
--source-volume-name / --source-volume-uuid	string	The volume from which to take the snapshot.
--retention-time	string	The time to keep the snapshot (e.g., 7200s, 2h).

cfcli get snapshot

Retrieves information about an existing snapshot. Specify the snapshot using its name or UUID.

Command Syntax	Example

cfcli get snapshot [flags]	cfcli get snapshot --project-name proj --uuid u-u-i-d

OPTION	TYPE	DESCRIPTION
--project-name	bool	The project the snapshot belongs to.
--uuid	string	Snapshot uuid: cannot be used together with the name flag.
--name	string	Snapshot name: cannot be used together with the uuid flag.

cfcli delete snapshot

Deletes an existing snapshot. Specify the snapshot using its name or UUID.

Command Syntax	Example
cfcli delete snapshot [flags]	cfcli delete snapshot --project-name proj --name snap

OPTION	TYPE	DESCRIPTION
--project-name	bool	The project the snapshot belongs to.

--uuid	string	Snapshot uuid: cannot be used together with the name flag.
--name	string	Snapshot name: cannot be used together with the uuid flag.

cfcli list snapshots

Lists all snapshots, optionally filtered by project.

Command Syntax
cfcli list snapshots [flags]

OPTION	TYPE	DESCRIPTION
--project-name	bool	The project the volume belongs to.

Inspect Workflow State

CF attach and detach operations are called a workflow. To track the state, status, and progress of both running and completed workflows, Lightbits' CF exposes list and get workflow APIs.

List All Workflows

Bash

```
None
cfcli list workflows
```

By default, the page-size is set to 100, so a list workflows command will hence fetch the latest 100 workflows.

To fetch a more limited list or a larger number of items in the list (the max is limited to 250), you should add the page-size flag.

Bash

None

```
cfcli list workflows --page-size=10
cfcli list workflows --page-size=100
```

Get a workflow by id

Bash

None

```
cfcli get workflow --id <wid>
```

Working with CF Using REST API

This section outlines the REST API endpoints for managing the CF service and its resources (clusters, volumes, and workflows). All endpoints are under the `/api/v2/` base path.
Note: volume/snapshot crud operations share the same API format as Lightbits core.

Authentication

GET /api/v2/cf-service-credentials

Summary: Gets cluster-federation service credentials.

Description: Gets the cluster-federation service credentials.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.GetCFServiceCredentialsResponse

401	Authentication required / Authentication failed	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

POST /api/v2/cflogin

Summary: Login to the cluster federation.

Description: Login to the cluster federation service.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.CFLoginResponse
401	Authentication required / Authentication failed	integer
500	Internal error in the cluster-federation service.	integer

default	An unexpected error response.	grpc.gateway.runtime.Error

Security: BasicAuth.

Cluster Management

GET /api/v2/cluster

Summary: Retrieves cluster information.

Description: Cluster information - e.g., cluster UUID and SubsystemNQN - is exposed via this API.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.ClusterInfo
401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer
500	Internal Lightbits error.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

GET /api/v2/clusters

Summary: List clusters.

Description: Lists all Lightbits storage clusters attached to the cluster-federation service.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.ListClustersResponse
401	Authentication required / Authentication failed	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

POST /api/v2/clusters

Summary: Attaches a new cluster.

Description: Attaches a new Lightbits storage cluster to the CF system.

Parameter Name	Located in	Required	Schema	Description
body	body	Yes	lightbits.api.duros.v2.AttachClusterRequest	Request to attach a cluster to the CF service.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.AttachClusterResponse
400	Invalid argument.	integer
401	Authentication required / Authentication failed	integer
500	Internal error in the cluster-federation service.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

DELETE /api/v2/clusters/{cid}

Summary: Detaches an existing cluster.

Description: Detaches an existing Lightbits storage cluster from the CF system.

Parameter Name	Located in	Required	Schema	Description
cid	path	Yes	string	UUID of the cluster to detach from the CF service (must be a valid UUID).

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.DetachClusterResponse
400	Invalid argument.	integer
401	Authentication required / Authentication failed	integer
500	Internal error in the cluster-federation service.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

Volume Management

GET /api/v2/volumes

Summary: List volumes.

Description: List volumes. Can be filtered by failure domain, project, or source snapshot. Supports partial return via offsetUUID and limit.

Parameter Name	Located in	Required	Schema	Description
UUID	query	No	string	List only volumes with a matching UUID.
name	query	No	string	List only volumes with a specified name.
failureDomain	query	No	string	List volumes according to failureDomains that volume replicas are placed on.
offsetUUID	query	No	string	When provided, the returned list starts with next to offsetUUID volume.
limit	query	No	string (int64)	Limit the number of volumes in the response (default/maximum value is 1000 volumes).

projectName	query	No	string	Optional. Return only volumes of the given project.
snapshotUUID	query	No	string	Source snapshot UUID. Return only volumes created from this source snapshot.
showAll	query	No	boolean	Optional. Show also volumes in Deleting state (the default is false).

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.ListVolumesResponse
400	Invalid argument.	integer
401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer

500	Internal Lightbits error.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

POST /api/v2/volumes

Summary: Create volume.

Description: A volume has a user-defined name, capacity, ACL, and optional compression and IP-ACL.

Parameter Name	Located in	Required	Schema	Description
body	body	Yes	lightbits.api.duros.v2.CreateVolumeRequest	Volume creation request body.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.Volume
400	Invalid argument (e.g., missing argument, illegal name, invalid size/replica count/ACL).	integer

401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer
409	A volume with the provided name already exists.	integer
500	Internal Lightbits error.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

GET /api/v2/volumes/{UUID}

Summary: Get volume information.

Description: Get volume information by the provided UUID.

Parameter Name	Located in	Required	Schema	Description
UUID	path	Yes	string	The volume's UUID (required in path).
name	query	No	string	The volume's name (optional; either UUID or Name must be provided to identify the volume).

projectName	query	No	string	Name of the project the volume belongs to.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.Volume
400	Invalid argument.	integer
401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer
404	The provided volume UUID is not found.	integer
500	Internal Lightbits error.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

DELETE /api/v2/volumes/{UUID}

Summary: Delete volume.

Description: Deletes a volume according to the provided UUID or Name. Deletion is a long operation; status remains Deleting until completed.

Parameter Name	Located in	Required	Schema	Description
UUID	path	Yes	string	The volume's UUID for the delete request (optional; either UUID or Name must be provided).
name	query	No	string	The volume's name for the delete request (optional; either UUID or Name must be provided).
projectName	query	No	string	The project name of the volume to delete.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.DeleteVolumeResponse
400	Invalid UUID, Name, or volume state does not allow deletion (Deleting/Failed).	integer

401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer
404	Volume with provided UUID or Name is not found.	integer
409	There is a user operation in progress on the volume.	integer
412	Etag mismatch.	integer
500	Internal Lightbits error.	integer
503	Volume is in a temporary state that does not currently allow deletion (Creating/Updating).	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

PUT /api/v2/volumes/{UUID}

Summary: Update volume.

Description: Update volume enables the change of attributes including size, compression, ACL, IP-ACL, QoS, and labels.

Parameter Name	Located in	Required	Schema	Description
UUID	path	Yes	string	Optional identifier of the volume to update (command must use either name or UUID fields).
body	body	Yes	lightbits.api.duros.v2.UpdateVolumeRequest	Volume update request body.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.UpdateVolumeResponse
400	Invalid argument or a volume is in a state that cannot be updated (Deleting/Failed).	integer

401	Unauthorized: authentication failed.	integer
403	Permission denied.	integer
404	Returned when the volume with the given UUID is not found.	integer
409	There is a user operation in progress on the volume.	integer
412	Etag mismatch.	integer
500	Internal Lightbits error.	integer
501	Unimplemented capability (updating capacity).	integer
503	Volume is in a temporary state and cannot be updated now (Creating/Updating).	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

Workflow Status

GET /api/v2/workflows

Summary: List workflows.

Description: List the running and completed workflows.

Parameter Name	Located in	Required	Schema	Description
pageSize	query	No	integer	The number of items to return per page.
nextPageToken	query	No	byte	The next page token to continue listing items.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.ListWorkflowsResponse
400	Invalid argument.	integer
401	Authentication required /	integer

	Authentication failed	
500	Internal error in the cluster-federation service.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

GET /api/v2/workflows/{workflowId}

Summary: Gets a specific workflow.

Description: Gets a specific workflow by its ID.

Parameter Name	Located in	Required	Schema	Description
workflowId	path	Yes	string	Workflow ID to get.

Response Code	Description	Schema
200	A successful response.	lightbits.api.duros.v2.GetWorkflowResponse

400	Invalid argument.	integer
401	Authentication required / Authentication failed	integer
500	Internal error in the cluster-federation service.	integer
default	An unexpected error response.	grpc.gateway.runtime.Error

Troubleshooting

Feature	Issue	Solution
Volumes Management	Create/update/get/delete command fails with the following error: rpc error: code = InvalidArgument desc = project <project_name> is not defined in cluster-federation, and multi-tenancy is not currently supported. Use the <cf_configured_project_name> project.	The <project_name> has to be the single project name defined in Cluster Federation (in the file /etc/cf/cf.yaml).
Volumes Management	Create command fails with the following error: rpc error: code = InvalidArgument desc = rpc error: code = InvalidArgument desc = no such project '<project_name>', operation details: failed to create volume in the cluster '<cluster_id>'. volume name: <volume_name>.	The <project_name> does not exist on the designated Lightbits cluster (identified by its ID).
Volumes Management	Update/get/delete command fails with the following error: rpc error: code = InvalidArgument desc = rpc error: code = InvalidArgument desc = ProjectName must be provided, operation details: failed to <update|get|delete> volume in cluster '<cluster_id>'. volume name: <volume_name>.	The --project-name attribute must be provided to the command.
Snapshot management	Create/get/delete command fails with the following error: rpc error: code = InvalidArgument desc = project	The <project_name> has to be the single project name defined in Cluster Federation (in the file /etc/cf/cf.yaml).

	<project_name> is not defined in cluster-federation, and multi-tenancy is not currently supported. Use the <cf_configured_project_name> project.	