Hitachi Content Software for File

4.0.x

CLI Reference

Hitachi Content Software for File is a high performance storage solution for AI, ML, analytics, and other GPU-accelerated workloads. It provides the speed of a distributed file system (DFS) with the capacity and hybrid cloud capabilities of an object store. The unique integration of file and object storage offers a tightly coupled, single solution for an appliance-like experience designed for ultra-high performance and capacity applications.

MK-HCSF001-02
October 2023

Contents

Preface

This guide provides information and instructions for managing the Hitachi Content Software for File (HCSF) system by using the command line interface (CLI) software.

Please read this document carefully to understand how to use this product and maintain a copy for your reference.

Intended audience

This document is intended for system administrators, Hitachi Vantara representatives, and authorized service providers who install, configure, and operate the HCSF system.

Readers of this document should be familiar with the following:

- Storage system and performance concepts, including clustering and networking.
- Storage array and tiering concepts.
- Object stores, including S3, Hitachi Content Platform, and Hitachi Content Platform for cloud scale.
- Data lifecycle management concepts.

Product version

This document revision applies to HCSF software version 4.0.*x* and later.

Release notes

Read the release notes before installing and using this product. They may contain requirements or restrictions that are not fully described in this document or updates or corrections to this document. Release notes are available on the Hitachi Vantara Support Website: https://knowledge.hitachivantara.com/Documents.

Document conventions

This document uses the following typographic conventions:

Convention	Description
Bold	▪ Indicates text in a window, including window titles, menus, menu options, buttons, fields, and labels. Example: Click **OK**. ▪ Indicates emphasized words in list items.
Italic	▪ Indicates a document title or emphasized words in text. ▪ Indicates a variable, which is a placeholder for actual text provided by the user or for output by the system. Example: `pairdisplay -g group` (For exceptions to this convention for variables, see the entry for angle brackets.)
`Monospace`	Indicates text that is displayed on screen or entered by the user. Example: `pairdisplay -g oradb`
< > angle brackets	Indicates variables in the following scenarios: ▪ Variables are not clearly separated from the surrounding text or from other variables. Example: `Status-<report-name><file-version>.csv` ▪ Variables in headings.
[ ] square brackets	Indicates optional values. Example: [a | b] indicates that you can choose a, b, or nothing.
{ } braces	Indicates required or expected values. Example: { a | b } indicates that you must choose either a or b.
| vertical bar	Indicates that you have a choice between two or more options or arguments. Examples: [a | b] indicates that you can choose a, b, or nothing. { a | b } indicates that you must choose either a or b.

This document uses the following icons to draw attention to information:

Icon	Label	Description
	Note	Calls attention to additional information.

Icon	Label	Description
	Tip	Provides helpful information, guidelines, or suggestions for performing tasks more effectively.
	Important	Highlights information that is essential to the completion of a task.
	Caution	Warns the user of adverse conditions and/or consequences (for example, disruptive operations, data loss, or a system crash).
	CAUTION	Warns the user of a hazardous situation that, if not avoided, could result in major or minor injury.
	WARNING	Warns the user of a hazardous situation which, if not avoided, could result in death or serious injury.

Accessing product documentation

Product user documentation is available on the Hitachi Vantara Support Website: https://knowledge.hitachivantara.com/Documents. Check this site for the most current documentation, including important updates that may have been made after the release of the product.

Getting help

The Hitachi Vantara Support Website is the destination for technical support of products and solutions sold by Hitachi Vantara. To contact technical support, log on to the Hitachi Vantara Support Website for contact information: https://support.hitachivantara.com/en_us/contact-us.html.

Hitachi Vantara Community is a global online community for Hitachi Vantara customers, partners, independent software vendors, employees, and prospects. It is the destination to get answers, discover insights, and make connections. **Join the conversation today!** Go to community.hitachivantara.com, register, and complete your profile.

Comments

Please send comments to doc.comments@hitachivantara.com. Include the document title and number, including the revision level (for example, -07), and refer to specific sections and paragraphs whenever possible. All comments become the property of Hitachi Vantara LLC.

Thank you!

Chapter 1: Managing the system using the Content Software for File CLI

The Content Software for File CLI is installed on each Content Software for File host and is available through the **weka** command. It's possible to connect to any of the hosts using ssh and running the **weka** command. The weka command displays a list of all top-level commands.

```
$ weka -h
Usage:
    weka [--help] [--build] [--version] [--legal]

Description:
    The base command for all weka related CLIs

Subcommands:
   agent      Command s that control the weka agent (outside the weka containers)
   alerts     List alerts in the Weka cluster
   cloud      Cloud commands. List the cluster's cloud status, if no subcommand
supplied.
   cluster    Commands that manage the cluster
   diags      Diagnostics commands to help understand the status of the cluster and
its environment
   events     List all events that conform to the filter criteria
   fs         List filesystems defined in this Weka cluster
   local      Commands that control weka and its containers on the local machine
   mount      Mounts a wekafs filesystem. This is the helper utility installed at /
sbin/mount.wekafs.
   nfs        Commands that manage client-groups, permissions and interface-groups
   org        List organizations defined in the Weka cluster
   security   Security commands.
   smb        Commands that manage Weka's SMB container
   stats      List all statistics that conform to the filter criteria
   status     Get an overall status of the Weka cluster
   umount     Unmounts wekafs filesystems. This is the helper utility installed at /
sbin/umount.wekafs.
   user       List users defined in the Weka cluster
   version    When run without arguments, lists the versions available on this
machine. Subcommands allow for
              downloading of versions, setting the current version and other actions
to manage versions.
   s3         Commands that manage Weka's S3 container

Options:
```

```
  --agent         Start the agent service
  -h, --help      Show help message
  --build         Prints the CLI build number and exits
  -v, --version   Prints the CLI version and exits
  --legal         Prints software license information and exits
```

The options that are common to many commands include:

Option	Flag description
-J|--json	Prints the raw JSON value returned by the cluster.
-H|--hostname	Directs the CLI to communicate with the cluster through the given hostname.
-raw-units	Sets the units such as capacity and bytes to be printed in their raw format, as returned by the cluster.
--UTC	Sets the timestamps to be printed in UTC timezone, instead of the local time of the machine running the CLI command.
-f|--format	Specifies the format to output the result (view, csv, markdown, or JSON).
-o|--output	Specifies the columns to include in the output.
-s|--sort	Specifies the order to sort the output. May include a '+' or '-' before the column name to sort by ascending or descending order.
-F| --filter	Specifies the filter values for a member (without forcing it to be in the output).
--no-header	Indicates that the column header should not be shown when printing the output.
C|--CONNECT-TIMEOUT	Modifies the default timeout used for connecting to the system via the JRPC protocol.
T|--TIMEOUT	Modifies the default timeout for which the commands wait for a response before giving up.

Commands hierarchy

Most Content Software for File system top-level commands are the default list command for their own collection. Additional sub-commands may be available under them.

Example:

The **`weka fs`** command displays a list of all filesystems and is also the top-level command for all filesystems, filesystem groups, and snapshot-related operations. It is possible to use the -h/--help flags or the help command to display a list of available commands at each level, as shown below:

```
$ weka fs
| FileSystem | Name    | Group   | SSD Bu | Total  | Is re | Is creat | Is remov
|  ID        |         |         | dget   | Budget | ady   | ing      | ing
+------------+---------+---------+--------+--------+-------+----------+----------
| FSId: 0    | default | default | 57 GiB | 57 GiB | True  | False    | False
```

```
$ weka fs -h
Usage:
    weka fs [--name name]
            [--HOST HOST]
            [--PORT PORT]
            [--CONNECT-TIMEOUT CONNECT-TIMEOUT]
            [--TIMEOUT TIMEOUT]
            [--format format]
            [--output output]...
            [--sort sort]...
            [--filter filter]...
            [--help]
            [--raw-units]
            [--UTC]
            [--no-header]
            [--verbose]
            [--json]

Description:
    List filesystems defined in this Weka cluster

Subcommands:
  create     Create a filesystem
  download   Download a filesystem from object store
  update     Update a filesystem
  delete     Delete a filesystem
  restore    Restore filesystem content from a snapshot
  group      List filesystem groups
  snapshot   List snapshots
  tier       Show object storage connectivity for each node in the cluster
```

```
Options:
    --name                  Filesystem name
    -H, --HOST              Specify the host. Alternatively, use the WEKA_HOST env
variable
    -P, --PORT              Specify the port. Alternatively, use the WEKA_PORT env
variable
    -C, --CONNECT-TIMEOUT   Timeout for connecting to cluster, default: 10 secs
(format: 3s, 2h, 4m, 1d, 1d5h, 1w)
    -T, --TIMEOUT           Timeout to wait for response, default: 1 minute (format:
3s, 2h, 4m, 1d, 1d5h, 1w)
    -f, --format            Specify in what format to output the result. Available
options are:
                            view|csv|markdown|json|oldview (format: 'view', 'csv',
'markdown', 'json' or 'oldview')
    -o, --output            Specify which columns to output. May include any of the
following:
                            id,name,group,usedSSDD,usedSSDM,usedSSD,freeSSD,
availableSSDM,availableSSD,usedTotalD,usedTotal,freeTotal,availableTotal,maxFiles,
status,encrypted,stores,auth
    -s, --sort              Specify which column(s) to take into account when sorting
the output. May include a '+' or
                            '-' before the column name to sort in ascending or
descending order respectively. Usage:
                            [+|-]column1[,[+|-]column2[,..]]
    F, --filter            Specify what values to filter by in a specific column.
Usage:
                           column1=val1[,column2=val2[,..]]
    h, --help              Show help message
    R, --raw-units         Print values in raw units (bytes, seconds, etc.). When not
set, sizes are printed in
                           human-readable format, e.g 1KiB 234MiB 2GiB.
    -U, --UTC               Print times in UTC. When not set, times are converted to
the local time of this host.
    --no-header             Don't show column headers when printing the output
    -v, --verbose           Show all columns in output
```

Connecting to another host

Most system commands deliver the same result on all cluster hosts. However, it is sometimes necessary to execute a command on a specific host. This is performed using the `-H`/`--hostname` option and specifying the hostname or IP address of the target host.

CLI auto-completion

Using **bash** you can use auto-completion for CLI commands and parameters. The auto-completion script is automatically installed.

- To disable the auto-completion script, run **weka agent autocomplete uninstall**
- To reinstall the script on a host, run **weka agent autocomplete install** and re-enter your shell session.

You can also use **weka agent autocomplete export** to get the bash completions script and write it to any desired location.

Cluster status

The **weka status** command displays the overall status of the Content Software for File system.

For example, if the cluster is healthy, a result similar to the following should be displayed:

Example: status of a healthy system

```
$ weka status
WekaIO v4.0.0 (CLI build 4.0.0)

       cluster: WekaProd (b231e060-c5c1-421d-a68d-1dfa94ff149b)
        status: OK (8 backends UP, 48 drives UP)
    protection: 6+2
     hot spare: 1 failure domains (1.23 TiB)
 drive storage: 82.94 TiB total
         cloud: connected
       license: OK, valid thru 2022-06-15T11:10:39Z

     io status: STARTED 2 minutes ago (8 io-nodes UP, 80 Buckets UP)
    link layer: Ethernet
       clients: 0 connected
         reads: 0 B/s (0 IO/s)
        writes: 0 B/s (0 IO/s)
    operations: 0 ops/s
        alerts: none
```

Example: status of a system with one host failure (DEGRADED)

```
$ weka status
WekaIO v4.0.0 (CLI build 4.0.0)

       cluster: WekaProd (b231e060-c5c1-421d-a68d-1dfa94ff149b)
        status: DEGRADED (7 backends UP, 42 drives UP)
                Rebuild in progress (3%)
```

```
   protection: 6+2
    hot spare: 1 failure domains (1.23 TiB)
drive storage: 82.94 TiB total
        cloud: connected
      License: OK, valid thru 2022-06-15T11:10:39Z

    io status: STARTED 54 minutes ago (8 io-nodes UP, 80 Buckets UP)
               Rebuild in progress (3%)
   link layer: Ethernet
      clients: 0 connected
        reads: 0 B/s (0 IO/s)
       writes: 0 B/s (0 IO/s)
   operations: 0 ops/s
   operations: 0 ops/s
```

Chapter 2: Getting started with Weka REST API

The Weka system supports a RESTful API. This is useful when automating the interaction with the Weka system and when integrating it into your workflows or monitoring systems.

The API is accessible at port 14000, via the `/api/v2` URL, you can explore it via `/api/v2/docs` when accessing from the cluster (e.g. `https://weka01:14000/api/v2/docs`).

Our static API documentation can be accessed from api.docs.weka.io (the version can be selected from the drop-down list). The `.json` file can also be used to create your client code, using an OpenAPI client generator.

Obtaining an access token

You must provide an access token to use the Weka REST API.

To obtain access/refresh tokens via the CLI, refer to Obtaining an authentication token (there you can also generate an access token with a longer expiry time). To obtain access/refresh tokens via the API, you can call the login API, providing it a **`username`** and **`password`**.

If you already obtained a refresh token, you can use the **`login/refresh`** API to refresh the access token.

Python example calling the login API

```
import requests
url = "https://weka01:14000/api/v2/login"
payload="{\n    \"username\": \"admin\",\n    \"password\": \"admin\"\n}"
headers = {
  'Content-Type': 'application/json'
}
response = requests.request("POST", url, headers=headers, data=payload)
print(response.text)
```

In response, you will get an access token (valid for 5 minutes), that can be used in the other APIs that require token authentication, along with the refresh token (valid for 1 year), for getting additional access tokens without using the username/password.

Login/Refresh Response

```
{
   "access_token": "ACCESS-TOKEN",
   "token_type": "Bearer",
   "expires_in": 300,
```

```
   "refresh_token": "REFRESH-TOKEN"
    }
  ]
}
```

Calling the REST API

Now, that you have obtained an access token, you can call Weka REST API commands with it. For example, you can query the cluster status:

Python example calling cluster status API

```
import requests
url = "https://weka01:14000/api/v2/cluster"
payload={}
headers = {
  'Authorization': 'Bearer REPLACE-WITH-ACCESS-TOKEN'
}
response = requests.request("GET", url, headers=headers, data=payload)
print(response.text)
```

Chapter 3: Run first IOs with WekaFS

Once the system is installed and you are familiar with the CLI and GUI, you can connect to one of the hosts and try it out.

This page guides you through:

1. The steps needed for performing IOs using a WekaFS filesystem (this is for testing the configuration):
   - Creating a filesystem (on page 21)
   - Mounting a filesystem (on page 23)
   - Writing to a filesystem (on page 23)
2. Conducting performance testing to make sure both the Weka cluster and the IT environment are best configured to reap the benefits of WekaFS.

Create the first filesystem

A filesystem must reside in a filesystem group, so first, create a filesystem group:

```
# to create a new filesystem group
$ weka fs group create my_fs_group
FSGroupId: 0

# to view existing filesystem groups details in the Weka system
$weka fs group
FileSystem Group ID | Name        | target-ssd-retention | start-demote
--------------------+-------------+----------------------+-------------
FSGroupId: 0        | my_fs_group | 1d 0:00:00h          | 0:15:00h
```

Then, you can create a filesystem within that group:

```
# to create a new filesystem
$ weka fs create new_fs my_fs_group 1TiB
FSId: 0

# to view existing filesystems details in the Weka system
$ weka fsFilesystem ID | Filesystem Name | Group       | Used SSD (Data) | Used
--------------+-----------------+-------------+---------------------------------
0             | new_fs          | my_fs_group | 0 B             | 4.09


SSD (Meta) | Used SSD | Free SSD | Available SSD (Meta) | Available SSD
```

```
+-----------------+----------+----------+-----------------------------
KB          | 4.09 KB   | 1.09 TB   | 274.87 GB            | 1.09 TB


 | Used Total (Data) | Used Total | Free Total | Available Total |
+---------------+------------------+------------+-----------------
 | 0 B               | 4.09 KB    | 1.09 TB    | 1.09 TB         |


Max Files | Status | Encrypted | Object Storages | Auth Required
+-----------------+-----------+--------+-----------+------------
22107463  | READY  | False     |                 | False
```

Note: In AWS installation via the self-service portal, default filesystem group and filesystem are created. The default filesystem is created with the entire SSD capacity.

For creating an additional filesystem, it is first needed to decrease the default filesystem SSD size:

```
# to reduce the size of the default filesystem
$ weka fs update default --total-capacity 1GiB

# to create a new filesystem in the default group
$ weka fs create new_fs default 1GiB

# to view existing filesystems details in the Weka system
$ weka fs
Filesystem ID | Filesystem Name | Group   | Used SSD (Data) | Used SSD
--------------+-----------------+---------+-----------------------------
0             | default         | default | 0 B             | 4.09 KB
1             | new_fs          | default | 0 B             | 4.09 KB


(Meta) | Used SSD | Free SSD | Available SSD (Meta) | Available SSD | +---------------
--+----------+----------+------------------------------
       | 4.09 KB  | 1.07 GB  | 268.43 MB            | 1.07 GB       |
       | 4.09 KB  | 1.09 TB  | 274.87 GB            | 1.09 TB       |


Used Total (Data) | Used Total | Free Total | Available Total | Max +-----------------
+-----------+--------+-----------+-----------------
0 B               | 4.09 KB    | 1.07 GB    | 1.07 GB         | 21589    0
B               | 4.09 KB    | 1.09 TB    | 1.09 TB         | 22107463


 Files | Status | Encrypted | Object Storages | Auth Required
+-------------------------------------------------------------
       | READY  | False     |                 | False
       | READY  | False     |                 | False
```

For more information about filesystems and filesystem groups, refer to Managing Filesystems, Object Stores & Filesystem Groups.

Mount the first filesystem

You can mount a filesystem by creating a mount point and calling the mount command:

```
$ sudo mkdir -p /mnt/weka
$ sudo mount -t wekafs new_fs /mnt/weka
```

To check the filesystem is mounted:

```
# using the mount command
$ mount | grep new_fs
new_fs on /mnt/weka type wekafs (rw,relatime,writecache,inode_bits=64,
dentry_max_age_positive=1000,dentry_max_age_negative=0)
```

Note: In AWS installation via the self-service portal, the default filesystem is already mounted under `/mnt/weka`.

For more information about mounting filesystems and mount options, refer to Mounting Filesystems.

Write to the filesystem

Now everything is set up, and you can write some data to the filesystem:

```
# to perform random writes
$ sudo dd if=/dev/urandom of=/mnt/weka/my_first_data bs=4096 count=10000
10000+0 records in
10000+0 records out
40960000 bytes (41 MB) copied, 4.02885 s, 10.2 MB/s

# to see the new file creted
$ ll /mnt/weka
total 40000
-rw-r--r-- 1 root root 40960000 Oct 30 11:58 my_first_data

# to check the WekaFS filesystems via the CLI shows the used SSD capacity:
$ weka fs
Filesystem ID | Filesystem Name | Group   | Used SSD (Data) | Used SSD (Meta) | Used
SSD | Free SSD | Available SSD (Meta) | Available SSD | Used Total (Data) | Used
Total | Free Total | Available Total | Max Files | Status | Encrypted | Object
Storages | Auth Required
--------------+-----------------+---------+-----------------+-----------------+-------
---+----------+----------------------+--------------+-------------------+------------
```

```
+------------+-----------------+-----------+--------+-----------+-----------------+---
-----------
0             | default         | default | 40.95 MB        | 180.22 KB       | 41.14
MB | 1.03 GB  | 268.43 MB            | 1.07 GB       | 40.95 MB          | 41.14 MB
| 1.03 GB    | 1.07 GB         | 21589     | READY  | False     |                 |
False
```

This has completed the check that the Content Software for File cluster is configured and IOs can be performed to it.

Validate the configuration

To make sure that the Content Software for File cluster and the IT environment are well configured, more complex IO patterns and benchmark tests should be conducted using the FIO utility.

Although results can vary using different hosts and networking, it is not expected to be very different than what we and many other customers achieved. A properly configured Weka cluster and IT environment should yield similar results as described in Testing Content Software for File Performance.

Note: The numbers achieved in the benchmark tests, as described in Testing Content Software for File Performance are not just achieved in a closed/controlled environment. Similar numbers should be achieved when using a similar configuration if the Content Software for File cluster and IT environment are properly configured. If the numbers achieved in your environment significantly vary from those, please contact the customer support before running any other workload on the Content Software for File cluster.

The example results shown in Testing Content Software for File Performance, are tested on AWS. In general, for any Content Software for File reference architecture, you should expect lower than 300 microseconds latency and 5.5 GB/s throughput per host (for a single 100gbps link).

Chapter 4: Managing object stores, filesystem groups, and filesystems

The management of object stores, filesystem groups and filesystems is an integral part of the operation and performance of the Content Software for File system and overall data lifecycle management.

Managing object stores using the CLI

Using the CLI, you can perform the following actions:

- View an object store (on page 25)
- Edit an object store (on page 25)
- View an object store bucket (on page 28)
- Add an object store bucket (on page 28)
- Edit an object store bucket (on page 31)
- Delete an object store bucket (on page 34)

Viewing object stores using the CLI

Command

```
weka fs tier obs
```

This command is used to view information on all the object stores configured to the Content Software for File system.

Note: Using the GUI only object-store buckets are present. Adding an object-store bucket will add it to the only `local` or `remote` object-store present. If more than one is present (such as during the time recovering from a remote snapshot), the CLI should be used.

Editing an object store using the CLI

Command

```
weka fs tier obs update
```

Use the following command line to edit an object store:

```
weka fs tier obs update <name> [--new-name new-name] [--site site] [--
hostname=<hostname>] [--port=<port>] [--auth-method=<auth-method>] [--
region=<region>] [--access-key-id=<access-key-id>] [--secret-key=<secret-key>] [--
protocol=<protocol>] [--bandwidth=<bandwidth>] [--download-bandwidth=<download-
bandwidth>] [--upload-bandwidth=<upload-bandwidth>] [--max-concurrent-downloads=<max-
concurrent-downloads>] [--max-concurrent-uploads=<max-concurrent-uploads>] [--max-
concurrent-removals=<max-concurrent-removals>] [--enable-upload-tags=<enable-upload-
tags>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
`name`	String	Name of the object store being edited	Must be a valid name	Yes	
`new-name`	String	New name for the object store	Must be a valid name	No	
`site`	String	`local` - for tiering +snapshots, `remote` - for snapshots only	`local` or `remote`	No	
`hostname`	String	Object store host identifier	Must be a valid name/IP	Yes	
`port`	String	Object store port	Must be a valid name	Yes	
`auth-method`	String	Authentication method	None	Yes	
`region`	String	Region name		Yes	
`access-key-id`	String	Object store access key ID		Yes	
`secret-key`	String	Object store secret key		Yes	

Name	Type	Value	Limitations	Mandatory	Default
`protocol`	String	Protocol type, to be used as a default for added buckets	`HTTP`, `HTTPS` or `HTTPS_UNVERIFIED`	No	
`bandwidth`	Number	Bandwidth limitation per core (Mbps)		No	
`download-bandwidth`	Number	Object-store download bandwidth limitation per core (Mbps)		No	
`upload-bandwidth`	Number	Object-store upload bandwidth limitation per core (Mbps)		No	
`max-concurrent-downloads`	Number	Maximum number of downloads concurrently performed on this object store in a single IO node	1-64	No	
`max-concurrent-uploads`	Number	Maximum number of uploads concurrently performed on this object store in a single IO node	1-64	No	

Name	Type	Value	Limitations	Mandatory	Default
`max-concurrent-removals`	Number	Maximum number of removals concurrently performed on this object store in a single IO node	1-64	No	
`enable-upload-tags`	String	Whether to enable object-tagging or not, to be used as a default for added buckets	`true` or `false`	No	

Viewing object store buckets

Command:

```
weka fs tier s3
```

This command is used to view information on all the object-store buckets configured to the Weka system.

Adding an object store bucket using the CLI

Command

`weka fs tier s3 add`

Use the following command line to add an object store:

```
weka fs tier s3 add <name> [--site site] [--obs-name obs-name] [--
hostname=<hostname>] [--port=<port> [--bucket=<bucket>] [--auth-method=<auth-method>]
[--region=<region>] [--access-key-id=<access-key-id>] [--secret-key=<secret-key>] [--
protocol=<protocol>] [--bandwidth=<bandwidth>] [--download-bandwidth=<download-
bandwidth>] [--upload-bandwidth=<upload-bandwidth>] [--errors-timeout=<errors-
timeout>] [--prefetch-mib=<prefetch-mib>] [--enable-upload-tags=<enable-upload-tags>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
`name`	String	Name of the object-store bucket being created	Must be a valid name	Yes	
`Site`	String	`local` - for tiering +snapshots, `remote` - for snapshots only	Must be the same as the object store site it is added to (`obs-name`)	No	`Local`
`obs-name`	String	Name of the object-store to add this object-store bucket to	Must be an existing object-store	No	If there is only one object-store of type mentioned in `site` it is chosen automatically
`hostname`	String	Object store host identifier	Must be a valid name/IP	Yes, if not specified in the object-store level	The hostname specified in `obs-name` if present
`port`	String	Object store port	Must be a valid name	No	The `port` specified in `obs-name` if present, otherwise 80
`bucket`	String	Object store bucket name	Must be a valid name	Yes	
`auth-method`	String	Authentication method	`None`, `AWSSignature2` or `AWSSignature4`	Yes, if not specified in the object-store level	The `auth-method` specified in `obs-name` if present
`region`	String	Region name		Yes, if not specified in the object-store level	The `region` specified in `obs-name` if present

Name	Type	Value	Limitations	Mandatory	Default
`access-key-id`	String	Object store bucket access key ID		Yes, if not specified in the object-store level (can be left empty when using IAM role in AWS)	The `access-key-id` specified in `obs-name` if present
`secret-key`	String	Object store bucket secret key		Yes, if not specified in the object-store level (can be left empty when using IAM role in AWS)	The `secret-key` specified in `obs-name` if present
`protocol`	String	Protocol type to be used	`HTTP`, `HTTPS` or `HTTPS_UNVERIFIED`	No	The `protocol` specified in `obs-name` if present, otherwise HTTP
`bandwidth`	Number	Bucket bandwidth limitation per core (Mbps)		No	
`download-bandwidth`	Number	Bucket download bandwidth limitation per core (Mbps)		No	
`upload-bandwidth`	Number	Bucket upload bandwidth limitation per core (Mbps)		No	

Name	Type	Value	Limitations	Mandatory	Default
`errors-timeout`	Number	If the object-store link is down for longer than this timeout period, all IOs that need data return with an error	1-15 minutes, e.g: 5m or 300s	No	300
`prefetch-mib`	Number	How many MiB of data to prefetch when reading a whole MiB on the object store		No	0
`enabme-upload-tags`	String	Whether to enable `object-tagging` or not	`true or false`	No	`false`

Note: When using the CLI, by default a misconfigured object store will not be created. To create an object store even when it is misconfigured, use the `--skip-verification` option.

Note: The max-concurrent settings are applied per Content Software for File compute process and the minimum setting of all object-stores is applied.

Make the relevant changes and click Update to update the object store bucket.

Editing an object store bucket using the CLI

Command

`weka fs tier s3 update`

Use the following command line to edit an object store bucket:

```
weka fs tier s3 update <name> [--new-name=<new-name>] [--new-obs-name new-obs-name] [--hostname=<hostname>] [--port=<port> [--bucket=<bucket>] [--auth-method=<auth-method>] [--region=<region>] [--access-key-id=<access-key-id>] [--secret-key=<secret-key>] [--protocol=<protocol>] [--bandwidth=<bandwidth>] [--download-
```

```
bandwidth=<download-bandwidth>] [--upload-bandwidth=<upload-bandwidth>] [--errors-
timeout=<errors-timeout>] [--prefetch-mib=<prefetch-mib>] [--enable-upload-
tags=<enable-upload-tags>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
`name`	String	Name of the object store being edited	Must be a valid name	Yes	
`new-name`	String	New name for the object store	Must be a valid name	No	
new-obs-name	String	New name of the object-store to add this object-store bucket to	Must be an existing object-store, with the same `site` value.	No	
`hostname`	String	Object store host identifier	Must be a valid name/IP	No	
`port`	String	Object store port	Must be a valid name	No	
bucket	String	Object store bucket name	Must be a valid name	No	
`auth-method`	String	Authentication method	None, `AWSSignature2` or `AWSSignature4`	No	
`region`	String	Region name		No	
`access-key-id`	String	Object-store bucket access key ID		No	
`secret-key`	String	Object-store bucket secret key		No	

Name	Type	Value	Limitations	Mandatory	Default
`protocol`	String	Protocol type to be used	`HTTP`, `HTTPS` or `HTTPS_UNVERIFIED`	No	
`bandwidth`	Number	Bandwidth limitation per core (Mbps)		No	
`download-bandwidth`	Number	Bucket download bandwidth limitation per core (Mbps)		No	
`upload-bandwidth`	Number	Bucket upload bandwidth limitation per core (Mbps)		No	
errors-timeout		If the object-store link is down for longer than this timeout period, all IOs that need data return with an error	1-15 minutes, e.g: 5m or 300s	No	
prefetch-mib		How many MiB of data to prefetch when reading a whole MiB on the object store		No	
Z	`enable-upload-tags`	String	Whether to enable object-tagging or not	`true` or `false`	No

Deleting an object store bucket using the CLI

Command

`weka fs tier s3 delete`

Use the following command line to delete an object store:

```
weka fs tier s3 delete <name>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the object store being deleted	Must be a valid name	Yes	

Managing filesystem groups

Using the CLI, you can perform the following actions:

- View filesystem groups (on page 34)
- Add filesystem groups (on page 34)
- Edit filesystem groups (on page 35)
- Delete filesystem groups (on page 36)

Viewing filesystem groups using the CLI

Command

`weka fs group`

Use this command to view information about the filesystem groups in the system.

Adding a filesystem group using the CLI

Command

`weka fs group create`

Use the following command to add a filesystem group:

```
weka fs group create <name> [--target-ssd-retention=<target-ssd-retention>] [--start-demote=<start-demote>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem group being created	Must be a valid name	Yes	
target-ssd-retention	Number	Target retention period (in seconds) before tiering to the object store	Must be a valid number	No	86400 (24 hours)
start-demote	Number	Target tiering cue (in seconds) before tiering to the object store	Must be a valid number	No	10

Editing a filesystem group using the CLI

Command

`weka fs group update`

Use the following command to edit a filesystem group:

```
weka fs group update <name> [--new-name=<new-name>] [--target-ssd-retention=<target-ssd-retention>] [--start-demote=<start-demote>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem group being edited	Must be a valid name	Yes	
new-name	String	New name for the filesystem group	Must be a valid name	Yes	

Name	Type	Value	Limitations	Mandatory	Default
target-ssd-retention	Number	New target retention period (in seconds) before tiering to the object store	Must be a valid number	No	
start-demote	Number	New target tiering cue (in seconds) before tiering to the object store	Must be a valid number	No	

Deleting a filesystem group using the CLI

Command

`weka fs group delete`

Use the following command line to delete a filesystem group:

```
weka fs group delete <name>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem group to delete	Must be a valid name	Yes	

Managing filesystems

Using the CLI, you can perform the following actions:

- View filesystems (on page 37)
- Add a filesystem (on page 37)
- Add a filesystme when thin-provisioning is used (on page 39)

- Edit a filesystem (on page 39)
- Delete a filesystem (on page 41)

Viewing filesystems using the CLI

Command

`weka fs`

Use this command to view information on the filesystems in the Content Software for File system.

Enter the relevant parameters and click Create to create the filesystem.

Adding a filesystem using the CLI

Command

`weka fs create`

Use the following command line to add a filesystem:

```
weka fs create <name> <group-name> <total-capacity> [--ssd-capacity <ssd-capacity>] [--thin-provision-min-ssd <thin-provision-min-ssd>] [--thin-provision-max-ssd <thin-provision-max-ssd>] [--max-files <max-files>] [--encrypted] [--obs-name <obs-name>] [--auth-required <auth-required>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem being created	Must be a valid name	Yes	
group-name	String	Name of the filesystem group to which the new filesystem is to be connected	Must be a valid name	Yes	
total-capacity	Number	Total capacity of the new filesystem	Minimum of 1GiB	Yes	

Name	Type	Value	Limitations	Mandatory	Default
ssd-capacity	Number	For tiered filesystems, this is the SSD capacity. If not specified, the filesystem is pinned to SSD	Minimum of 1GiB	No	SSD capacity will be set to total capacity
thin-provision-min-ssd	Number	For thin-provisioned filesystems, this is the minimum SSD capacity that is ensured to be always available to this filesystem	Minimum of 1GiB	No. Must be set when defining a thin-provisioned filesystem.	
thin-provision-max-ssd	Number	For thin-provisioned filesystem, this is the maximum SSD capacity the filesystem can consume	Cannot exceed the **`total-capacity`**		
max-files	Number	Metadata allocation for this filesystem	Must be a valid number	No	Automatically calculated by the system based on the SSD capacity
encrypted	Boolean	Encryption of filesystem		No	No
obs-name	String	Object store name for tiering	Must be a valid name	Mandatory for tiered filesystems	

Name	Type	Value	Limitations	Mandatory	Default
auth-required	String	Determines if mounting the filesystem requires to be authenticated to Content Software for File	`yes` or `no` For a filesystem hosting NFS exports or SMB shares, enabling authentication is not allowed.	No	no

Note: When creating an encrypted filesystem a KMS must be defined.

Note:

- To define an encrypted filesystem without a KMS, it is possible to use the **`--allow-no-kms`** parameter in the command. This can be useful when running POCs but should not be used in production, since the security chain is compromised when a KMS is not used.
- If filesystem keys exist when adding a KMS, they are automatically re-encrypted by the KMS for any future use.

Add a filesystem when thin-provisioning is used

To create a new filesystem, the SSD space for the filesystem must be free and unprovisioned. When using thin-provisioned filesystems, that might not be the case. SSD space can be occupied for the thin-provisioned portion of other filesystems. Even if those are tiered, and data can be released (to object-store) or deleted, the SSD space can still get filled when data keeps being written or rehydrated from the object-store.

To create a new filesystem in this case, use the **`weka fs reserve`** CLI command. Once enough space is cleared from the SSD (either by releasing to object-store or explicit deletion of data), it is possible to create the new filesystem using the reserved space.

Editing a filesystem using the CLI

Command

`weka fs update`

Use the following command line to edit an existing filesystem:

```
weka fs update <name> [--new-name=<new-name>] [--total-capacity=<total-capacity>] [--
ssd-capacity=<ssd-capacity>] [--thin-provision-min-ssd <thin-provision-min-ssd>] [--
```

```
thin-provision-max-ssd <thin-provision-max-ssd>] [--max-files=<max-files>] [--auth-
required=<auth-required>]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem being edited	Must be a valid name	Yes	
new-name	String	New name for the filesystem	Must be a valid name	Optional	Keep unchanged
total-capacity	Number	Total capacity of the edited filesystem	Must be a valid number	Optional	Keep unchanged
ssd-capacity	Number	SSD capacity of the edited filesystem	Minimum of 1GiB	Optional	Keep unchanged
thin-provision-min-ssd	Number	For thin-provisioned filesystems, this is the minimum SSD capacity that is ensured to be always available to this filesystem	Minimum of 1GiB	Optional	
thin-provision-max-ssd	Number	For thin-provisioned filesystems, this is the maximum SSD capacity the filesystem can consume	Cannot exceed the **`total-capacity`**	Optional	
max-files	Number	Metadata limit for the filesystem	Must be a valid number	Optional	Keep unchanged

Name	Type	Value	Limitations	Mandatory	Default
auth-required	String	Determines if mounting the filesystem requires to be authenticated to Content Software for File	`yes` or `no` For a filesystem hosting NFS exports or SMB shares, enabling authentication is not allowed.	No	no

Deleting a filesystem using the CLI

Command

`weka fs delete`

Use the following command line to delete a filesystem:

```
weka fs delete <name> [--purge-from-obs]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem to be deleted	Must be a valid name	Yes	
purge-from-obs	Boolean	For a tiered filesystem, if set, all filesystem data is deleted from the object store bucket.		No	False

Note: Using **`purge-from-obs`** will remove all data from the object-store. This includes any backup data or snapshots created from this filesystem (if this filesystem has been downloaded from a snapshot of a different filesystem, it will leave the original snapshot data intact).

- If any of the removed snapshots have been (or are) downloaded and used by a different filesystem, that filesystem will stop functioning correctly, data might be unavailable and errors might occur when accessing the data.

It is possible to either un-tier or migrate such a filesystem to a different object store bucket before deleting the snapshots it has downloaded.

Attach or detach object store buckets using the CLI

Using the CLI, you can:

- Attach an object store bucket to a filesystem (on page 42)
- Detach an object store bucket from a filesystem (on page 43)

Attaching an object stores bucket to a filesystem using the CLI

Command

`weka fs tier s3 attach`

To attach an object store to a filesystem, use the following command:

```
weka fs tier s3 attach <fs-name> <obs-name> [--mode mode]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
`fs-name`	String	Name of the filesystem to be attached to the object store	Must be a valid name	Yes	
`obs-name`	String	Name of the object store to be attached	Must be a valid name	Yes	
`mode`	String	`local` or `remote`	A `local` bucket can only be attached as `local` and a `remote` bucket can only be attached as `remote`	No	

Detaching an object store bucket from a filesystem using the CLI

Command:

`weka fs tier s3 detach`

To detach an object store from a filesystem, use the following command:

```
weka fs tier s3 detach <fs-name> <obs-name>
```

Parameters:

Name	Type	Value	Limitations	Mandatory	Default
`fs-name`	String	Name of the filesystem to be detached from the object store	Must be a valid name	Yes	
`obs-name`	String	Name of the object store to be detached	Must be a valid name	Yes	

Note: To recover from a snapshot that has been uploaded when two **`local`** object stores have been attached, use the **`--additional-obs`** parameter in **`weka fs download`** command. The primary object-store should be the one where the locator has been uploaded to.

Mounting filesystems

How to use a filesystem through the Content Software for File filesystem driver, it has to be mounted on one of the cluster hosts. This section describes how this is performed.

Overview

There are two methods available for mounting a filesystem in one of the cluster hosts:

1. Using the traditional method: See below and also refer to Adding Clients (Bare Metal Installation) or Adding Clients (AWS Installation), where first a client is configured and joins a cluster, after which a mount command is executed.
2. Using the stateless clients feature: See Mounting Filesystems Using the Stateless Clients Feature (on page 44), which simplifies and improves the management of clients in the cluster and eliminates the adding clients process.

Mounting a filesystem using the traditional method

> **Note:** Using the **`mount`** command as explained below first requires the installation of the Content Software for File client, configuring the client, and joining it to a Content Software for File cluster.

To mount a filesystem on one of the cluster hosts, let's assume the cluster has a filesystem called `demo`. To add this filesystem to a host, SSH into one of the hosts and run the **`mount`** command as the `root` user, as follows:

```
mkdir -p /mnt/weka/demo
mount -t wekafs demo /mnt/weka/demo
```

The general structure of a **`mount`** command for a Content Software for File filesystem is:

```
mount -t wekafs [-o option[,option]...]] <fs-name> <mount-point>
```

There are two options for mounting a filesystem on a cluster client: read cache and write cache. For more information on the differences between these modes, see read cache and write cache mount modes in the *Hitachi Content Software for File User Guide*.

Mounting a filesystem using the stateless clients feature

The *Stateless Clients* feature defers the process of joining the cluster until the mount is performed. Simplifying and improving the management of clients in the cluster. It removes tedious client management procedures, which is particularly beneficial in AWS installations where clients may join and leave at high frequency.

Furthermore, it unifies all security aspects in the mount command, eliminating the search for separate credentials at cluster join and mount.

To use the Stateless Clients feature, a Content Software for File agent must be installed. Once this is complete, mounts can be created and configured using the mount command and can be easily removed from the cluster using the **`unmount`** command.

> **Note:** To allow only Content Software for File authenticated users to mount a filesystem, set the filesystem `auth-required` flag to `yes`. For more information about mount authentication for organization filesystems, see *Hitachi Content Software for File User Guide*.

Assuming the Content Software for File cluster is using the backend IP of 1.2.3.4, running the following command as `root` on a client will install the agent:

```
curl http://1.2.3.4:14000/dist/v1/install | sh
```

On completion, the agent is installed on the client machine.

Run the mount command

Command:

```
mount -t wekafs
```

Use one of the following command lines to invoke the mount command (note, the delimiter between the server and filesystem can be either `:/` or `/`):

```
mount -t wekafs -o <options> <backend0>[,<backend1>,...,<backendN>]/<fs> <mount-point>
```

```
mount -t wekafs -o <options> <backend0>[,<backend1>,...,<backendN>]:/<fs> <mount-
point>
```

1.

Parameters

Name	Type	Value	Limitations	Mandatory	Default
Options		See additional mount options below			
backend	String	IP/hostname of a backend host	Must be a valid name	Yes	
fs	String	Filesystem name	Must be a valid name	Yes	
mount-point	String	Path to mount on the local machine	Must be a valid path-name	Yes	

Mount command options

Each mount option can be passed by an individual **`-o`** flag to **`mount`**.

For all clients types

Option	Value	Description	Default
`readcache`	None	Set mode to read cache	No
`writecache`	None	Set mode to write cache	Yes

Option	Value	Description	Default
`dentry_max_age_positive`	Number in milliseconds	After the defined time period, every metadata cached entry is refreshed from the system, allowing the host to take into account metadata changes performed by other hosts.	1000
`dentry_max_age_negative`	Number in milliseconds	Each time a file or directory lookup fails, an entry specifying that the file or directory does not exist is created in the local dentry cache. This entry is refreshed after the defined time, allowing the host to use files or directories created by other hosts.	0
`ro`	None	Mount filesystem as read-only	No
`rw`	None	Mount filesystem as read-write	Yes
`inode_bits`	32, 64 or auto	Size of the inode in bits, which may be required for 32-bit applications.	Auto
`verbose`	None	Write debug logs to the console	No
`quiet`	None	Don't show any logs to console	No

Option	Value	Description	Default
`acl`	None	Can be defined per mount. Setting POSIX ACLs can change the effective group permissions (via the `mask` permissions). When ACLs defined but the mount has no ACL, the effective group permissions are granted.)	No
`obs_direct`	None	See Object-store Direct Mount section	No
`noatime`	None	Do not update inode access times	No
`strictatime`	None	Always update inode access times	No
`relatime`	None	Update inode access times only on modification or change, or if inode has been accessed and `relatime_threshold` has passed.	Yes
`relatime_threshold`	Number in seconds	How much time (in seconds) to wait since an inode has been accessed (not modified) before updating the access time. 0 means to never update the access time on access only. This option is relevant only if `relatime` is on.	0 (infinite)
`nosuid`	None	Do not take `suid`/`sgid` bits into effect.	No

Option	Value	Description	Default
`nodev`	None	Do not interpret character or block special devices.	No
`noexec`	None	Do not allow direct execution of any binaries.	No
`file_create_mask`	Numeric (octal) notation of POSIX permissions	Newly created file permissions are masked with the creation mask. For example, if a user creates a file with permissions=777 but the `file_create_mask` is 770, the file will be created with 770 permissions. First, the `umask` is taken into account, followed by the `file_create_mask` and then the `force_file_mode`.	0777

Option	Value	Description	Default
`directory_create_mask`	Numeric (octal) notation of POSIX permissions	Newly created directory permissions are masked with the creation mask. For example, if a user creates a directory with permissions=777 but the `directory_create_mask` is 770, the directory will be created with 770 permissions. First, the `umask` is taken into account, followed by the `directory_create_mask` and then the `force_directory_mode`.	0777
`force_file_mode`	Numeric (octal) notation of POSIX permissions	Newly created file permissions are logically OR'ed with the mode. For example, if a user creates a file with permissions 770 but the `force_file_mode` is 775, the resulting file will be created with mode 775. First, the `umask` is taken into account, followed by the `file_create_mask` and then the `force_file_mode`.	0

Option	Value	Description	Default
`force_directory_mode`	Numeric (octal) notation of POSIX permissions	Newly created directory permissions are logically OR'ed with the mode. For example, if a user creates a directory with permissions 770 but the `force_directory_mode` is 775, the resulting directory will be created with mode 775. First, the `umask` is taken into account, followed by the `directory_create_mask` and then the `force_directory_mode`.	0

Remount of general options

You can remount using the mount options marked as `Remount Supported` in the above table (`mount -o remount`).

When a mount option has been explicitly changed, you must set it again in the remount operation to ensure it retains its value. For example, if you mount with ro, a remount without it changes the mount option to the default `rw`. If you mount with `rw`, it is not required to re-specify the mount option because this is the default.

Additional mount options using the stateless clients feature

Option	Value	Description	Default	Remount Supported
`memory_mb=<memory_mb>`	Number	Amount of memory to be used by the client (for huge pages).	1400 MiB	Yes

Option	Value	Description	Default	Remount Supported
`num_cores=<frontendcores>`	Number	The number of frontend cores to allocate for the client. Either `<num_cores>` or `<core>` can be specified, but not both. If none are specified, the client will be configured with 1 core. If 0 is specified then you must use `net=udp`.	1	No
`core=<core>`	Number	Specify explicit cores to be used by the Content Software for File FS client. Multiple cores can be specified. Core 0 is not allowed.		No
`net=<netdev>[/<ip>/<bits>[/<gateway>]]`	String	For more info refer to Advanced network configuration using mount options (on page 55) section.		No
`bandwidth_mbps=<bandwidth_mbps>`	Number	Maximum network bandwidth in Mb/s, which limits the traffic that the container can send.	Auto-select	Yes
`remove_after_secs=<secs>`	Number	The number of seconds without connectivity after which the client will be removed from the cluster. Minimum value: 60 seconds.	86,400 seconds (24 hours)	Yes
`traces_capacity_mb=<size-in-mb>`	Number	Traces capacity limit in MB. Minimum value: 512 MB.		No
`reserve_1g_hugepages`	None	Controls the page allocation algorithm if to reserve only 2MB huge pages or also 1GB ones.	Yes	Yes
`readahead_kb=<readahead>`	Number in KB	Controls the readahead per mount (higher readahead better for sequential reads of large files).	32768	Yes

Option	Value	Description	Default	Remount Supported
`auth_token_path`	String	Path to the mount authentication token (per mount).	`~/.weka/auth-token.json`	
dedicated_mode	`full` or `none`	Determine whether DPKD networking dedicates a core (`full`) or not (`none`). none can only be set when the NIC driver supports it. See DPDK Without Code Dedication section.This option is relevant when using DPDK networking (`net=udp` is not set).	full	
qos_preferred_throughput_mbps	Number	Preferred requests rate for QoS in megabytes per second.	No limit. The cluster admin can set this default. See mount option defaults.	Yes
qos_max_throughput_mbps	Number	Maximum requests rate for QoS in megabytes per second. This option allows bursting above the specified limit but aims to keep this limit on average.	No limit. The cluster admin can set this default. See mount option defaults.	Yes
qos_max_ops	Number	Maximum number of IO operations a client can perform per second. Set a limit to a client or clients to prevent starvation from the rest of the clients.	No limit. Do not set this option for mounting from a backend.	Yes

Option	Value	Description	Default	Remount Supported
connect_timeout_secs	Number	The timeout in seconds for establishing a connection to a single host.	10	Yes
response_timeout_secs	Number	The timeout in seconds for waiting for the response from a single host.	60	Yes
join_timeout_secs	Number	The timeout, in seconds, for the client container to join the Content Software for File cluster.	360	Yes

Note: These parameters, if not stated otherwise, are only effective on the first **`mount`** command for each client.

Note: By default, the command selects the optimal core allocation for Content Software for File. If necessary, multiple `core` parameters can be used to allocate specific cores to the WekaFS client. For example,

```
mount -t wekafs -o core=2 -o core=4 -o net=ib0 backend-host-0/my_fs /mnt/weka
```

On-Premise Installations

```
mount -t wekafs -o num_cores=1 -o net=ib0 backend-host-0/my_fs /mnt/weka
```

Running this command on a host installed with the Content Software for File agent will download the appropriate version from the host `backend-host-0` and create a container which allocates a single core and a named network interface `ib0`. Then it will join the cluster that `backend-host-0` is part of and mount the filesystem `my_fs` on `/mnt/weka`.

```
mount -t wekafs -o num_cores=0 -o net=udp backend-host-0/my_fs
/mnt/weka
```

Running this command will use UDP mode (usually selected when the use of DPDK is not available).

For stateless clients, the first `mount` command installs the weka client software and joins the cluster). Any subsequent `mount` command, can either use the same syntax or just the traditional/per-mount parameters as defined in Mounting Filesystems since it is not necessary to join a cluster.

It is now possible to access Content Software for File filesystems via the mount-point, by the `cd /mnt/weka/` command.

After the execution of anumount command, which unmounts the last Weka filesystem, the client is disconnected from the cluster and will be uninstalled by the agent. Consequently, executing a new mount command requires the specification of the cluster, cores, and networking parameters again.

Note: Memory allocation for a client is predefined. Contact contact your Hitachi representative when it is necessary to change the amount of memory allocated to a client.

Remount of stateless clients options

Mount options marked as `Remount Supported` in the above table can be remounted (using `mount -o remount`). When a mount option is not set in the remount operation, it will retain its current value. To set a mount option back to its default value, use the `default` modifier (e.g., `memory_mb=default)`.

Set mount option default values

The defaults of the mount options `qos_max_throughput_mbps` and `qos_preferred_throughput_mbps` have no limit.

The cluster admin can set these default values to meet the organization's requirements, reset to the initial default values (no limit), or show the existing values.

The mount option defaults are only relevant for new mounts performed and do not influence the existing ones.

Commands:

```
weka cluster mount-defaults set
```

```
weka cluster mount-defaults reset
```

```
weka cluster mount-defaults show
```

To set the mount option default values, run the following command:

```
weka cluster mount-defaults set [--qos-max-throughput qos-max-throughput] [--qos-preferred-throughput qos-preferred-throughput]
```

Parameters

Option	Value	Description
`qos_max_throughput`	Number	Sets the default value for the `qos_max_throughput_mbps` option, which is the max requests rate for QoS in megabytes per second
`qos_preferred_throughput`	Number	Sets the default value for the `qos_preferred_throughput_mbps` option, which is the preferred requests rate for QoS in megabytes per second.

Advanced network configuration using mount options

When using a stateless client, it is possible to alter and control many different networking options, such as:

- Virtual functions.
- IPs.
- Gateway (in case the client is on a different subnet).
- Physical network devices (for performance and HA).
- UDP mode.

Use `-o net=<netdev>` mount option with the various modifiers as described below.

`<netdev>` is either the name, MAC address, or PCI address of the physical network device (can be a bond device) to allocate for the client.

Note: When using **`wekafs`** mounts, both clients and backends should use the same type of networking technology (either IB or Ethernet).

IP, subnet, gateway, and virtual functions

For higher performance, the usage of multiple Frontends may be required. When using a NIC other than Mellanox or Intel E810, or when mounting a DPDK client on a VM, it is required to use SR-IOV to expose a VF of the physical device to the client. Once exposed, it can be configured via the mount command.

When you want to determine the VFs IP addresses, or when the client resides in a different subnet and routing is needed in the data network, use:

```
net=<netdev>/[ip]/[bits]/[gateway]
```

The `ip, bits, gateway` parameters are optional. In case they are not provided, the Content Software for File system tries to deduce them when in IB environments or allocate from the default data network otherwise. If both approaches fail, the mount command will fail.

For example, the following command will allocate two cores and a single physical network device (intel0). It will configure two VFs for the device and assign each one of them to one of the frontend nodes. The first node will receive 192.168.1.100 IP address, and the second will use 192.168.1.101 IP address. Both of the IPs have a 24 network mask bits and default gateway of 192.168.1.254.

```
mount -t wekafs -o num_cores=2 -o
net=intel0/192.168.1.100+192.168.1.101/24/192.168.1.254 backend1/my_fs /mnt/weka
```

Multiple physical network devices for performance and HA

For performance or high availability, it is possible to use more than one physical network device.

Using multiple physical network devices for better performance

It's easy to saturate the bandwidth of a single network interface when using WekaFS. For higher throughput, it is possible to leverage multiple network interface cards (NICs\). The `-o net` notation shown in the next example can be used to pass the names of specific NICs to WekaFS host driver.

For example, the following command will allocate two cores and two physical network devices for increased throughput:

```
mount -t wekafs -o num_cores=2 -o net=mlnx0,net=mlnx1 backend1/my_fs /mnt/weka
```

Using multiple physical network devices for HA configuration

Multiple NICs can also be configured to achieve redundancy (refer to Content Software for File Installation Guide, HA networking configuration section for more information) in addition to higher throughput, for a complete, highly available solution. For that, use more than one physical device and specify the client management IPs using the command-line option:

```
-o mgmt_ip=<ip>+<ip2>
```

For example, the following command will use two network devices for HA networking and allocate both devices to four Frontend processes on the client. The modifier `ha` is used here, which stands for using the device on all processes.

```
mount -t wekafs -o num_cores=4 -o net:ha=mlnx0,net:ha=mlnx1 backend1/my_fs -o
mgmt_ip=10.0.0.1+10.0.0.2 /mnt/weka
```

Advanced mounting options for multiple physical network devices

With multiple Frontend processes (as expressed by `-o num_cores`), it is possible to control what processes use what NICs. This can be accomplished through the use of special command line modifiers called *slots*. In WekaFS, *slot* is synonymous with a process number. Typically, the first WekaFS Frontend process will occupy *slot 1*, then the second *slot 2* and so on.

Examples of slot notation include `s1`, `s2`, `s2+1`, `s1-2`, `slots1+3`, `slot1`, `slots1-4`, where `-` specifies a range of devices, while `+` specifies a list. For example, `s1-4` implies slots 1, 2, 3 and 4, while `s1+4` specifies slots 1 and 4 only. For example, in the following command, `mlnx0` is bound to the second Frontend process while `mlnx1` to the first one for improved performance.

```
mount -t wekafs -o num_cores=2 -o net:s2=mlnx0,net:s1=mlnx1 backend1/my_fs /mnt/weka
```

For example, in the following HA mounting command, two cores (two Frontend processes) and two physical network devices (`mlnx0, mlnx1`) are allocated. By explicitly specifying `s2+1, s1-2` modifiers for network devices, both devices will be used by both Frontend processes. Notation `s2+1` stands for the first and second processes, while `s1-2` stands for the range of 1 to 2, and are effectively the same.

```
mount -t wekafs -o num_cores=2 -o net:s2+1=mlnx0,net:s1-2=mlnx1 backend1/my_fs -o
mgmt_ip=10.0.0.1+10.0.0.2 /mnt/weka
```

UDP mode

In cases where the Data Plane Development Kit (DPDK) cannot be used, it is possible to use WekaFS in User Datagram Protocol (UDP) mode through the kernel. Use **`net=udp`** in the mount command to set the UDP networking mode, for example:

```
mount -t wekafs -o num_cores=0 -o net=udp backend-host-0/my_fs /mnt/weka
```

Note: A client in UDP mode cannot be configured in HA mode. However, the client can still work with a highly available cluster.

Note: Providing multiple IPs in the <mgmt-ip> in UDP mode will utilize their network interfaces for more bandwidth (can be useful in RDMA environments), rather than using only one NIC.

Mounting filesystems using fstab

Note: This option works when using stateless clients and with OS that supports `systemd` (for example, RHEL/CentOS 7.2 and up, Ubuntu 16.04 and up, Amazon Linux 2 LTS).

Edit `/etc/fstab` file to include the filesystem mount entry:

- A comma-separated list of backend hosts, with the filesystem name
- The mount point
- Filesystem type - `wekafs`
- Mount options:
  - Configure `systemd` to wait for the `weka-agent` service to come up, and set the filesystem as a network filesystem, for example:

    ```
    x-systemd.requires=weka-agent.service,x-systemd.mount-timeout=infinity,_netdev
    ```

  - Any additional `wekafs` supported mount option

```
# create a mount point
mkdir -p /mnt/weka/my_fs

# edit fstab file
vi /etc/fstab

# fstab with weka options (example, change with your desired settings)
backend-0,backend-1,backend-3/my_fs /mnt/weka/my_fs wekafs num_cores=1,net=eth1,
x-systemd.requires=weka-agent.service,x-systemd.mount-timeout=infinity,_netdev
0 0
```

Reboot the machine for the `systemd` unit to be created and marked correctly.

The filesystem should now be mounted at boot time.

Note: Do not configure this entry for a mounted filesystem before un-mounting it **`unmount`**, as the `systemd` needs to mark the filesystem as a network filesystem (occurs as part of the `reboot`. Trying to reboot a host when there is a mounted WekaFS filesystem when setting its `fstab` configuration might yield a failure to unmount the filesystem and leave the system hanged.

Mounting filesystems using autofs

It is possible to mount a Content Software for File filesystem using the **`autofs`** command.

Procedure

1. Install `autofs` on the host using one of the following commands according to your deployment:
   - On RedHat or Centos:

     ```
     yum install -y autofs
     ```

   - On Debian or Ubuntu:

     ```
     apt-get install -y autofs
     ```

2. To create the `autofs` configuration files for Content Software for File filesystems, do one of the following depending on the client type:
   - For a stateless client, run the following commands (specify the backend names as parameters):

     ```
     echo "/mnt/weka   /etc/auto.wekafs -fstype=wekafs,num_cores=1,
     net=<netdevice>" > /etc/auto.master.d/wekafs.autofs
     echo "*   <backend-1>,<backend-2>/&" > /etc/auto.wekafs
     ```

   - For a stateful client (traditional), run the following commands:

     ```
     echo "/mnt/weka /etc/auto.wekafs -fstype=wekafs" > /etc/auto.master.d/
     wekafs.autofs
     echo "* &" > /etc/auto.wekafs
     ```

3. Restart the `autofs` service:

   ```
   service autofs restart
   ```

4. The configuration is distribution-dependent. Verify that the service is configured to start automatically after restarting the host. Run the following command:

   ```
   systemctl is-enabled autofs.
   ```

   If the output is enabled the service is configured to start automatically.

Example

In Amazon Linux, you can verify that the `autofs` service is configured to start automatically by running the command `chkconfig`. If the output is `on` for the current runlevel (you can check with the `runlevel` command), `autofs` is enabled upon restart.

```
# chkconfig | grep autofs
autofs          0:off 1:off 2:off 3:on 4:on 5:on 6:off
```

Once you complete this procedure, it is possible to access Content Software for File filesystems using the command `cd /mnt/weka/<fs-name>`.

Chapter 5: Manual fetch and release of data using the CLI

This page describes how to manually force-fetching tiered data back to SSDs, and force-releasing SSD data to object-store.

Pre-fetching API for data lifecycle management using the CLI

Pre-fetching API for data lifecycle management using the CLI.

Tiered files are always accessible and should generally be treated like regular files Moreover, while files may be tiered, their metadata is always maintained on the NVMe SSDs. This allows traversing files and directories without worrying about how such operations may affect performance.

Sometimes, it may be necessary to access previously-tiered files quickly. In such situations, it is possible to request the Content Software for File system to fetch the file back to the SSD without accessing them directly. This is performed using the prefetch **`weka fs tier fetch`** command, which can be issued using the command.

Fetching files from an object store using the CLI

Tiered files are always accessible and should generally be treated like regular files. Moreover, while files may be tiered, their metadata is always maintained on the SSDs. This allows traversing files and directories without worrying about how such operations may affect performance.

Sometimes, it's necessary to access previously-tiered files quickly. In such situations, it is possible to request the Weka system to fetch the files back to the SSD without accessing them directly. This is performed using the prefetch command, which can be issued via the **`weka fs tier fetch`** command, as follows:

Command

`weka fs tier fetch`

Use the following command to release files:

```
weka fs tier fetch <path> [-v]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
path	A comma-separated list of string	List of file paths		Yes	
-v, --verbose	Boolean	Showing fetch requests as they are submitted		No	Off

Fetching a directory containing many files using the CLI

In order to fetch a directory that contains a large number of files, it is recommended to use the `xargs` command in a similar manner, as follows:

```
find -L <directory path> -type f | xargs -r -n512 -P64 weka fs tier fetch -v
```

Note: The pre-fetching of files does not guarantee that they will reside on the SSD until they are accessed.

In order to ensure that the fetch is effective, the following must be taken into account:

- Free SSD Capacity: There has to be sufficient free SSD capacity to retain the filesystems that are to be fetched.
- Tiering Policy: The tiering policy may release some of the files back to the object store after they have been fetched, or even during the fetch if it takes longer than expected. The policy should be long enough to allow for the fetch to complete and the data to be accessed before it is released again.

Release API for data lifecycle management

How to release API for data lifecycle management.

Releasing files from SSD to an object store using the CLI

Using the manual release command, it is possible to clear SSD space in advance (for example, for shrinking one filesystem SSD capacity for a different filesystem without releasing important data, or for a job that needs more SSDs space from different files). The metadata will still remain on SSD for fast traversal over files and directories but the data will be marked for release and will be released to the object-store as soon as possible, and before any other files are scheduled to release due to other lifecycle policies.

Command

```
weka fs tier release [-v]
```

Use the following command to release files:

```
weka fs tier release <path>
```

Table 1 Parameters

Name	Type	Value	Limitations	Mandatory	Default
path	A comma-separated list of string	List of file paths		Yes	
-v, --verbose	Boolean	Showing release requests as they are submitted		No	Off

Releasing a directory containing many files using the CLI

In order to release a directory that contains a large number of files, it is recommended to use the **`xargs`** command in a similar manner, as follows:

```
# directory
find -L <directory path> -type f | xargs -r -n512 -P64 weka fs tier release

# similarly, a file containing a list of paths can be used
cat file-list | xargs -P32 -n200 weka fs tier release
```

Chapter 6: Managing clients

How to manage clients using the CLI.

Joining the cluster using the CLI

Command

`weka cluster host add`

Once the client host is in the stem mode (this is the mode defined immediately after running the **`install.sh`** command), use the following command line on the client host to add it to the cluster:

```
weka -H <backend-hostname> cluster host add <client-hostname>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
backend-hostname	String	IP/hostname of one of the existing backend instances in the cluster	Valid hostname (FQDN or IP)	Yes	
client-hostname	String	IP/hostname of the client currently being added	Valid hostname (FQDN or IP)	Yes	

Note: On completion of this stage, the host-ID of the newly added host will be received. Make a note of it for the next steps.

Configuring the host as client using the CLI

Command

`weka cluster host cores`

To configure the new host as a client, run the following command:

```
weka cluster host cores <host-id> <cores> --frontend-dedicated-cores=<fe_cores>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
host-id	String	Identifier of the host to be added to the cluster	Must be a valid host identifier	Yes	
cores	Number	Number of physical cores to be allocated to the Content Software for File client	Maximum 19 cores	Yes	
frontend-dedicated-cores	Number	Number of physical cores to be dedicated to FrontEnd processes	For clients, the number of total cores and frontenddedicated-cores must be equal	Yes, in order to configure a host as a client	

Configuring client networking using the CLI

Command

`weka cluster host net add`

Note: If the new client is to communicate with the Content Software for File system cluster over the kernel UDP stack, it is not necessary to run this command. If a high-performance client is required and the appropriate network NIC is available, use the following command to configure the networking interface used by the client to communicate with the Content Software for File system cluster hosts:

```
weka cluster host net add <host-id> <device> --ips=<ip-address> --netmask=<netmask> --
gateway=<gateway>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
host-id	String	Identifier of the host to be added to the cluster	Must be a valid host identifier	Yes	
device	String	Network interface device name for example, **`eth1`**	Must be a valid network device name	Yes	
ips	IP address	The IP address of the new interface	Must be a valid IP address	Yes	
gateway	IP address	The IP address of the default routing gateway	The gateway must reside within the same IP network of ip-address (as described by netmask). Not relevant for IB / L2 non-routable networks	No	
netmask	Number	Number of bits in the net mask, for example, the net mask of 255.255.0.0 has 16 netmask bits	Describes the number of bits that identify a network ID (also known as CIDR)	No	

Note: When configuring an InfiniBand client, do not pass the `--ips`, `--netmask`, and `--gateway` parameters.

Note: InfiniBand clients can only join a cluster with InfiniBand backends. It is not possible to mix InfiniBand and Ethernet clients/backends.

Applying the host configuration using the CLI

Command

`weka cluster host apply`

After successfully configuring the host and its network device, run the following command to finalize the configuration by activating the host:

```
weka cluster host apply <host-id> [--force]
```

Parameter

Name	Type	Value	Limitation	Mandatory	Default
hostid	Comma-separated string	Identifier of host to be added to the cluster	Must be a valid host identifier	Yes	
force	Boolean	Do not prompt for confirmation		No	Off

Chapter 7: Managing snapshots

How to manage snapshots using the CLI.

Viewing snapshots using the CLI

Command

`weka fs snapshot`

This command is used to display all snapshots of all filesystems in a single table.

Creating a snapshot using the CLI

Command

`weka fs snapshot create`

Use the following command line to add a snapshot:

```
weka fs snapshot create <file-system> <name> [<access-point>] [--source-
snap=<source>] [--is-writable]
```

Table 2 Parameters

Name	Type	Value	Limitations	Mandatory	Default
file-system	String	A valid filesystem identifier	Must be a valid name	Yes	
name	String	Unique name for filesystem snapshot	Must be a valid name	Yes	

Name	Type	Value	Limitations	Mandatory	Default
access-point	String	Name of newly-created directory for filesystem level snapshots, which will serve as the access point for the snapshots	Must be a valid name	No	Controlled by `weka fs snapshot access-point-naming-convention update-<date/name>`. By default it is `<date>` format: `@GMT_%Y.%m.%d-%H.%M.%S` which is compatible with previous versions of Windows.
source	String	Must be an existing snapshot	Must be a valid name	No	The snapshot name of the specified filesystem.
is_writable	Boolean	Sets the created snapshot to be writable		No	False

Deleting a snapshot using the CLI

Command

`weka fs snapshot delete`

Use the following command line to delete a snapshot:

```
weka fs snapshot delete <file-system> <name>
```

Table 3 Parameters

Name	Type	Value	Limitations	Mandatory	Default
file-system	String	A valid filesystem identifier	Must be a valid name	Yes	
name	String	Unique name for filesystem snapshot	Must be a valid name	Yes	

Restoring a filesystem or snapshot from another snapshot using the CLI

You can restore a filesystem or snapshot from another snapshot.

Command

`weka fs restore`

Use the following command line to restore a snapshot:

```
weka fs restore <file-system> <source-name>
```

Command

`weka fs snapshot copy`

Use the following command line to restore a snapshot to another snapshot:

```
weka fs snapshot copy <file-system> <source-name> <destination-name>
```

Table 4 Parameters

Name	Type	Value	Limitations	Mandatory	Default
file-system	String	A valid filesystem identifier	Must be a valid name	Yes	
source-name	String	Unique name for the source of the snapshot	Must be a valid name	Yes	
destination-name	String	Name of the destination to which the snapshot should be copied	Must be an existing snapshot	Yes	

Caution: When restoring a filesystem from a snapshot (or copying over an existing snapshot), the filesystem data and metadata are changed. Make sure IOs to the filesystem are stopped during this time.

Updating a snapshot using the CLI

Command

`weka fs snapshot update`

This command changes the snapshot attributes. Use the following command line to update an existing snapshot:

```
weka fs snapshot update <file-system> <name> [--new-name=<new-name>] [--iswritable] [--access-point=<access-point>]
```

Parameters

Name	String	Value	Limitations	Mandatory	Default
file-system	String	A valid filesystem identifier.	Must be a valid name	Yes	
name	String	Unique name for the updated snapshot	Must be a valid name.	Yes	
new-name	String	New name for the updated snapshot	Must be a valid name.	No	
is-writable	Boolean	Sets the snapshot to be writable		No	
access-point	String	Name of directory for snapshot, which will serve as the access point for the snapshot.	Must be a valid name.	No	

Uploading a snapshot using the CLI

Command

`weka fs snapshot upload`

Use the following command line to upload an existing snapshot:

```
weka fs snapshot upload <file-system> <snapshot>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
filesystem	String	Name of the filesystem		Yes	
snapshot	String	Name of snapshot to upload	Must be a snapshot of the `<file-system>` filesystem	Yes	

Creating a filesystem from a snapshot using the CLI

Command

`weka fs download`

Use the following command line to create a filesystem from an existing snapshot:

```
weka fs download <name> <group-name> <total-capacity> <ssd-capacity> <obs> <locator>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Name of the filesystem to be created		Yes	
group-name	String	Name of the filesystem group in which the new filesystem will be placed		Yes	
total-capacity	Capacity	Total capacity of the downloaded filesystem		Yes	
ssd-capacity	Capacity	SSD capacity of the downloaded filesystem		Yes	
obs	String	Object store name for tiering		Yes	

Name	Type	Value	Limitations	Mandatory	Default
locator	String	Object store locator obtained from a previously successful snapshot upload		Yes	
additional-obs	String	An additional objec-store name. In case te data to recover resides in two object stores (a second object-store attached to the filesystem, and the filesystem has not undergone full migration). This object-store will be attached in a `read-only` mode.	The snapshot locator must reside in the primary object-store supplied in the `obs` parameter	No	
snapshot-name	String	The downloaded snapshot name.		No	The uploaded snapshot name
access-point	String	The downloaded snapshot access point		No	The uploaded access point

The locator is either a locator saved previously for disaster scenarios, or can be obtained using the **`weka fs snapshot`** command on a system with a live filesystem with snapshots.

Note:

- Due to the bandwidth characteristics and potential costs when interacting with remote object stores it is not allowed to download a filesystem from a remote object-store bucket. If a snapshot on a local object-store bucket exists, it is advisable to use that one. Otherwise, follow the procedure in Recover from a remote snapshot (on page 69).
- For encrypted filesystem, when downloading the same KMS master-key should be used to decrypt the snapshot data. For more information about encryption, see KMS management in the *Content Software for File User Guide*.

Chapter 8: Managing SMB

The Content Software for File system has a number of CLI commands for setting up an SMB cluster over Content Software for File filesystems. Used for managing the cluster itself, they are all located under theweka smb cluster command. They define what Content Software for File hosts will participate in the SMB cluster, and what (if any) public IPs will be exposed by the SMB cluster.

Showing an SMB cluster using the CLI

Command

`weka smb cluster`

Use this command to view information about the SMB cluster managed by the Content Software for File system.

Showing an SMB domain configuration using the CLI

Command

`weka smb domain`

Use this command to view information about the SMB domain configuration.

Creating an SMB cluster using the CLI

Command

`weka smb cluster create`

Use the following command line to create a new SMB cluster to be managed by the Content Software for File system:

```
weka smb cluster create <name> <domain> [--samba-hosts samba-hosts]... [--smb-ips-pool smb-ips-pool]... [--smb-ips-range smb-ips-range]...
```

Table 5 Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	NetBIOS name for the SMB cluster	Must be a valid name (ASCII)	Yes	
domain	String	The domain which the SMB cluster is to join	Must be a valid name (ASCII)	Yes	
samba-hosts	Comma-separated strings	List of 3-8 Content Software for File system hosts to participate in the SMB cluster, based on the host IDs in Content Software for File	Must be valid host IDs	Yes	
smb-ips-pool	Comma-separated IP addresses	The public IPs used as floating IPs for the SMB cluster to serve the SMB over and thereby provide HA; should not be assigned to any host on the network	Must be valid IP addresses	No	
smb-ips-range	IP address range	The public IPs used as floating IPs for the SMB cluster to serve the SMB over and thereby provide HA; should not be assigned to any host on the network	Format: A.B.C.D-E for example, 10.10.0.1-100	No	
domain-netbios-name	String	Domain NetBIOS name	Must be a valid name (ASCII)	No	First part of `domain` parameter

Note: Content Software for File

- All IPs must reside on the same subnet, in order to enable HA through IP takeover.
- The IPs must be configured but MUST NOT be in use by any other application/host in the subnet, including Content Software for File system management nodes, Content Software for File system IO nodes, or Content Software for File system NFS floating IPs. In AWS environments, this is not supported and these IPs should not be provided.
- The `--smb-ips` parameter is supposed to accept the public IPs that the SMB cluster will expose. To mount the SMB cluster in an HA manner, they should be mounted via one of the exposed public IPs, thereby ensuring that they will not lose connection if one of the SMB hosts fails.
- If it is necessary to set global options to the SMB library, contact customer support.

For example:

```
weka smb cluster create wekaSMB mydomain --samba-hosts 0,1,2,3,4 --smb-ips-pool
1.1.1.1,1.1.1.2 --smb-ips-range 1.1.1.3-5
```

In this example of a full command, an SMB cluster is configured over the Content Software for File system hosts 0-4. The SMB cluster is called `wekaSMB`, the domain name is called `mydomain`, and is directed to use public IPs 1.1.1.1 to 1.1.1.5.

Checking status of SMB host readiness using the CLI

Command

`weka smb cluster status`

Use this command to check the status of the hosts which are part of the SMB cluster. Once all host are prepared and ready, it is possible to join an SMB cluster to an Active Directory.

Joining an SMB cluster to an Active Directory using the CLI

Command

`weka smb domain join`

Use the following command line to join an SMB domain to an Active Directory:

```
weka smb domain join <username> <password>
```

Parameters

Name	Type	Value	Limitation	Mandatory	Default
username	String	Name of a user with permissions to add a machine to the domain	Must be a valid name (ASCII)	Yes	
password	String	The password of the user	Must be a valid password (ASCII)	Yes	

In order to join another Active Directory to the current SMB cluster configuration, it is necessary to leave the current Active Directory. This is performed using the following command line:

```
weka smb domain leave <username> <password>
```

On completion of this operation, it is possible to join another Active Directory to the SMB cluster.

Note: To configure a new SMB cluster, the current SMB cluster has to be deleted.

Deleting an SMB cluster using the CLI

Command

`weka smb cluster destroy`

Use this command to destroy an SMB cluster managed by the Content Software for File system. Deleting an existing SMB cluster managed by the system does not delete the backend Content Software for File filesystems, but removes the SMB share exposures of these filesystems.

Note: Editing an existing cluster is not supported. Consequently, to change an SMB cluster configuration, the cluster has to be deleted and recreated.

Configuring trusted domains using the CLI

How to configure trusted domains using the CLI.

Listing trusted domains using the CLI

Command

`weka smb cluster trusted-domains`

Use this command to list all the configured trusted domains and their ID ranges.

Adding trusted domains using the CLI

Command

`weka smb cluster trusted-domains add`

Use the following command line to add an SMB trusted domain:

```
weka smb cluster trusted-domains add <domain-name> <from-id> <to-id>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
domain-name	String	The name of the domain being added	Must be a valid name (ASCII)	Yes	
from-id	Number	The first ID of the range for the domain ID mapping	The range cannot overlap with other domains	Yes	
to-id	Number	The last ID of the range for the domain ID mapping	The range cannot overlap with other domains	Yes	

Removing trusted domains using the CLI

Command

`weka smb cluster trusted-domains remove`

Use the following command line to remove an SMB trusted domain:

```
weka smb cluster trusted-domains remove <domain-id>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
domain-id	Number	The internal ID of the domain to remove		Yes	

Listing SMB shares using the CLI

Command

`weka smb share`

Use this command to list all existing SMB shares.

Adding SMB shares using the CLI

Command

`weka smb share add`

Use the following command line to add a new share to be exposed to SMB:

```
smb share add <share-name> <fs-name> [--description description] [--internal-path
internal-path] [file-create-mask] [directory-create-mask]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
share-name	String	Name of the share being added	Must be a valid name (ASCII)	Yes	
fs-name	String	Name of the filesystem to share	Must be a valid name. A filesystem set with required authentication cannot be used for SMB share.	Yes	
description	String	Description of what the share will receive when viewed remotely	Must be a valid string	No	No description

Name	Type	Value	Limitations	Mandatory	Default
internal-path	String	The internal path within the filesystem (relative to its root) which will be exposed	Must be a valid path	No	
file-create-mask	String	POSIX permissions for the file created through the SMB share	Numeric (octal) notation	No	0744
directory-create-mask	String	POSIX permissions for directories created through the SMB share	Numeric (octal) notation	No	0755

Note: If it is necessary to set share specific options to the SMB library, contact customer support.

For example: The following is an example for adding users to a share mounted on a filesystem named "default":

```
weka smb share add rootShare default weka smb share add internalShare default --
internal-path some/dir --description "Exposed share"
```

In this example, the first SMB share added has the Content Software for File system share for default. The second SMB share has internal for default.

Updating SMB shares using the CLI

Command

`weka smb share update`

Use the following command line to update an existing share:

```
weka smb share update <share-id> [--encryption encryption]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
share-id	Number	The ID of the share to be updated	Must be a valid share ID	Yes	
encryption	String	The share encryption policy. desired - turns on data encryption for this share for clients that support encryption if negotiation has been enabled globally. required - enforces encryption for the shares. Clients that do not support encryption will be denied access to the share. If the global option is set to disabled access will be denied to these shares for all clients	cluster_default desired or required	No	

Controlling SMB shares users lists using the CLI

How to control SMB shares users lists using the CLI.

Showing SMB share user lists using the CLI

Command

```
weka smb share lists show
```

Use this command to view the various user-list settings:

Adding SMB share user to list using the CLI

Command

```
weka smb share lists add
```

Use the following command line to add users to a share user-list:

```
weka smb share lists add <share-id> <user-list-type> <--users users>...
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
share-id	Number	The ID of the share to be updated	Must be a valid share ID	Yes	
user-list-type	String	The type of permissions list for users	`read_only` - list of users that will not be given write access to the share, regardless of the read-only setting. `read_write` - list of users that will be given write access to the share, regardless of the read-only setting. `valid` - list of users that are allowed to log-in to this share SMB service (empty list - all users are allowed) `invalid` - list of users that are not allowed to log-in to this share SMB service	Yes	
users	A comma-separated list of strings	A list of users to add to the `user-list-type` list. Can use the `@` notation to allow groups of users, for example `root, Jack, @domain\admins`	Up to 8 users/groups for all lists combined per share	Yes	

Removing SMB share user from lists using the CLI

Command

`weka smb share lists remove`

Use the following command line to remove users from a share user-list:

```
weka smb share lists remove <share-id> <user-list-type> <--users users>...
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
share-id	Number	The ID of the share to be updated	Must be a valid share ID	Yes	
user-list-type	String	The type of permissions list for users	`read_only` - list of users that will not be given write access to the share, regardless of the read-only setting. `read_write` - list of users that will be given write access to the share, regardless of the `read-only` setting. `valid` - list of users that are allowed to log-in to this share SMB service (empty list - all users are allowed) `invalid` - list of users that are not allowed to login to this share SMB service	Yes	
users	A commas eparated list of Strings	A list of users to remove from the user-list-type list. Can use the `@` notation to allow groups of users, for example, `root, Jack, @domain \admins`	Up to 8 users/groups for all lists combined per share	Yes	

Resetting SMB share user lists using the CLI

Command

`weka smb share lists reset`

Use the following command line to remove all users from a share user-list:

```
weka smb share lists reset <share-id> <user-list-type>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
share-id	Number	The ID of the share to be updated	Must be a valid share ID	Yes	
user-list-type	String	The type of permissions list to reset	`read_only` - list of users that will not be given write access to the share, regardless of the `read-only` setting. `read_write` - list of users that will be given write access to the share, regardless of the read-only setting. `valid` - list of users that are allowed to log-in to this share SMB service (empty list - all users are allowed) `invalid` - list of users that are not allowed to log-in to this share SMB service	Yes	

Removing SMB shares using the CLI

Command

`weka smb share remove`

Use the following command line to remove a share exposed to SMB:

```
weka smb share remove <share-id>
```

Parameters

Name	Type	Value	Limitation	Mandatory	Default
share-id	String	The ID of the share to be removed	Must be a valid share ID	Yes	

For example: The following is an example for removing an SMB share defined as ID 1:

```
weka smb share remove 1
```

Chapter 9: Managing NFS

How to manage NFS networking configuration (interface groups) and access control (client access groups) using the CLI.

User groups resolution

The NFS protocol, using AUTH_SYS protocol, has a limitation of 16 security groups users can be part of. The protocol truncates the group list to 16 if a user is part of more than 16 groups, and a permissions check can fail for authorized users.

As in many cases, a user can be part of more than 16 security groups. It is possible to configure the Weka system to ignore the groups passed by the NFS protocol and resolve the user's groups external to the protocol. For that, several steps should be taken:

1. Define an interface group that supports external group-IDs resolution (`allow-manage-gids` option).
2. Define the NFS client permissions to use external group-IDs resolution (`manage-gids` option).
3. Set up the relevant hosts to retrieve the user's group-IDs information.

Set up the hosts to retrieve user's group-IDs information

For the hosts that are part of the interface group, you can set the host to retrieve the user's group-IDs information in any method that is part of the environment.

You can also set the group resolution by joining the AD domain, the Kerberos domain, or using LDAP with a read-only user.

Configure the sssd on the host to serve as a group IDs provider. For example, you can configure the `sssd` directly using LDAP, or as a proxy to a different nss group IDs provider.

Example: set sssd directly for nss services using LDAP with a read-only user

```
[sssd]
services = nss
config_file_version = 2
ldap_search_base = dc=example,dc=com

# The DN used to search the ldap directory with.
ldap_default_bind_dn = cn=ro_admin,ou=groups,dc=example,dc=com

# The password of the bind DN.
ldap_default_authtok = password
```

If you use another method than the sssd, but with a different provider, configure an sssd proxy on each relevant host. The proxy is used for the Content Software for File container to resolve the groups by any method defined on the host.

To configure sssd proxy on a host, use the following:

```
# install sssd
yum install sssd

# set up a proxy for weka in /etc/sssd/sssd.conf
[sssd]
services = nss
config_file_version = 2
domains = proxy_for_weka

[nss]
[domain/proxy_for_weka]
id_provider = proxy
auth_provider = none

# the name of the nss lib to be proxied, e.g. ldap, nis, winbind, vas4, etc.
proxy_lib_name = ldap
```

Note: All users must be present and resolved in the method used in the `sssd` for the groups resolution. In the above example, using an LDAP-only provider, local users (such as a local root) that are not present in LDAP do not receive their groups resolved and they are denied. For such users or applications, add the LDAP user.

Supported NFS client mount options

This section describes the supported mount options for NFS clients.

Non-coherent mount options

- `ac`
- `async`
- `noatime`
- `lookupcache=all`

Coherent mount options

- `noac`
- `sync`
- `atime`
- `lookupcache=none`

Common mount options

Note: The following options can be changed. These values are commonly used with the Content Software for File system:

- `rw`
- `hard`
- `rsize=524288`
- `wsize=524288`
- `namlen=255`
- `timeo=600`
- `retrans=2`

Fixed mount options

Note: Make sure to set these values on the mount command, as different values are not supported, and the server cannot enforce it.

- `nolock`

Note: The following options should have fixed values, but usually are either the NFS mount defaults or will be negotiated to these values by the protocol.

- `sec=sys`
- `proto=tcp`
- `mountproto=tcp`

Manage NFS networking using the CLI

Creating interface groups using the CLI

Command

`weka nfs interface-group add`

Use the following command line to add an interface group:

```
weka nfs interface-group add <name> <type> [--subnet subnet] [--gateway gateway] [--
allow-manage-gids allow-manage-gids]
```

Parameters

Name	Type	Value	Limitation	Mandatory	Default
name	String	Unique interface group name	Up to 11 characters length	Yes	
type	String	Group type	Can only be NFS	Yes	
subnet	String	The subnet mask in the 255.255.0.0 format	Valid netmask	No	255.255.255.255
gateway	String	Gateway IP	Valid IP	No	255.255.255.255
allow-manage-gids	String	Allows the hosts within this interface group to use `manage-gids` when set in exports. With `manage-gids`, the list of group IDs received from the client will be replaced by a list of group IDs determined by an appropriate lookup on the server.	`on` or `off` Cannot be set if one of the hosts belongs to an interface group which does not have the `allow-manage-gids` flag set.	No	`on`

Note: [Review Note: This note is replaced by the two bulleted notes below for v4.0.5.19] Each host can be set to be part of interface groups with the same value of `allow-manage-gids. In` addition, you must not mount the same filesystem by the hosts residing in interface groups with different values of `allow-manage-gids.`

Note:

- Do not mount the same filesystem by containers residing in interface groups with different values of the `allow-manage-gids.
- As a best practice, it is recommended to have only one of the following protocol containers, NFS or SMB installed on the same server.

  [Review note: Left out this sentence bc we are not releasing 4.2 yet] Starting from version 4.2, setting more than one additional protocol to the existing POSIX is not allowed.

Setting interface group ports using the CLI

Command

`weka nfs interface-group port` and `weka nfs interface-group port delete`

Use the following command lines to add or delete an interface group port:

```
weka nfs interface-group port add <name> <host-id> <port>
```

```
weka nfs interface-group port delete <name> <host-id> <port>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Interface group name	None	Yes	
host-id	String	Host ID on which the port resides (can be obtained by running the `weka cluster host` command)	Valid host ID	Yes	
port	String	Port's device, for example, eth1	Valid device	Yes	

Setting interface group IPs using the CLI

Command

`weka nfs interface-group ip-range add`

and

`weka nfs interface-group ip-range delete`

Use the following command lines to add or delete an interface group IP:

```
weka nfs interface-group ip-range add <name> <ips>
```

```
weka nfs interface-group ip-range delete <name> <ips>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Interface group name	None	Yes	
ips	String	IP range	Valid IP range	Yes	

Configuring the service mountd port

The mountd service receives requests from clients to mount to the NFS server. When working with interface groups (with `allow-manage-gids=on`), it is possible to set it explicitly, rather than have it randomly selected on each server startup. This allows an easier setup of the firewalls to allow that port.

Use the following command to set and view the mountd configuration: `weka nfs global-config set --mountd-port <mountd-port>` and `weka nfs global-config show`.

Manage NFS access (client access groups) using the CLI

Defining client access groups using the CLI

Command

`weka nfs client-group`

Use the following command lines to add or delete a client access group:

```
weka nfs client-group add <name>
```

```
weka nfs client-group delete <name>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Group name	Valid name	Yes	

Managing client access groups using the CLI

Adding or deleting DNS

Command

`weka nfs rules`

Use the following command lines to add or delete a client group DNS:

```
weka nfs rules add dns <name> <dns>
```

```
weka nfs rules delete dns <name> <dns>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Group name	Valid name	Yes	
dns	String	DNS rule with *?[] wildcard rules		Yes	

Adding or deleting an IP using the CLI

Command

`weka nfs rules`

Use the following command lines to add or delete a client group IP:

```
weka nfs rules add ip <name> <ip>
weka nfs rules delete ip <name> <ip>
```

Parameters

Name	Type	Value	Limitation	Mandatory	Default
name	String	Group name	Valid name	Yes	
ip	String	IP with netmask rule, in the 1.1.1.1/255.255.0.0 format	Valid IP	Yes	

Managing NFS client permissions using the CLI

Command

`weka nfs permission`

Use the following command lines to add, update, or delete NFS permissions:

```
weka nfs permission add <filesystem> <group> [--path path] [--permission-type
permission-type] [--root-squashing root-squashing] [--anon-uid anon-uid] [--anon-gid
anon-gid] [--obs_direct]
```

```
weka nfs permission update <filesystem> <group> [--path path] [--
permissiontypepermission-type] [--root-squashing root-squashing] [--non-uid anon-uid]
[--anon-gid anon-gid] weka nfs permission delete <filesystem> <group> [--path path]
```

```
weka nfs permission delete <filesystem> <group> [--path path]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
filesystem	String	Filesystem name	Existing filesystem. A filesystem set with required authentication cannot be used for NFS export.	Yes	
group	String	Client group name	Existing client group	Yes	
path	String	The root of the share	Valid path	No	/
permission-type	String	Permission type	ro for readonly orrw for readwrite	No	RW
squash	String	Squashing type	none, root or all (all is supported only when working on hosts with intrface-groups set with allow-manage-gids, otherwise it is treated as root)	No	On

Name	Type	Value	Limitations	Mandatory	Default
anon-uid	Number	Anonymous user ID (relevant only for root squashing)	Valid UID (between 1 and 65535)	Yes (if root squashing is enabled)	65534
anon-gid	Number	Anonymous user group ID (relevant only for root squashing)	Valid GID (between 1 and 65535)	Yes (if root squashing is enabled)	65534
obs-direct	Boolean	See Object-store Direct Mount section	on or off	No	No
manage-gids	String	Sets external group IDs resolution. The list of group IDs received from the client will be replaced by a list of group IDs determined by an appropriate lookup on the server.	on or off Relevant only when usingallow-manage-gids interface groups.	No	Off
privileged-port	String	Sets the share to only be mounted via privileged ports (1-1024), usually only allowed by the root user.	on or off. Relevant only when usingallow-manage-gids interface groups.	No	pff
supported-versions	String	A comma-separated list of supported NFS versions.	v3, v4	No	v3

Chapter 10: Managing alerts

How to manage alerts using the CLI. For a list of alerts, see List of alerts in the *Content Software for File User Guide*.

Displaying alert types using the CLI

Command

`weka alerts types`

Use this command to lists all possible types of alerts that can be returned from the Content Software for File cluster.

Describing alerts using the CLI

Command

`weka alerts describe`

Use this command to describe all the alert types that might be returned from the Content Software for File cluster along with possible action items for each alert.

Viewing alerts using the CLI

Command

`weka alerts`

Use the following command line to list all alerts (muted and unmuted) in the Content Software for File cluster:

```
weka alerts [--muted]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
muted	Boolean	List muted alerts alongside the unmuted ones		No	False

Muting alerts using the CLI

Command

`weka alerts mute`

Use the following command line to mute an alert-type:

```
weka alerts mute <alert-type> <duration>
```

Muted alerts will not be prompted when listing active alerts. Alerts cannot be suppressed indefinitely, so a duration for the muted period must be provided. After expiry of the muted period, the alert-type is automatically unmuted.

Parameters

Name	Type	Value	Limitations	Mandatory	Default
alert-type	String	An alert-type to mute, use weka alerts types to list types		Yes	
duration	String	How long to mute this alert type for	Format: 3s, 2h, 4m, 1d, 1d5h, 1w	Yes	

Unmuting alerts using the CLI

Command

`weka alerts unmute`

Use the following command line to unmute a previously-muted alert-type:

```
weka alerts unmute <alert-type>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
alert-type	String	An alert-type to unmute, use `weka alerts types` to list types. For a list of alerts, see List of alerts in the *Content Software for File User Guide*.	Yes	Yes	

Chapter 11: Managing events

How to manage events using the CLI. For a list of events, see List of events in the *Content Software for File User Guide*.

Viewing events using the CLI

Command

`weka events`

Use the following command line to list events in the Content Software for File cluster:

```
weka events [--num-results num-results] [--start-time <start-time>] [--end-time <end-time>] [--severity severity] [--direction direction] [--fetch-order fetch-order] [--type-list type-list] [--exclude-type-list exclude-type-list] [--category-list category-list] [--cloud-time] [--show-internal] [--raw-units] [--UTC]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
num-results	Integer	Maximum number of events to display	Positive integer. 0 shows all events	No	50
start-time	String	Include events that occurred at this start time and later	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12-31T10:00, 9:15Z, 10:00+2:00	No	-365 days
end-time	String	Include events that occurred up to this time	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12-31T10:00, 9:15Z, 10:00+2:00	No	Set to a time represents 'now'

Name	Type	Value	Limitations	Mandatory	Default
severity	String	Include events with this level of severity and higher	'info', 'warning', 'minor', 'major' or 'critical'	No	INFO
direction	String	Sort events by ascending or descending time	'asc' or 'dsc'	No	asc
fetch-order	String	Fetch from end-time backwards or from start-time forwards	'fw' or 'bw'	No	bw
type-list	String	Filter events by type (can be used multiple times)	Use `weka events list-types` to see available types	No	None
exclude-type-list	String	Filter-out events by type (can be used multiple times)	Use `weka events list-types` to see available types		
category-list	String	Include only events matching the defined category	Categories can be Alerts, Cloud, Clustering, Drive, Events, Filesystem, IO, InterfaceGroup, Licensing, NFS, Network, Node, ObjectStorage, Raid, Statistics, System, Upgrade, and User	No	All
cloud-time	Boolean	Query and sort results by the digested time in the cloud		No	False
show-internal	Boolean	Also displays internal events		No	False
raw-units	Boolean	Print values in raw units (bytes, seconds, among others)		No	Human-readable format, for example, 1KiB 234MiB 2GiB

Name	Type	Value	Limitations	Mandatory	Default
UTC	Boolean	Print times in UTC		No	Host's local time

Listing local events using the CLI

Command

`weka events list-local`

Use the following command line to list recent events on the specific host running the command from.

This command is helpful for the following cases:

- No connectivity to the central monitoring site
- No connectivity from a specific host
- Hosts that are not part of the cluster

```
weka events list-local [--start-time <start-time>] [--end-time <end-time>] [--next
next] [--stem-mode] [--show-internal] [--raw-units] [--UTC]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
start-time	String	Include events that occurred at this start time and later	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12- 31T10:00, 9:15Z, 10:00+2:00	no	-365 days
end-time	String	Include events that occurred up to this time	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12- 31T10:00, 9:15Z, 10:00+2:00	No	Set to a time represents 'now'

Name	Type	Value	Limitations	Mandatory	Default
next	String	Identifier to the next page of events	As returned in the previous call to `weka events list-local`	No	
stem-mode	Boolean	Displays events when the host has not been attached to the cluster		No	False
show-internal	Boolean	Also displays internal events		No	False
raw-units	Boolean	Print values in raw units (bytes, seconds, among others.)		No	Human-readable format, for example, 1KiB 234MiB 2GiB
UTC	Boolean	Print times in UTC		No	Host's local time

Triggering a custom event using the CLI

Command

`weka events trigger-event`

It can be useful to mark specific activities, maintenance work, or important changes/new usage of the system, and see that as part of the system events timeline.

To trigger a custom event, use:

```
weka events trigger-event <text>
```

Chapter 12: Managing statistics

How to manage statistics available in the Content Software for File system using the CLI. For a list of statistics, see List of statistics in the *Content Software for File User Guide*.

Listing statistic types using the CLI

Command

`weka stats list-types`

Use the following command line to obtain statistics definition information:

```
weka stats list-types [<name-or-category>] [--show-internal]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name-or-category	String	Name or category to filter by	Valid name or category	No	
show-internal	Boolean	Also displays internal statistics		No	False

Viewing statistics in realtime using the CLI

Command

`weka stats realtime`

Use the following command line to obtain the current performance-related statistics of the hosts, in a one-second interval:

```
weka stats realtime [<node-ids>] [--raw-units] [--UTC]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
node-ids	Comma-separated strings	Only show realtime stats of these nodes		No	
raw-units	Boolean	Print values in raw units (bytes, seconds, among others)		No	Humanreadable format, for example, 1KiB 234MiB 2GiB
UTC	Boolean	Print times in UTC		No	Host's local time

Viewing statistics over time using the CLI

Command

`weka stats`

The collected statistics can help analyze system performance and determine the source of issues that may occur during Content Software for File system runs. Statistics are divided according to categories. When selecting a category, a list of the possible statistics is displayed, from which you can select the specific statistics.

Note: Content Software for File averages all statistics over one-second intervals. Consequently, the total value or other aggregates relate to a specific minute.

Use the following command line to manage filters and read statistics:

```
weka stats [--start-time <start-time>] [--end-time <end-time>] [--interval interval]
[--resolution-secs resolution-secs] [--category category][--stat stat] [--node-ids
node-ids] [--param param] [--accumulated] [--per-node] [--no-zeros] [--show-internal]
[--raw-units] [--UTC]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
start-time	String	Start time of the reported period	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12-31T10:00, 9:15Z, 10:00+2:00	Yes	
end-time	String	End time of the reported period	Format: 5m, -5m, -1d, -1w, 1:00, 01:00, 18:30, 18:30:07, 2018-12-31 10:00, 2018/12/31 10:00, 2018-12-31T10:00, 9:15Z, 10:00+2:00	No	Current time
interval	String	Period of time to be reported	Valid interval in seconds (positive integer number)	Yes	
resolution-secs	String	Length of each interval in the reported period	Must be multiples of 60 seconds	No	60
category	String	Specific categories for retrieval of appropriate statistics	Valid existing categories: CPU, Object Store, Operations, Operations (NFS), Operations (Driver), and SSD	No	All
stat	String	Statistics names	Valid statistics names	No	All
node-ids	String	Node id	Valid node-id	No	All

Name	Type	Value	Limitations	Mandatory	Default
param	String	For parameterized statistics, retrieve only the instantiations where the specified parameter is of the specified value. Multiple values can be supplied for the same key, for example, '-- param method:putBlocks -- param method:initBlock'	Format: key:val	No	
accumulated	Boolean	Display accumulated statistics, not rate statistics		No	False
per-node	Boolean	Does not aggregate statistics across nodes		No	False
no-zeros	Boolean	Filters results where the value is 0		No	False
show-internal	Boolean	Also displays internal statistics		No	False
raw-units	Boolean	Print values in raw units (bytes, seconds, among others.)		No	Humanreadable format, for example 1KiB 234MiB 2GiB
UTC	Boolean	Print times in UTC		No	Host's local time

Setting statistic retention using the CLI

Command

```
weka stats retention set
```

Use the following command line to set the statistics retention period:

```
weka stats retention set <--days days> [--dry-run]
```

Parameters

Name	Type	Value	Limitation	Mandatory	Default
days	Number	The number of days to keep the statistics.	Should have enough free disk space per server.	Yes	
dry-run	Boolean	Only test the required capacity per the retention period.		No	

Use **`weka stats retention status`** to view the current retention and **`weka stats retention restore-default`** to restore the default retention settings.

Chapter 13: Security management

This page describes important security consideration for the Content Software for File cluster management.

The Content Software for File system is a secured environment. It deploys a combination of security controls to ensure secured communication and secured user data.

The security controls include the following:

- HTTPS access: To access the Weka GUI, you connect only to one of the system servers using HTTPS through port 14000.
- Authentication tokens: The authentication tokens are used for accessing the Weka system API and to allow the mounting of secure filesystems.
- KMS: When creating an encrypted filesystem, a KMS must be used to properly secure the encryption keys. The KMS encrypts and decrypts filesystem keys.
- TLS certificates: By default, the system deploys a self-signed certificate to access the GUI, CLI, and API through HTTPS. You can deploy your certificate by providing an unencrypted private key and certificate PEM files.
- CA certificates: The system uses well-known CA certificates to establish trust with external services. For example, when using a KMS.
- Account lockout: To prevent brute force attacks, if several login attempts fail (default: 5), the user account is locked for several minutes (default: 2 minutes).
- Login banner: The login banner provides a security statement or a legal message displayed on the sign-in page.
- GUI session automatic termination: The user is signed out after 30 minutes of inactivity.

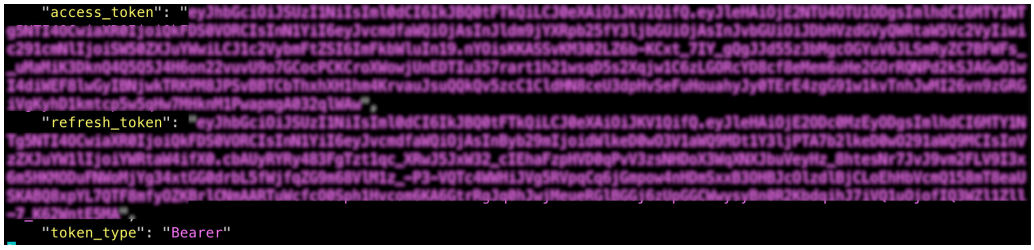
Obtaining authentication tokens

The authentication tokens include two types: an access token and a refresh token.

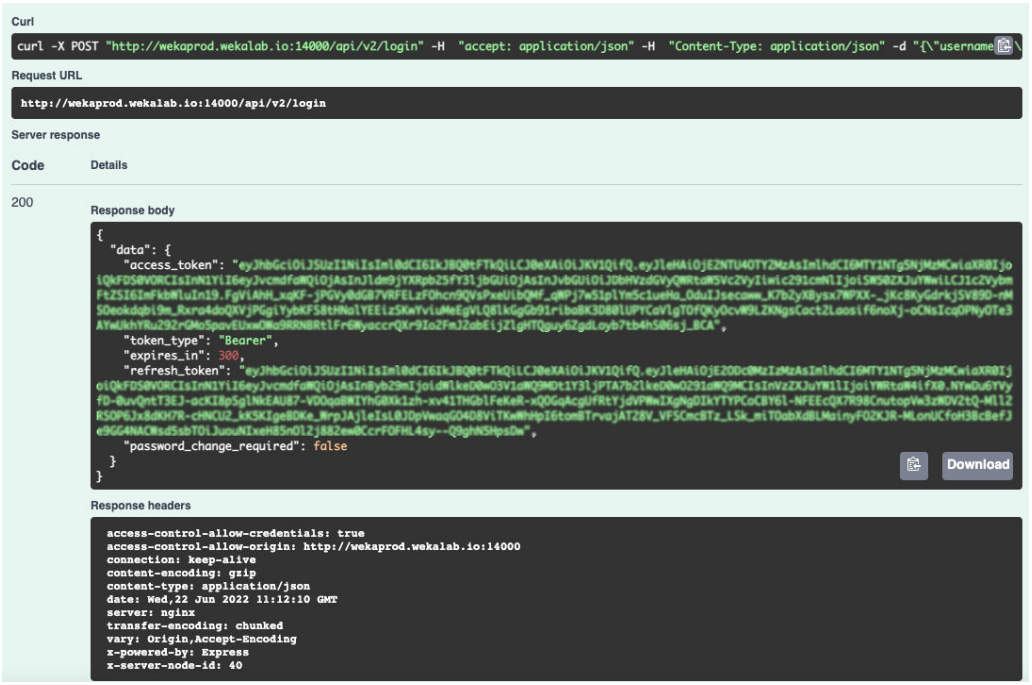
- Access token: The access token is a short-live token (five minutes) used for accessing the Weka system API and to allow the mounting of secure filesystems.
- Refresh token: The refresh token is a long-live token (one year) used for obtaining an additional access token.

Procedure

1. Do one of the following:
   - To obtain the refresh token and access token, through the CLI, log in to the system using the command: `weka user login`. The system creates an authentication token file and saves it in: `~/.weka/auth-token.json`. The token file contains both the access token and refresh token.



   - To obtain the refresh token and access token, through the REST API, use the `POST /login`. The API returns the token in the response body.



KMS management using the CLI

Describes the management of the Key Management System (KMS) within the Content Software for File system using the CLI.

Adding or updating a KMS using the CLI

Command

`weka security kms set`

Use the following command line to add or update the Vault KMS configuration in the Content Software for File system:

```
weka security kms set <type> <address> <key-identifier> [--token token] [--client-cert client-cert] [--client-key client-key] [--ca-cert ca-cert]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
type	String	Type of the KMS	Either `vault` or `kmip`	Yes	
address	String	KMS server address	`URL` for Vault, `hostname:port` for KMIP	Yes	
key-identifier	String	Key to be used for encryptionas-aservice in the KMS	Key name (for Vault) or a key UID (for KMIP)	Yes	
token	String	API token to access Vault KMS	Must have: Read permissions to `transit/keys/<master-key-name>` write permissions to `transit/encrypt/<master-keyname>` and `transit/decrypt/<masterkeyname>` permissions to `/transit/rewrap and auth/token/lookup`	Must be supplied for Vault and must not be supplied for `kmip`	
client-cert	String	Path to the client certificate PEM file	Must permit encrypt and decrypt permissions	Must be supplied for `kmip` and must not be supplied for Vault	

Name	Type	Value	Limitations	Mandatory	Default
client-key	String	Path to the client key PEM file		Must be supplied for `kmip` and must not be supplied for `vault`	
ca-cert	String	Path to the CA certificate PEM file		Optional for `kmip` and must not be supplied for `vault`	

Note: For the add or update command to succeed, the KMS should be preconfigured and available with the key and a valid token.

For example: Setting the Content Software for File system with a Vault KMS:

```
weka security kms set vault https://vault-dns:8200 weka-key --token
s.nRucA9Gtb3yNVmLUK221234
```

Setting the Content Software for File system with a KMIP complaint KMS (for example, SmartKey):

```
weka security kms set kmip amer.smartkey.io:5696 b2f81234-c0f6-4d63-b5b3-84a82e231234
--client-cert smartkey_cert.pem --client-key smartkey_key.pem
```

Viewing the KMS using the CLI

Command

`weka security kms`

Use this command to show the details of the configured KMS.

Removing the KMS using the CLI

Command

`weka security kms unset`

Use this command to remove the KMS from the Content Software for File system. It is only possible to remove a KMS configuration if no encrypted filesystems exist.

Note: To force remove a KMS even if encrypted filesystems exist, use the `--allow-downgrade` attribute. In such cases, the encrypted filesystem keys are re-encrypted with local encryption and may be compromised.

Re-wrapping filesystem keys using the CLI

Command

`weka security kms rewrap`

If the KMS key is compromised or requires rotation, the KMS admin can rotate the key in the KMS. In such cases, this command is used to re-encrypt the encrypted filesystem keys with the new KMS master key.

```
weka security kms rewrap [--new-key-uid new-key-uid]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
new-key-uid	String	Unique identifier for the new key to be used to wrap filesystem keys		Must be supplied for kmip and must not be supplied for Vault	

Note: Existing filesystem keys that are part of the Snap-To-Object feature will not be automatically re-encrypted with the new KMS key.

Note: Unlike in Vault KMS, re-wrapping a KMIP-based KMS requires generating a new key in the KMS, rather than rotating the same key. Hence, the old key should be preserved in the KMS in order to be able to decrypt old Snap2Obj snapshots.

Setting-up Vault configuration

The setting up of the Vault configuration is described.

Enabling transit secret engine in vault

The Weka system uses encryption-as-a-service capabilities of the KMS to encrypt/decrypt the filesystem keys. This requires the configuration of Vault with the `transit secret engine`.

```
$ vault secrets enable transit
Success! Enabled the transit secrets engine at: transit/
```

For more information, see Vault transit secret-engine documentation.

Setting-up the master key for the Content Software for File system

Once the transit secret engine is set up, a master key for use with the Content Software for File system must be created.

```
$ vault write -f transit/keys/weka-key
Success! Data written to: transit/keys/weka-key
```

Note: It is possible to either create a different key for each Content Software for File cluster or to share the key between different clusters.

For more information, refer to Vault transit secret-engine documentation.

Creating a policy for master key permissions

Create a `wekaio_policy.hcl` file with the following content:

```
path "transit/+/weka-key" {
  capabilities = ["read", "create", "update"]
}
path "transit/keys/weka-key" {
  capabilities = ["read"]}
```

This limits the capabilities so there is no permission to destroy the key, using this policy. This protection is important when creating an API token.

Create the policy using the following command:

```
$ vault policy write weka weka_policy.hcl
```

Obtaining an API token from Vault

Authentication from the Content Software for File system to Vault relies on an API token. Since the Content Software for File system must always be able to communicate with the KMS, a periodic service token must be used. To obtain the token, follow the next steps:

- Verify that the `token` authentication method in Vault is enabled. This can be performed using the following command:

  ```
  $ vault auth list

  Path         Type        Description
  ----         ----        -----------
  token/       token       token based credentials
  ```

- To enable the token authentication method use the following command:

  ```
  $ vault auth enable token
  ```

- Log into the KMS system using any of the identity methods supported by Vault. The identity should have permission to use the previously-set master key.

- Create a token role for the identity using the following command:

```
$ vault write auth/token/roles/weka allowed_policies="weka" period="768h"
```

> **Note:** he `period` is the time set for a renewal request. If no renewal is requested during this time period, the token will be revoked and a new token must be retrieved from Vault and set in the Weka system.

- Generate a token for the logged-in identity using the following command:

```
$ vault token create -role=weka

Key                  Value
---                  -----
token                s.nRucA9Gtb3yNVmLUK221234
token_accessor       4Nm9BvIVS4HWCgLATc3r1234
token_duration       768h
token_renewable      true
token_policies       ["default"]
identity_policies    []
policies             ["default"]
```

For more information on obtaining an API token, see Vault Tokens documentation.

> **Note:** The Content Software for File system does not automatically renew the API token lease. It can be renewed using the Vault CLI/API. It is also possible to define a higher maximum token value (`max_lease_ttl)` by changing the Vault Configuration file.

Obtaining a certificate for a KMIP-based KMS

The method for obtaining a client certificate and key and set it using the KMS is different for each KMS. The certificate itself will be generated using OpenSSL, with some UID obtained from the KMS, for example:

```
openssl req -x509 -newkey rsa:4096 -keyout client-key.pem -out client-cert.pem -days
365 -nodes -subj '/CN=f283c99b-f173-4371-babc-572961161234'
```

Consult the specific KMS documentation to create a certificate and link it to the Content Software for File cluster in the KMS with sufficient privileges (encrypt/decrypt).

See SmartKey KMS to create a client certificate and key, and to assign a certificate for Content Software for File within SmartKey.

Getting started with REST API

The Content Software for File system supports a RESTful API. This is useful when automating the interaction with the Content Software for File system and when integrating it into your workflows or monitoring systems.

The API is accessible at port 14000, via the `/api/v2` URL, you can explore it via `/api/v2/docs` when accessing from the cluster (e.g. `https://weka01:14000/api/v2/docs`).

Our static API documentation can be accessed from api.docs.weka.io (the version can be selected from the drop-down list). The .json file can also be used to create your client code, using an OpenAPI client generator.

Obtain an access token

You must provide an access token to use the Content Software for File REST API.

To obtain access/refresh tokens via the CLI, refer to Obtaining an Authentication Token section (there you can also generate an access token with a longer expiry time). To obtain access/refresh tokens via the API, you can call the **`login`** API, providing it a `username` and `password`.

If you already obtained a refresh token, you can use the **`login/refresh`** API to refresh the access token.

{% tabs %} {% tab title="Login" %} {% code title="Python example calling the login API" %}

```
import requests

url = "https://weka01:14000/api/v2/login"

payload="{\n    \"username\": \"admin\",\n    \"password\": \"admin\"\n}"
headers = {
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

{% endcode %} {% endtab %}

{% tab title="Refresh" %} {% code title="Python example calling the login refresh API" %}

```
import requests

url = "https://weka01:14000/api/v2/login/refresh"

payload="{\n    \"refresh_token\": \"REPLACE-WITH-REFRESH-TOKEN\"\n}"
headers = {
  'Content-Type': 'application/json'
}

response = requests.request("POST", url, headers=headers, data=payload)

print(response.text)
```

{% endcode %} {% endtab %} {% endtabs %}

In response, you will get an access token (valid for 5 minutes), that can be used in the other APIs that require token authentication, along with the refresh token (valid for 1 year), for getting additional access tokens without using the username/password.

{% code title="Login/Refresh Response" %}

```
{
  "data": [
    {
      "access_token": "ACCESS-TOKEN",
      "token_type": "Bearer",
      "expires_in": 300,
      "refresh_token": "REFRESH-TOKEN"
    }
  ]
}
```

{% endcode %}

Call the REST API

Now, that you have obtained an access token, you can call Content Software for File REST API commands with it. For example, you can query the cluster status:

{% code title="Python example calling cluster status API" %}

```
import requests

url = "https://weka01:14000/api/v2/cluster"

payload={}
headers = {
  'Authorization': 'Bearer REPLACE-WITH-ACCESS-TOKEN'
}

response = requests.request("GET", url, headers=headers, data=payload)

print(response.text)
```

{% endcode %}

Managing the TLS certificate using the CLI

How to deploy and replace the TLS certificate using the CLI.

Setting the TLS certificate using the CLI

You can set your TLS certificates using the CLI command:

```
weka security tls set
```

The command receives an unencrypted private key.

This command is similar to the OpenSSL command that Content Software for File uses to generate the self-signed certificate: .

```
openssl req -x509 -newkey rsa:1024 -keyout key.pem -out cert.pem -days <days> -nodes
```

Replacing the TLS certificate using the CLI

To replace the TLS certificate with a new one, use the CLI command:

```
weka security tls set
```

Once you issue a TLS certificate, it is used for connecting to the cluster (for the time it is issued), while the revocation is handled by the CA and propagating its revocation lists into the various clients.

Unsetting the TLS certificate

You can unset your TLS certificates using the CLI command:

```
weka security tls unset
```

Downloading the TLS certificate using the CLI

To download the TLS certificate, use the CLI command:

```
weka security tls download
```

Viewing the TLS certificate status using the CLI

To view the cluster TLS status and certificate, use the CLI command:

```
weka security tls status
```

Managing the CA certificate using the CLI

Content Software for File uses well-known CAs to establish trust with external services. For example, when using a KMS.

Use the CLI command:

```
weka security ca-cert set
```

Managing the account lockout threshold policy using CLI

To control the default values, use the following CLI commands:

```
weka security lockout-config set|show|reset
```

Commands options:

`set`: Sets the number of failed attempts until the account is locked (`--failed-attempts`) and the lockout duration (`--lockout-duration`).

`reset`: Resets the number of failed attempts until the account is locked and the lockout duration to their default values.

`show`: Shows the number of failed attempts until the account is locked and the lockout duration.

Managing the login banner using the CLI

To manage the login banner, use the following CLI command:

```
weka security login-banner set|show|reset|enable|disable
```

Command options:

`set`: Sets the login banner text.

`show`: Shows the login banner text.

`reset`: Clears the login banner text.

`enable`: Displays the login banner when accessing the cluster.

`disable`: Prevents displaying the login banner when accessing the cluster.

Chapter 14: Managing users

How to manage users for Content Software for File using the CLI.

Creating users using the CLI

Command

`weka user add`

Use the following command line to create a user:

```
weka user add <username> <role> [password] [--posix-uid uid] [--posix-gid gid]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
username	String	Name of the user to change the password for	Must be a valid local user	Yes	
role	String	Role of the new created user	`regular`, `readonly`, `orgadmin` or `clusteradmin`	Yes	
password	String	New password		No	If not supplied, command will prompt to supply the password

Name	Type	Value	Limitations	Mandatory	Default
posix-uid	Number	POSIX UID of underlying files representing objects created by this S3 user access/keys credentials	For S3 user roles only	No	0
posix-gid	Number	POSIX GID of underlying files representing objects created by this S3 user access/keys credentials	For S3 user roles only	No	0

For example,

```
$ weka user add my_new_user regular S3cret
```

This command line creates a user with a username of `my_new_user`, a password of `S3cret` and a role of Regular user. It is then possible to display a list of users and verify that the user was created:

```
1 $ weka user
2 Username    | Source   | Role
3 ------------+----------+--------
4 my_new_user | Internal | Regular
5 admin       | Internal | Admin
```

Using the **`weka user whoami`** command, it is possible to receive information about the current user running the command.

To use the new user credentials, use the `WEKA_USERNAME` and `WEKA_PASSWORD` environment variables:

```
$ WEKA_USERNAME=my_new_user WEKA_PASSWORD=S3cret 1 weka user whoami
2 Username    | Source   | Role
3 ------------+----------+--------
4 my_new_user | Internal | Regular
```

Changing user passwords using the CLI

Command

`weka user passwd`

Use the following command line to change a local user password:

```
weka user passwd <password> [--username username]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
password	String	New password		Yes	
username	String	Name of the user to change the password for	Must be a valid local user	No	Current logged-in user

Note: If necessary, provide or set `WEKA_USERNAME` or `WEKA_PASSWORD`.

Revoking user access using the CLI

Command:

`weka user revoke-tokens`

Use the following command to revoke internal user access to the system and mounting filesystems:

```
weka user revoke-tokens <username>
```

You can revoke the access for LDAP users by changing the user-revocation-attribute defined in the LDAP server configuration.

Parameters:

Name	Type	Value	Limitations	Mandatory	Default
username	String/ Integer	A valid user in the organization of the Organization Admin running the command		Yes	

Note: NFS and SMB are different protocols from WekaFS, which require additional security considerations when used. For example, The system grants NFS permissions per host. Therefore, manage the permissions for accessing these hosts for NFS export carefully.

Updating a local user using the CLI

Command:

`weka user update`

Use the following command line to update a local user:

```
weka user update <username> [--role role] [--posix-uid uid] [--posix-gid gid]
```

Parameters:

Name	Type	Value	Limitations	Mandatory	Default
username	String	Name of an existing user	Must be a valid local user	Yes	
role	String	Updated user role	`regular`, `s3`,`readonly`, `orgadmin` or `clusteradmin`	No	
posix-uid	Number	POSIX UID of underlying files representing objects created by this S3 user access/keys credentials	For S3 user roles only	No	

Name	Type	Value	Limitations	Mandatory	Default
posix-gid	Number	POSIX GID of underlying files representing objects created by this S3 user access/keys credentials	For S3 user roles only	No	

Deleting users using the CLI

Command

`weka user delete`

To delete a user, use the following command line:

```
weka user delete <username>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
username	String	Name of the user to delete.	Must be a valid local user.	Yes	

For example:

```
$ weka user add my_new_user
```

Then run the **`weka user`** command to verify that the user was deleted:

```
1 $ weka user
2 Username | Source | Role
3 ---------+----------+------
4 admin | Internal | Admin
```

User sign in

When a login is attempted, the user is first searched in the list of internal users, i.e., users created using the`weka user add` command.

However, if a user does not exist in the Content Software for File system but does exist in an LDAP directory, it is possible to configure the LDAP user directory to the Content Software for File system. This will enable a search for the user in the directory, followed by password verification.

On each successful login, a `UserLoggedIn` event is issued, containing the username, role and whether the user is an internal or LDAP user.

When a login fails, an "Invalid username or password" message is displayed and a UserLoginFailed event is issued, containing the username and the reason for the login failure.

When users open the GUI, they are prompted to provide their username and password. To pass username and password to the CLI, use the `WEKA_USERNAME` and `WEKA_PASSWORD` environment variables.

Alternatively, it is possible to log into the CLI as a specific user using the `weka user login <username> <password>`command. This will run each CLI command from that user. When a user logs in, a token file is created to be used for authentication (default to ~/.weka/auth-token.json, which can be changed using the --path attribute). To see the logged-in CLI user, run theweka user whoami command.

Note: The `weka user login` command is persistent, but only applies to the host on which it was set.

Note: If the`WEKA_USERNAME/WEKA_PASSWORD` environment variables are not specified, the CLI uses the default token file. If no CLI user is explicitly logged-in, and no token file is present the CLI uses the default `admin/admin`.

To use a non-default path for the token file, use the `WEKA_TOKEN` environment variable

Authenticating users from an LDAP user directory using the CLI

To authenticate users from an LDAP user directory, the LDAP directory must first be configured to theContent Software for File system. This is performed as follows.

Configuring an LDAP server using the CLI

Command

```
weka user ldap setup
```

```
weka user ldap setup-ad
```

One of two CLI commands is used to configure an LDAP user directory for user authentication. The first is for configuring a general LDAP server and the second is for configuring an Active Directory server.

To configure an LDAP server, use the following command line:

weka user ldap setup <server-uri> <base-dn> <user-object-class> <user-id-attribute> <group-object-class> <group-membership-attribute> <group-id-attribute> <reader-username> <reader-password> <cluster-admin-group> <org-admin-group> <regular-group> <readonly-group> [--start-tls start-tls] [--ignore-start-tls-failure ignore-start-tls-failure] [--server-timeout-secs server-timeout-secs] [--protocol-version protocol-version] [--user-revocation-attribute user-revocation-attribute]

To configure an Active Directory server, use the following command line:

```
weka user ldap setup-ad <server-uri> <domain> <reader-username> <reader-password>
<cluster-admin-group> <org-admin-group> <regular-group> <readonly-group> [--start-tls
start-tls] [--ignore-start-tls-failure ignore-start-tls-failure] [--server-timeout-
secs server-timeout-secs] [--user-revocation-attribute user-revocation-attribute]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
server-uri	String	Either the LDAP server host name/IP or a URI	URI must be in format ldap://hostname:port or ldaps://hostname:port	Yes	
base-dn	String	Base DN under which users are stored	Must be valid name	Yes	
user-id-attribute	String	Attribute storing user IDs	Must be valid name	Yes	
user-object-class	String	Object class of users	Must be valid name	Yes	
group-object-class	String	Object class of groups	Must be valid name	Yes	
group-membership-attribute	String	Attribute of group containing the DN of a user membership in the group	Must be valid name	Yes	

Name	Type	Value	Limitations	Mandatory	Default
group-id-attribute	String	Attribute storing the group name	Name has to match names used in the `<admin-group>`, `<regular group>` and `<readonly group>`	Yes	
reader-username and reader-password	String	Credentials of a user with read access to the directory	Password is kept in the Content Software for File cluster configuration in plain text, as it is used to authenticate against the directory during user authentication	Yes	
cluster-admin-group	String	Name of group containing users defined with cluster admin role	Must be valid name	Yes	
org-admin-group	String	Name of group containing users defined with organization admin role	Must be valid name	Yes	
regular-group	String	Name of group containing users defined with regular privileges	Must be valid name	Yes	
readonly-group	String	Name of group containing users defined with read only privileges	Must be valid name	Yes	
server-timeout-secs	Number	Server connection timeout	Seconds	No	
protocol-version	String	Selection of LDAP version	LDAP v2 or v3	No	LDAP v3

Name	Type	Value	Limitations	Mandatory	Default
user-revocation-attribute	String	The LDAP attribute; when its value changes in the LDAP directory, user access and mount tokens are revoked	User must re-login after a change is detected	No	
start-tls	String	Issue StartTLS after connecting	`yes` or `no`	No	No
ignore-start-tls-failure	String	Ignore start TLS failure	`yes` or `no`	No	No

Viewing a configured LDAP user directory using the CLI

Command

`weka user ldap`

This command is used for viewing the current LDAP configuration used for authenticating users.

Disabling or enabling a configured LDAP user directory using the CLI

Command

`weka user ldap disable`

`weka user ldap enable`

These commands are used for disabling or enabling user authentication through a configured LDAP user directory.

Note: You can only disable an LDAP configuration, but not delete it.

Chapter 15: Managing organizations

How to manage organizations using the CLI.

Managing organizations

Only users defined as Cluster Admins can manage organizations. When no organization is created, the root organization is the default organization and all operations are regular. That is, it is not necessary to authenticate the mounts or supply an organization name when logging in using the GUI/CLI.

Once a new organization is created, the organization name must be provided in every login command, using the `--org` attribute in the **`weka user login`** command.

Creating an organization using the CLI

Command

`weka org create`

Use the following command line to create an organization:

```
weka org create <name> <username> <password> [--ssd-quota ssd-quota] [--total-quota total-quota]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
name	String	Organization name	Must be a valid name	Yes	
username	String	Username of the created Organization Admin	Must be a valid name	Yes	
password	String	Password of the created Organization Admin		Yes	

Name	Type	Value	Limitations	Mandatory	Default
ssd-quota	Number	Allowed quota out of the system SSDs to be used by the organization	Must be a valid number	No	0 (not limited)
total-quota	Number	Total allowed quota for the organization (SSD and object store)	Must be a valid number	No	0 (not limited)

Viewing organizations using the CLI

Command

`weka org`

```
# weka org

ID | Name | Allocated SSD | SSD Quota | Allocated Total | Total Q
---+------------+---------------+-----------+-----------------+--------
 0 | Root | 0 B | 0 B | 0 B | 0 B
 1 | Local IT | 500.00 GB | 500.00 GB | 500.00 GB | 0 B
 2 | CUSTOMER_1 | 100.00 GB | 300.00 GB | 200.00 GB | 900.00
```

Renaming organizations using the CLI

Command

`weka org rename`

Use the following command line to rename an organization:

```
weka org rename <org> <new-name>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
org	String/Integer	Current organization name or ID		Yes	
new-name	String	New organization name		Yes	

Updating the quota of an organization using the CLI

Command

`weka org set-quota`

Use the following command line to update an organization's quota:

```
weka org set-quota <org> [--ssd-quota ssd-quota] [--total-quota total-quota]
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
org	String/ Integer	Organization name or ID	The root organization (org ID = 0 cannot be limited)	Yes	
ssd-quota	Number	Allowed quota out of the system SSDs to be used by the organization	Must be a valid number	No	
total-quota	Number	Total allowed quota for the organization (SSD and object store)	Must be a valid number	No	

Deleting an organization using the CLI

Command

```
weka org delete
```

Use the following command line to delete an organization:

```
weka org delete <org>
```

Note: Deleting an organization is irreversible. It removes all entities related to the organization, such as filesystems, object stores, and users.

Parameters

Name	Type	Value	Limitations	Mandatory	Default
org	String/Integer	Organization name or ID		Yes	

Mount authentication for organization filesystems

Once the Cluster Admin has created an organization and the Organization Admin has created filesystems, users, or configured the LDAP for the organization, regular users of the organization can mount filesystems.

The purpose of organizations is to provide separation and security for organization data, which requires authentication of the Content Software for File system filesystem mounts. This authentication of mounts prevents users of other organizations and even the Cluster Admin from accessing organization filesystems.

Mounting filesystems in an organization (other than the Root organization) is only supported using a stateless client. If the user is not logged into the Content Software for File system, a login prompt will appear as part of the mount command.

Mounting a filesystem using the CLI

To securely mount a filesystem, first log into the Content Software for File system:

```
weka user login my_user my_password --org my_org -H backend-host-0
```

Then mount the filesystem:

```
mount -t wekafs backend-host-0/my_fs /mnt/weka/my_fs
```

Mount authentication

Authentication is achieved by obtaining a mount token and including it in the mount command. Logging into the Content Software for File system using the CLI (the `weka user login` command) creates an authentication token and saves it in the client (default to `~/.weka/auth-token.json`, which can be changed using the`--path`attribute).

The Content Software for File system assigns the token that relates to a specific organization. Only mounts that pass the path to a correct token can successfully access the filesystems of the organization.

Once the system authenticates a user, the mount command uses the default location of the authentication token. It is possible to change the token location/name and pass it as a parameter in the mount command using the `auth_token_path` mount option, or the`WEKA_TOKEN` environment variable.

```
mount -t wekafs backend-host-0/my_fs /mnt/weka/my_fs -o auth_token_path=<path>
```

This option is useful when mounting several filesystems for several users/organizations on the same host or when using Autofs.

When a token is compromised or no longer required, such as when a user leaves the organization, the Organization Admin can prevent using that token for new mounts by revoking the user access.

Chapter 16: Expansion of specific resources

Guidelines provided for the expansion processes that only involve the addition of specific resources.

Dynamic modifications using the CLI

Most modifications to host configurations can be performed dynamically, without deactivating the host. Such configurations include the addition or removal of memory and network resources, changing IPs, extending network subnets and limiting the Content Software for File system bandwidth on the host.

All these changes can be performed using the relevant `weka cluster host`command. Once this command is used with a specific `host-id`selected, it will be staged for update on the cluster. To view the un-applied configuration, use the `weka cluster host resources <host-id>` command. To apply the changes, use the `weka cluster host apply <host-ids>` command. You can also apply these changes locally using the `weka local resources apply` command.

The last local configuration (of a host that successfully joined a cluster) is saved. If a failure/problem occurs with the new configuration, the host will automatically revert to the last known good configuration. To view this configuration, use the `weka cluster host resources <host-id> --stable` command.

Memory modifications

To dynamically change the memory configuration, use the steps described for the configuration of memory on an active host, followed by the **`weka cluster host apply`** command.

For example: to change host-id 0 memory to 1.5 GB, run the following commands:

```
weka cluster host memory 0 1.5GB
weka cluster host apply 0
```

Network modifications

To dynamically change the network configuration, use the steps described for the Configuration of networking section in the *Content Software for File User Guide* on an active host, followed by the **`weka cluster host apply`** command.

For example: To add another network device to host-id 0, run the following commands:

```
weka cluster host net add 0 --device=eth2
weka cluster host apply 0
```

Note: It is possible to accumulate several changes on a host and apply only once on completion.

For additional information, contact customer support.

Host IPs modifications

To dynamically change the host's management IPs, you can use the **`management-ips resource editing`** command.

For example: To change the management IPs onhost-id 0, run the following commands:

```
weka cluster host management-ips 192.168.1.10 192.168.1.20
weka cluster host apply 0
```

Note: The number of management IPs determines whether the host will use Highly Available Networking mode (HA), causing each IO process to use both hosts NICs. A host with 2 IPs will use HA mode and a host with only 1 IP will not use HA mode. It is also possible to define up to 4 IPs, in case the cluster is using both Infiniband and Ethernet network technologies.

Local resources editing commands using the CLI

It is also possible to run modification commands locally on the host by connecting to the desired host and running a **`local resources`** command equivalent to its **`weka cluster host`** counterpart. These local commands have the same semantics of their remote counterparts only that they don't receive the host-id as the first parameter and operate instead on the local host. Commands that can be performed dynamically on an Active host:

```
weka local resources [--stable]
weka local resources apply
weka local resources net
weka local resources net add
weka local resources net remove
weka local resources memory
weka local resources bandwidth
weka local resources management-ips
weka local resources dedicate
```

The following commands cannot be performed on an Active host and require deactivating the host first using **`weka cluster host deactivate`**:

```
weka local resources failure-domain
weka local resources cores
```

Chapter 17: Addition of CPU cores

The addition of CPU cores to the cluster is not performed dynamically but on an inactive host. For more information, contact customer support.

Chapter 18: Expansion of only SSDs

For information, contact customer support.

Chapter 19: Managing clusters

How to expand and shrink a cluster in a homogeneous Content Software for File system configuration.

Note: The cluster expansion process described here is only applicable to a homogeneous Content Software for File system configuration, which is highly recommended. For non-homogeneous system configurations, contact your Hitachi representative.

Shrinking a cluster using the CLI

The procedures are described that are involved in the shrinking of a cluster and may be required when it is necessary to reallocate cluster hardware.

Options for shrinking a cluster

Cluster shrinking can involve either the removal of some of the assigned SSDs or the removal of hosts from the system. The following operations are available:

1. Listing all the drives and their states, in order to receive a view of currently-allocated resources and their status.
2. Deactivating drives as the first step before removing a host.
3. Removing (a subset of) SSD drives allocated for the cluster.
4. Deactivating hosts, which can be used after deactivating drives in preparation for the removal of the host.
5. Removing hosts in order to complete the cluster shrinking.

Listing drives and their states using the CLI

Command

```
weka cluster drive
```

Use this command to display a list of all the drives in the cluster and their status.

Deactivating a drive using the CLI

Command

```
weka cluster drive deactivate
```

Running this command will redistribute the stored data on the remaining drives and can be performed on multiple drives.

Note: After running this command, the deactivated drives will still appear in the list.

Note: It is not possible to deactivate a drive if it will lead to an unstable state, that is, if the system capacity after drive deactivation is insufficient for the SSD capacity of currently-provisioned filesystems.

Drive deactivation starts an asynchronous process known as phasing out, which is a gradual redistribution of the data between the remaining drives in the system. On completion, the phased-out drive is in an inactive state, that is, not in use by the Content Software for File system, but still appearing in the list of drives.

Note: Running the **`weka cluster drive`** command will display whether the redistribution is still being performed.

To deactivate a drive, run the following command:

```
weka cluster drive deactivate <uuids>
```

Parameters

Name	Type	Value	Limitations	Mandatory	Default
uuids	Comma-separated strings	Comma-separated drive identifiers		Yes	

Removing a drive using the CLI

Command

`weka cluster drive remove`

This command is used to completely remove a drive from the cluster. After removal, the drive will not be recoverable. To remove a drive, run the following command:

```
weka cluster drive remove <uuids>
```

Table 6 Parameters

Name	Type	Value	Limitations	Mandatory	Default
uuids	Comma-separated strings	Comma-separated strings		Yes	

Deactivating an entire host using the CLI

Command

`weka cluster host deactivate`

his command is used as the first step when seeking to shrink a cluster. Running this command will automatically deactivate all the host's drives. To deactivate an entire host, run the following command:

```
weka cluster host deactivate <host-ids> [--allow-unavailable-host]
```

Table 7 Parameters

Name	Type	Value	Limitations	Mandatory	Default
host-ids	Space-separated integers	Space-separated host identifiers		Yes	
allow-unavailable-host	Boolean	Allow deactivation of an unavailable host	If the host returns, it will join the cluster in an active state	No	No

Removing a host using the CLI

Command

`weka cluster host remove`

Running this command will eliminate the host from the cluster, that is, the host will switch to the `stem mode` after the removal, at which point it can be reallocated either to another cluster or purpose. To remove the host from the cluster, run the following command:

```
weka cluster host remove <host-id>
```

Table 8 Parameters

Name	Type	Value	Limitations	Mandatory	Default
host-id	Comma-separated strings	Comma-separated host identifiers		Yes	

Chapter 20: Managing background tasks

The management of background tasks running on Content Software for File clusters is described.

The Content Software for File system has some internal/external asynchronous operations and maintenance tasks, such as migrating an object store and downloading/uploading snapshots. These tasks are performed in the background and should not interfere nor starve the Content Software for File system from serving IOs with high performance.

The Content Software for File system limits the CPU resources these tasks consume to 5% per host CPU.

Note: When the CPU is idle, background tasks can use more than the configured resources, but they are immediately freed if needed for serving IOs.

Note: The configured limit affects both external tasks (that are visible using the GUI/CLI) and internal low-priority asynchronous operations.

Viewing background tasks using the CLI

It is possible to view currently-running background tasks, including their status and progress.

Command

`weka cluster tasks`

This command is used for viewing all background tasks. For each task, a range of data can be displayed, as shown in the following example:

```
# weka cluster tasks
Type        | State    | Progress | Description
------------+----------+----------+-------------------------------------
OBS_DETACH | RUNNING | 94 | Detaching Object Storage `obs_1` from
```

Limiting background tasks using the CLI

It is possible to limit the resources being used by background tasks.

Command

`weka cluster tasks limits`

This command is used to view the currently-defined limits.

Command

`weka cluster tasks limits set <cpu-limit limit>`

This command is used to update the CPU limit.

Pause/Resume/Abort a background task

It is possible to pause and later resume a background task, as well as completely abort it. This is useful in case there are other tasks/activities that are of higher priority.

Command:

```
weka cluster task pause / resume / abort <task-id>
```

This command is used to pause/resume/abort the running of a specific task.

Note: Up to 16 background tasks can run in parallel. A paused (or aborting) task still consumes one of these spots.

Chapter 21: Running cluster diagnostics

The details for running cluster diagnostic commands are provided.

Managing diagnostics using the CLI

Command

`weka diags`

Command

`weka local diags`

The command **`weka diags`** is used for cluster-wide diagnostics from any host in the cluster. The **`weka local diags`** command creates diagnostics information about the Content Software for File software and saves it for further analysis by customer support.

The commands can be run with the following options:

```
weka [local] diags <--collect|--upload> [--pack-to dir]
```

When **`weka local diags`** receives a directory using the `-o` option, the diagnostics dump of the host is moved to that directory on completion of the collection process.

Note: In the following situations the local option should be used: when no functioning manager in the originating host or the hosts being addressed or when there is no connectivity between the manager and the cluster leader, the cluster has no leader, the local container is down, the host cannot reach the leader or a remote host fails to respond to the **`weka diags remote`** command.

Chapter 22: Container storage interface (CSI) plugin

The Content Software for File CSI Plugin prerequisites, capabilities, deployment, and usage are described.

CSI plugin overview

The Container Storage Interface (CSI) is a standard for exposing arbitrary block and file storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes.

The Content Software for File CSI Plugin provides the creation and configuration of persistent storage external to Kubernetes. CSI replaces plugins developed earlier in the Kubernetes evolution. It replaces the hostPath method to expose WekaFS mounts as Kubernetes volumes.

Interoperability

- CSI protocol: 1.0-1.2
- Kubernetes: 1.18-1.2
- WekaFS: 3.8 and up
- AppArmor is not supported yet

Note: Quota enforcement on persistent volumes requires WekaFS version 3.13 and up .

Prerequisites

The Prerequisites include:

- The privileged mode must be allowed on the Kubernetes cluster
- The following Kubernetes feature gates must be enabled: DevicePlugins, CSINodeInfo, CSIDriverRegistry, ExpandCSIVolumes (if not changed, they should be enabled by default)

- A Content Software for File cluster is installed and accessible from the Kubernetes worker nodes.
- The Content Software for File client is installed on the Kubernetes worker nodes.
  - It is recommended to use a client which is part of the cluster rather than a stateless client.
  - If the Kubernetes nodes are part of the Content Software for File cluster (converged mode on the servers), make sure the Content Software for File processes come up before `kubelet`.
- Filesystems are pre-configured on the Content Software for File system.

Capabilities

The capabilities listed in this section are catagorized as supported and unsupported capabilities.

Supported capabilities

Supported capabilities

- Static and dynamic volumes provisioning
- Mounting a volume as a WekaFS filesystem directory
- All volume access modes are supported: ReadWriteMany, ReadWriteOnce, and ReadOnlyMany
- Volume expansion

Unsupported capabilities

Snapshots

Deployment

The Content Software for File CSI Plugin deployment is performed using a daemon set.

Download

To obtain the CSI Plugin package, see your Hitachi representative.

Installation

From the downloaded location in the Kubernetes master node, run the following command to deploy the Content Software for File CSI Plugin as a DaemonsSet:

```
$ ./deploy/kubernetes-latest/deploy.sh
```

On successful deployment, you will see the following output:

```
creating wekafsplugin 1 namespace
2 namespace/csi-wekafsplugin created
3 deploying wekafs components
4 ./deploy/kubernetes-latest/wekafs/csi-wekafs-plugin.yaml
5 using image: quay.io/k8scsi/csi-node-driver-registrar
6 using image: quay.io/weka.io/csi-wekafs:v0.0.2-25-g7d
7 using image: quay.io/k8scsi/livenessprobe:v1.1.0
8 using image: quay.io/k8scsi/csi-provisioner:v1.6.0
9 using image: quay.io/k8scsi/csi-attacher:v3.0.0-rc1
10 using image: quay.io/k8scsi/csi-resizer:v0.5.0
11 namespace/csi-wekafsplugin configured
12 csidriver.storage.k8s.io/wekafs.csi.k8s.io created
13 serviceaccount/csi-wekafsplugin created
14 clusterrole.rbac.authorization.k8s.io/csi-wekafsplugin-cluster-role cre
15 clusterrolebinding.rbac.authorization.k8s.io/csi-wekafsplugin-cluster-r
16 role.rbac.authorization.k8s.io/csi-wekafsplugin-role created
17 rolebinding.rbac.authorization.k8s.io/csi-wekafsplugin-role-binding cre
18 daemonset.apps/csi-wekafsplugin created
19 12:04:54 deployment completed successfully
20 12:04:54 2 plugin pods are running:
21 csi-wekafsplugin-dvdh2 6/6 Running 0 3h1m
22 csi-wekafsplugin-xh182 6/6 Running 0 3h1m
```

The number of running pods should be the same as the number of Kubernetes worker nodes. This can be inspected by running:

```
1 $ kubectl get pods -n csi-wekafsplugin
2 NAME READY STATUS RESTARTS AGE
3 csi-wekafsplugin-dvdh2 6/6 Running 0 3h2m
4 csi-wekafsplugin-xh182 6/6 Running 0 3h2m
```

Provision usage

The Content Software for File CSI Plugin supports both dynamic (persistent volume claim) and static (persistent volume) volume provisioning.

It is first required to define a storage class to use the CSI Plugin.

Storage class example

```
csi-wekafs/examples/dynamic/storageclass-wekafs-dir.yaml
apiVersion: 1 storage.k8s.io/v1
2 kind: StorageClass
3 metadata:
4 name: storageclass-wekafs-dir
5 provisioner: csi.weka.io
```

```
6 reclaimPolicy: Delete
7 volumeBindingMode: Immediate
8 allowVolumeExpansion: true
9 parameters:
10 volumeType: dir/v1
11 filesystemName: podsFilesystem
```

Storage class parameters

Parameter	Description	Limitation
filesystemName	The name of the Content Software for File filesystem to create directories in as Kubernetes volumes	The filesystem should exist in the cluster

Apply the StorageClass and check it has been created successfully:

```
1 # apply the storageclass .yaml file
2 $ kubectl apply -f storageclass-wekafs-dir.yaml
3 storageclass.storage.k8s.io/storageclass-wekafs-dir created
4
5 # check the storageclass resource has been created
6 $ kubectl get sc
NAME PROVISIONER RECLAIMPOLICY 7 VOLU
8 storageclass-wekafs-dir csi.weka.io Delete Imme
```

It is possible to define multiple storage classes with different filesystems.

Dynamic provisioning

Using a similar storage class to the above, it is possible to define a persistent volume claim (PVC) for the pods.

Persistent volume claim example

```
csi-wekafs/examples/dynamic/pvc-wekafs-dir.yaml
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4 name: pvc-wekafs-dir
5 spec:
6 accessModes:
7 - ReadWriteMany
8 storageClassName: storageclass-wekafs-dir
9 volumeMode: Filesystem
```

```
10 resources:
11 requests:
12 storage: 1Gi
```

Persistent volume claim parameters

Parameter	Description	Limitation
spec.accessModes	The volume access mode	ReadWriteMany, ReadWriteOnce, or ReadOnlyMany
spec.storageClassName	The storage class to use to create the PVC	Must be an existing storage class
spec.resources.requests.storage	A desired capacity for the volume	The capacity quota is not enforced but is stored on the filesystem directory extended attributed for future use

Apply the PersistentVolumeClaim and check it has been created successfully:

```
# apply 1 the pvc .yaml file
2 $ kubectl apply -f pvc-wekafs-dir.yaml
3 persistentvolumeclaim/pvc-wekafs-dir created
4
5 # check the pvc resource has been created
6 $ kubectl get pvc
7 NAME STATUS VOLUME
8 pvc-wekafs-dir Bound pvc-d00ba0fe-04a0-4916-8fea-ddbbc8f43380
```

Note: The directory will be created inside the filesystem under `csi-volumes` directory, starting with the volume name.

Static provisioning

The Kubernetes admin can prepare some persistent volumes in advance to be used by pods, they should be an existing directory, and can contain pre-populated data to be used by the PODs.

It can be a directory previously provisioned by the CSI or a pre-existing directory in WekaFS. To expose an existing directory in WekaFS using CSI, define a persistent volume, and link a persistent volume claim to this persistent volume.

Persistent volume example

```
csi-wekafs/examples/static/pv-wekafs-dir-static.yaml
1 apiVersion: v1
2 kind: PersistentVolume
3 metadata:
4 name: pv-wekafs-dir-static
5 spec:
6 storageClassName: storageclass-wekafs-dir
7 accessModes:
8 - ReadWriteMany
9 persistentVolumeReclaimPolicy: Retain
10 volumeMode: Filesystem
11 capacity:
12 storage: 1Gi
13 csi:
14 driver: csi.weka.io
15 # volumeHandle must be formatted as following:
16 # dir/v1/<FILE_SYSTEM_NAME>/<INNER_PATH_IN_FILESYSTEM>
17 # The path must exist, otherwise publish request will fail
18 volumeHandle: dir/v1/podsFilesystem/my-dir
```

Persistent volume parameters

Parameter	Description	Limitation
spec.accessModes	The volumeaccess mode	ReadWriteMany, ReadWriteOnce, or ReadOnlyMany
spec.storageClassName	The storage class to use to create the PV	Must be an existing storage class
spec.capacity.storage	A desired capacity for the volume	The capacity quota is not enforced but is stored on the filesystem directory extended attributed for future use
spec.csi.volumeHandle	A string specifying a previously created path	A string containing the volumeType (`dir/v1`) filesystem name, and the directory path. For example, `dir/v1/podsFilesystem/my-dir` Must be an existing filesystem and path

Apply the PersistentVolume and check it has been created successfully:

```
1 the pv .yaml file
2 $ kubectl apply -f pv-wekafs-dir-static.yaml
```

```
3 persistentvolume/pv-wekafs-dir-static created
4
5 # check the pv resource has been created
6 $ kubectl get pv
7 NAME CAPACITY ACCESS MODES RE
8 pv-wekafs-dir-static 1Gi RWX Re
```

Now, bind a PVC to this specific PV, use the `volumeName` parameter under the PVC `spec` and provide it with the specific PV name.

Persistent volume claim for static provisioning example

```
csi-wekafs/examples/static/pvc-wekafs-dir-static.yaml
1 apiVersion: v1
2 kind: PersistentVolumeClaim
3 metadata:
4 name: pvc-wekafs-dir-static
5 spec:
6 accessModes:
7 - ReadWriteMany
8 storageClassName: storageclass-wekafs-dir
9 volumeName: pv-wekafs-dir-static
10 volumeMode: Filesystem
11 resources:
12 requests:
13 storage: 1Gi
```

Parameter	Description	Limitation
spec.accessModes	The volume access mode	ReadWriteMany, ReadWriteOnce, or ReadOnlyMany
spec.storageClassName	The storage class to use to create the PVC	Must be the same storage class as the PV requested to bind in `spec.volumeName`
spec.resources.requests.storage	A desired capacity for the volume	The capacity quota is not enforced but is stored on the filesystem directory extended attributed for future use
spec.volumeName	A name of a preconfigured persistent volume	Must be an existing PV name

Apply the PersistentVolumeClaim and check it has been created successfully:

```
# apply 1 the pvc .yaml file
2 $ kubectl apply -f pvc-wekafs-dir-static.yaml
3 persistentvolumeclaim/pvc-wekafs-dir-static created
4
5 # check the pvc resource has been created
6 $ kubectl get pvc
7 NAME STATUS VOLUME CAPACITY ACCES
8 pvc-wekafs-dir-static Bound pv-wekafs-dir-static 1Gi RWX
```

The PV will change the status to `Bound` and state the relevant claim it is bounded to:

```
1 # check the pv resource has been created
2 $ kubectl get pv
3 NAME CAPACITY ACCESS MODES RE
4 pv-wekafs-dir-static 1Gi RWX Re
```

Launching an application using CSF as the POD's storage

Now that we have a storage class and a PVC in place, we can configure the Kubernetes pods to provision volumes using the Content Software for File system.

We'll take an example application that echos the current timestamp every 10 seconds, and provide it with the previously created `pvc-wekafs-dir` PVC.

Multiple pods can share a volume produced by the same PVC as long as the `accessModes` parameter is set to `ReadWriteMany`.

```
csi-wekafs/examples/dynamic/csi-app-on-dir.yaml
1 kind: Pod
2 apiVersion: v1
3 metadata:
4 name: my-csi-app
5 spec:
6 containers:
7 - name: my-frontend
8 image: busybox
9 volumeMounts:
10 - mountPath: "/data"
11 name: my-csi-volume
12 command: ["/bin/sh"]
13 args: ["-c", "while true; do echo `date` >> /data/temp.txt; sleep
14 volumes:
15 - name: my-csi-volume
16 persistentVolumeClaim:
17 claimName: pvc-wekafs-dir # defined in pvc-wekafs-dir.yaml
```

Now we will apply that pod:

```
1 $ kubectl apply -f csi-app-on-dir.yaml
2 pod/my-csi-app created
```

Kubernetes will allocate a persistent volume and attach it to the pod, it will use a directory within the WekaFS filesystem as defined in the storage class mentioned in the persistent volume claim. The pod will be in `Running` status, and the `temp.txt` file will get updated with occasional `date` information.

```
1 get pod my-csi-app
2 NAME READY STATUS RESTARTS AGE
3 my-csi-app 1/1 Running 0 85s
4
5 # if we go to a wekafs mount of this filesystem we can see a directory
6 $ ls -l /mnt/weka/podsFilesystem/csi-volumes
7 drwxr-x--- 1 root root 0 Jul 19 12:18 pvc-d00ba0fe-04a0-4916-8fea-ddbbc
8
9 # inside that directory, the temp.txt file from the running pod can be
10 $ cat /mnt/weka/podsFilesystem/csi-volumes/pvc-d00ba0fe-04a0-4916-8fea
11 Sun Jul 19 12:50:25 IDT 2020
12 Sun Jul 19 12:50:35 IDT 2020
13 Sun Jul 19 12:50:45 IDT 2020
```

Troubleshooting

Here are some useful basic commands to check the status and debug the service:

```
1 # get all resources
2 kubectl get all --all-namespaces
3
4 # get all pods
5 kubectl get pods --all-namespaces -o wide
6
7 # get all k8s nodes
8 kubectl get nodes
9
10 # get storage classes
11 $ kubectl get sc
12
13 # get persistent volume claims
14 $ kubectl get pvc
15
16 # get persistent volumes
17 $ kubectl get pv
18
19 # kubectl describe pod/<pod-name> -n <namespace>
20 kubectl describe pod/csi-wekafsplugin-dvdh2 -n csi-wekafsplugin
```

```
21
22 # get logs from a pod
kubectl logs <pod name> 23 <container name>
24
25 # get logs from the weka csi plugin
26 # container (-c) can be one of: [node-driver-registrar wekafs liveness27
kubectl logs pods/csi-wekafsplugin-<ID> --namespace csi-wekafsplugin -c
```

Known issues

Mixed Hugepages Size Issue

Due to a Kubernetes v1.18 issue with allocating mixed hugepages sizes (https://github.com/kubernetes/kubernetes/pull/80831) is required that the Content Software for File system will not try to allocate mixed sizes of hugepages on the Kubernetes nodes.

To workaround the Kubernetes issue (required only if the default memory for the client has been increased):

- If the client is installed on the K8s nodes using a manual stateless client mount, set the `reserve_1g_hugepages` mount option to `false` in the mount command.
- If this is a server or a client, which is part of the Content Software for File cluster, contact customer support.

Hitachi Vantara

Corporate Headquarters
2535 Augustine Drive
Santa Clara, CA 95054 USA

HitachiVantara.com/contact