Use the ONTAP command-line interface

ONTAP 9

NetApp
November 28, 2022
Table of Contents

Using the ONTAP command-line interface................................................................. 1
  About the different shells for CLI commands (cluster administrators only).............. 1
  Methods of navigating CLI command directories..................................................... 3
  Rules for specifying values in the CLI................................................................. 4
  Methods of viewing command history and reissuing commands................................. 5
  Keyboard shortcuts for editing CLI commands......................................................... 5
  Use of administrative privilege levels....................................................................... 7
  Set the privilege level in the CLI............................................................................... 8
  Set display preferences in the CLI............................................................................. 8
  Methods of using query operators.............................................................................. 9
  Methods of using extended queries............................................................................ 10
  Methods of customizing show command output by using fields................................. 11
  About positional parameters..................................................................................... 12
  Methods of accessing ONTAP man pages.................................................................... 14
Using the ONTAP command-line interface

The ONTAP command-line interface (CLI) provides a command-based view of the management interface. You enter commands at the storage system prompt, and command results are displayed in text.

The CLI command prompt is represented as `cluster_name::>`.

If you set the privilege level (that is, the `-privilege` parameter of the `set` command) to advanced, the prompt includes an asterisk (*), for example:

`cluster_name::*>`

About the different shells for CLI commands (cluster administrators only)

About the different shells for CLI commands overview (cluster administrators only)

The cluster has three different shells for CLI commands, the `clustershell`, the `nodeshell`, and the `systemshell`. The shells are for different purposes, and they each have a different command set.

- The `clustershell` is the native shell that is started automatically when you log in to the cluster.

  It provides all the commands you need to configure and manage the cluster. The `clustershell` CLI help (triggered by `?` at the `clustershell` prompt) displays available `clustershell` commands. The `man command_name` command in the `clustershell` displays the man page for the specified `clustershell` command.

- The `nodeshell` is a special shell for commands that take effect only at the node level.

  The `nodeshell` is accessible through the `system node run` command.

  The `nodeshell` CLI help (triggered by `?` or `help` at the `nodeshell` prompt) displays available `nodeshell` commands. The `man command_name` command in the `nodeshell` displays the man page for the specified `nodeshell` command.

  Many commonly used `nodeshell` commands and options are tunneled or aliased into the `clustershell` and can be executed also from the `clustershell`.

- The `systemshell` is a low-level shell that is used only for diagnostic and troubleshooting purposes.

  The `systemshell` and the associated “diag” account are intended for low-level diagnostic purposes. Their access requires the diagnostic privilege level and is reserved only for technical support to perform troubleshooting tasks.

Access of `nodeshell` commands and options in the `clustershell`

`Nodeshell` commands and options are accessible through the `nodeshell`:
Many commonly used nodeshell commands and options are tunneled or aliased into the clustershell and can be executed also from the clustershell.

Nodeshell options that are supported in the clustershell can be accessed by using the `vserver options` command. To see these options, you can do one of the following:

- Query the clustershell CLI with `vserver options -vserver nodename_or_clustername -option-name`?
- Access the `vserver options` man page in the clustershell CLI with `man vserver options`

If you enter a nodeshell or legacy command or option in the clustershell, and the command or option has an equivalent clustershell command, ONTAP informs you of the clustershell command to use.

If you enter a nodeshell or legacy command or option that is not supported in the clustershell, ONTAP informs you of the “not supported” status for the command or option.

**Display available nodeshell commands**

You can obtain a list of available nodeshell commands by using the CLI help from the nodeshell.

**Steps**

1. To access the nodeshell, enter the following command at the clustershell’s system prompt:

   ```
   system node run -node {nodename|local}
   ```

   `local` is the node you used to access the cluster.

   - The `system node run` command has an alias command, `run`.

2. Enter the following command in the nodeshell to see the list of available nodeshell commands:

   ```
   [commandname] help
   ```

   `commandname` is the name of the command whose availability you want to display. If you do not include `commandname`, the CLI displays all available nodeshell commands.

   You enter `exit` or type Ctrl-d to return to the clustershell CLI.

**Example of displaying available nodeshell commands**

The following example accesses the nodeshell of a node named node2 and displays information for the nodeshell command `environment`: 

```
Methods of navigating CLI command directories

Commands in the CLI are organized into a hierarchy by command directories. You can run commands in the hierarchy either by entering the full command path or by navigating through the directory structure.

When using the CLI, you can access a command directory by typing the directory’s name at the prompt and then pressing Enter. The directory name is then included in the prompt text to indicate that you are interacting with the appropriate command directory. To move deeper into the command hierarchy, you type the name of a command subdirectory followed by pressing Enter. The subdirectory name is then included in the prompt text and the context shifts to that subdirectory.

You can navigate through several command directories by entering the entire command. For example, you can display information about disk drives by entering the `storage disk show` command at the prompt. You can also run the command by navigating through one command directory at a time, as shown in the following example:

```
cluster1::> storage
cluster1::storage> disk
cluster1::storage disk> show
```

You can abbreviate commands by entering only the minimum number of letters in a command that makes the command unique to the current directory. For example, to abbreviate the command in the previous example, you can enter `st d sh`. You can also use the Tab key to expand abbreviated commands and to display a command’s parameters, including default parameter values.

You can use the `top` command to go to the top level of the command hierarchy, and the `up` command or `. ..` command to go up one level in the command hierarchy.

Commands and command options preceded by an asterisk (*) in the CLI can be executed only at the advanced privilege level or higher.
Rules for specifying values in the CLI

Most commands include one or more required or optional parameters. Many parameters require you to specify a value for them. A few rules exist for specifying values in the CLI.

• A value can be a number, a Boolean specifier, a selection from an enumerated list of predefined values, or a text string.

Some parameters can accept a comma-separated list of two or more values. Comma-separated lists of values do not need to be in quotation marks (" "). Whenever you specify text, a space, or a query character (when not meant as a query or text starting with a less-than or greater-than symbol), you must enclose the entity in quotation marks.

• The CLI interprets a question mark ("?") as the command to display help information for a particular command.

• Some text that you enter in the CLI, such as command names, parameters, and certain values, is not case-sensitive.

For example, when you enter parameter values for the vserver cifs commands, capitalization is ignored. However, most parameter values, such as the names of nodes, storage virtual machines (SVMs), aggregates, volumes, and logical interfaces, are case-sensitive.

• If you want to clear the value of a parameter that takes a string or a list, you specify an empty set of quotation marks (""") or a dash ("-").

• The hash sign ("#"), also known as the pound sign, indicates a comment for a command-line input; if used, it should appear after the last parameter in a command line.

The CLI ignores the text between "#" and the end of the line.

In the following example, an SVM is created with a text comment. The SVM is then modified to delete the comment:

```bash
cluster1::> vserver create -vserver vs0 -subtype default -rootvolume root_vs0 -aggregate aggr1 -rootvolume-security-style unix -language C.UTF-8 -is -repository false -ipspace ipspaceA -comment "My SVM"
cluster1::> vserver modify -vserver vs0 -comment ""
```

In the following example, a command-line comment that uses the "#" sign indicates what the command does.

```bash
cluster1::> security login create -vserver vs0 -user-or-group-name new-admin -application ssh -authmethod password #This command creates a new user account
```
Methods of viewing command history and reissuing commands

Each CLI session keeps a history of all commands issued in it. You can view the command history of the session that you are currently in. You can also reissue commands.

To view the command history, you can use the `history` command.

To reissue a command, you can use the `redo` command with one of the following arguments:

- A string that matches part of a previous command
  
  For example, if the only `volume` command you have run is `volume show`, you can use the `redo volume` command to reexecute the command.

- The numeric ID of a previous command, as listed by the `history` command
  
  For example, you can use the `redo 4` command to reissue the fourth command in the history list.

- A negative offset from the end of the history list
  
  For example, you can use the `redo -2` command to reissue the command that you ran two commands ago.

For example, to redo the command that is third from the end of the command history, you would enter the following command:

```
cluster1::> redo -3
```

Keyboard shortcuts for editing CLI commands

The command at the current command prompt is the active command. Using keyboard shortcuts enables you to edit the active command quickly. These keyboard shortcuts are similar to those of the UNIX tcsh shell and the Emacs editor.

The following table lists the keyboard shortcuts for editing CLI commands. “Ctrl-” indicates that you press and hold the Ctrl key while typing the character specified after it. “Esc-” indicates that you press and release the Esc key and then type the character specified after it.

<table>
<thead>
<tr>
<th>If you want to...</th>
<th>Use the following keyboard shortcut...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Move the cursor back by one character</td>
<td>Ctrl-B</td>
</tr>
<tr>
<td></td>
<td>Back arrow</td>
</tr>
<tr>
<td>If you want to...</td>
<td>Use the following keyboard shortcut...</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------</td>
<td>----------------------------------------</td>
</tr>
<tr>
<td>Move the cursor forward by one character</td>
<td>Ctrl-F</td>
</tr>
<tr>
<td></td>
<td>Forward arrow</td>
</tr>
<tr>
<td>Move the cursor back by one word</td>
<td>Esc-B</td>
</tr>
<tr>
<td>Move the cursor forward by one word</td>
<td>Esc-F</td>
</tr>
<tr>
<td>Move the cursor to the beginning of the line</td>
<td>Ctrl-A</td>
</tr>
<tr>
<td>Move the cursor to the end of the line</td>
<td>Ctrl-E</td>
</tr>
<tr>
<td>Remove the content of the command line from the beginning of the line to the cursor, and save it in the cut buffer. The cut buffer acts like temporary memory, similar to what is called a clipboard in some programs.</td>
<td>Ctrl-U</td>
</tr>
<tr>
<td>Remove the content of the command line from the cursor to the end of the line, and save it in the cut buffer</td>
<td>Ctrl-K</td>
</tr>
<tr>
<td>Remove the content of the command line from the cursor to the end of the following word, and save it in the cut buffer</td>
<td>Esc-D</td>
</tr>
<tr>
<td>Remove the word before the cursor, and save it in the cut buffer</td>
<td>Ctrl-W</td>
</tr>
<tr>
<td>Yank the content of the cut buffer, and push it into the command line at the cursor</td>
<td>Ctrl-Y</td>
</tr>
<tr>
<td>Delete the character before the cursor</td>
<td>Ctrl-H</td>
</tr>
<tr>
<td></td>
<td>Backspace</td>
</tr>
<tr>
<td>Delete the character where the cursor is</td>
<td>Ctrl-D</td>
</tr>
<tr>
<td>Clear the line</td>
<td>Ctrl-C</td>
</tr>
<tr>
<td>Clear the screen</td>
<td>Ctrl-L</td>
</tr>
<tr>
<td>If you want to…</td>
<td>Use the following keyboard shortcut…</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>--------------------------------------</td>
</tr>
<tr>
<td>Replace the current content of the command line with the previous entry on the history list.</td>
<td>Ctrl-P</td>
</tr>
<tr>
<td>With each repetition of the keyboard shortcut, the history cursor moves to the previous entry.</td>
<td>Esc-P</td>
</tr>
<tr>
<td></td>
<td>Up arrow</td>
</tr>
<tr>
<td>Replace the current content of the command line with the next entry on the history list. With each repetition of the keyboard shortcut, the history cursor moves to the next entry.</td>
<td>Ctrl-N</td>
</tr>
<tr>
<td></td>
<td>Esc-N</td>
</tr>
<tr>
<td></td>
<td>Down arrow</td>
</tr>
<tr>
<td>Expand a partially entered command or list valid input from the current editing position</td>
<td>Tab</td>
</tr>
<tr>
<td></td>
<td>Ctrl-I</td>
</tr>
<tr>
<td>Display context-sensitive help</td>
<td>?</td>
</tr>
<tr>
<td>Escape the special mapping for the question mark (“?”) character. For instance, to enter a question mark into a command’s argument, press Esc and then the “?” character.</td>
<td>Esc-?</td>
</tr>
<tr>
<td>Start TTY output</td>
<td>Ctrl-Q</td>
</tr>
<tr>
<td>Stop TTY output</td>
<td>Ctrl-S</td>
</tr>
</tbody>
</table>

### Use of administrative privilege levels

ONTAP commands and parameters are defined at three privilege levels: **admin**, **advanced**, and **diagnostic**. The privilege levels reflect the skill levels required in performing the tasks.

- **admin**
  
  Most commands and parameters are available at this level. They are used for common or routine tasks.

- **advanced**
  
  Commands and parameters at this level are used infrequently, require advanced knowledge, and can cause problems if used inappropriately.
  
  You use advanced commands or parameters only with the advice of support personnel.

- **diagnostic**
Diagnostic commands and parameters are potentially disruptive. They are used only by support personnel to diagnose and fix problems.

**Set the privilege level in the CLI**

You can set the privilege level in the CLI by using the `set` command. Changes to privilege level settings apply only to the session you are in. They are not persistent across sessions.

**Steps**

1. To set the privilege level in the CLI, use the `set` command with the `-privilege` parameter.

**Example of setting the privilege level**

The following example sets the privilege level to advanced and then to admin:

```
cluster1::> set -privilege advanced
Warning: These advanced commands are potentially dangerous; use them only when directed to do so by technical support.
Do you wish to continue? (y or n): y
cluster1::*> set -privilege admin
```

**Set display preferences in the CLI**

You can set display preferences for a CLI session by using the `set` command and `rows` command. The preferences you set apply only to the session you are in. They are not persistent across sessions.

**About this task**

You can set the following CLI display preferences:

- The privilege level of the command session
- Whether confirmations are issued for potentially disruptive commands
- Whether `show` commands display all fields
- The character or characters to use as the field separator
- The default unit when reporting data sizes
- The number of rows the screen displays in the current CLI session before the interface pauses output

If the preferred number of rows is not specified, it is automatically adjusted based on the actual height of the terminal. If the actual height is undefined, the default number of rows is 24.

- The default storage virtual machine (SVM) or node
- Whether a continuing command should stop if it encounters an error

**Steps**

1. To set CLI display preferences, use the `set` command.
To set the number of rows the screen displays in the current CLI session, you can also use the `rows` command.

For more information, see the man pages for the `set` command and `rows` command.

**Example of setting display preferences in the CLI**

The following example sets a comma to be the field separator, sets GB as the default data-size unit, and sets the number of rows to 50:

```
cluster1::> set -showseparator "," -units GB
cluster1::> rows 50
```

**Methods of using query operators**

The management interface supports queries and UNIX-style patterns and wildcards to enable you to match multiple values in command-parameter arguments.

The following table describes the supported query operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>*</td>
<td>Wildcard that matches all entries.</td>
</tr>
<tr>
<td>!</td>
<td>NOT operator. Indicates a value that is not to be matched; for example, !vs0 indicates not to match the value vs0.</td>
</tr>
<tr>
<td></td>
<td>OR operator. Separates two values that are to be compared; for example, vs0</td>
</tr>
<tr>
<td>.</td>
<td>Range operator. For example, 5..10 matches any value from 5 to 10, inclusive.</td>
</tr>
<tr>
<td>&lt;</td>
<td>Less-than operator. For example, &lt;20 matches any value that is less than 20.</td>
</tr>
<tr>
<td>Operator</td>
<td>Description</td>
</tr>
<tr>
<td>----------</td>
<td>-------------</td>
</tr>
<tr>
<td>&gt;</td>
<td>Greater-than operator. For example, ( &gt;5 ) matches any value that is greater than 5.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Less-than-or-equal-to operator. For example, ( \leq 5 ) matches any value that is less than or equal to 5.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater-than-or-equal-to operator. For example, ( \geq 5 ) matches any value that is greater than or equal to 5.</td>
</tr>
<tr>
<td>{query}</td>
<td>Extended query. An extended query must be specified as the first argument after the command name, before any other parameters. For example, the command <code>volume modify {-volume *tmp*} -state offline</code> sets offline all volumes whose names include the string tmp.</td>
</tr>
</tbody>
</table>

If you want to parse query characters as literals, you must enclose the characters in double quotes (for example, "\^", ".", "\*", or "\$") for the correct results to be returned.

You can use multiple query operators in one command line. For example, the command `volume show -size >1GB -percent-used <50 -vserver !vs1` displays all volumes that are greater than 1 GB in size, less than 50% utilized, and not in the storage virtual machine (SVM) named “vs1”.

**Methods of using extended queries**

You can use extended queries to match and perform operations on objects that have specified values.

You specify extended queries by enclosing them within curly brackets ({}). An extended query must be specified as the first argument after the command name, before any other parameters. For example, to set offline all volumes whose names include the string tmp, you run the command in the following example:

```
cluster1::> volume modify {-volume *tmp*} -state offline
```

Extended queries are generally useful only with `modify` and `delete` commands. They have no meaning in `create` or `show` commands.

The combination of queries and modify operations is a useful tool. However, it can potentially cause confusion and errors if implemented incorrectly. For example, using the (advanced privilege) `system node image modify` command to set a node’s default software image automatically sets the other software image not to be the default. The command in the following example is effectively a null operation:
cluster1:*> system node image modify {-isdefault true} -isdefault false

This command sets the current default image as the non-default image, then sets the new default image (the previous non-default image) to the non-default image, resulting in the original default settings being retained. To perform the operation correctly, you can use the command as given in the following example:

cluster1:*> system node image modify {-iscurrent false} -isdefault true

Methods of customizing show command output by using fields

When you use the –instance parameter with a show command to display details, the output can be lengthy and include more information than you need. The –fields parameter of a show command enables you to display only the information you specify.

For example, running volume show –instance is likely to result in several screens of information. You can use volume show –fields fieldname[,fieldname...] to customize the output so that it includes only the specified field or fields (in addition to the default fields that are always displayed.) You can use –fields ? to display valid fields for a show command.

The following example shows the output difference between the –instance parameter and the –fields parameter:
cluster1::> volume show -instance

Vserver Name: cluster1-1
Volume Name: vol0
Aggregate Name: aggr0
Volume Size: 348.3GB
Volume Data Set ID: -
Volume Master Data Set ID: -
Volume State: online
Volume Type: RW
Volume Style: flex
...
Space Guarantee Style: volume
Space Guarantee in Effect: true
...
Press <space> to page down, <return> for next line, or 'q' to quit...
...
cluster1::>

cluster1::> volume show -fields space-guarantee,space-guarantee-enabled

vserver volume space-guarantee space-guarantee-enabled
-------- ------ --------------- -----------------------
cluster1-1 vol0   volume          true
cluster1-2 vol0   volume          true
vs1   root_vol
    volume          true
vs2   new_vol
    volume          true
vs2   root_vol
    volume          true
...
cluster1::>

About positional parameters

You can take advantage of the positional parameter functionality of the ONTAP CLI to increase efficiency in command input. You can query a command to identify parameters that are positional for the command.

What a positional parameter is

- A positional parameter is a parameter that does not require you to specify the parameter name before specifying the parameter value.
- A positional parameter can be interspersed with nonpositional parameters in the command input, as long
as it observes its relative sequence with other positional parameters in the same command, as indicated in
the `command_name` ? output.

- A positional parameter can be a required or optional parameter for a command.
- A parameter can be positional for one command but nonpositional for another.

Using the positional parameter functionality in scripts is not recommended, especially when the
positional parameters are optional for the command or have optional parameters listed before
them.

### Identify a positional parameter

You can identify a positional parameter in the `command_name` ? command output. A positional parameter has
square brackets surrounding its parameter name, in one of the following formats:

- `[-parameter_name] parameter_value` shows a required parameter that is positional.
- `[[parameter_name] parameter_value]` shows an optional parameter that is positional.

For example, when displayed as the following in the `command_name` ? output, the parameter is positional for
the command it appears in:

- `[-lif] <lif-name>
- `[[lif] <lif-name>]

However, when displayed as the following, the parameter is nonpositional for the command it appears in:

- `-lif <lif-name>
- `[-lif <lif-name>]

### Examples of using positional parameters

In the following example, the `volume create` ? output shows that three parameters are positional for the
command: `-volume`, `-aggregate`, and `-size.`
<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>-vserver &lt;vserver name&gt;</code></td>
<td>Vserver Name</td>
</tr>
<tr>
<td><code>[-volume] &lt;volume name&gt;</code></td>
<td>Volume Name</td>
</tr>
<tr>
<td><code>[-aggregate] &lt;aggregate name&gt;</code></td>
<td>Aggregate Name</td>
</tr>
<tr>
<td>`[[-size] {&lt;integer&gt;[KB</td>
<td>MB</td>
</tr>
<tr>
<td>`[ -state {online</td>
<td>restricted</td>
</tr>
<tr>
<td>`[ -type {RW</td>
<td>DP</td>
</tr>
<tr>
<td><code>[ -policy &lt;text&gt; ]</code></td>
<td>Export Policy</td>
</tr>
<tr>
<td><code>[ -user &lt;user name&gt; ]</code></td>
<td>User ID</td>
</tr>
<tr>
<td>`[ -space-guarantee</td>
<td>-s {none</td>
</tr>
<tr>
<td><code>[ -percent-snapshot-space &lt;percent&gt; ]</code></td>
<td>Space Reserved for Snapshot Copies</td>
</tr>
</tbody>
</table>

In the following example, the `volume create` command is specified without taking advantage of the positional parameter functionality:

```
cluster1::> volume create -vserver svm1 -volume vol1 -aggregate aggr1 -size 1g -percent-snapshot-space 0
```

The following examples use the positional parameter functionality to increase the efficiency of the command input. The positional parameters are interspersed with nonpositional parameters in the `volume create` command, and the positional parameter values are specified without the parameter names. The positional parameters are specified in the same sequence indicated by the `volume create ?` output. That is, the value for `-volume` is specified before that of `-aggregate`, which is in turn specified before that of `-size`.

```
cluster1::> volume create vol2 aggr1 1g -vserver svm1 -percent-snapshot-space 0
cluster1::> volume create -vserver svml vol3 -snapshot-policy default aggr1 -nvfail off 1g -space-guarantee none
```

### Methods of accessing ONTAP man pages

ONTAP manual (man) pages explain how to use ONTAP commands. They are available at the command line and on the NetApp Support Site.

The `man command_name` command displays the manual page of the specified command. If you do not specify a command name, the manual page index is displayed. You can use the `man man` command to view information about the `man` command itself. You can exit a man page by entering `q`.

The ONTAP 9 manual pages contains command references for the admin-level and advanced-level ONTAP commands.