Contribute to NetApp product documentation in GitHub

Contributor's Guide

NetApp
September 18, 2025

This PDF was generated from https://docs.netapp.com/us-en/contribute/index.html on September 18, 2025. Always check docs.netapp.com for the latest.

Table of Contents

Contribute to NetApp product documentation in GitHub

The documentation for NetApp's products and services is open source. This enables you to contribute to the content by making improvements, corrections, and suggestions. All you need is a GitHub account and a little initiative.

Overview

You can contribute to our docs by using the following options:

- Select **Request doc changes** to submit general feedback or to ask a question about the content. The NetApp content lead then reviews your request to determine what changes are needed to the docs. This is the most common option.

  View step-by-step instructions for this option.

- Select **Edit this page** to directly edit the content yourself. The NetApp content lead then reviews your edits and merges them.

  View step-by-step instructions for this option.

The following video provides a brief overview of these two options.

Overview of contributing to docs.netapp.com

The sections below provide step-by-step instructions.

Request doc changes

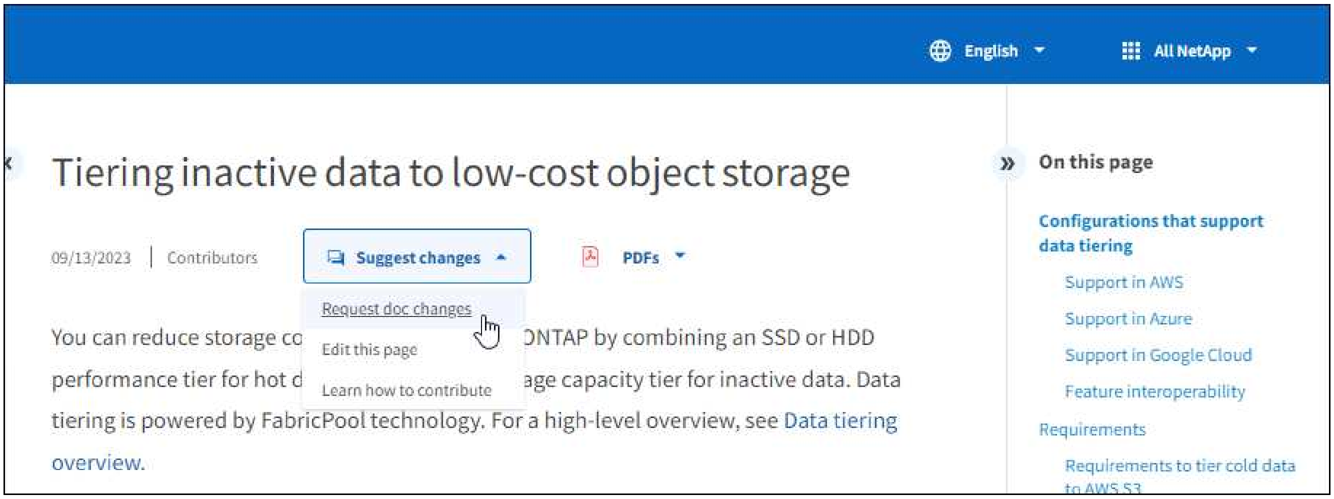
Submitting a request for a doc change is the most common way to contribute to NetApp docs. After you submit the request, the content lead will confirm that they've received your feedback. You'll get an email notification from GitHub when they do.

If the content lead agrees that your suggestion can make the content better, they will commit the change shortly after that. You'll receive another notification that your feedback was incorporated.

> All comments that you provide are publicly visible. Anyone who navigates to the issues in the GitHub repo can see your comments.

Steps

1. If you don't already have a GitHub account, create one from github.com
2. Log in to your GitHub account.
3. Using your web browser, open the page on docs.netapp.com that's related to your feedback.
4. At the top of the page, select **Suggest changes > Request doc changes**.

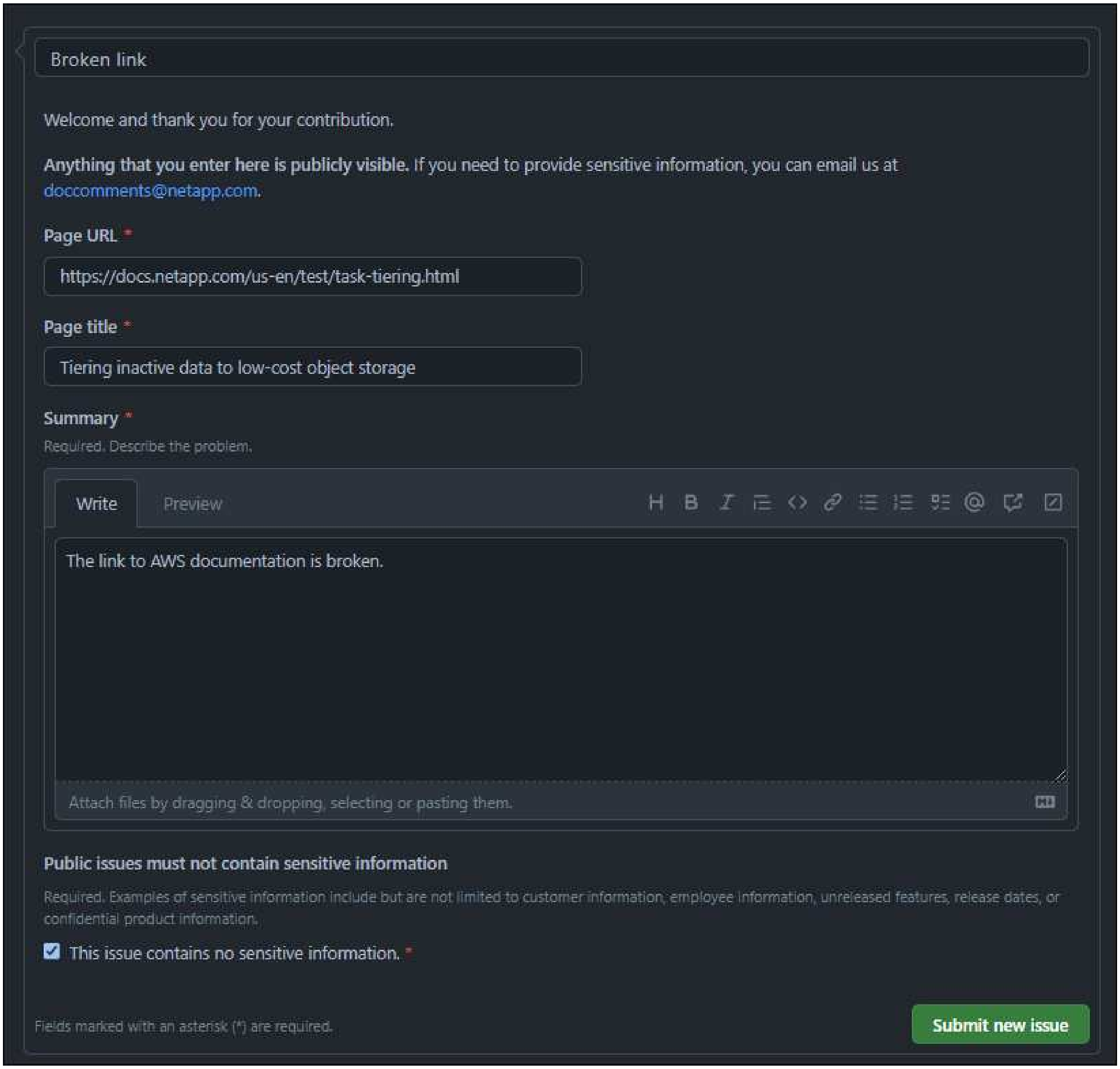


You can reduce storage co ONTAP by combining an SSD or HDD performance tier for hot d age capacity tier for inactive data. Data tiering is powered by FabricPool technology. For a high-level overview, see Data tiering overview.

A new browser tab opens with a GitHub form that you can use to provide details to our doc team.

5. Enter a title, summary, and acknowledge that the issue doesn’t contain any sensitive information.

   The form is prepopulated with the URL and title of the page. Don’t delete this information because we’ll need it to understand your request.



Broken link

Welcome and thank you for your contribution.

Anything that you enter here is publicly visible. If you need to provide sensitive information, you can email us at doccomments@netapp.com.

Page URL *

https://docs.netapp.com/us-en/test/task-tiering.html

Page title *

Tiering inactive data to low-cost object storage

Summary *

Required. Describe the problem.

The link to AWS documentation is broken.

Attach files by dragging & dropping, selecting or pasting them.

Public issues must not contain sensitive information

Required. Examples of sensitive information include but are not limited to customer information, employee information, unreleased features, release dates, or confidential product information.

☑ This issue contains no sensitive information. *

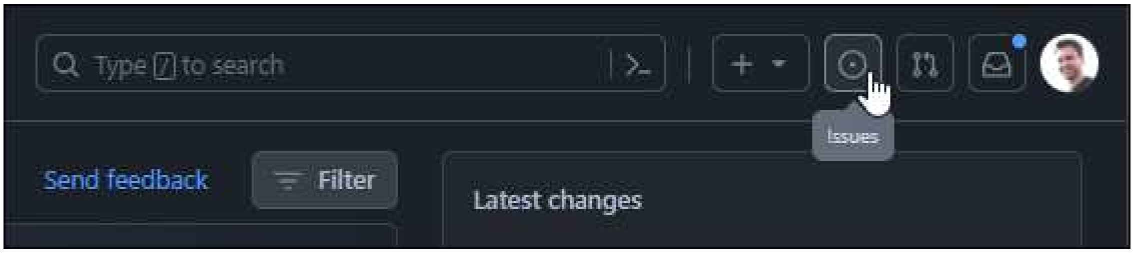
Fields marked with an asterisk (*) are required.

6. Select **Submit new issue** to create an issue for your request.

After you’re done

Opening an issue enables collaboration through GitHub comments. You’ll receive email notifications based on the preferences that you indicated in your GitHub account settings.

You can also view the status of the request by selecting **Issues** from the GitHub banner:



Submit edits to the docs

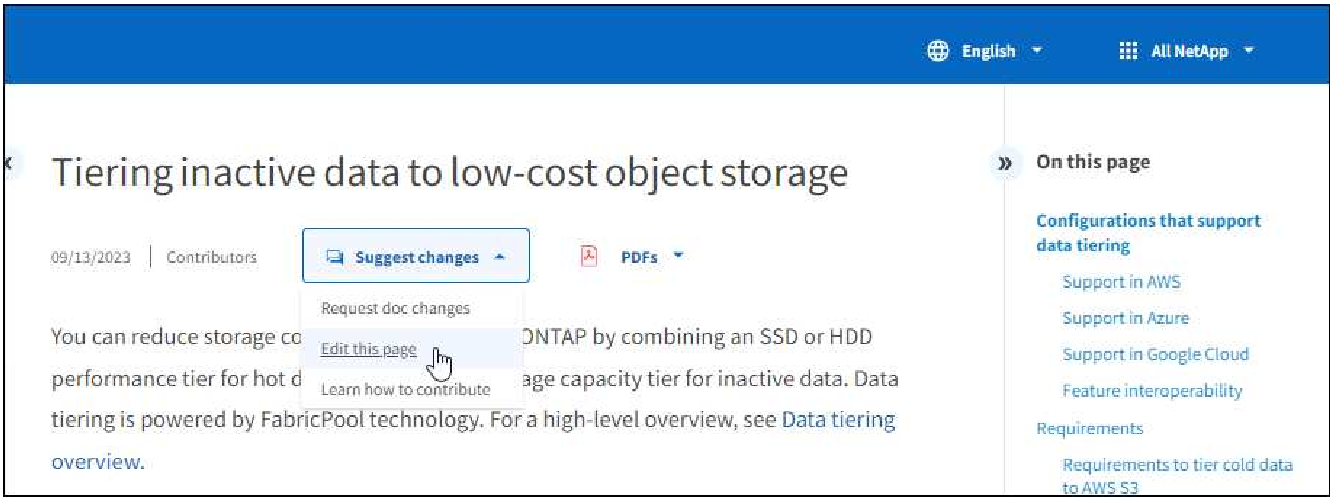
If you’re comfortable editing the content yourself, you can submit the exact doc changes that you’d like to see by directly editing the source file.

As an outside contributor, you won’t be able to publish the change directly. The content lead will review the changes, make any required edits, and then merge the changes. You’ll get an email notification from GitHub when this happens.

If you need help editing the source file, you can learn about AsciiDoc syntax.

Steps

1. If you don’t already have a GitHub account, create one from github.com
2. Log in to your GitHub account.
3. Using your web browser, open the page on docs.netapp.com that you want to edit.
4. At the top of the page, select **Suggest changes > Edit this page**.

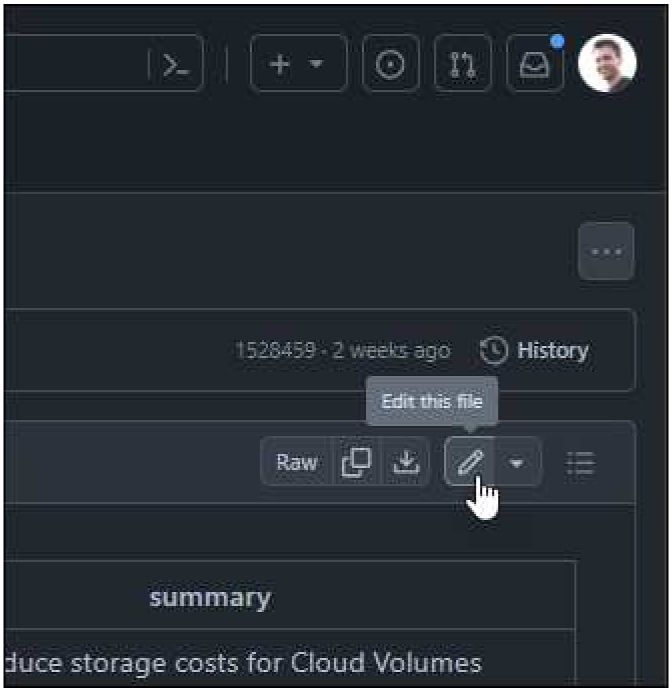


Tiering inactive data to low-cost object storage

You can reduce storage co ONTAP by combining an SSD or HDD performance tier for hot d age capacity tier for inactive data. Data tiering is powered by FabricPool technology. For a high-level overview, see Data tiering overview.

A new browser tab opens and brings you to the file in the GitHub repository for the documentation site.

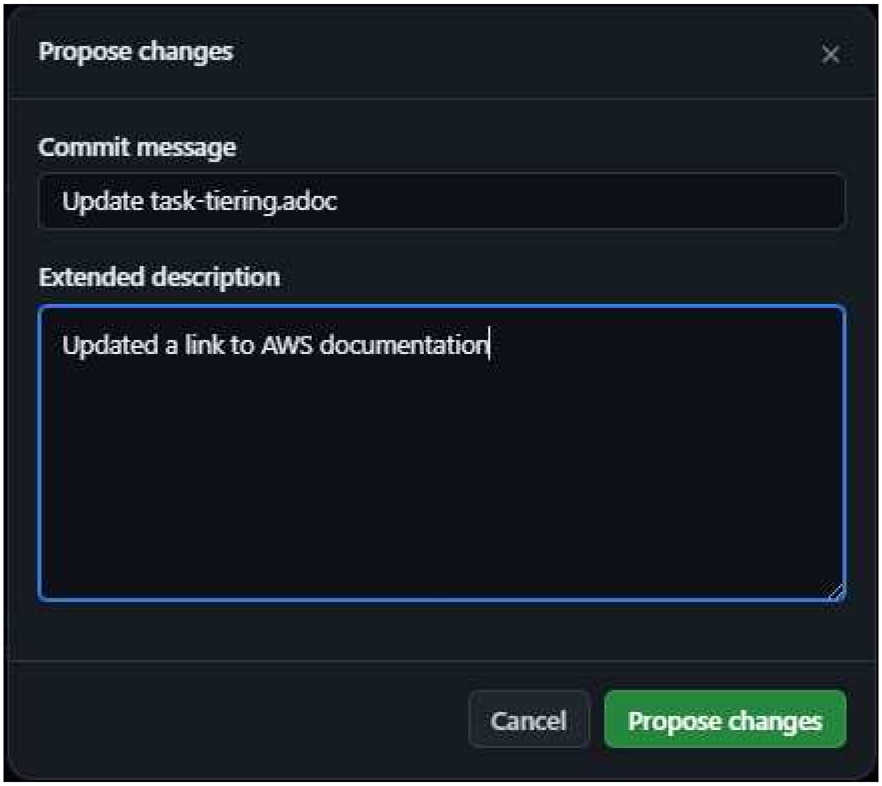
5. Select the pencil icon.



6. If you’re prompted to create a fork of the repository, select **Fork this repository**.
7. Edit the content.

   The content is written in AsciiDoc, a lightweight markup language. Learn about AsciiDoc syntax.

8. To commit your changes, select **Commit changes** and fill out the form:
   a. Optionally modify the default commit message.
   b. Add an optional description.
   c. Select **Propose changes**.



Commit message

Update task-tiering.adoc

Extended description

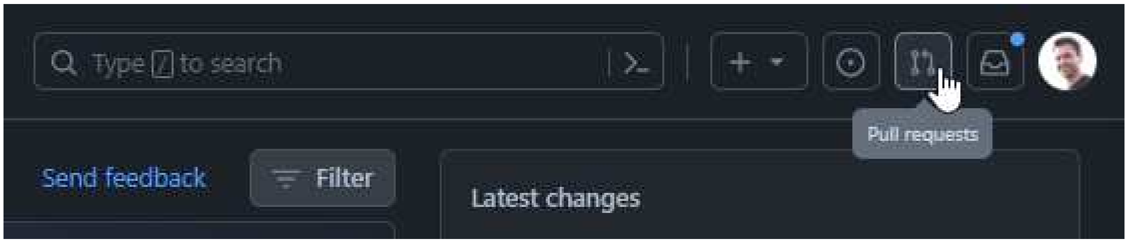
Updated a link to AWS documentation

9. Select **Create pull request**.

After you’re done

After you propose the changes, we’ll review them, make any necessary edits, and then merge the changes into the GitHub repository.

You can view the status of the pull request by selecting **Pull requests** from the GitHub banner:



AsciiDoc reference

AsciiDoc is a lightweight markup language, similar to Markdown. We chose AsciiDoc over standard Markdown because it provides more out-of-box capabilities. While it’s more powerful, it’s still simple to use. Refer to the sections below to get started writing in AsciiDoc.

See the AsciiDoctor User Manual for additional help.

The basics

You need to know a few things to contribute simple doc updates.

Headings

```
= Page title
== Level 1 section
=== Level 2 section
==== Level 3 section
===== Level 4 section
```

You can have only one page title, but you can have multiple section titles. For example, you might have three level 1 sections that include level 2 and 3 sections:

```
= Page title
== Level 1 section
=== Level 2 section
== Level 1 section
== Level 1 section
=== Level 2 section
==== Level 3 section
```

Bold text

```
*Text*
```

Italic text

```
_Text_
```

Bulleted lists

```
* Item 1
+
Continuation text for the previous list item.

* Item 2
** Item 2a

* Item 3
```

The + is a list continuation. It keeps the text inline with the list item. Omitting the + affects the formatting of that line.

Labeled lists

```
Item 1::
Description 1

Item 2::
Description 2
```

or

```
[horizontal]
Item 1::
Description 1

Item 2::
Description 2
```

When you add [horizontal] above item 1, the label and description appear on the same line. That works well when you have very short descriptions.

Example without [horizontal]

Item 1

> Description 1

Item 2

> Description 2

Example with [horizontal]

Item 1	Description 1
Item 2	Description 2

Steps

```
.Steps

. Step 1

. Step 2
+
Info for step 2

. Step 3
.. Step 3a
.. Step 3b

. Step 4
```

The + is a list continuation. It keeps the text inline with the list item. Omitting the + affects the formatting of that line.

Images

```
image:file.png["alt text"]
```

alt text means alternative text. It describes the image that appears on the page. The primary use is for visually-impaired users who use screen readers.

Two notes:

1. It’s best to enclose alt text in quotes because punctuation like commas can affect the ability to transform the content from AsciiDoc to HTML.
2. The AsciiDoctor docs state that *block images* should be on their own line with *two* colons: `image::file.png`

   But we prefer to use one colon, as shown above. Using one colon has the same result and it works better with our internal tools.

Videos

Hosted on YouTube:

```
video::id[youtube]
```

Hosted locally in GitHub:

```
video::https://docs.netapp.com/us-en/contribute//media/file.mp4
```

Links

The syntax that you should use depends on what you're linking to:

- Link to an external site
- Link to a section on the same page
- Link to another page in the docs

Link to an external site

```
url[link text^]
```

The ^ opens the link in a new browser tab.

Link to a section on the same page

```
<<section_title>>
```

For example:

```
For more details, see <<Headings>>.
```

The link text can be something other than the section title:

```
<<section_title,Different link text>>
```

For example:

```
<<Headings,Learn the syntax for headings>>.
```

Link to another page in the docs

The file needs to be in the same GitHub repository:

```
xref:{relative_path}<file_name>.html[Link text]
```

To link directly to a section in the file, add a hash (#) and the section's title:

```
xref:{relative_path}<file_name>.html#<section-name-using-dashes-and-all-
lower-case>[Link text]
```

For example:

```
xref:{relative_path}style.html#use-simple-words[Use simple words]
```

Notes, tips, and cautions

You might want to draw attention to certain statements by using notes, tips, or caution statements. Format them as follows:

```
NOTE: text

TIP: text

CAUTION: text
```

Use each of these sparingly. You don't want to create pages that are full of notes and tips. They become less meaningful if you do.

Here's what each of these looks like when the AsciiDoc content is turned into HTML:

ⓘ This is a note. It includes extra info that a reader might need to know.

💡 A tip provides useful information that can help a user do something or understand something.

❗ A caution advises the reader to act carefully. Use this in rare circumstances.

Advanced stuff

If you're authoring new content, you'll want to review this section for some nitty-gritty details.

Document headers

Each AsciiDoc file includes two types of headers. The first is for GitHub and the second is for AsciiDoctor, which is the publishing tool that turns the AsciiDoc content into HTML.

The GitHub header is the very first set of content in the .adoc file. It needs to include the following:

```
---
sidebar: sidebar
permalink: <file_name>.html
keywords: keyword1, keyword2, keyword3, keyword4, keyword5
summary: "A summary."
---
```

The keywords and summary directly affect search results. In fact, the summary itself displays in the search results. You should make sure that it’s user friendly. The best practice is to have the summary mirror your lead paragraph.

It’s best to enclose the summary in quotes because punctuation like colons can affect the ability to transform the content from AsciiDoc to HTML.

The next header goes directly underneath the document title (see Headings). This header should include the following:

```
:hardbreaks:
:icons: font
:linkattrs:
:relative_path: ./
:imagesdir: {root_path}{relative_path}./media/
```

You won’t need to touch any of the parameters in this heading. Just paste it in and forget it.

Lead paragraph

The first paragraph that appears under the document title should include the following syntax directly above it:

```
[.lead]
This is my lead paragraph for this content.
```

[.lead] applies CSS formatting to the lead paragraph, which has a different format than the text that follows it.

Tables

Here’s syntax for a basic table:

```
[cols=2*,options="header",cols="25,75"]
|===
| heading column 1
| heading column 2
| row 1 column 1 | row 1 column 2
| row 2 column 1 | row 2 column 2
|===
```

There are *many* ways to format a table. Refer to the AsciiDoctor User Manual for additional help.

If a cell contains formatted content like bulleted lists, it’s best to add an "a" in the column header to enable formatting. For example:
[cols="2,2,4a" options="header"]

See the AsciiDoc Syntax Quick Reference for more table examples.

Task headings

If you’re explaining how to perform a task, you might include introductory information before you get to the steps. And you might need to say what to do after completing the steps. If you do, it’s best to organize that information using headers, which enables scanning.

Use the following headings as needed:

What you’ll need

The information the user needs to complete the task.

About this task

Some extra contextual info the user might need to know about this task.

Steps

The individual steps to complete the task.

What’s next?

What the user should do next.

Each of these should include a . right before the text, like so:

```
.What you'll need
.About this task
.Steps
.What's next?
```

This syntax applies bold text in a larger font.

Command syntax

When providing command input, enclose the command within ` to apply monospace font:

```
`volume show -is-encrypted true`
```

Here’s what that looks like:

`volume show -is-encrypted true`

For command output or command examples, use the following syntax:

```
----
cluster2::> volume show -is-encrypted true

Vserver  Volume  Aggregate  State  Type  Size  Available  Used
-------  ------  ---------  -----  ----  -----  --------- ----
vs1      vol1    aggr2     online    RW  200GB    160.0GB  20%
----
```

The four dashes enable you to enter separate lines of text that appear together.
Here’s the result:

```
cluster2::> volume show -is-encrypted true

Vserver  Volume  Aggregate  State  Type  Size  Available  Used
-------  ------  ---------  -----  ----  -----  --------- ----
vs1      vol1    aggr2     online    RW  200GB    160.0GB  20%
```

Variable text

In commands and command output, enclose variable text within underscores to apply italics.

```
`vserver nfs modify -vserver _name_ -showmount enabled`
```

Here’s what that command and the variable text looks like:

`vserver nfs modify -vserver` *`name`* `-showmount enabled`

Underscores aren’t supported with code syntax highlighting at this time.

Code syntax highlighting

Code syntax highlighting provides a developer-focused solution for documenting the most popular languages.

Output example 1

```
POST https://netapp-cloud-account.auth0.com/oauth/token
Header: Content-Type: application/json
Body:
{
              "username": "<user_email>",
              "scope": "profile",
              "audience": "https://api.cloud.netapp.com",
              "client_id": "UaVhOIXMWQs5i1WdDxauXe5Mqkb34NJQ",
              "grant_type": "password",
              "password": "<user_password>"
}
```

Output example 2

```
[
    {
        "header": {
            "requestId": "init",
            "clientId": "init",
            "agentId": "init"
        },
        "payload": {
            "init": {}
        },
        "id": "5801"
    }
]
```

Supported languages

- bash
- curl
- https
- json
- powershell
- puppet
- python
- yaml

Implementation

Copy and paste the following syntax and then add a supported language and the code:

```
[source,<language>]
<code>
```

For example:

```
[source,curl]
curl -s https:///v1/ \
-H accept:application/json \
-H "Content-type: application/json" \
-H api-key: \
-H secret-key: \
-X [GET,POST,PUT,DELETE]
```

Content reuse

If you have a chunk of content that's repeated across different pages, you can easily write it once and reuse it across those pages. Reuse is possible from within the same repository and across repositories. Here's how it works.

1. Create a folder in your repository named _include
2. Add a .adoc file in that folder that includes the content that you'd like to reuse.

   It can be a sentence, a list, a table, one or more sections, and so on. Don't include anything else in the file—no headers or anything.

3. Now go to the files where you'd like to reuse that content.
4. If you're reusing the content from within the *same* GitHub repository, use the following syntax on a line by itself:

   ```
   include::_include/<filename>.adoc[]
   ```

   For example:

   ```
   include::_include/s3regions.adoc[]
   ```

5. If you're reusing the content in a *different* repository, use the following syntax on a line by itself:

   ```
   include::https://raw.githubusercontent.com/NetAppDocs/<reponame>/main/_i
   nclude/<filename>.adoc[]
   ```

   That's it!

If you want to learn more about the include directive, check out the AsciiDoctor User Manual.

Policy forbidding the use of AI-generated content in NetApp technical documentation

At NetApp, we value the expertise of our employees, contractors, and partners in creating, publishing, and maintaining accurate, relevant, useful technical documentation for our customers. We believe that the use of AI-generated content can compromise these values and undermine the integrity and accuracy of our documentation. Therefore, NetApp's policy is to prohibit publication of AI-generated content that has not been verified and validated by a human. This applies to all technical documentation published on docs.netapp.com. Note that this policy is similar to and extends NetApp's policy that forbids the use of languages models to write code for the products it sells.

We expect all NetAppDocs contributors to adhere to the following policy:

- We do not allow publication of customer-facing documentation (how to use NetApp products) generated from AI without human review and validation. This includes written materials, graphics, multimedia, or other materials that are published on docs.netapp.com. All content published on docs.netapp.com must be verified and validated by human content creators.
- Part of the role of content creator for docs.netapp.com involves adhering to the style and contributor guides. This includes staying informed about the latest developments in AI-generated content and their implications for NetApp technical documentation.
- All employees, contractors, and partners are expected to comply with the policy. Repeated violations may result in suspended authoring privileges.

NetAppDocs contributors should keep the following in mind:

- All AI-based content-generating technologies embedded in software you have installed on your work computer, including ones like Microsoft 365 Co-pilot, send your data to the company that hosts the assistive technology. Do not prompt such a technology with company confidential content, including but not limited to specifications for unreleased product features.
- Do not use public AI technologies to generate or edit technical documentation for **previously unreleased** products, versions of products, or features within products.
- Copyright for new technical documentation must remain within NetApp and not be shared with AIs outside NetApp prior to official product release dates and the corresponding publication of related technical documentation.
- Do not send confidential documentation or newly-written content into AI (ChatGPT or any other tool). Any information fed into ChatGPT becomes a part of its training dataset, which in turn becomes property of OpenAI and can be used by them and their users. While much of this data is publicly accessible, we do not want to train the model on our own IP.

Legal notices

Legal notices provide access to copyright statements, trademarks, patents, and more.

Copyright

https://www.netapp.com/company/legal/copyright/

Trademarks

NETAPP, the NETAPP logo, and the marks listed on the NetApp Trademarks page are trademarks of NetApp, Inc. Other company and product names may be trademarks of their respective owners.

https://www.netapp.com/company/legal/trademarks/

Patents

A current list of NetApp owned patents can be found at:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Privacy policy

https://www.netapp.com/company/legal/privacy-policy/