

SUSE TRD

SUSE Technical Reference Documentation Contributors Guide

How to Create Documentation for SUSE TRD

SUSE Technical Reference Documentation

Terry Smith, Global Partner Solutions Director (SUSE)
Bryan Gartner, Senior Technology Strategist (SUSE)

SUSE Technical Reference Documentation Contributors Guide

How to Create Documentation for SUSE TRD

Date: 2023-12-18

Summary

This document provides guidance for contributing to SUSE Technical Reference Documentation.

Disclaimer

Documents published as part of the series SUSE Technical Reference Documentation have been contributed voluntarily by SUSE employees and third parties. They are meant to serve as examples of how particular actions can be performed. They have been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. SUSE cannot verify that actions described in these documents do what is claimed or whether actions described have unintended consequences. SUSE LLC, its affiliates, the authors, and the translators may not be held liable for possible errors or the consequences thereof.

Contents

1	Introduction	4
2	Prerequisites	5
3	Terminology	5
4	Workflow	7
5	Repository structure	17
6	Templates and framework	20
7	Style	36
8	AsciiDoc	37
9	DAPS	50
10	Summary	56
11	Legal notice	57
12	GNU Free Documentation License	58

1 Introduction

1.1 Motivation

SUSE Technical Reference Documentation (<https://documentation.suse.com/?tab=trd>) (SUSE TRD) is an open collection of high quality guidance for solutions that address real-world use cases with featured SUSE, partner, and community components. Documents are modular and rendered from multiple source files. These source files are freely available and open source licensed in the [GitHub repository \(https://github.com/SUSE/technical-reference-documentation\)](https://github.com/SUSE/technical-reference-documentation).

SUSE TRD can be generally divided into four types:

- **Getting Started Guide**

Solution introduction with step-by-step guidance for installation and configuration.

- **Reference Implementation**

Introductory architectural approach and basis for deployment of solution with the SUSE portfolio.

- **Reference Configuration**

Reference Implementation with specified partner hardware and software components.

- **Enterprise Architecture**

Architectural overview of solution from an enterprise landscape perspective.

Community engagement with SUSE TRD is encouraged. You can easily submit quick fixes and even section updates through *Report an issue* links in the HTML version of any document.

If you have an interest in contributing a new document, you first should become familiar with the style, tools, and workflows of a successful contributor.

1.2 Scope

This guide outlines style, tools, and workflows for successful contributions to SUSE TRD.

1.3 Audience

This document is intended for architects, engineers, and technical writers who would like to contribute to SUSE TRD.

2 Prerequisites

Contributors to SUSE TRD need the following:

- A plain text editor



Note

Your editor should save files encoded to [UTF-8](https://en.wikipedia.org/wiki/UTF-8) and end lines only with the line feed character.

See also [Configuring Git to handle line endings](https://docs.github.com/en/get-started/git-basics/configuring-git-to-handle-line-endings?versionId=free-pro-team%40latest&productId=get-started&platform=windows).

- [GitHub](https://github.com/) account and basic Git skills
- Basic understanding of the [AsciiDoc](https://asciidoc.org/) markup language
See [Section 8, "AsciiDoc"](#).
- DAPS toolchain (for rendering documents)
You have two options:
 - Install [DAPS](https://opensuse.github.io/daps/doc/cha.daps.user.inst.html#sec.daps.user.inst) and its dependencies directly, including the latest [SUSE XSL Stylesheets](https://github.com/openSUSE/suse-xsl).
 - Use the [DAPS2Docker](https://github.com/openSUSE/daps2docker) container implementation.

See [Section 9, "DAPS"](#) for more information.

- PDF viewer and Web browser (for reviewing rendered documents)

3 Terminology

Branch

A branch represents an independent line of development. It allows you to make edits and commits without affecting the primary or 'main' branch of repository. Your branch must be merged into the 'main' branch of the *upstream* repository to be published.

Clone

A clone is a copy of a repository, typically hosted on your local system.

Commit

A commit represents a specific revision or version in your project's Git history.

Fork

A fork is a copy of a repository. When you fork a repository, you create a copy in which you can safely make changes without affecting the original project. For this guide, your fork of SUSE TRD is hosted in your GitHub account.

Git

[Git \(https://git-scm.com/\)](https://git-scm.com/)  is a free and open source, distributed version control system.

GitHub

[GitHub \(https://github.com/\)](https://github.com/)  is a developer platform and service powered by Git and used by millions of developers to build, scale, and deliver secure software. SUSE TRD is hosted on GitHub to ease collaboration and contributions from the global community.

Pull request

A pull request is submission to a repository's collaborators to accept your proposed changes. In the SUSE TRD workflow, you create a branch, make all your changes in this branch, then create a pull request to the *upstream* repository. Once your pull request is accepted, it is merged into the *upstream* 'main' branch.

Remote


A remote is a repository that is used to track a project but resides somewhere else. For the workflow outlined in this guide, you have two remotes:

- your fork of the SUSE TRD repository on GitHub, referred to by the name, *origin*.
- the SUSE TRD repository on GitHub, referred to by *upstream*.

Repository

A repository is a data structure containing files, directories, and metadata about the project (such as commit history, tags, branches, authors, and more). The SUSE TRD repository hosted on GitHub contains the source files for the published documents, history of changes in those documents, and more.

4 Workflow

The SUSE TRD [GitHub repository \(https://github.com/SUSE/technical-reference-documentation\)](https://github.com/SUSE/technical-reference-documentation)  is an organized collection of source files for each document. You can learn more about types of source files and how they are organized in [Section 5, "Repository structure"](#).

Contributors use a Git workflow with the following major steps:

1. Fork the SUSE TRD repository.
2. Clone the fork.
3. Make a branch.
4. Create content.
5. Submit a pull request.

Each of these steps is explored in more detail in the following sections.


4.1 Fork the SUSE TRD repository

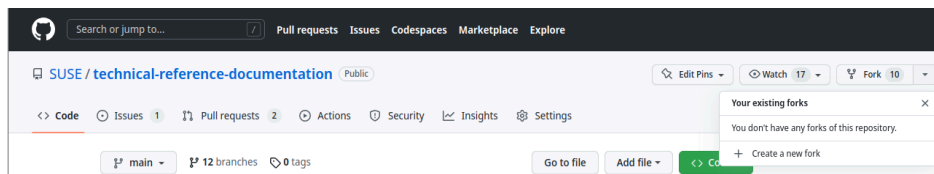
The first step is to create a fork of the SUSE TRD repository into your own GitHub account. This only needs to be done once.



Note

If you work for SUSE or a partner company, use your company-sanctioned GitHub account.

1. Open your Web browser and log into your GitHub account.
2. Navigate to the SUSE TRD [GitHub repository \(https://github.com/SUSE/technical-reference-documentation\)](https://github.com/SUSE/technical-reference-documentation) .
3. Fork this repository into your own account.
 - a. In the GitHub UI, click *Fork*, then *+ Create a new fork*.



- b. In the *Create a new fork* dialog, you can keep the defaults.

Create a new fork

A *fork* is a copy of a repository. Forking a repository allows you to freely experiment with changes without affecting the original project. [View existing forks.](#)

Owner * **Repository name ***

geeko / technical-reference-documentatik ✓

By default, forks are named the same as their upstream repository. You can customize the name to distinguish it further.

Description (optional)

Tux's SUSE Technical Reference Documentation

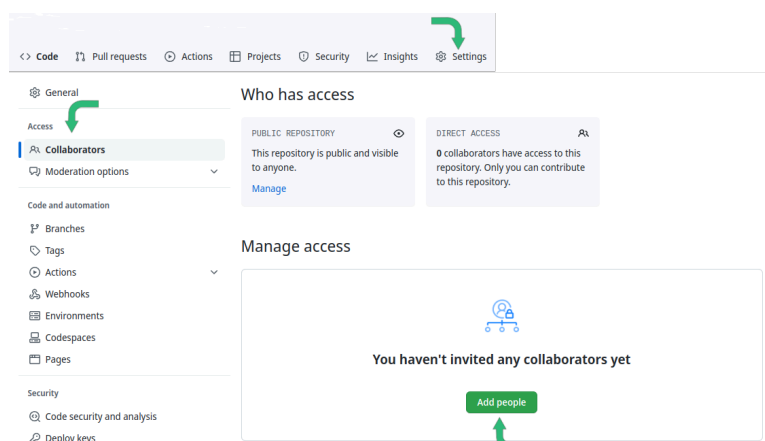
☒ **Copy the `main` branch only**
Contribute back to SUSE/technical-reference-documentation by adding your own branch. [Learn more.](#)

i You are creating a fork in your personal account.

Create fork

- c. Click *Create fork*.

4. If you are working with others on your document, you can invite them to collaborate in your repository fork.





Note

Alternatively, others can fork your fork and follow a similar workflow and submit content and suggestions to you through pull requests.

- a. Navigate to your fork.
- b. Click *Settings* > *Collaborators* > *Add people*.
- c. Enter the GitHub username, full name, or email address of your collaborator into the search field, then select the individual you wish to add.



Warning

Be sure to verify that you select the correct individual.

SUSE employees should use approved GitHub accounts associated with their SUSE email addresses. Partner contributors should use GitHub accounts associated with their employer email account.

- d. Click *Add to this repository*.

4.2 Clone the fork

Create a clone of your fork so you can work on your local system. When you clone a Git repository to your local system, you download a full copy of all the repository data at that point in time. This includes not just source files but Git metadata too, including branches, commit history, and more.

1. Launch a terminal on your local system.
2. On the command line, change to the directory where you will store the clone, creating it if necessary.

For example,

```
mkdir -p ~/git/GITHUB_USER
cd ~/git/GITHUB_USER
```

where GITHUB_USER is your GitHub username.



Note

All forks of a Git repository share the same directory structure. Creating the additional directory layer gives you the flexibility to host clones of other forks of the *upstream* repository or even of the *upstream* repository itself.

3. Clone your fork.

```
git clone git@github.com:GITHUB_USER/technical-reference-documentation.git
```



Tip

Find the URL of your fork in the GitHub UI.

1. Log into GitHub and go to your fork.
2. Click `< > Code`.
3. Select the *Local* tab.
4. Click *SSH*.
5. Click the "copy" icon to copy the URL to your clipboard.

4. Enter the top-level directory of your clone.

```
cd technical-reference-documentation
```

5. Add the SUSE TRD repository as your *upstream* remote.

```
git remote add upstream git@github.com:SUSE/technical-reference-documentation.git
```



Note

Your clone's primary remote is named *origin* and it represents the fork in your GitHub account.

6. Verify your remotes.

```
git remote -v
```

```
origin  git@github.com:GITHUB_USER/technical-reference-documentation.git (fetch)
origin  git@github.com:GITHUB_USER/technical-reference-documentation.git (push)
upstream      git@github.com:SUSE/technical-reference-documentation.git (fetch)
upstream      git@github.com:SUSE/technical-reference-documentation.git (push)
```



Note

GITHUB_USER in this example would be replaced with your GitHub username.

4.3 Make a branch

Branches help you keep your work separate from that of others. Think of a branch as an independent, project work area. You create your document in a branch of your fork. Later, you submit a pull request to merge this branch into the 'main' *upstream* branch.

1. Create a branch for your project.

For example,

```
git branch my-tuxy-project
```

2. To work in this branch, you have to check it out.

```
git checkout my-tuxy-project
```



Tip

You can create and check out a branch at the same time with:

```
git checkout -b my-tuxy-project
```



Important

It is a good practice to make sure your branch is active at the start of each editing session. You can verify your current branch with:

```
git branch --show-current
```

4.4 Configure the framework

Your documentation project consists of multiple types of files, placed in specific locations within the directory structure of the SUSE TRD repository. Setting up this structure for your project can seem challenging. Fortunately, there are automation tools to help you quickly generate a compliant framework with templates for the required source files. See [Section 6, “Templates and framework”](#) to learn more.

4.5 Create content

Good documentation often results from collaborative efforts and multiple editing sessions. The following workflow can help you manage this process.

1. Enter the local directory containing the clone of your GitHub fork. For example:

```
cd ~/git/GITHUB_USER/technical-reference-documentation
```

2. Check out your project's branch.

```
git checkout my-tuxy-project
```



Tip

Always remember to work in your branch.

3. Update the clone from your *origin* remote.

This allows you to integrate any edits or other content from your contributors, helping you minimize merge conflicts later.

```
git pull origin
```



Note

It is not necessary to specify the *origin* remote, since it is configured as your default for tracking.



Important

Fix any merge errors before proceeding.

4. Create your content.

A typical content session involves editing source files and copying assets (such as image files) into appropriate project directories. Be sure to refer to [Section 7, “Style”](#) and [Section 8, “AsciiDoc”](#) to learn more about writing style and content formatting.

5. Render your document with DAPS to verify content, layout, and style.



Note

You may get validation errors if you have invalid AsciiDoc syntax. You must then find and correct these errors.

6. Commit your changes locally.

For example:

```
git add .
git commit -m "updated section 5; added screenshot"
```



Tip

Always include a commit message as a reminder to yourself and to let your collaboration team know what changes you made in this commit.

7. Push the commit to your *origin* remote.

```
git push origin
```



Note

The first time you push a commit on your branch, you will see a warning like:

```
fatal: The current branch my-tuxy-project has no upstream branch.
To push the current branch and set the remote as upstream, use
```

```
git push --set-upstream origin my-tuxy-project
```

```
To have this happen automatically for branches without a tracking
upstream, see 'push.autoSetupRemote' in 'git help config'.
```

Simply follow the instructions to fix the issue.

You only need to do this once.

8. Repeat these steps until you are finished with new content.



Tip

It is a good idea to break up long content creation sessions. Pause frequently to commit and push edits to your *origin* remote.

4.6 Submit a pull request

Before you submit your document, be sure to update your fork with changes in the 'main' branch of the SUSE TRD GitHub repository.

1. Open a local terminal and change to your clone directory.

For example:

```
cd ~/git/GITHUB_USER/technical-reference-documentation
```

2. Check out your 'main' branch.

```
git checkout main
```

3. Update your clone with changes in the *upstream* remote.

```
git pull upstream main
```

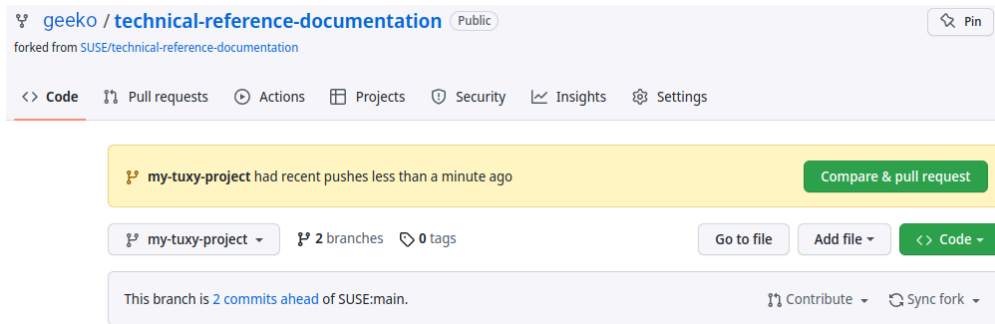
4. Synchronize these updates to your *origin* remote.

```
git push origin
```

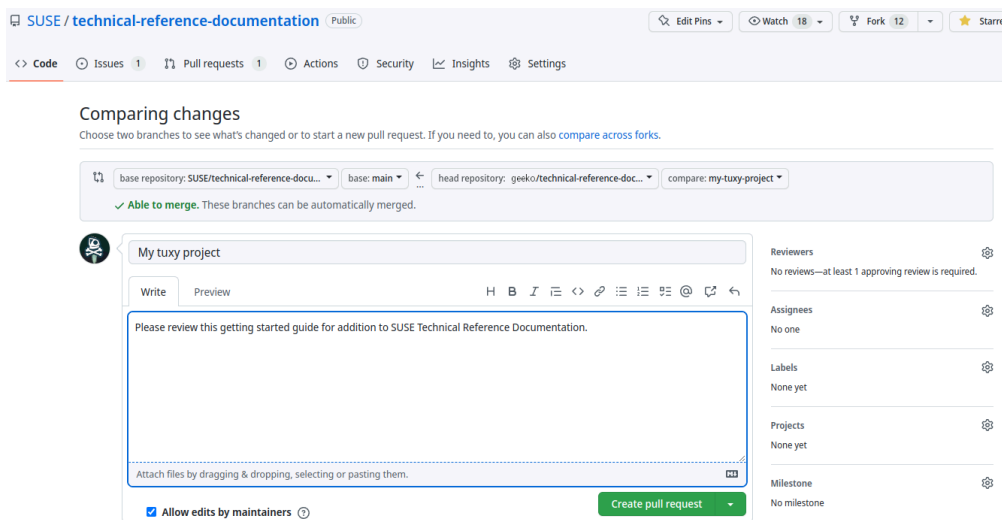
With your document in good shape and your fork synchronized, it is time to submit your document for official review. To do this, you submit a pull request (PR) from the GitHub UI.

1. Log into your GitHub account.
2. Select your fork of the *upstream* repository.

3. Select your branch and click *Compare & pull request*.

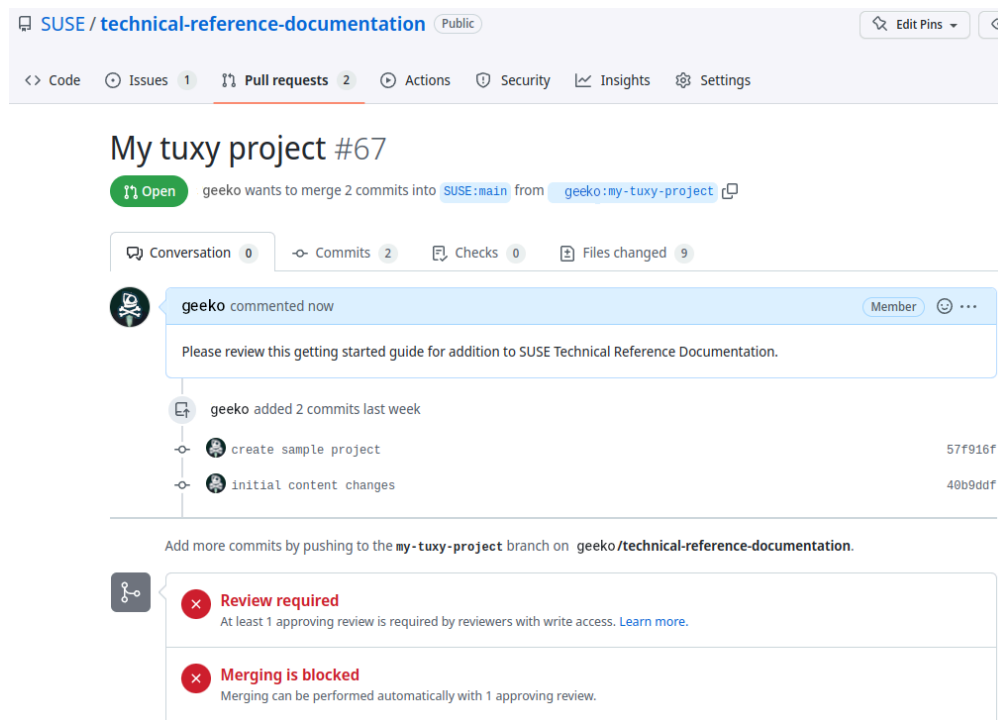


4. Verify that your branch is able to be merged and click *Create pull request*. You can add any helpful notes to the reviewer in the provided space.



5. Follow the status of your pull request in the GitHub *Pull requests* page of the upstream repository.

Respond to comments and suggestions in the *Conversation* tab.



6. When your document is ready, verify that is published to the SUSE TRD [website \(https://documentation.suse.com/?tab=trd\)](https://documentation.suse.com/?tab=trd).

7. After your document has been published, clean up your work area.

a. Delete your previous branch.

```
git checkout main
git branch -d my-tuxy-project
```

b. Merge upstream changes to your local fork.

```
git pull upstream main
```

c. Synchronize your fork.

```
git push origin
```


5 Repository structure

SUSE TRD is both a [website \(https://documentation.suse.com/?tab=trd\)](https://documentation.suse.com/?tab=trd) with links to published documents and a [GitHub repository \(https://github.com/SUSE/technical-reference-documentation\)](https://github.com/SUSE/technical-reference-documentation) of their source files. As a contributor, you need a general understanding of the structure of the source repository.

The repository's top-level directories are:

- *linux*: contains documentation focused primarily on aspects of Linux and related products, projects, and tools
- *kubernetes*: contains documentation focused primarily on Kubernetes and related products, projects, and tools
- *common*: contains text (such as the license and disclaimer), images (such as official SUSE logos), templates (for various document types), and scripts (for simplifying and automating tasks) that may be used across the two major categories.

```
├── common
│   ├── adoc
│   ├── bin
│   ├── images
│   └── templates
├── kubernetes
│   ├── enterprise
│   ├── reference
│   └── start
└── linux
    ├── enterprise
    ├── reference
    └── start
```

Inside the *linux* and *kubernetes* directories, are directories for the document types:

- *start*: getting started guides
- *reference*: reference implementations and reference configurations
- *enterprise*: enterprise architectures

5.1 Getting started guide structure

A getting started guide features a combination of components by one or more "entities," such as SUSE, a partner company, or an open source project. The source files used to generate these guides are maintained under the *linux/start* or *kubernetes/start* directories, depending on the primary implementation focus. Guide files are further organized into directories named for a featured entity. This is typically the provider of the component at the highest layer of the technology stack and closest to the user's workload or use case. Thus, for example, you find directories for Kubeflow, MongoDB, Ondat, and so on.

```
start
├─ bin
├─ kubeflow
├─ minio
├─ mongodb
├─ ondat
├─ suse
├─ sysdig
├─ upbound
└─ veeam
```






Note

The *bin* directory contains scripts to help contributors perform various tasks.

Inside an "entity" directory, you find further structure. For example:

```
veeam
├─ DC-gs_rancher_veeam-kasten ❶
├─ adoc ❷
│   └─ common_gfdl1.2_i.adoc -> ../../../../common/adoc/common_gfdl1.2_i.adoc
│   └─ ...
│   └─ gs_rancher_veeam-kasten.adoc ❸
│       └─ gs_rancher_veeam-kasten-docinfo.xml ❹
├─ images -> media
└─ media ❺
    └─ src
        └─ png
            └─ rancher_veeam-kasten_k10_dashboard-welcome.png
            └─ ...
            └─ rancher_veeam-kasten_architecture-1.png
        └─ svg
            └─ logo-lockup_rancher-by-suse-kasten-by-veeam_hor_dark.svg
```

```
└─ suse.svg -> ../../../../../../common/images/src/svg/suse.svg
```

- ❶ **Doc Config (DC) file:** There is one DC file for each documentation deliverable in which you specify parameters that define the name of the main content file, the rendering stylesheet, the revision date, and so on.
- ❷ ***adoc* directory:** Contains your AsciiDoc content files and your DocBook metadata file.
- ❸ **Main content file:** This plain text file, formatted with the [AsciiDoc \(https://asciidoc.org/\)](https://asciidoc.org/)  markup language, is where you write most of the text of your document. You can use the `include` directive to merge content from other AsciiDoc files during the rendering process.
- ❹ **DocBook metadata (*docinfo.xml*) file:** This file specifies metadata about your document, including title, author information, cover logo file, and executive summary. Much of this information is actually defined as document attributes (variables) in your main content file and referenced here to save you from having to type the same information twice.
- ❺ ***media* directory:** You can include logos, diagrams, screenshots, and other images in your guides. These media files are stored in subdirectories by file type under *media/src*. Many image file types are supported, but preference is given to:
 - [Scaled Vector Graphics \(https://www.w3.org/Graphics/SVG/\)](https://www.w3.org/Graphics/SVG/)  or 'SVG' is a widely-deployed, royalty-free, vector graphics format developed and maintained by the W3C. A vector graphics image is rendered from a set of instructions that defines the image from basic, geometric shapes. This allows SVGs to be resized without loss of clarity. SVGs are typically used for logos and architectural diagrams.
 - [Portable Network Graphics \(https://en.wikipedia.org/wiki/PNG\)](https://en.wikipedia.org/wiki/PNG)  or 'PNG' is a raster graphics format. A raster image encodes information (like position, color, or brightness) for each pixel (picture element) that makes up the image. PNG uses lossless compression, allowing PNGs to be edited and resaved without loss of clarity (unlike Joint Photographic Experts Group or 'JPEG' images). Scaling raster images can introduce artifacts. This is particularly apparent when images are enlarged. Choose image resolutions to minimize these artifacts.



Note

In the "entity" directory, *images* is a symbolic link to the *media* directory, and is used to maintain compatibility with existing processes and tooling.

You can learn more about Doc Config, AsciiDoc, and DocBook metadata files in [Section 6, “Templates and framework”](#).

5.2 Reference Implementation and Reference Configuration structure

Coming soon.

6 Templates and framework

Your SUSE TRD documentation project relies on different types of source files, placed in specific locations within the repository’s directory structure. The files and structures can be created from scratch. But you can use the provided automation tools to generate a framework for you, including templates for your source files.

6.1 Getting started guides

Getting started guides are relatively simple documents, but they do require multiple files located in a specific directory structure. You can create this structure and the requisite files manually, but the `gssetup.sh` script is available to help automate the process.

6.1.1 Generating the framework

1. Open a local terminal and change to the directory containing your clone.

```
cd ~/git/GITHUB_USER/technical-reference-documentation
```

2. If you have not done so already, create a branch for your project.

```
git branch my-tuxy-project
```

3. Check out your project branch.

```
git checkout my-tuxy-project
```

4. Change to either the `kubernetes/start/` or `linux/start/` subdirectory.

For example:

```
cd kubernetes/start
```



Tip

Linux is likely a part of every solution, but, if any portion of SUSE's Enterprise Container Management portfolio is involved, the documentation should be created in the kubernetes directory tree.

5. Execute the setup script to generate directory structures and basic files from templates.

```
./bin/gssetup.sh
```

6. Verify that you have performed the prerequisite actions presented in the banner, then press *ENTER* to continue.

```
= = = = =
= Set up workspace for new TRD getting started guide =
= = = = =

This script will prompt you for information about your
guide, then use your responses to create the directories
and template files for your guide

- - - - -
- Before proceeding make sure you have:
- 1. created a local branch: `git branch myproject`
- 2. checked out your new branch: `git checkout myproject`
- 3. changed to the 'kubernetes/start' or
-   'linux/start' directory (as appropriate)
- - - - -

Are you ready to proceed?
Press ENTER to continue or CTRL+C to cancel.
```



Note

The setup script makes the following checks:

- that you do not have the *main* branch checked out
- that your current working directory is either kubernetes/start or linux/start

7. Identify the primary SUSE product name.

```
- - - - -
- Gathering some information
- - - - -

- - - - -
- Identify the featured SUSE products by entering
- one product abbreviation at a time.
- When done, press ENTER with no value.
-
- TIP: Start at the top of the software stack.
-     For example: 'rancher' then 'rke2' then 'sles'
-
- Additional options:
- 'list': display accepted abbreviations.
- 'clear': clear the product list and start over.
- Press CTRL+C to cancel and exit the script.
- - - - -

>> SUSE product : rancher
```



Note

Accepted SUSE product abbreviations are listed in the table below.

Abbreviation	Product name
sles	SUSE Linux Enterprise Server
slessap	SUSE Linux Enterprise Server for SAP applications

Abbreviation	Product name
slehpc	SUSE Linux Enterprise High Performance Computing
slemicro	SUSE Linux Enterprise Micro
slmicro	SUSE Linux Micro
slelp	SUSE Linux Enterprise Live Patching
slert	SUSE Linux Enterprise Real Time
sleha	SUSE Linux Enterprise High Availability
slebci	SUSE Linux Enterprise Base Container Images
suma	SUSE Manager
smls	SUSE Multi-Linux Manager
rancher	SUSE Rancher Prime
neuvector	SUSE Security (previously NeuVector Prime by SUSE)
harvester	SUSE Virtualization (previously Harvester by SUSE)
rke	Rancher Kubernetes Engine
rke2	Rancher Kubernetes Engine 2
k3s	K3s by SUSE
longhorn	SUSE Storage (previously Longhorn by SUSE)

8. Identify a primary partner.

- - - - -
- Enter the name of the primary partner.
-
- TIP: Select one partner whose product is at the top
- of the software stack and provides the key
- functionality for the featured use case.

```
- - - - -  
  
>> Primary partner : tuxy
```



Note

In general, the product at the highest layer of the software stack is key for addressing the use case. Enter the name of the provider of this product as the primary partner.

The primary partner name is used to help organize the documentation in the repository's directory structure and also appears in the names of some source files as well as in the URL of the published guide. If it is necessary to include additional partner names, this can be done in a manual process after running this script.

9. Identify the primary partner's product.

```
- - - - -  
- Enter the name of the primary partner's product.  
-  
- TIP: If the primary partner and the product  
-     share the same name, you can leave the  
-     partner product blank to avoid repetition.  
-  
- - - - -  
  
>> Primary partner's product : penguin
```



Note

The primary partner product name appears in file names and the URL of the published guide. If you need to include multiple products by the primary partner, list each one separated by a hyphen (-).

10. Optionally enter a use case or description.

```
- - - - -  
- OPTIONAL  
-  
- If a solution can address multiple use cases,  
- it may be useful to create a separate guide to  
- address unique concerns of each use case.  
- Since the product stack is insufficient to distinguish  
- each guide, some additional text can be added to the
```



```

- file name.
-
- TIP: It is preferable to leave this blank.
-     If needed, use fewer than 20 characters for the
-     additional text.
- - - - -
>> Distinctive text :

```



Note

This descriptive text appears in file names and the URL of the published guide.

It can be useful to distinguish guides targeting different use cases with the same solution stack.

You can also use this distinctive text to list an additional partner and product, but be sure to separate these with a hyphen (-).

In most cases, you should leave this entry blank.

11. Review the proposed structure and naming.

```

- - - - -
- Preparing to create the following structure:
-
- /home/terry/git/technical-reference-documentation/kubernetes
- └─ start
-   └─ tuxy
-       ├── DC-gs_rancher_tuxy-penguin
-       ├── adoc
-       │   ├── gs_rancher_tuxy-penguin.adoc
-       │   ├── gs_rancher_tuxy-penguin-docinfo.xml
-       ├── images -> media
-       └─ media
-           └─ src
-               ├── png
-               └─ svg
-
- NOTE: Several symbolic links will also be created.
- - - - -

>> Press ENTER to create document structure or CTRL+C to cancel.

=====
= Workspace for your new guide has been set up.

```

```
=
= Access your workspace in:
= /home/terry/git/tls/technical-reference-documentation/kubernetes/start/tuxy
= = = = =
```



Note

No directories or files are created until you press *ENTER*. If you press *CTRL + C*, you cancel the planned operations and return to the command line.

12. Confirm that the structure has been created as intended.

```
cd tuxy
tree .
```

```
tree .
.
├── adoc
│   ├── common_docinfo_vars.adoc -> ../../../../common/adoc/common_docinfo_vars.adoc
│   ├── common_gfdl1.2_i.adoc -> ../../../../common/adoc/common_gfdl1.2_i.adoc
│   ├── common_sbp_legal_notice.adoc -> ../../../../common/adoc/
common_sbp_legal_notice.adoc
│   ├── common_trd_legal_notice.adoc -> ../../../../common/adoc/
common_trd_legal_notice.adoc
│   ├── gs_rancher_tuxy-penguin.adoc
│   └── gs_rancher_tuxy-penguin-docinfo.xml
├── DC-gs_rancher_tuxy-penguin
├── images -> media
├── media
│   └── src
│       ├── png
│       └── svg
│           └── suse.svg -> ../../../../common/images/src/svg/suse.svg

7 directories, 8 files
```

6.1.2 Understanding the templates

The `gssetup.sh` script creates the standard directory structure, symbolic links to common files, and the three principal files you will edit for your document. These three files include copious comments to help you understand how to use them. The following sections provide some highlights.

6.1.2.1 Doc Config (DC) file

The DC file (DC-gs_rancher_tuxy-penguin in the example) is located in the root of the generated partner directory. It specifies parameters that define how the document will be rendered. As a contributor, you only need to be concerned with two of these parameters:

DRAFT=yes

Specifies that the document is in draft mode and watermarks the document accordingly.



Important

When you are ready to submit your document for final review, you must comment out this parameter by preceding it with a hash mark:

```
#DRAFT=yes
```

MAIN="gs_rancher_tuxy-penguin.adoc"

Specifies the AsciiDoc file with the main contents of your guide.



Note

This parameter is set by the gssetup.sh script. You only need to change it if you rename the main document file.

6.1.2.2 DocBook metadata (docinfo) file

The docinfo file (gs_rancher_tuxy-penguin-docinfo.xml, in the example) is located in the adoc subdirectory. It defines metadata about your document, such as title and subtitle, brief descriptions, featured products and partners, authors, and even which logo to feature on the cover page. You only need to edit the docinfo file if you need to add metadata tags that are not already defined.



Important

The docinfo file references attributes (variables) defined in your AsciiDoc file. These appear in the docinfo file as words (or any sequence of characters, like "scomp1-version") enclosed in curly braces ({ and }).

All referenced attributes must be defined in your AsciiDoc file. Otherwise, your document will fail to render.

The docinfo template provides metadata tags for only one SUSE product, but you can easily add metadata for additional SUSE products.

```
<meta name="productname"> ❶  
  <productname version="{scomp1-version}">{scomp1}</productname> ❷  
</meta>  
<meta name="platform">{scomp1-full}</meta> ❸
```

- ❶ The `<meta name="productname"> </meta>` tag pair encloses the list of named SUSE products.
- ❷ List each product with a separate `<productname> </productname>` entry.
For example, to include a second SUSE product, you would add a new tag pair below the first:

```
<productname version="{scomp2-version}">{scomp2}</productname>
```

where:

- `scomp2` must be defined in your AsciiDoc file with the short name (or official abbreviation) for the SUSE product
 - `scomp2-version` must be defined in your AsciiDoc file with the relevant versions or versions of the SUSE product
- ❸ The primary SUSE product is identified with the `<meta name="platform"> </meta>` tag pair.
The `scomp1-full` attribute, defined in your AsciiDoc file, must be the full SUSE product name.

List technical partners (those companies or organizations) supplying components featured in the solution with:

```
<meta name="techpartner">  
  <phrase>{pcomp1-provider}</phrase>  
</meta>
```

Identify the document author or authors in the `<authorgroup> </authorgroup>`.

```
<authorgroup>
```

```

<author> ❶
  <personname> ❷
    <firstname>{author1-firstname}</firstname>
    <surname>{author1-surname}</surname>
  </personname>
  <affiliation> ❸
    <jobtitle>{author1-jobtitle}</jobtitle>
    <orgname>{author1-orgname}</orgname>
  </affiliation>
</author>
</authorgroup>

```

- ❶ Each author's information is enclosed in `<author>` `</author>` tag pairs. To add another author, replicate the contents in a new `<author>` `</author>` tag pair and update the attribute references.
- ❷ Include first (or given) name and surname (family name).
- ❸ Include company, organization, or project affiliation along with job, position, or role title.



Note

The authors identified in this section are displayed on the rendered document.

You can also acknowledge the contributions of editors and others in an "Acknowledgement" section of your AsciiDoc file.

The last section of the docinfo file you may want to edit identifies the logo that appears on the cover image.

```

<cover role="logos">
  <mediaobject>
    <imageobject role="fo">
      <imagedata fileref="suse.svg" width="5em"/>
    </imageobject>
    <imageobject role="html">
      <imagedata fileref="suse.svg" width="152px"/>
    </imageobject>
  </mediaobject>
</cover>

```

The default logo is the official SUSE company logo, which is always appropriate for all SUSE Technical Reference Documentation. If you wish, you can specify a logo "lock-up" or "mash-up" by updating the "fileref" option with the name of the appropriate image file.



Note

You must have documented approval from the partner before using the logo and the usage must follow SUSE and partner branding guidelines.

6.1.2.3 Main AsciiDoc (asciidoc) content file

The text of your document is contained in one or more [AsciiDoc \(https://asciidoc.org/\)](https://asciidoc.org/) (adoc) files, located in the `adoc` subdirectory. For a getting started guide, you typically put all your content in a single adoc file (`gs_rancher_tuxy-penguin.adoc` in the example). This is the file specified in the DC file with the `MAIN` parameter.



Tip

There are many reasons you might want to split your contents into multiple adoc files, such as to organize more complex documents or to make it easier for multiple contributors to work independently on different sections. If you choose to split your document, you will need to use the AsciiDoc [include directive \(https://docs.asciidoctor.org/asciidoc/latest/directives/include/\)](https://docs.asciidoctor.org/asciidoc/latest/directives/include/) to link all the files so the content is properly merged in a single rendered document.

Document attributes and variables

[Document attributes \(https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/\)](https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/) are name-value pairs you declare in your adoc file. Attributes enable you to configure the AsciiDoc processor, declare document metadata, and define reusable content that you can reference elsewhere within the document like variables in a programming language.

You define an attribute with the following pattern:

```
:name-of-attribute: value of attribute
```

- The attribute name is preceded and followed by a colon (:). Attribute names should begin with a letter and may include numbers, hyphens, and underscores.
- The value of the attribute can include any text up to a new line character.



Tip

There must be a space between the closing colon of the attribute name and the first character of the value text.

The adoc template provides a guide for defining your document attributes and variables.

Some attributes are required, such as:

```
// - - - - -
// ORGANIZATION
//   Do NOT modify this section.
// -
:trd: Technical Reference Documentation ❶
:type: Getting Started ❷
// - - - - -
```

❶ Declares this document as part of SUSE Technical Reference Documentation.

❷ Declares this document to be a getting started guide.

You also specify the document revision date in a variable:

```
// - - - - -
// DOCUMENT REVISION DATE
// -
:revision-date: YYYY-MM-DD ❶
:docdate: {revision-date} ❷
// - - - - -
```

❶ Be sure to enter the date you revised the document in the specified format.

❷ The `docdate` attribute, which is used by some processes, simply references the value of the `revision-date` attribute.

Your document title and subtitle are defined with:

```
// - - - - -
// DOCUMENT TITLE AND SUBTITLE
:title: (<75 characters) Your Guide Title
:subtitle: (<75 characters) Your Guide Subtitle
// - - - - -
```

See [Section 8.3, “Document title and subtitle”](#) for additional guidance.

You also define document attributes for the technical components featured in your guide. This allows you to define product names, versions, website URLs, and more, then reference these throughout your guide to reduce typing, ensure consistency, and minimize errors.

```
// - - - - -  
// TECHNICAL COMPONENTS  
  
:comp1-provider: SUSE ❶  
:comp1: component 1 short name ❷  
:comp1-full: component 1 long name ❸  
:comp1-version: component 1 relevant versions ❹  
:comp1-website: component 1 product website URL ❺  
:comp1-docs: component 1 product documentation URL ❻  
  
:comp2-provider: component 2 provider name ❼  
:comp2: component 2 short name  
:comp2-full: component 2 full name  
:comp2-version: component 2 relevant versions  
:comp2-website: component  product website URL  
:comp2-docs: component 2 product documentation URL  
// - - - - -
```

- ❶ Identify the company or organization providing the component.
- ❷ Provide an official short name for referring to the component. SUSE products have short names, such as SLES, SMLM, Rancher, and so on. Other providers may have official short names. Be sure to verify for proper branding.
- ❸ Provide the official full name of the component product. Some SUSE products include: SUSE Linux Enterprise Server, SUSE Manager, Rancher Prime by SUSE, NeuVector Prime by SUSE, and so on.
- ❹ Provide the relevant version or versions of the component. Guides are developed with a specific version, such as '15 SP5' for SLES or '2.7.9' for Rancher. However, the guide may be applicable to multiple versions of the component. This should be indicated whenever possible, either by listing all relevant versions ('15 SP4, 15 SP5', '2.7.8, 2.7.9') or by indicating a range ('15 SP4 +', '2.7.X').
- ❺ Provide a product (or project) website URL (for example, 'https://www.suse.com/products/server/'). By setting the URL in a document attribute, you can easily update it in one place, if needed, and reference it throughout your document.
- ❻ Provide the URL to the component's technical documentation.
- ❼ Replicate the attribute declarations for each of the major components.

Document descriptions express the purpose, value, and contents of your guide.

```
// - - - - -
// DOCUMENT DESCRIPTIONS

:usecase: (<55 characters) use case ❶

:description: (<150 characters) description ❷

:description-short: (<55 characters) social media description ❸

:executive-summary: (<300 characters) brief summary ❹

// - - - - -
```

- ❶ Provide a brief statement of the use case addressed by your guide, such as "database-as-a-service" or "Kubernetes multi-tenancy".
- ❷ Give a brief description of the guide. This can reference usecase or other attributes you have already defined.
- ❸ Shorten the description so that it might be used in a social media post.
- ❹ Provide an executive summary of your guide. The executive summary is the only one of these descriptions that is automatically printed in the published document.

Identify yourself and other authors. For each include first and last names, job title, affiliation (company, organization, project, etc.). For additional authors, you must also update the contents of the <authorgroup></authorgroup> section of the docinfo file for the authors to be listed.

```
// - - - - -
// CONTRIBUTORS

:author1-firstname: first (given) name
:author1-surname: surname
:author1-jobtitle: job title
:author1-orgname: organization affiliation
//:author2-firstname: first (given) name
//:author2-surname: surname
//:author2-jobtitle: job title
//:author2-orgname: organization affiliation

// - - - - -
```



Tip

To identify other contributors and editors, use the same format for the variable names, but replace 'author' with 'contrib' or 'editor'. It is common to mention contributors and editors in an "Acknowledgements" section of your guide.

Document attributes can be incredibly useful for maintaining consistency, reducing errors, and saving a little typing. A portion of this part of the guide is available for you to define any additional attributes you need.

```
// - - - - -
// MISCELLANEOUS
//   Define any additional variables here for use within the document.
// -

// - - - - -
```

6.1.2.4 Contents

After defining document attributes, you can begin developing your content. The adoc template generated by the `gssetup.sh` script provides an outline for a general getting started guide. You will find that the template is copiously documented with guidance and suggestions.

In general a getting started guide is structured as follows:

- **Title page**

In addition to title and subtitle, products and authors are also listed.

- **Disclaimer page**

The disclaimer is provided, along with the title, subtitle, revision date, and the executive summary.

- **Contents**

A table of contents is created from the top-level sections of your document.

- **Introduction**

The first actual contents of your guide is an introduction that briefly but clearly describes:

- what the solution is, including the participating providers and products
- the motivation, purpose, or use case is being addressed

The Introduction contains a few important subsections:

- **Scope**

This is a clear list or statement of what the guide covers and (sometimes) what it does not cover.

- **Audience**

Identify who would be most interested or helped by the information provided. This intended audience is typically by industry standard roles, such as systems administrator, cloud-native developer, infrastructure architect. You should list the knowledge and skills the reader should have to successfully follow the guide.

- **Acknowledgements**

Acknowledge others who contributed to the development of the guide.

- **Prerequisites**

List the resources the reader will need in order to follow the procedures detailed in the guide. You typically use an unordered list for these resources. For each, include product name, versions, link to product website, and any other information to help the reader set up the environment.

- **Procedure**

Your guide should detail the procedure the reader should follow to prepare, install, configure, and validate the components of the solution. Each of these phases can be rendered into its own section, such as:

- Setting up your environment
- Installing the components
- Configuring and tuning your installation
- Validating your deployment



Tip

Dividing a complex series of steps into subsections can make it easier for the reader to follow. For example, an 'Installing the solution' section might have subsections for installing each component.

- **Summary**

Briefly review the solution, motivation, and what was covered in the guide. You may also consider offering suggested next steps for the reader to continue the learning journey.

6.2 Reference implementations and reference configurations

Coming soon.


7 Style


SUSE TRD is part of the [SUSE Documentation \(https://documentation.suse.com/\)](https://documentation.suse.com/) ecosystem. Its objective is to provide well-structured, clear, and concise technical information in a way that facilitates real-world problem-solving. Consistent writing style and formatting contribute to the creation of high quality documentation.

Writing style goes beyond the essential elements of spelling, grammar, and punctuation to include tone of voice, choice of words, sentence and paragraph structure, and more.


Keep in mind the following general writing style guidance as you create your documentation for SUSE TRD:

- Make your content compelling, relevant, and valuable for your target audience.
- Be clear, concise, and complete.
- Use a conversational yet professional tone.
- Refer to the reader in the second person (*you*).
- Use the active voice and the present tense.
- Use inclusive language.
- Avoid contractions and abbreviations.

- Follow brand guidelines.
See <https://brand.suse.com> .
- Leverage Search Engine Optimization (SEO) practices to raise ranking by search engines and grow visibility.



For more on writing effective technical documentation, see the [SUSE Documentation Style Guide \(https://documentation.suse.com/style/current/\)](https://documentation.suse.com/style/current/) .

8 AsciiDoc

SUSE TRD documentation is rendered from plain text files. Formatting (including structure and text styling) is indicated with [AsciiDoc \(https://asciidoc.org/\)](https://asciidoc.org/) . This is a lightweight and semantic markup language with an intuitive syntax that is primarily designed for writing technical documentation.

8.1 General guidelines

When creating your content:

- Use a plain text editor.
- Ensure files are encoded to [UTF-8 \(https://en.wikipedia.org/wiki/UTF-8\)](https://en.wikipedia.org/wiki/UTF-8) .
- Start each line in the first column.
There should be no leading whitespace.
- Strip trailing whitespace from lines.
- End lines with the line feed character only.
Some operating systems add both a carriage return and a line feed to the end of a line. You must strip these carriage return characters from your files before submission. This can also be done by [configuring Git to handle line endings \(https://docs.github.com/en/get-started/getting-started-with-git/configuring-git-to-handle-line-endings?platform=windows\)](https://docs.github.com/en/get-started/getting-started-with-git/configuring-git-to-handle-line-endings?platform=windows) .
- Do not edit files outside the scope of your project.
This includes common files, scripts, and the files associated with other published documentation.
- Start each sentence on a new line, even if it is part of the same paragraph.

This makes it easier to edit and handle Git merge conflicts.

The following sections address specific aspects of document structure and text styling with AsciiDoc.

8.2 Document attributes and variables

[Document attributes \(https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/\)](https://docs.asciidoctor.org/asciidoc/latest/attributes/document-attributes/) are name-value pairs that you can use to configure the AsciiDoc processor, define document metadata, and create reusable content. Think of document attributes as document-scoped variables. Document attributes can be defined anywhere in your document and used from that point forward.

SUSE TRD uses document attributes to define some required content, such as document title, author information, and product names. You can define your own document attributes and reference them throughout your document for frequently repeated content.

The definition of a document attribute takes the form:

```
:name-of-an-attribute: value of the attribute
```

Once defined, you can reference your document attributes in your text by surrounding the attribute name with curly braces (`{ }` and `_`).

For example:

```
:comp1-full: Rancher Prime by SUSE
:comp1-website: https://www.suse.com/solutions/enterprise-container-management/#rancher-product
:comp2-full: Kubeflow
:kubeflow_website: https://www.kubeflow.org/
:title: Deploy {comp2-full} with {comp1-full}
:executive-summary: Learn how to deploy {comp2-website}[{comp2-full}] to a cluster managed by {comp1-website}[{comp1-full}].
```

8.3 Document title and subtitle

With AsciiDoc, the title of your document is usually placed on the first line of the main content file and is prefixed by a single equal sign (`=`). An optional subtitle can be included on the same line, separated from the title by a colon and a space (`:`).

```
= My Document Title: My Document Subtitle
```

For getting started guides, you define title and subtitle as document attributes:

```
:title: Rancher by SUSE and Priority Support on AWS Marketplace  
:subtitle: Deploy Rancher on Elastic Kubernetes Service (EKS)
```

and reference them in your adoc file as:

```
= {title}: {subtitle}
```

and in your docinfo file as:

```
<title>{title}</title>  
<subtitle>{subtitle}</subtitle>
```

Titles and subtitles use title-style capitalization. You can implement title-style by following a few rules:

- Capitalize the first and last word.
- Write articles (*a*, *an*, *the*) in lowercase, unless it is the first word.
- Write prepositions in lowercase unless they are used with a verb (*Logging In*) or in a noun (*The On Button*).
- Write certain conjunctions in lowercase: *and*, *but*, *for*, *nor*, and *or*.
- Write *as* and *to* in lowercase.
- Capitalize everything that is not mentioned above.

8.4 Sections

Sections help you group and organize related text. Each section is marked by a section title or heading. This is short text, prefixed by a section marker that indicates the section level. Section markers are two or more equal signs (`==`, `===`, and so on).

```
== Section title level 1  
  
=== Section title level 2
```

==== Section title level 3

Guidance for section titles:

- Choose a section title that clearly identifies what contents to expect.
- If the section focuses on actions, begin with a verbal noun (gerund) or an imperative verb.
For example:

```
== Preparing the cluster  
  
== Prepare the cluster
```

- Use sentence case.

That is, only the first word and proper nouns should be capitalized.

- Minimize section title length (typically, 1 to 4 words).
- Avoid using more than three section levels.
- Ensure all sections have content.

This means that a subsection title should never immediately follow section title without some intervening text.

Consider this example:

```
== Enabling Podman  
  
Podman, which is short for Pod Manager, is a daemonless, open source tool ...  
  
=== Install Podman  
  
Podman is not installed by default in SLES 15, but you can install it with these steps.  
...  
  
=== Define subordinate UIDs and GIDs  
  
By default, only the root user can run Podman containers. Running ...  
...  
  
== Creating container images  
  
You can build an application container image ...
```



```
=== Make a build directory
```

```
You need a place in your file system to contain ...
```

8.5 Lists

Lists can provide structure to your content.

Unordered list

is a series of items in no particular order and is sometimes called a bulleted list.

- Prefix an item by an asterisk (`*`) in the first column.
- Create subordinate or nested lists by adding more asterisks.
- Avoid creating more than three nest levels.

```
* Security and compliance

** Prompt response by SUSE engineers to security incidents

** Premium quality security updates

** Configuration, auditing, and automation features

* Adaptability

** Modular design

** Broad hardware architecture support
```

Ordered list

is a series of items for which order matters, such as a series of instructions.

- Prefix an item with a period or full stop (`.`) in the first column.
The rendering engine will replace the period with an appropriate number or letter.
- Create subordinate or nested lists by adding more periods.
- Avoid creating more than three levels.

```
. Verify target hardware support.
```

```
.. Confirm CPU capabilities.

.. Confirm RAM size.

.. Confirm space on target hard disk.

. Install system.

.. Prepare installation media.

.. Boot installation media.

.. Follow installation wizard.

. Reboot into new system.
```

Description list

is a collection of terms and their descriptions.

- Start a term-description item with the term followed by two colons (`::`).
- Place the description on the next line without any indentation.
- Include at least one blank line between each term-description pair.

```
Cluster::
A set of worker machines, called nodes, that run containerized applciations.
Every cluster has at least one worker node.

Pod::
The smallest deployable unit of computing that you can create and manage in
Kubernetes.

Control plane::
The container orchestration layer that exposes the API and interfaces to
define, deploy, and manage the lifecycle of containers.
```

8.6 File and directory names

Use the grave accent or back tick (```) to delimit file and directory names.

For example:

```
The `manifest.yaml` file is located in the `$HOME/my-tuxy-project` directory.
```

For directory names, a trailing slash (`/`) can be used if it is not clear that you are referring to a directory.

8.7 Admonitions

Use admonitions are used to help draw the reader's attention to content. AsciiDoc supports five admonition types represented by the following labels:

- NOTE: provide additional information
- TIP: suggest a helpful tip
- IMPORTANT: highlight an important point
- CAUTION: advise that care should be taken
- WARNING: inform of danger, harm, or consequences

The basic admonition style places the admonition label followed by a colon (`:`) at the beginning of a line of text. This is useful for short admonitions that do not contain a line break. For example:

```
TIP: By default, some `zypper` commands perform `refresh` automatically.
```

This is rendered as:



Tip

By default, some zypper commands perform refresh automatically.

The block admonition style provides more flexibility for the content, as illustrated with this example:

```
[IMPORTANT]
====
When working with snapshots to restore data, it is important to know
that there are two fundamentally different scenarios Snapper can handle:

Undoing changes::
When undoing changes as described in the following, two snapshots are
being compared and the changes between these two snapshots are made undone.
Using this method also allows to explicitly select the files that should be
restored.

Rollback::
```

```
When doing rollbacks as described in <<System rollback>>, the system is
reset to the state at which the snapshot was taken.
```

```
====
```

8.8 Source code, commands, and output

Technical guides often need to present source code, commands, and other console output. AsciiDoc provides listing blocks for these purposes, in which text is rendered with a fixed-width font and other features to present this special content.

The basic listing block presents text verbatim. That is, text is rendered just as it is entered in terms of line and character spacing. The form of a listing block is:

```
[listing]
----
This   text has      really weird      spacing   that is   preserved.

Line spacing is also preserved.
----
```

A source code block is a special version of a listing block that enables syntax highlighting, using color and text styles to distinguish code structures (such as keywords, variables, constants, comments, and so on).

To illustrate, review the following Python code snippet.

```
[source, python] ❶
---- ❷
# import libraries
import matplotlib.pyplot as plt
import numpy as np

# define data points
xpoints = np.array([1, 2, 5, 12])
ypoints = np.array([5, 3, 11, 6])

plt.plot(xpoints, ypoints)
plt.show()
----
```

- ❶ `[source, python]` identifies this as a source code block and the source language as Python. Some common source language identifiers are: `bash`, `c`, `html`, `python`, `sql`, `yaml`, and `xml`.
- ❷ `----` delimits the beginning and end of the block.



Note

As with ordinary listing blocks, line spacing is preserved.

Commands entered on the command line are like source code, so they are handled with source code blocks but with console as the source language identifier. For example:

```
[source, console]
----
sudo zypper install vim
----
```

Commands can also be included inline with other text by using grave accents (or back ticks) to enclose the command, as in:

```
Use `zypper refresh` to update your enabled repositories.
```

A helpful feature of AsciiDoc is that document attributes can be referenced inside your source code blocks. You do this with the subs option. Consider this example:

```
// attribute in your document
:myPath: /home/geeko/myproject/

// attribute referenced in source code block
[source, console, subs="attributes+"]
----

tree {myPath}

----
```

This renders as:

```
tree /home/geeko/myproject/
```



Tip

Attribute substitution can be tricky. The AsciiDoc rendering engine has no way of knowing if curly braces enclosing text in your code should be treated literally or substituted with the value of a document attribute.

In Bash, for instance, it is common to reference variables as `${myVariable}`. It may be important for `${myVariable}` to appear in your code as-is. However, if attribute substitutions are enabled, the rendering engine would try to find an attribute, named myVariable, and substitute its value. If there is no such attribute, the render would fail.

One way to solve this is to not use the `subs="attributes+"` option. But, if you need some substitutions in your code block and not others, you can use [escaping \(https://docs.asciidoctor.org/asciidoc/latest/subs/prevent/\)](https://docs.asciidoctor.org/asciidoc/latest/subs/prevent/) to let the rendering engine know your intentions. Simply place a backslash (\) just before each of the curly braces. This lets the rendering engine know to treat the curly braces as literal characters and not as indication of an attribute reference.

Here is an example:

```
:myPrompt: geeko@mangrove.lane:~/myproject:

[source, console, subs="attributes+"]
----

{myPrompt} echo "My stored value is ${myVariable\}."

----
```

With `myPrompt` defined in your document, this code block would render as:

```
geeko@mangrove.lane:~/myproject: echo "My stored value is ${myVariable}."
```

For command output, use a simple listing block.

```
[listing]
----
Type   | # |      | Cleanup | Description                | Userdata
-----+---+ ... +-----+-----+-----+
single | 0 |      |          | current                    |
single | 1 |      | number  | first root filesystem      |
single | 2 |      | number  | after installation         | important=yes
single | 3 |      | number  | rollback backup of #1     | important=yes
single | 4 |      |          |                             |
----
```

8.9 Graphical user interface elements

As with commands on the command line, you may need to reference named elements of an application's graphical user interface (GUI). Use a double underscore (`__`) to delimit each element. If providing a series of element selections, separate each element with the greater than symbol (`>`).

```
. Open _myfile_.  
  
.. Click __File__ > __Open__.  
  
.. Select _myfile_ from the list.  
  
.. Click __Open__.
```

8.10 Links

AsciiDoc hyperlinks, allowing you to provide direct links to external, addressable resources.

A link consists of two parts:

target

This is the external resource and is represented by an address, known as a [Uniform Resource Identifier \(https://en.wikipedia.org/wiki/Uniform_Resource_Identifier\)](https://en.wikipedia.org/wiki/Uniform_Resource_Identifier) (URI). A common type of URI is the familiar [Uniform Resource Locator \(https://en.wikipedia.org/wiki/URL\)](https://en.wikipedia.org/wiki/URL) (URL) or address of a Web page.

link text

This is the text you wish the reader to see (typically in place of the URI) and be able to click to open the remote resource in an appropriate application (such as a Web browser).

The AsciiDoc processor detects common URL protocols, including HTTP, HTTPS, FTP, IRC, and MAILTO. That is, if you include a URL (such as <https://documentation.suse.com/>) in your text, it will be rendered as a link with the URL itself as the link text. However, the preference is to use custom link text in place of the URL.

To use a custom link text, append it to the URL enclosed in square brackets ([and]). That is,

```
https://target-URL[custom link text]
```

Consider this example:

```
Visit https://documentation.suse.com/[SUSE Documentation] to continue your learning  
journey.
```

When rendered, it appears as:

Visit [SUSE Documentation \(https://documentation.suse.com/\)](https://documentation.suse.com/) to continue your learning journey.

8.11 Images

You can enhance your document with graphical content, such as diagrams, screenshots, logos, and more. In AsciiDoc, the most common way to insert an image is with a block image macro. This takes the form:

```
image::TARGET[ATTRIBUTES]
```

where:

- `image::` designates the block image macro.

- `TARGET` is typically is your target image file.

AsciiDoc supports many image formats, but [Scaled Vector Graphics \(https://www.w3.org/Graphics/SVG/\)](https://www.w3.org/Graphics/SVG/) (SVG) and [Portable Network Graphics \(https://en.wikipedia.org/wiki/PNG\)](https://en.wikipedia.org/wiki/PNG) (PNG) are preferred.

Place your image files in the appropriate subdirectory of `media/src` by type. For example, if you are using a SVG file for a diagram, it would be located in `media/src/svg`. If you do this, you do not need to specify the path to the file.

- `[ATTRIBUTES]` is a comma-delimited list of attributes, as key = value pairs.

The most common attributes you will use include:

- *alt text*: alternate text that briefly identifies the image. It is useful for text-to-speech readers and situations when the image cannot be displayed.
- *scaledwidth*: preferred width of the image for PDF renderings. This is typically specified as a percentage of the content width (area between margins).
- *align*: suggest horizontal alignment for the image (that is, left, center, and right).

Thus, to display the image, `media/src/svg/my-tuxy-architecture.svg` at 75 percent of the page width and centered horizontally, you would use:

```
image::my-tuxy-architecture.svg[Tuxy Architecture, scaledwidth="75%", align="center"]
```

8.12 Blocks and continuations

Block elements (or blocks) are discrete, line-oriented chunks of content that form the basic structure of an AsciiDoc document.

The most common type of block is a paragraph, which is a contiguous set of lines of text, bounded by one or more blank lines. Sections, lists, and tables are also types of blocks. And, obviously, block image macros, listing blocks, code blocks, and admonitions are as well.



Tip

Blocks should always be bounded by an empty line (or document boundary), except when "attached" using list continuation.

List continuation (<https://docs.asciidoctor.org/asciidoc/latest/lists/continuation/#list-continuation>) (explicitly denoted by a plus symbol (+) by itself on the line between two blocks) can be used to join two adjacent blocks together. The most common use for this is to allow a list item to contain one or more blocks, ensuring alignment of elements.

Consider the example AsciiDoc code below.

```
. Verify that `kernel-default` has been installed.
+
[source, console]
----
sudo zypper se kernel-default
----
+
[IMPORTANT]
====
After `kernel-default` has been installed, be sure to remove `kernel-default-base`.
====

. Reboot each node to enable the `kernel-default` kernel.

[TIP]
====
Learn more about the `zypper` command:

[source, console]
----
man zypper
----
=====
```

Rendered, this code looks like:

1. Verify that kernel-default has been installed.

```
sudo zypper se kernel-default
```



Important

After `kernel-default` has been installed, be sure to remove `kernel-default-base`.

2. Reboot each node to enable the `kernel-default` kernel.



Tip

Learn more about the `zypper` command:

```
man zypper
```


Notice that the code block and **IMPORTANT** admonition are correctly aligned under the list item, but the **TIP** admonition is aligned with the document margin. As a bonus, there is a source code block inside and properly aligned with **TIP** admonition.

By understanding blocks and continuation, you can create complex and beautifully rendered documents.



9 DAPS

Among the important tools for your workflow as a contributor to SUSE Technical Reference Documentation is the [DocBook Authoring and Publishing Suite \(https://opensuse.github.io/daps/\)](https://opensuse.github.io/daps/) (DAPS). DAPS helps you author and publish documentation written in [DocBook XML \(https://docbook.org/\)](https://docbook.org/) and [AsciiDoc \(https://asciidoc.org/\)](https://asciidoc.org/) to a variety of different formats, such as HTML, EPUB, and PDF. It also includes tooling for validation, link checking, spell checking, and editor macros.

DAPS is dual-licensed under GPL 2.0 and GPL 3.0, and it is available as binary packages for the openSUSE and SUSE Linux Enterprise, and it can be installed from source on any Linux distribution. DAPS has several software dependencies, some of which are required and others are recommended to enable certain functionality. For details, see the [System Requirements and Installation \(https://opensuse.github.io/daps//doc/cha.daps.user.inst.html\)](https://opensuse.github.io/daps//doc/cha.daps.user.inst.html) section of the DAPS User Guide.


Daps2Docker (<https://github.com/openSUSE/daps2docker>)  aims to simplify and expand access to the rendering capabilities of DAPS. It does this by packaging DAPS and some of its dependencies in an easy-to-use, OCI compliant container.

Below are details for installing and configuring Daps2Docker on a bare metal or virtual machine running one of the following operating systems:

- SUSE Linux Enterprise Server (<https://www.suse.com/products/server/>)  (SLES) 15 SP5
- openSUSE Leap (<https://www.opensuse.org/#Leap>)  (Leap) 15.5
- openSUSE Tumbleweed (<https://www.opensuse.org/#Tumbleweed>)  (Tumbleweed)






Note

It should require little effort to adapt these steps for other operating systems that can run OCI-compliant containers (<https://opencontainers.org/>) .


9.1 Prerequisites

The installation steps outlined here assume that you have the following prerequisites:

- A bare metal or virtual machine with access to the Internet and running one of the following operating systems:
 - SUSE Linux Enterprise Server (<https://www.suse.com/products/server/>)  (SLES) 15 SP5
 - openSUSE Leap (<https://www.opensuse.org/#Leap>)  (Leap) 15.5
 - openSUSE Tumbleweed (<https://www.opensuse.org/#Tumbleweed>)  (Tumbleweed)



Note

If you are using SUSE Linux Enterprise Server, some of the tools you will require are curated through [modules and extensions](https://documentation.suse.com/sles/15-SP5/html/SLES-all/article-modules.html) (<https://documentation.suse.com/sles/15-SP5/html/SLES-all/article-modules.html>) . You can use YaST or follow the steps below at the command line to activate the needed modules and extensions.

1. Enable the SLE Desktop Applications Module

```
sudo SUSEConnect -p sle-module-desktop-applications/15.5/x86_64
```

2. Enable the SLE Development Tools Module

```
sudo SUSEConnect -p sle-module-development-tools/15.5/x86_64
```

3. Enable the SLE Workstation Extension

```
sudo SUSEConnect -p sle-we/15.5/x86_64
```

4. Enable the SLE Containers Module

```
sudo SUSEConnect -p sle-module-containers/15.5/x86_64
```

- Podman (<https://documentation.suse.com/container/all/single-html/SLES-container/#chapodman-overview>)  installed and accessible to your user account.

Podman is an open source, daemonless, container engine and toolset for deploying, running, and managing OCI compliant containers.

```
sudo zypper install podman
```





Note

As Daps2Docker's name implies, you can use Docker instead.

9.2 Enabling the Documentation:Tools repository

Configure your system to use the Documentation:Tools software repository.

1. Open your Web browser to the [Documentation:Tools repository](https://download.opensuse.org/repositories/Documentation:/Tools/) (<https://download.opensuse.org/repositories/Documentation:/Tools/>) .
2. Select the directory link for your operating system (for example, 'SLE_15_SP5/').
3. Download the `Documentation:Tools.repo` file and name it `Documentation_Tools.repo`. This [RepoInfo](https://en.opensuse.org/openSUSE:Standards_RepoInfo) file (https://en.opensuse.org/openSUSE:Standards_RepoInfo)  specifies some characteristics of the repository, such as a unique identifier, URL, and GPG checking.
4. On the command line, change to the directory containing the `Documentation_Tools.repo` file you just downloaded.

For example:

```
cd ~/Downloads
```

5. Copy or move the `Documentation_Tools.repo` to `/etc/zypp/repos.d/`.

```
sudo cp Documentation_Tools.repo /etc/zypp/repos.d/
```

6. Refresh repository data.

```
sudo zypper refresh
```



Tip

If you see the prompt:

```
Do you want to reject the key, trust temporarily, or trust always? [r/t/a/?]
(r): a
```

Enter 'a' to always trust the repository signing key.

9.3 Installing Daps2Docker

You can install Daps2Docker with `zypper` on the command line.

1. Install Daps2Docker from the Documentation:Tools repository.

```
sudo zypper install --repo Documentation_Tools daps2docker
```



Important

When you install Daps2Docker, you may see output like the following:

```
Problem: nothing provides 'docker' needed by the to be installed
daps2docker-0.18-150400.1.1.noarch
Solution 1: do not install daps2docker-0.18-150400.1.1.noarch
Solution 2: break daps2docker-0.18-150400.1.1.noarch by ignoring some of its
dependencies

Choose from above solutions by number or cancel [1/2/c/d/?] (c): 2
```

Because Podman is being used, you can safely choose to "break" this dependency by selecting option '2'.

9.4 Configuring Daps2Docker

Configure how Daps2Docker works for you in a local configuration file.

1. Make a directory for your configuration file.

```
mkdir -p ~/.config/daps2docker
```

2. Copy and execute this command to create your configuration file.

```
cat >> ~/.config/daps2docker/config <<EOL
#
# This is the configuration file for the daps2docker package
#
# set the default rendering formats
formats="html,pdf"
# set the container engine
container_engine="podman"
# set the container name
containername="registry.opensuse.org/documentation/containers/containers/opensuse-
daps-toolchain:latest"
EOL
```

9.5 Using Daps2Docker

Daps2Docker is run from the command line with the `/usr/bin/daps2docker` script.

1. Enter your documentation directory, containing your DocBook Config (DC) file.

```
cd PATH/TO/YOUR/DC-FILE
```

2. Run Daps2Docker to render your document.

```
daps2docker DC-FILE [FORMAT]
```



Tip

If you do not specify `FORMAT`, the configured default formats will be assumed.
See `daps2docker --help` for a list of supported formats and other options.



Note

If you have not configured your system to run Podman in rootless mode, you will be prompted for your root or sudo password when you run this script.

```
podman needs to be run as root.  
[sudo] password for root: *****
```

3. Review the command output.

Daps2Docker prints to your console various status and error messages. If your document can be rendered, you will get output like the following to tell you where to find your documents:

```
Your output documents are:  
/tmp/daps2docker-MavBGLXK/<filename>/html/FILENAME_draft/  
/tmp/daps2docker-MavBGLXK/<filename>/FILENAME_draft_en.pdf
```



Important

The contents of the `/tmp` directory are ephemeral. If you wish to save your rendered documents, move them to another location.

9.6 Validating your document code

Your contribution to SUSE TRD is submitted as a collection of source files, such as DocBook metadata, AsciiDoc text, and images. These files are combined and rendered into an accessible format, like HTML and PDF, which can then be published. If source files are missing or if they contain code errors, your document may render incorrectly or not at all. And, thus, your contribution may be rejected for publication.

If your document cannot be rendered, Daps2Docker prints messages to the console to help you identify the issue. For example, if you did not define the document attribute, 'product1', but you reference it within your document, you would see:

```
asciidoctor: WARNING: skipping reference to missing attribute: product1
```

Even if your document renders, be sure to validate it for content, style, and formatting by reviewing the renderings. Peer reviews can help immensely to identify both content errors and better ways to express the information you wish to share.

10 Summary


SUSE Technical Reference Documentation is a repository of high quality documentation for solutions that address real-world use cases with featured SUSE, partner, and community components.

This document provides an overview of the tools, techniques, workflows, and styles that you will use to make successful contributions to SUSE TRD.

11 Legal notice

Copyright © 2006–2025 SUSE LLC and contributors. All rights reserved.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or (at your option) version 1.3; with the Invariant Section being this copyright notice and license. A copy of the license version 1.2 is included in the section entitled "GNU Free Documentation License".

SUSE, the SUSE logo and YaST are registered trademarks of SUSE LLC in the United States and other countries. For SUSE trademarks, see <https://www.suse.com/company/legal/> .

Linux is a registered trademark of Linus Torvalds. All other names or trademarks mentioned in this document may be trademarks or registered trademarks of their respective owners.

Documents published as part of the series SUSE Technical Reference Documentation have been contributed voluntarily by SUSE employees and third parties. They are meant to serve as examples of how particular actions can be performed. They have been compiled with utmost attention to detail. However, this does not guarantee complete accuracy. SUSE cannot verify that actions described in these documents do what is claimed or whether actions described have unintended consequences. SUSE LLC, its affiliates, the authors, and the translators may not be held liable for possible errors or the consequences thereof.

12 GNU Free Documentation License

Copyright © 2000, 2001, 2002 Free Software Foundation, Inc. 51 Franklin St, Fifth Floor, Boston, MA 02110-1301 USA. Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

0. PREAMBLE

The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondly, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition. The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

- A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
- B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
- C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
- D. Preserve all the copyright notices of the Document.

- E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
- F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
- G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
- H. Include an unaltered copy of this License.
- I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
- J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
- K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
- L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
- M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
- N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
- O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS

You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all

Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See <http://www.gnu.org/copyleft/>.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

Copyright (c) YEAR YOUR NAME.

Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2

```
or any later version published by the Free Software Foundation;  
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.  
A copy of the license is included in the section entitled "GNU  
Free Documentation License".
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “ with... Texts.” line with this:

```
with the Invariant Sections being LIST THEIR TITLES, with the  
Front-Cover Texts being LIST, and with the Back-Cover Texts being LIST.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.