SUSE Linux Enterprise Server 15 SP6, SUSE Linux Enterprise Base Container Images 15 SP6, JupyterLab 4.2.5

Accelerate AI/ML Development with JupyterLab and SUSE Linux Enterprise Base Container Images

Support Collaborative Development with Containers

SUSE Linux Enterprise Server 15 SP6
SUSE Linux Enterprise Base Container Images 15 SP6
JupyterLab 4.2.5

Terry Smith, Director, Global Alliance Solutions (SUSE)
Brian Fromme, Manager, Partner Alliances (SUSE)

Accelerate AI/ML Development with JupyterLab and SUSE Linux Enterprise Base Container Images

Support Collaborative Development with Containers

Date: 2024-10-23

Summary

Deploy JupyterLab and Python with Podman and SLE BCI for accelerated, collaborative development in data analytics, artificial intelligence, and machine learning.

Contents

1 Introduction

Modern software development is largely agile, open, and collaborative. Fostering a healthy, open source community can deliver tremendous benefits. This is illustrated in today's innovative domain of artificial intelligence and machine learning (AI/ML).

The AI/ML developer community is diverse. Contributors from a broad spectrum of disciplines leverage and contribute to a vast and growing ecosystem of open source libraries, frameworks, and other tools. This shared resource accelerates innovation, enabling developers to more quickly address real-world challenges and use cases.

The rapid pace of innovation in AI/ML can also be challenging. Complex applications often rely on functions and structures found in multiple, rapidly changing libraries. As the underlying libraries change, the dependent applications can break or produce unexpected results. This means that many AI/ML applications have specific version requirements for the underlying libraries. In multi-application and multi-user environments, this can introduce significant challenges with version conflicts that are difficult or impossible to manage through traditional methods.

Today, developers have access to container technologies. Containers provide a convenient, lightweight way to package applications and their dependencies while isolating them from other applications and their dependencies. This makes containerized applications highly portable and scalable, letting you run your applications everywhere they are needed - in the data center, in the cloud, and at the edge.

Building application containers starts with choosing a base container image. SUSE Linux Enterprise Base Container Images (https://www.suse.com/products/base-container-images) (SLE BCI) are tested, certified, and enterprise-ready. They offer a secure and compliant foundation for any development project, including optimized container images designed to furnish robust programming language environments.

JupyterLab (https://jupyter.org) by Project Jupyter is an interactive development environment that is popular with AI/ML, data science, and other researchers and developers. It is not built on a microservices architecture, so it may seem to be an unusual application to put into a container. However, containerizing JupyterLab with all required libraries can be both a useful illustration of the power of containerization and a convenient way to share a pre-configured development environment with other researchers.

1.1 Scope

This guide illustrates how to containerize and deploy JupyterLab, the Python development platform for data analytics and AI/ML, using SUSE Linux Enterprise Base Container Images and Podman.

1.2 Audience

AI/ML developers, researchers, operations teams, systems architects, and others interested in building and deploying application containers can explore a step-by-step example in this guide. To get the most from this document, the reader should have basic Linux command line skills and familiarity with container concepts.

1.3 Acknowledgements

The authors wish to thank the following individuals for their contributions to this guide:

- Dan Čermák, Software Engineer, Technology and Product, SUSE
- Victor Gregorio, Senior Manager, Solution Architects, SUSE
- Darragh O'Reilly, Senior Cloud Engineer, SUSE

2 Overview

In this guide, you learn how to:

- set up Podman (https://podman.io) on SUSE Linux Enterprise Server (https://www.suse.com/products/server) to run containers as a non-privileged user.
- define and build a container image for a JupyterLab (https://jupyter.org) ML development environment.
- run your JupyterLab container with access to files on your host system.
- share your JupyterLab container with others.

2.1 Software ecosystem

1. SUSE Linux Enterprise Server (https://www.suse.com/products/server)

   This guide leverages SUSE Linux Enterprise Server 15 SP6 as a host operating system for building and running containers. SUSE Linux Enterprise Server can be run on bare metal or in virtual machines to provide a stable, supportable Linux environment.

   ### Note

   The steps detailed in this guide were developed and tested on SUSE Linux Enterprise Server 15 SP6. Little to no modification should be required to follow this guide with newer versions of SUSE Linux Enterprise Server or corresponding versions of openSUSE Leap (https://www.opensuse.org/#Leap).

2. SUSE Linux Enterprise Base Container Images (https://www.suse.com/products/base-container-images) (SLE BCI)

   A base container image is a foundational layer for your application container. It typically provides a stripped-down Linux operating system user space without the kernel. SUSE Linux Enterprise Base Container Images are freely available, redistributable, and designed to be a secure, flexible foundation onto which you can add your application components. SLE BCI development container images provide optimized programming language environments, such as Python, to provide needed components from a trusted source and improve application container build times.

   ### Tip

   Ready-to-use container images are available in the SUSE Container Registry (https://registry.suse.com).

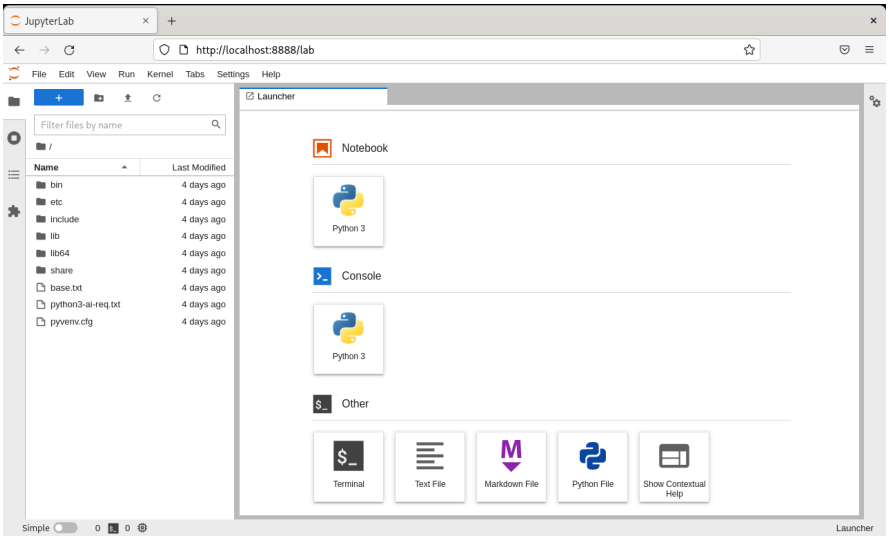
3. Podman (https://podman.io)

   Podman is a daemonless, open source tool for finding, building, managing, and running Open Container Initiative (https://opencontainers.org/) (OCI) containers on Linux systems. This guide uses Podman 4.9.5.

4. Python (https://www.python.org)

Python has become a popular programming ecosystem for ML developers for its simplicity, elegance, extensibility, cross-platform availability, and active, open source community of developers and users. With Python, ML developers have access to a vast collection of libraries, frameworks, and toolkits with which to build solutions to complex problems. This guide references Python 3.12.6 but should work with minimal issues for other 3.x releases.

5. JupyterLab (https://jupyter.org) from Project Jupyter
   JupyterLab is a modular, interactive development environment with a Web interface that enables data scientists and developers to create and share computational documents, called notebooks.



JupyterLab 4.2.5 is available at the time of publication.

Tip

See Replicable Python Environments with JupyterLab and SUSE Linux Enterprise Server (https://documentation.suse.com/trd/jupyter/html/gs_sles_jupyter-jupyterlab/index.html) for more details about working with Python and JupyterLab on SUSE Linux Enterprise Server.

3 Enabling Podman

Podman, which is short for Pod Manager, is a daemonless, open source tool for finding, building, managing, and running Open Container Initiative (https://opencontainers.org/) (OCI) containers on Linux systems. Podman is similar to other container engines, such as **Docker**, **CRI-O**, and **containerd**. It relies on an OCI-compliant container runtime (like *runc*, *crun*, *runv*, etc.) to interface with the operating system for creating the running containers. This means that running Podman containers is nearly indistinguishable from running those created by other common container engines. You can also run Podman containers as an ordinary, non-privileged user, making it easier and more secure.

Tip

See the SUSE Container Guide (https://documentation.suse.com/container/all/single-html/Container-guide) for a deeper discussion on containers and Podman with SUSE Linux Enterprise Server.

3.1 Install Podman

Podman is not installed by default in SUSE Linux Enterprise Server, but you can install it with these steps.

1. Enable SUSE Containers Module (a free add-on).

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

   Replace *RELEASE_VERSION* with the SUSE Linux Enterprise Server release you are using. For example: If you are using SUSE Linux Enterprise Server 15 SP6, then you would use:

   ```
   sudo SUSEConnect -p sle-module-containers/15.6/x86_64
   ```

2. Install Podman.

   ```
   sudo zypper in podman
   ```

Tip

It is a good idea to make sure your system is up-to-date with security patches and updates prior to installing new software. You can do this with the command:

```
sudo zypper up
```

3.2 Define subordinate UIDs and GIDs

By default, only the root user can run Podman containers. Running rootless Podman can improve security and enable multiple, unprivileged users to run containers on the same system. To enable rootless Podman, you must configure subordinate UIDs and GIDs.

Subordinate UIDs and GIDs are assigned in ranges to each user, and are used to map users running inside a container to users on your host system. These subordinate UID and GID ranges are specified for each user in `/etc/subuid` and `/etc/subgid`, respectively.

1. Verify that a range is not already defined for your user.
   Check the `/etc/subuid` and `/etc/subgid` files with:

   ```
   grep "${USER}" /etc/subuid /etc/subgid
   ```

   If output like the following is returned, ranges are already specified and you can proceed to the next section.

   ```
   /etc/subuid:tux:100000:65536
   /etc/subgid:tux:100000:65536
   ```

2. You can add subUID and subGID ranges with `usermod`.

   ```
   sudo usermod --add-subuids 100000-165535 --add-subgids 100000-165535 ${USER}
   ```

   ### Warning

   The subUID and subGID ranges ('100000-165535' in the command example) must be unique and non-overlapping for each user on the host.

3. Activate the subUID and subGID ranges by restarting the host system.

Note

If you modify either `/etc/subuid` or `/etc/subgid`, you must stop all running containers owned by the user and terminate the pause process running for that user.

You can do this automatically with the command:

```
podman system migrate
```

3.3 Configure the container storage driver

It is important to consider where and how your containers and container images are stored. By default, Podman stores container data in `/var/lib/containers/storage`. This is defined by the 'graphroot' option in `/etc/containers/storage.conf` and only applies to the root user.

For unprivileged users, Podman defaults to `$HOME/.local/share/containers/storage`. You can change this default location, say to another volume dedicated to container data. This is done system-wide by specifying the location with the 'rootless_storage_path' setting in the '[storage]' section of `/etc/containers/storage.conf`. This can be overridden for a particular user by setting 'rootless_storage_path' in `$HOME/.config/containers/storage.conf`.

Another important storage container setting determines how data are written. This is determined by the storage driver and is defined with the 'driver' setting in the '[storage]' section of `/etc/containers/storage.conf`. For the root user, valid drivers include 'overlay', 'vfs', 'zfs', and 'btrfs'.

Tip

The 'overlay' driver is a safe default choice.

If your home directory is on a different, supported file system, create or edit the file, `$HOME/.config/containers/storage.conf`, and add a '[storage]' section, such as:

```
[storage]
driver = "btrfs"
```

3.4 Grant access to your SUSE Customer Center credentials

It is only necessary to grant your user access to your SUSE Customer Center (SCC) credentials if you intend to install RPM software packages from the official SUSE Linux Enterprise Server repositories.

In this guide, you use the latest SUSE Linux Enterprise Base Container Images Python container image as your base. This container image already includes the official development and runtime components of a recent Python release, saving you from having to install them. All other software and libraries required for this guide can be installed from the Python Package Index (https://pypi.org/) with the `pip` command. Thus, you do not need to grant your user this access to follow this guide.

If you choose to use a different base image or want to use other software packages, your user may require access to your SCC credentials. You can do this by simply granting read permission to the files in `/etc/zypp/credentials.d`:

```
sudo setfacl -m u:${USER}:r /etc/zypp/credentials.d/*
```

Note

If you do not grant access to your SCC credentials, you may see a warning of an "error mounting subscriptions" when building your container. This is because `/etc/SUSEConnect` and `/etc/zypp/credentials.d/SCCcredentials` are specified as default mounts in `/etc/containers/mounts.conf`.

4 Creating container images

You build an application container image by adding the application with the libraries and the other software it needs to run (its dependencies) as layers on top of a base container image (https://documentation.suse.com/container/all/single-html/Container-guide/#cha-bci). In this section, you create a container image for the JupyterLab Python development environment along with some useful Python packages for machine learning and data analysis.

4.1 Make a build directory

You need a place in your file system to contain the files you will use for building your container image. Your build directory can be located anywhere you have write access, such as a directory under your home directory.

Create your build directory and change to it.

For example:

```
mkdir $HOME/jupcontainer
cd $HOME/jupcontainer
```

4.2 Define the Python environment

Replicability is critical in science and engineering. Complex applications, like an ML model, can produce wildly different results or not run at all because of minor code variations in a library dependency. Thus, it can be critical when building such applications to explicitly define all the software dependencies and their versions.

Fortunately, it is quite easy to create replicable Python environments (https://documentation.suse.com/trd/jupyter/html/gs_sles_jupyter-jupyterlab/index.html). Keys to this process are the `pip` Python package management tool, the Python Package Index (https://pypi.org/), and a list of required Python packages, usually in a file named `requirements.txt`.

Your `requirements.txt` file is a simple list of the names and versions of required packages. For example:

```
matplotlib==3.9.2
numpy==2.1.1
pandas==2.2.3
```

Each package you specify may have several dependencies of its own, and you can explicitly list every package and its version in your `requirements.txt` file. Often, though, you only need to list the major packages you want to install, and the `pip` resolver will attempt to identify and install compatible versions of the dependencies.

If you do not specify the version of a package you want, the `pip` dependency resolver will attempt to install the latest version that is compatible with the other packages. Because of code changes in different versions, this can result in unexpected behaviors in your code.

For this guide, you can let the `pip` resolver pick the versions for you. Create this `requirements.txt` file in your build directory:

```
jupyterlab
matplotlib
numpy
pandas
scikit-learn
scipy
seaborn
```

Tip

When you have verified a functioning configuration, you can automatically generate an updated `requirements.txt` file.

Within the environment, issue `pip freeze > requirements.txt`. This will include all installed packages, except `pip` itself and its dependencies.

4.3 Define the container image

Define your application container image by providing the instructions to add the components of your application onto the base image. These instructions are provided in a Containerfile (https://github.com/containers/common/blob/main/docs/Containerfile.5.md). If you are familiar with the Docker container ecosystem, a Containerfile shares the same format and most of the same options as a Dockerfile. After defining your application container in a Containerfile, you use another tool to execute the instructions and build the image.

An important consideration for your container build is the base image. You can obtain container images from a variety of sources, including the SUSE container registry (https://registry.suse.com).

SUSE Linux Enterprise Base Container Images are tested, certified, and enterprise-ready with up-to-date security patches. For this guide, you use the 'latest' Python container image, which, at the time of this writing, is built on SUSE Linux Enterprise Server 15 SP6 and features Python 3.12.6.

When choosing a base container image, keep in mind the following considerations:

Security

Is the image from and signed by a trusted source, verifiable, and up to date with vulnerability fixes?

Reliability

Is the image based on an operating system with a reputation for reliability and has the image undergone quality assurance testing?

Integration

Are you able to use the image with your preferred software, tooling, and workflows?

Redistribution

Are you legally able to redistribute the custom application container images you build on it?

Learn why SUSE Linux Enterprise base container images (https://documentation.suse.com/container/all/single-html/Container-guide/#cha-bci) offer the ideal foundation for your application containers.

Now create your Containerfile to define the container image.

1. Change to your build directory.

```
cd $HOME/jupcontainer
```

2. Create your file, named `Containerfile`, with the following contents:

```
# jupyterlab Containerfile

# https://registry.suse.com/
FROM registry.suse.com/bci/python:latest

# create unprivileged user
RUN useradd --uid 1000 --user-group --create-home jupyter
USER jupyter

# set notebooks directory
ENV NOTEBOOKS="/home/jupyter/notebooks"
RUN mkdir -p $NOTEBOOKS

# update the user's path to find the jupyter-lab executable
ENV PATH="/home/jupyter/.local/bin:/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/
bin:/sbin:/bin"

# set working directory for subsequent commands
WORKDIR /home/jupyter

# copy requirements file into working directory and set ownership
COPY --chown=1000:1000 requirements.txt .
```

```
# install JupyterLab
RUN set -euxo pipefail; \
    pip install --upgrade pip; \
    pip install -r requirements.txt; \
    pip cache purge;

# expose JupyterLab web port
EXPOSE 8888/tcp

# change to notebooks directory and launch JupyterLab
CMD cd $NOTEBOOKS; jupyter-lab --ip=* --port=8888 --no-browser --notebook-dir=
$NOTEBOOKS
```

Instructions used in this `Containerfile` to automate the build of your container include:

FROM
: Specify the base image to use.

RUN
: Run a command.

ENV
: Set an environment variable, whose value is passed to subsequent commands, like `RUN` and `CMD`.

WORKDIR
: Set the working directory for subsequent `RUN`, `CMD`, `COPY` and other instructions.

COPY
: Copy files into the container.

EXPOSE
: Inform the container engine that the container listens on the specified network ports at runtime.

CMD
: Provide a default command to be executed when the container is run.

4.4 Build the container image

At this point, you have:

- `Containerfile` to specify how to build the container
- `requirements.txt` file to specify the Python packages you want

You can now build your application container image. To do this you use the `podman build` command.

Note

Another popular tool that simplifies building OCI-compliant container images is Buildah (https://buildah.io/).

1. Make sure you are in your build directory.

```
cd $HOME/jupcontainer
```

2. Build the container image.

```
podman build --tag jupyter .
```

Tip

By using the `--tag jupyter` or `-t jupyter` option, your container image will be named `localhost/jupyter`.

3. Verify your container image exists.

```
podman image list
```

The output of this command looks something like:

```
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/jupyter               latest      d8bbefc6be69  2 minutes ago   837 MB
registry.suse.com/bci/python    latest      35ebb6a7981e  11 minutes ago  125 MB
```

You should see your container image with the name, 'jupyter', in your 'localhost' repository. Your container image is also assigned a unique identifier. The first several digits of this unique identifier is shown in the 'IMAGE ID' column.

Tip

You can also list images with the command `podman images`. Learn more about Podman commands at https://docs.podman.io/en/latest/Commands.html.

5 Running containers

5.1 Create a launch script

Now that you have a container image, you can use the `podman run` command to instantiate it as a container. You must also pass a few options so Podman can enable:

- access to the JupyterLab Web interface (listening on TCP network port 8888) from your local system.
- read and write files on your local file system.

Entering a long command each time you want to launch your container can be tedious and error-prone. In this section, you simplify this with a shell script.

1. Change to your build directory to create your launch script.

```
cd $HOME/jupcontainer
```

Tip

After you get things working, you may find it convenient to move your launch script into a directory in your PATH, such as `$HOME/bin`.

2. Create the text file `juplaunch` with this content.

```
#!/bin/bash

# define some variables
container_name=jupyter
container_uid=1000
container_gid=1000
container_port=8888
```

```
host_port=8888
container_volume=/home/jupyter/notebooks
host_volume=$(pwd)

# launch container with options
podman run \
  --userns=keep-id:uid=$container_uid,gid=$container_gid \
  --name $container_name \
  --publish $host_port:$container_port \
  --volume $host_volume:$container_volume:Z \
  localhost/$container_name
```

3. Grant yourself execute permissions for the script.

```
chmod u+x juplaunch
```

In the `juplaunch` script you pass the following options to `podman run`:

--name

Assign a name to the container.

--publish

Publish the container's exposed network port or range of ports to the host, making it accessible.

--volume

Create a binding to mount the specified volume from the host into to container.

Note

The `:z` or `:Z` option is required when using a host system with SELinux to ensure proper labels are placed on volume content.

Use `:z` to tell Podman to relabel file objects so two or more containers can share access to the volume and its contents.

Use `:Z` to tell Podman to relabel file objects so that only the current container can use the volume.

--userns

Set the user namespace mode for the container. With the release of Podman 4.3.0, you can use the `--userns=keep-id` option to map the UID and GID of the user inside the container to your UID and GID outside the container. This allows you to avoid ownership and per-

mission issues when accessing files in mounted volumes. Further discussion of user namespace mapping can be found in the Podman run documentation (https://docs.podman.io/en/latest/markdown/podman-run.1.html).

Note

Prior to Podman 4.3.0, user namespace mapping was approached using the `--uidmap` and `--gidmap` options.

5.2 Launch the container

1. Prepare your workspace.
   Your workspace is where you will save your Jupyter notebooks. This is a directory on your local system that you will access with JupyterLab from within the application container. For example,

   a. Create and enter a directory to hold your notebooks.

   ```
   mkdir $HOME/mynotebooks
   cd $HOME/mynotebooks
   ```

   b. Add one or more files (such as Jupyter notebooks) to your working directory.

   ### Tip

   If you do not have your own Jupyter notebook to use, you can find a plethora to download from the Web, including Alexis Cook's Titanic Tutorial (https://www.kaggle.com/code/alexisbcook/titanic-tutorial/notebook) for Kaggle (https://kaggle.com/).

2. Call your launch script.

   ```
   $HOME/jupcontainer/juplaunch
   ```

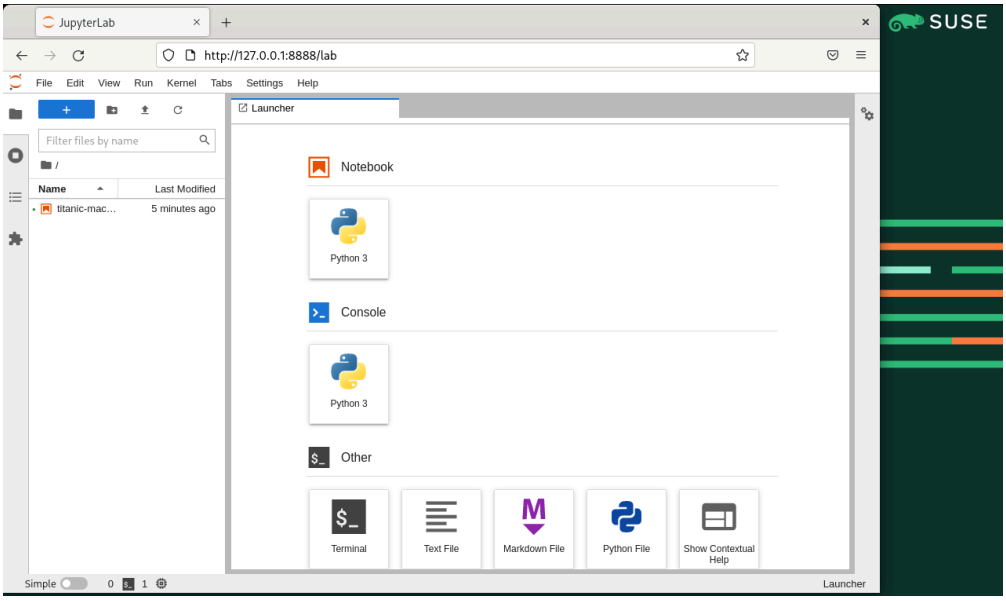
3. After launching your container, note the output in your terminal. There you can find the URL with embedded access token for the containerized JupyterLab environment.

   ```
       To access the server, open this file in a browser:
           file:///tux/.local/share/jupyter/runtime/jpserver-1-open.html
   ```

```
    Or copy and paste one of these URLs:
        http://95038cde3755:8888/lab?
token=77810fba9aec755d0a289a38cf2152bc3e1bf9b927082f30
     or http://127.0.0.1:8888/lab?
token=77810fba9aec755d0a289a38cf2152bc3e1bf9b927082f30
```

4. Access JupyterLab.

   Open the Web browser on your host system to the last URL shown in your terminal (the one that begins with http://127.0.0.1). If all goes well, you are presented with the main JupyterLab user interface (UI).



5. List your running containers.

   Open another terminal and enter the command:

```
podman container list
```

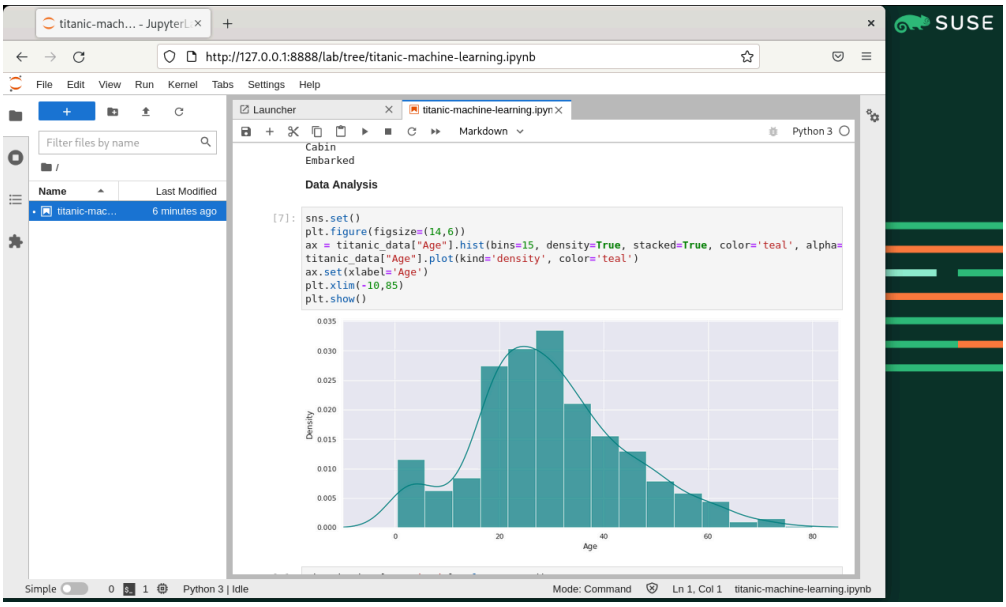
   This produces output like:

```
CONTAINER ID  IMAGE                       COMMAND               CREATED         STATUS
             PORTS                     NAMES
95038cde3755  localhost/jupyter:latest  /bin/sh -c cd $NO...  10 seconds ago  Up 10
 seconds ago  0.0.0.0:8888->8888/tcp  jupyter
```

Tip

The first column, 'CONTAINER ID', is the truncated, unique identifier of your container. You can use this container ID with various Podman commands to manage your container.

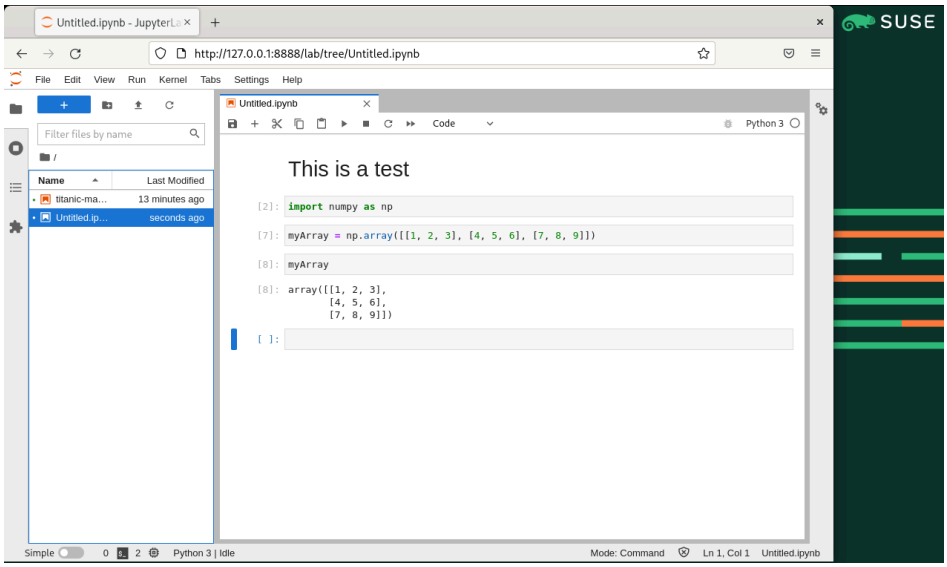
6. Verify you can access an existing file.
   Your shared files are listed in the left pane of the UI. Double-click a file name to open the file in a tab in the main pane.



7. Verify you can create a new notebook.

   a. In the JupyterLab UI *Launcher* tab, click *Python 3* under *Notebook*.

   b. Add some content to the new notebook.
      See the Jupyter notebooks user documentation (https://jupyter-notebook.readthedocs.io/en/latest/notebook.html) to learn about cell types, computational kernels, basic usage, and more.



Tip

Notebooks default to `untitled.ipynb`. You can rename your notebook with *File* > *Rename Notebook* or by right-clicking the file in the file listing and selecting *Rename*.

c. Verify file ownership.

On your host computer, open a terminal, change to your working directory, and list the contents.

```
ls -l
```

This produces output similar to the following:

```
total 328
-rw-r--r-- 1 tux  users   1586 Nov  9 15:44 test1.ipynb
-rw-r--r-- 1 tux  users 328257 Nov  9 15:32 titanic-machine-learning.ipynb
```

where you should see your user and group associated with the new Jupyter notebook.

8. Shut down JupyterLab.

a. Save and close any open Jupyter notebooks.

b. Select *File* > *Shutdown* in the JupyterLab UI.

Then, confirm that you want to shut down JupyterLab.

In the terminal you used to launch your container, you will see messages from JupyterLab that it is shutting down before you are returned to the command prompt of your local system.

5.3 Restart the container

In the previous section, you shut down the JupyterLab server, which caused the container to also shut down. This did not delete the container. It still exists, and you can start and stop it again and again.

Tip

You can override this behavior and automatically remove containers when they are shut down by using the `--rm` option with the `podman run` command.

1. Identify your stopped container.
   The default behavior of `podman container list` is to list only running containers, you need to use the `--all` option.

   ```
   podman container list --all
   ```

   ```
   CONTAINER ID  IMAGE                       COMMAND               CREATED         STATUS
                       PORTS                   NAMES
   95038cde3755  localhost/jupyter:latest  /bin/sh -c cd $NO...  19 minutes ago  Exited
    (0) 8 minutes ago  0.0.0.0:8888->8888/tcp  jupyter
   ```

   Notice that the STATUS of your container is 'Exited'.

2. Start your container.

   ```
   podman start CONTAINER-ID
   ```

   Replace `CONTAINER-ID` with the ID of your container.

   ### Tip

   You only need to type the first few unique digits of the container ID. Alternatively, you can refer to the container by its name ('jupyter').

3. Verify that your container is running.

```
podman ps
```

```
CONTAINER ID  IMAGE                       COMMAND               CREATED         STATUS
                   PORTS                   NAMES
95038cde3755  localhost/jupyter:latest  /bin/sh -c cd $NO...  22 minutes ago  Up 2
 minutes  0.0.0.0:8888->8888/tcp  jupyter
```

Tip

`podman ps`, `podman container ps`, `podman container list`, and `podman container ls` are synonyms for the same command.

4. Obtain the URL for JupyterLab running in your container.

   a. Enter a Bash shell inside your container.

   ```
   podman exec -it CONTAINER-ID /bin/bash
   ```

   b. Display any running JupyterLab servers.

   ```
   jupyter-server list
   ```

   Resulting in output like the following:

   ```
   [JupyterServerListApp] Currently running servers:
   [JupyterServerListApp] http://localhost:8888/?
   token=889b9af1f0c76fb9cda24c3327b3d723a928a8298af1dd22 :: /home/jupyter/
   notebooks
   ```

   ### Note

   In starting the container, you only launched a single instance of JupyterLab. However, it is possible to have multiple, simultaneous JupyterLab instances running on a system. Each would be listed here with its access URL and notebook directory.

   c. When you have copied the URL, you can return to your local command line.

   ```
   exit
   ```

5. Access JupyterLab by opening your Web browser to the URL you obtained.

6. Log out of the JupyterLab session by selecting *File* > *Log Out*.

Tip

By using the *Log Out* option, the JupyterLab server and your container continue to run. Check this with the `podman ps` command.

7. Access JupyterLab by opening your Web browser to the same URL you used in the previous step.

8. This time, shut down JupyterLab by selecting *File* > *Shutdown*.
   Verify that your container has shut down.

5.4 Clean up

You may find that you no longer need some of your containers or container images. Podman provides commands to help you clean up your container environment.

In this section, you delete a stopped container and then delete the image from which it was instantiated.

Tip

You can unceremoniously stop a running container with `podman stop CONTAINER-ID`.

1. Delete your container.

   a. Identify the container you want to delete.

   ```
   podman container list --all
   ```

   ```
   CONTAINER ID  IMAGE                       COMMAND               CREATED
    STATUS                     PORTS                     NAMES
   95038cde3755  localhost/jupyter:latest  /bin/sh -c cd $NO...  2 hours ago
    Exited (0) 23 minutes ago  0.0.0.0:8888->8888/tcp  jupyter
   ```

   b. Remove the selected container.

   ```
   podman container rm CONTAINER-ID
   ```

Replace *CONTAINER-ID* with the appropriate container ID.

2. Delete an application container image.

   a. Identify the container image you want to delete.

```
podman image list
```

```
REPOSITORY                      TAG         IMAGE ID      CREATED         SIZE
localhost/jupyter               latest      d8bbefc6be69  2 hours ago     837
 MB
registry.suse.com/bci/python    latest      35ebb6a7981e  3 hours ago     125
 MB
```

   b. Remove the selected container image.

```
podman image rm IMAGE-ID
```

Be sure to replace *IMAGE-ID* with the appropriate container image ID.

```
Deleted: d8bbefc6be69e61120bad0449e227dda02097a41cfce3cd019dca5f427a7cd9d
Deleted: ce0883c81e5160373abf23e7ebb3c760130981cd587a52dd7cddda1af5a097bb
Deleted: c7de32119cd1a574bb5ad8172bcd94295322875a1a1cefc1e4ea3cb112fe5082
Deleted: fe38a6f6a6f22899da836dabdd4e590176ce9a8f7fa218bafec8cc5d6ed2615f
Deleted: 93aa38c86937ff21f60d36464cd7a19cadd3e6a3929a7d774b3654cf9340cbdd
Deleted: 4e55ce66ad92c43e5f790021f65b29dfa14dbb230b5838d7b83bc86b26bbe8c0
Deleted: ccce1f01ac19afae8d46cdac27fe734ef39ffd736b8d03e92524912d93c5e0c1
Deleted: 13850ddeaac18d17b52569d6f65b9c0bcd585f1259f037e7a1e090c512bfb67a
Deleted: 9c808de46ef1ff01a2e7cb90fbdc2a60157748f8052379fd896c0cdbdf57da00
Deleted: 9837e4f38101da542ed17059a09c666ee07245a7831aeb34f79dbfa774acc4d5
Deleted: a00eadade3fe2bfd4f6f42905691c143b2826fb120913d7b6cd087b773550a77
```

The command output lists the image IDs of deleted container images.
Recall that container images are built of layers, and each layer is a container image. If these "parent" container images are not used by another image, they are deleted as well.

Important

The base container image, 'registry.suse.com/bci/python', is not deleted.

Podman provides many commands (https://docs.podman.io/en/latest/Commands.html) to help you manage your containers and container images. Some Podman commands also have synonyms that can save you a few keystrokes (for example, `podman rmi IMAGE-ID` is the same as `podman image rm IMAGE-ID`).

6 Sharing

Now that you have an application container image and you have verified that it works, you can share it with your team, fellow researchers, or others for whom it may be of value. Some sharing options are outlined here.

1. Use the source.
   An easy method is to simply share the source files you used to create your container image. These are small text files that you could share via e-mail, a file share service, a Git repository, and so on. The recipient can use the received files and follow the same steps you used to create and run the container.

2. Use an image archive.
   Use podman image save (https://docs.podman.io/en/latest/markdown/podman-save.1.html) to save your container image and its parent layers to a TAR archive that you can share. The recipient of the TAR archive can load the image with podman image load (https://docs.podman.io/en/latest/markdown/podman-load.1.html) and run it with `podman run`.

Tip

Your recipient would likely appreciate receiving a copy of your launch script or, at least, some guidance on the options to pass with `podman run`.

   a. Save your container image to a TAR archive.

```
podman image save --output jupyter.tar CONTAINER-ID
```

Tip

You could also refer to your container by name.

For example:

```
podman image save --output jupyter.tar localhost/jupyter:latest
```

```
Copying blob d555e1b0b42f done
Copying blob fee5cefcf4f2 done
Copying blob c20cfe968665 done
Copying blob 9fe0f3109612 done
Copying blob 85cf7f4629b6 done
Copying blob e66fef9acae6 done
Copying blob 5f70bf18a086 done
Copying config 9952705ff5 done
Writing manifest to image destination
Storing signatures
```

b. Send the image archive to the recipient.

Tip

Image archive files can be large. Even compressed, they may be too large to send via e-mail. Make sure to choose a sharing method that both you and your intended recipient can use.

c. When you receive an image archive, use Podman to load it into your local registry.

```
podman image load --input jupyter.tar
```

```
Getting image source signatures
Copying blob e66fef9acae6 done
Copying blob 85cf7f4629b6 done
Copying blob fee5cefcf4f2 done
Copying blob 9fe0f3109612 done
Copying blob c20cfe968665 done
Copying blob d555e1b0b42f done
Copying blob 5f70bf18a086 done
Copying config 9952705ff5 done
Writing manifest to image destination
Storing signatures
Loaded image: localhost/jupyter:latest
```

Tip

It is a good idea to share your launch script or, at least, some guidance on the options the recipient will need to pass to `podman run`.

3. Use a container registry.

   Podman provides the podman push (https://docs.podman.io/en/latest/markdown/podman-push.1.html) command for publishing your container images to a registry and the podman pull (https://docs.podman.io/en/latest/markdown/podman-pull.1.html) command for retrieving images from a registry.

   A container registry hosts repositories for storing and accessing container images. A container registry is often an essential component of cloud native DevOps, enabling automation in continuous integration and continuous delivery/deployment (CI/CD). You can host your own container registry or use a container registry service. Factors to consider when choosing a registry solution include (but are not limited to):

   - Pricing: Are you able to budget operating costs for your needs today and for tomorrow?
   - Content types: Do you need to store more artifacts than only a container image?
   - Availability: Does the registry offer uptime and regional availability to meet your needs?
   - Authentication and authorization: What identity and access management features and integrations are offered?
   - Security: Are security features and controls sufficient to meet your policy and regulatory requirements?
   - Rate limits: Are the allowed number of pushes and pulls per unit of time sufficient?
   - Ease of migration: How difficult would it be to migrate to a different registry?

7 Summary

Modern application development eschews closed, monolithic structures in favor of open, interoperable, microservices architectures designed and implemented with containers. Container technologies accelerate innovation by enabling complex applications and processes to be developed, deployed, and managed across a broad spectrum of infrastructures and services - in the data center, in the cloud, and at the edge. Fast-developing fields, like machine learning, have benefited greatly from the use of container technologies.

Modern, OCI-compliant containers are built in layers on top of a base image. SUSE Linux Enterprise Base Container Images provides a reliable, secure, supportable, and freely redistributable base image for your application containers. There are a variety of tools for building and managing containers. A simple but powerful container management tool is Podman. In this guide you used Podman to build a container image featuring JupyterLab, a Web-based Python development environment. You then used Podman to run, manage and share your application container.

8 Legal notice

9 GNU Free Documentation License

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. Secondarily, 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.