Open Energy Data Initiative Web Portal for Solar Integration (OEDI SI)

Users Guide

Table of Contents

A. Setting up OEDI SI Web Platform

Before a user can run the publicly available algorithms and data on OEDI SI Web Platform, the user needs to setup a few environments on their local workstations.

OEDI SI Cloud Server hosts the main functions of OEDI SI Platform through the landing page at here. OESI AI Platform provides reference algorithms to facilitate system integration of distributed solar generation with high deployment along with composite input data sets as references and new algorithms verifications.

The webpages that are served by the OEDI SI Cloud servers can be accessed at this address. OEDI SI Instructions tab provides detailed instructions to be able to install and run all the components required for OEDI SI simulations. These components are shown in Figure 1:

1. OEDI SI UI Server Docker container
2. OEDI SI Runtime
   a. More details can be found at this github page.
   b. The users need to download these at least once. Downloading these containers initially will take some time.
   c. However, as updates are made to these, the update installations will be incremental.
3. The user will have to install Docker Desktop on the user's workstation (one time process).
   a. A docker container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another.

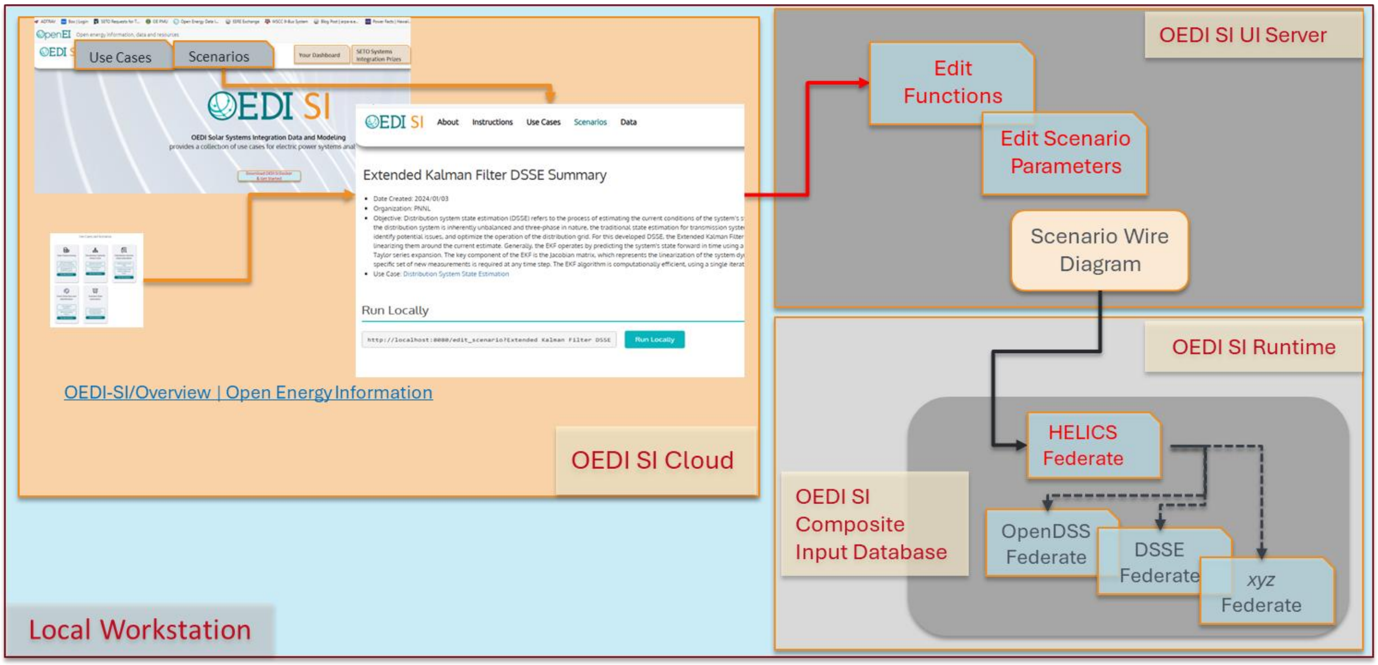


Figure 1: High Level OEDI SI Architecture.

B. Setting Up Docker/Podman Environment

a. Containers - An Introduction

The use cases and their scenarios that OEDI SI makes available to public are exported as single docker image, referred to as UIRuntime Container in reference documents for OEDI SI. UIRuntime Container can simulate the scenarios on OEDI SI by running the image that scenario mimicked initially, thus, reproducing the results that are exactly the same as the original simulation run.

A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. This makes it easy to replicate results of a scenario on many different systems without having to load all the different dependencies that each component may require.

The scenarios that OEDI SI have created have been exported as docker/podman images. Docker/Podman can load these images into containers which then simulate the scenario that has been created. Users can load and run these docker images to replicate the scenarios that have been created.

b. Installing Docker on your Workstation

i. Installing Docker on Ubuntu

To install docker on a linux environment, firstly update the linux package manager and install dependencies to allow the package manager to install them over https:

```
$ sudo apt-get update
$ sudo apt-get install ca-certificates curl gnupg lsb-release
```

Then add docker's GPG keys and set up the repository with the commands:

```
$ sudo mkdir -p /etc/apt/keyrings
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
/etc/apt/keyrings/docker.gpg
```

Finally update the package index and install the latest version of docker:

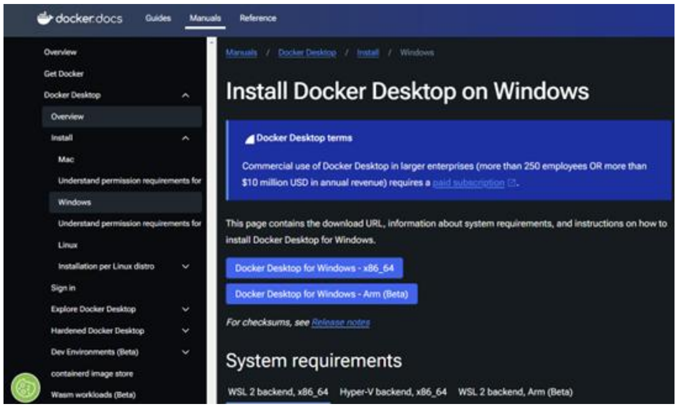
```
$ sudo apt-get update
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-compose-plugin
```

See https://docs.docker.com/engine/install/ for more detail and installation instructions for other distros in the "Installation per distro" entry on the left hand tab.

ii. Installing Docker on Windows

If you are running a windows machine, you will need to install docker desktop, which will emulate an environment for you to run docker containers in. Below is a detailed guide for the installation process:

1. Go to the docker website: https://docs.docker.com/desktop/install/windows-install/



Install Docker Desktop on Windows

Docker Desktop terms

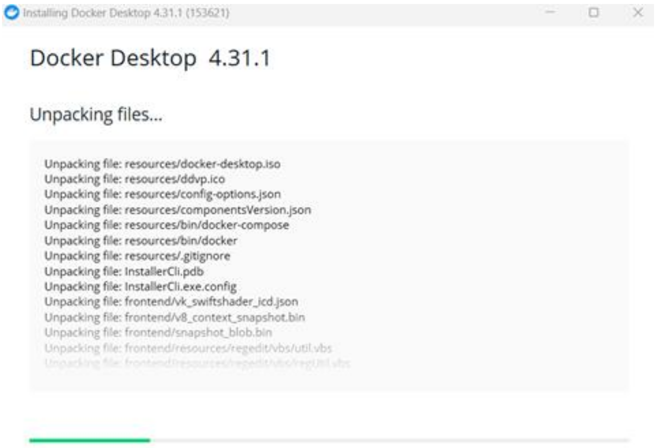
Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains the download URL, information about system requirements, and instructions on how to install Docker Desktop for Windows.

For checksums, see Release notes

System requirements

2. Click on the button to download the installer and start installation:

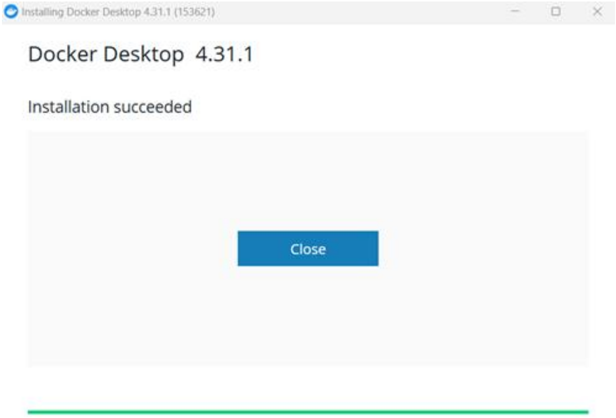


Docker Desktop 4.31.1

Unpacking files...

```
Unpacking file: resources/docker-desktop.iso
Unpacking file: resources/ddvp.ico
Unpacking file: resources/config-options.json
Unpacking file: resources/componentsVersion.json
Unpacking file: resources/bin/docker-compose
Unpacking file: resources/bin/docker
Unpacking file: resources/.gitignore
Unpacking file: InstallerCli.pdb
Unpacking file: InstallerCli.exe.config
Unpacking file: frontend/vk_swiftshader_icd.json
Unpacking file: frontend/v8_context_snapshot.bin
Unpacking file: frontend/snapshot_blob.bin
Unpacking file: frontend/resources/regedit/vbs/util.vbs
Unpacking file: frontend/resources/regedit/vbs/regUtil.vbs
```

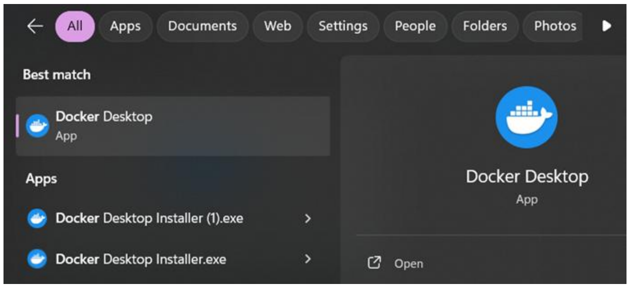
3. Installation complete:



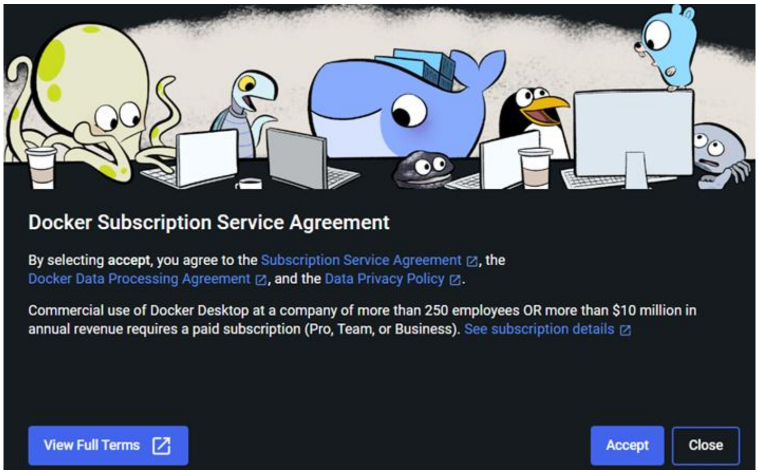
Docker Desktop 4.31.1

Installation succeeded

4. Search for docker desktop and open it:



5. Read and accept the agreement:

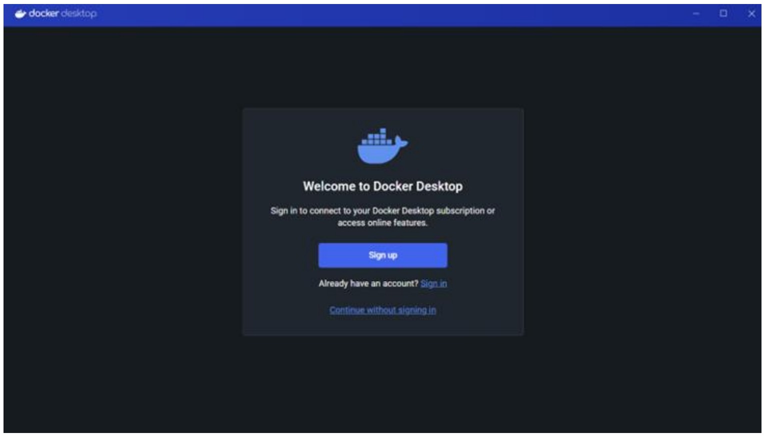


Docker Subscription Service Agreement

By selecting **accept**, you agree to the Subscription Service Agreement, the Docker Data Processing Agreement, and the Data Privacy Policy.

Commercial use of Docker Desktop at a company of more than 250 employees OR more than $10 million in annual revenue requires a paid subscription (Pro, Team, or Business). See subscription details

6. Proceed with account registration or continue without signing in:

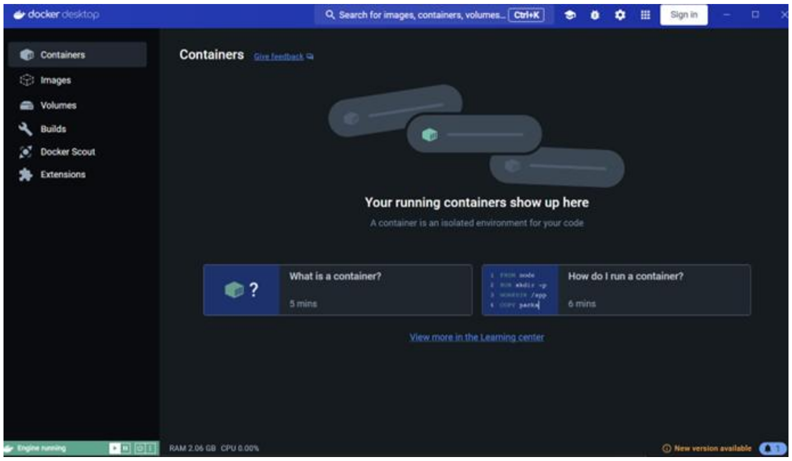


Welcome to Docker Desktop

Sign in to connect to your Docker Desktop subscription or access online features.

Already have an account? Sign in

7. Start using Docker Desktop:



Your running containers show up here

A container is an isolated environment for your code

What is a container?
5 mins

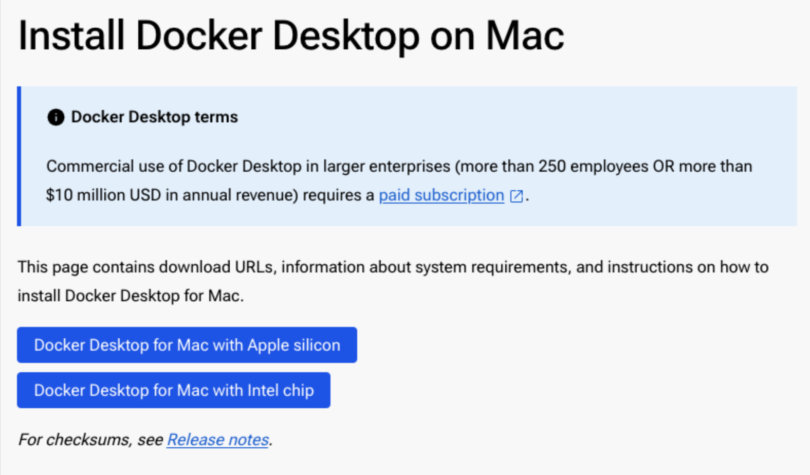
How do I run a container?
6 mins

View more in the Learning center

iii. Installing Docker Desktop on Mac

Installation link -- https://docs.docker.com/desktop/install/mac-install/

1. Choose the right version for your Mac



Install Docker Desktop on Mac

Docker Desktop terms

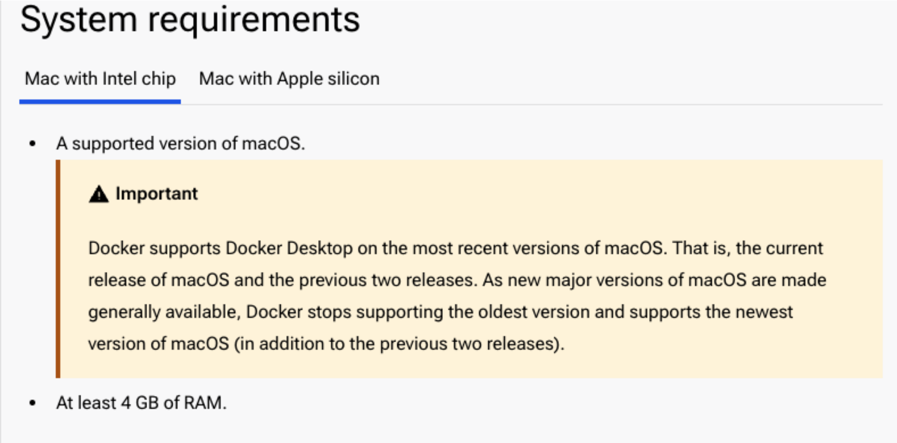
Commercial use of Docker Desktop in larger enterprises (more than 250 employees OR more than $10 million USD in annual revenue) requires a paid subscription.

This page contains download URLs, information about system requirements, and instructions on how to install Docker Desktop for Mac.

Docker Desktop for Mac with Apple silicon
Docker Desktop for Mac with Intel chip

For checksums, see Release notes.

2. Check system requirements
   Check the system requirements based on the chip type



System requirements

- A supported version of macOS.

  Important

  Docker supports Docker Desktop on the most recent versions of macOS. That is, the current release of macOS and the previous two releases. As new major versions of macOS are made generally available, Docker stops supporting the oldest version and supports the newest version of macOS (in addition to the previous two releases).

- At least 4 GB of RAM.

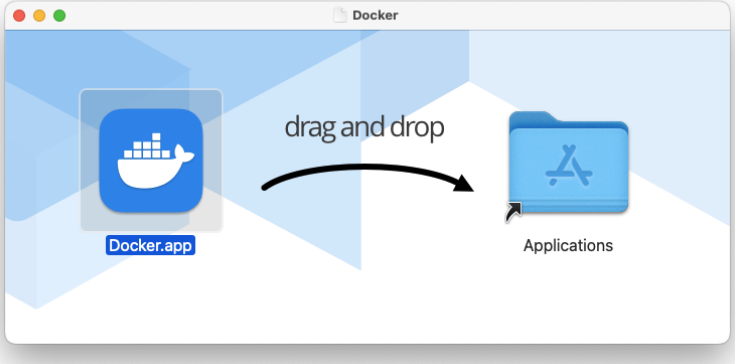
3. Install and run Docker Desktop on Mac

   Two ways: Install interactively, or install from the command line

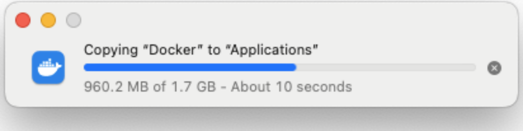
   1) Download the installer using the download buttons

Docker.dmg 502.7 MB Disk Image Today at 9:22 AM

   2) Double-click Docker.dmg to open the installer, then drag the Docker icon to the **Applications** folder. By default, Docker Desktop is installed at /Applications/Docker.app.



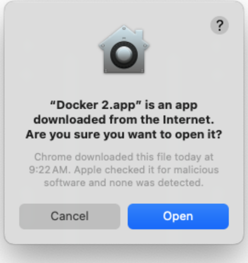
   The installation process.



Copying "Docker" to "Applications"

960.2 MB of 1.7 GB - About 10 seconds

   3) Double-click Docker.app in the **Applications** folder to start Docker.



"Docker 2.app" is an app downloaded from the Internet. Are you sure you want to open it?

Chrome downloaded this file today at 9:22 AM. Apple checked it for malicious software and none was detected.

   4) The Docker menu displays the Docker Subscription Service Agreement.
   5) Select **Accept** to continue.
   6) From the installation window, select either:
      - **Use recommended settings (Requires password)**. This lets Docker Desktop automatically set the necessary configuration settings.
      - **Use advanced settings**. You can then set the location of the Docker CLI tools either in the system or user directory, enable the default Docker socket, and enable

privileged port mapping. See Settings, for more information and how to set the location of the Docker CLI tools.

7) Select **Finish**. If you have applied any of the above configurations that require a password in step 6, enter your password to confirm your choice.

c. Installing Podman on your Workstation

i. Installing Podman on Ubuntu

```
sudo apt-get -y install podman
```

ii. Installing Podman on Windows

For latest and complete documentation refer to https://github.com/containers/podman/blob/main/docs/tutorials/podman-for-windows.md

Podman for Windows

While “containers are Linux,” Podman also runs on Mac and Windows, where it provides a native CLI and embeds a guest Linux system to launch your containers. This guest is referred to as a Podman machine and is managed with the `podman machine` command. On Windows, each Podman machine is backed by a virtualized Windows Subsystem for Linux (WSLv2) distribution. The podman command can be run directly from your Windows PowerShell (or CMD) prompt, where it remotely communicates with the podman service running in the WSL environment. Alternatively, you can access Podman directly from the WSL instance if you prefer a Linux prompt and Linux tooling. In addition to command-line access, Podman also listens for Docker API clients, supporting direct usage of Docker-based tools and programmatic access from your language of choice.

Prerequisites

Since Podman uses WSL, you need a recent release of Windows 10 or Windows 11. On x64, WSL requires build 18362 or later, and 19041 or later is required for arm64 systems. Internally, WSL uses virtualization, so your system must support and have hardware virtualization enabled. If you are running Windows on a VM, you must have a VM that supports nested virtualization.

It is also recommended to install the modern “Windows Terminal,” which provides a superior user experience to the standard PowerShell and CMD prompts, as well as a WSL prompt, should you want it.

You can install it by searching the Windows Store or by running the following `winget` command:

```
winget install Microsoft.WindowsTerminal
```

Installing Podman

Installing the Windows Podman client begins by downloading the Podman Windows installer. The Windows installer is built with each Podman release and can be downloaded from the official GitHub release page. Be sure to download a 4.1 or later release for the capabilities discussed in this guide.

Installing Podman 4.1.0

Installing Podman 4.1.0

Once downloaded, simply run the EXE file, and relaunch a new terminal. After this point, podman.exe will be present on your PATH, and you will be able to run the `podman machine init` command to create your first machine.

```
PS C:\Users\User> podman machine init
```

Automatic WSL Installation

If WSL has not been installed on your system, the first machine init command will prompt a dialog to begin an automated install. If accepted, this process will install the necessary Windows components, restart the system, and after login, relaunch the machine creation process in a terminal window. Be sure to wait a minute or two for the relaunch to occur, as Windows has a delay before executing startup items. Alternatively, you can decline automatic installation and install WSL manually. However, this will require additional download and setup time.

Machine Init Process

After WSL is installed, the init command will install a minimal installation of Fedora, customizing it to run podman.

```
PS C:\Users\User> podman machine init
Extracting compressed file
Importing operating system into WSL (this may take 5+ minutes on a new WSL in
stall)...
Installing packages (this will take a while)...
Complete!
Configuring system...
Generating public/private ed25519 key pair.
Your identification has been saved in podman-machine-default
Your public key has been saved in podman-machine-default.pub
The key fingerprint is:
SHA256:RGTGg2Q/LX7ijN+mzu8+BzcS3cEWP6Hir6pYllJtceA root@WINPC
Machine init complete
To start your machine run:

        podman machine start
```

Starting Machine

After the machine init process completes, it can then be started and stopped as desired:

```
PS C:\Users\User> podman machine start

Starting machine "podman-machine-default"

This machine is currently configured in rootless mode. If your containers
require root permissions (e.g. ports < 1024), or if you run into compatibilit
y
issues with non-podman clients, you can switch using the following command:

        podman machine set --rootful

API forwarding listening on: npipe:////./pipe/docker_engine

Docker API clients default to this address. You do not need to set DOCKER_HOS
T.
Machine "podman-machine-default" started successfully
```

First Podman Command

From this point on, podman commands operate similarly to how they would on Linux.

For a quick working example with a small image, you can run the Linux date command on PowerShell.

```
PS C:\Users\User> podman run ubi8-micro date
Thu May 5 21:56:42 UTC 2022
```

Port Forwarding

Port forwarding also works as expected; ports will be bound against localhost (127.0.0.1). Note: When running as rootless (the default), you must use a port greater than 1023. See the Rootful and Rootless section for more details.

To launch httpd, you can run:

```
PS C:\Users\User> podman run --rm -d -p 8080:80 --name httpd docker.io/librar
y/httpd
f708641300564a6caf90c145e64cd852e76f77f6a41699478bb83a162dceada9
```

A curl command against localhost on the PowerShell prompt will return a successful HTTP response:

```
PS C:\Users\User> curl http://localhost:8080/ -UseBasicParsing

StatusCode : 200
```

```
StatusDescription : OK
Content : <html><body><h1>It works!</h1></body></html>
```

As with Linux, to stop, run:

```
podman stop httpd
```

Using API Forwarding

API forwarding allows Docker API tools and clients to use podman as if it was Docker. Provided there is no other service listening on the Docker API pipe; no special settings will be required.

```
PS C:\Users\User> .\docker.exe run -it fedora echo "Hello Podman!"
Hello Podman!
```

Otherwise, after starting the machine, you will be notified of an environment variable you can set for tools to point to podman. Alternatively, you can shut down both the conflicting service and podman, then finally run `podman machine start` to restart, which should grab the Docker API address.

```
Another process was listening on the default Docker API pipe address.
You can still connect Docker API clients by setting DOCKER HOST using the
following PowerShell command in your terminal session:

        $Env:DOCKER_HOST = 'npipe:////./pipe/podman-machine-default'

Or in a classic CMD prompt:

        set DOCKER_HOST=npipe:////./pipe/podman-machine-default

Alternatively, terminate the other process and restart podman machine.
Machine "podman-machine-default" started successfully

PS C:\Users\User> $Env:DOCKER_HOST = 'npipe:////./pipe/podman-machine-default
'
PS C:\Users\User>.\docker.exe version --format '{{(index .Server.Components 0
).Name}}'
Podman Engine
```

Rootful & Rootless

On the embedded WSL Linux distro, podman can either be run under the root user (rootful) or a non-privileged user (rootless). For behavioral consistency with Podman on Linux, rootless is the default. Note: Rootful and Rootless containers are distinct and isolated from one another. Podman commands against one (e.g., podman ps) will not represent results/state for the other.

While most containers run fine in a rootless setting, you may find a case where the container only functions with root privileges. If this is the case, you can switch the machine to rootful by stopping it and using the set command:

```
podman machine stop
podman machine set --rootful
```

To restore rootless execution, set rootful to false:

```
podman machine stop
podman machine set --rootful=false
```

Another case in which you may wish to use rootful execution is binding a port less than 1024. However, future versions of podman will likely drop this to a lower number to improve compatibility with defaults on system port services (such as MySQL)

Volume Mounting

New in Podman v4.1 is the ability to perform volume mounts from Windows paths into a Linux container. This supports several notation schemes, including:

Windows Style Paths:

```
podman run --rm -v c:\Users\User\myfolder:/myfolder ubi8-micro ls /myfolder
```

Unixy Windows Paths:

```
podman run --rm -v /c/Users/User/myfolder:/myfolder ubi8-micro ls /myfolder
```

Linux paths local to the WSL filesystem:

```
podman run --rm -v /var/myfolder:/myfolder ubi-micro ls /myfolder
```

All of the above conventions work, whether running on a Windows prompt or the WSL Linux shell. Although when using Windows paths on Linux, appropriately quote or escape the Windows path portion of the argument.

Listing Podman Machine(s)

To list the available podman machine instances and their current resource usage, use the `podman machine ls` command:

```
PS C:\Users\User> podman machine ls


NAME                    VM TYPE     CREATED      LAST UP            CPUS
MEMORY      DISK SIZE
podman-machine-default  wsl         2 hours ago  Currently running  4
331.1MB     768MB
```

Since WSL shares the same virtual machine and Linux kernel across multiple distributions, the CPU and Memory values represent the total resources shared across running systems. The opposite applies to the Disk value. It is independent and represents the amount of storage for each individual distribution.

Accessing the Podman Linux Environment

While using the podman.exe client on the Windows environment provides a seamless native experience supporting the usage of local desktop tools and APIs, there are a few scenarios in which you may wish to access the Linux environment:

Updating to the latest stable packages on the embedded Fedora instance
Using Linux development tools directly
Using a workflow that relies on EXT4 filesystem performance or behavior semantics

There are three mechanisms to access the embedded WSL distribution: 1. SSH using `podman machine ssh` 2. WSL command on the Windows PowerShell prompt 3. Windows Terminal Integration

Using SSH

SSH access provides a similar experience as Podman on Mac. It immediately drops you into the appropriate user based on your machine's rootful/rootless configuration (root in the former, 'user' in the latter). The –username option can be used to override with a specific user.

An example task using SSH is updating your Linux environment to pull down the latest OS bugfixes:

```
podman machine ssh sudo dnf upgrade -y
```

Using the WSL Command

The `wsl` command provides direct access to the Linux system but enters the shell as root first. This is due to design limitations of WSL, where running systemd (Linux's system services) requires the usage of a privileged process namespace.

Unless you have no other distributions of WSL installed, it's recommended to use the `-d` option with the name of your podman machine (podman-machine-default is the default)

```
PS C:\Users\User> wsl -d podman-machine-default
```

You will be automatically entered into a nested process namespace where systemd is running. If you need to access the parent namespace, hit `ctrl-d` or type exit. This also means to log out, you need to exit twice.

```
[root@WINPC /]# podman --version
podman version 4.1.0
```

To access commands as the non-privileged user (rootless podman), you must first type `su user`. Alternatively, you can prefix the `wsl` command to use the special `enterns`:

```
wsl -d podman-machine-default enterns su user
[user@WINPC /]$ id
uid=1000(user) gid=1000(user) groups=1000(user),10(wheel)
```

Likewise, running commands as root without entering a prompt should also be prefixed with `enterns`.

```
wsl -d podman-machine-default enterns systemctl status
```

Accessing the WSL instance as a specific user using `wsl -u` or using inline commands without `enterns` is not recommended since commands will execute against the incorrect namespace.

Using Windows Terminal Integration

Entering WSL as root is a 2-click operation. Simply click the drop-down tag, and pick ‘podman-machine-default,’ where you will be entered directly as root.

Using WSL in Windows Terminal

Using WSL in Windows Terminal

As before, to switch to a non-privileged user for rootless podman commands, type `su user`.

```
[root@WINPC /]# su user
[user@WINPC /]$ podman info --format '{{.Store.RunRoot}}'
/run/user/1000/containers
```

Stopping a Podman Machine

To stop a running podman machine, use the `podman machine stop` command:

```
PS C:\Users\User> podman machine stop
Machine "podman-machine-default" stopped successfully
```

Removing a Podman Machine

To remove a machine, use the `podman machine rm` command:

```
PS C:\Users\User> podman machine rm

The following files will be deleted:

C:\Users\User\.ssh\podman-machine-default
C:\Users\User\.ssh\podman-machine-default.pub
C:\Users\User\.local\share\containers\podman\machine\wsl\podman-machine-defau
lt_fedora-35-x86_64.tar
C:\Users\User\.config\containers\podman\machine\wsl\podman-machine-default.js
```

```
on
C:\Users\User\.local\share\containers\podman\machine\wsl\wsldist\podman-machi
ne-default


Are you sure you want to continue? [y/N] y
```

Troubleshooting

Recovering from a failed auto-installation of WSL

If auto-install fails and retrying is unsuccessful, you can attempt to reset your WSL system state and perform a manual WSL installation using the `wsl --install` command. To do so, perform the following steps:

Launch PowerShell as administrator

```
Start-Process powershell -Verb RunAs
```

Disable WSL Features

```
dism.exe /online /disable-feature /featurename:Microsoft-Windows-Subsystem-Li
nux /norestart
dism.exe /online /disable-feature /featurename:VirtualMachinePlatform /norest
art
```

Reboot
Run manual WSL install

```
wsl --install
```

Continue with podman machine init

Install Certificate Authority

Instructions for installing a CA certificate can be found here.

C. OEDI SI Single Container

For latest and complete documentation refer to https://github.com/openEDI/oedi-si-single-container/tree/main/docs.

Running Directly Using Docker/Podman

Single container consists of **build**, **runtime** and **orchestration** tools. If the user is interested in only the **orchestration** portion, this can be accomplished without using the CLI by directly interacting with Docker/Podman. If you are using Podman instead of Docker, replace all docker commands with podman commands i.e. instead of *docker pull* use *podman pull*

UIRuntime

The UIRuntime images are available here

You would need to run the following command to get the latest version of the image,

```
docker pull openenergydatainitiative/uiruntime:latest
```

UIServer

The UIServer images are available here

```
docker pull openenergydatainitiative/uiserver:latest
```

Networking

Containers provide security through namespace isolation. We create a user defined bridge network for different modes (rootful/rootless) of Docker/Podman to work without issues. Here we are creating a subnet at the private IP space 172.20.0.0/24 i.e. 172.20.0.0 - 172.20.0.255. This allows us to assign the network and IP to the containers during runtime.

```
docker network create --gateway 172.20.0.1 --subnet 172.20.0.0/24
oedisi_local_network
```

P.S. You will need to do this only once.

Run the Container

uiruntime

```
docker run --rm --name=uiruntime --net=oedisi_local_network --ip=172.20.0.2 -
p 12500:12500 -p 8888:8888 openenergydatainitiative/uiruntime:latest
```

uiserver

```
docker run --rm --name=uiserver --net=oedisi_local_network --ip=172.20.0.3 -p
8080:80 -p 8088:8088 openenergydatainitiative/uiserver:latest
```

Stop the Container

uiruntime

```
docker stop -t=0 uiruntime
```

uiserver

```
docker stop -t=0 uiserver
```

Loading the webpage

Open your browser and go to the following address,

For edit scenario,

```
localhost:8080
```

For analysis (viewing logs and visualization). This page can also be reached by clicking the “Dashboard” button on localhost:8080.

```
localhost:8080/analysis
```

If localhost does not resolve as expected, then replace it with 127.0.0.1

The analysis page comes with an interactive wiring diagram (flow chart on the left side of the screen). Each block in the wiring diagram is clickable. The context of the click is dependent on the “View Type” selection on the right panel i.e. log or plot. When the view type selection is log, a click event on the wiring diagram block will display that federates log output. The plot event context is available only for the rightmost recorder federates i.e. federates with the naming convention "recorder_*".

Examples

You can find sample configurations at the following folder,

```
examples/edit_scenario/
```

Troubleshooting

Conflict, container name already in use

This implies that a previously run container is either still running or stopped but not removed. The following command will list running containers,

```
docker ps
```

If **uiruntime** and/or **uiserver** is in the list, then you can use,

```
docker stop -t=0 uiruntime
docker stop -t=0 uiserver
```

The **ps** command with **-a** flag will list all containers, even the stopped ones.

```
docker ps -a
```

If **uiruntime** and/or **uiserver** is in the list, then you can use,

```
docker container rm uiruntime
docker container rm uiserver
```

Getting system_runner

```
http://127.0.0.1:12500/get_system_runner?uuid=<uuid>
```