



ELISA
Enabling **Linux** in
Safety Applications

WORKSHOP

NASA Goddard

Containerization in Space. Podman for Mission-Critical Operations and Resilience

Dan Walsh <dwalsh@redhat.com>

Douglas Schilling Landgraf <dougslan@redhat.com>



Please read
out loud all
text in

RED

I Promise

To say
Make a copy
Rather than
Make a Xerox

I Promise

To say
Tissue
Rather than
Kleenex

I Promise

To say
Container Registries
Rather than
Docker registries

I Promise

To say
Container Images
Rather than
Docker images

I Promise

To say

Containers

Or

OCI Containers

Rather than

Docker Containers

I Promise

To
give this presentation
a
5 Star review

Sit Down

```
$ podman run -ti docker.io/osrf/space-ros
Trying to pull docker.io/osrf/space-ros:latest...
Getting image source signatures
Copying blob bd159e9d0602 done |
Copying blob b65f1d11f71a done |
Copying blob aae34eb940be done |
Copying blob a7f2481d1ef1 done |
```

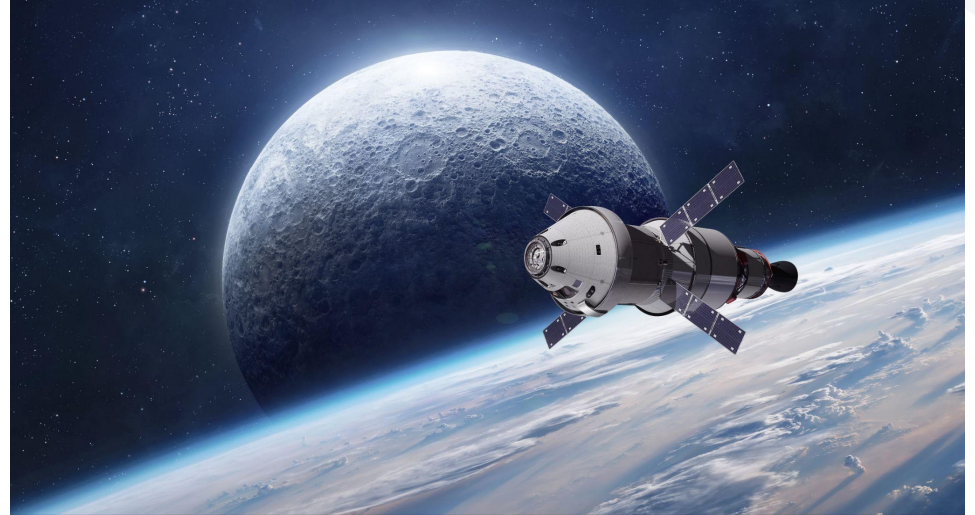
...

```
spaceros-user@194fad252765:~$ ros2 --help
usage: ros2 [-h] [--use-python-default-buffering] Call `ros2 <command> -h` for
more detailed usage. ...
```

ros2 is an extensible command-line tool for ROS 2.

...

How can Podman and friends help in critical mission?



Spacecraft on orbit of Earth planet.

Dan Walsh: Ursula of containers

Tentacles in every project



Dan Walsh: Ursula of containers

Tentacles in every project



Dan Walsh: Ursula of containers Tentacles in every project



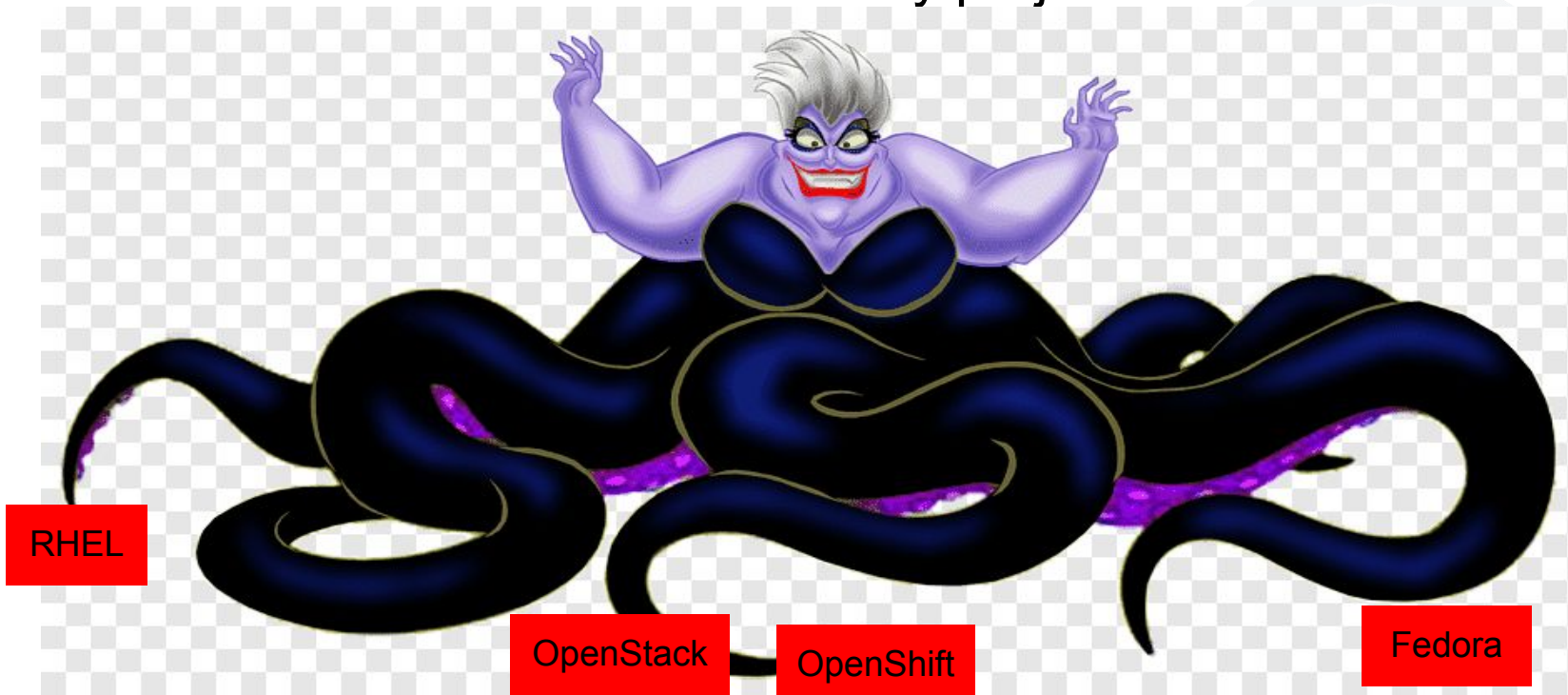
Dan Walsh: Ursula of containers

Tentacles in every project



Dan Walsh: Ursula of containers

Tentacles in every project



Dan Walsh: Ursula of containers

Tentacles in every project



Dan Walsh: Ursula of containers

Tentacles in every project



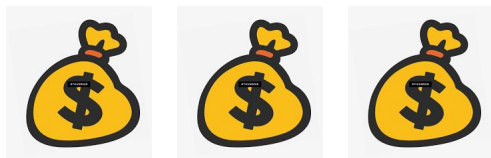
Dan Walsh: Ursula of containers

Tentacles in every project



Shameless Plug Buy my book

Podman in Action

A screenshot of the Manning website product page for the book 'Podman in Action'. The page shows the book cover, a 'look inside' button, a 5-star rating, and a price of \$29.99 (discounted from \$47.99). It also features a 'BECOME A REVIEWER' button, a 'you might also like' section, and a 'RECENTLY VIEWED' sidebar. The URL in the browser is https://www.manning.com/books/podman-in-action.

Podman in Action **you own this product**
Secure, rootless containers for Kubernetes, microservices, and more
★★★★★ 1 review 520 views in the last week
Daniel Walsh
December 2022 - ISBN 9781633439689 - 312 pages - printed in black & white
filed under **Operations & Cloud**

eBook **\$29.99** | print + eBook \$41.99

Our eBooks come in DRM-free Kindle, ePub, and PDF formats + *liveBook*, our enhanced eBook format accessible from any web browser.

~~\$47.99~~ **\$29.99**
you save \$18 (38%)

add to cart

\$0.00 with subscription

get all Manning content with a subscription

BECOME A REVIEWER

ePub + Kindle available Feb 7, 2023

you might also like

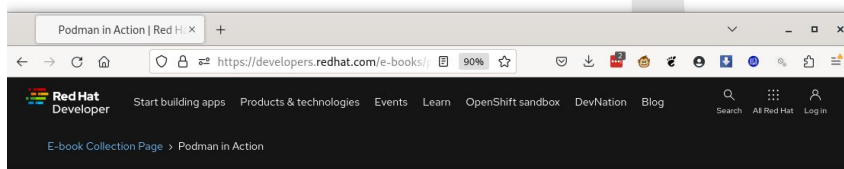
In Cloud Native Spring in Action, you'll learn

<https://www.manning.com/books/podman-in-action>

Shameless Plug Red Hat gives it away:

Podman in Action

Thanks



E-books

Podman in Action

Daniel Walsh

September 8, 2022



Containers, Linux, Kubernetes, Microservices



Get the e-book

[Download the pdf now](#)

About

Dive into Podman, the next-generation container engine that manages containers rootlessly. Podman is an excellent tool for developers looking to containerize their applications securely because it provides extra layers of security unavailable in Docker and other container platforms.

Podman in Action introduces Podman's features and capabilities, including how to work with containers, build container images, and convert containerized applications into either single-node services to run on edge devices or Kubernetes-based microservices. It discusses Podman's unique advantages over Docker and shows how you can easily migrate your Docker-based infrastructure.

Written by Daniel Walsh, who leads the Podman team at Red Hat, this book contains easy-to-follow examples to help you learn Podman quickly, including steps to deploy a complete containerized web service.

Download *Podman in Action* and learn how to:

- Build and run containers in rootless mode
- Develop and manage pods
- Use `systemd` to oversee a container's life cycle
- Work with the Podman service via Python

<https://developers.redhat.com/e-books/podman-action>

Open to
the world

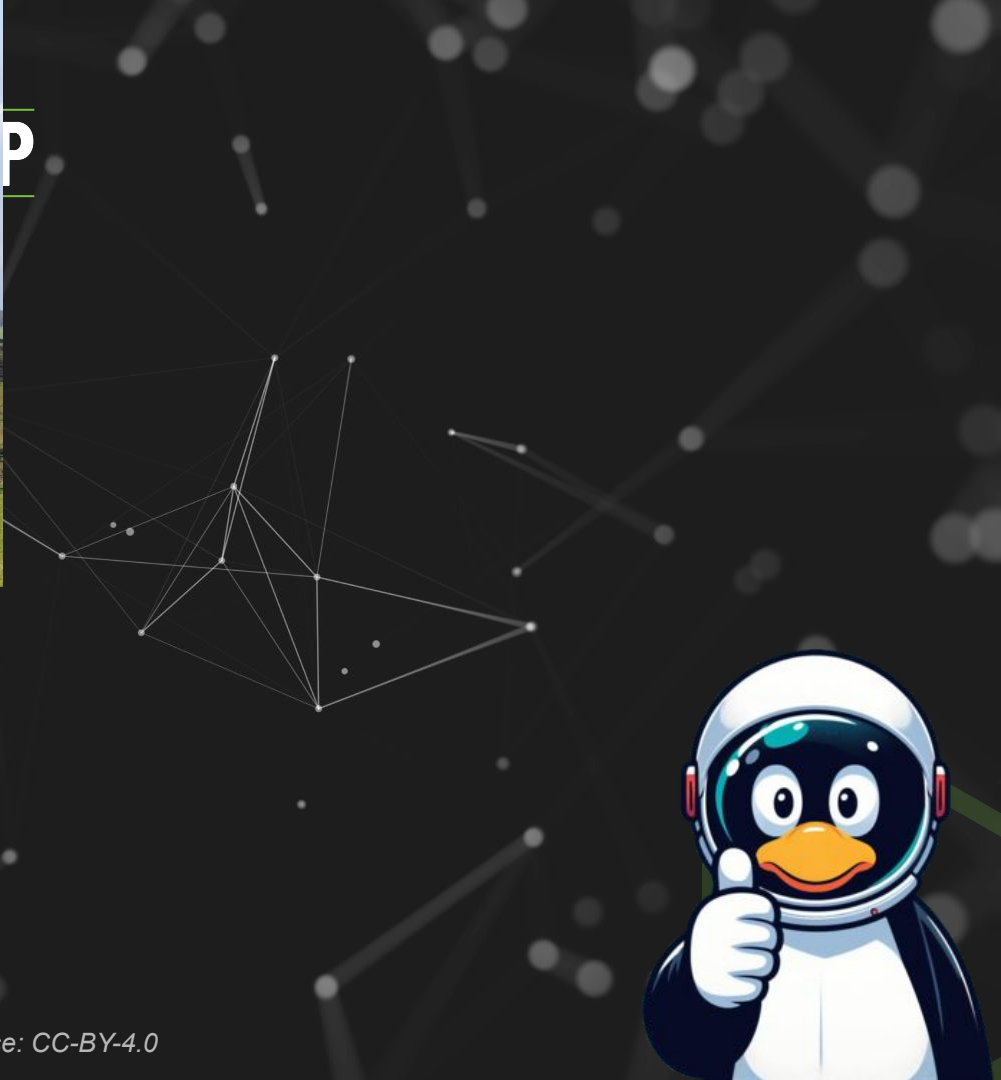
Tech Exchange | 2023







P











How can Podman and friends help in critical mission?

- OCI Compliant
- Minimal footprint
- Rootless
- Support out of box:
 - Namespaces, SELinux, Cgroups, seccomp
- CNCF Donation



KubeCon



CloudNativeCon

North America 2024

SCALING NEW HEIGHTS



Red Hat announces submission of
Podman Desktop, Podman, and Bootc
CNCF Sandbox



ELISA
Enabling **Linux** in
Safety Applications



WORKSHOP

License: CC-BY-4.0

CNCF Donations



podman



bootc

RHEL/RHIVOS with Podman is working thru FuSA (ISO26262-2)



Red Hat is currently working in certify RHEL as platform (includes Podman) for Automobile industry

Red Hat In-Vehicle Operating System

RHIVOS



CENTOS AUTOMOTIVE SIG

by Eric Curtin



**CentOS
Connect**



THANK YOU TO OUR SPONSORS!



Image based (bootc)

Binary distribution

based on

Red Hat Enterprise Linux

RHIVOS

Real Time Kernel

RHIVOS

Design Minimal OS Image as a container

Same OS Platform everywhere

Atomic update

Applications built, tested and deployed as containers

ComposeFS



bootc

ComposeFS

RHIVOS

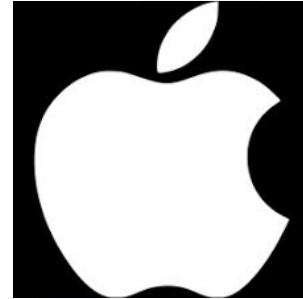
<https://github.com/containers/composefs>

- New file system feature in Linux kernel
 - Built on top of Errorfs and OverlayFS
- File System integrity
 - Supports [fs-verity](#) validation of the content files.
 - Backing content cannot be changed (by mistake or malice) without being detected when file is used.

(very) container friendly

RHIVOS

Build container applications from desktop



Software independent from Base OS

FuSa Functional Safety

Functional Safety is the process of reducing the risks of both simple and complex systems so that they function safely if a hardware, operational, or human failure occurs. When every safety function is carried out as prescribed and the performance standards for each safety function are met, “Functional Safety” has been achieved.

Traditional Functional Safety

- System design documents are written
- Code is produced to meet the requirements
- Tests are written to guarantee that the code functions as designed.

Linux Functional Safety

- ~~System design documents are written~~
Linux system is already written, with no real design document.

Could you please explain why not OpenShift in this case?





kubernetes

In A Vehicle?

Kubernetes is based on "eventual consistency."

- System might be in a inconsistent state, but is progressing toward it.

Kubernetes is based on "eventual consistency."

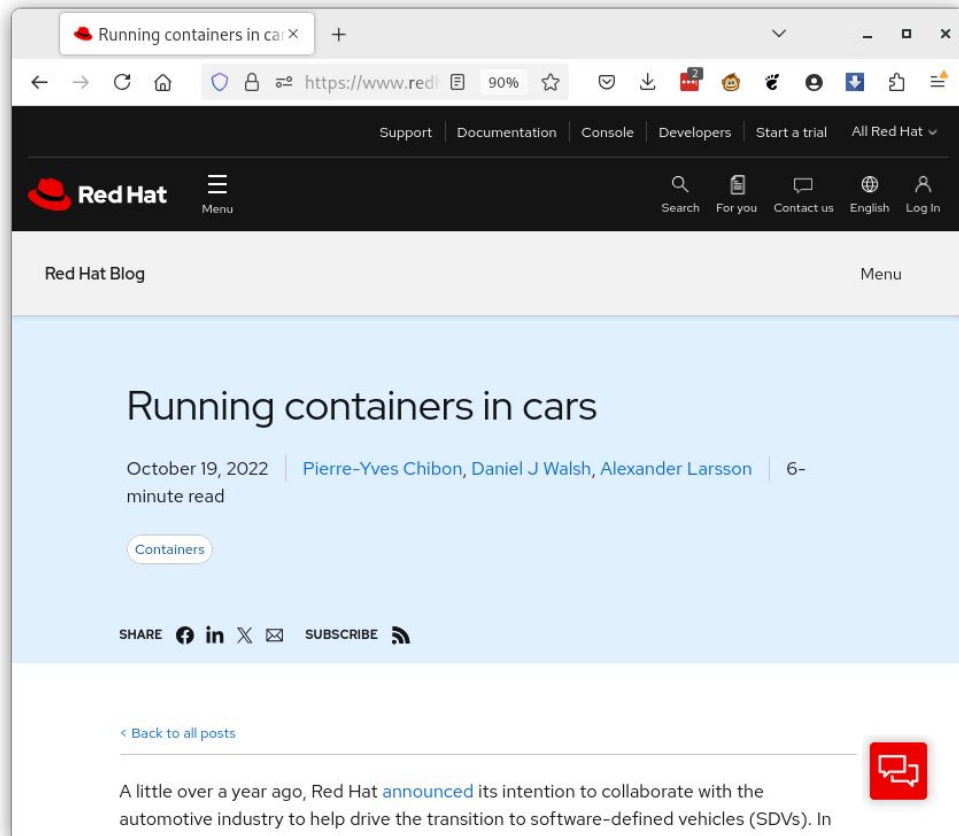
- System might be in a inconsistent state, but is progressing toward it.
- The braking system will **eventually** activate?

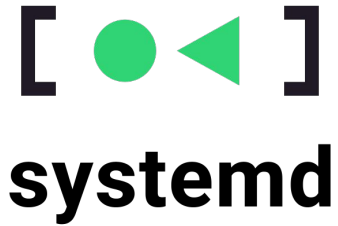
Kubernetes is based on "eventual consistency."

- System might be in a inconsistent state, but is progressing toward it.
- The braking system will **eventually** activate?
 - Not consistent with Functional Safety.



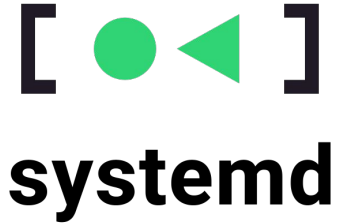
https://www.redhat.com/en/blog/running-containers-cars





- Think of Systemd as your local host orchestrator
- Application profiles
 - 1 Profile \Rightarrow 1 or more applications
 - 1 Application \Rightarrow 1 systemd service file
 - Capability to switch between different profiles
 - Bootup
 - Network
 - Multi-User
 - Graphical User

Think of Systemd as your local host orchestrator

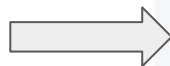
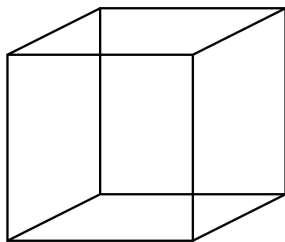


- RHIVOS Application profiles
 - 1 Profile \Rightarrow 1 or more applications
 - 1 Application \Rightarrow 1 systemd service file
 - Capability to switch between different profiles
 - Startup
 - Reverse
 - Drive

What do you get if you squash a Kubernetes **kubelet**?

What do you get if you squash a Kubernetes `kubelet`?

A Podman Quadlet



Quadlet example

```
$ sudo cat /etc/containers/systemd/mysleep.container
```

```
[Unit]
```

```
Description=The sleep container
```

```
After=local-fs.target
```

```
[System]
```

```
Restart=always
```

```
[Container]
```

```
Image=registry.access.redhat.com/ubi9-minimal:latest
```

```
Exec=sleep 1000
```

```
[Install]
```

```
# Start by default on boot
```

```
WantedBy=multi-user.target default.target
```

```
[Unit]
Description=The sleep container
After=local-fs.target
SourcePath=/etc/containers/systemd/mysleep.container
RequiresMountsFor=%t/containers
```

```
[System]
Restart=always
```

```
[X-Container]
Image=registry.access.redhat.com/ubi9-minimal:latest
Exec=sleep 1000
```

```
[Install]
# Start by default on boot
WantedBy=multi-user.target default.target
```

```
[Service]
Environment=PODMAN_SYSTEMD_UNIT=%n
KillMode=mixed
ExecStop=/usr/bin/podman rm -f -i --cidfile=%t/%N.cid
ExecStopPost=-/usr/bin/podman rm -f -i --cidfile=%t/%N.cid
Delegate=yes
Type=notify
NotifyAccess=all
SyslogIdentifier=%N
ExecStart=/usr/bin/podman run --name=systemd-%N --cidfile=%t/%N.cid --replace --rm --cgroups=split
--sdnotify=common -d registry.access.redhat.com/ubi9-minimal:latest sleep 1000
```



ELISA
Enabling **Linux** in
Safety Applications

WORKSHOP

Freedom From Interference using Containers



License: CC-BY-4.0

Why FFI is important for critical systems?

“**Ensuring** that **safety-critical components** operate **independently** and are **not disrupted by faults** or unintended interactions from other system components”

ASIL versus QM

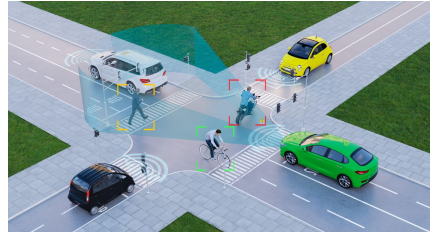
- **Automotive Safety Integrity Level (ASIL)**
 - Functional Safety for Road Vehicles standard.
 - Treating by default all running software on the system while in safety mode as ASIL-B with the exception of the QM software.
- **Quality Management (QM)**
 - All assessed risks are tolerable.
 - Safety assurance controls unnecessary.
 - Standard quality management processes are sufficient for development.

Examples of ASIL Services

Collision Warning Systems



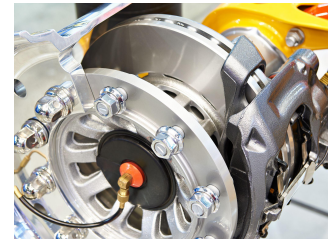
Blind Spot Detection Systems



Airbag Control Systems



Brakes Systems



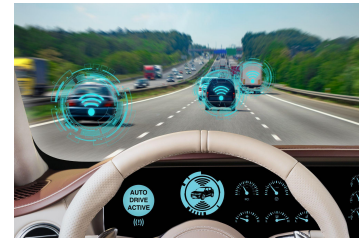
Driver Drowsiness Detection Systems



Rear-View Camera and Parking Assistance



Advanced Driver Assistance Systems



Tyre Pressure Systems



Examples of QM Services

Navigation Systems



Climate Control Systems



Infotainment Systems



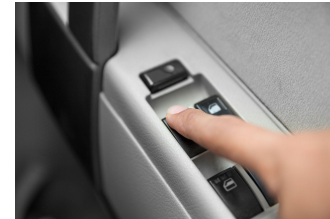
Power Seats



Interior Lighting Systems



Power Window



The QM project

containers **q** Public

Edit Pins

Unwatch 13

Fork 26

Starred 24

main 5 Branches 24 Tags

Go to file t Add file <> Code

Yarboa Merge pull request #670 from Yarboa/kvm-test 7ae2141 · 2 days ago 612 Commits

.fmf	TMT plan to be used by Packit	last year
.github	drop-in-target-build-only: use f40	last month
build-aux	Address pre-commit reported issues	6 months ago
demos	demos: Add slide from DevConf.US FFI	4 months ago
docs	Adding supported quadlet vars mappings	last week
etc	Adding KVM tier-0 rpm subpackage testing	last week
plans	Add fedora fix	2 days ago
pre-commit-hooks	Add packit pre-commit	2 months ago
qm-windowmanager	qm-windowmanager: update content	2 months ago

About

QM is a containerized environment for running Functional Safety qm (Quality Management) software

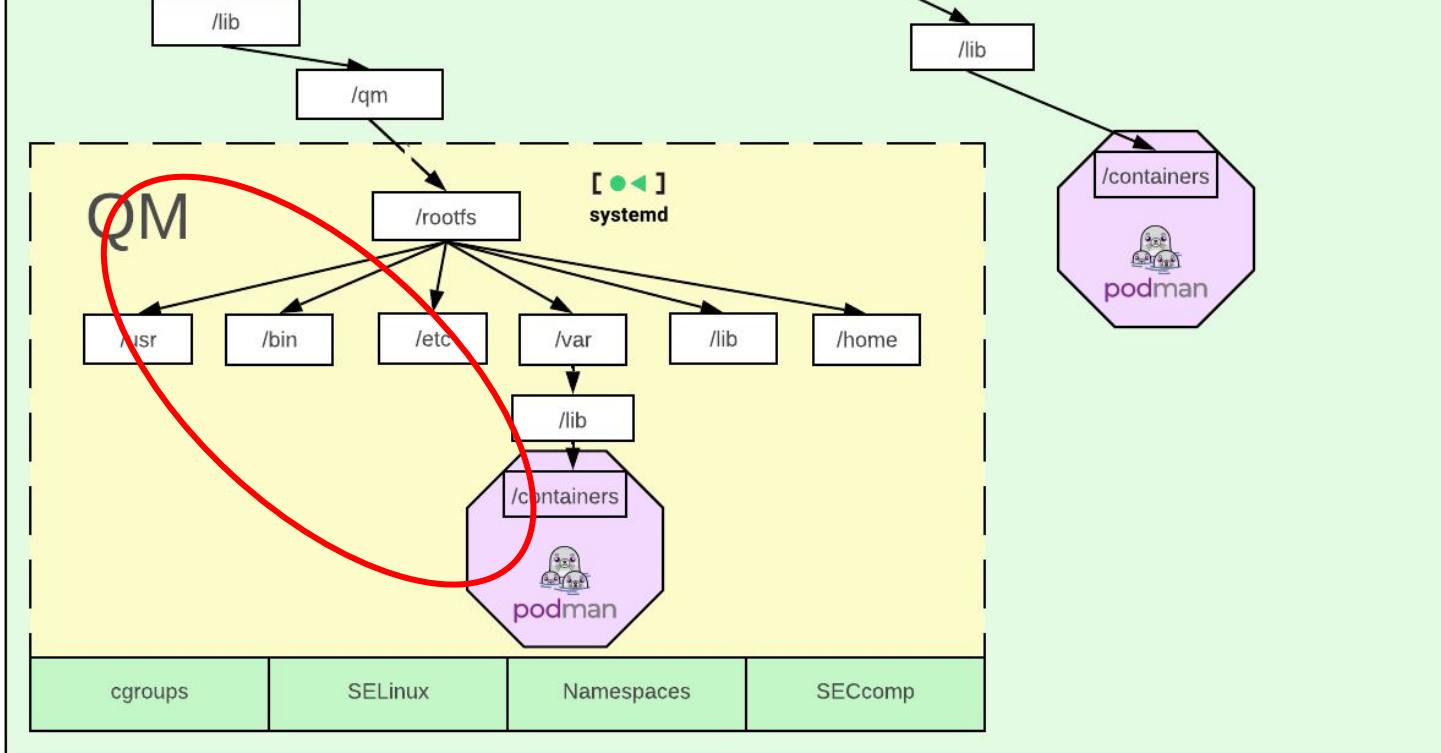
github.com/containers/qm

- linux car embedded container
- secure safety auto safety-critical
- qm secure-by-design podman
- quadlet

- Readme
- GPL-2.0 license
- Code of conduct
- Security policy
- Activity
- Custom properties

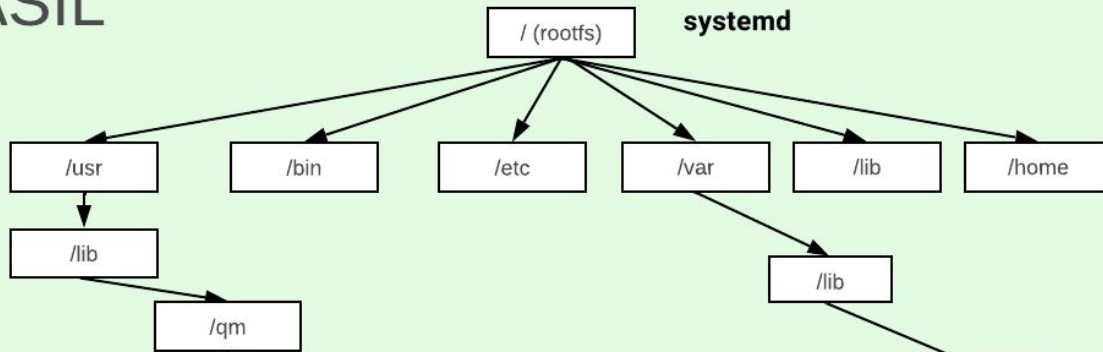


License: CC-BY-4.0



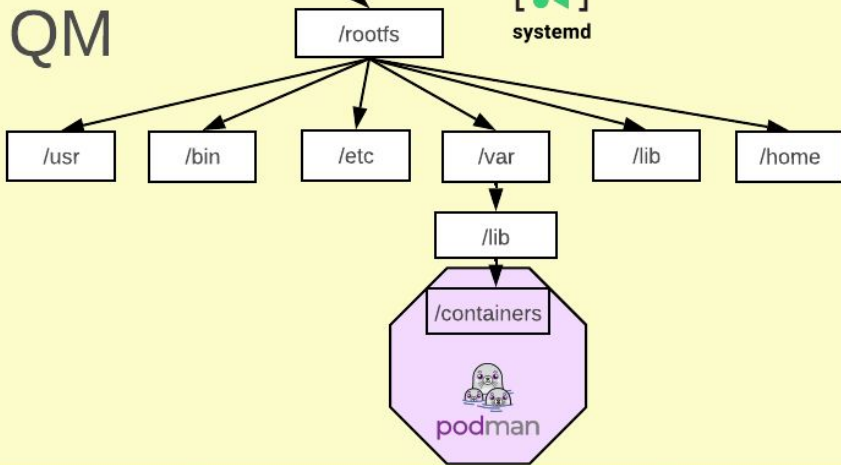
ASIL

[● ◀]
systemd



QM

[● ◀]
systemd

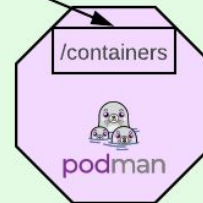


cgroups

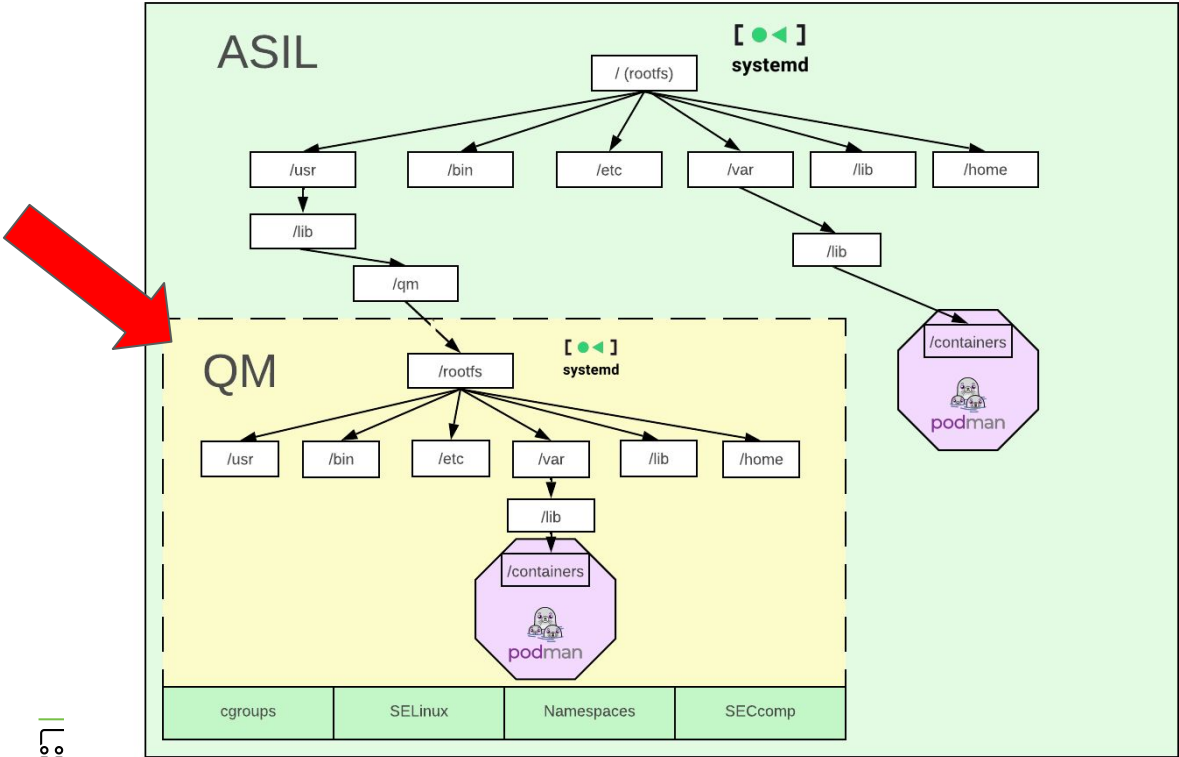
SELinux

Namespaces

SECcomp



Isolate QM environment with Podman Quadlet



Quadlet: Describe a container running within systemd

/usr/share/containers/systemd/qm.container

[Service]

```
AllowedCPUs=6-11
CPUWeight=50
Delegate=true
IOWeight=50
ManagedOOMSwap=kill
MemorySwapMax=0
Slice=QM.slice
Restart=always
OOMScoreAdjust=500
Environment=ROOTFS=/usr/lib/qm/rootfs
```

[Container]

```
AddCapability=all
AddDevice=/dev/kvm
ContainerName=qm
Exec=/sbin/init
Network=host
PodmanArgs=--security-opt label=nested --security-opt unmask=all
ReadOnly=true
Rootfs=${ROOTFS}
SecurityLabelFileType=qm_file_t
SecurityLabelLevel=s0
SecurityLabelType=qm_t
```

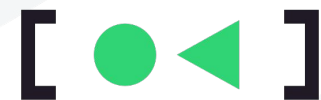
```
Volume=${ROOTFS}/etc:/etc
Volume=${ROOTFS}/var:/var
```



ELISA

Safety Applications

WORKSHOP



systemd



podman

[Service] configures systemd Cgroups & service handling

[Service]

AllowedCPUs=6-11

CPUWeight=50

Delegate=true

IOWeight=50

ManagedOOMSwap=kill

MemorySwapMax=0

Slice=QM.slice

Restart=always

OOMScoreAdjust=500

Environment=ROOTFS=/usr/lib/qm/rootfs

[● ◀]
systemd

Note

Names the Slice for all processes within the QM

ASIL Manager process can Modify all QM processes using the QM.slice

[Service] configures systemd Cgroups & service handling

[Service]

AllowedCPUs=6-11

CPUWeight=50

Delegate=true

IOWeight=50

ManagedOOMSwap=kill

MemorySwapMax=0

Slice=QM.slice

Restart=always

OOMScoreAdjust=500

Environment=ROOTFS=/usr/lib/qm/rootfs

Note

Configures cgroups only run QM on CPUs 6-11

ASIL apps can run on CPUS 0-11

Very system specific

[● ◀]
systemd



ELISA
Enabling Linux in
Safety Applications



WORKSHOP

License: CC-BY-4.0



[Service] configures systemd Cgroups & service handling

[Service]

AllowedCPUs=6-11

CPUWeight=50

Delegate=true

IOWeight=50

ManagedOOMSwap=kill

MemorySwapMax=0

Slice=QM.slice

Restart=always

OOMScoreAdjust=500

Environment=ROOTFS=/usr/lib/qm/rootfs

[● ◀]
systemd

Note

Default CPUWeight is 100.

QM Apps only get ½ CPU priority of ASIL Processes

[Service] configures systemd Cgroups & service handling

[Service]

AllowedCPUs=6-11

CPUWeight=50

Delegate=true

IOWeight=50

ManagedOOMSwap=kill

MemorySwapMax=0

Slice=QM.slice

Restart=always

OOMScoreAdjust=500

Environment=ROOTFS=/usr/lib/qm/rootfs

[● ◀]
systemd

Note

Default IOWeight is 100.

QM Apps only get ½ IO priority of ASIL Processes

[Service] configures systemd Cgroups & service handling

[Service]

AllowedCPUs=6-11

CPUWeight=50

Delegate=true

IOWeight=50

ManagedOOMSwap=kill

MemorySwapMax=0

Slice=QM.slice

Restart=always

OOMScoreAdjust=500

Environment=ROOTFS=/usr/lib/qm/rootfs

Note

1. Kernel under Memory pressure kill QM before ASIL
2. Default OOM Score is 0
3. Possible OOM Scores -1000 -> 1000
4. Containers kill prioritized over QM process.

[● ◀]

systemd

Hunger Games
Katniss Everdeen Options



[Service] configures systemd Cgroups & service handling

```
[Service]
AllowedCPUs=6-11
CPUWeight=50
Delegate=true
IOWeight=50
ManagedOOMSwap=kill
MemorySwapMax=0
Slice=QM.slice
Restart=always
OOMScoreAdjust=500
Environment=ROOTFS=/usr/lib/qm/rootfs
```

Note

Set the environment variable ROOTFS to be used in the container section of the quadlet file.

[● ◀]
systemd

How complex it can be to an OS?



Car Manufacturer's End to end vehicle software platform

Platform running on: Automobile Operational System based on Linux



composefs

crun

qm

bluechi

glibc

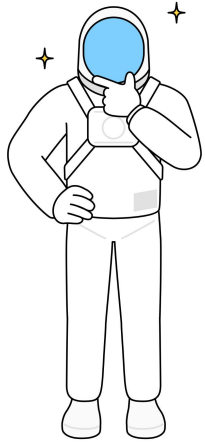
kernel

cgroups

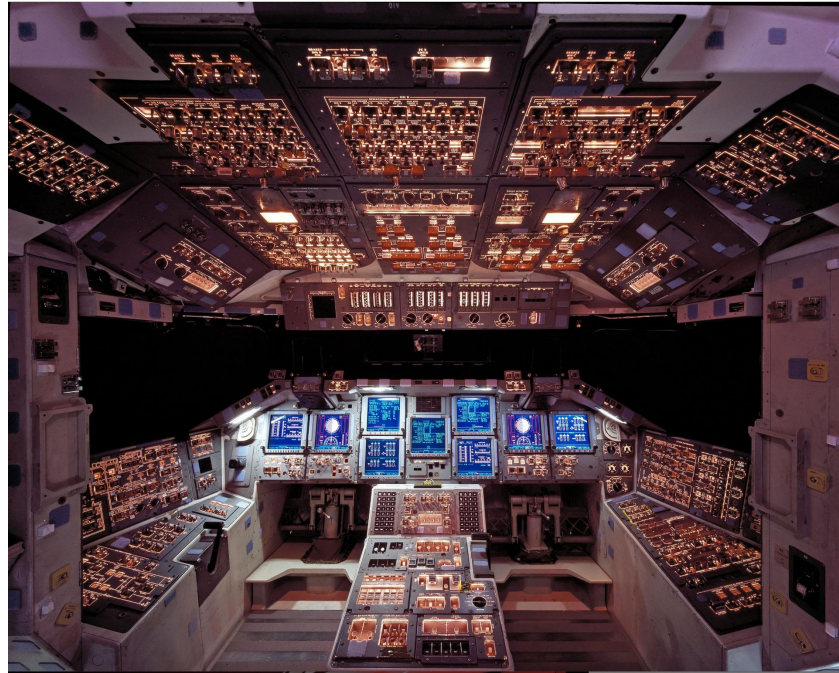


RedHat

But... How complex it can be to create a Space Grade Linux?



glibc
composefs
cgroups
[● ◀]
systemd
bootc



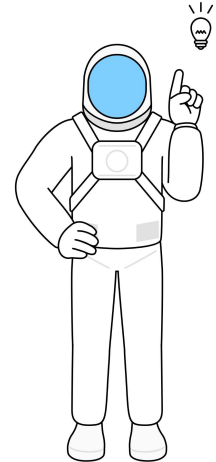
Space Shuttle Columbia Cockpit. Credit: NASA

kernel



bluechi

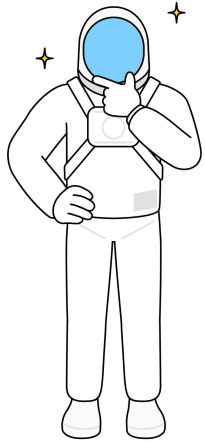
qem



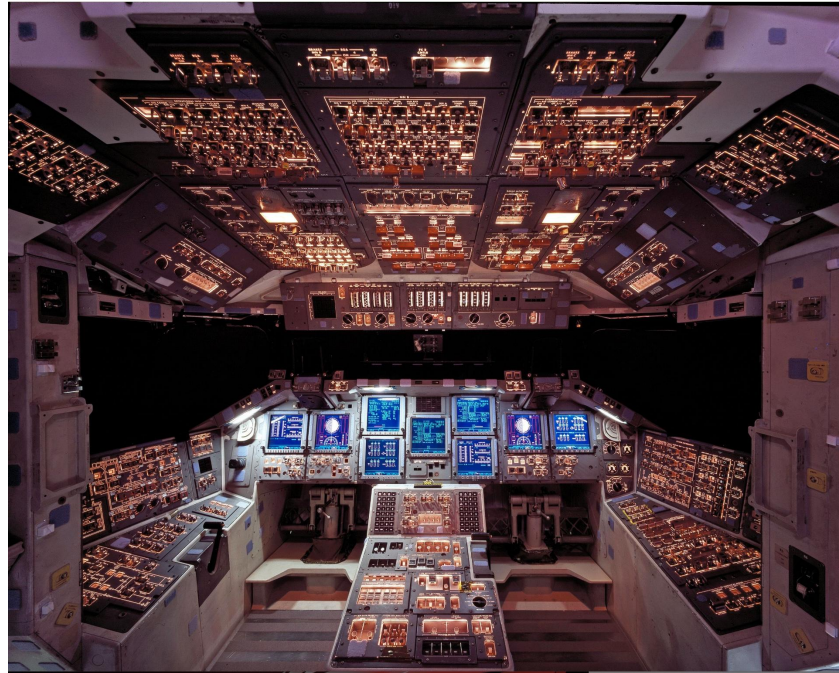
crun



But... How complex it can be to create a Space Grade Linux?



glibc
composefs
cgroups
[● ◀]
systemd
bootc



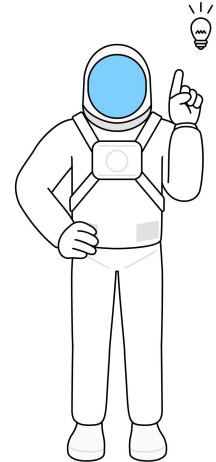
Space Shuttle Columbia Cockpit. Credit: NASA

kernel



bluechi

qem



Due time limit, let's have a group discussion after the talk :)



crun



Is Linux tested to satisfy and mitigate risk analysis for automotive?



podman



STRESS

Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

Let's image the hacker is smart enough and is able to break the initial security layers and it's ALSO able to connect to a nested container **as root**....





Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

**Next step: Deploy the crypto miner and steal all CPUs priority to mine while the car is in charge mode and send it to his/her digital wallet.
(from 9PM until 5AM - owner is sleeping)**



```
# subZer0> ./make-me-rich
10:24:45 - reading the system .....
10:24:46 - Setting make-me-rich as daemon and hiding files ...
10:24:47 - collecting current OS scheduler .....
10:24:48 - waiting car be in charge mode ....
.....
21:55:51 - Car is now connected to be charge ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy to make-me-rich...
FAILED, unable to access Operational System system call
```



Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

Lets understand what just happened....

Nested Container

```
# subZer0> ./make-me-rich
10:24:45 - reading the system .....
10:24:46 - Setting make-me-rich as daemon and hiding files
....
10:24:47 - collecting current OS scheduler .....
10:24:48 - waiting car be in charge mode ....
.....
21:55:51 - Car is now connected to be charge ...
21:55:52 - +++ make-me-rich mode starting +++
21:56:53 - +++ reading the current scheduler policy +++
21:56:54 - +++ Setting priority scheduler policy...
FAILED, unable to access Operational System system call
```

ASIL host (side)

```
# journalctl -r
```

```
<SNIP>
```

```
SELinux is preventing make-me-rich
from map access on the file
/usr/lib64/ld-linux-x86-64.so.2.
```

```
... avc: denied { map } <----- HERE
```

```
....avc: denied { read } <----- HERE
```





Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

But guess what, let's keep with our imagination....

For some reason, the trainee Disabled SELinux in that car model for tests and all car models got updated from the cloud image... OH NO! :-/



Let's simulate this situation setting the the car OS to permissive mode

```
[root@RHIVOS-carOS ~]# setenforce 0
```



Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"



```
# subZero> ./make-me-rich
```

```
21:55:51 - Car is now connected to charged ...
```

```
21:55:52 - +++ make-me-rich mode starting +++
```

```
21:56:53 - +++ reading the current scheduler policy +++
```

```
21:56:54 - +++ Setting priority scheduler policy to
```

```
make-me-rich: steal_cycles_sched_deadline failed to  
boost pid 0: Operation not permitted
```



Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"



```
# subZer0> ./make-me-rich
```

```
21:55:51 - Car is now connected to charged ...
```

```
21:55:52 - +++ make-me-rich mode starting +++
```

```
21:56:53 - +++ reading the current scheduler policy +++
```

```
21:56:54 - +++ Setting priority scheduler policy to
```

```
make-me-rich: steal_cycles_sched_deadline failed to  
boost pid 0: Operation not permitted
```

BUT WHO SAVED THE DAY?



Sample Risk Analysis according to ISO 26262:

SELinux + QM Blocking nested containers attack from "stealing CPU priority"

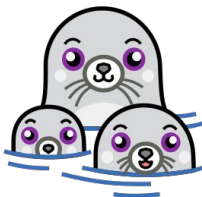


ME ?



Sample Risk Analysis according to ISO 26262: *SELinux + QM Blocking nested containers attack from "stealing CPU priority"*

"Several layers of security..."



+ **SECComp**

podman

Seccomp is a Linux kernel feature that provides a way to filter and limit the system calls available to a process. By using seccomp, Podman enhances the security of containers by minimizing the attack surface and reducing the risk of malicious activities.

Now let's IMAGINE such distro in SPACE - What's required?

- Rocket launch Schema



Saturn V by NASA

Now let's IMAGINE such distro in SPACE - What's required?

```
class RocketLaunch(BaseNode):
```

```
    """
```

```
    def __init__(self, rocket_name, payload, mission_type="standard"):
```

```
        """
```

Initialize a RocketLaunch instance and a ROS2 Node.

Parameters:

- rocket_name (str): Name of the rocket.
- payload (str): Description of the payload.
- mission_type (str): Type of mission (e.g., 'curiosity', or 'standard').

```
        """
```

```
        super().__init__('rocket_launch_node')
```

```
        self.rocket_name = rocket_name
```

```
        self.payload = payload
```

```
        self.mission_type = mission_type.lower()
```

Now let's IMAGINE such distro in SPACE - What's required?

[ros2-rocket-demo](#) / [src](#) / [rocket_launch_simulator](#) / 









Add file ▾



dougsland rocket_launch

f82bbbb · last week

 History

Name	Last commit message	Last commit date
 ..		
 <code>__init__.py</code>	rocket_launch	last week
 <code>blue-origin.py</code>	rocket_launch	last week
 <code>boeing.py</code>	rocket_launch	last week
 <code>lockheed_martin.py</code>	rocket_launch	last week
 <code>nasa.py</code>	rocket_launch	last week
 <code>setup.py</code>	rocket_launch	last week
 <code>spacex.py</code>	rocket_launch	last week



ELISA
Enabling Linux in
Safety Applications

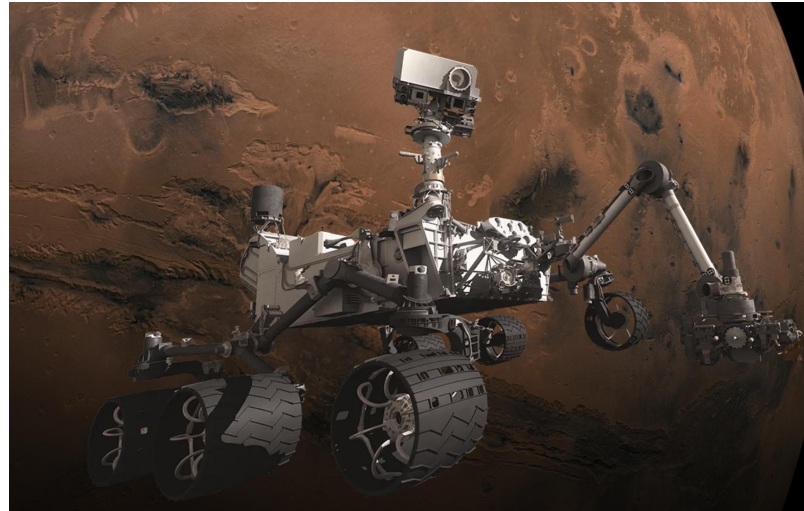


WORKSHOP

License: CC-BY-4.0

Now let's IMAGINE such distro in SPACE - What's required?

- Rocket launch Schema
- Rover



Curiosity Rover by NASA

Now let's IMAGINE such distro in SPACE - What's required?

```
class BaseUGVController:
```

```
    """
```

```
    This code is all about talking to a robot car (called a UGV, or Unmanned  
    Ground Vehicle).
```

```
    It supports basic wheel movement, independent wheel control (if the robot  
    supports it), ROS-based velocity control,  
    motor PID configuration, and now movement commands for turning,  
    moving backward, and forward.
```

```
    """
```

```
    def __init__(self, ssid="UGV", password="12345678", ip="192.168.4.1",  
    interface_name=None):
```

```
        """
```

```
        Initialize the UGV controller.
```

Now let's IMAGINE such distro in SPACE - What's required?

<SNIP>

```
def move_right(self, speed=0.5):
```

```
def move_left(self, speed=0.5):
```

```
def move_backwards(self, speed=0.5):
```

```
def cmd_ros_control(self, linear_velocity, angular_velocity):
```

```
    """
```

Control the robot using ROS-style velocity commands.

Args:

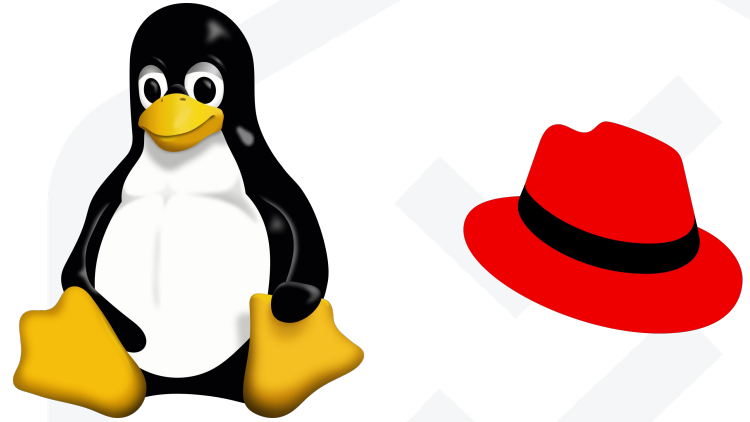
linear_velocity (float): The moving linear velocity in m/s (meters per second).

angular_velocity (float): The steering angular velocity in rad/s (radians per second).

<SNIP>

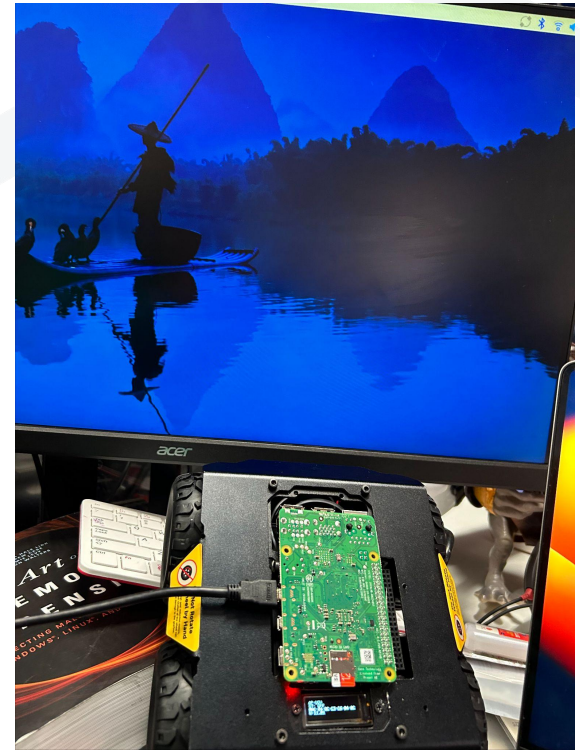
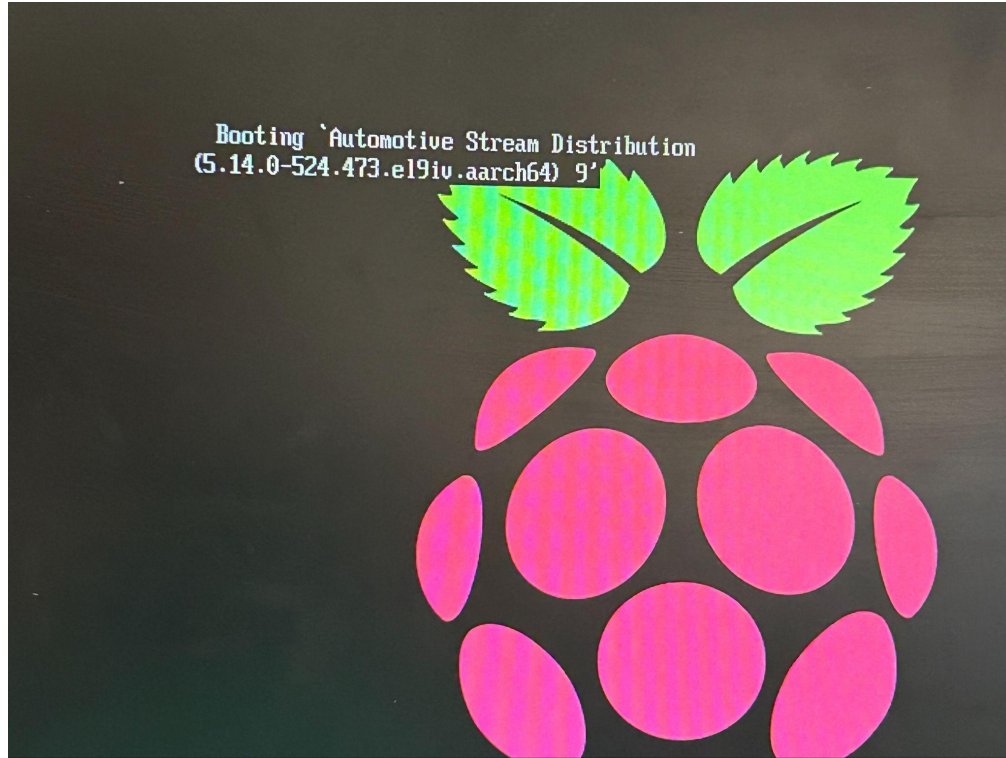
Now let's IMAGINE such distro in SPACE - What's required?

- Rocket launch Schema
- Rover
- Linux Distribution



Space Grade Linux Distribution

Example of similar industry: CentOS Auto/Fedora/RHIVOS:



Wow!



Initial Code already available

```
git clone https://gitlab.com/fedora/sigs/robotics/src/ros2-rover-demo
```

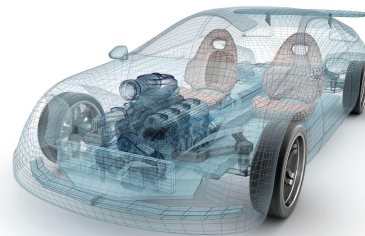
BUT Wait. As we are speaking about “Space Grade Linux Spec...”

Should we also start thinking about constructing a minimum API Abstract for space related industry?

Should we use this code demo here as initial base?

Due time limit, let's have a group discussion after the talk :)

Rocket Launcher Schema (ROS2 version)



```
git clone https://gitlab.com/fedora/sigs/robotics/src/ros2-rover-demo
```

```
cd ros2-rocket-demo/
```

```
podman build -f Containerfile.autosd -t localhost/ros2-rocket:latest .
```

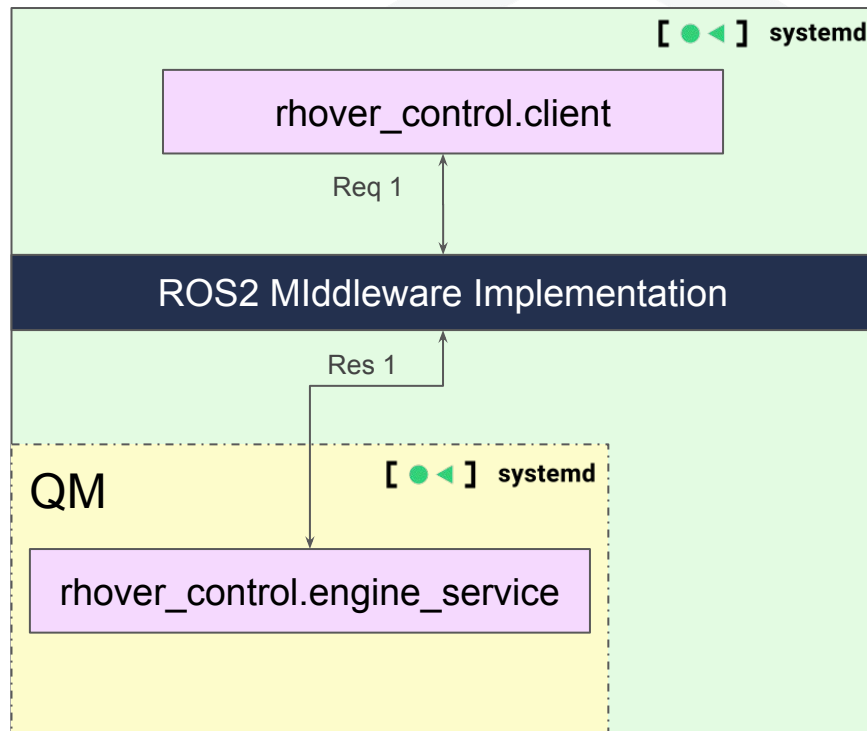
```
podman run --privileged --network -d --rm --name rhover localhost/ros2-rocket:latest
```

```
podman exec -it ros2-rocket /bin/bash
```

```
systemctl status rhover-pull-image
```

RHover: ROS2 + Containers + QM

- Control node (critical) runs in QM
- Requests/Messages sent through the ROS2 MW
- ROS2 inside Linux Containers
- <https://github.com/autosd-vss-mw/ros2-rocket-demo>



Talk is cheap, show me the code/demo... (Linus Torvalds)



SPACE DEMO



The Fedora Robotics SIG

- Goal: Enable Fedora as a robotics development environment
 - Enable robotics frameworks in containers such as ROS2
 - Development environment using containers and toolbox
- Leverage edge container technologies such as AutoSD
- What we have:
 - A base ROS2 fedora image built from source
 - A CentOS Stream 9 image that uses ROS' RHEL repositories
- Where:
 - <https://gitlab.com/fedora/sigs/robotics>
 - <https://docs.fedoraproject.org/en-US/robotics-sig/>
- Help wanted!

Looking for more info? *Contact the SIG Leader: Leonardo Rossetti*

Thank you NASA and ELISA/Linux Foundation

To the NASA folks:

"Please keep leading the path to space and make us dream of the impossible, making it possible."



Licensing of Workshop Results

All work created during the workshop is licensed under Creative Commons Attribution 4.0 International (CC-BY-4.0) [<https://creativecommons.org/licenses/by/4.0/>] by default, or under another suitable open-source license, e.g., GPL-2.0 for kernel code contributions.

You are free to:

- Share — copy and redistribute the material in any medium or format
- Adapt — remix, transform, and build upon the material for any purpose, even commercially.

The licensor cannot revoke these freedoms as long as you follow the license terms.

Under the following terms:

Attribution — You must give appropriate credit, provide a link to the license, and indicate if changes were made. You may do so in any reasonable manner, but not in any way that suggests the licensor endorses you or your use.

No additional restrictions — You may not apply legal terms or technological measures that legally restrict others from doing anything the license permits.

Behind the Scenes: What could go possibility wrong?

