



Can MicroOS Desktop Be Your "Daily Driver" ?

(SPOILER ALERT: Probably YES!)

Dario Faggioli, dfaggioli@suse.com



Can MicroOS Desktop Be Your “Daily Driver” ?

Proof that it ~~can~~ is:

```
dario@Wayrath:~> cat /etc/os-release
NAME="openSUSE MicroOS"
# VERSION="20201009"
ID="opensuse-microos"
ID_LIKE="suse opensuse opensuse-tumbleweed"
VERSION_ID="20201009"
PRETTY_NAME="openSUSE MicroOS"
ANSI_COLOR="0;32"
CPE_NAME="cpe:/o:opensuse:microos:20201009"
BUG_REPORT_URL="https://bugs.opensuse.org"
HOME_URL="https://www.opensuse.org/"
DOCUMENTATION_URL="https://en.opensuse.org/Portal:MicroOS"
LOGO="distributor-logo"
```

- The first part of this talk is also covered [by this blog post](#)

About Me What I do

- Virtualization Specialist Sw. Eng. @ SUSE since 2018, working on Xen, KVM, QEMU, mostly about performance related stuff
- Daily activities ⇒ how and what for I use my workstation
 - Read and send emails (Evolution, git-send-email, stg mail, ...)
 - Write, build & test code (Xen, KVM, Libvirt, QEMU)
 - Work with the Open Build Service (OBS)
 - Browse Web
 - Meetings / Video calls / Online conferences
 - Chat, work and personal
 - Occasionally play games
 - Occasional video-editing
 - Maybe scan / print some document
- Can all of the above be done with MicroOS already ?

What is MicroOS

- Immutable single purpose OS, based on Tumbleweed, born as container host but not limited to that use case
 - <https://microos.opensuse.org/>
 - <https://en.opensuse.org/Portal:MicroOS>
 - Richard's and Ish's talks!



<https://youtu.be/nIwgzGbX-oc>



<https://youtu.be/8gGjcKdOWIc>

What is MicroOS as a Desktop

- MicroOS ⇒ Single purpose OS
- Each install does only one thing:
 - One thing == Hosting containers
 - One thing == Hosting VMs
 - One thing == Set Top Box
 - One thing == Your Desktop
 - More talks from Richard
 - The latest one, yesterday!



<https://youtu.be/7p4y9Meyy0M>

openSUSE Conference 2019

openSUSE MicroOS Desktop

A New openSUSE Desktop Distribution?

Richard Brown

<https://youtu.be/ASSkQH9kNao>

MicroOS Desktop - The Road to Daily Driving

Schedule

Presented by:

Richard Brown

No video of the event yet, sorry!

Richard is from England but currently lives in Nuremberg in Germany, and is employed as openSUSE Chairman and as a Linux Distribution Engineer in the Future Technology Team at SUSE. Involved in openSUSE/SUSE since 2003, Richard has contributed to various aspects of the project, including supporting users on IRC, testing/bug reporting, packaging, marketing, ambassadors and artwork. In addition to being Chairman, Richard is still involved as a maintainer of GNOME and the openSUSE branding packages, and working on openQA

The desktop operating systems we've been using for the last decades require far too much effort to maintain. What we need is a simple desktop that can be given to anyone, that always updates itself, that always repairs itself, and while the base system might always be up to date, users should be able to get to pick whatever apps they want to on top of it.

This is the basic premise of the MicroOS Desktop which came to life at oSC 2019 and is now in an Alpha state offered as part of the MicroOS installation ISO for both Intel and ARM architectures.

This session will demonstrate the current state of the MicroOS Desktop, discuss future plans and invite more contributors and users to get involved with this new generation of desktop linux.

Date: 2020 October 16 - 14:30
Duration: 30 min
Room: Room 1
Conference: openSUSE + LibreOffice Virtual Conference (Time in UTC)
Language:
Track: **openSUSE**
Difficulty: **Easy**

Happening at the same time:

1. Google Summer of Code 2020 Panel

<https://events.opensuse.org/conferences/oSLO/program/proposals/3322>

How I Got Involved

- SUSE Hack Week 19 (which happened in 2020)
 - Chance for SUSE employees to work on do whatever they find cool
- MicroOS as a Desktop
 - Immutable, taking advantage of BTRFS
 - Base OS from distro, apps from other (proper?) sources
 - Rolling base, as based on Tumbleweed
 - Rolling but reliable... as based on Tumbleweed
- I found it cool! :-)
 - Tried and tested it
 - Started hacking on toolbox (see later)

The screenshot shows a project page titled "Hack Week 19" with a green lizard icon. The main title is "MicroOS Desktop". It's described as a project by RBrownSUSE, updated 7 days ago, with 12 hackers and 13 followers. A video recording from an openSUSE Conference session is linked. The "Core Project" section discusses MicroOS as a Kubernetes-focused distribution that adds solutions for updating running systems and becoming a base for containers. It notes that while servers enjoy automatic updates, desktop users want to be lazy like server admins. The page asks if tools like MicroOS can help create the desktop distribution of the future. A "Let's find out!" section is present, along with a note about building, testing, and introducing the world to the openSUSE MicroOS Desktop, a desktop variant of MicroOS based on Tumbleweed. On the right, there's a grid of developer portraits and a "Leave this project" button. Below the grid, a "Looking for hackers with the skills:" section lists various technologies: opensuse x, tumbleweed x, gnome x, flatpak x, kiwi x, obs x, and distribution x.

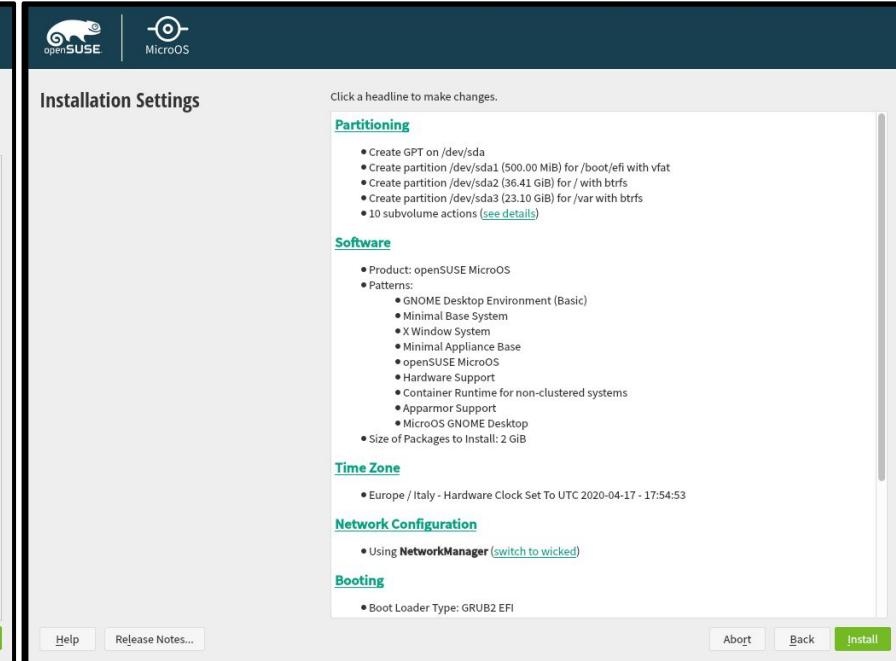
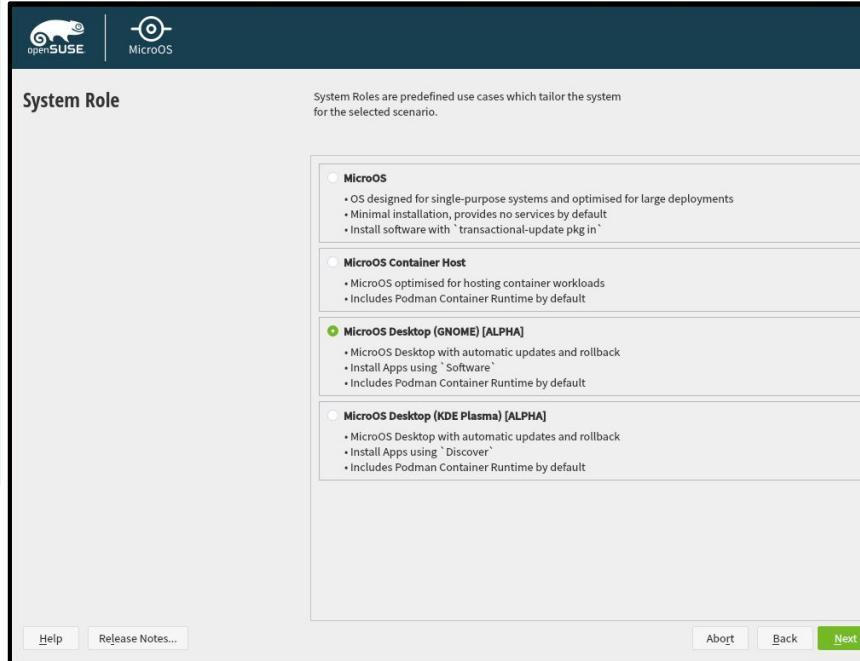
(<https://hackweek.suse.com/projects/microos-desktop>)

Why I Tried and Why I'm Liking it

- A relatively small and immutable base OS
 - Stable and reliable
 - Immutable ⇒ much more difficult to mess-up
- Issues with package dependencies:
<<Oh, no!! X can't be installed/upgraded because libY needed by Z is too old>>
 - Fewer packages ⇒ a lot less likely to happen (in fact, never happened in months...)
- BTRFS at its finest:
 - Updates in non-running snapshots. Automatic rollback with [health-check](#)
- Apps from Flatpak/Flathub
 - Contributed to Flathub directly from upstream app developers
 - ⇒ Effort done once, multiple (all?) distro can profit from that
 - ⇒ Distro/OS developers can focus on OS, app developers can focus on apps
- Tumbleweed is rock solid, thanks to OpenQA, etc
 - As soon as you add an additional repository, this may change ...
 - Technically you're not using the distro that has been developed & tested any longer
 - (In practice, fine, especially for Packman, etc. But, still.)
 - Here you don't need **any** additional repository !

Installing

- Just grab it: <https://microos.opensuse.org/>, and install it!
- Choose "MicroOS Desktop [GNOME] [ALPHA]"
- Choose "KDE Plasma" if you want, but I've never tested it. No idea if/how it works!



Immediately After Installing

- Add FlatHub as flatpak remote
 - \$ flatpak remote-add --user flathub <https://flathub.org/repo/flathub.flatpakrepo>
- Some GNOME Software (black) magic:
 - \$ gsettings set org.gnome.software install-bundles-system-wide false
\$ gsettings set org.gnome.software allow-updates false
\$ gsettings set org.gnome.software download-updates false
\$ gsettings set org.gnome.software enable-repos-dialog false
\$ gsettings set org.gnome.software first-run true
- Some zypper (black) magic:
 - \$ sudo rm -Rf /var/cache/app-info
\$ sudo transactional-update shell
rpm -e --nodeps libzypp-plugin-appdata
zypper al libzypp-plugin-appdata
exit
\$ sudo reboot
- Shouldn't this should all be done automatically?
 - Indeed ! Patches / SRs welcome :-P

Some More Customization

- For toolbox (see later) ↪

- ```
echo "<yourusername>:100000:65536" > /etc/subuid
echo "<yourusername>:100000:65536" > /etc/subgid
```

- I want passwordless sudo

- ```
# usermod -a -G wheel <yourusername>
# echo "%wheel    ALL = (root) NOPASSWD:ALL" > /etc/sudoers.d/wheel
```

- I want to disable automatic updating and rebooting

- I will deal with updating (and rebooting) manually

- ```
$ sudo systemctl disable --now rebootmgr.service
Removed /etc/systemd/system/multi-user.target.wants/rebootmgr.service.
```

- Let's check:

- ```
$ sudo rebootmgrctl is-active
RebootMgr is dead
$ sudo rebootmgrctl status
Error: The name org.opensuse.RebootMgr was not provided by any .service
files
```

Should be done automatically too, IMO.
Again, contributions welcome!

Additional Repositories & Packages

- Add repositories, e.g. Packman:
 - https://en.opensuse.org/Additional_package_repositories
 - All of Packman:
 - zypper ar -cfp 90 http://ftp.gwdg.de/pub/linux/misc/packman/suse/openSUSE_Tumbleweed/_packman
 - Install codecs
- Add <more repositories>
- Install <a lot of packages for whatever I need>

Right?

NO, GOD! NO, GOD, PLEASE NO! NO! NO!

NOOOOOOOOOOOOO!

Installing Packages

- No zypper (well, it's there but it's locked ⇒ try it, it won't work!)
- Transactional-update , directly:
 - \$ sudo transactional update pkg install wget unzip
\$ sudo reboot
- transactional-update , via shell:
 - \$ sudo transactional-update shell
zypper ref
zypper in wget unzip
exit
\$ sudo reboot
- Multiple sessions:
 - \$ sudo transactional -update pkg install wget
[. . .]
\$ sudo transactional-update shell --continue
zypper in unzip
exit
\$ sudo reboot
- Reboot always necessary, for seeing and using new packages

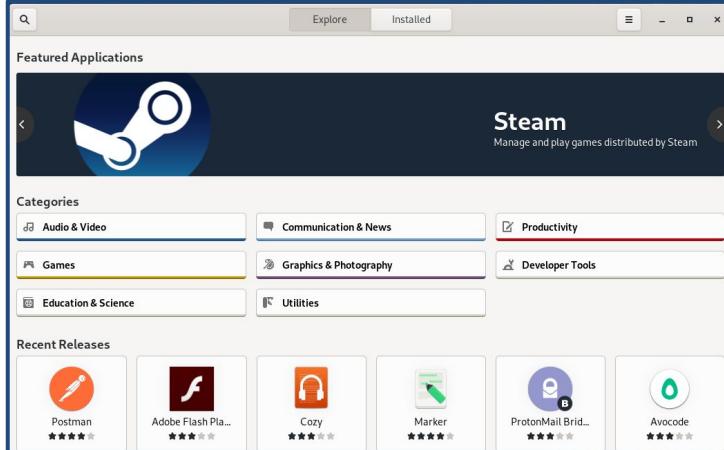
Are We Constantly Rebooting ?

- Nah!
 - For instance, I haven't rebooted this workstation since 3 days and 16 hours (and counting!)

```
dario@Wayrath:~> uptime
08:51:42  up 3 days 16:33,
dario@Wayrath:~> █
```
- How so?
 - For apps:
 - Flatpak (from Flathub, <https://flathub.org/>)
 - For troubleshooting / debugging:
 - toolbox
 - For development
 - toolbox
 - For "development apps":
 - toolbox
- Installing/removing activities RPMs on the base OS tends to zero

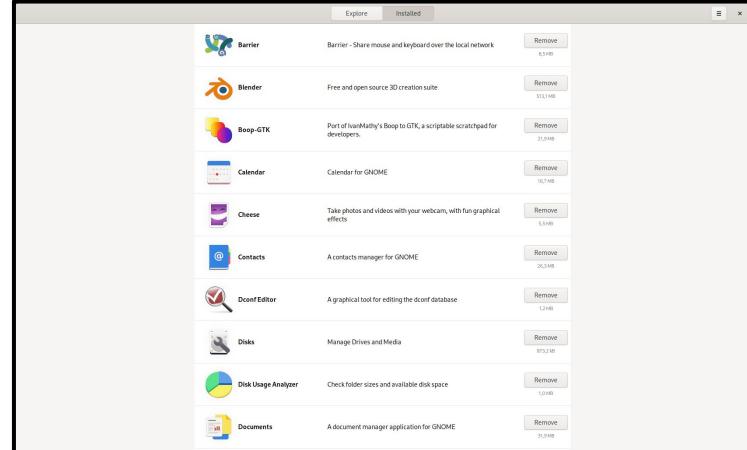
Flatpak

- It will be our main install source, for all applications
- Via GNOME Software
 - Once configured as shown
- Via cli
 - flatpak install org.gnome.gedit
alias gedit='flatpak run org.gnome.gedit'



```
dario@Wayrath:~> flatpak ps
```

Instance	PID	Application	Runtime
2416651609	17472	org.vim.Vim	org.freedesktop.Platform
4132321407	17296	org.libreoffice.LibreOffice	org.freedesktop.Platform
1453829993	16762	org.mozilla.firefox	org.freedesktop.Platform
588396198	16391	chat.rocket.RocketChat	org.freedesktop.Platform
729209139	16107	org.gnome.Evolution	org.gnome.Platform
2888792739	3792	org.gnome.gedit	org.gnome.Platform
4074027177	2547	org.telegram.desktop	org.kde.Platform
1538198007	2541	com.github.debauchee.barrier	org.kde.Platform
3656091025	2542	im.pidgin.Pidgin	org.gnome.Platform
3380805255	2545	me.kozec.syncingtk	org.gnome.Platform



Toolbox

- A shell script that launches a privileged container
 - Check: <https://kubic.opensuse.org/blog/2019-10-22-toolbox/>
 - Most other immutable OSes has something similar (e.g., [Silverblue](#))
 - The host file system will be visible/accessible while inside the container (bind mounts, etc)
- The container can run:
 - As root
 - You may or may not have your regular user in the toolbox container
 - When you are root in the toolbox container run as root, you're kind of root on the host
 - As your regular user
 - Thanks to "[rootless podman](#)"
 - You have your regular user in the toolbox container
 - Even when you are root in the toolbox container, you are not root on the host
- BEWARE: "privileged container" & "can run as root"
 - It's **not** a security enhancing tool
 - I.e.: <<I can do whatever I want, I'm in a container, I won't affect or disrupt the base OS, right?>>
 - No, this is not the right mindset
 - You're not less secure or safe than when you're working directly on the base OS
 - You're not more secure or safe either!

Different Kind of Toolbox-es

- Creating and entering a toolbox that runs as your user, and be your own user while inside it:
 - Useful for using toolbox as your user / developer environment
 - \$ toolbox -u # -u ⇒ you will have your user, your /home, etc
 > # you're inside the toolbox already!
 - \$ toolbox -u -t foo # -t ⇒ to give this toolbox a name ('t' for 'tag')
 > # you're now inside the toolbox tagged 'foo'
 > sudo su # you're becoming root in container. But, e.g., you still
 #> # won't be able to touch files owned by root on the host!
- Creating and entering a toolbox that runs as your user, but has only root user inside it:
 - Useful for using toolbox as a debugging and troubleshooting environment
 - \$ toolbox # no -u ⇒ no user except root, nothing in /home
 #> # you are root already. But root in toolbox
 #> # **does not map** on root on the host

Different Kind of Toolbox-es

- Creating and entering a toolbox that run as root, and be your own user while inside it:
 - Useful for using toolbox as your user / developer environment (that needs "special powers")
 - \$ toolbox -r -u # -u ⇒ you will have your user, your /home, etc
 > # -r ⇒ the toolbox run as root on the host
 - \$ toolbox -r -u -t foo # -t ⇒ to give this toolbox a name ('t' for 'tag')
 > # you're now inside the toolbox tagged 'foo'
 > sudo su # you're becoming root in container and that maps with
 #> # root on the host (you'll be able to touch files owned
 #> # by root on the host, etc)
- Creating and entering a toolbox that runs as root, and has only root user root inside it:
 - Useful for using toolbox as a debugging/troubleshooting environment (with "special powers")
 - \$ toolbox -r # -r ⇒ the toolbox run as root on the host
 #> # no -u ⇒ no user except root, nothing in /home. Also,
 #> # you are root already, and that **does map** with root on the host

Managing Your Toolbox-es

- Toolbox is **stateful**:
 - Yesterday you created a toolbox, and you install stuff, change configs, etc
 - Today you stop the toolbox, you turn off the PC and take the day off
 - Tomorrow toolbox will still have all the software and all the config changes you made
- Listing toolbox-es running as user:

- \$ podman ps

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
5cb19adef1fb1	[...]toolbox:latest	sleep +Inf	3 weeks ago	Up 3 hours ago	toolbox-dario-user

- Listing all toolbox-es created as user (running or not):

- \$ posman ps --all

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	NAMES
5cb19adef1fb1	[...]toolbox:latest	sleep +Inf	3 weeks ago	Up 3 hours	toolbox-dario-user
502722d98390	[...]toolbox:latest	sleep +Inf	3 weeks ago	Exited	toolbox-dario-user-dev

- For toolbox-es created as root:

- \$ sudo podman ps # list the running ones
 - Sudo podman ps --all # list all of them

- Removing toolbox-es:

- \$ podman rm <toolbox_name/ID> # for a toolbox running as user
 - \$ sudo podman rm <toolbox_name/ID> # for a toolbox running as root

Toolbox For Troubleshooting

Toolbox is super handy for debugging and troubleshooting

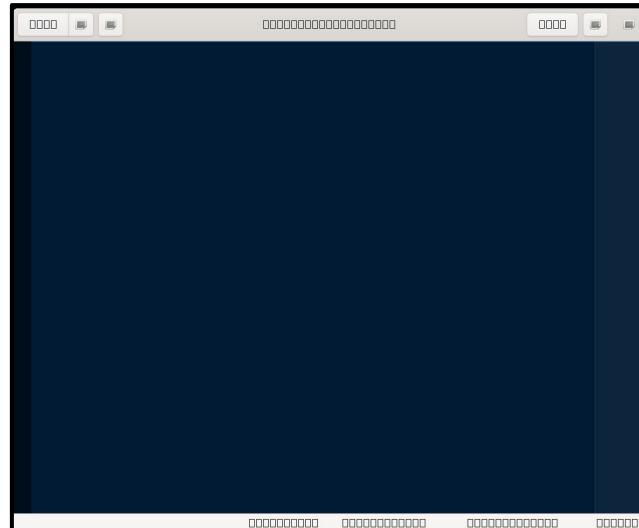
- Example: you need to do a strace ls
 - You can try... but strace is not installed!
 - Install it with transactional-update pkg inand then reboot ?!?
 - No!
 - \$ toolbox # runs as your user on the host (no -r)
#> zypper in strace # you're root in toolbox, but that
does not map to root on the host
#> strace ls # here you go your strace!
- Example, you need to nmap some host
 - Again, nmap is not there, and you don't want to reboot!
 - Nmap needs "real root", to scan low ports
 - \$ toolbox -r # runs as root on the host (-r)
#> zypper install nmap # we can add packages, no problem

Toolbox Config File

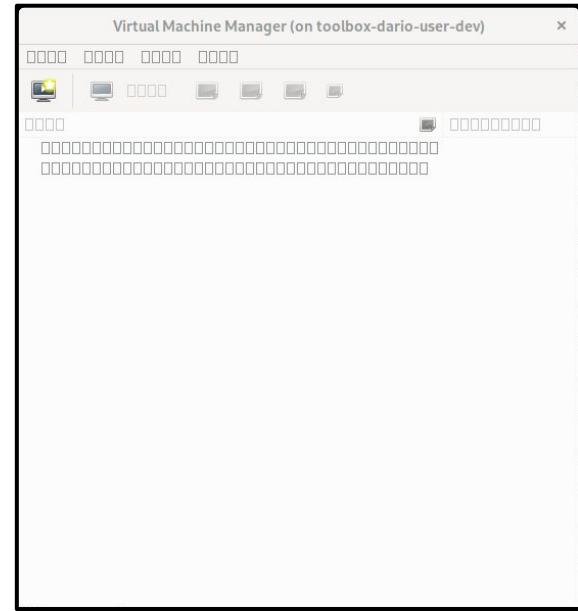
- Some tweaking possible (and more possibilities of tweaking being worked on ;-P)
- Config file:
 - ```
$ cat ~/.toolboxrc
REGISTRY=registry.opensuse.org
IMAGE=openSUSE/toolbox:latest
TOOLBOX_NAME=special-debug-container
TOOLBOX_SHELL="/bin/bash"
```
- 
- `TOOLBOX_NAME`: allows to tweak the basename of the toolbox-es
- `REGISTRY + IMAGE`: allows to use a different image for your toolbox-es
  - `toolbox/latest` is based on Tumbleweed
  - You can have Leap toolbox-es
  - You can make toolbox-es from your ([Kiwi](#) / [OBS](#) built) images
  - You can have toolbox-es based on different distros!
    - (possible already, but needs a little more work for dealing well with `-u`)

# Toolbox for Graphical Apps

- They work too! ⇒ No need installing them in base OS
- \$ `toolbox -u`
  - > `sudo zypper in gedit virt-manager`
  - > `gedit`
  - > `virt-manager`



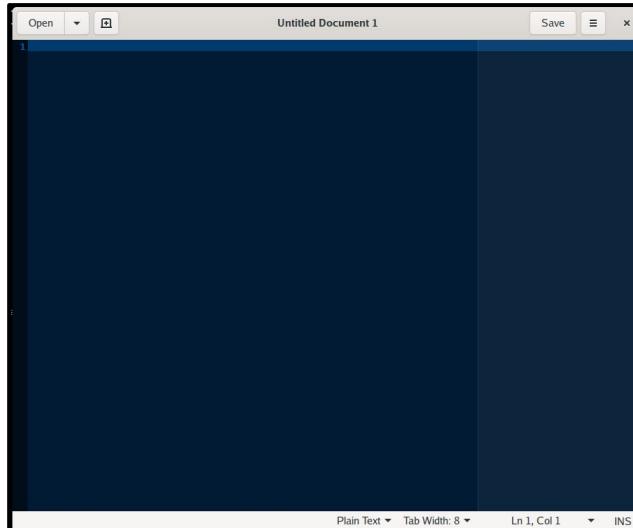
**Errr... What?**



# Toolbox for Graphical Apps

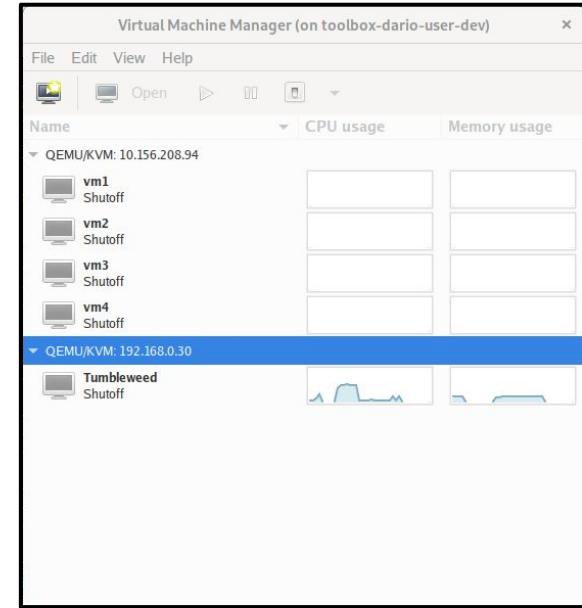
- They work too! ⇒ No need installing them in base OS
- \$ `toolbox -u`

```
> sudo zypper in gedit virt-manager
> sudo zypper in xorg-x11-fonts-core
> sudo zypper in adwaita-icon-theme
> gedit
> virt-manager
```



**Ok, now we're  
Talking**

(are we missing some deps  
somewhere, maybe?)



# Toolbox for “GL” Graphical Apps

- Kernelshark as an example:

- ```
$ toolbox -u
> kernelshark
libGL error: No matching fbConfigs or visuals found
libGL error: failed to load driver: swrast
QOpenGLWidget: Failed to create context
QOpenGLWidget: Failed to create context
qt.qpa.backingstore: composeAndFlush: QOpenGLContext creation failed
qt.qpa.backingstore: composeAndFlush: makeCurrent() failed
...
...
```

- What if...

- ```
$ toolbox
> sudo zypper addrepo https://download.nvidia.com/opensuse/tumbleweedNVIDIA
> sudo zypper ref
> sudo zypper in x11-video-nvidiaG05
```

- Install stuff like:

- kernel-default-devel, nvidia-gfxG05-kmp-default, nvidia-glG05 ...
  - ... Inside the container ?

# Toolbox for “GL” Graphical Apps

- Kernel Profiling
  - Virtualization
  - Instrumentation
- 
- The screenshot shows the Kernel Shark interface with the title "Kernel Shark (trace.dat) (on toolbox-dario-user-dev)". The main window displays two CPU timelines: CPU 0 and CPU 1. CPU 0 shows a dense sequence of magenta vertical bars representing kernel events. CPU 1 shows a single short magenta bar. Above the timelines, three time markers are shown: 377678.945123, 377678.945995, and 377678.946867. Below the timelines is a search bar and a table of kernel events:
- | #  | CPU | Time Stamp    | Task   | PID    | Latency | Event           | Info                                                   |
|----|-----|---------------|--------|--------|---------|-----------------|--------------------------------------------------------|
| 0  | 132 | 377678.945125 | <idle> | 0      | .N..    | funcgraph_entry | 2.258 us   sched_ttwu_pending();                       |
| 1  | 132 | 377678.945127 | <idle> | 0      | .N..    | funcgraph_exit  | 2.258 us   }                                           |
| 2  | 140 | 377678.945142 | <...>  | 104728 | d...    | funcgraph_entry | wake_up_process() {                                    |
| 3  | 140 | 377678.945143 | <...>  | 104728 | d...    | funcgraph_entry | try_to_wake_up() {                                     |
| 4  | 140 | 377678.945145 | <...>  | 104728 | d...    | sched_waking    | comm=kworker/u292:2 pid=104368 prio=120 target_cpu=054 |
| 5  | 140 | 377678.945146 | <...>  | 104728 | d...    | funcgraph_entry | 0.385 us   ttwu_queue_wakelist();                      |
| 6  | 140 | 377678.945147 | <...>  | 104728 | d...    | funcgraph_exit  | 0.385 us   }                                           |
| 7  | 140 | 377678.945147 | <...>  | 104728 | d...    | funcgraph_entry | ttwu_do_activate() {                                   |
| 8  | 140 | 377678.945149 | <...>  | 104728 | d...    | funcgraph_entry | 0.674 us   ttwu_do_wakeup();                           |
| 9  | 140 | 377678.945150 | <...>  | 104728 | d...    | funcgraph_exit  | 0.674 us   }                                           |
| 10 | 140 | 377678.945150 | <...>  | 104728 | d...    | funcgraph_exit  | 2.731 us   }                                           |
| 11 | 140 | 377678.945150 | <...>  | 104728 | d...    | funcgraph_exit  | 7.554 us   }                                           |
| 12 | 140 | 377678.945150 | <...>  | 104728 | d...    | funcgraph_exit  | 9.008 us   }                                           |

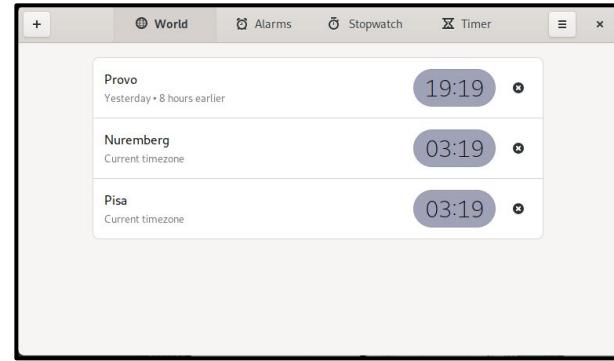
- ... Inside the container ? ⇒ Well, it works!

# Remember this?

- Virtualization Specialist Sw. Eng. @ SUSE since 2018, working on Xen, KVM, QEMU, mostly about performance related stuff
- Daily activities ⇒ how and what for I use my workstation
  - Read and send emails (Evolution, git-send-email, stg mail, ...)
  - Write, build & test code (Xen, KVM, Libvirt, QEMU)
  - Work with the Open Build Service (OBS)
  - Browse Web
  - Meetings / Video calls / Online conferences
  - Chat, work and personal
  - Occasionally play games
  - Occasional video-editing
  - Maybe scan / print some document
- Can all of the above be done with MicroOS already ?

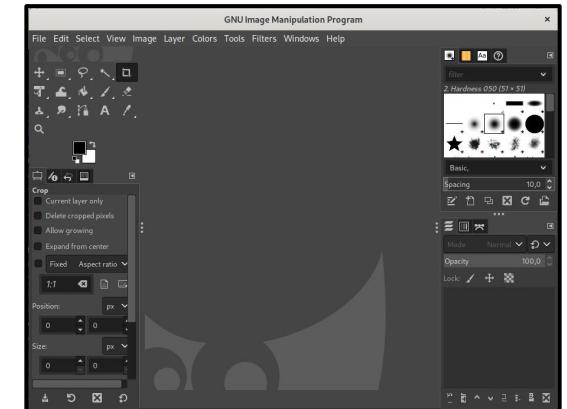
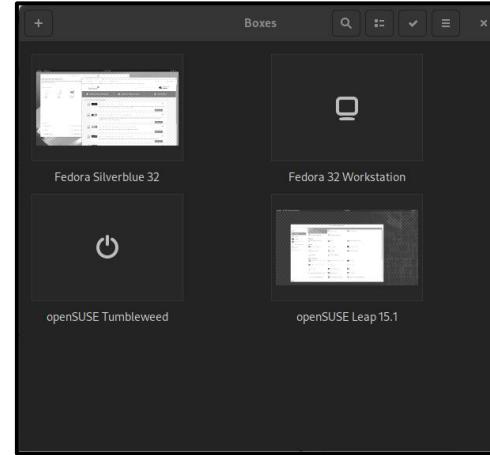
# Email, Calendaring, IM & Office Apps

- Mail, calendaring, contacts, ...
  - Evolution, [org.gnome.Evolution](#)
  - Calendar, [org.gnome.Calendar](#)
  - Contacts, [org.gnome.Contacts](#)
  - GNOME Clocks, [org.gnome.clocks](#)
  - Weather, [org.gnome.Weather](#)
- Documents
  - Evince, [org.gnome.Evince](#)
  - GNOME Documents, [org.gnome.Documents](#)
  - LibreOffice, [org.libreoffice.LibreOffice](#)
- Messaging
  - RocketChat, [chat.rocket.RocketChat](#)
  - Pidgin, [im.pidgin.Pidgin](#)
  - Telegram, [org.telegram.desktop](#)
  - Signal, [org.signal.Signal](#)



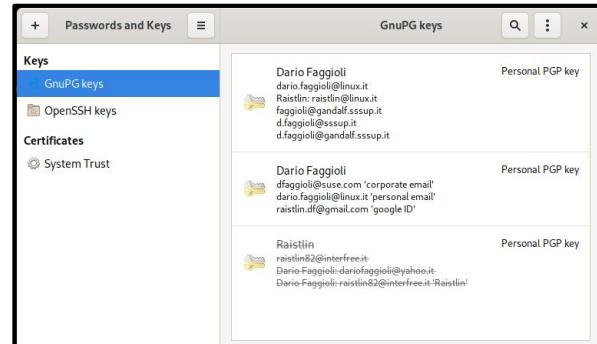
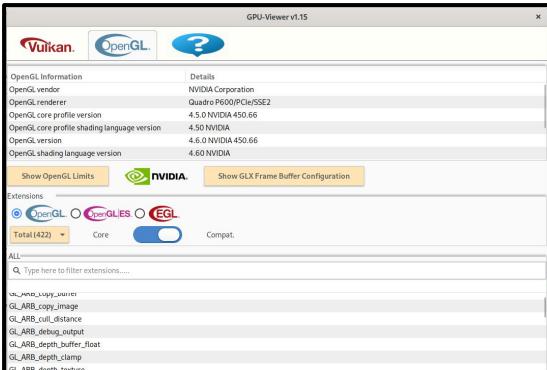
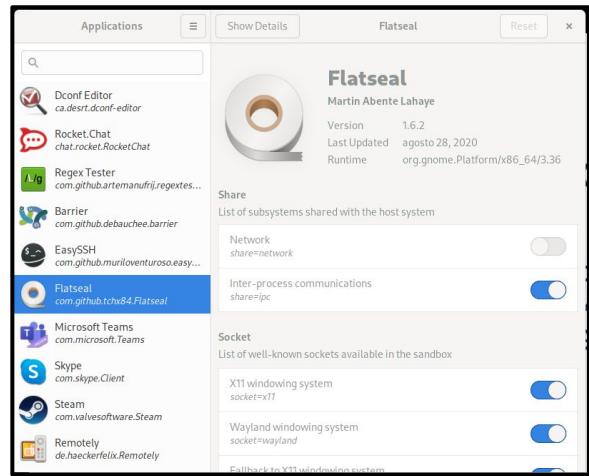
# Editors, Tools, Graphics

- Editors:
  - Vim, [org.vim.Vim](#)
  - Gedit, [org.gnome.gedit](#)
  - Setzer, [org.cvfosammm.Setzer](#)
  - Eclipse, [org.eclipse.Java](#)
- Graphics
  - GIMP, [org.gimp.GIMP](#)
  - Krita, [org.kde.krita](#)
  - Blender, [org.blender.Blender](#)
- VMs:
  - GNOME Boxes, [org.gnome.Boxes](#)
- Tools:
  - Regex Tester, [com.github.artemanufrij.regextester](#)
  - Meld, [org.gnome.meld](#)
  - Boop-GTK, [uk.co.mrbenshef.Boop-GTK](#)



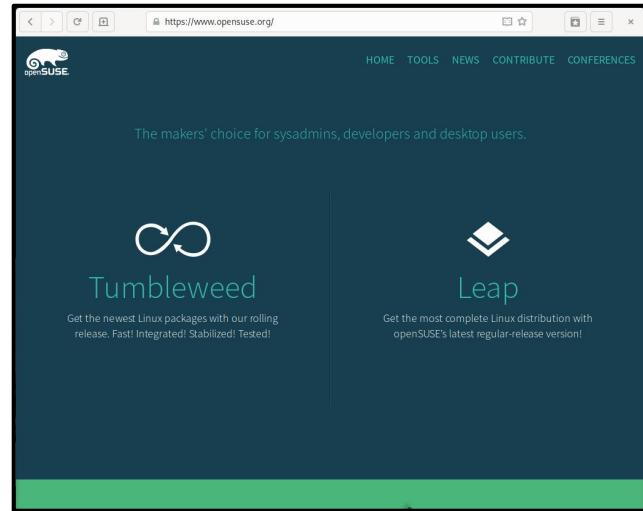
# Utilities, Configuration

- Misc utilities:
  - SyncThing, [me.kozec.syncthinggtk](#)
  - Barrier, [com.github.debauchee.barrier](#)
  - Seahorse, [org.gnome.seahorse.Application](#)
- Config:
  - Dconf Editor, [ca.desrt.dconf-editor](#)
  - Flatseal, [com.github.tchx84.Flatseal](#)
  - GPU-Viewer, [io.github.arunsivaramanneo.GPUViewer](#)



# Browsing

- Firefox, from the Flatpak ([org.mozilla.firefox](https://org.mozilla.firefox))
  - Works great, including video codecs  
(and without having to add Packman repos)
- Epiphany (GNOME Web, [org.gnome.Epiphany](https://org.gnome.Epiphany))
- Chrome[ium]
  - There is no Flatpak for that yes  
(but no, but [it's being worked on](#))
  - Installed in the base OS, with  
Transactional-update (and reboot)
- NB: GNOME Shell Extension can't be installed  
from a "Flatpak-ed" browser yet
  - You probably need at least one browser  
in the base OS (I have Chrome)



The screenshot shows the openSUSE MicroOS website at https://microos.opensuse.org/. The page has a white header with the openSUSE MicroOS logo and navigation links for Documentation, System Requirements, and Downloads. The main content area features a large green "MicroOS" logo and the text "openSUSE MicroOS". Below this, a paragraph explains the purpose of MicroOS. The page is divided into several sections: "Philosophy" (with sub-points like "Is predictable" and "Is scalable"), "Atomic Updates" (with sub-points like "Transactional Updates" and "Unique"), and "Flexible" (with sub-points like "No new package format is needed, use standard openSUSE RPMs" and "Easy to enhance").

# Gaming

- Steam, com.valvesoftware.Steam
  - Works great, even SteamPlay/Proton



- NVIDIA Drivers

- ```
# transactional-update shell
# zypper ar --refresh https://download.nvidia.com/opensuse/tumbleweed NVIDIA
# zypper in nvidia-g1G05 x11-video-nvidiaG05
# exit
# reboot
```
 - Brings in gcc and some development packages (not ideal... Thanks NVIDIA :-/)

- NB flatpak picked up automatically:

```
org.freedesktop.Platform.GL.nvidia-450-66
org.freedesktop.Platform.GL32.nvidia-450-66
```



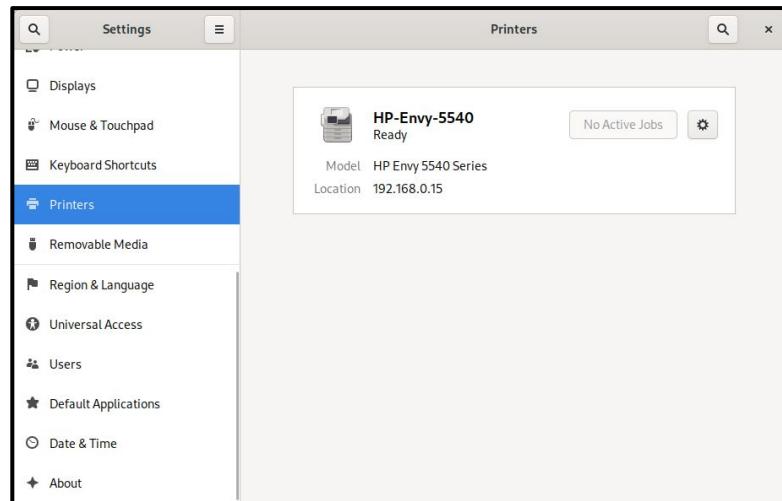
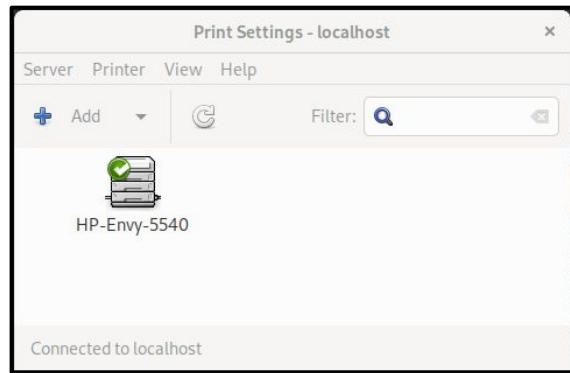
Video: Viewing, Editing & Codecs

- Remember: we did not add Packman
- VLC, [org.videolan.VLC](#)
 - Has the proper codecs
- Pitivi, [org.pitivi.Pitivi](#)
 - Has the proper codecs
- Openshot, [org.openshot.OpenShot](#)
 - Has the proper codecs
- Cheese, [org.gnome.Cheese](#)
 - Works well with my webcam



Printing & Scanning

- Printing
 - By default: no cups, no PPDs, ...
 - Tried installing (transactional-update)
 - It works!
 - OBS request [840921](#)
 - Should just work for everyone now
- Scanning
 - By default: no sane packages
 - Tried installing (transactional-update)
 - Flatpak apps (e.g., Paper) don't work yet
 - Still working on it
 - (yeah, most scanners, e.g., from all-in-one printers, have Web-ish interface, but still)



Writing On / Building QEMU

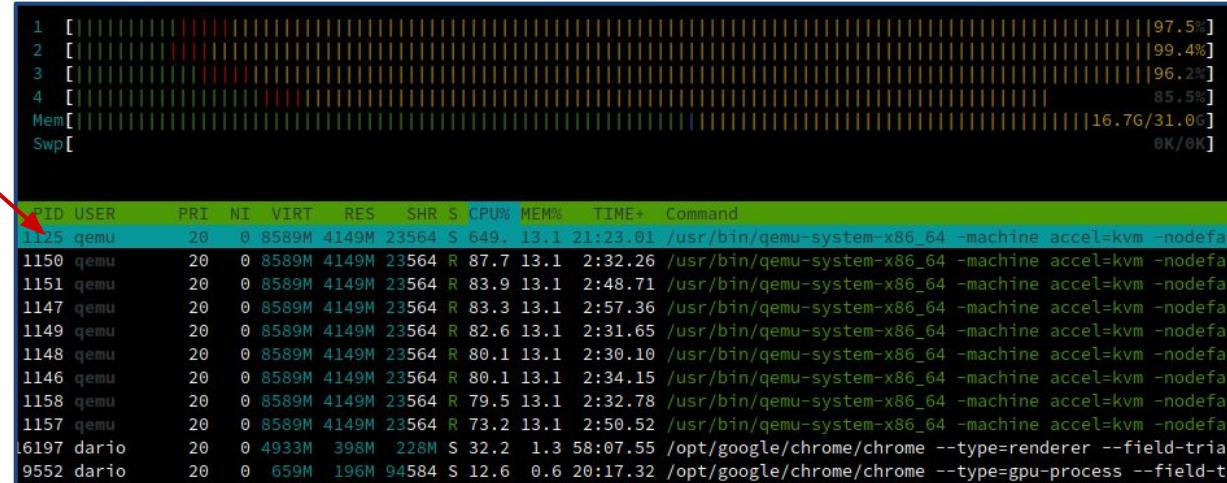
- Dependencies for building QEMU from sources:
 - bc bison bluez-devel brlapi-devel bzip2 ccache clang cyrus-sasl-devel flex gcc gcc-c++ gettext-tools git glib2-devel glusterfs-devel gtk3-devel gtkglext-devel gzip hostname libSDL2-devel libaio-devel libasan4 libcap-devel libcap-ng-devel libcurl-devel libfdt-devel libgcrypt-devel libgnutls-devel libjpeg62-devel libnettle-devel libnuma-devel libpixman-1-0-devel libpng16-devel librbd-devel libseccomp-devel libspice-server-devel libssh-devel libssh2-devel libtasn1-devel libudev-devel libxml2-devel lzo-devel make makeinfo multipath-tools-devel ncurses-devel perl pkg-config python3 python3-PyYAML python3-Sphinx rdma-core-devel snappy-devel sparse tar usbredir-devel virglrenderer-devel vte-devel which xen-devel zlib-devel
 - You don't want to install them with transactional-update and reboot
 - Oh, you forgot one / there is a new one needed:
 - Install with transactional-update and reboot again?
 - Do try! I promise that it **won't** be funny :-/
- Toolbox to the rescue:
 - \$ toolbox -u -t dev # -r may or may not be needed. Generally not for building
 - > sudo zypper in <all_the_dependencies_above>
 - > cd <your QEMU sources directory in your home (it's there in the toolbox)>
 - > <do your changes>
 - > <build it>

Working With OBS

Requires installing packages, using VMs for building, etc.

- toolbox, what else ?!
- I need a -r one, for mounting filesystems in the build VM (I think)

```
$ toolbox -u -r -t dev
> zypper ar https://download.opensuse.org/.../openSUSE\_Tumbleweed/openSUSE:Tools.repo
> zypper in cpio osc build [...]
> osc mkpac / co / vc
> [...]
> osc vc
> osc build --vm-type=kvm
> osc commit
```



Building outside of VMs

Currently not working

- (but it's better to build In VMs anyway...)

Working on Libvirt and QEMU

Real scenario:

- I make a change in QEMU
- I make a change in Libvirt
- I want to build and test both, with my changes

How it works for me:

1. I work on the changes themselves inside my development toolbox
2. Still in there, I start my modified libvirtd, make it listed on TCP (no socket activation)
 - \$ toolbox -r -u -t dev
\$> <work on QEMU> && <build QEMU> && <install my QEMU>
\$> <work on libvirt> && <build libvirt> && <install my libvirt>
\$> sudo ./build/src/virtlogd &
\$> sudo ./build/src/libvirtd -v -l
3. From either the same or a different toolbox I start virsh and/or virt-manager and connect to my modified libvirtd
 - \$ toolbox -u # this is my user/dev apps toolbox
\$> virsh --connect=qemu+tcp://localhost/system
\$> virsh # list --all
Id Name State

- Tumbleweed shut off

```
$ toolbox -u # this is my user/dev apps toolbox
$> virsh --connect=qemu+tcp://localhost/system
$> virsh # list --all
Id Name State
-----
- Tumbleweed shut off
```

Working on Libvirt and QEMU

The screenshot shows a terminal window with a tmux session running libvirtd, and a separate window showing the virt-manager interface.

Terminal (tmux session):

```
@:~/src/libvirt.git
Tumbleweed on QEMU/KVM: 192.168.0.30 (on toolbox-dario-user)
File Virtual Machine View Send Key
[2.590083] pcieport 0000:00:02.6: pciehp: Failed to check link status
[2.591085] pcieport 0000:00:02.8: pciehp: Failed to check link status
Welcome to openSUSE Tumbleweed 20201004 - Kernel 5.8.12-1-default (tty1).
localhost login:
```

Virt-manager window:

Virt-manager running in my `toolbox -u` and connecting to libvirtd in the other toolbox

Name	Status
QEMU/KVM: 10.156.208.91 - Not Connected	
QEMU/KVM: 10.156.208.94	
leap-vm	Shutoff
tw-vm1	Shutoff
vm-leap01	Running
vm-leap02	Shutoff
vm-leap03	Shutoff
vm-leap04	Shutoff
vm-leap05	Shutoff
vm-leap06	Shutoff
vm-leap07	Shutoff
vm-leap08	Shutoff
vm-leap09	Shutoff
vm-leap10	Shutoff
vm-leap11	Shutoff
vm-leap12	Shutoff
GEMU/KVM: 192.168.0.30	
Tumbleweed	Running

Libvirtd running in a tmux session running inside my `toolbox -r -u -t dev`

A Day in the Life of a Developer who Uses MicroOS as Workstation...

- I hacked on toolbox in such a way that:
 - With toolbox -u and/or toolbox -r -u:
 - You have your user inside the toolbox
 - You have your home, in its usual place
 - Your files have the proper owner, group, permissions
 - You reach your SSH agent (running on the host)
 - You can launch graphical apps
 - You have sudo
- Also:
 - With -t, you can have multiple toolbox-es, e.g.:
 - One per each project you're working on?
 - One for work projects and one for home projects?
 - One for
- IOW: It's a quite cool development environment
 - I adopted it even on Tumbleweed, before moving to MicroOS!

A Day in the Life of a Developer who Uses MicroOS as Workstation...

My morning routine:

1. Wake-up / wake-up the kids / have breakfast with them / bring them to school ;-P
2. Brew some more coffee
3. Open gnome-terminal
4. Enter a toolbox -r -u -t dev (brings me inside toolbox-dario-user-dev)
5. Start tmux inside that toolbox
 - a. all panes will be inside the toolbox already!
 - b. Stay in there until end of day
6. Maybe, enter my toolbox -u (brings me inside toolbox-dario-user)
 - a. Use some apps from there that I need but don't want to install in the base OS
7. <<Hey network to the office seems slow!>>
 - a. \$ toolbox -r

```
#> zypper in traceroute  
#> traceroute www.suse.com
```
8.

Some Stats

- RPM Packages
 - On my MicroOS Desktop: ~1000
 - But I've done a few experiments, added stuff, ...
 - On a stock Fedora Silverblue: ~1200
 - On a Tumbleweed box I also have: ~3500
 - Not used for development (so no -devel pkgs)
 - A few apps as flatpak there as well
 - On a development toolbox on my MicroOS Desktop: ~1300
 - No Desktop Environment packages
 - But with some GUI apps & libs
- Flatpaks
 - Apps installed: 68
 - All flatpaks (including runtimes): 110
 - Disk space: 12 GB

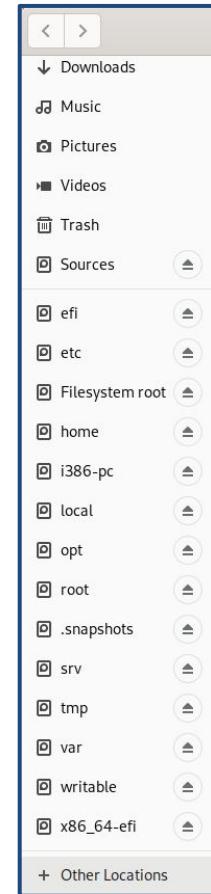
Example: Nautilus, Trash, USB Keys, From “not working” to “it works!”

Problem:

- Nautilus was looking weird (showing all BTRFS subvolumes, etc)
- Trash was not working
 - Files going in .local/share/Trash
 - Not being shown when clicking on “Trash” icon
- USB keys not being (auto)mounted, /run/media/<user> not appearing

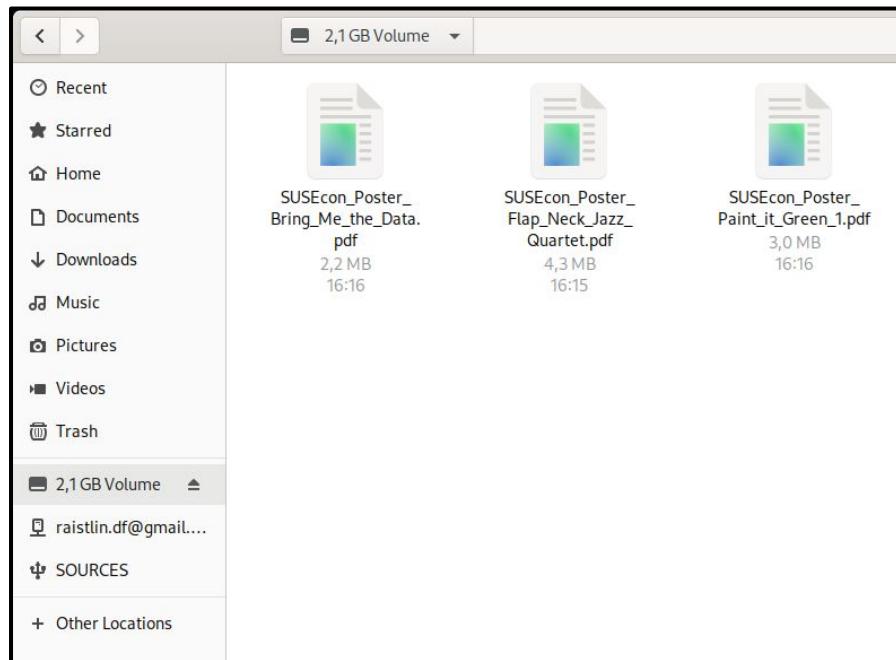
Let's try something...

- Mounting USB keys in /run/<user>/<volume> ⇒ it's udisks2
- On a Tumbleweed:
 - ps aux | grep udisk ⇒
/usr/libexec/gvfs/gvfs-udisks2-volume-monitor
/usr/libexec/udisks2/udisksd
 - rpm -qf ⇒
gvfs-backends-1.44.1-2.4.x86_64
udisks2-2.8.4-1.3.x86_64



Example: Nautilus, Trash, USB Keys

- Let's fix it!
 - \$ sudo transactional-update pkg in gvfs-backends udisks2
\$ sudo reboot
- It works!
- OBS request [840921](#)
- Should just work for everyone now

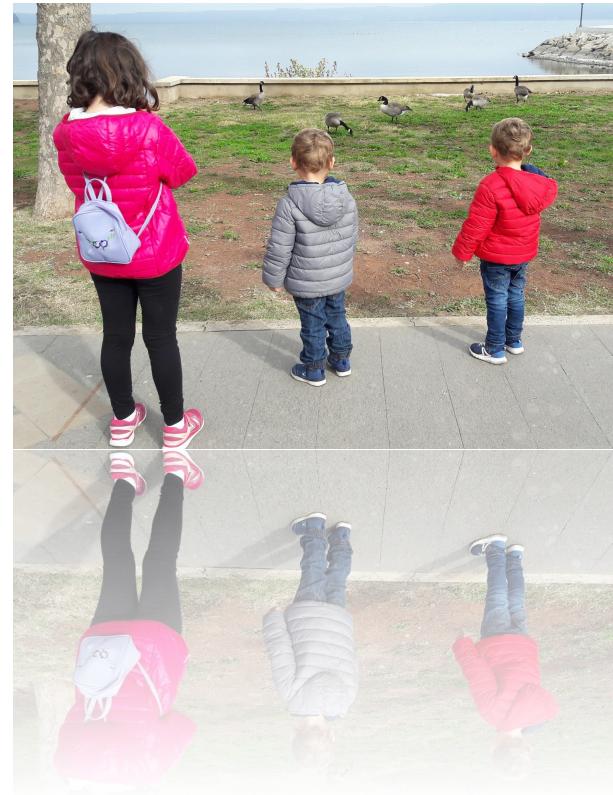


Conclusions

- Using MicroOS as a Desktop / Workstation is already possible, IME
 - Requires some manual fiddling with configurations, but it's mostly something done right after install and then forgotten
- It's pretty comfortable to use
 - In fact, I started using it just as an experiment. But I'm definitely staying!
- It pushes you to do things properly
 - No quick-&-dirty hacks, like symlinking that library to make that other app work
 - Results is a much cleaner and stable system
- It's not perfect yet:
- It asks for a password too many times, post install manual config steps should be done automatically, we may want to have a GUI way for updating the base OS (like Silverblue does), etc.
- **It needs you!** As a user, as a tester, as a contributor, as an "evangelist", as...
Well, whatever you want to do, you're welcome!

About Myself

- Ph.D on Real-Time Scheduling, [SCHED DEADLINE](#)
- 2011, Sr. Software Engineer @ Citrix
[The Xen-Project](#), hypervisor internals,
NUMA-aware scheduler, Credit2 scheduler,
Xen scheduler maintainer
- 2018, Virtualization Software Engineer @ [SUSE](#)
Still Xen, but also [KVM](#), [QEMU](#), [Libvirt](#);
Scheduling, VM's virtual topology,
performance evaluation & tuning





Finish

Thank You



All text and image content in this document is licensed under the Creative Commons Attribution-Share Alike 4.0 License (unless otherwise specified). "LibreOffice" and "The Document Foundation" are registered trademarks. Their respective logos and icons are subject to international copyright laws. The use of these thereof is subject to trademark policy.