“By The Power Of Toolbox!”

Dario Faggioli

Virtualization SW. Eng. @ SUSE

dfaggioli@opensuse.org

dariof

@DarioFaggioli

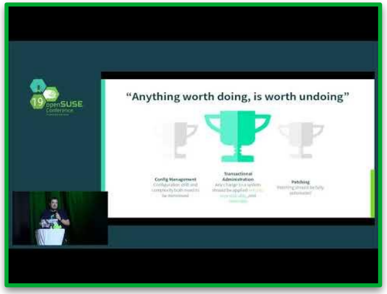
https://dariofaggioli.wordpress.com/
https://about.me/dario.faggioli

About ~~Me~~ What I do

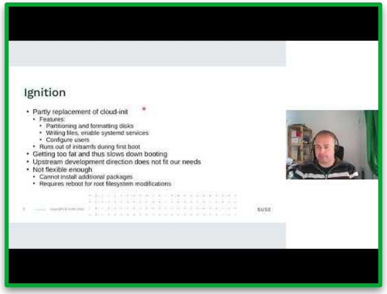
- Virtualization Specialist Sw. Eng. @ SUSE since 2018, working on Xen, KVM, QEMU, mostly about performance related stuff
- Daily activities ⇒ how and what for I use my workstation
  - Read and send emails (Evolution, git-send-email, stg mail, ...)
  - Write, build & test code (Xen, KVM, Libvirt, QEMU)
  - Work with the Open Build Service (OBS)
  - Browse Web
  - Test OSes in VMs
  - Meetings / Video calls / Online conferences
  - Chat, work and personal
  - Some 3D printing
  - Occasionally play games
  - Occasional video-editing
  - Maybe scan / print some document
- And all that, I do it with an openSUSE MicroOS, an immutable OS

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



19 openSUSE Conference
"Anything worth doing, is worth undoing"
Config Management
Transactional Administration
Patching



Ignition

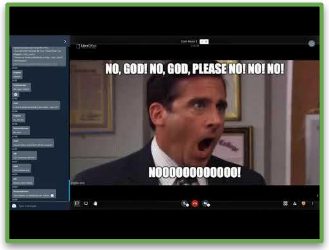
- Partly replacement of cloud-init
  - Features:
    - Partitioning and formatting disks
    - Writing files, enable systemd services
    - Configure users
  - Runs out of initramfs during first boot
- Getting too fat and thus slows down booting
- Upstream development direction does not fit our needs
- Not flexible enough
  - Cannot install additional packages
  - Requires reboot for root filesystem modifications

- Maybe, check this other talks (from yesterday):
  - An User & Developer Perspective on Immutable OSes
  - openSUSE MicroOS, a platform for everything from containers, to IoT, and even the desktop

MicroOS: Your Immutable Desktop

Single Purpose ⇒ Your Desktop / Workstation

- Rolling, but super stable
- Still early stage ~= ALPHA state
  - But usable already
  - (it's actually what I'm using since a few months)
- Growing community of users
- Small community of developers
  - We need your help! :-)



MicroOS

openSUSE MicroOS is a predictable & immutable. It cannot be altered during runtime.

MicroOS is reliable with automated updates and automated recover from faulty updates.

MicroOS is small with only what is needed to run it's "one job". Applications/Services are expected to be Containerised or Sandboxed.

Psst... For now, right after install, remember to do this:

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

Other Immutable Desktop OSes

- Fedora Silverblue

https://silverblue.fedoraproject.org/

"[...] unlike other operating systems, Silverblue is immutable. [...] Silverblue's immutable design is intended to make it more stable, less prone to bugs, and easier to test and develop."

- EndlessOS

https://endlessos.com/

"Endless is designed to feel natural and intuitive, making it easy to use even if you have little or no computer experience."

Installing Packages

Filesystems are real-only (that's the immutability!)

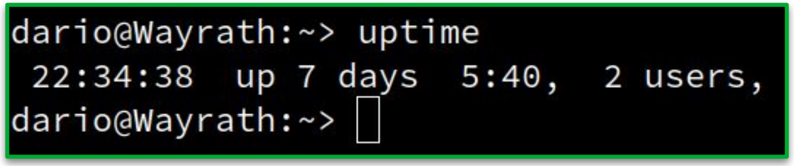
- On MicroOS, `zypper` does not work.
  Instead:
  - `transactional-update`

    -
      ```
      $ sudo transactional-update pkg install git-core
      $ systemctl reboot
      ```

    -
      ```
      $ sudo transactional-update shell
        # zypper ref
        # zypper install git-core
        # exit
      $ systemctl reboot
      ```
- Reboot always necessary, for seeing and using new Packages
  - See: The Transactional Update Guide

Are We Constantly Rebooting ?

This is my MicroOS workstation. Judge yourself:



```
dario@Wayrath:~> uptime
 22:34:38  up 7 days  5:40,  2 users,
dario@Wayrath:~> 
```

How so?

- For apps:
  - Flatpak
- For troubleshooting or debugging:
  - toolbox
- For development or "non-Flatpaked" apps:
  - toolbox

Installing/removing packages on the base OS tends to zero

What's a Toolbox ?

An easy way to start a read-write environment:

- With your user configured
- You have your home there, 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`
- You can install and remove packages

Sounds pretty handy:

- For installing apps not available/not working as Flatpaks
- For doing development inside it
- For troubleshooting and debugging the immutable OS "host"

Also:

- You can have multiple toolbox-es, active at the same time

Toolbox

Implemented as a privileged podman container

- Silverblue & EndlessOS
  - github.com/containers/toolbox
  - Was Bash, now Go (EndlessOS still using the old Bash version)
  - Works both "rootless" and as root
- openSUSE MicroOS (& Kubic)
  - github.com/kubic-project/microos-toolbox
  - Bash
  - Works both "rootless" and as root

BEWARE: "privileged container" & "can run as root"

- It's **not** a security enhancing tool
- <<I can do whatever I want, I'm in a container, I won't damage or corrupt the base OS, right?>> ⇒ **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!

MicroOS Toolbox

- A shell script that launches a privileged container
  - Check: toolbox - bring your own (debugging) utilities with you
  - Born with the troubleshooting use-case in mind. Evolved since then
  - Under active development, to improve it even further
- The container can be run:
  - As root
    - ⇒ When you are root inside the toolbox, you're kind of root on the host
    - (you may or may not also have your regular user in the toolbox, this depends on other parameters)
  - As your regular user
    - ⇒ Even if you are root in the toolbox, you are not root on the host
    - (you always have your regular user inside the toolbox)
    - Works thanks to "rootless podman"

MicroOS Toolbox Config File

- MicroOS toolbox has a config file:

  -
    ```
    $ cat ~/.toolboxrc
    REGISTRY=registry.opensuse.org
    IMAGE=opensuse/toolbox:latest
    TOOLBOX_NAME=special-debug-container
    TOOLBOX_SHELL="/bin/bash"
    ```

- `TOOLBOX_NAME`: for tweaking the basename of the containers
- `REGISTRY + IMAGE`: allows to use a different image for your toolbox-es
  - Can be overridden on command line
- `TOOLBOX_SHELL`: what shell --or program, for what matters-- to run by default
  - Can be overridden on command line

Using a Custom Container Image

Default toolbox container image:

- `toolbox/latest`is based on Tumbleweed.
- Really really minimal

Can be changed:

- You can have Leap toolbox-es
- You can make toolbox-es from your (Kiwi / OBS built) images
- In theory, you can have toolbox-es based on different distros

Config file:

- `REGISTRY=<registry>`
- `IMAGE=<image>`

Command line:

-
  ```
  toolbox -R <registry> -I <image>
  toolbox -i <full_image_URI>
  ```

Different Kind of Toolbox-es (I)

- toolbox running as your user, and you can only be root inside it:
  - Useful if you "only" want to do some simple debugging and troubleshooting of the host OS

```
$ toolbox         # no -u ⇒ no user except root, nothing in /home
 #> whoami
 root
 #> pwd
 /
 #>               # your are root already, but not "mapped" to root on host
 #> cat /proc/self/uid_map
         0       1000          1
         1     100000      65536
 #> exit
$                 # you are back in the host
```

Different Kind of Toolbox-es (II)

- toolbox running as your user, and you are your own user inside it:
    - Useful for using toolbox for running your apps and/or as your development environment

```
$ toolbox -u    # -u ⇒ you will have your user, your /home, etc when inside
 $> whoami
 dario
 $> pwd
 /home/dario
 $> sudo su     # you're becoming root in container. You still won't be
 #>             # be able to touch files owned by root on the host!
 #>
 #> cat /proc/self/uid_map
          0          1       1000
       1000          0          1
       1001       1001      64536
 #> exit
 $> exit
$               # you are back in the host
```

Different Kind of Toolbox-es (III)

- toolbox running as root, and you can only be root inside it:
    - Useful if you need to do serious debugging and troubleshooting of the host OS

```
$ toolbox -r      # -r ⇒ the toolbox runs as root on the host (started with sudo)
 #> whoami        # no -u ⇒ your own user is not there, only root
 root
 #> pwd
 /
 #>               # you're root in the container and that maps with
 #>               # root on the host (e.g., you'll be able to touch files
 #>               # owned by root on host)
 #> cat /proc/self/uid_map
         0           0 4294967295
 #> exit
$                 # you are back in the host
```

Different Kind of Toolbox-es (IV)

- Toolbox running as root on the host, and you are your own user inside it:
  - Useful for using toolbox for running your apps and/or your development environment, for things that need "special powers" and would not work rootless

```
$ toolbox -r -u # -u ⇒ you will have your user, your /home, etc when inside
 $>             # -r ⇒ the toolbox runs as root on the host (started with sudo)
 $> whoami
 dario
 $> pwd
 /home/dario
 $> sudo su     # you're becoming root in the container and that maps
 #>             # on root in the host (e.g., you'll be able to touch
 #>             # files owned by root on host)
 #> cat /proc/self/uid_map
         0          0 4294967295
 #> exit
 $> exit
$               # you are back in the host
```

Tagging

You can have multiple toolbox-es

- You can create multiple (different) toolbox-es and have them ready on the system
- You can be inside multiple (different) toolbox-es at the same time

How to distinguish? By their container's names

- `toolbox ⇒ ${TOOLBOX_NAME}-${USER}`
    - `toolbox-dario`
- `toolbox -t foo ⇒ ${TOOLBOX_NAME}-${USER}-foo`
    - `toolbox-dario-foo`
- `toolbox -u ⇒ ${TOOLBOX_NAME}-${USER}-user`
    - `toolbox-dario-user`
- `toolbox -u -t foo ⇒ ${TOOLBOX_NAME}-${USER}-user-foo`
    - `Toolbox-dario-user-foo`
- `toolbox [-u,-r] -c foo`
    - `foo`

Alternative User Interface

An alternative UI can be used, on MicroOS toolbox (so that we're more compatible with Silverblue toolbox)

- Create a rootless toolbox, with your own user inside it:
  - `toolbox -u`
  - `toolbox create`
    `toolbox enter`

- Create a "rootfull" toolbox, with your own user inside it:
  - `toolbox -r -u`
  - `toolbox create -r`
    `toolbox enter -r`

Toolbox-es Keep Their State

- toolbox is **stateful**
  - <u>Yesterday</u> you created a toolbox, and you install stuff, change configs, etc
  - <u>Today</u> you stop the toolbox, you turn off the PC and take the day off
  - <u>Tomorrow</u> toolbox will still have all the software and all the config changes you made

- For starting from scratch
  - Create another toolbox, with different tag (⇒ different name)
  - Remove the existing one

- Toolboxes are something in between "Pets" and "Cattles"

Managing Your Toolbox-es

- Listing running toolbox-es, created as user
  - Just list containers...
    - `$ podman ps`

```
CONTAINER ID  IMAGE                    COMMAND    CREATED     STATUS           NAMES
5cb19ade1fb1  [...]toolbox:latest sleep +Inf 3 weeks ago Up 3 hours ago toolbox-dario-user
```

- Listing all toolbox-es created as user (running ot not)
  - `$ toolbox list`
  - Or just list containers again:
    - `$ posman ps --all`

```
CONTAINER ID  IMAGE                    COMMAND    CREATED     STATUS      NAMES
5cb19ade1fb1  [...]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
```

Managing Your Toolbox-es

- Listing toolbox-es created as root:
    - Running ones:
        - `$ sudo podman ps`
    - All of them:
        - `sudo podman ps --all`
        - `toolbox list -r`

- Removing toolbox-es
    - Created as user:
        - `$ podman rm` *`<toolbox_name/ID>`*
    - Created as root:
        - `$ sudo podman rm` *`<toolbox_name/ID>`*

Toolbox For TroubleShooting

As said, toolbox is really handy for debugging and troubleshooting

For example, you need to do a:

```
$ strace ls
```

- You can try… but `strace` is not installed!
- Install it with and then reboot before being able to use it ? NO!

```
$ toolbox enter         # runs as your user on the host
 $> sudo su
 #> zypper in strace    # root in toolbox. Does not map
                        # to root on the host in this case
 #> strace ls           # here you go your strace!
```

Toolbox For TroubleShooting

As said, toolbox is really handy for debugging and troubleshooting

For (another) example, you need to:

```
$ sudo nmap -sS 192.168.0.2
```

- Again, `nmap` is not there, and you don't want to reboot

```
$ toolbox enter -r    # we need "root on host", to do SYN scan
 $> sudo su
 #> zypper install nmap
 #> nmap -sS 192.168.0.2
```

Toolbox for Development: Hacking On 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`
  - Install with `transactional-update`... ... ... and reboot every time you forget one?
    - No!
- Toolbox to the rescue:

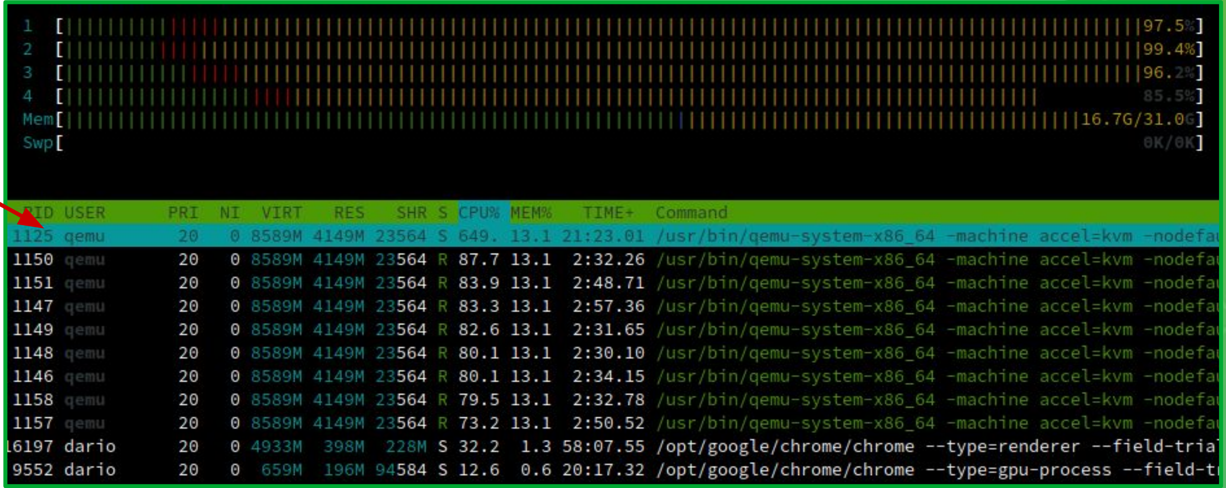
  -
    ```
    $ toolbox enter -c devel
     $> sudo zypper in <all_the_dependencies_above>
     $> cd <your QEMU sources directory in your home>
     $> <do your changes>
     $> <build it>
    ```

Toolbox for Development: Working With Open Build Service

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 enter -r -t devel
 $> 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
```



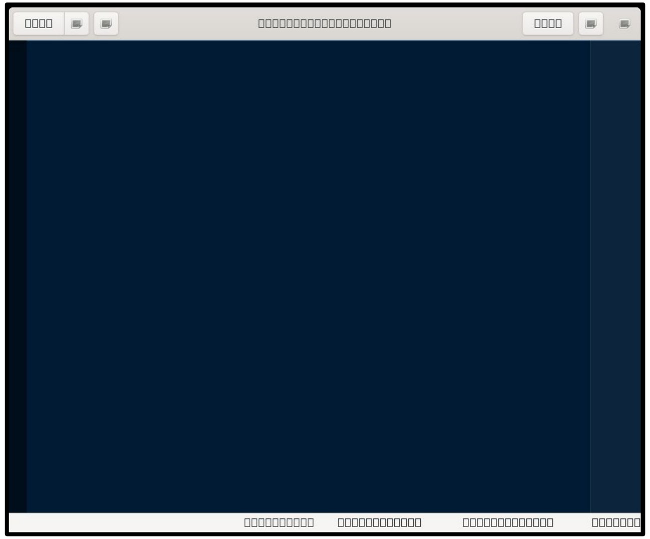
```
  1  [97.5%]
  2  [99.4%]
  3  [96.2%]
  4  [85.5%]
  Mem[16.7G/31.0G]
  Swp[0K/0K]

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 1125 qemu       20   0 8589M 4149M 23564 S 649. 13.1 21:23.01 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1150 qemu       20   0 8589M 4149M 23564 R 87.7 13.1  2:32.26 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1151 qemu       20   0 8589M 4149M 23564 R 83.9 13.1  2:48.71 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1147 qemu       20   0 8589M 4149M 23564 R 83.3 13.1  2:57.36 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1149 qemu       20   0 8589M 4149M 23564 R 82.6 13.1  2:31.65 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1148 qemu       20   0 8589M 4149M 23564 R 80.1 13.1  2:30.10 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1146 qemu       20   0 8589M 4149M 23564 R 80.1 13.1  2:34.15 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1158 qemu       20   0 8589M 4149M 23564 R 79.5 13.1  2:32.78 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
 1157 qemu       20   0 8589M 4149M 23564 R 73.2 13.1  2:50.52 /usr/bin/qemu-system-x86_64 -machine accel=kvm -nodefa
6197 dario       20   0 4933M  398M  228M S 32.2  1.3 58:07.55 /opt/google/chrome/chrome --type=renderer --field-tria
 9552 dario      20   0  659M  196M 94584 S 12.6  0.6 20:17.32 /opt/google/chrome/chrome --type=gpu-process --field-t
```

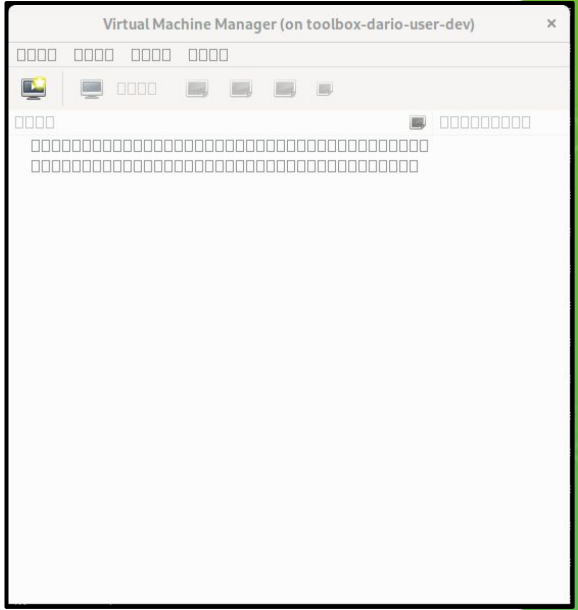
Toolbox for Graphical Apps

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

```
$> sudo zypper in gedit virt-manager
$> gedit
$> virt-manager
```

Errr… What?

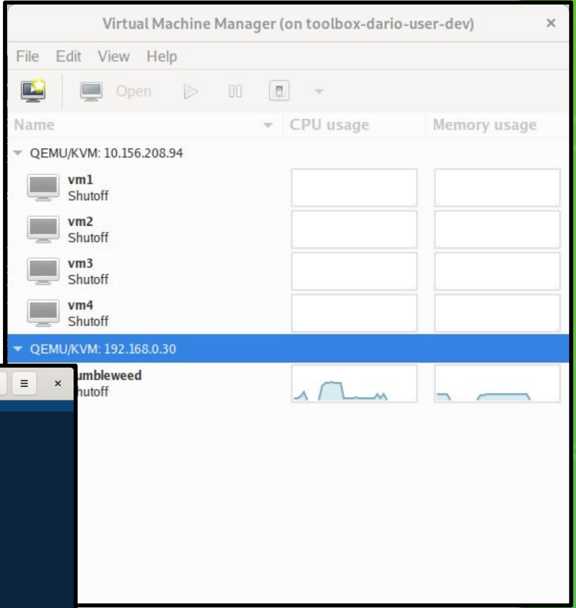


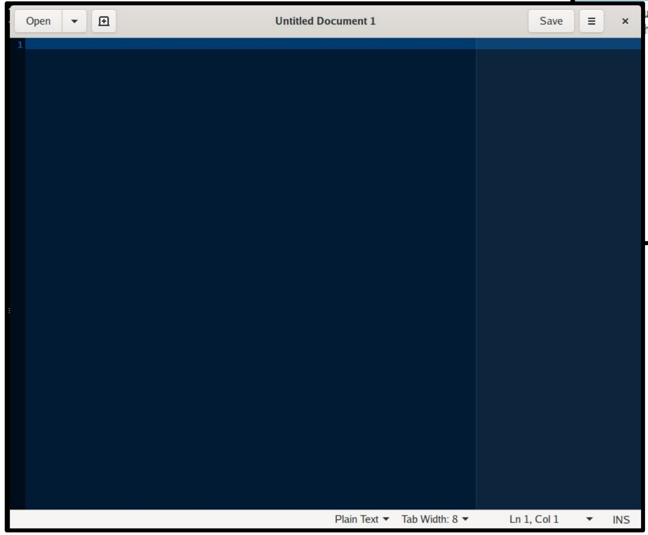


Toolbox for Graphical Apps

- 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?)

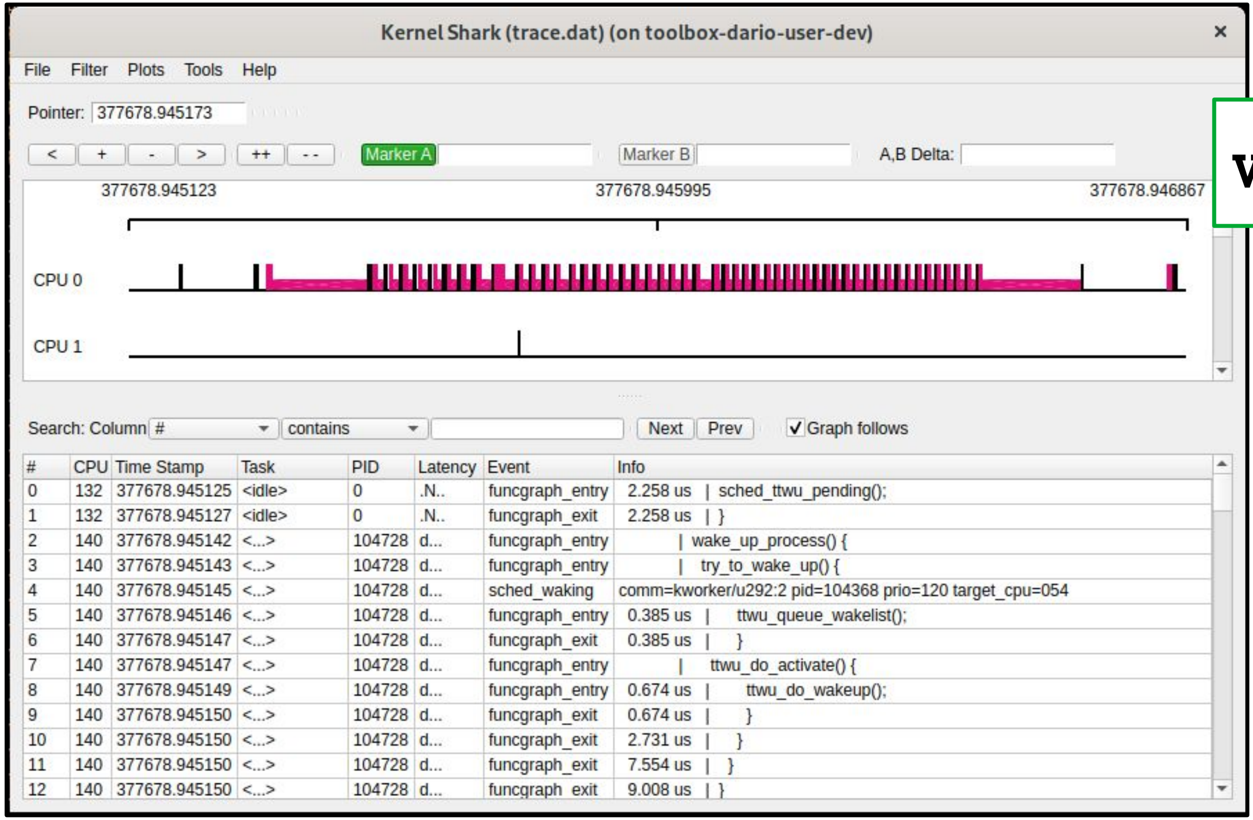
Toolbox for "GL" Graphical Apps

- You want to use Kernelshark:

  -
    ```
    $ 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
     ...
    ```
- I have NVIDIA with proprietary drivers here. What if...

  -
    ```
    $ toolbox
     $> sudo zypper addrepo https://download.nvidia.com/opensuse/tumbleweed NVIDIA
     $> sudo zypper ref
     $> sudo zypper in x11-video-nvidiaG05
    ```
- It installs stuff like:
  - `kernel-default-devel`, `nvidia-gfxG05-kmp-default`, `nvidia-glG05` … … Inside the container ?!?!

Toolbox for "GL" Graphical Apps



#	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 | }

Well, it works!

Working on Libvirt, QEMU & Kernel

Real scenario:

- I make a change in the Linux kernel
- I make a change in QEMU
- I make a change in Libvirt
- I want to build **and also test** my changes

How it works for me:

- I work on the changes inside my development toolbox
  - Run as root on the host... it's easier
  - `toolbox-dario-user-devel`

```
$ toolbox enter -r -t devel
 $> <work on Linux kernel> && <build the Linux kernel>
 $> <work on QEMU> && <build QEMU> && <install my QEMU>
 $> <work on libvirt> && <build libvirt> && <install my libvirt>
```

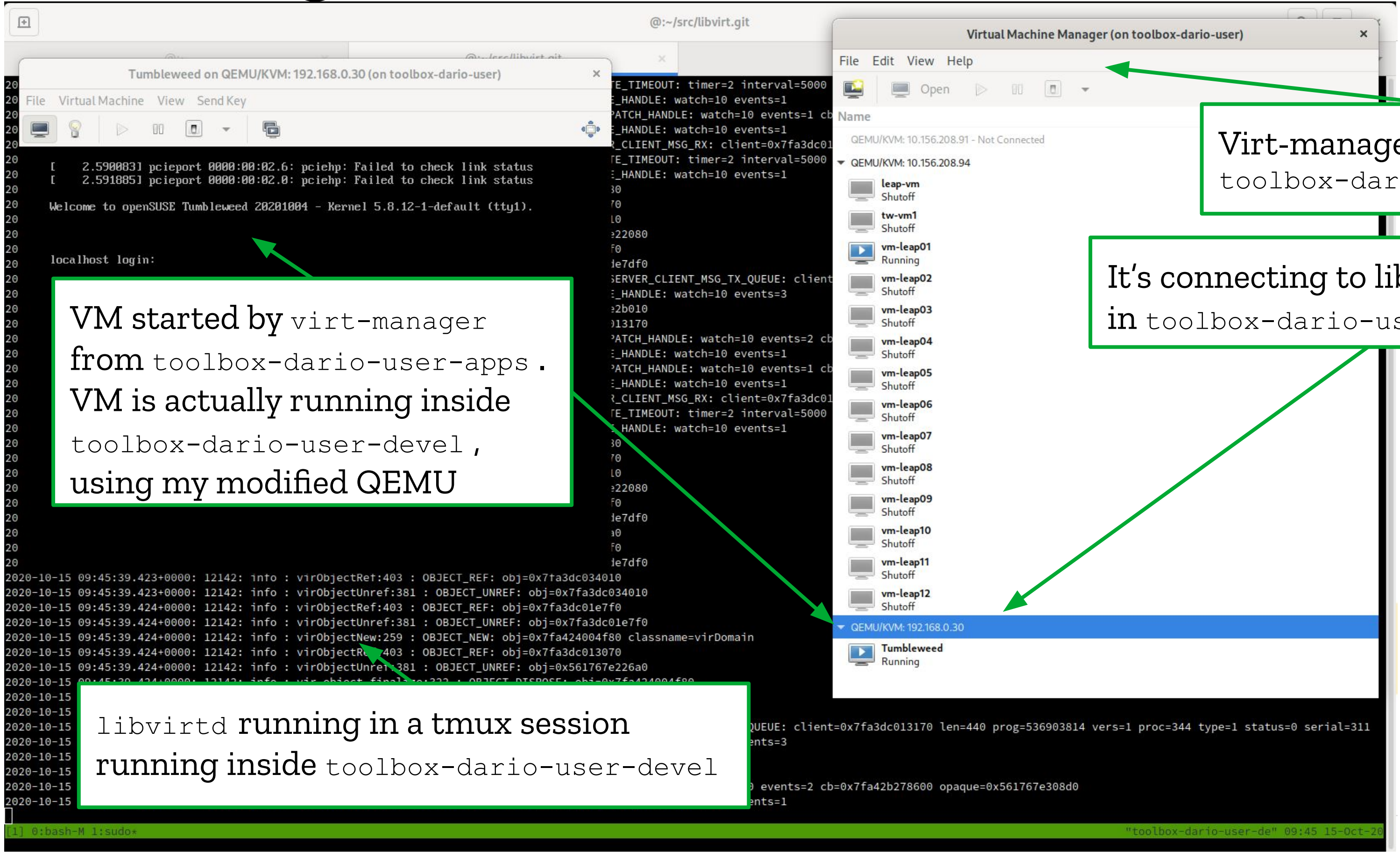
Working on Libvirt, QEMU & Kernel

- Still in the "dev toolbox", I can start my modified `libvirtd`
  - Make it listed on TCP (no socket activation)

  -
    ```
    $ toolbox enter -r -c devel
     $> <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
    ```
- From (either the same or a different) toolbox I start `virsh` and/or `virt-manager`
- I can connect to my modified libvirtd

  -
    ```
    $ toolbox enter -c apps  # this is my user/dev apps toolbox
     $> virsh --connect=qemu+tcp://localhost/system
     $> virsh # list --all
     Id   Name          State
     -----------------------------
      -    Tumbleweed   shut off
    ```
- I can define or edit a VM so that it boots my modified QEMU & kernel

Working on Libvirt, QEMU & Kernel


@:~/src/libvirt.git
Tumbleweed on QEMU/KVM: 192.168.0.30 (on toolbox-dario-user)
File
Virtual Machine
View
Send Key
Virtual Machine Manager (on toolbox-dario-user)
File
Edit
View
Help
Open
Name

```
[    2.590083] pcieport 0000:00:02.6: pciehp: Failed to check link status
[    2.591885] pcieport 0000:00:02.0: pciehp: Failed to check link status

Welcome to openSUSE Tumbleweed 20201004 - Kernel 5.8.12-1-default (tty1).


localhost login:
```

```
TE_TIMEOUT: timer=2 interval=5000
E_HANDLE: watch=10 events=1
PATCH_HANDLE: watch=10 events=1 cb
E_HANDLE: watch=10 events=1
R_CLIENT_MSG_RX: client=0x7fa3dc01
TE_TIMEOUT: timer=2 interval=5000
E_HANDLE: watch=10 events=1
30
70
10
e22080
f0
de7df0
SERVER_CLIENT_MSG_TX_QUEUE: client
E_HANDLE: watch=10 events=3
e2b010
013170
PATCH_HANDLE: watch=10 events=2 cb
E_HANDLE: watch=10 events=1
PATCH_HANDLE: watch=10 events=1 cb
E_HANDLE: watch=10 events=1
R_CLIENT_MSG_RX: client=0x7fa3dc01
TE_TIMEOUT: timer=2 interval=5000
E_HANDLE: watch=10 events=1
30
70
10
e22080
f0
de7df0
a0
f0
de7df0
2020-10-15 09:45:39.423+0000: 12142: info : virObjectRef:403 : OBJECT_REF: obj=0x7fa3dc034010
2020-10-15 09:45:39.423+0000: 12142: info : virObjectUnref:381 : OBJECT_UNREF: obj=0x7fa3dc034010
2020-10-15 09:45:39.424+0000: 12142: info : virObjectRef:403 : OBJECT_REF: obj=0x7fa3dc01e7f0
2020-10-15 09:45:39.424+0000: 12142: info : virObjectUnref:381 : OBJECT_UNREF: obj=0x7fa3dc01e7f0
2020-10-15 09:45:39.424+0000: 12142: info : virObjectNew:259 : OBJECT_NEW: obj=0x7fa424004f80 classname=virDomain
2020-10-15 09:45:39.424+0000: 12142: info : virObjectRef:403 : OBJECT_REF: obj=0x7fa3dc013070
2020-10-15 09:45:39.424+0000: 12142: info : virObjectUnref:381 : OBJECT_UNREF: obj=0x561767e226a0
2020-10-15
2020-10-15          QUEUE: client=0x7fa3dc013170 len=440 prog=536903814 vers=1 proc=344 type=1 status=0 serial=311
2020-10-15          ents=3
2020-10-15
2020-10-15
2020-10-15          events=2 cb=0x7fa42b278600 opaque=0x561767e308d0
2020-10-15          ents=1
[1] 0:bash-M 1:sudo*                                              "toolbox-dario-user-de" 09:45 15-Oct-20
```

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

QEMU/KVM: 192.168.0.30
- Tumbleweed Running

Virt-manager running in `toolbox-dario-user-apps`

It's connecting to libvirtd running in `toolbox-dario-user-devel`

VM started by `virt-manager` from `toolbox-dario-user-apps`. VM is actually running inside `toolbox-dario-user-devel`, using my modified QEMU

`libvirtd` running in a tmux session running inside `toolbox-dario-user-devel`

Where's the Catch ?

<<It sounds complex, having to manage all those toolbox-es that you have around!>>

a real comment that I got after a presentation)

Answer: **Not really**

My Setup:

- I have: 2 toolbox-es in total, really (my "pet toolbox-es")
- I may fire up some throwaway ones during the day, depending on what I do ("cattle toolbox-es)

My "morning routine":

1. Open `gnome-terminal`
2. `toolbox enter -r -t devel` ⇒ `toolbox-dario-user-devel`
3. Start `tmux` inside that toolbox
   a. all panes will be inside the toolbox already
   b. I'm pretty much in there all the time
4. Maybe, in other terminal tabs:
   a. `toolbox enter -t apps` ⇒ `toolbox-dario-user-apps`)
   b. For non-flatpak apps, for testing, etc

MicroOS toolbox "vs" Silverblue toolbox

Different projects

- Started at different times, with different goals:
  - MicroOS / Kubic had no desktop flavour
  - Goal was "Only" troubleshooting

- Now they share a common goal, but:
  - MicroOS toolbox need to stay compatible with backward compatible (don't upset users)
  - They're grown apart (e.g., Bash vs. Go)
  - Silverblue one is more advanced and fancy, but also complex
  - MicroOS one is very simple and yet it delivers quite well

- (My) Current goal:
  - Make the user experience similar
  - so users can jump between the two without issues

MicroOS toolbox "vs" Silverblue toolbox

UI is almost compatible (at least!)

- Create a toolbox
  - Silverblue: `toolbox create`
  - MicroOS
    - Either: `toolbox -u`
    - Or: `toolbox create`
- Entering a toolbox:
  - Silverblue:
    - `toolbox enter`
  - MicroOS
    - Either: `toolbox -u    # creates it, if doesn't exist`
    - Or: `toolbox enter`
- Create (and enter) a toolbox as root:
  - Silverblue: `sudo toolbox create && sudo toolbox enter`
  - MicroOS:
    - Either: `toolbox -u -r`
    - Or: `toolbox create -r && toolbox enter -r`

MicroOS toolbox "vs" Silverblue toolbox

- Tagging
  - Silverblue:
    - `toolbox create -c foo` ⇒ `foo`
  - MicroOS:
    - `toolbox -u -t foo` ⇒ `toolbox-dario-user-foo`
    - `toolbox create -c foo` ⇒ `foo`
- Building the toolbox from a specific image
  - Silverblue: `toolbox create -i` *`<full_image_URI>`*
  - MicroOS:
    - `Toolbox create -R` *`<registry>`* `-I` *`<image>`*
    - `Toolbox create -i` *`<full_image_URI>`*
- Removing containers and images
  - Only Silverblue:
    - `toolbox rm` *`<container>`*
    - `toolbox rmi` *`<container_image>`*

And There Are (Were?) Even More...

- Podbox, github.com/DimaZirix/podbox
    - Specifically targeting GUI apps
    - Typical usage: one (or a couple, at most) app(s) per podbox
    - E.g., `podbox create firefox --gui --net --ipc --audio`
    - Kind of resembles flatpaks
        - ... Good idea for apps not available as such?
- Coretoolbox, github.com/cgwalters/coretoolbox
    - Reimplementation of Silverblue toolbox
    - In Rust (started when toolbox was still Bash)
        - Goal was making it more generic and flexible
        - Push toward "cattle toolbox-es" rather than "pet toolbox-es"
        - Now, however:
            - (Silverblue) toolbox is no longer Bash (Go)
            - (Silverblue) toolbox has been generalized a bit itself

Conclusions

- An immutable OS, like openSUSE MicroOS, can be your desktop
- It's going to be great!
  - (Reach out if you try: t.me/openSUSE_MicroOS_Desktop)
- If you wouldn't have a toolbox... not so much great
  - How do you do troubleshooting ?
  - How do you do development ?
- "Toolbox" is an idea, and various implementations exists
- Make your workflow fit Immutable OS + toolbox seems complicated...

  ... but it's not :-)

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

openSUSE®