Containerization as a means of extending the lifetime of HDL development tools

Clyde Laforge, Hamza Boukabache, CROME Team

24 May 2022

About me

- Electronic Engineer
- Joined CERN as Fellow in July 2022
- Working in HSE-RP-IL
  - CROME Radiation Monitoring System
  - Vivado/Petalinux 2018.1
  - Moving to Alma Linux 9
- Not a security expert

- Don’t hesitate to ask questions during or after the presentation: clyde.laforge@cern.ch

Main issue

- Crucial HDL software's End-Of-Life (EOL) are tied to the life of their officially supported distribution
- Distribution choice is limited by CERN's selection
- Xilinx often does not target long term support releases
- Death of CentOS complicates the situation

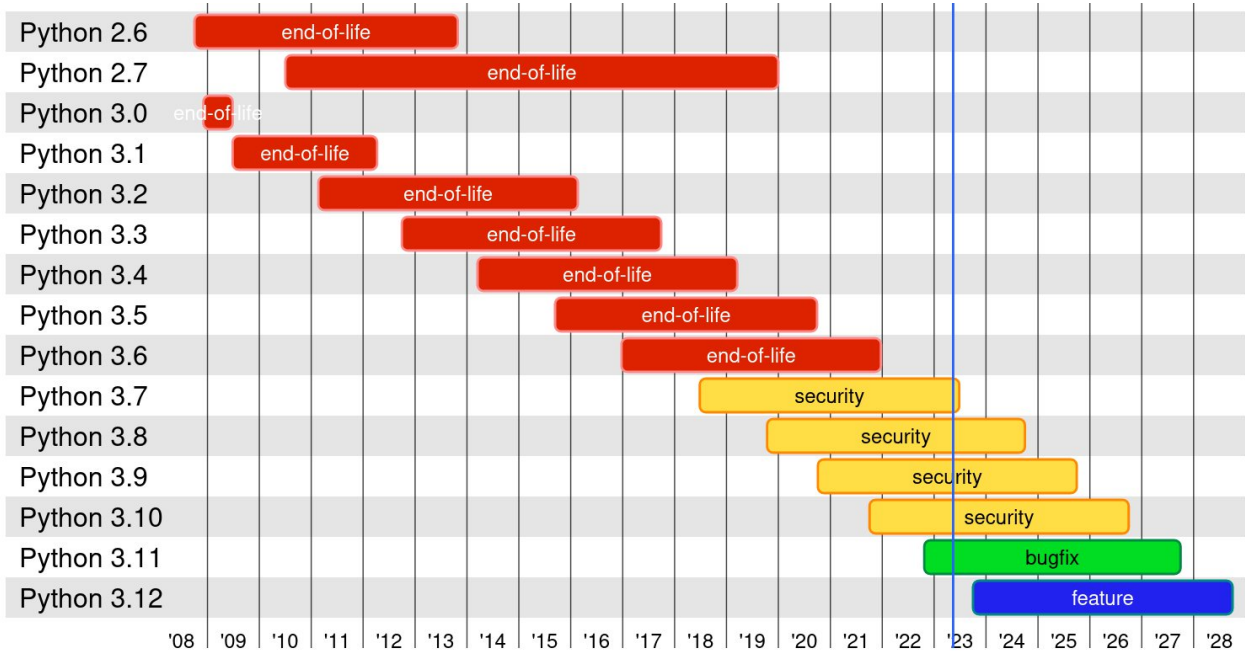
Vivado version	Longest lived distro	Longest distro	Longest lived series
2018.1	July 2018	CC6.9	CC7
2019.1	September 2019	CC7.6	CC7
2020.1	June 2020	CC8.1	RHEL8
2021.1	June 2021	CC8.3	RHEL8
2022.1	June 2024	CC7.9	RHEL8
2023.1	June 2024	CC7.9	RHEL9

Always update distro?

- Usually fine when the same major version is kept
- May not be possible when breaking changes are introduced
- Support may be lacking

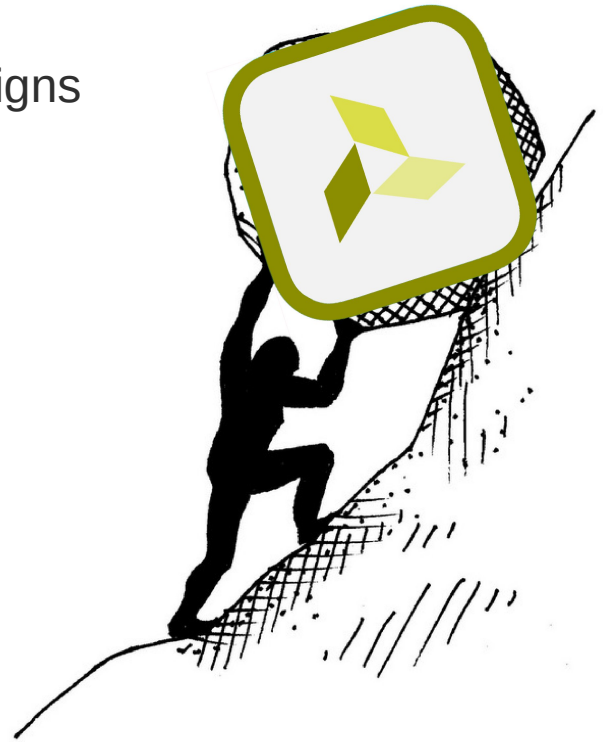
Staying on old distributions?

- Software may only be available for newer distributions/software version (e.g: cocotb-test requires python >= 3.7)
- Newer hardware drivers may be required
- Bad security-wise
- User experience has improved since then

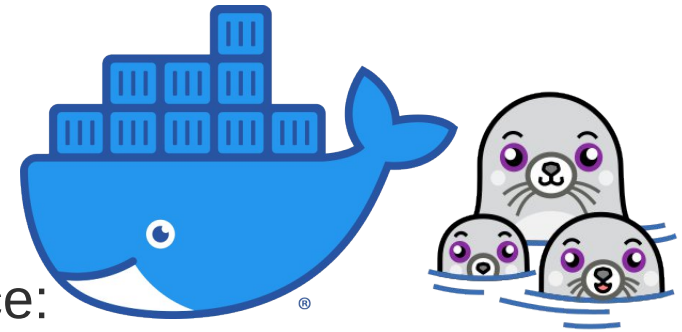


Always update software?

- Time and resources wasted on software install
- Design may need to be updated
- Changes to the build chain may introduce bugs
  - Long tests may be necessary, especially for reliability-critical designs



What about containers?



podman

Great solution with poor developer experience:

- How to access the project directory from within the container?
- How can I open the GUI?
- Do I need to update all of my scripts to call `docker […] vivado -mode tcl myscript.tcl` rather than vivado directly?

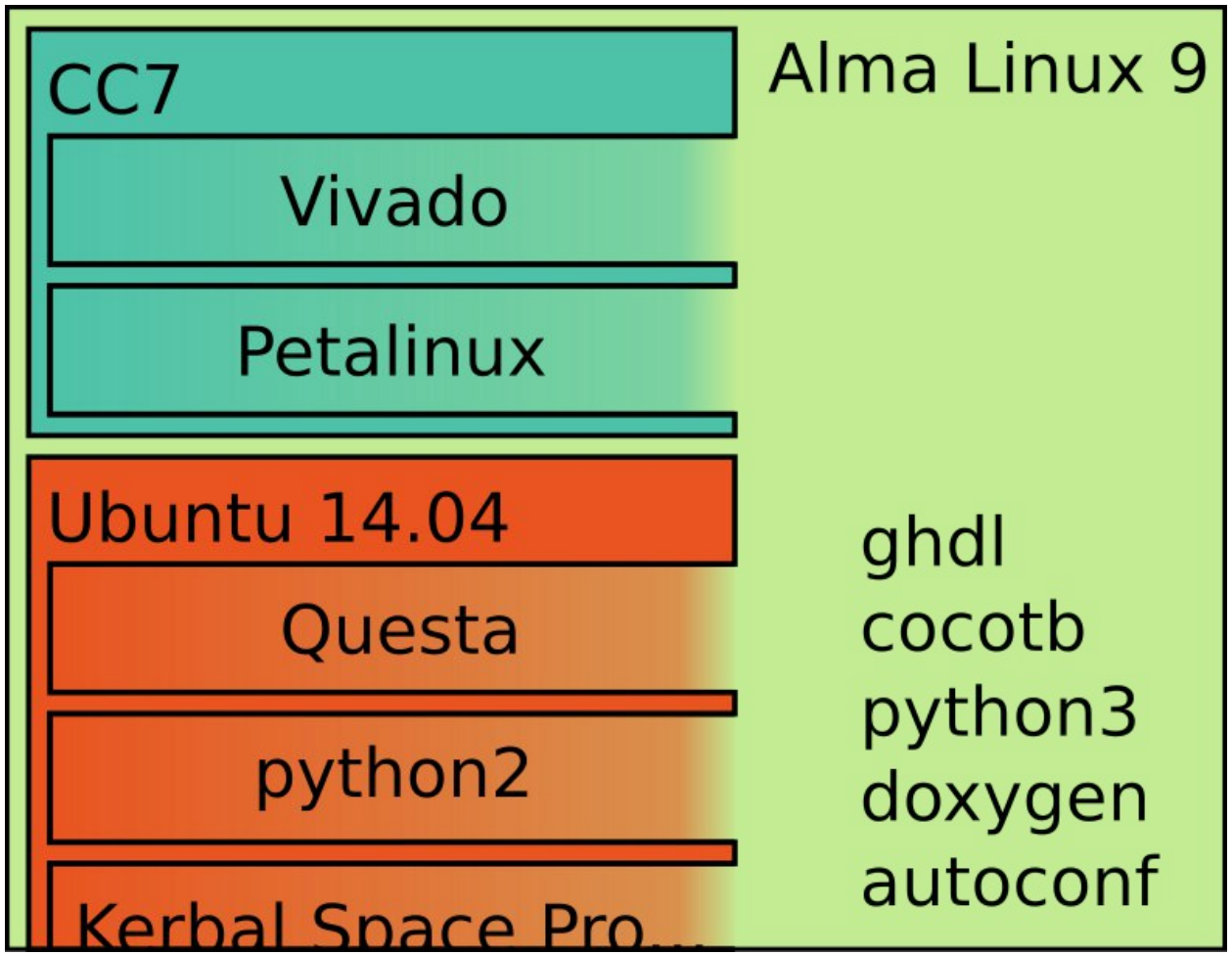
A way to present the software as-if it is installed would be the best fit as it keeps the interface the same for both the scripts and user

Introducing distrobox

- Collection of shell scripts wrapping around podman/docker
- Abstracts complexity and allow seamless integration with host environment
- Can run on top of existing images
  - CCE's docker images can be used!

Distrobox – Main idea

- Software live in their officially supported environment
- Software is accessible from the host environment by:
  - Entering the distrobox, or
  - Exported binary/application
- Integration is transparent:
  - GUI (Wayland / Xorg)
  - Audio (Pipewire/Pulse)
  - SSH/GPG agents
  - ...



Security

- This is **not a security enhancing tool**
  - You're not less secure than when on base OS
  - You're not much more safe either
- Rootless Distrobox using Podman:
  - Even when you are root in the container, you are not root on the host
  - You have your regular user
- Rootful Distrobox (Podman or Docker):
  - **You're root in the host and the container!**
- EOL distribution's attack surface is likely reduced, but is it enough?

Compatibility

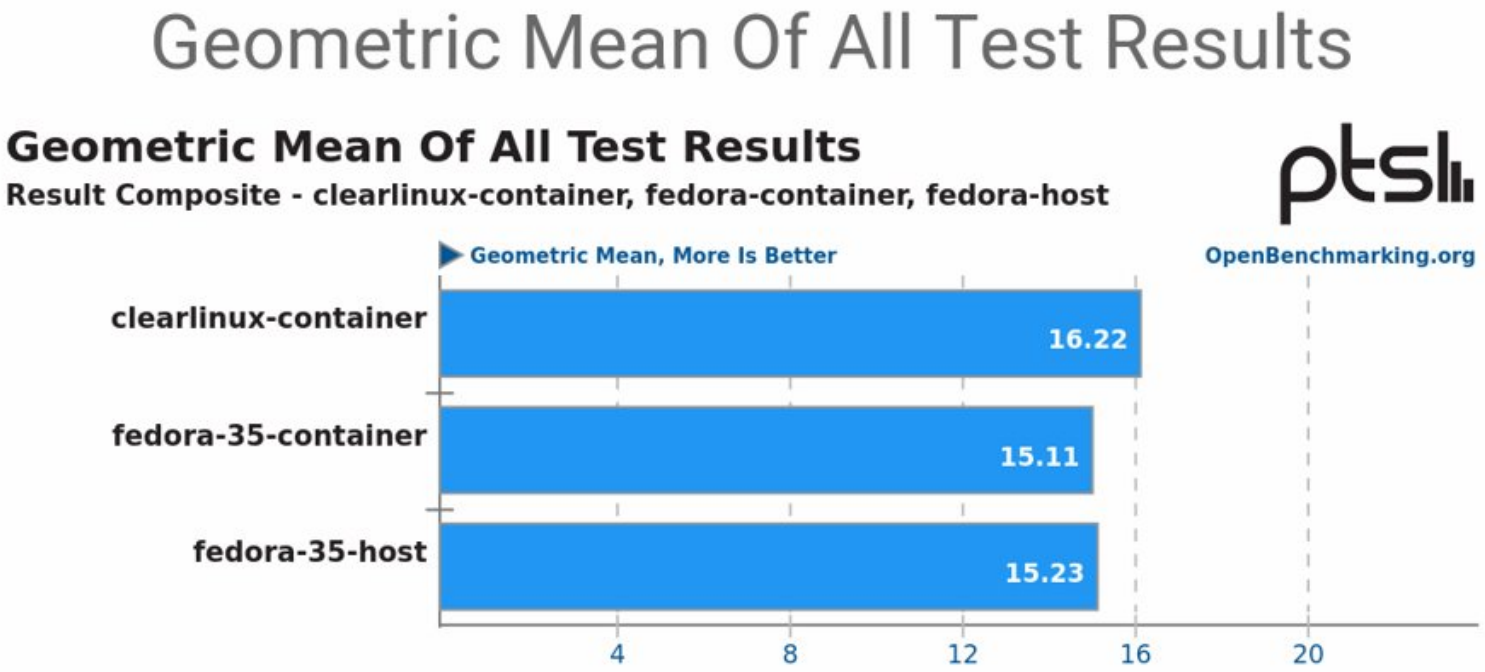
- Container compatibility starting from Debian 7, CentOS 7, Ubuntu 14.04 to most modern distributions
- Host compatibility to most modern distribution including Alma Linux 9 and RHEL9

- CentOS 7, Alma Linux/RHEL 8 and 9 are all both host and container compatible!
  - "Installing" Alma Linux 9 software on CentOS 7 is possible
  - "Installing" CentOS 7 software on Alma Linux 9 is possible

Performance

Containers are not Virtual Machines (VMs)!

- Insignificant overhead (1-2%)
- Newer, more optimized components may even run faster!



https://openbenchmarking.org/result/2201215-NE-CLEARLINU53

Workflow

1. Download HDL installers (optionnal)
2. Create a new distrobox container
```
clyde@host > distrobox create --image gitlab-registry.cern.ch/linuxsupport/cc7-base --name myCentOS7
```
3. Enter the distrobox
```
clyde@host > distrobox enter myCentOS7
clyde@myCentOS >
```
4. Install software and its dependencies
```
clyde@myCentOS > ~/Downloads/vivado-installer.sh
```
5. Export the software to the host OS
```
clyde@myCentOS > distrobox-export --app --login vivado
clyde@myCentOS > distrobox-export --bin --login /opt/Xilinx/2018.1/bin/vivado --export-path ~/.local/bin
```
6. Exit the distrobox and use the exported software as if it was installed
```
clyde@myCentOS > exit
clyde@host > vivado -mode tcl -source myScript.tcl
```

To be aware - Pitfalls

- Vivado puts its launcher file inside `~/Desktop` instead of `/usr/share/applications`
  - Must copy it inside the distrobox before exporting:
    ```
    sudo cp ~/Desktop/Vivado\ XXXX.X.desktop /usr/share/applications
    ```
- Environment variables needed must be sourced at login (inside `/etc/profile` or `/etc/profile.d/XXX`)
  - ```
    sudo sh -c 'echo "source /opt/Xilinx/Vivado/2018.1/settings64.sh" >> /etc/profile.d/source-vivado.sh'
    ```
- Known bugs:
  - Spaces in exported filenames may cause issues
  - Arguments of exported binaries may not be parsed correctly
- Fixes to the bugs above are available in my fork:
  - https://github.com/Scafir/distrobox
  - Pending rewrite and inclusion in upstream
    ```

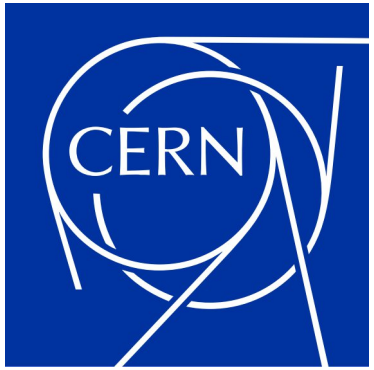
Demo

https://youtu.be/gf4JimmP4nw

Highlights

- Special thanks to Adrian Byszuk (SY-EPC) and his colleagues for their continued work on the container images

- Distrobox Author: Luca Di Maio
  - Inspiration was taken from his presentation at the Open Source Summit Europe



home.cern