Κεφάλαιο 13:Containers – RH134 Additional material

Γεώργιος Μαγκλάρας PhD
Associate Instructor
RedHat Academy
Πανεπιστήμιο Δυτικής Αττικής / University of West Attica

Scenarios on why?

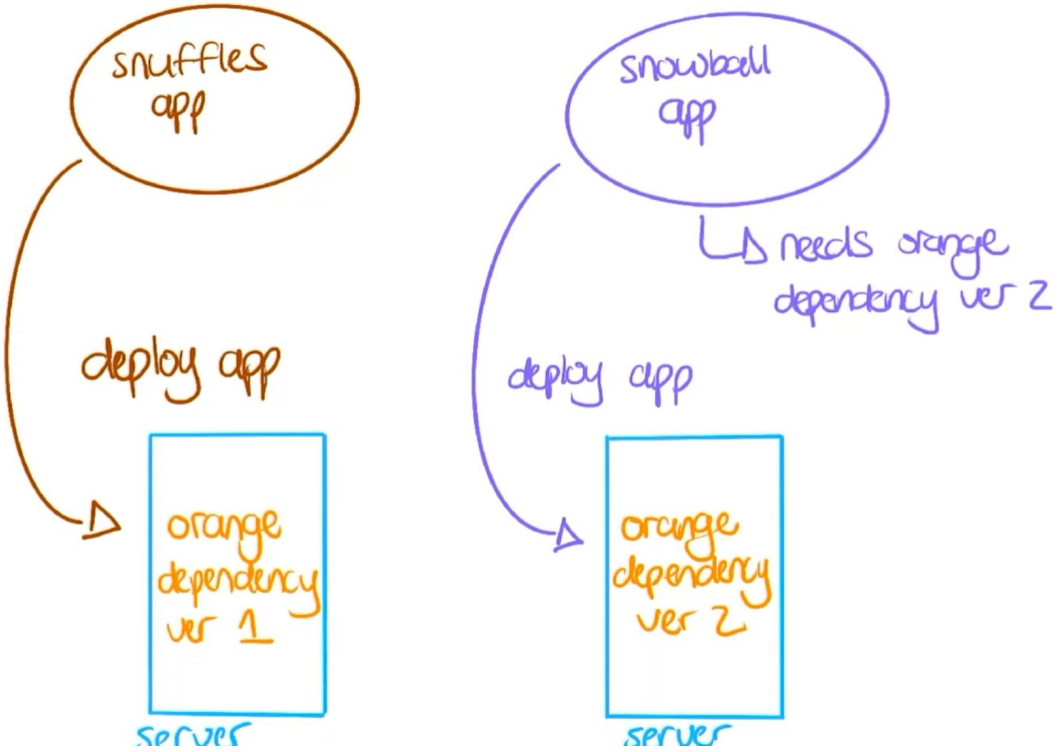
“I am often confused with which modules or conda environments I have to use. Can I not have a simpler mechanism?” (ISC 2022)

“I am developing on Ubuntu, but my supercomputer vendor runs on some sort of RedHat variant.” (MET)

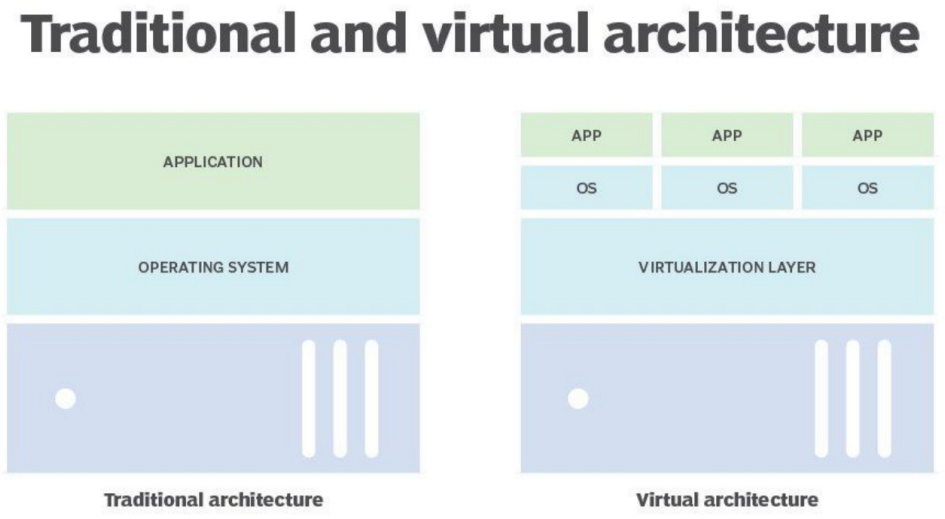
“When I distribute my complex application with a gazillion dependencies, how do I ensure it will run properly in systems I do not control?” (ISC 2022)

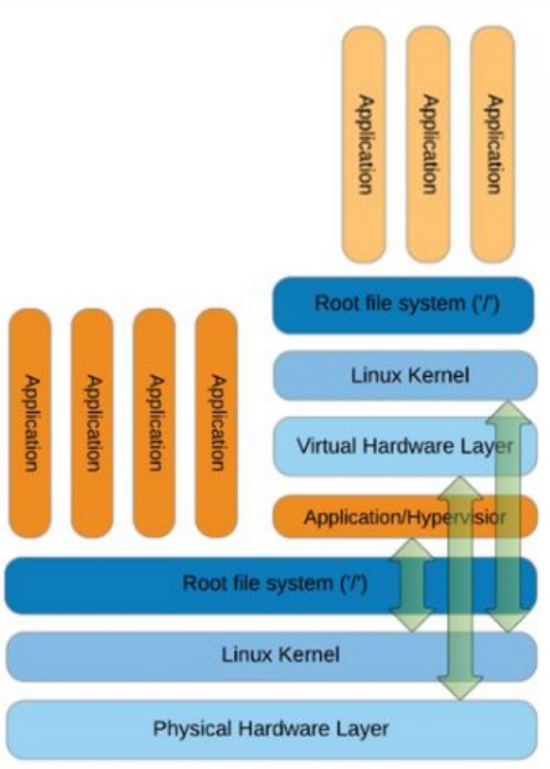
“I want to execute my workload and be done, not consume compute resources while I am not running.” (USENIX LISA 2020)

Library and runtime dependency conflicts

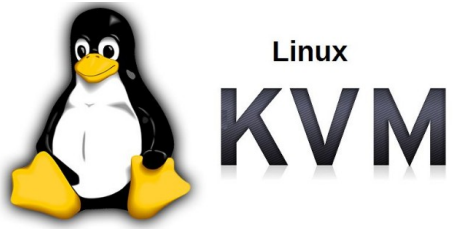


Bare metal and virtualization





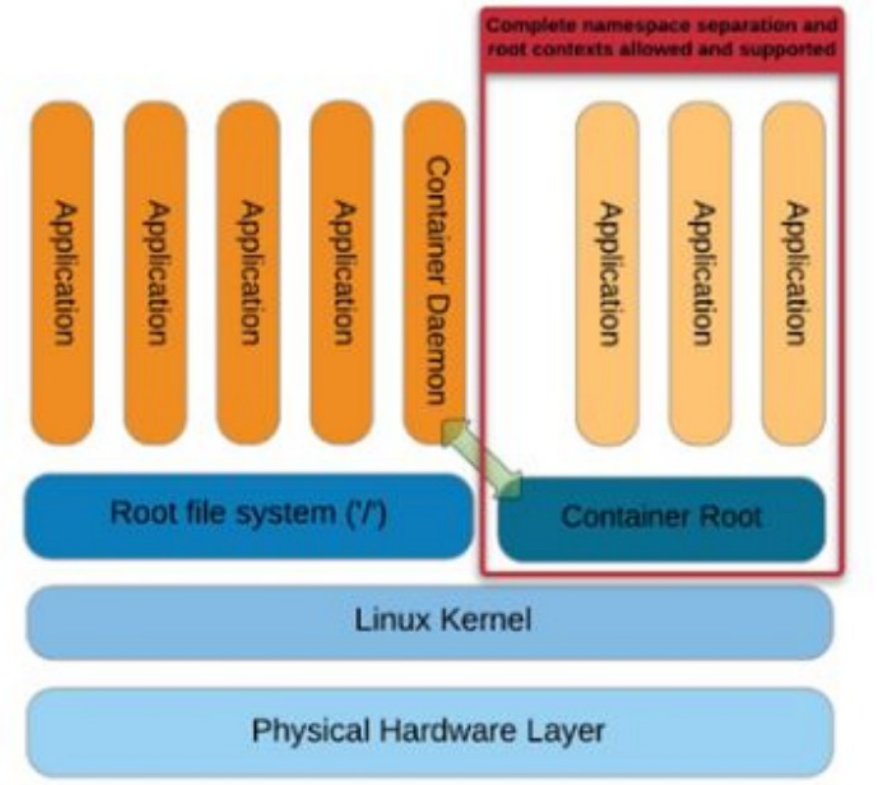
Bare metal and virtualization (2)



openstack®

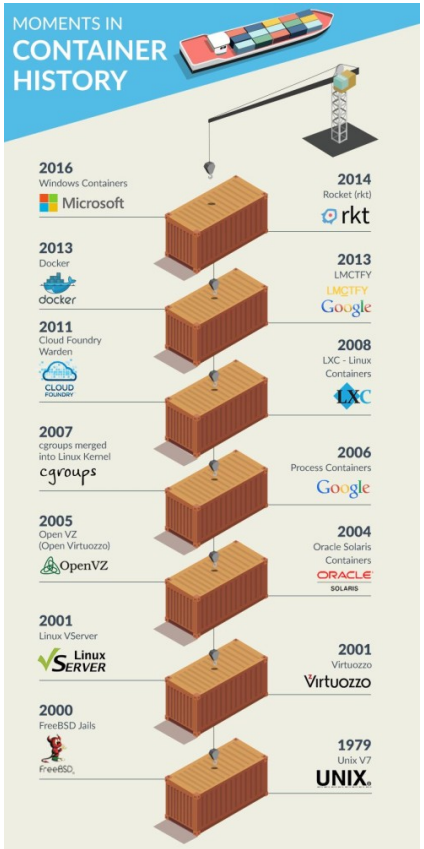
- Does the traditional virtualization architecture scale beyond the limits of a single server?
- Does the traditional virtualization architecture scale to thousands of independent OSes?
- How do the hyperscalers (Google, Amazon, Microsoft) achieve huge scale?

Introducing containers



- Software containers are a a form of virtualization, a very efficient one.
- It removes the entire hypervisor layer
- No need to emulate/run an entire guest operating system anymore.
- We trick the application to think it's the only customer whose needs we have to satisfy
- We only focus on three key technologies that enable us to satisfy runtime and library dependencies:
  - Linux Kernel Namespaces
  - Cgroups
  - Union filesystems

History of software containers



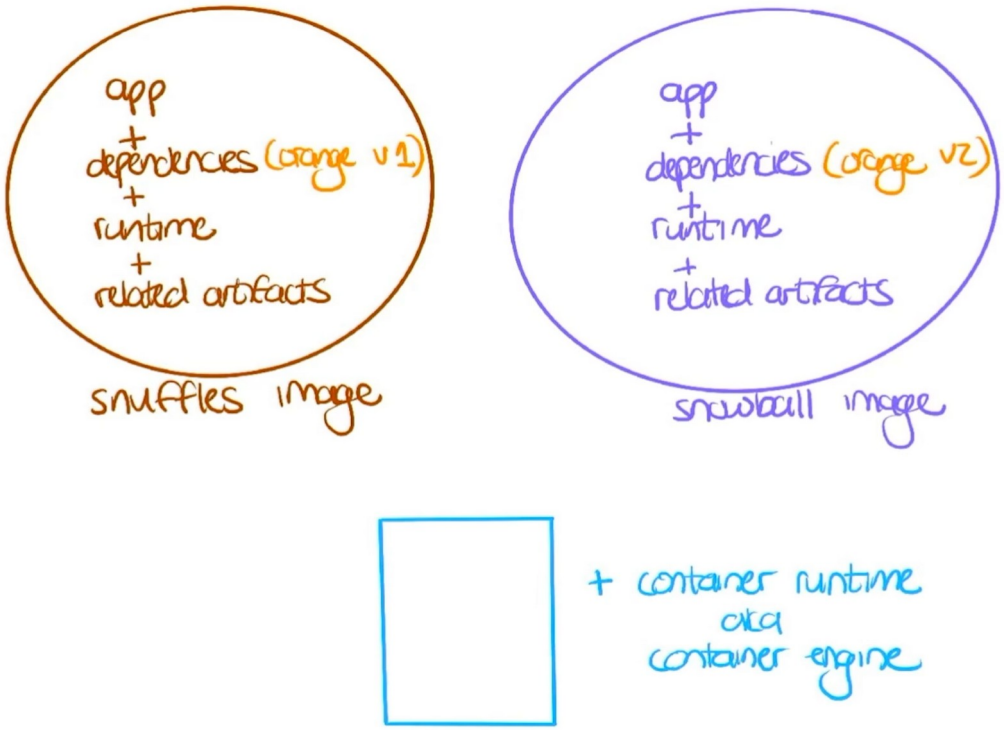
Containers are newer as a technology than traditional full virtualization, but is NOT a new concept.

For the last 10 years they have been used extensively in the industry.

The Open Container Initiative OCI plays a key role in ensuring interoperability amongst different container runtime environments:

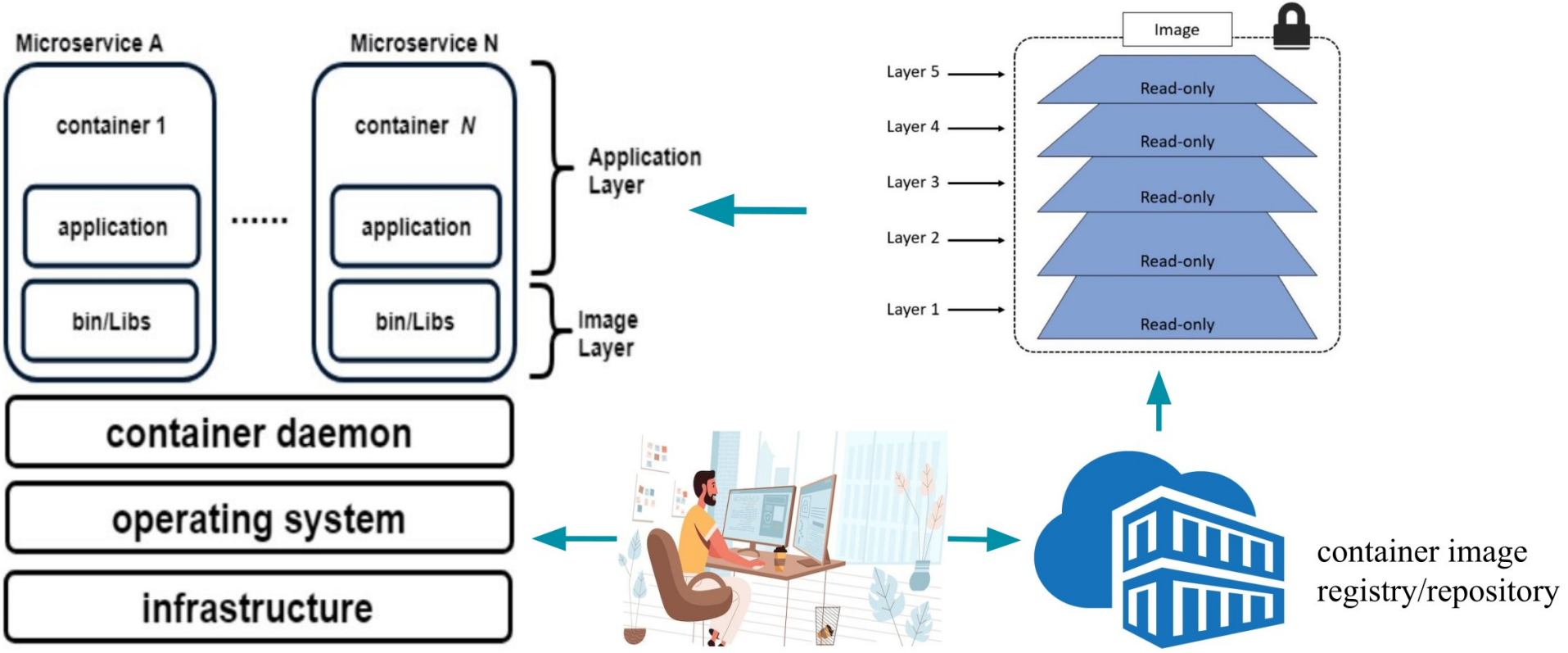
Example: Take a Docker runtime container and run it in a Singularity runtime or a Podman runtime environment.

Software container ecosystem

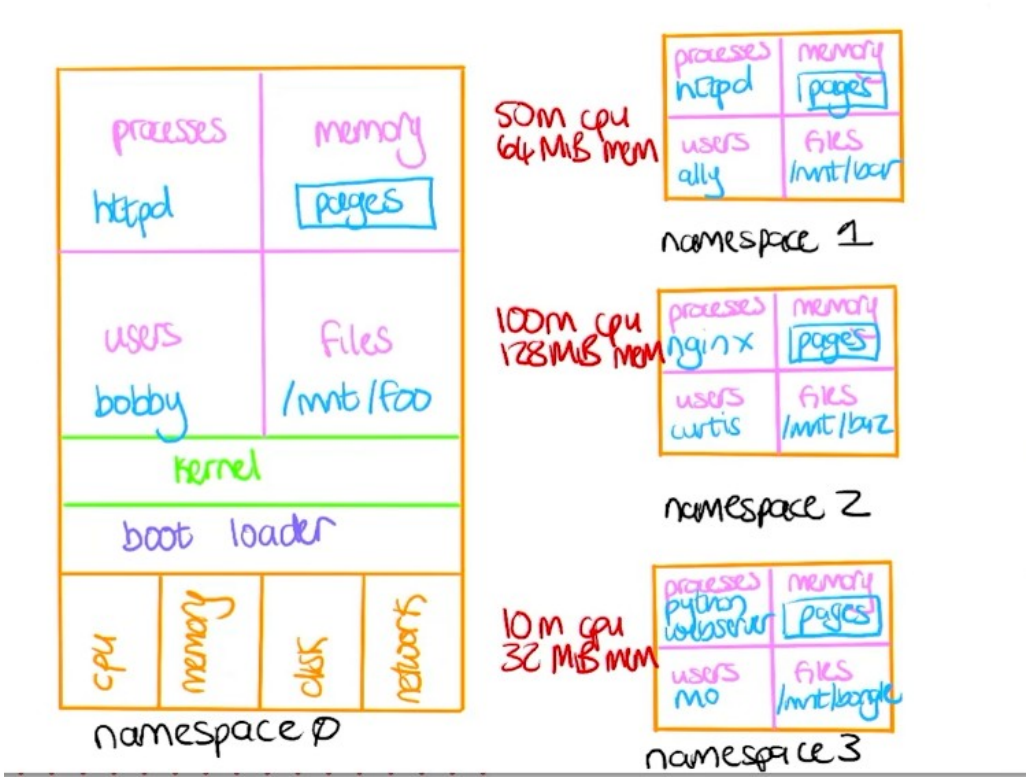


- Image → A blueprint of the container, self sustained, containing all the dependencies for the application.
- Container runtime and engine: The equivalent of the hypervisor (Vmware, KVM…) for the containers (Docker, Podman, Singularity, Apptainer)
- These two can co-exist on the same physical server WITHOUT emulating an extra (guest) Operating System

Software container ecosystem



Namespaces and Cgroups at work



- Each namespace is your application, thinking is the only customer.
- Running with a specific user id
- Cgroups ensure that one namespace cannot take over all the computing resources of the system.

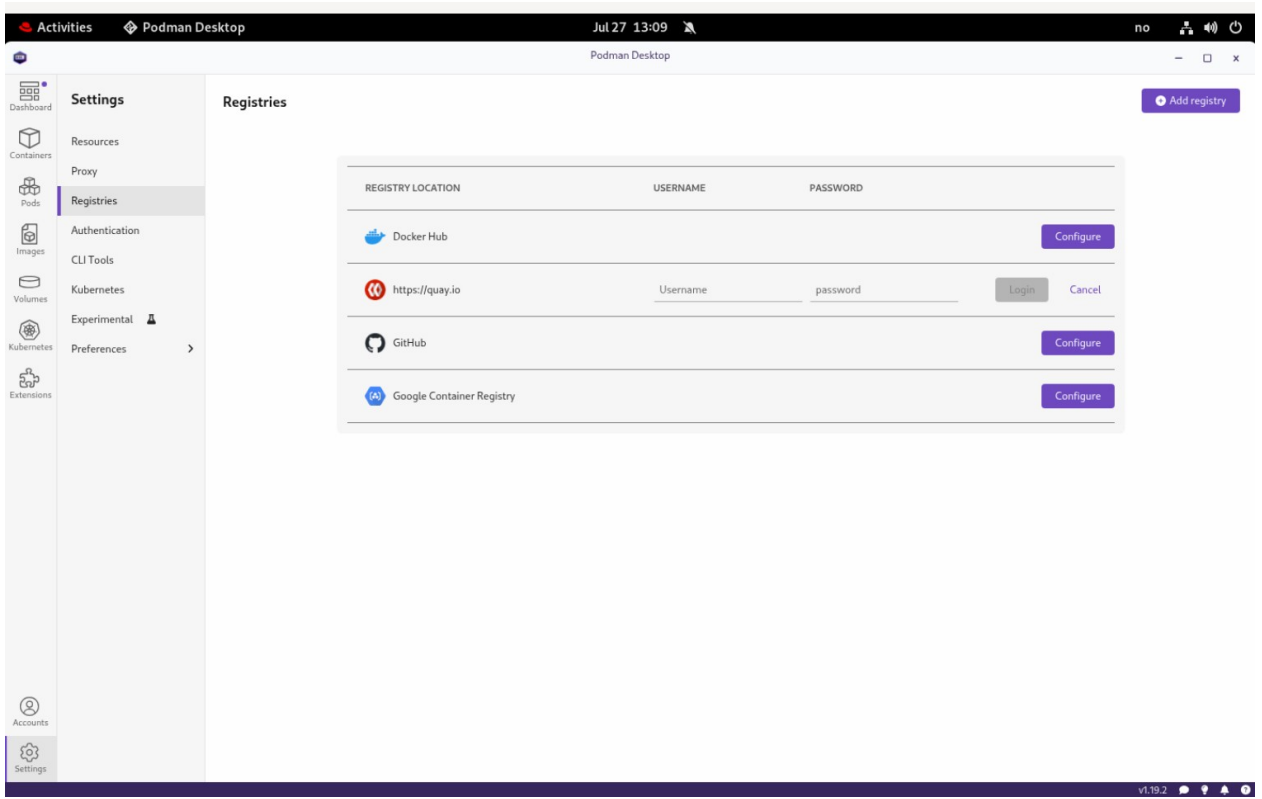
Container registries

- You either contsruct a container from scratch or pull/use a ready made one.
- For pulling a ready made one, you need access to a registry:
  - podman pull *registry.redhat.io/ubi9/ubi:9.1* OR
  - podman run *registry.redhat.io/ubi9/ubi:9.1*
- Some registries are open to pull requests. Others require authentication.

```
[user@host ~]$ podman login registry.redhat.io
Username: YOUR_USER
Password: YOUR_PASSWORD
Login Succeeded!
[user@host ~]$ podman pull registry.redhat.io/rhel9/nginx-120
Trying to pull registry.redhat.io/rhel9/nginx-120:latest...
Getting image source signatures
...output omitted...
```

- Consult also the files /etc/containers/registries.conf and files under /etc/containers/registries.conf.d/

Container registries (2)



If you work with podman desktop, registry administration (with authentication) is easier. Go to:
→Settings (bottom left corner)
→Registries

How to download and install podman desktop:

https://podman-desktop.io/downloads

Making a container from scratch

- Sometimes it is necessary (security/customisation) to create your own container.
- Not currently a requirement for the RHCE/RHCSA exam, so this is extra knowledge, essential for DevOps and Developers
- To build the blueprint for a container (container image), you need to describe how the image is going to be built exactly in a recipe file.
- You will normally start with the base container image of a Linux distribution, then move on to specify the applications and their library dependencies.

Making a container from scratch (2)

```
# Use a base Linux image
FROM ubuntu:22.04

# Set environment variables to avoid interactive prompts
ENV DEBIAN_FRONTEND=noninteractive

# Install R and common dependencies
RUN apt-get update && \
    apt-get install -y --no-install-recommends \
        r-base \
        r-base-dev \
        build-essential \
        libcurl4-openssl-dev \
        libssl-dev \
        libxml2-dev && \
    apt-get clean && \
    rm -rf /var/lib/apt/lists/*

# Set default command
CMD ["R"]
```

- Each directive adds an image layer starting from the top.
- In this example, we build an Ubuntu image that runs the R program.
- Save the listing on the left to a file called Dockerfile in an empty directory
- Then as a normal user, type the following to build the container with name ‘myrcontainer’ :

  podman build -t myrcontainer .

Questions/Ερωτήσεις