RHEL 10 - how to make the most out of your development and testing environment with virtualization (and without)

4. Enterprise Linux Anwendertreffen
Hamburg, 13.3.2025

Joachim von Thadden
Principal Specialist Solution Architect

Intro

There are different ways to use virtualization in RHEL to create a nice

- testing environment or
- development environment.

We will take a short, but practical look on most of them:

- What different ways are there for virtualization in RHEL (not only 10)?
  - cockpit, virt-manager, qemu, SNO/CRC, ovirt?
- How to **not** virtualize (but still have a confined, encapsulated OS with less ressources but all advantages of a VM)
  - an introduction to Incus, former lxc/lxd by Canonical

$ whoami

Joachim von Thadden
EMEA Principal Specialist Solution Architect

- based in Germany, near Düsseldorf
- more than 25 years in IT
- more than 30 years working with Linux
- >10 years experience with OpenStack & OpenShift
- 9 years at Red Hat

Red Hat

Ways of virtualization

How to Virtualize?

- What different ways are there for virtualization in RHEL (not only 10)?
  - A
  - B
  - C
  - D
  - E
  - F

How to Virtualize?

- What different ways are there for virtualization in RHEL (not only 10)?
  - cockpit
  - virt-manager
  - qemu-kvm via virt-install
  - Single Node OpenShift (SNO)
  - OpenShift Local (OSL) aka CoreReady Containers (CRC)
  - ovirt?

Profile: cockpit

- cockpit
  - Install: `dnf install -y cockpit-machines`
  - Enable: `systemctl enable --now cockpit.socket`
  - Use: browse to http://localhost:9090

Demo

Profile: virt-manager

- virt-manager
  - Install: `dnf -y install virt-manager`
  - Config: `sudo usermod -aG libvirt,kvm $USER`
  - Use: `virt-manager`

Demo

Profile: qemu-kvm

- qemu-kvm via virt-install
  - Install: `dnf -y install virt-install`
  - Config: `wget https://cdimage.debian.org/debian-cd/current-live/amd64/iso-hybrid/debian-live-12.9.0-amd64-kde.iso`
  - Use:

```
sudo virt-install \
  --name debian  --ram 2048 --vcpus 2 \
  --disk path=/var/lib/libvirt/images/debian.qcow2,size=20 \
  --os-variant debian11 --network bridge=virbr0 --graphics vnc \
  --cdrom ./debian-live-12.9.0-amd64-kde.iso
```

Demo

Profile: SNO

- Single Node OpenShift (SNO)
  - Docu: https://docs.redhat.com/en/documentation/openshift_container_platform/4.18/html/installing_on_a_single_node/install-sno-installing-sno
  - Install via console.redhat.com: https://console.redhat.com/openshift/assisted-installer/clusters/~new
  - ```
    sudo virt-install \
      --name sno  --ram $((1024*48)) --vcpus 23 \
      --disk path=/var/lib/libvirt/images/sno-vda.qcow2,size=100 \
      --disk path=/var/lib/libvirt/images/sno-vdb.qcow2,size=100 \
      --os-variant fedora-coreos-stable --network bridge=virbr0 \
      --graphics vnc --cdrom discovery_image_sno.iso
    ```
    ```

Demo

Profile: CRC

- OpenShift Local (OSL) aka CoreReady Containers (CRC)
  - Docu: https://www.redhat.com/en/blog/install-openshift-local
  - Install: `crc setup`
  - Config: `crc config set cpus 8; crc config set memory 32768; crc config set disk-size 100;crc config view`
  - Start: `crc start -p ~/pull-secret.json`
  - Use:
    `eval $(crc oc-env) && oc login -u kubeadmin https://api.crc.testing:6443`
  - Or: browse `https://console-openshift-console.apps-crc.testing`

Demo

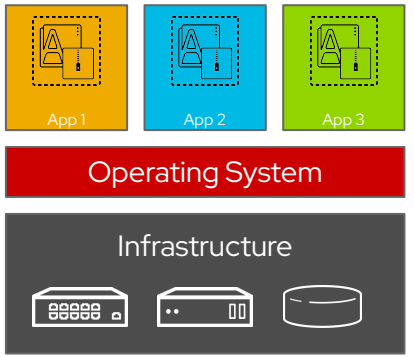
Containers

vs

Virtual Machines

Containers are not virtual machines

- Containers are process isolation
- Kernel namespaces provide isolation, selinux provides security and cgroups provide resource controls
- No hypervisor needed for containers
- Contain only binaries, libraries, and tools which are needed by the application
- Ephemeral

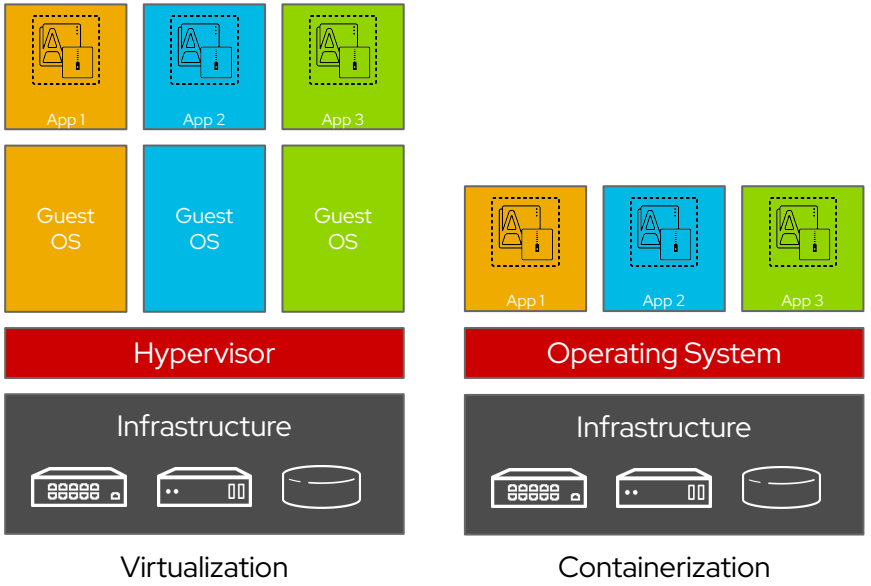


Containerization

Red Hat

Containers are not virtual machines

- VMs are simulated HW
- Kernel hypervisor provide isolation, svirt (selinux) provides security and cgroups provide resource controls
- HW backed hypervisor on Baremetal machines needed
- Running complete Operating Systems for general purpose workloads
- Persistent



Virtualization

Containerization

What-If

So what if we would be able to run a complete OS in a container and use it like a VM?

- starting VMs
- running Containers
- testing Software
- using a normal Desktop GUI

BUT

- lightweight as a container is
- without the memory restrictions of a VM
- without the CPU restrictions of a VM
- but still confined as much as possible

Introducing Incus (former lxd)

Demo

Incus

- based on kernels lxc subsystem
- initially started as lxd daemon by Canonical
- daemon system that allows for an easier consumption of lxc
- forked because of Canonical changing the rules...

But what is it?

- a container technology
- allows for full operating system containers
- uses cgroups and selinux (like VMs) to separate
- allows for fully privileged access
- can run all of the technologies, we have just seen within the boundaries of a container

Profile: Incus

Install

```
# on RHEL 9 use the following
sudo subscription-manager repos --enable codeready-builder-for-rhel-9-$(arch)-rpms
sudo dnf install https://dl.fedoraproject.org/pub/epel/epel-release-latest-9.noarch.rpm

sudo dnf config-manager --enable crb
sudo dnf -y copr enable neil/incus

# on fedora 41 there is already a native package
sudo dnf -y install incus-tools

# start incus
sudo systemctl enable incus --now

# initial configuration
incus admin init
```

Profile: Incus

Run

```
# start, stop and remove
incus launch images:fedora/41 f41               # download and start an OS container, use -vm to start a VM
incus start f41
incus stop f41
incus delete f41

# if ssh is not there or there is no password or ssh key set, push a file or enter the container directly
incus file push ~/.ssh/id_ed25519.pub f41/root/.ssh/authorized_keys
incus exec f41 bash

# find images
incus image list                    # local
incus image list images:      # online
incus image delete <image>

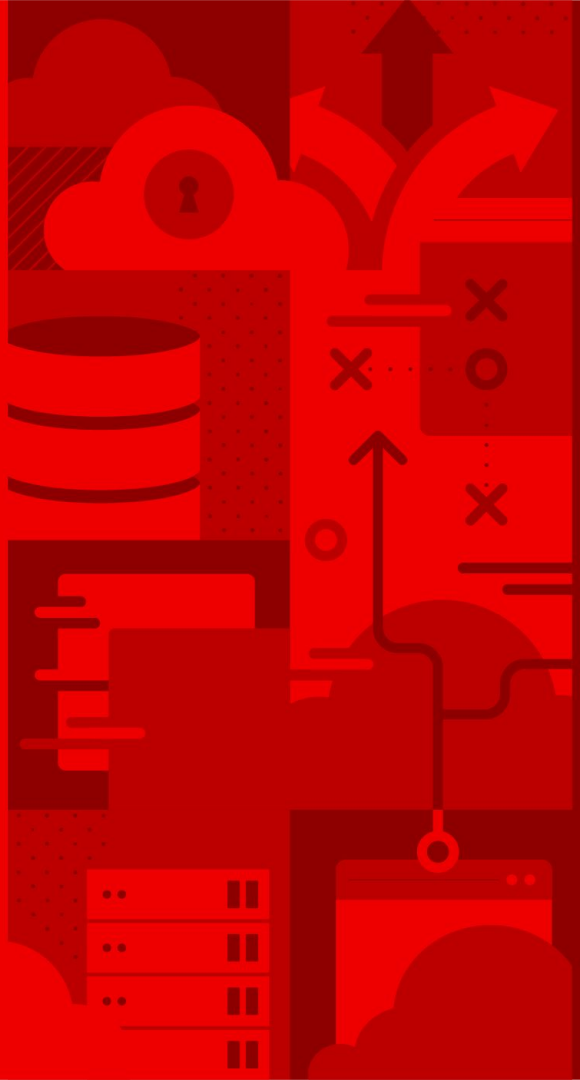
# set some constraints
incus config set f41 limits.cpu=8
incus config set f41 limits.memory=8GiB
incus config set f41 limits.memory.enforce=soft
```

Profile: Incus

Configure privileged Incus container

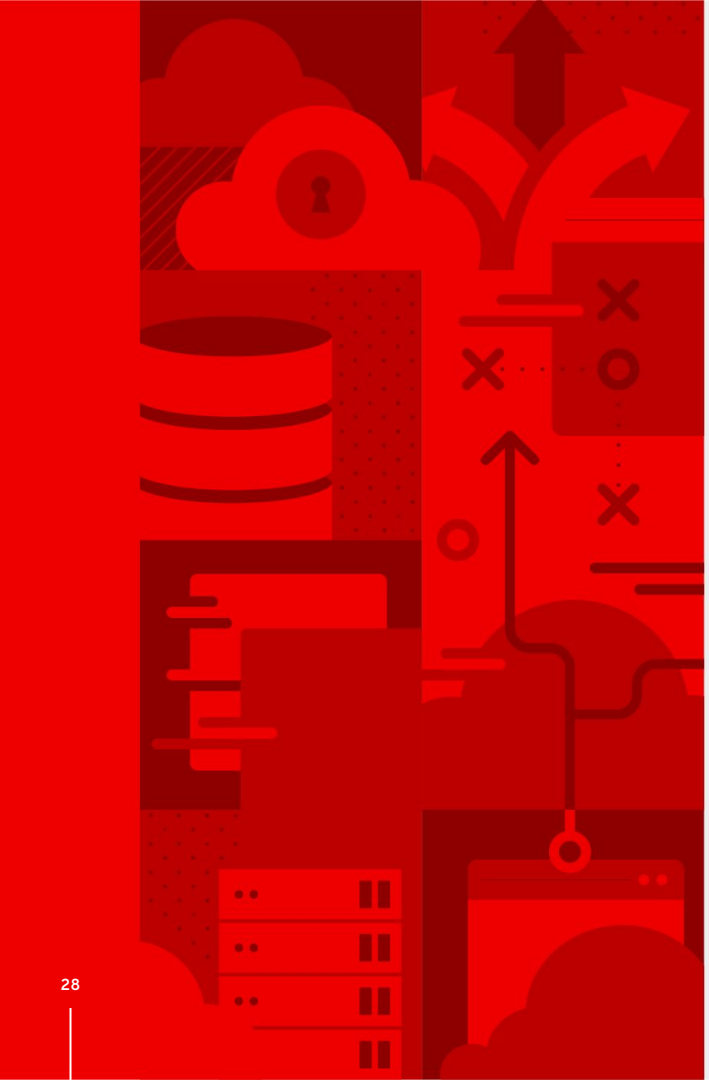
```
# if this container needs mknod and other privileged permission
incus config set $CONTAINER security.privileged="true"
incus config set $CONTAINER security.nesting="true"
echo -e "lxc.cgroup.devices.allow = a\nlxc.mount.auto=proc:mixed sys:mixed cgroup-full:mixed\nlxc.cap.drop =" | \
  incus config set $CONTAINER raw.lxc -
incus config set $CONTAINER raw.idmap "both 0 0"
incus config device add $CONTAINER loop-control unix-char path=/dev/loop-control major=10 minor=237 mode=660
for n in {0..20}; do incus config device add $CONTAINER loop$n unix-block path=/dev/loop$n major=7 minor=$n
mode=660; done

# if you want virtualization/crc, use instead
incus config device add $CONTAINER tun unix-char path=/dev/net/tun major=10 minor=200 mode=666
incus config device add $CONTAINER kvm unix-char path=/dev/kvm major=10 minor=232 mode=666
incus config device add $CONTAINER vhost-net unix-char path=/dev/vhost-net major=10 minor=238 mode=600
incus config device add $CONTAINER vsock unix-char path=/dev/vsock major=10 minor=16 mode=666
incus config device add $CONTAINER vhost-vsock unix-char path=/dev/vhost-vsock major=10 minor=241 mode=666
incus config device add $CONTAINER vhost-net unix-char path=/dev/vhost-net major=10 minor=238 mode=666
# don't do: incus config device add $CONTAINER selinux disk source=/sys/fs/selinux path=/sys/fs/selinux
```



Q&A

Red Hat



Thank you

Red Hat is the world's leading provider of enterprise open source software solutions. Award-winning support, training, and consulting services make Red Hat a trusted adviser to the Fortune 500.

linkedin.com/company/red-hat

facebook.com/redhatinc

youtube.com/user/RedHatVideos

twitter.com/RedHat

Red Hat