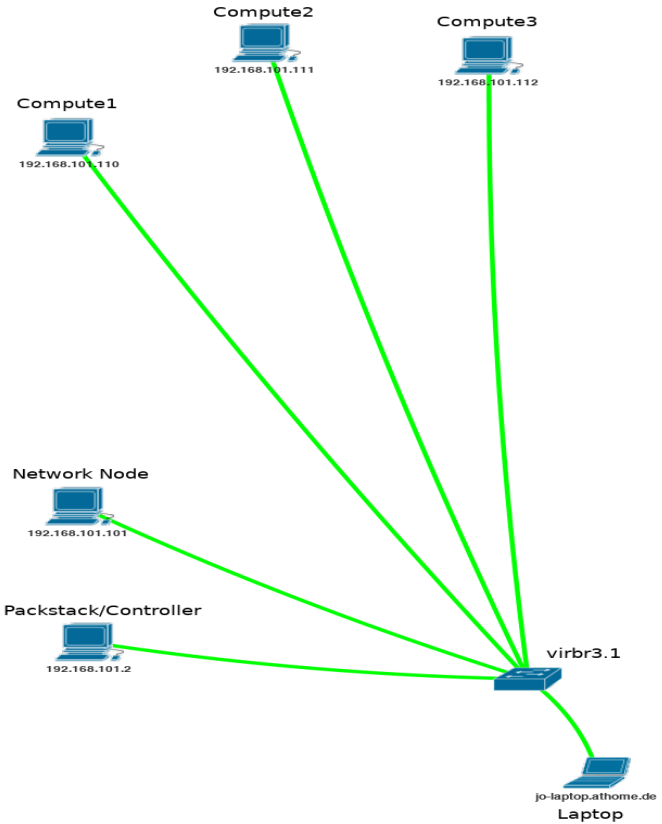
Containerize your life!

How to live with always not enough resources

Joachim von Thadden
EMEA Senior Specialist Solution Architect OpenStack
Thu 24.5.2018, 1:50pm - 3:20pm

Demo


Compute2
192.168.101.111
Compute3
192.168.101.112
Compute1
192.168.101.110
Network Node
192.168.101.101
Packstack/Controller
192.168.101.2
virbr3.1
jo-laptop.athome.de
Laptop

AGENDA

The Promise

How to Virtualize without Virtualization?

The culprits of misusing a Container

Let’s get our fingers dirty...

First Steps with LXD

And now the heavy stuff!

First Steps with libvirt

OpenStack: Install Packstack

Disclaimer

Warning

… this is not a traditional lab!

This Lab differs from others! It is not meant to be a guided tour through a technology! See it more as an experimental session to figure out the borders of technology…

So expect failures, questions and “agile customization”!

And: You **must not** copy+paste blindly as you will have to adapt commands to your environment!

Instead of an USB Stick...

Content URL

The slides can be found under the following URL:
https://github.com/jthadden/OpenStack_Summit_2018_Vancouver/doc

The Promise

You can (mis-)use a Container as a full blown Virtual Machine!
Even better: You can virtualize without nesting!

The Culprit of Virtualization

The reason why we don’t like it if we are not an enterprise!

- Resources
- Resourcessss
- Resourcesssssss

OpenStack

... is a Whale!

A full blown OpenStack consists of at least

- 1 Installer (Director or other) (16GB RAM, 4CPUs, 100GB Disk)
- 3 Controllers (>12GB RAM, >4 CPUs, 100GB Disk)
- >2 Computes (>4GB RAM, >4 CPUs, 100GB Disk)
- Some Storage (e.g. for live migration) (>4GB RAM, 2 CPUs, >100GB Disk)

At least 60GB RAM, 30 CPUs, 700GB Disk.

I can do it with 24GB RAM, 2 CPUs + HT, 100GB Disk :-)

How to Virtualize without Virtualization?

How to Virtualize without Virtualization?

Some type of Container is your friend!

- chroot
- Container
- Other Ideas?

Useful Container Technologies...

... and why we can't or don't want to use them (and I tried them all!)

I tried them all:

- Chroot
  - lacking of namespace separation and no own networking stack (because of the former)
- Docker
  - working but problems with (needed) privileged actions
- Legacy OpenVZ (vzctl)
  - does the job, but you might need to use the upstream package
  - not supported very well
- Lxc
  - does the job, but horrible to configure and use
- Systemd-nspawn
  - nested virtualization is a problem

The best way today

- and why we have to cry

LXD - the LXC Daemon

- It’s actually a daemon for LXC.
- Coming from Canonical - so will probably die when Shuttleworth dies
  - BUT it's OpenSource! So who cares!
- LXD is based on LXC and that's running out of support of Red Hat (and might be removed in the future)
  - Never mind, we will use Fedora…

If you don't like the LXD appraoch

- you can do it as well with Legacy OpenVZ (install vzctl-core)

The culprits of misusing a Container

Containers have several restrictions

- no nested selinux
- restricted systemd
- no auditd
- no sysctl
- no module loading
- problems with subscribing
- error with iscsi
  - open bug since YEARS now with missing namespacing in scsi-modules

- Probably more you will face!

And many of them can not even been circumvented by running the container privileged!

BUT:
so many things are working like a charm

- Booting :-)
- 99% networking
- firewalling
- namespaces
- virtualization
- 99.9% feeling of real machine

Let’s get our fingers dirty…

Need a VM?

There is a VM-Image of an already installed and configured Fedora-26 VM (qcow2):
http://<IP>/isos/Fedora-26-RHTE.qcow2

- make one with at least 8GB, 2CPU, 80GB HDD based on Fedora 26, copy CPU config
- start the VM and install Fedora 26
  - ➔ on other distros see: https://linuxcontainers.org/lxd/getting-started-cli/
- install needed Software

```
dnf -y upgrade
dnf -y groupinstall cloud-management
dnf -y install libguestfs-tools
```

Enabling LXD

```
dnf -y copr enable ganto/lxd           # use ganto/lxd3 for Fedora28 on
dnf -y install lxd lxd-client lxd-tools

getent group lxd > /dev/null || groupadd -f -r lxd
echo "root:1000000:65536" >> /etc/subuid
echo "root:1000000:65536" >> /etc/subgid
setenforce 0
systemctl start lxd.service
usermod --append --groups lxd rhte
```

To make it permanent (e.g. in a special VM):

```
sed -i -e "s/SELINUX=enforcing/SELINUX=permissive/" /etc/selinux/config
systemctl enable lxd.service
```

LXD can be used as a Server daemon

E.g. if you have some stronger machine in the cellar.

Add the following:

```
lxc config set core.https_address "[::]"
lxc config set core.trust_password <your PW>
firewall-cmd --add-port=8443/tcp --permanent
firewall-cmd --reload
```

Add some storage

There are numerous storage options like

- ZFS
- Btrfs
- LVM
- Plain Filesystem <- we will use that

Re-login as normal user rhte and execute:

```
lxc storage create default dir
lxc profile device add default root disk path=/ pool=default
```

First Steps with LXD

Start your first lightweight VM

All lxc commands should be issued as user rhte!

```
lxc image list images:|grep -i centos     # remote repository images
lxc image list                            # local available images
lxc launch images:centos/7 my-centos7-container
lxc exec my-centos7-container /bin/bash
lxc list [regexp]                         # to list containers with state
```

Some more Commands

```
lxc info my-centos7-container                  # getting Info
lxc {network|profile|config|storage} list
lxc {network|profile|config|storage} show
lxc config show compute0 --expanded            # getting all configs
                                               # (like image, storage, security"
ps aux | grep "containers compute0"            # find the monitoring process of a
                                               # container (e.g. if hanging)
lxc list security.privileged=true              # list all privileged containers
lxc start|stop|restart|pause
lxc file pull|push|edit <container>/<path> [-] # - means stdout

lxc config device add mycontainer vartest disk source=/var/www path=/var/test
                                              # attach a dir to a container
```

Prepare a CentOS image for our special usage with LXD

- learn here the easy way
  - to do the things which are not working in an LXD-Container

- we need
  - a profile
  - the CentOS image in lxc (you know how to get it)
  - a rc.local file: https://raw.githubusercontent.com/jthadden/OpenStack_Summit_2018_Vancouver/master/roles/layer2_bootstrap_container/files/rc.local-container

Create a new Profile

Remember: user is always rhte!

```
lxc profile create rhosp
lxc network attach-profile virbr0 rhosp eth0
lxc profile device add rhosp root disk path=/ pool=default
lxc profile set rhosp security.privileged true
```

rc.local

→ let us go through it, to understand what to do ←

on the git repo go to:
roles/layer2_bootstrap_container/files/rc.local-container

Prepare your CentOS with the rc.local

```
lxc image list
lxc launch -p rhosp images:centos/7 my-centos7-container
lxc exec my-centos7-container /bin/bash
# install and start sshd
yum -y install openssh-server file
systemctl enable sshd && systemctl start sshd
passwd root
# and try to sshd into it
lxc list
ssh root@<IP>
# get rc.local and execute
curl <...> -o /etc/rc.local
chmod 755 /etc/rc.local
systemctl enable rc-local
reboot
# look into logfile for execution of rc.local
```

Let’s try some real work!

Install an IPA

Prepare the Container

Set hostname, enable repos and install firewalld

```
# ssh into the container
hostnamectl set-hostname `hostname`.rhte.internal

yum repolist
yum install firewalld
systemctl enable firewalld && systemctl start firewalld
```

Install and Configure IPA

```
yum -y install ipa-server ipa-server-dns
ipa-server-install --setup-dns --forwarder=8.8.8.8 -r \
`dnsdomainname|tr a-z A-Z` -p "changeme" -a "changeme" -U 2>&1 | \
tee ipa-install.out--setup-dns

# open the firewall accordingly
firewall-cmd --permanent \
--add-port={80/tcp,443/tcp,389/tcp,636/tcp,88/tcp,464/tcp,53/tcp,88/
udp,464/udp,53/udp,123/udp}
firewall-cmd --reload
```

Test IPA

```
kinit admin
klist
ipa user-find admin
```

1. if you want to see the webinterface, use on your host something like

```
sshuttle -r root@<yourVMsIP> <yourContainersIP>
```

2. add the host to your /etc/hosts

```
echo "<IPofContainer> my-centos7-container.rhte.internal" >> /etc/hosts
```

3. Browse to your IPA

```
https://my-centos7-container.rhte.internal
```

Now try out VMs in Containers!

Install libvirt

```
yum -y install libvirt libvirt-daemon-kvm
systemctl enable libvirtd
systemctl start libvirtd
```

Try out libvirt in a container

Note: You should always ssh -AX to your host…

Attention: The following is done on the VM/your host, not in the container!

Map an image into the Container:

```
curl http://download.cirros-cloud.net/0.3.5/cirros-0.3.5-x86_64-disk.img \
-o cirros-0.3.5-x86_64-disk.img

lxc config device add my-centos7-container image disk \
source=`pwd`/cirros-0.3.5-x86_64-disk.img \
path=/var/lib/libvirt/images/cirros-0.3.5-x86_64-disk.img
```

Start virt-manager:

- Either install in the VM or use the sshuttle connection from the IPA test on your host.
- Add a new connection to your Container via ssh.
- Add a new VM, select the Image and start it.
- Look at your Host and Container for the VM: pgrep -af kvm|grep kvm .

You might want to stop this container now!

And now the heavy stuff!

If OpenStack is working on that, then everything will do!

The git repo has the code for an ansible install of RDO in it:

https://github.com/jthadden/OpenStack_Summit_2018_Vancouver.git

git it and start it

Note: You should always ssh -AX to your host…
We are now working of the VM/Host again…

```
yum -y install git ansible libselinux-python python-firewall
git clone https://github.com/jthadden/OpenStack_Summit_2018_Vancouver.git
cd OpenStack_Summit_2018_Vancouver/
# add a host entry for localhost (see config_infrastructure.yml)
echo "<yourVMorHostIP> myserver" >> /etc/hosts
# when working with a VM: generate an ssh key or copy it over from your host:
ssh-keygen && ssh-add
./create.sh

# be patient...
```

playbook usage

- to stop the containers use:
```
ansible-playbook -v -i hosts-OSSummit-minimal -e @config/config_OSSummit_minimal.yml -e
@config/config_infrastructure.yml stop.yml
```
- to start again the containers use:
```
ansible-playbook -v -i hosts-OSSummit-minimal -e @config/config_OSSummit_minimal.yml -e
@config/config_infrastructure.yml start.yml
```
- to destroy the cloud use:
```
ansible-playbook -v -i hosts-OSSummit-minimal -e @config/config_OSSummit_minimal.yml -e
@config/config_infrastructure.yml destroy.yml
```
- to install an extended version with 3 computes and a network node use:
```
ansible-playbook -v -i hosts-OSSummit -e @config/config_OSSummit.yml -e
@config/config_infrastructure.yml create.yml
```

Please provide feedback.
Please clone.

Thank you!