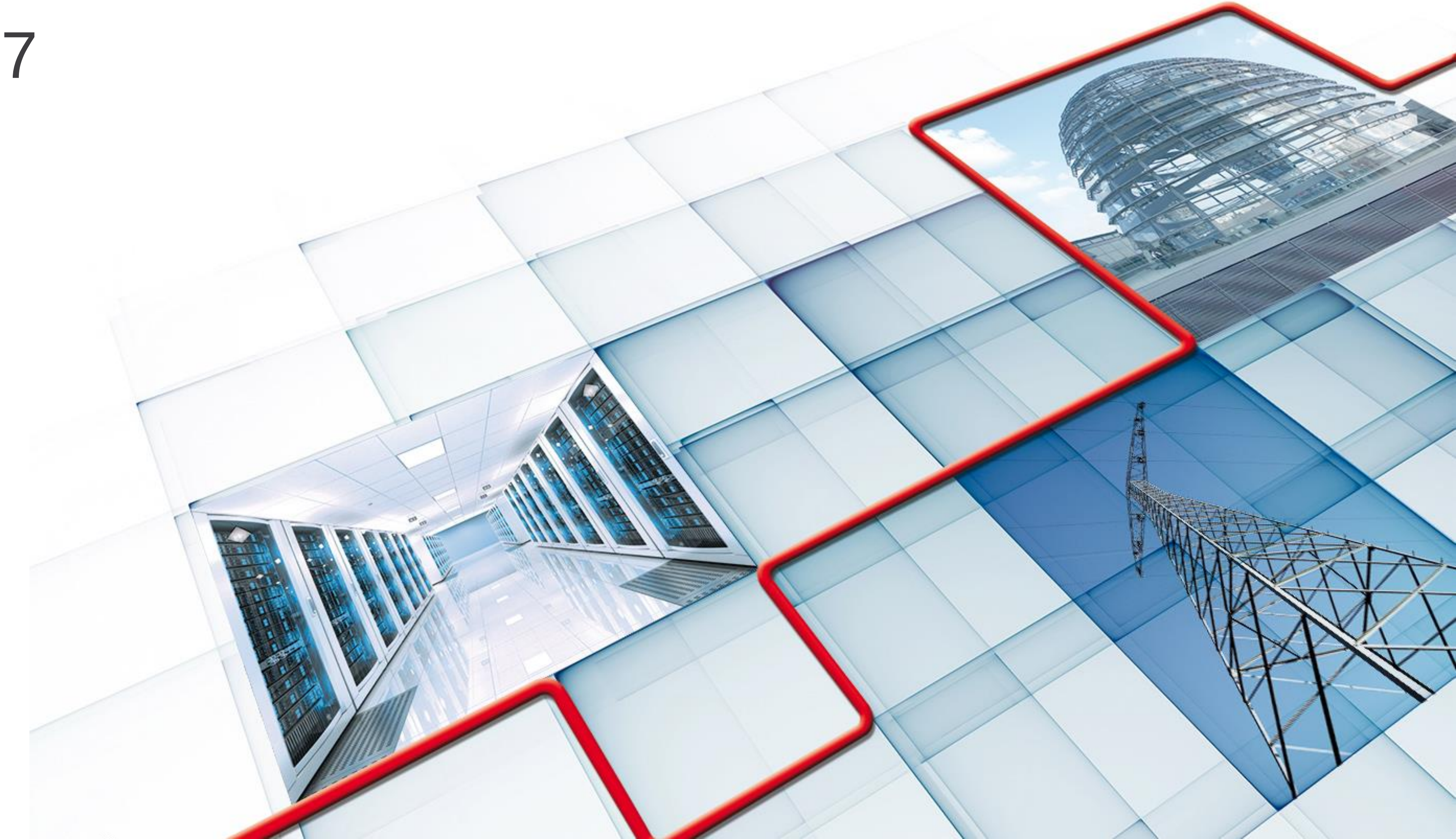
IT. Beyond default. GISA®

Ganeti usage @GISA

Mini GanetiCon 2017

Sascha Lucas
Leipzig, 15.12.2017



Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

Introduction

GISA

- owning two datacenters, located in germany, ISO27001 and BSI certified
- IT outsourcing provider
  - managed services: complete stack of infrastructure, servers, OS, applications (i.e. SAP, no Ganeti yet ☺)
  - or reduced stack: manage parts (i.e. the application) on your own
  - 24x7 operation / service desk / everything in-house / from a single source (no subcontractors)
  - historic background: energy industry and public sector

Introduction

GISA

- owning two datacenters, located in germany, ISO27001 and BSI certified
- IT outsourcing provider
  - managed services: complete stack of infrastructure, servers, OS, applications (i.e. SAP, no Ganeti yet ☺)
  - or reduced stack: manage parts (i.e. the application) on your own
  - 24x7 operation / service desk / everything in-house / from a single source (no subcontractors)
  - historic background: energy industry and public sector

me, myself and I

- 18+ years linux admin, 5+ years @GISA
- since debian lenny (ganeti-1.2) „Ganeti included“
- sascha.lucas@gisa.de

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

some stats

- 5 clusters (two bigger, two 2-node, a 2-node test-cluster)
- SLES based (like RHEL, but from Novell)
  - Ganeti-2.9 and 2.14 build from source for SLES-11 and SLES-12
  - from source means: latest release + “diff to git stable-2.X head”

some stats

- 5 clusters (two bigger, two 2-node, a 2-node test-cluster)
- SLES based (like RHEL, but from Novell)
  - Ganeti-2.9 and 2.14 build from source for SLES-11 and SLES-12
  - from source means: latest release + “diff to git stable-2.X head”

what	„big“ cluster #01	„big“ cluster #03
number of nodes	12	14
number of instances	132	53
total vCPUs assigned to instances (be/vcpus)	636	491
total RAM assigned to instances (be/memory)	2407GiB	9166.4GiB
total disk space assigned to instances (disk.sizes)	19269.6GiB	47898.6GiB

our setup

- typical node hardware
  - 2x 10G ethernet (LACP/bonding)
  - 12 Core blade, 256GB RAM → sharedfile (NFS)
  - 16 Core rackmount, 512GB RAM, 16x1.2TB disk → DRBD
  - 40 Core rackmount, 1536GB RAM → blockdev (FC-SAN)

our setup

- typical node hardware
  - 2x 10G ethernet (LACP/bonding)
  - 12 Core blade, 256GB RAM → sharedfile (NFS)
  - 16 Core rackmount, 512GB RAM, 16x1.2TB disk → DRBD
  - 40 Core rackmount, 1536GB RAM → blockdev (FC-SAN)
- KVM hypervisor
  - `security_model: pool` (using grnet‘s nss-uidpool) and `use_chroot: True`
  - `disk_cache: none`
  - `migration_bandwidth: 625` (=5Gb/s) and `migration_downtime: 300`
  - no `kernel_path` (Bootloader+Kernel/initrd from inside VM)
  - `serial_console: True`
  - `vhost_net: True` and `vnet_hdr: True`

our setup (continued)

- multi cluster: `mac prefix: aa:00:xx` (where xx=cluster number)

  @VM

```
$ ip l l dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether aa:00:03:86:85:ab brd ff:ff:ff:ff:ff:ff
                     ^^
               VM is at cluster 03 (handy for dynamic grouping with ansible)
```

our setup (continued)

- multi cluster: `mac prefix: aa:00:xx` (where xx=cluster number)

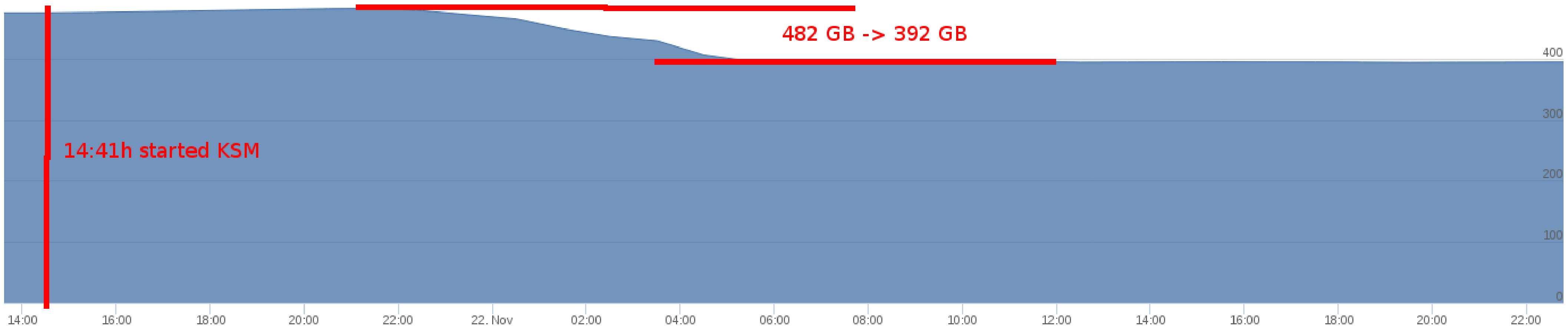
  @VM

```
$ ip l l dev eth0
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP mode DEFAULT group default qlen 1000
    link/ether aa:00:03:86:85:ab brd ff:ff:ff:ff:ff:ff
                     ^^
               VM is at cluster 03 (handy for dynamic grouping with ansible)
```

- DRBD
  - `net-custom: --max-buffers 8000 --max-epoch-size 8000` (sufficient for 160MB/s)
  - `disk-barriers: bf` and `meta-barriers: True` (use only with battery backed write cache)
  - `dynamic-resync: False und disk-custom: --c-plan-ahead 0` (disable dynamic resync)
  - `resync-rate: 150000` (~0.5TB/h)
  - the `noop` I/O scheduler on the host and guest
- OS-Interface: based on libguestfs (later more)
- bridged networking (VLANs)

memory oversubscribtion (KSM)

- `echo 1 > /sys/kernel/mm/ksm/run` is sufficient
  - fedora has ksm-tuned: will run KSM only and more aggressive when node memory is exhausted
  - example 512GB node:



- even more possible with: compressed swap (zswap?), and swap to SSD

current conversions (cluster #01)

- SLES → Ubuntu, because:
  - ganeti und DRBD is included

    Have you noticed that Ganeti is no longer in Ubuntu? It vanished with artful / 17.10.

    ???
  - no subscription fee
  - enough Ubuntu workloads

- sharedfile → DRBD, because:
  - better cluster scale-out (add or remove nodes to scale disk capacity and performance)
  - the only way for storage migration (change old hardware with new one)

past conversions (cluster #03: LXC → KVM)

past goal: LXCs as VMs → will work, but limitations and additional effort doesn’t outweigh benefits

- benefits: near bare metal performance (network, memory bandwidth)
- limitations: isolation frameworks are diverse/complex or incomplete (cgroups, capabilities, namespaces, …)
- additional effort
  - work around limitations: (NFS mounts, Ganeti + additional disks)
  - understand apparmor (i.e. for mounting NFS inside LXC)
  - like in a VM you need an init-system (something which starts your services and your tools / your costumers can handle)
  - sysv-init (systemd?) fails without CAP_SYS_ADMIN → fix system initialization scripts (mostly disable)

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

building Ganeti from source

how to get the right Haskell requirements (compiler, libraries)

- download an install the right version of the Haskell platform (suiting to the desired Ganeti version):

  https://www.haskell.org/platform/prior.html

  2014 (7.8.3) for Ganeti 2.14

  ?2015 (7.10.3) for Ganeti 2.15+?

- install all needed Haskell libraries:

```
cabal install --only-dependencies cabal/ganeti.template.cabal --flags="confd mond metad"
```

- meet python dependencies / configure / make / install

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

Options for the OS interface

- popular OS interfaces: ganeti-instance-debootstrap, ganeti-os-noop, snf-image

- What’s wrong with ganeti-instance-debootstrap?
  - it runs in the context of the node/kernel and is doing potentially “bad things” like mkfs, sfdisk, grub-install etc.

- snf-image does it right: encapsulate the instance customisation inside a (helper) VM
  - in cloud environments you want to enable the user to further customize the VM (user scripts etc.)
  - but snf-image is bound to debian based distros, what to do with SLES?

- my solution: libguestfs (part of libvirt), 5 years ago
  - essentially it is a qemu-VM with an API
  - many bindings (shell interface, python, C, …)
  - should work on any distro (SLES and Ubuntu confirmed)

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
dd if=/path/to/NFS/${OS_VARIANT}.img \
           | dd of=${DISK_0_PATH}
```

Or

```
curl ${URL}?os=${OS_VARIANT} \
           | dd of=${DISK_0_PATH}
```

Or

```
qemu-img convert ${URI} ${DISK_0_PATH}
```

THIS IS Node CONTEXT !

If `${DISK_0_PATH}` is wrong, you lose.

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
eval $(/usr/bin/guestfish --listen -a ${DISK_0_PATH} -i)
```

again:

THIS IS Node CONTEXT !

If `${DISK_0_PATH}` is wrong, you lose.

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
# the disk with /
rootdev=/dev/sda1 or rootdev=/dev/vg/root

distro="$(${GFR} -- inspect-get-distro ${rootdev})"
major_v="$(${GFR} -- inspect-get-major-version \
        ${rootdev})"
```

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
for i in $(${GFR} -- glob-expand /etc/ssh/ssh_host_'*')
do
        ${GFR} -- rm ${i}
done
```

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
upnr=/etc/udev/rules.d/70-persistent-net.rules
tmp="$(${GFR} -- is-file ${upnr})"
if [[ "${tmp}" = "true" ]]; then
      ${GFR} -- rm ${upnr}
fi
```

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
case ${distro} in
ubuntu)
  ${GFR} -- write /etc/hostname "${INSTANCE_NAME%%.*}";;
sles)
  ${GFR} -- write /etc/HOSTNAME "${INSTANCE_NAME}";;
esac
```

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

```
netcfg=$(mktemp)

if [[ ${distro} = ubuntu ]]; then

cat << EOF > ${netcfg}
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
  address ${NIC_0_IP}
  netmask ${NIC_0_NETWORK_SUBNET##*/}
  gateway ${NIC_0_NETWORK_GATEWAY}
EOF

${GFR} -- upload ${netcfg} /etc/network/interfaces

fi
```

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

```
netcfg=$(mktemp)
if [[ ${distro} = sles ]]; then
        cat << EOF > ${netcfg}
BOOTPROTO='static'
BROADCAST=''
ETHTOOL_OPTIONS=''
IPADDR="${NIC_0_IP}/${NIC_0_NETWORK_SUBNET##*/}"
MTU=''
NAME='Ethernet Card 0'
NETMASK=''
NETWORK=''
REMOTE_IPADDR=''
STARTMODE='auto'
USERCONTROL='no'
EOF

${GFR} -- \
  upload ${netcfg} /etc/sysconfig/network/ifcfg-eth0

${GFR} -- write /etc/sysconfig/network/routes \
  "default ${NIC_0_NETWORK_GATEWAY} - -“
fi
```

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

```
${GFR} -- exit
```

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

your own OS interface based on libguestfs in 8 steps (bash)

1. get your OS-Image on the instance disk
2. start the helper VM
3. get the distro and version
4. remove SSH host keys
5. remove persistent NIC names
6. set hostname
7. create network config
8. stop the helper VM

- put this steps in the create function of the Ganeti OS interface
- further customisations steps are usual:
  - generate new UUIDs (filesystems, LVM-PVs and VGs, machine-id)
  - remove users
  - set passwords (i.e. root)
  - delete logs, temp files, histories, coredumps, blkid-tab, ca-keys/certificates, cron-spool, kerberos keys, mails, …
  - firewall rules

- `GFR=/usr/bin/guestfish --remote=${GUESTFISH_PID}`

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

simple webconsole using novnc

- somehow (periodically?) generate a HTML page from the output of

```
gnt-instance list -o name,pnode,network_port,hv/vnc_bind_address --no-headers
```

- containing links for every instance with vnc_bind_address set

```
<a href="./cgi-bin/ganeti-simple-webconsole.cgi
      ?node=${pnode}&vnc_port=${vnc_port}&ws_port=$(( vnc_port + OFFSET ))
      &cluster=${cluster}>${name}</a>
```

- simple CGI webserver on master to serve this HTML-Page, the CGI-script, and NOVNC
- the CGI script uses parameters to launch:

```
websockify --daemon --wrap-mode=exit ${ws_port} -- \
    ssh -o … -L "${ws_port}":127.0.0.1:"${vnc_port}" "${node}" sleep 10s
```

- and redirect the browser to NOVNC which in turn uses the WS

```
<meta http-equiv="Refresh" content="0;
url=http://${cluster}:${PORT}/novnc/vnc_auto.html?host=${cluster}&port=${ws_port}" />
```

simple webconsole using novnc (continued)

advantage

- console can be served from the master-node (cluster-name / -IP as single point of contact)
- vnc_bind_address 127.0.0.1 can be used (no VNC auth necessary)

improvements

- used webserver is python built-in → real webserver with HTTPS/WSS and authentication
- the CGI script allows “tunnelling” (through SSH and WS) of every requested port
  - check for a “safe” port range?
  - better use websockify tokens (>=v0.7): client knows the token, server maps to host and port
- serve the HTML and WS from the same server/port to use authentication also for the WS
- CGI-script could implement console-ACL by mapping instance tags to users/groups
- use hooks to start/stop webserver with the master-role

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

grow an instance disk online (with a hook)

- grow with Ganeti:

  `gnt-instance grow-disk --no-wait-for-sync some.vm 0 1G`

- implemented as disk-grow-post.d hook
- Connect to the qemu HMP socket on the instance primary node

  `socat STDIO UNIX-CONNECT:/var/run/ganeti/kvm-hypervisor/ctrl/${GANETI_INSTANCE_NAME}.monitor`

- identify the disk by UUID: `info block`

  Compare with `$GANETI_INSTANCE_DISK0_UUID`

- inform qemu and guest about the new size: `block_resize ${disk_id_qemu} ${new_size}`
  - `new_size=$GANETI_POST_INSTANCE_DISK0_SIZE`
  - `disk_id_qemu` from above, something like `hotdisk-fb3a86d3-pci-4`
- safe with DRBD, plain, blockdev; with sharedfile new_size must match exactly, or qemu will adjust to the new size (may even shrink the disk)
- Demo?

Network security (anti spoofing)

- use ebtalbes to
  - stop IP, MAC and ARP-Spoofing
  - but still allow QEMU-ARP-like announcements (i.e. for live migration)

- implemented as instance-start-post.d and instance-stop.pre-d (needs the tapX of the instance)
  - tapX not available in hook ENV → use `/var/run/ganeti/kvm-hypervisor/nic/some.vm/0`
  - Ganeti hasn’t allocated a tapX in phase start-pre and stop-post
  - ruleset https://libvirt.org/firewall.html clean-traffic
  - support for live migration pending

- costs performance?

- Demo?

Topics

1 Introduction

2 Ganeti @GISA

3 build from source

4 OS interface based on libguestfs

5 simple webconsole

6 hooks (online grow, anti-spoof)

7 future of Ganeti

wishes and ideas for the future of Ganeti

for users

- make releases 2.16/2.17
  - https://github.com/ganeti/ganeti/issues/292 (RSA instead of DSA keys -> fails inUbuntu 16.04)
- push/hold Ganeti to/in Ubuntu/Debian
- a proper website (maybe) some wiki with guides for common setups, problems, solutions
  - high quality → should work out of the box

for devs

- try to make contribution easy (like in ansible), but keep quality

features we would like

- more on (automatic) HA
- storage live migration (qemu block-mirror?)
- ESI openstack-cinder?
- like ESI an external network interface would be nice (if SDN is more used)

Thanks

Q&A

past conversions (cluster #03: LXC → KVM)

past goal: LXCs as VMs → will work, but limitations and additional effort doesn’t outweigh benefits

- limitations: isolation frameworks are diverse/complex or incomplete
  - access to block devices is granted by maj:min number → not static per instance
  - /proc/{meminfo,uptime,stats,diskstats} are from the host → vmstat, sar, free, uptime, snmpd all „lie“

    lxcfs: „Upgrading LXCFS without breaking running containers”[3]
  - /proc/sys (aka sysctl) is partial namespaced (net) → otherwise mounted „special“ (lxc.mount.auto)
  - CAP_SYS_ADMIN „is overloaded“[2] but needed for mount inside LXC (i.e. NFS) → mitigate with apparmor
  - cgroup v1 „seems to present a number of inconsistencies and a lot of chaos“[1] (cgroup v2 in linux-4.5)
  - a container reaching mem-limit? → host will swap until death, killing all other LXCs