ctys-uc-QEMU(7)
Use-Cases for QEMU

September 29, 2020

Contents

List of Figures

.

1 Install and Configure a VM with ctys-createConfVM

The current version supports a new installer with minimal required remaining manual actions. The Installation process id described within the document **ctys-configuration-QEMU** for QEMU/KVM. This is basically a 2-stage-approach. The first stage is the call of the interactive tool **ctys-configuration-QEMU** for creation of a generic Wrapper-Script and an additional configuration file with appropriate values for most of practical cases. The second stage starts QEMU/KVM and begins the boot of the selected GuestOS from the configured bootmedia. This could be performed by several optional interfaces, either from the standard ctys call or by direct-execution of the wrapper-script.
The whole process is designed to be executed in a straight forward manner and should be prefered for the first steps instead of the following legacy-examples.

2 CREATE a session

The first tests and examples of the QEMU plugin are based on the "arm-tst" VM contained in the examples of QEMU. This is a ready to use VM, but the TCP/IP address is hardcoded to "10.0.0.2" thus might be required to be configured. The coldfire test VM contained in the QEMU examples supports DHCP, thus is ready to use within the network. Anyhow, for the first tests the actual usage of the network is not yet required. All following examples, if not stated else, rely on the provided configuration file "arm-test.ctys" and the QEMU VM "arm-test". These have to be installed as described within the examples chapter for SECTIONs:qemuInstall .

The first call now creates a session and starts the VM with VNC as a console which will be attached automatically.

```
ctys -t qemu -a create=p:\$HOME/qemu/tst/arm-test/arm-test.ctys,reuse,console:vnc lab00
```

When the **vde_switch** is not configured yet the following error message occurs:

```
Missing management socket for "vde\_switch"
   QEMUMGMT=/var/tmp/vde\_mgmt0.acue
Call: "ctys-vnetctl" on "lab00.soho"
```

The solution is simply to proceed as requested and just create the UNIX Domain sockets by the following call with root permissions:

```
ctys-steupVDE -u \$USER create
```

The call could be executed remote too:

```
ctys-steupVDE -u \$USER create root@lab00
```

The setup should be operational now. The support of the types of CONSOLEs depends on the actually implemented call within the "arm-tst.ctys" script, which is a shell script with a defined interface. The currently supported CONSOLE types by arm-test are: "CLI, SDL, VNC". The CLI and SDL types are supported as DISPLAYFORWARDING in synchronous mode only for this version.

The following call creates an SDL CONSOLE.

```
ctys -t qemu -a create=p:\$HOME/qemu/tst/arm-test/arm-test.ctys,reuse,console:sdl lab00
```

As might be expected, the following call creates a CLI CONSOLE.

```
ctys -t qemu -a create=p:\$HOME/qemu/tst/arm-test/arm-test.ctys,reuse,console:cli lab00
```

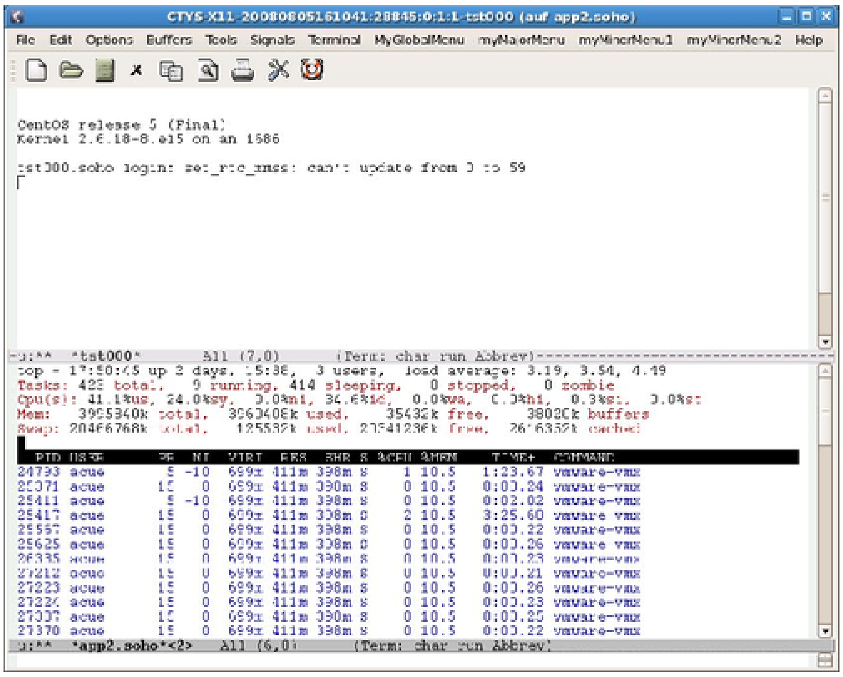
The monitor as configured within "arm-test.ctys" could be attache by usage of "netcat"

```
nc -U \${MYQEMUMONSOCK}
```

which could be generated by the function "netGetUNIXDomainSocket" and is derived from "QEMUMONSOCK" as raw-pattern, for additional information refer to the "arm-text.ctys" inline comments.
The QEMU monitor now could be entered by typing 'Ctrl-a'+'c'+'<RET>', the console is recovered by typing the same sequence again. For additional information refer to the QEMU User-Manual. A second terminal emulation to be used is the 'unixterm' command of VDE.

Alternatively EMACS could be used as a terminal emulation for CONSOLE access, either in "shell-mode" or in "ansi-term" mode. This work s the same way as an ordinary xterm session, where within the the "display-window" a cli is started connecting to a local UNIX domain socket. The socket has to be configured as a serial device within the GuestOS. For EMACS two additional variants exist for both modes, where the frame is divided into two windows, which connects the <execution-target> and the <machine-address> representing the GuestOS.

In the following example in the upper window a login-prompt of the GuestOS is displayed, whereas in the bottom window the "top" comman is shown for the hosting machine.


CTYS-X11-20080805161041:28845:0:1:1-tst000 (auf app2.soho)
File Edit Options Buffers Tools Signals Terminal MyGlobalMenu myMajorMenu myMinorMenu1 myMinorMenu2 Help

```
CentOS release 5 (Final)
Kernel 2.6.18-8.el5 on an i686

tst000.soho login: set_rtc_mmss: can't update from 0 to 59

-u:**  *tst000*        All (7,0)       (Term: char run Abbrev)------------------------------
top - 17:50:45 up 2 days, 15:38,  3 users,  load average: 3.19, 3.54, 4.49
Tasks: 423 total,   9 running, 414 sleeping,   0 stopped,   0 zombie
Cpu(s): 41.1%us, 24.0%sy,  0.0%ni, 34.6%id,  0.0%wa,  0.0%hi,  0.3%si,  0.0%st
Mem:   3955340k total,  3963406k used,    35432k free,    38020k buffers
Swap: 20466768k total,   125532k used, 20541236k free,  2616352k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
24793 acue       5 -10  699m 411m 398m S    1 10.5   1:23.67 vmware-vmx
25071 acue      15   0  699m 411m 398m S    0 10.5   0:00.24 vmware-vmx
25411 acue       5 -10  699m 411m 398m S    0 10.5   0:02.02 vmware-vmx
25417 acue      15   0  699m 411m 398m S    2 10.5   3:25.60 vmware-vmx
25567 acue      15   0  699m 411m 398m S    0 10.5   0:00.22 vmware-vmx
25625 acue      15   0  699m 411m 398m S    0 10.5   0:00.26 vmware-vmx
26335 acue      15   0  699m 411m 398m S    0 10.5   0:00.23 vmware-vmx
27212 acue      15   0  699m 411m 398m S    0 10.5   0:00.21 vmware-vmx
27223 acue      15   0  699m 411m 398m S    0 10.5   0:00.26 vmware-vmx
27224 acue      15   0  699m 411m 398m S    0 10.5   0:00.23 vmware-vmx
27307 acue      15   0  699m 411m 398m S    0 10.5   0:00.25 vmware-vmx
27370 acue      15   0  699m 411m 398m S    0 10.5   0:00.22 vmware-vmx
-u:**  *app2.soho*<2>   All (6,0)      (Term: char run Abbrev)
```

Figure 1: CONSOLE:EMACSAM for a QEMU Session

The console with pure CLI access could be combined with an VNC console allowing additional graphical access. This is particularly forseen, and will be offered soon, same as a debugging facility for GDB access to QEMU and to applications within the GuestOS.

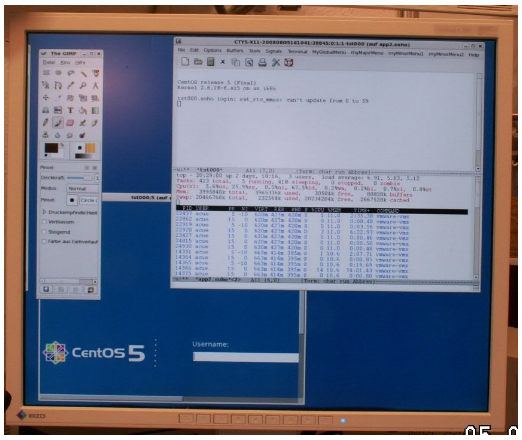


Figure 2: CONSOLE:EMACSAM and CONSOLE:VNC

3 CANCEL a session

The following call CANCELs the arm-test session.

```
ctys -t qemu -a cancel=p:\$HOME/qemu/tst/arm-test/arm-test.ctys,force,poweroff:0 lab00
```

4 LIST sessions

The following call LISTs all sessions:

```
ctys -a list lab00
```

resulting to:

```
TCP-container|TCP-guest  |label   |sesstype|c|user|group
\sout{---------}+\sout{-------}+\sout{----}+\sout{----}+-+\sout{--+-------}
lab00.soho   |-          |arm-test|VNC     |C|acue|ldapusers
lab00.soho   |-          |LAB00   |VNC     |C|root|root
lab00.soho   |-          |LAB00   |VNC     |S|root|root
lab00.soho   |tst109     |arm-test|QEMU-arm|S|acue|ldapusers
lab00.soho   |-          |Domain-0|XEN     |S|-   |-
lab00.soho   |lab00.soho.|lab00   |PM      |S|-   |-
```

The following call LISTs all sessions by usage of a specific LIST-MACRO for QEMU:

```
ctys macro:listconnpid lab00
```

Resulting in:

```
label   |stype   |c|DIS|cport|sport|pid  |PM        |TCP
\sout{----}+\sout{----}+-+\sout{-+---}+\sout{-}+\sout{-}+\sout{------}+\sout{--------}
LAB00   |VNC     |C|1  |     |     |18933|lab00.soho|
LAB00   |VNC     |S|1  |5901 |     |5642 |lab00.soho|
arm-test|QEMU-arm|S|17 |     |25704|25832|lab00.soho|
Domain-0|XEN     |S|   |     |     |     |lab00.soho|
lab00   |PM      |S|   |     |     |1    |lab00.soho|192.168.1.71
```

5 ENUMERATE sessions

The following call ENUMERATEs all stored session configurations within the subdirectory of the HOME.

```
ctys -t qemu -a enumerate=b:qemu/tst lab00
```

The following call displays a listing formatted as a table:

```
ctys -t qemu macro:listconn lab00
```

6 SHOW

The following call SHOWs dynamic data.

```
ctys -t qemu -a show lab00
```

7 INFO

Particularly the available capabilities for QEMU are displayed, which contains a list of all available CPUs and the related system boards.

```
ctys -t qemu -a info lab00
```

This leads to:

```
Node:lab00.soho  -  ctys(01\_04\_001A03)
   System      :Linux
   OS          :Linux
   RELEASE     :2.6.18-8.1.15.el5.centos.plusxen
   MACHINE     :x86\_64
   KERNEL\#CPU  :SMP-KERNEL
   CPU-INFO
     processor:0
         vendor\_id           :GenuineIntel
         cpu family          :6
         model               :22
         model name          :Intel(R) Celeron(R) CPU 420  @ 1.60GHz
         stepping            :1
         cpu MHz             :1599.853
         cache size          :512 KB

      Flags assumed equal for all processors on same machine:
         flags
            vmx(VT-x - Pacifica)    = 0
            svm(AMD-V - Vanderpool) = 0
            PAE                     = 1

   MEM-INFO
      MemTotal               :   523 M
      SwapTotal              :  2031 M

   SOFTWARE
      Mandatory:
         bash        :GNU bash, version 3.1.17(1)-release
                        (x86\_64-redhat-linux-gnu)
         gawk        :GNU Awk 3.1.5
         sed         :GNU sed version 4.1.5
         SSH         :OpenSSH\_4.3p2, OpenSSL 0.9.8b 04 May 2006
         top         :top: procps version 3.2.7

      Optional:
         wmctrl      :wmctrl is on this machine not available
         lm\_sensors   :sensors version 2.10.0 with libsensors version
                      2.10.0
         hddtemp     :hddtemp version 0.3-beta13

   PLUGINS     :QEMU CLI X11 VNC
      QEMU:        Plugin Version:01.01.001a00pre
                   Operational State:ENABLED
                   QEMU version:
                   ->QEMU-0.9.1
                      Magic-Number:QEMU\_091
                      Verified Prerequisites:
                      ->CLI-ValidatedBy(hookInfoCheckPKG)
                      ->X11-ValidatedBy(hookInfoCheckPKG)
                      ->VNC-ValidatedBy(hookInfoCheckPKG)
                      -><LocalClientCLI>
                      -><LocalClientX11>
                      -><LocalClientVNC>
                      -><LocalXserverDISPLAY>
                      -><delayedValidationOnFinalTarget>
```

```
                        -><QEMU-0.9.1>
                        ->\_/usr/local/bin/vde\_switch\_info-USER=
                              acue-ACCESS-PERMISSION-GRANTED
                        ->\_/usr/local/bin/unixterm\_info-USER=
                              acue-ACCESS-PERMISSION-GRANTED
                        ->\_/usr/local/bin/vdeq\_info-USER=
                              acue-ACCESS-PERMISSION-GRANTED
                        ->\_/usr/local/bin/vdeqemu\_info-USER=
                              acue-ACCESS-PERMISSION-GRANTED
                        -><QEMUSOCK=/var/tmp/vde\_switch0.acue\_info-USER=
                              acue-ACCESS-GRANTED>
                        -><QEMUMGMT=/var/tmp/vde\_mgmt0.acue\_info-USER=
                              acue-ACCESS-GRANTED>
                        -><CPU-Emulation:qemu-alpha>
                        -><CPU-Emulation:qemu-arm>
                        -><CPU-Emulation:qemu-armeb>
                        -><CPU-Emulation:qemu-cris>
                        -><CPU-Emulation:qemu-i386>
                        -><CPU-Emulation:qemu-img>
                        -><CPU-Emulation:qemu-m68k>
                        -><CPU-Emulation:qemu-mips>
                        -><CPU-Emulation:qemu-mipsel>
                        -><CPU-Emulation:qemu-ppc>
                        -><CPU-Emulation:qemu-ppc64>
                        -><CPU-Emulation:qemu-ppc64abi32>
                        -><CPU-Emulation:qemu-sh4>
                        -><CPU-Emulation:qemu-sh4eb>
                        -><CPU-Emulation:qemu-sparc>
                        -><CPU-Emulation:qemu-sparc32plus>
                        -><CPU-Emulation:qemu-sparc64>
                        -><CPU-Emulation:qemu-system-arm>
                        -><CPU-Emulation:qemu-system-cris>
                        -><CPU-Emulation:qemu-system-m68k>
                        -><CPU-Emulation:qemu-system-mips>
                        -><CPU-Emulation:qemu-system-mips64>
                        -><CPU-Emulation:qemu-system-mips64el>
                        -><CPU-Emulation:qemu-system-mipsel>
                        -><CPU-Emulation:qemu-system-ppc>
                        -><CPU-Emulation:qemu-system-ppc64>
                        -><CPU-Emulation:qemu-system-ppcemb>
                        -><CPU-Emulation:qemu-system-sh4>
                        -><CPU-Emulation:qemu-system-sh4eb>
                        -><CPU-Emulation:qemu-system-sparc>
                        -><CPU-Emulation:qemu-system-x86\_64>
                        -><CPU-Emulation:qemu-x86\_64>

      CLI:           Plugin Version:01.01.001a02
                     Operational State:DISABLED

      X11:           Plugin Version01.01.001a02
                     Operational State:DISABLED

      VNC:           Plugin Version:01.02.001b01
                     Operational State:DISABLED
```

8 QEMU Examples

8.1 ARM

After installation of QEMU and VDE as described the utility **ctys-plugins** should be called for validation of the operational state of the QEMU installation. The following call verifies the different plugins operational states for server functionality.

```
ctys-plugins -T all -e
```

The client functionality could be verified with the call:

```
ctys-plugins -T all
```

Now, with a properly installed test environment from QEMU and the additional ctys call-scripts setup as described before, the following call should start the **arm-test** QEMU VM with and CONSOLE of type SDL.

```
ctys -t qemu -a create=f:qemu/tst/arm-test/arm-test.ctys,console:sdl lab00
```

In case of ambiguous filenames in the cacheDB e.g. due to multiple access paths on multiple nodes by NFS the following approaches could be applied

use "p:<pathname>"
When the full absolute path by **p:<pathname>** is provided, no local ambiguity may occur within the execution context. This is recommended for the first steps, because it does not require any additional action.

<machine-address>: Additional entries may lead to non-ambiguity. This depends on the contents of the distributed caches and requires some knowledge of the system.

deactivate cacheDB:
Another quick solution is the disabling of any caching, therefore the options **-c off** and **-C off** could be set. This leads to a filesystem scan, which of course results in some performance degradation, which could be serious in case of deep filestructures with a "late match". The scan is performed by usage of the system utility **find**.

The supported CONSOLE types for the from-the-box "arm-test" VM are CLI, SDL, and VNC. Additional information is available as inline comment within the "arm-test.ctys" configuration from the

```
\$HOME/ctys/templates
```

directory. After this call an SDL terminal window should be opened. In case of networking problems the most common error is the forgotten call of **ctys-vnetctl -u <USER> create**.

8.2 Coldfire

```
ctys -t qemu -a create=f:/qemu/tst/coldfire-test-0.1/coldfire.ctys,console:cli lab00
```

9 SEE ALSO

ctys-CLI(1) , *ctys-configuration-QEMU(7)* , *ctys-createConfVM(1)* , *ctys-plugins(1)* , *ctys-QEMU(1)* , *ctys-uc-CLI(7)* , *ctys-uc-VNC(7)* , *ctys-uc-X11(7)* , *ctys-vhost(1)*
, *ctys-VNC(1)* , *ctys-X11(1)*

For GuestOS Setups:
ctys-uc-Android(7) , *ctys-uc-CentOS(7)* , *ctys-uc-MeeGo(7)*

10 AUTHOR

Arno-Can Uestuensoez <https://arnocan.wordpress.com/>
<https://unifiedsessionsmanager.sourceforge.io/>
<https://github.com/unifiedsessionsmanager>

11 COPYRIGHT