kdump autotest setup

This document takes Fedora server 28 for example, describle how to setup the kdump autotest environment from source in newly installed OS, and all the other relevant configuration, in order to run our test script. The scripts **highly depends** on this configuration! So, please make sure you follow this doc before testing.

We suppose your KVM guest is on the same host you are operating.

Components compilation

There are 4 components: linux kernel, kexec-tools, makedumpfile, crash, need to build. For your convenience, you can directly install all the necessary packages via:

```
sudo dnf install gcc gcc-c++ bison flex openssl-devel ncurses-devel gcc-plugin-devel elfutils-libelf-devel automa
ke autoconf lzo-devel patch bzip2-devel git
```

build kexec-tools

```
./bootstrap
./configure
make
```

build makedumpfile

Download elfutils first at: here. In my case I use elfutils-0.172.tar.bz2. Decompress and build it like this:

```
tar xvf elfutils-0.172.tar.bz2
cd elfutils-0.172
./configure
make
```

You may see compilation errors, but it doesn't matter, because the header files & libbraries we need have been generated. Then copy the following header files & libraries:

```
sudo mkdir -p /usr/local/include/elfutils/
sudo cp elfutils-0.172/libdw/libdw.h /usr/local/include/elfutils/
sudo cp elfutils-0.172/libdwfl/libdwfl.h /usr/local/include/elfutils/
sudo cp elfutils-0.172/libdw/dwarf.h /usr/local/include/
sudo cp elfutils-0.172/libdw/libdw.a   /usr/local/lib/libdw.a
sudo cp elfutils-0.172/libebl/libebl.a /usr/local/lib/libebl.a
```

And then build makedumpfile with:

```
make LINKTYPE=dynamic
```

build crash

crash need gdb-7.6.tar.gz, and it could be downloaded automatically during make. But limited by FNST's network, we download it first and put it into crash directory. Type `make lzo`, if you see compilation error seems non-sense to you, try remove gdb directory, and `make lzo` again.

Other Configurations

ssh

For connecting KVM Guest without password, **public-key authentication** a simple solution.

1.generate key-pair with

```
ssh-keygen
```

without any parameter. Just press "Enter" all the way, it will generates file "id_rsa" & "id_rsa.pub" under your ~/.ssh/

2.copy your local public key to KVM Guest with

```
ssh-copy-id -i .ssh/id_rsa.pub username@host_ip
```

It will prompt you to input password, but next time, when login with "ssh username@host_ip", no need password anymore.

3.Create alias for hosts in you ~/.ssh/config like this

> Host vm
> HostName 192.168.122.217
> User iaas

If there is no ~/.ssh/config, create it manually, then modify the permission with:

```
chmod 600 ~/.ssh/config
```

Then you could use `ssh vm` instead of `ssh iaas@192.168.122.217`

grub

Edit your /etc/default/grub, append

> crashkernel=256M console=ttyS0,115200n8 unknown_nmi_panic=1

to GRUB_CMDLINE_LINUX. **NOTE**: crashkernel=256M may not be enough for kdump to work! When it doesn't work, consider to increase this value and retry!

After installing new kernel & reboot, the default choice of kernel is not the one just installed, it should be selected **manually** in the grub menu. so, after reboot, please **double check** your current kernel version.

kdump

In order to test kdump with the latest components, After executed test script 1, please check the current kernel version with `uname -a`. In order to use the latest kexec & makedumpfile, follow this step:

1.Edit /etc/kdump.conf with

> core_collector /home/pino/workspace/makedumpfile/makedumpfile -l --message-level 1 -d 31

Using your own path of makedumpfile.

2.Customize kdumpctl with:

```
cp /usr/bin/kdumpctl ~/workspace/
vi workspace/kdumpctl
```

Then modify workspace/kdumpctl with your own kexec tool like this:

> KEXEC=/home/pino/workspace/kexec-tools/build/sbin/kexec

sudo

Both in host & guest, we offen need sudo to execute some command that would prompt you input your account password, for example, `sudo inject-nmi domain` to trigger kdump, `sudo crash vmlinux vmcore` to analyse dumpfile. To save the effort of inputting password and make life easier, `sudo visudo` to add the following line to /etc/sudoers in both host & guest:

> yourUserName ALL=(ALL) NOPASSWD: ALL

Substitute "yourUserName" with your own, means it won't prompt you to input password when execute `sudo` under "yourUserName".

kernel

Generally speaking, when configuring kernel, `make localmodconfig` is the best choice, it use your current running kernel's config, and drop all unnecessary modules. But when you want test certain feature, this is what you should do: prepare your own config file, say iaas.config, put it under kernel/configs/, then `make iaas.config`. the content of iaas.config is like .config, for

example:

> CONFIG_EFI=n

it is the feature you want to enable(=y) or disable(=n).

BUT, this is not a easy thing because of the complex dependency among config items! Even if you enable one feature in iaas.config, it may also be disabled automatically by its dependency in the final .config. The dependency relationship can be got by searching it in `make menuconfig`. So Please be very carefully if you want it work as expected.