Using Docker to Test Build Perf
Learning by Building Missing Infrastructure

Arnaldo Carvalho de Melo

Red Hat Inc.

July 21, 2016
One way of doing containers

- Seems to be the most popular
- Adopted by Red Hat
- Together with kubernetes, for clustering
- But it is possible to export its images as tarballs
- And use it with systemd-nspawn
- And replace kubernetes with swarm, fleet, mesos...
- Good enough for my purposes so far
Basic facts

- Has a daemon
- Uses cgroups, namespaces, etc, just like Neil described
- Uses git concepts/subcommands
- push, pull, create, image layer ids named by content hashes
- Public repository
- And a RedHat one
- Lots of pre-built images
- You can publish some more
- Pulls to your machine at first use
- 2nd use is super fast
Good for test building

- Lots of base images (chroots)
- After building, everything is thrown away
- Lots of documentation
- Public infrastructure
- Better than learning something specific for testing
- We can use existing cloud infrastructure
- To scale building to many targets using kubernetes
- Other people can reproduce your tests without much setup time
# dnf install docker
Installed size: 35 M
# systemctl start docker
# cat /etc/redhat-release
Fedora release 24 (Twenty Four)
# time docker run -ti debian cat /etc/debian_version
Unable to find image 'debian:latest' locally
Trying to pull repository docker.io/library/debian ...
latest: Pulling from docker.io/library/debian
5c90d4a2d1a8: Pull complete
Digest: sha256:8b1fc3a7a55c42e3445155b2f8f40c55de5f8bc8012992b26b570530c4
Status: Downloaded newer image for docker.io/debian:latest
8.5
real 0m32.537s
[root@zoo ~]# time docker run -ti debian cat /etc/debian_version
8.5
real 0m1.280s
# docker images
<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>docker.io/debian</td>
<td>latest</td>
<td>1b088884749b</td>
<td>5 weeks ago</td>
<td>125.1 MB</td>
</tr>
</tbody>
</table>
RedHat’s docker registry

- RHEL images
- Edit /etc/sysconfig/docker
- --add-registry registry.access.redhat.com
```
# docker search rhel7

<table>
<thead>
<tr>
<th>INDEX</th>
<th>NAME</th>
<th>DESCRIPTION</th>
<th>STARS</th>
</tr>
</thead>
<tbody>
<tr>
<td>redhat.com</td>
<td>registry.access.redhat.com/rhel7</td>
<td>minimal runtime</td>
<td>0</td>
</tr>
<tr>
<td>redhat.com</td>
<td>registry.access.redhat.com/rhel7.0</td>
<td>minimal runtime</td>
<td>0</td>
</tr>
<tr>
<td>redhat.com</td>
<td>registry.access.redhat.com/rhel7.1</td>
<td>minimal runtime</td>
<td>0</td>
</tr>
<tr>
<td>redhat.com</td>
<td>registry.access.redhat.com/rhel7.2</td>
<td>minimal runtime</td>
<td>0</td>
</tr>
</tbody>
</table>
```

Arnaldo Carvalho de Melo

Using Docker to Test Build Perf
Base containers are minimal, no toolchain
Perf has many features, needs lots of -devel packages
Does feature detection
Some libraries miss some features
Start with bare minimum: make, bison, flex, gcc
Try to run perf inside the container
Use –volume to map source dir in the host to the container
Use :Z suffix to SELinux relabel and avoid –privileged
Building Software in a Container

```
[root@jouet 5]# docker run --volume /home/acme/git:/git:Z 
    --tty --interactive fedora:24 /bin/bash
[root@545f2335f316 ~]# dnf install make bison flex gcc
Total download size: 44 M
[root@545f2335f316 ~]# make -C tools/perf 0=/tmp/
  BUILD: Doing 'make -j4' parallel build
Auto-detecting system features:
  ...
    dwarf: [ OFF ]
  ...
    dwarf_getlocations: [ OFF ]
  ...
    glibc: [ on ]
  ...
    gtk2: [ OFF ]
  ...
    libaudit: [ OFF ]
  ...
    libbfs: [ OFF ]
  ...
    libelf: [ OFF ]
  ...
    libnuma: [ OFF ]
  ...
    numa_num_possible_cpus: [ OFF ]
  ...
    libperl: [ OFF ]
  ...
    libpython: [ OFF ]
  ...
    libslang: [ OFF ]
  ...
    libcrypto: [ OFF ]
  ...
    libunwind: [ OFF ]
  ...
    libdwarf-unwind: [ OFF ]
  ...
    zlib: [ OFF ]
  ...
    lzma: [ OFF ]
  ...
    get_cpuid: [ on ]
  ...
    bpf: [ on ]
```

No libelf found, disables 'probe' tool and BPF support in 'perf record', please install libelf-dev, libelf-devel or elfutils-libelf-devel
No sys/sdt.h found, no SDT events are defined, please install systemtap-sdt-devel or systemtap-sdt-dev
Disabling post unwind, no support found.
No libaudit.h found, disables 'trace' tool, please install audit-libs-devel or libaudit-dev
No libcrypto.h found, disables jitted code injection, please install libssl-devel or libssl-dev
sling not found, disables TUI support. Please install slang-devel, libslang-devel or libslang2-dev
GTK2 not found, disables GTK2 support. Please install gtk2-devel or libgtk2.0-dev
Missing perl dev files. Disabling perl scripting support, please install perl-ExtUtils-Embed/libperl-dev
No python interpreter was found: disables Python support - please install python-devel/python-dev
No liblzma found, disables xz kernel module decompression, please install xz-devel/liblzma-dev
No numa.h found, disables 'perf bench numa mem' benchmark, please install numacl-devel/libnuma-devel/libnuma-dev
Running perf on container

[root@545f2335f3 linux]# /tmp/perf stat ls
Error:
You may not have permission to collect stats.

Consider tweaking /proc/sys/kernel/perf_event_paranoid, which controls use of the performance events system by unprivileged users (without CAP_SYS_ADMIN).

The current value is 2:

-1: Allow use of (almost) all events by all users
>= 0: Disallow raw tracepoint access by users without CAP_IOC_LOCK
>= 1: Disallow CPU event access by users without CAP_SYS_ADMIN
>= 2: Disallow kernel profiling by users without CAP_SYS_ADMIN

[root@545f2335f3 linux]# cat /proc/sys/kernel/perf_event_paranoid
2
[root@545f2335f3 linux]# echo 0 > /proc/sys/kernel/perf_event_paranoid
bash: /proc/sys/kernel/perf_event_paranoid: Read-only file system
Privileged Containers

- CAP_SYS_ADMIN dropped by default by docker
- Outside the container, 2 is the default for perf_event_paranoid
- Use –privileged
- But, have to create new container, install packages again
Privileged Containers: Running perf

[root@jouet rawhide]# docker run --privileged \\n-v /home/acme/git:/git:Z \\
-ti fedora:24 /bin/bash

<SNIP installing make bison flex gcc, building the tool>
[root@31a2376a4c /]# /tmp/perf stat sleep 1

Performance counter stats for 'sleep 1':

<table>
<thead>
<tr>
<th>Metric</th>
<th>Value</th>
<th>Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>task-clock (msec)</td>
<td>0.416815</td>
<td>0.000 CPUs utilized</td>
</tr>
<tr>
<td>context-switches</td>
<td>2</td>
<td>0.005 M/sec</td>
</tr>
<tr>
<td>cpu-migrations</td>
<td>0</td>
<td>0.000 K/sec</td>
</tr>
<tr>
<td>page-faults</td>
<td>49</td>
<td>0.118 M/sec</td>
</tr>
<tr>
<td>cycles</td>
<td>1059597</td>
<td>2.542 GHz</td>
</tr>
<tr>
<td>instructions</td>
<td>661786</td>
<td>0.62 insn per cycle</td>
</tr>
<tr>
<td>branches</td>
<td>125219</td>
<td>300.419 M/sec</td>
</tr>
<tr>
<td>branch-misses</td>
<td>6831</td>
<td>5.46% of all branches</td>
</tr>
</tbody>
</table>

1.002728350 seconds time elapsed
[root@31a2376a4c76 /]# /tmp/perf test 2>&1 | grep -v Ok
  1: vmlinux symtab matches kallsyms : Skip
  14: struct perf_event_attr setup : Skip
  21: Test object code reading : FAILED!
  33: Test kmod_path__parse function : FAILED!
  35: Test LLVM searching and compiling :
     35.1: Basic BPF llvm compiling test : Skip
     35.2: Test kbuild searching : Skip
     35.3: Compile source for BPF prologue : Skip
     35.4: Compile source for BPF relocation : Skip
  37: Test BPF filter : Skip (not compiled in)
  47: Test SDT event probing : Skip
  52: Test intel cqm nmi context read : Skip
Automate all this: Dockerfiles

- Create images on top of those "fedora:N" base images
- Dockerfile: Docker’s Makefile
- To create base images: tarball, debootstrap, fedpkg, etc
Fedora rawhide Dockerfile, simplified

[root@jouet rawhide]# cat Dockerfile
FROM docker.io/fedora:rawhide
RUN dnf -y install make gcc flex bison elfutils-libelf-devel \ 
    elfutils-devel libunwind-devel xz-devel \ 
    numactl-devel openssl-devel slang-devel \ 
    gtk2-devel perl-ExtUtils-Embed python-devel \ 
    binutils-devel audit-libs-devel && \
    mkdir -p /tmp/build/perf
ENTRYPOINT make -C /git/linux/tools/perf O=/tmp/build/perf
Building the Image

[root@jouet rawhide]# docker build -t fedora-build-perf .
Sending build context to Docker daemon 2.048 kB
Step 1: FROM docker.io/fedora:rawhide
    ---> 575f262f71e0
Step 2: RUN dnf -y install make gcc flex bison elfutils-libelf-devel \
    elfutils-devel libunwind-devel xz-devel \
    numactl-devel openssl-devel slang-devel \
    gtk2-devel perl-ExtUtils-Embed python-devel \
    binutils-devel audit-libs-devel && \
    mkdir -p /tmp/build/perf
    ---> Running in 7c760a77f9f9
Dependencies resolved.
<SNIP>
Complete!
    ---> cd2d189c994e
Removing intermediate container 7c760a77f9f9
Step 3: ENTRYPOINT make -C /git/linux/tools/perf O=/tmp/build/perf
    ---> Running in 7254db4a0d8a
    ---> d1b9d76573cf
Removing intermediate container 7254db4a0d8a
Successfully built d1b9d76573cf

Arnaldo Carvalho de Melo
Using Docker to Test Build Perf
Local image
Layer on top of the FROM image
ENTRYPOINT: start building perf right away
make exit value becomes the 'docker run' one
<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>fedora-build-perf</td>
<td>latest</td>
<td>d1b9d76573cf</td>
<td>10 minutes ago</td>
<td>851.1 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>24</td>
<td>f9873d530588</td>
<td>4 weeks ago</td>
<td>204.4 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>20</td>
<td>a3c57c6e3f04</td>
<td>5 weeks ago</td>
<td>290.6 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>rawhide</td>
<td>575f262f71e0</td>
<td>5 weeks ago</td>
<td>196.5 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>21</td>
<td>1a4b6ed2b9da</td>
<td>5 weeks ago</td>
<td>241.3 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>22</td>
<td>2d3da2084d08</td>
<td>5 weeks ago</td>
<td>188.7 MB</td>
</tr>
<tr>
<td>docker.io/fedora</td>
<td>23</td>
<td>507cb13a2160</td>
<td>4 months ago</td>
<td>204.7 MB</td>
</tr>
</tbody>
</table>
[root@jouet rawhide]# docker run -v /home/acme/git:/git:Z \
   -ti fedora-build-perf

  BUILD: Doing 'make -j4' parallel build
Auto-detecting system features:
...                     dwarf: [ on ]
...                     dwarf_getlocations: [ on ]
...                     glibc: [ on ]
...                     gtk2: [ on ]
<SNIP>
  CC       /tmp/build/perf/scripts/perl/Perf-Trace-Util/Context.o
In file included from /usr/lib64/perl5/CORE/perl.h:3905:0,
    from Context.xs:23:
/usr/lib64/perl5/CORE/inline.h: In function 'S_cx_popsub_args':
/usr/lib64/perl5/CORE/cop.h:612:13: error: declaration of 'av' shadows
    a previous local [-Werror=shadow]
<SNIP>
make: *** [all] Error 2
make: Leaving directory '/git/linux/tools/perf'
[root@jouet rawhide]# echo $?
2
Issues so far

- The container is needs to shrink (851 MB)
- It failed, what to do?
- Enter the container and do the build interactively
Fixing it all: Full Dockerfile

# docker.io/acmel/linux-perf-tools-build-fedora:rawhide
FROM docker.io/fedora:rawhide
MAINTAINER Arnaldo Carvalho de Melo <acme@kernel.org>
# Remove NO_LIBPERL=1 and fix perl breakage
RUN dnf -y install make gcc flex bison elfutils-libelf-devel \ 
    elfutils-devel libunwind-devel xz-devel \ 
    numactl-devel openssl-devel slang-devel \ 
    gtk2-devel perl-ExtUtils-Embed python-devel \ 
    binutils-devel audit.libs-devel && \ 
    dnf -y clean all && \ 
    rm -rf /usr/share/doc /usr/share/gtk-doc && \ 
    mkdir -m 777 -p /tmp/build/perf /tmp/build/objtool && \ 
    groupadd -r perfbuilder && \ 
    useradd -r -g perfbuilder perfbuilder

USER perfbuilder
ENTRYPOINT make -C /git/linux/tools/objtool O=/tmp/build/objtool && \ 
    make NO_LIBPERL=1 -C /git/linux/tools/perf O=/tmp/build/perf
<table>
<thead>
<tr>
<th>REPOSITORY</th>
<th>TAG</th>
<th>IMAGE ID</th>
<th>CREATED</th>
<th>SIZE</th>
</tr>
</thead>
<tbody>
<tr>
<td>alpine</td>
<td>3.4</td>
<td>f303da6b9c30</td>
<td>4 days ago</td>
<td>297 MB</td>
</tr>
<tr>
<td>android-ndk</td>
<td>r12b</td>
<td>9fadf6cf0601</td>
<td>4 days ago</td>
<td>512.4 MB</td>
</tr>
<tr>
<td>centos</td>
<td>5</td>
<td>e9df0e31192c</td>
<td>4 days ago</td>
<td>468 MB</td>
</tr>
<tr>
<td>centos</td>
<td>6</td>
<td>2b42f9b929c2</td>
<td>4 days ago</td>
<td>364.7 MB</td>
</tr>
<tr>
<td>centos</td>
<td>7</td>
<td>886b664936f8</td>
<td>4 days ago</td>
<td>432.6 MB</td>
</tr>
<tr>
<td>debian</td>
<td>7</td>
<td>d44dbbf005e5</td>
<td>4 days ago</td>
<td>479.8 MB</td>
</tr>
<tr>
<td>debian</td>
<td>8</td>
<td>8f4164c99b32</td>
<td>4 days ago</td>
<td>573.3 MB</td>
</tr>
<tr>
<td>debian</td>
<td>experimental</td>
<td>7e5f96e69180</td>
<td>4 days ago</td>
<td>946.5 MB</td>
</tr>
<tr>
<td>fedora</td>
<td>21</td>
<td>8e048f2a7bff</td>
<td>4 days ago</td>
<td>630.7 MB</td>
</tr>
<tr>
<td>fedora</td>
<td>22</td>
<td>ff03f5504609</td>
<td>4 days ago</td>
<td>663.3 MB</td>
</tr>
<tr>
<td>fedora</td>
<td>23</td>
<td>b22953c81fe5</td>
<td>4 days ago</td>
<td>690.5 MB</td>
</tr>
<tr>
<td>fedora</td>
<td>24</td>
<td>db43ea772bd4</td>
<td>4 days ago</td>
<td>548.3 MB</td>
</tr>
<tr>
<td>fedora</td>
<td>rawhide</td>
<td>3f643429c61f</td>
<td>4 days ago</td>
<td>714.4 MB</td>
</tr>
<tr>
<td>mageia</td>
<td>5</td>
<td>7097022c3c7b</td>
<td>24 hours ago</td>
<td>594 MB</td>
</tr>
<tr>
<td>opensuse</td>
<td>13.2</td>
<td>3c6179aac365</td>
<td>4 days ago</td>
<td>523.7 MB</td>
</tr>
<tr>
<td>opensuse</td>
<td>42.1</td>
<td>53bbba672842</td>
<td>4 days ago</td>
<td>535.6 MB</td>
</tr>
<tr>
<td>ubuntu</td>
<td>14.04.4</td>
<td>df3454f0b3b4</td>
<td>4 days ago</td>
<td>543.3 MB</td>
</tr>
<tr>
<td>ubuntu</td>
<td>15.10</td>
<td>cd775ef633bc</td>
<td>4 days ago</td>
<td>596.4 MB</td>
</tr>
<tr>
<td>ubuntu</td>
<td>16.04</td>
<td>245e625f955d</td>
<td>4 days ago</td>
<td>611.5 MB</td>
</tr>
<tr>
<td>ubuntu</td>
<td>16.04-x-armhf</td>
<td>1fe15532783c</td>
<td>26 hours ago</td>
<td>332.3 MB</td>
</tr>
</tbody>
</table>
Run the build from inside the container

- Replace the ENTRYPOINT with /bin/bash
- Using the full Dockerfile, it's not root anymore
- Just call make
- Outside the container, do changes to the source
- Rebuild inside and outside

```
# docker run --entrypoint=/bin/bash \
  -v /home/acme/git:/git:Z \
  -ti docker.io/acmel/linux-perf-tools-build-fedora:rawhide
bash-4.3$ cd /git/linux
bash-4.3$ make -C tools/perf O=/tmp
make: Entering directory '/git/linux/tools/perf'
  BUILD: Doing 'make -j4' parallel build
```
Publishing Images

- hub.docker.com
- Need to create an account at the image registry
- `docker login + docker push repository`
- Edit repository details on the web interface
- https://hub.docker.com/r/acmel/
- `docker pull` also exists, done implicitly by `docker run`
- `docker tag fedora:24 fedora:latest`
- Then `docker run fedora == docker run fedora:24`
Testing them all

- Simplified script loop:
- Full output on the next slide

```bash
for img in $(docker images docker.io/acmel/*) ; do
docker run -v /home/acme/git:/git:Z -ti $img
done
```
Testing them all

[root@jouet ~]# time dm
alpine:3.4: Ok
android-ndk:r12b: Ok
centos:5: Ok
centos:6: Ok
centos:7: Ok
debian:7: Ok
debian:8: Ok
debian:experimental: Ok
fedora:21: Ok
fedora:22: Ok
fedora:23: Ok
fedora:24: Ok
fedora:rawhide: Ok
mageia:5: Ok
opensuse:13.2: Ok
opensuse:42.1: Ok
ubuntu:14.04.4: Ok
ubuntu:15.10: Ok
ubuntu:16.04: Ok
ubuntu:16.04-x-armhf: Ok
real 12m9.994s
What is next

- Add to the upstream kernel sources
- Already goes in each pull request I send upstream
- Kubernetes to distribute the build over multiple machines
- Nodes with Fedora, RHEL and RHEL Atomic Host
- perf test
- make -C tools/perf build-test
# docker.io/acmel/linux-perf-tools-build-debian:8
FROM docker.io/debian:8
MAINTAINER Arnaldo Carvalho de Melo <acme@kernel.org>
ENV DEBIAN_FRONTEND noninteractive
RUN apt-get update && \
    apt-get install -y apt-utils && \
    apt-get install -y make gcc flex bison libelf-dev libdw-dev \ 
                        libunwind-dev libaudit-dev libssl-dev \ 
                        libclang2-dev libgtk2.0-dev libperl-dev \ 
                        python-dev libiberty-dev binutils-dev \ 
                        liblzma-dev libnuma-dev && \
apt-get clean -y && \
rmdir /usr/share/doc /usr/share/gtk-doc && \
mkdir -m 777 -p /tmp/build/perf /tmp/build/objtool && \
groupadd -r perfbuilder && \
useradd -r -g perfbuilder perfbuilder
USER perfbuilder
ENTRYPOINT make -C /git/linux/tools/perf O=/tmp/build/perf && \
    make -C /git/linux/tools/objtool O=/tmp/build/objtool
Another Dockerfile: Android NDK

```
[root@jouet arm]# cat Dockerfile
# docker.io/acmel/linux-perf-tools-build-android-ndk:r12b
FROM docker.io/fedora:24
MAINTAINER Arnaldo Carvalho de Melo <acme@kernel.org>
ENV SOURCEFILE=android-ndk-r12b-linux-x86_64.zip
RUN dnf -y install make bison flex unzip && 
    dnf -y clean all && 
    mkdir -m 777 -p /tmp/build/perf && 
    curl -OL http://dl.google.com/android/repository/${SOURCEFILE} && 
    unzip -d /opt ${SOURCEFILE} && 
    rm -f ${SOURCEFILE} && 
    rm -rf /opt/android-ndk-r12b/sources 
        /opt/android-ndk-r12b/platforms/android-[19]* 
        /opt/android-ndk-r12b/platforms/android-2[0-3]* 
        /opt/android-ndk-r12b/platforms/android-24/arch-mips* 
        /opt/android-ndk-r12b/platforms/android-24/arch-x86* 
        /opt/android-ndk-r12b/toolchains/x86* 
        /opt/android-ndk-r12b/toolchains/mips* 
        /opt/android-ndk-r12b/toolchains/llvm* 
        /opt/android-ndk-r12b/prebuilt/ 
        /opt/android-ndk-r12b/python* 
        /opt/android-ndk-r12b/shader-tools/ &&
    groupadd -r perfbuilder && 
    useradd -r -g perfbuilder perfbuilder
USER perfbuilder
ENV NDK=/opt/android-ndk-r12b/
ENV NDK_TOOLCHAIN=${NDK}/toolchains/.../bin/arm-linux-androideabi-
ENV NDK_SYSROOT=${NDK}/platforms/android-24/arch-arm
ENTRYPOINT make -C /git/linux/tools/perf 0=:/tmp/build/perf 
    ARCH=arm CROSS_COMPILE=${NDK_TOOLCHAIN} 
    EXTRA_CFLAGS="-pie --sysroot=${NDK_SYSROOT}"
[root@jouet arm]#
```
That is all folks!

Thanks!

Arnaldo Carvalho de Melo

acme@kernel.org

acme@redhat.com