RELIABLEEMBEDDEDSYSTEMS

The Yocto Project for mere m0rtals

Docker [1] + Yoctoproject Autobuilder [2] [3] (qemux86 [4])

Robert Berger - Reliable Embedded Systems - Consulting Training Engineering
http://www.ReliableEmbeddedSystems.com
robert.berger@ReliableEmbeddedSystems.com

last commit: *2014-07-21 14:52:23 +0300*
revision: *694cf8d69e55a4b322007cc23350a39a9c03ae7a*

Disclaimer

The views, opinions, positions or strategies expressed by the author and those providing comments are theirs alone, and do not necessarily reflect the views, opinions, positions or strategies of anybody else.

Intro

Objectives

The goal of this paper is to make it as easy as possible (but not easier) to enter the beatiful world of Embedded GNU/Linux with the Yocto project [6]. Docker [5] is used to minimize the "Works for me and I can't test your setup!" effect which could be a big stumbling stone for people who want to give Yocto a quick try. Another use-case would be for experienced users who need to have a defined build enviroment which they can pass around to others. In this paper we'll see how to build qemux86 [4] images with Yocto Autobuilder [3] and how to run the results. Ideally you only need to install a working version of Docker on your preferred Linux distro and off you go. Please note that a fast machine (8 cores+) with big (1TB+) and fast discs (RAID0, SSD) and a speedy broadband connection will make your Yocto experience more pleasant.

What is Docker?

This is how the official Docker site describes it [5]:

> Docker is an open platform for developers and sysadmins to build, ship, and run distributed applications. Consisting of Docker Engine, a portable, lightweight runtime and packaging tool, and Docker Hub, a cloud service for sharing applications and automating workflows, Docker enables apps to be quickly assembled from components and eliminates the friction between development, QA, and production environments. As a result, IT can ship faster and run the same app, unchanged, on laptops, data center VMs, and any cloud.

What is the Yocto Project?

This is how the official Yocto Project site describes it [6]:

> The Yocto Project is an open source collaboration project that provides templates, tools and methods to help you create custom Linux-based systems for embedded products regardless of the hardware architecture. It was founded in 2010 as a collaboration among many hardware manufacturers, open-source operating systems vendors, and electronics companies to bring some order to the chaos of embedded Linux development.

What is Yocto Autobuilder?

This is how the official Yocto Project site describes it [3]:

> AutoBuilder is a project to automate build tests and QA.
>
> One of the most neglected areas of open source development is testing and QA. A goal of The Yocto Project is to lead the industry in being the first open source project that targets embedded developers who
>
> - publish their QA and testing plans,
> - demonstrate their testing and QA in public,
> - and develop tools to automate test and QA procedures for the benefit of everyone.

What is QEMU?

This is how the official QEMU site describes it [4]:

> QEMU is a generic and open source machine emulator and virtualizer.
>
> When used as a machine emulator, QEMU can run OSes and programs made for one machine (e.g. an ARM board) on a different machine (e.g. your own PC). By using dynamic translation, it achieves very good performance.
>
> When used as a virtualizer, QEMU achieves near native performances by executing the guest code directly on the host CPU. QEMU supports virtualization when executing under the Xen hypervisor or using the KVM kernel module in Linux. When using KVM, QEMU can virtualize x86, server and embedded PowerPC, and S390 guests.

Install Docker and the Yocto Autobuilder Docker container

Install Docker

Just follow the instructions here [7]. For the instructions which follow I assume the account/user you'll utilize with docker is member of the `docker` group and a `sudoer`. I use Docker version 1.1.1 on Ubuntu 12.04 and 14.04 64 bit machines.

Download the Yocto Autobuilder Docker container

Listing 1: docker pull the Yocto Autobuilder container

```
1 docker pull reliableembeddedsystems/yocto-autobuilder
```

Depending on your broadband connection this might take some time. You should see something along those lines:

Listing 2: docker pull the Yocto Autobuilder container - output

```
1 Pulling repository reliableembeddedsystems/yocto-autobuilder
2 d208875f1b38: Pulling dependent layers
3 511136ea3c5a: Download complete
4 5e66087f3ffe: Download complete
5 4d26dd3ebc1c: Download complete
6 ...
7 6ab32805089f: Download complete
8 2cb1989dff4e: Download complete
9 16d9ac03df73: Download complete
10 e82a7002441e: Download complete
11 f1a6c27fcd3d: Download complete
```

Script to run the Yocto Autobuilder Docker container

Listing 3: get script to run Yocto Autobuilder Docker container

```
1 mkdir -p ~/docker/yocto-autobuilder-mainline
2 cd ~/docker/yocto-autobuilder-mainline
3 wget https://raw.githubusercontent.com/RobertBerger/yocto-autobuilder/master/
    non-local_scripts/docker_run.sh
4 chmod +x docker_run.sh
```

Run the Yocto Autobuilder Docker container

Make sure you choose the right network interface for your computer. In my case it's `br0`.

Listing 4: run the Yocto Autobuilder Docker container

```
1 ./docker_run.sh reliableembeddedsystems/yocto-autobuilder br0
```

You will need to enter a few things as indicated below.

Listing 5: run the Yocto Autobuilder Docker container - output

```
1 + sudo modprobe tun
2 [sudo] password for <user>:
3 + ID=$(docker run -t -i -d -p 22 -p 8010 --privileged reliableembeddedsystems
    /yocto-autobuilder /sbin/my_init -- bash -l)
4 + ssh to the container like this:
5 ssh -X genius@192.168.44.253 -p 49190
```

```
+ point your browser to:
http://192.168.44.253:49191
+ docker attach
    b5dfaa13e484d8cb23de157d7266ebd00e8b784f08ed4c3c1a419db742ba668b
*** Running /etc/my_init.d/00_regen_ssh_host_keys.sh...
*** Running /etc/my_init.d/01_create_tun.sh...
tun node does not exist. Generating it ...
*** Running /etc/my_init.d/02_start_autobuilder.sh...
starting yocto-autobuilder...
if [ -e twistd.pid ]; \
        then kill `cat twistd.pid`; \
        else echo "Nothing to stop."; \
        fi
/bin/sh: 2: kill: No such process

make: *** [stop] Error 1
if [ -e twistd.pid ]; \
        then kill `cat twistd.pid`; \
        else echo "Nothing to stop."; \
        fi
/bin/sh: 2: kill: No such process

make: *** [stop] Error 1
########################################################################

    [43/2842]
 Setting envvars.
 In the future though please add the following to your shell environment:
 PYTHONPATH=/home/genius/test/yocto-autobuilder/lib/python2.7/site-packages
    /:/home/genius/test/yocto-autobuilder/lib/python2.7/site-packages/
    autobuilder/:/home/genius/test/yocto-autobuilder/lib/python2.7/site-
    packages/autobuilder/buildsteps:$PYTHONPATH
 PATH=/home/genius/test/yocto-autobuilder/bin:/home/genius/test/yocto-
    autobuilder/yocto-autobuilder/scripts:$PATH
 YOCTO_AB_CONFIG=/home/genius/test/yocto-autobuilder/buildset-config/yoctoAB.
    conf

 If you don't, you should source this script everytime you want start the
 autobuilder.
 To start the autobuilder:
 ./yocto-start-autobuilder <worker|controller|both>

 To stop the autobuilder:
 ./yocto-stop-autobuilder <worker|controller|both>


Reading /home/genius/test/yocto-autobuilder/config/autobuilder.conf

 Setting WORKERBASEDIR to /home/genius/test/yocto-autobuilder/yocto-worker
 Setting OPTIMIZED_GIT_CLONE to True
 Setting OGIT_TRASH_DIR to /tmp/yocto-autobuilder/git/trash
 Setting OGIT_MIRROR_DIR to /tmp/yocto-autobuilder/git/mirror
 Setting OGIT_TRASH_CRON_TIME to "0 0 * * *"
 Setting OGIT_TRASH_NICE_LEVEL to "19"
 Setting BUILD_HISTORY_COLLECT to True
 Setting BUILD_HISTORY_DIR to "/tmp/yocto-autobuilder/buildhistory"
 Setting BUILD_HISTORY_REPO to "file:////tmp/yocto-autobuilder/buildhistory-
    repo"
 Setting ERROR_REPORT_COLLECT to True
 Setting PUBLISH_BUILDS to True
 Setting PUBLISH_SOURCE_MIRROR to False
```

```
 Setting PUBLISH_SSTATE to False
 Setting MAINTAIN_PERSISTDB to True
 Setting MACHINE_PUBLISH_DIR to "machines"
 Setting BA_PUBLISH_DIR to "build-appliance"
 Setting QEMU_PUBLISH_DIR to "/machines/qemu"
 Setting RPM_PUBLISH_DIR to "rpm"
 Setting DEB_PUBLISH_DIR to "deb"
 Setting IPK_PUBLISH_DIR to "ipk"
 Setting MARKED_RELEASE_DIR to "releases"
 Setting ADT_INST_PUBLISH_DIR to "adt-installer"
 Setting ADTQA_INST_PUBLISH_DIR to "adt-installer-QA"
 Setting SSTATE_PUBLISH_DIR to "/tmp/yocto-autobuilder/sstate_mirror"
 Setting SOURCE_PUBLISH_DIR to "/tmp/yocto-autobuilder/source"
 Setting BUILD_PUBLISH_DIR to "/tmp/yocto-autobuilder/builds"
 Setting X86TC_PUBLISH_DIR to "toolchain/i686"
 Setting X8664TC_PUBLISH_DIR to "toolchain/x86_64"
 Setting DL_DIR to "/tmp/yocto-autobuilder/downloads"
 Setting SSTATE_DIR to "/tmp/yocto-autobuilder/sstate"
 Setting IMAGE_TYPEFS to " tar.gz"
 Setting PERSISTDB_DIR to "/tmp/yocto-autobuilder/persistdb"
 Setting BB_NUMBER_THREADS to "16"
 Setting PARALLEL_MAKE to "16"
 Setting RESOLVE_TRIGGERED_HEAD to True
 Setting DEVKERNEL_MUT_REPO to "{'git://git.yoctoproject.org/poky-contrib':['
    stage/master_under_test', 'sgw/mut']}"
 Setting DEVKERNEL to "linux-yocto-dev"
 Setting ADTREPO_POPULATE to False
 Setting ADTREPO_DEV_POPULATE to True
 Setting ADTREPO_URL to "http://adtrepo.yoctoproject.org/"
 Setting ADTREPO_PATH to "/tmp/adtrepo.yoctoproject.org/"
 Setting ADTREPO_DEV_URL to "http://adtrepo-dev.yoctoproject.org/"
 Setting ADTREPO_DEV_PATH to "/tmp/adtrepo-dev/"
 Setting EMGD_DRIVER_DIR to "/tmp/yocto-autobuilder/emgd-driver"

twistd --no_save -y buildbot.tac
Removing stale pidfile /home/genius/test/yocto-autobuilder/yocto-controller/
    twistd.pid
twistd --no_save -y buildbot.tac
Removing stale pidfile /home/genius/test/yocto-autobuilder/yocto-worker/
    twistd.pid
fixing pemissions
fixing pemissions
Xtightvnc Stopped -  starting it

You will require a password to access your desktops.

Password:
Verify:
Would you like to enter a view-only password (y/n)? n
xauth:  file /home/genius/.Xauthority does not exist

New 'X' desktop is b5dfaa13e484:1

Creating default startup script /home/genius/.vnc/xstartup
Starting applications specified in /home/genius/.vnc/xstartup
Log file is /home/genius/.vnc/b5dfaa13e484:1.log

*** Running /etc/rc.local...
*** Booting runit daemon...
*** Runit started as PID 428
*** Running bash -l...
```

```
116 [ root@b5dfaa13e484:/ ]$
```

- In line 2 you need to enter the password of the account/user you'll utilize with docker.
- Line 5 suggests how to access the docker container via ssh and line 7 how to access it via a web interface. (We'll see more about this later.)
- In lines 101 and 102 enter `genius`, since that's the password of the `genius` user used inside the docker container
- Line 116 is a root prompt inside the docker container, which we will not really utilize for now

Yocto Autobuilder

Preparation

Now you could connect to the web interface of the Yocto Autobuilder, but we'll create new qemux86 related jobs. To do so we'll connect, as suggested in listing 5 line 5, via ssh to the docker container.

Listing 6: connect via ssh to the docker container

```
1 ssh -X genius@192.168.44.253 -p 49190
```

Note that in ip adress and port will be different in your case. You should see somthing along those lines:

Listing 7: connect via ssh to the docker container

```
1 The authenticity of host '[192.168.44.253]:49190 ([192.168.44.253]:49190)'
    can't be established.
2 ECDSA key fingerprint is 25:dd:3b:fc:87:cc:6d:1f:ae:2d:c2:dd:af:b4:7d:10.
3 Are you sure you want to continue connecting (yes/no)? yes
4 Warning: Permanently added '[192.168.44.253]:49190' (ECDSA) to the list of
    known hosts.
5 genius@192.168.44.253's password:
6 [ genius@b5dfaa13e484:~ ]$
```

- In lines 5 enter `genius` as a password
- Line 6 is the prompt of our `genius` user which we'll use to work with our Yocto Autobuilder docker container

I already prepared some qemux86 sample jobs, which we'll add to Yocto Autobuilder.

Listing 8: connect via ssh to the docker container

```
1 [ genius@b5dfaa13e484:~ ]$ cd /home/genius/test/yocto-autobuilder
2 [ genius@b5dfaa13e484:~/test/yocto-autobuilder {(b6f4f74...) *}]$ byobu
```

I open 3 shells in byobu (press F2). (With F3 and F4 you can navigate between the shells)

Listing 9: byobu shell 0: monitor Yocto Autobuilder

```
1 tailf yocto-worker/twistd.log
```

You should see something along those lines:

Listing 10: byobu shell 0: monitor Yocto Autobuilder - output

```
2014-07-17 13:56:16+0000 [Broker,client] Connected to localhost:9989; slave
    is ready
2014-07-17 13:56:16+0000 [Broker,client] sending application-level keepalives
     every 600 seconds
2014-07-17 14:06:16+0000 [-] sending app-level keepalive
2014-07-17 14:16:16+0000 [-] sending app-level keepalive
2014-07-17 14:26:16+0000 [-] sending app-level keepalive
2014-07-17 14:36:16+0000 [-] sending app-level keepalive
2014-07-17 14:46:16+0000 [-] sending app-level keepalive
2014-07-17 14:56:16+0000 [Broker,client] message from master: Received
    keepalive from master
2014-07-17 14:56:16+0000 [-] sending app-level keepalive
2014-07-17 15:06:16+0000 [-] sending app-level keepalive
```

Listing 11: byobu shell 1: add new build jobs

```
cd ~/test/yocto-autobuilder/buildset-config
cp  ~/test/meta-mainline/qemux86-ml/yocto-autobuilder/* .
vim yoctoAB.conf
```

add to it at the beginning the new jobs, and it should looks like this:

Listing 12: byobu shell 1: update yoctoAB.conf

```
[BuildSets]
order: ['custom-daisy-qemux86-core-image-minimal',
        'custom-daisy-qemux86-core-image-sato-sdk',
        'nightly', 'eclipse-plugin-juno', 'eclipse-plugin-kepler',
        'nightly-arm', 'nightly-arm-lsb', 'nightly-mips',
        'nightly-mips-lsb', 'nightly-ppc', 'nightly-ppc-lsb',
        'nightly-x86-64', 'nightly-x86-64-lsb', 'nightly-x86',
        'nightly-x86-lsb', 'nightly-x32', 'nightly-multilib',
        'nightly-world', 'nightly-non-gpl3', 'nightly-oecore',
        'nightly-intel-gpl', 'poky-tiny', 'build-appliance',
        'nightly-qa-extras', 'nightly-qa-systemd']
```

As you can see we added in line 2 instructions to build a `core-image-minimal` image for a `qemux86` machine utilizing `daisy` and on line 3 a `core-image-sato-sdk` image for the same machine and Yocto distro version.

Listing 13: custom-daisy-qemux86-core-image-minimal.conf

```
[custom-daisy-qemux86-core-image-minimal]
builders: 'builder1'
repos: [{'poky':
            {'repourl':'git://git.yoctoproject.org/poky',
             'layerversion':{'core':'meta', 'yoctobsp':'meta-yocto-bsp'},
             'branch':'master'}}]
steps: [{'SetDest':{}},
        {'CheckOutLayers': {}},
        {'RunPreamble': {}},
        {'GetDistroVersion' : {'distro': 'poky'}},
        {'CreateAutoConf': {'machine': 'qemux86', 'SDKMACHINE' : 'i686', '
           distro': 'poky', 'buildhistory' : True}},
        {'CreateBBLayersConf': {'buildprovider' : 'yocto'}},
        {'SyncPersistDB' : {'distro' : 'poky'}},
        {'BuildImages': {'images': 'core-image-minimal'}},
        {'RunSanityTests': {'images': 'core-image-minimal'}},
```

```
16         {'PublishArtifacts': {'artifacts': ['qemux86', 'md5sums']}}]
```

Listing 14: custom-daisy-qemux86-core-image-sato-sdk.conf

```
1  [custom-daisy-qemux86-core-image-sato-sdk]
2  builders: 'builder1'
3  repos: [{'poky':
4              {'repourl':'git://git.yoctoproject.org/poky',
5               'layerversion':{'core':'meta', 'yoctobsp':'meta-yocto-bsp'},
6               'branch':'master'}}]
7  steps: [{'SetDest':{}},
8          {'CheckOutLayers': {}},
9          {'RunPreamble': {}},
10         {'GetDistroVersion' : {'distro': 'poky'}},
11         {'CreateAutoConf': {'machine': 'qemux86', 'SDKMACHINE' : 'i686', '
               distro': 'poky', 'buildhistory' : True}},
12         {'CreateBBLayersConf': {'buildprovider' : 'yocto'}},
13         {'SyncPersistDB' : {'distro' : 'poky'}},
14         {'BuildImages': {'images': 'core-image-sato-sdk'}},
15         {'RunSanityTests': {'images': 'core-image-sato-sdk'}},
16         {'PublishArtifacts': {'artifacts': ['qemux86', 'md5sums']}}]
```

Listing 15: byobu shell 2: restart Yocto-Autobuilder

```
1  /etc/my_init.d/02_start_autobuilder.sh
```

Listing 16: byobu shell 2: restart Yocto-Autobuilder - output

```
1  starting yocto-autobuilder...
2  Password:
3  if [ -e twistd.pid ]; \
4          then kill `cat twistd.pid`; \
5          else echo "Nothing to stop."; \
6          fi
7  if [ -e twistd.pid ]; \
8          then kill `cat twistd.pid`; \
9          else echo "Nothing to stop."; \
10         fi
11 ######################################################################
12  Setting envvars.
13  In the future though please add the following to your shell environment:
14  PYTHONPATH=/home/genius/test/yocto-autobuilder/lib/python2.7/site-packages
       /:/home/genius/test/yocto-autobuilder/lib/python2.7/site-packages/
       autobuilder/:/home/genius/test/yocto-autobuilder/lib/python2.7/site-pack
15 ages/autobuilder/buildsteps:$PYTHONPATH
16  PATH=/home/genius/test/yocto-autobuilder/bin:/home/genius/test/yocto-
       autobuilder/yocto-autobuilder/scripts:$PATH
17  YOCTO_AB_CONFIG=/home/genius/test/yocto-autobuilder/buildset-config/yoctoAB.
       conf
18
19  If you don't, you should source this script everytime you want start the
20  autobuilder.
21  To start the autobuilder:
22  ./yocto-start-autobuilder <worker|controller|both>
23
24  To stop the autobuilder:
25  ./yocto-stop-autobuilder <worker|controller|both>
26
```

```
27
28 Reading /home/genius/test/yocto-autobuilder/config/autobuilder.conf
29
30  Setting WORKERBASEDIR to /home/genius/test/yocto-autobuilder/yocto-worker
31  Setting OPTIMIZED_GIT_CLONE to True
32  Setting OGIT_TRASH_DIR to /tmp/yocto-autobuilder/git/trash
33  Setting OGIT_MIRROR_DIR to /tmp/yocto-autobuilder/git/mirror
34  Setting OGIT_TRASH_CRON_TIME to "0 0 * * *"
35  Setting OGIT_TRASH_NICE_LEVEL to "19"
36  Setting BUILD_HISTORY_COLLECT to True
37  Setting BUILD_HISTORY_DIR to "/tmp/yocto-autobuilder/buildhistory"
38  Setting BUILD_HISTORY_REPO to "file:////tmp/yocto-autobuilder/buildhistory-
       repo"
39  Setting ERROR_REPORT_COLLECT to True
40  Setting PUBLISH_BUILDS to True
41  Setting PUBLISH_SOURCE_MIRROR to False
42  Setting PUBLISH_SSTATE to False
43  Setting MAINTAIN_PERSISTDB to True
44  Setting MACHINE_PUBLISH_DIR to "machines"
45  Setting BA_PUBLISH_DIR to "build-appliance"
46  Setting QEMU_PUBLISH_DIR to "/machines/qemu"
47  Setting RPM_PUBLISH_DIR to "rpm"
48  Setting DEB_PUBLISH_DIR to "deb"
49  Setting IPK_PUBLISH_DIR to "ipk"
50  Setting MARKED_RELEASE_DIR to "releases"
51  Setting ADT_INST_PUBLISH_DIR to "adt-installer"
52  Setting ADTQA_INST_PUBLISH_DIR to "adt-installer-QA"
53  Setting SSTATE_PUBLISH_DIR to "/tmp/yocto-autobuilder/sstate_mirror"
54  Setting SOURCE_PUBLISH_DIR to "/tmp/yocto-autobuilder/source"
55  Setting BUILD_PUBLISH_DIR to "/tmp/yocto-autobuilder/builds"
56  Setting X86TC_PUBLISH_DIR to "toolchain/i686"
57  Setting X8664TC_PUBLISH_DIR to "toolchain/x86_64"
58  Setting DL_DIR to "/tmp/yocto-autobuilder/downloads"
59  Setting SSTATE_DIR to "/tmp/yocto-autobuilder/sstate"
60  Setting IMAGE_TYPEFS to " tar.gz"
61  Setting PERSISTDB_DIR to "/tmp/yocto-autobuilder/persistdb"
62  Setting BB_NUMBER_THREADS to "16"
63  Setting PARALLEL_MAKE to "16"
64  Setting RESOLVE_TRIGGERED_HEAD to True
65  Setting DEVKERNEL_MUT_REPO to "{'git://git.yoctoproject.org/poky-contrib':['
       stage/master_under_test', 'sgw/mut']}"
66  Setting DEVKERNEL to "linux-yocto-dev"
67  Setting ADTREPO_POPULATE to False
68  Setting ADTREPO_DEV_POPULATE to True
69 Setting ADTREPO_URL to "http://adtrepo.yoctoproject.org/"
70  Setting ADTREPO_PATH to "/tmp/adtrepo.yoctoproject.org/"
71  Setting ADTREPO_DEV_URL to "http://adtrepo-dev.yoctoproject.org/"
72  Setting ADTREPO_DEV_PATH to "/tmp/adtrepo-dev/"
73  Setting EMGD_DRIVER_DIR to "/tmp/yocto-autobuilder/emgd-driver"
74
75 twistd --no_save -y buildbot.tac
76 twistd --no_save -y buildbot.tac
77 Xtightvnc already running
```

In line 2 enter genius as a password and monitor what happened with autobuilder:

Listing 17: byobu shell 0: monitor Yocto Autobuilder - output

```
1 ...
2 2014-07-17 15:27:17+0000 [Broker,client] lost remote
3 2014-07-17 15:27:17+0000 [Broker,client] lost remote
```

```
2014-07-17 15:27:17+0000 [Broker,client] lost remote
2014-07-17 15:27:17+0000 [Broker,client] lost remote
2014-07-17 15:27:17+0000 [Broker,client] lost remote
2014-07-17 15:27:17+0000 [Broker,client] lost remote
2014-07-17 15:27:17+0000 [Broker,client] Lost connection to localhost:9989
2014-07-17 15:27:17+0000 [Broker,client] Stopping factory <buildslave.bot.
    BotFactory instance at 0x2ad728243128>
2014-07-17 15:27:17+0000 [-] Main loop terminated.
2014-07-17 15:27:17+0000 [-] Server Shut Down.
2014-07-17 15:27:18+0000 [-] Log opened.
2014-07-17 15:27:18+0000 [-] twistd 12.2.0 (/usr/bin/python 2.7.6) starting
    up.
2014-07-17 15:27:18+0000 [-] reactor class: twisted.internet.epollreactor.
    EPollReactor.
2014-07-17 15:27:18+0000 [-] Starting BuildSlave -- version: 0.8.8
2014-07-17 15:27:18+0000 [-] recording hostname in twistd.hostname
2014-07-17 15:27:18+0000 [-] Starting factory <buildslave.bot.BotFactory
    instance at 0x2b45b1aa8248>
2014-07-17 15:27:18+0000 [-] Connecting to localhost:9989
2014-07-17 15:27:18+0000 [Uninitialized] Connection to localhost:9989 failed:
     Connection Refused
2014-07-17 15:27:18+0000 [Uninitialized] <twisted.internet.tcp.Connector
    instance at 0x2b45b1aa8560> will retry in 2 seconds
2014-07-17 15:27:18+0000 [Uninitialized] Stopping factory <buildslave.bot.
    BotFactory instance at 0x2b45b1aa8248>
2014-07-17 15:27:20+0000 [-] Starting factory <buildslave.bot.BotFactory
    instance at 0x2b45b1aa8248>
2014-07-17 15:27:20+0000 [-] Connecting to localhost:9989
2014-07-17 15:27:20+0000 [Broker,client] message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    fsl-ppc-lsb): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    x86-64): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(custom-
    daisy-qemux86-core-image-minimal): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(minnow):
    message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    arm): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    ppc): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(eclipse-
    plugin-juno): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    x86-lsb): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    x86-64-lsb): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    x32): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(eclipse-
    plugin-kepler): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
    -skeleton): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(custom-
    daisy-qemux86-core-image-sato-sdk): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    mips): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(minnow-lsb
    ): message from master: attached
2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
    non-gpl3): message from master: attached
```

```
41 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       fsl-arm-lsb): message from master: attached
42 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       world): message from master: attached
43 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
       -pam): message from master: attached
44 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
       -extras): message from master: attached
45 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       fsl-arm): message from master: attached
46 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       multilib): message from master: attached
47 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       world-uclibc): message from master: attached
48 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly):
       message from master: attached
49 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       fsl-ppc): message from master: attached
50 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       x86): message from master: attached
51 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
       -logrotate): message from master: attached
52 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       ppc-lsb): message from master: attached
53 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(buildtools
       ): message from master: attached
54 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
       -targetbuilds): message from master: attached
55 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       ipk): message from master: attached
56 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       oecore): message from master: attached
57 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       rpm): message from master: attached
58 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       arm-lsb): message from master: attached
59 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(poky-tiny)
       : message from master: attached
60 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       intel-gpl): message from master: attached
61 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-oe
       -selftest): message from master: attached
62 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(build-
       appliance): message from master: attached
63 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       deb): message from master: attached
64 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-qa
       -systemd): message from master: attached
65 2014-07-17 15:27:20+0000 [Broker,client] SlaveBuilder.remote_print(nightly-
       mips-lsb): message from master: attached
66 2014-07-17 15:27:20+0000 [Broker,client] Connected to localhost:9989; slave
       is ready
67 2014-07-17 15:27:20+0000 [Broker,client] sending application-level keepalives
        every 600 seconds
```

Build

- as suggested in listing 5 line 7 let's access the Yocto Autobuilder via a web browser. In my case I browse to `http://192.168.44.253:49191/`
- log in as you can see in figure 1 (username: `genius`/password: `genius`)

Figure 1: login in to the Yocto Autobuilder `genius`/`genius`

- click Waterfall
- click `custom-daisy-qemux86-core-image-minimal` as you can see in figure 2

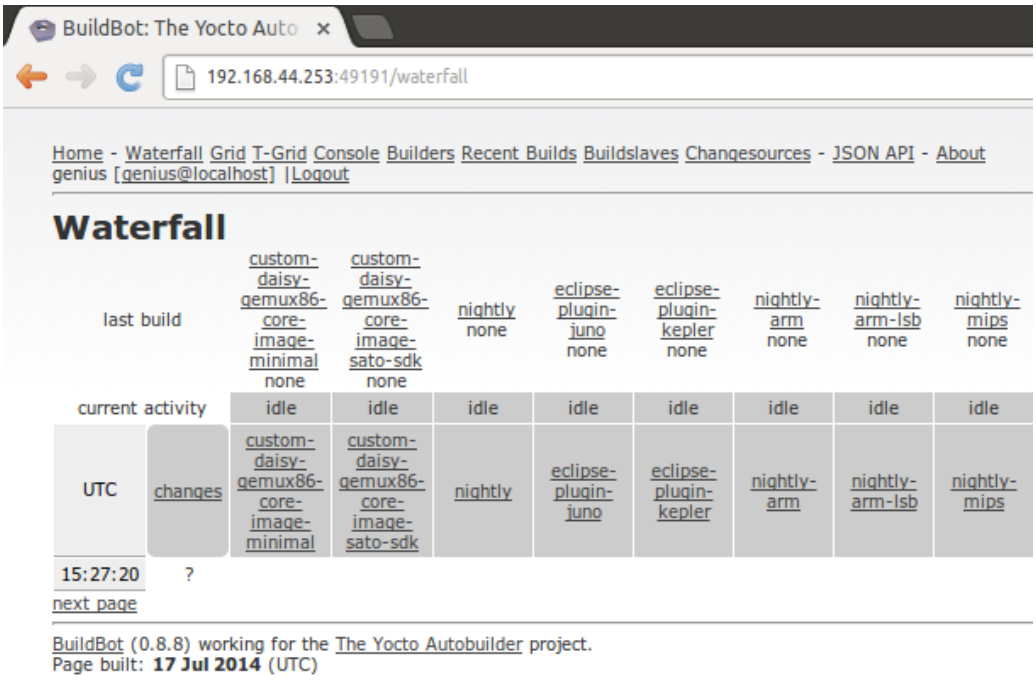

Figure 2: `core-image-minimal` and `core-image-sato-sdk` on the left

- click `Force Build` as you can see in figure 3

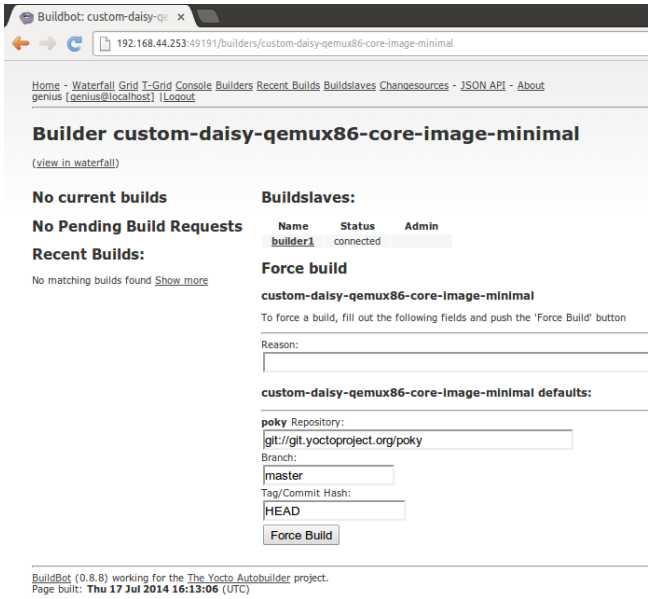

Figure 3: `core-image-minimal` Force Build

- click Waterfall
- click `custom-daisy-qemux86-core-image-sato-sdk` as you can see in figure 2
- click `Force Build` as you can see in figure 4

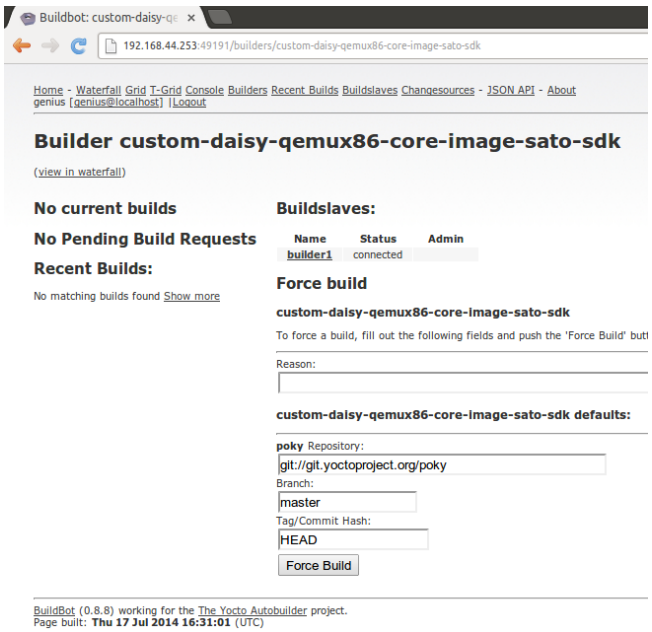

Figure 4: `core-image-minimal` Force Build

- initially you should see `core-image-minimal` building and `core-image-sato-sdk` pending as in figure 5

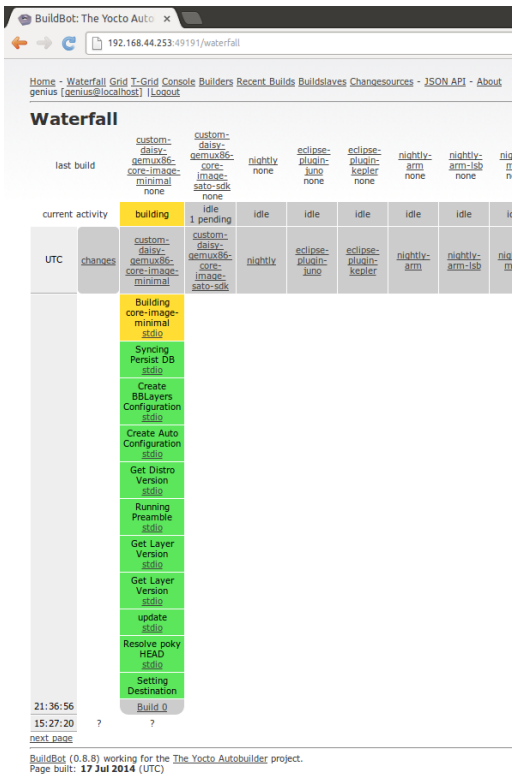

Figure 5: `core-image-minimal` building, `core-image-sato-sdk` pending

- if everything went well you should see something like in figure 6

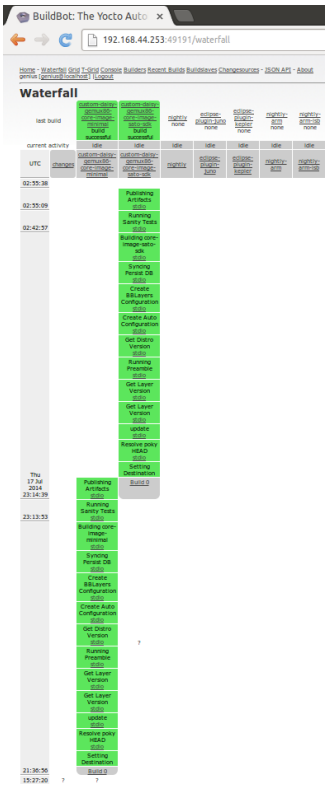

Figure 6: `core-image-minimal` and `core-image-sato-sdk` built

- Note this took for me `1 hrs, 37 mins, 42 secs + 3 hrs, 40 mins, 57 secs`

Run

- to make sure X-forwarding works start an xterm from a byobu shell

Listing 18: byobu shell 2: start xterm

```
1 xterm &
```

Listing 19: xterm: run qemux86 for core-image-minimal

```
1 cd ~/test/yocto-autobuilder/yocto-worker/custom-daisy-qemux86-core-image-
      minimal/build/
2 source oe-init-build-env
3 runqemu qemux86
```

Listing 20: xterm: run qemux86 for core-image-minimal - output

```
1 Continuing with the following parameters:
2 KERNEL: [/home/genius/test/yocto-autobuilder/yocto-worker/custom-daisy-
      qemux86-core-image-minimal/build/build/tmp/deploy/images/qemux86/bzImage-
      qemux86.bin]
3 ROOTFS: [/home/genius/test/yocto-autobuilder/yocto-worker/custom-daisy-
      qemux86-core-image-minimal/build/build/tmp/deploy/images/qemux86/core-
      image-minimal-qemux86-20140717213701.rootfs.ext3]
4 FSTYPE: [ext3]
5 Setting up tap interface under sudo
6 Acquiring lockfile for tap0...
7 Running qemu-system-i386...
8 /home/genius/test/yocto-autobuilder/yocto-worker/custom-daisy-qemux86-core-
      image-minimal/build/build/tmp/sysroots/x86_64-linux/usr/bin/qemu-system-
      i386 -kernel /home/genius/test/yocto-autobuilder/yocto-worker/custom-daisy
      -qemux86-core-image-minimal/build/build/tmp/deploy/images/qemux86/bzImage-
      qemux86.bin -net nic,vlan=0 -net tap,vlan=0,ifname=tap0,script=no,
      downscript=no -cpu qemu32 -hda /home/genius/test/yocto-autobuilder/yocto-
      worker/custom-daisy-qemux86-core-image-minimal/build/build/tmp/deploy/
      images/qemux86/core-image-minimal-qemux86-20140717213701.rootfs.ext3 -show
      -cursor -usb -usbdevice wacom-tablet -vga vmware -no-reboot -m 256 --
      append "vga=0 uvesafb.mode_option=640x480-32 root=/dev/hda rw mem=256M ip
      =192.168.7.2::192.168.7.1:255.255.255.0 oprofile.timer=1 "
```

- QEMU should pop up as you can see in figure 7
- If QEMU resizes just press the full screen button twice and you will see yocto booting

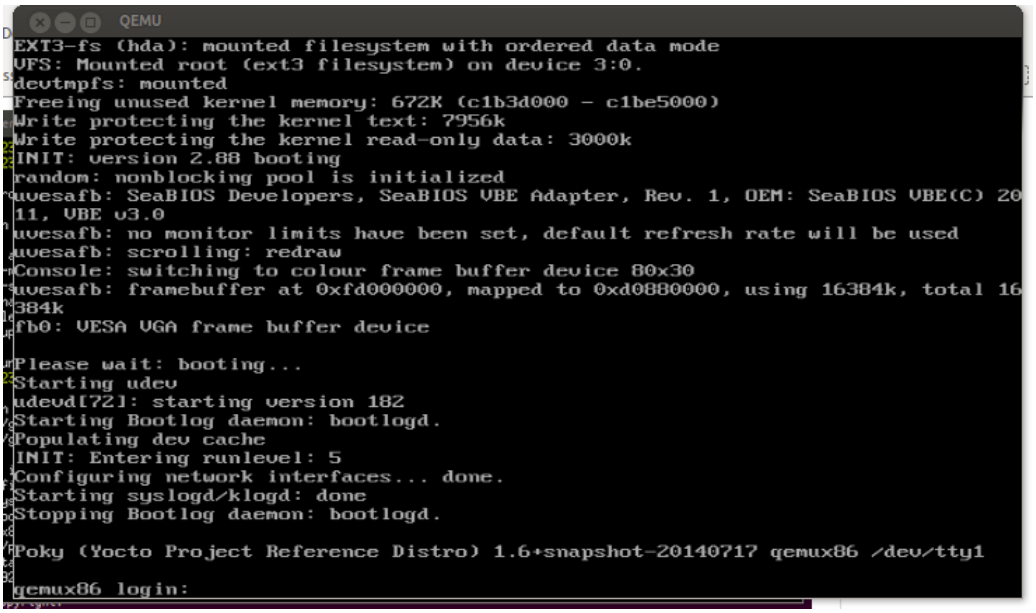

Figure 7: run `core-image-minimal`

- Now you can login to your qemux86 as a `root` user - no password
- if you want your mouse from QEMU back in Linux press CTRL+ALT
- issue `halt` to power down your qemux86
- I leave it as an exercise of the reader to run `core-image-sato-sdk`, figure 8 is how it should look like.

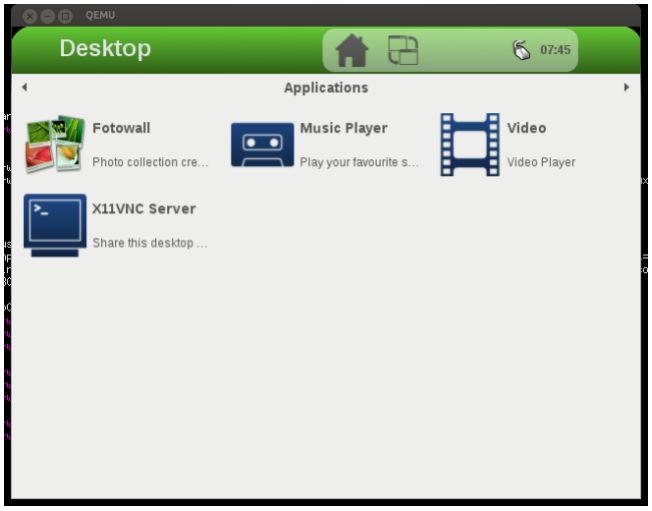

Figure 8: run `core-image-sato-sdk`

Snapshot

In order not to loose all the files which we downloaded/built up to know it would be a good idea to make a `docker commit`. The `id` we need to commit the currenty running docker container can be seen e.g. in the shell prompt of one of the ssh sessions we opened to the container. In listing 21 you can see that the `id` is `b5dfaa13e484`.

Listing 21: byobu shell 2: docker id

```
1 genius@b5dfaa13e484:~/test/yocto-autobuilder>
```

Listing 22: shell on docker host: check available docker images

```
1 docker images
```

Listing 23: shell on docker host: check available docker images - output

```
1 REPOSITORY                                    TAG     IMAGE ID      CREATED
      VIRTUAL SIZE
2 reliableembeddedsystems/yocto-autobuilder latest d208875f1b38 3 days ago
      1.196 GB
```

Listing 24: shell on docker host: commit changes

```
1 docker commit b5dfaa13e484 custom-daisy-qemux86-core-images-built
```

Listing 25: shell on docker host: check available docker images

```
1 REPOSITORY                                    TAG     IMAGE ID      CREATED
      VIRTUAL SIZE
2 custom-daisy-qemux86-core-images-built    latest a157740d1a9b 14 hours ago
      91.61 GB
3 reliableembeddedsystems/yocto-autobuilder latest d208875f1b38 3 days ago
      1.196 GB
```

As you can see in listing 25 we consumed `91.61 GB`, which means you need enough space on your disc and also that it will take quite some time to create the snapshot.

Exit

You can now exit from the docker container as you see in figure 26 and next time start up the snapshot.

Listing 26: docker container root shell: exit

```
1 [ root@b5dfaa13e484:/ ]$ exit
```

Listing 27: docker container root shell: exit - output

```
1 logout
2 *** bash exited with status 127.
3 *** Shutting down runit daemon (PID 428)...
4 *** Killing all processes...
5 *** Not all processes have exited in time. Forcing them to exit.
```

Start snapshot

Listing 28: docker host shell: start up snapshot

```
./docker_run.sh custom-daisy-qemux86-core-images-built br0
```

Please note that the `ssh` and `http` ports change every time you restart the docker container (because that's how I want it for now).

If you like this you might want to check out out training offers!

References

[1] *"Docker"*
`http://www.docker.com/`

[2] *"Yoctoproject"*
`https://www.yoctoproject.org/`

[3] *"Autobuilder"*
`https://www.yoctoproject.org/tools-resources/projects/autobuilder`

[4] *"QEMU"*
`http://wiki.qemu.org/Main_Page`

[5] *"What is Docker?"*
`http://www.docker.com/whatisdocker/`

[6] *"What is the Yocto Project?"*
`https://www.yoctoproject.org/about`

[7] *"Docker Installation"*
`https://docs.docker.com/installation/`

Trainings

- Introduction to Embedded Linux in Theory and Practice - a Crash Course (3 days)
- Embedded GNU/Linux Systems Architecture (5 days)
- Embedded GNU/Linux Kernel Internals and Device Drivers (5 days)
- (Embedded) Linux Debugging (3 days)
- FreeRTOS in Theory and Practice (3 days)
- Multicore Programming in C and C++ (3 days)

Training Delivery Options

- Public
- On-Site
- On-Line/Virtual

And combinations of the above!

Contact

ReliaBleEmBeddedSystems Consulting Training Engineering

Robert Berger

Embedded Software Specialist

email:
robert.berger@ReliableEmbeddedSystems.com
phone:
+43 (0) 699 17 69 07 19
web:
http://ReliableEmbeddedSystems.com

Building world class world wide win/win cooperations by helping you to create better embedded software!

Feedback

Please report comments/suggestions/issues regarding this document here:
`https://github.com/RobertBerger/articles-yocto/issues`

Author

Robert Berger consults and trains people all over the globe on a mission to help them create better embedded software. His specialties are trainings and consulting in the broad field of Embedded Software from small Real-time Systems to multi-core Embedded Linux.

For comments/inquiries and suggestions you can contact Robert here:

robert.berger@ReliableEmbeddedSystems.com