How to Build

A tutorial to set up your environment and Download Source Code

1. Build Setup Instructions

1.1. Setting up the Host Environment

Pre-Requisites

Requirement	Yocto 2.2 (Morty)	Yocto 3.1 LTS (Dunfell)
Linux	32/64 bit Ubuntu 16.04 LTS Precise supported distributions and versions are here	64 bit Ubuntu 18.04 LTS Precise supported distributions and versions are here
Free HDD Space	Minimum 100GB Free Memory	Minimum 100GB Free memory space
Oracle Virtual Box	5.0.40 or higher	-
Wireless Adapter	Brand Name: Tenda ralink & Model Number:W311MI TP-Link Archer T4U AC 1200	-
USB to Ethernet Switch	To connect with Ethernet Switch & Multiple Clients	
Host Tools version	• Git 1.8.3.1 or greater • tar 1.24 or greater • Python 2.7.3	• Git 1.8.3.1 or greater • tar 1.28 or greater • Python 3.5.0 or greater

1.1.1. Install the following packages for setting up your host VM

The instructions provided below are meant to be executed via the command line on an Ubuntu machine

for yocto 2.2 (morty)

```
# essential packages installation
# super user mode is required

# major essential packages
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib g++-multilib build-essential
chrpath socat bison curl
# supportive packages
sudo apt-get install libfile-slurp-perl libncurses-dev autoconf flex doxygen libtool automake libpcre3-dev
zlib1g-dev libbz2-dev subversion minicom putty libssl-dev rpm python-pexpect python-svn python-argparse vim
tofrodos meld dos2unix cmake uuid-dev ruby transfig libglib2.0-dev xutils-dev lynx-cur gperf autopoint python-
dulwich python-dev openjdk-7-jre
```

Yocto 2.2 (Morty)

Note : Please note openjdk-7-jre package is not available for Ubuntu-16.04 anymore. Presumably openjdk-8-jre should be used instead.

for yocto 3.1 (dunfell)

```
# essential packages installation
# super user mode is required

# major essential packages
sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib g++-multilib build-essential
chrpath socat bison curl cpio python3 python3-pip python3-pexpect xz-utils debianutils iputils-ping python3-git
python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 xterm
```

1.1.2. Configure bash as default command interpreter for shell scripts

```
sudo dpkg-reconfigure dash
```

Select “No”
To choose bash, when the prompt asks if you want to use dash as the default system shell - select “No”

1.1.3. Configure Git

Upgrade your Git version to 1.8.x or higher

On Ubuntu 16.04 LTS, if you are unable to upgrade your git version using apt-get, then follow the below steps in order to upgrade

```
sudo apt-get install software-properties-common
sudo add-apt-repository ppa:git-core/ppa
sudo apt-get update
sudo apt-get install git
```

Once git is installed, configure your name and email using the below commands

```
# review your existing configuration
git config --list --show-origin

# configure user name and email address
git config --global user.name "John Doe"
git config --global user.email johndoe@example.com

# configure git cookies. Needed for Gerrit to only contact the LDAP backend once.
git config --global http.cookieFile /tmp/gitcookie.txt
git config --global http.saveCookies true
```

1.1.4. Configure repo

In order to use Yocto build system, first you need to make sure that repo is properly installed on the machine:

```
# create a bin directory
mkdir ~/bin
export PATH=~/bin:$PATH

# Download the repo tool and ensure that it is executable
curl http://commondatastorage.googleapis.com/git-repo-downloads/repo > ~/bin/repo
chmod a+x ~/bin/repo
```

Trivia : Repo is a repository management tool that is built on top of Git. Its main purpose is to help manage projects that consist of many Git repositories, it can also be used to manage uploads to the CMF Gerrit instance and automate aspects of the development workflow.

Repo does not replace Git, it simply aids management of projects that contain multiple Git repositories into a single local working directory. Git will still be used for local operation such as commits etc.

Repo manages this for you by means of an XML based Manifest file. The Manifest file defines which repositories the project uses and links to appropriate revisions of each git repository, i.e where the upstream repositories reside and where they should be cloned locally. It is the manifest.xml (or default.xml) that determines which Git repositories and revisions repo will manage. This manifest.xml file is hosted in a Git repository along with all the other git repositories.

1.1.5. Credential configuration

Note: it is also recommended to put credentials in .netrc when interacting with the repo.

A sample .netrc file is illustrated below

```
machine code.rdkcentral.com
  login <YOUR_USERNAME>
  password <YOUR_PASSWORD>
```

1.2. Downloading Source Code & Building

1.2.1. Downloading Source Code

Following commands fetch the source code using repo tool

$ mkdir <Directory-Name> && cd <Directory-Name>

Please use the following repo init command

$ repo init -u https://user@code.rdkcentral.com/r/manifests -m manifest.xml -b <branch_name>

Examples :

repo init -u https://code.rdkcentral.com/r/manifests -m rdkb.xml -b rdkb-20180527

repo init -u https://code.rdkcentral.com/r/manifests -m rdkb.xml -b master

repo init -u https://code.rdkcentral.com/r/manifests -m rdkb.xml -b morty

$ repo sync --no-clone-bundle

- Cloning the code before login once to code.rdkcentral.com, user would get the Authentication error, even though the account is in good standing and has all the required access.
- Please login to code.rdkcentral.com before attempting to clone.

1.2.2. Building

$ source <setup-environment>

The above step configures and sets up your directory to start an appropriate build.

There are different kinds of builds listed. Please read the options and select the number of the build you need.

1) meta-rdk-bsp-emulator/conf/machine/qemuarmbroadband.conf
2) meta-rdk-bsp-emulator/conf/machine/qemux86broadband.conf
3) meta-rdk-bsp-emulator/conf/machine/qemux86hyb.conf
4) meta-rdk-bsp-emulator/conf/machine/qemux86mc.conf
5) openembedded-core/meta/conf/machine/qemuarm.conf
6) openembedded-core/meta/conf/machine/qemux86-64.conf
7) openembedded-core/meta/conf/machine/qemux86.conf

Next, you would need to initiate the build using the following command:

$ bitbake <image-name>

On Successful build, the ROOTFS (in vmdk format) would be available at the following reference location based on the build type :

-$ {HOME}/emulator/build-qemux86broadband/tmp/deploy/images/qemux86broadband/rdk-generic-broadband-image-qemux86broadband-<timestamp>.vmdk

Example :

…/build-qemux86broadband/tmp/deploy/images/qemux86broadband/rdk-generic-broadband-image-qemux86broadband-20160217080610.vmdk