Fuel@OPNFV build instructions

Release draft (34a852a)

OPNFV

August 18, 2016

CHAPTER

ONE

OPNFV BUILD INSTRUCTION FOR THE COLORADO RELEASE OF OPNFV WHEN USING FUEL AS A DEPLOYMENT TOOL

1.1 License

1.2 Abstract

This document describes how to build the Fuel deployment tool for the Colorado release of OPNFV build system, dependencies and required system resources.

1.3 Introduction

This document describes the build system used to build the Fuel deployment tool for the Colorado release of OPNFV, required dependencies and minimum requirements on the host to be used for the build system.

The Fuel build system is designed around Docker containers such that dependencies outside of the build system can be kept to a minimum. It also shields the host from any potential dangerous operations performed by the build system.

The audience of this document is assumed to have good knowledge in network and Unix/Linux administration.

1.4 Requirements

1.4.1 Minimum Hardware Requirements

- ~30 GB available disc
- 4 GB RAM

1.4.2 Minimum Software Requirements

The build host should run Ubuntu 14.04 operating system.

On the host, the following packages must be installed:

- An x86_64 host (Bare-metal or VM) with Ubuntu 14.04 LTS installed

  - **Note:** Builds on Wily (Ubuntu 15.x) are currently not supported
  - A kernel equal- or later than 3.19 (Vivid), simply available through

```
$ sudo apt-get install linux-generic-lts-vivid
```

- docker - see https://docs.docker.com/installation/ubuntulinux/ for installation notes for Ubuntu 14.04. Note: use the latest version from Docker (docker-engine) and not the one in Ubuntu 14.04.
- git (simply available through $ sudo apt-get install git)
- make (simply available through $ sudo apt-get install make)
- curl (simply available through $ sudo apt-get install curl)
- p7zip-full (simply available through $ sudo apt-get install p7zip-full)

1.5 Preparations

1.5.1 Setting up the Docker build container

After having installed Docker, add yourself to the docker group:

```
$ sudo usermod -a -G docker [userid]
```

Also make sure to define relevant DNS servers part of the global DNS chain in your </etc/default/docker> configuration file. Uncomment, and modify the values appropriately.

For example:

```
DOCKER_OPTS=" --dns=8.8.8.8 --dns=8.8.8.4"
```

Then restart docker:

```
$ sudo service docker restart
```

Setting up OPNFV Gerrit in order to being able to clone the code

- Start setting up OPNFV gerrit by creating a SSH key (unless you don't already have one), create one with ssh-keygen
- Add your generated public key in OPNFV Gerrit <https://gerrit.opnfv.org/> (this requires a Linux foundation account, create one if you do not already have one)
- Select "SSH Public Keys" to the left and then "Add Key" and paste your public key in.

Clone the OPNFV code Git repository with your SSH key

Now it is time to clone the code repository:

```
$ git clone ssh://<Linux foundation user>@gerrit.opnfv.org:29418/fuel
```

Now you should have the OPNFV fuel repository with the Fuel directories stored locally on your build host.

Check out the Colorado release:

```
$ cd fuel
$ git checkout colorado.1.0
```

Clone the OPNFV code Git repository without a SSH key

You can also opt to clone the code repository without a SSH key:

```
$ git clone https://gerrit.opnfv.org/gerrit/fuel
```

Make sure to checkout the release tag as described above.

1.5.2 Support for building behind a http/https/rsync proxy

The build system is able to make use of a web proxy setup if the http_proxy, https_proxy, no_proxy (if needed) and RSYNC_PROXY or RSYNC_CONNECT_PROG environment variables have been set before invoking make.

The proxy setup must permit port 80 (http), 443 (https) and 873 (rsync).

Important note about the host Docker daemon settings

The Docker daemon on the host must be configured to use the http proxy for it to be able to pull the base Ubuntu 14.04 image from the Docker registry before invoking make! In Ubuntu this is done by adding a line like:

```
export http_proxy="http://10.0.0.1:8888/"
```

to /etc/default/docker and restarting the Docker daemon.

Setting proxy environment variables prior to build

The build system will make use the following environment variables that needs to be exported to subshells by using export (bash) or setenv (csh/tcsh).

```
http_proxy (or HTTP_PROXY)
https_proxy (or HTTP_PROXY)
no_proxy (or NO_PROXY)
RSYNC_PROXY
RSYNC_CONNECT_PROG
```

As an example, these are the settings that were put in the user's .bashrc when verifying the proxy build functionality:

```
export RSYNC_PROXY=10.0.0.1:8888
export http_proxy=http://10.0.0.1:8888
export https_proxy=http://10.0.0.1:8888
export no_proxy=localhost,127.0.0.1,.consultron.com,.sock
```

Using a ssh proxy for the rsync connection

If the proxy setup is not allowing the rsync protocol, an alternative solution is to use a SSH tunnel to a machine capable of accessing the outbound port 873. Set the RSYNC_CONNECT_PROG according to the rsync manual page (for example to "ssh <username>@<hostname> nc %H 873") to enable this. Also note that netcat needs to be installed on the remote system!

Make sure that the ssh command also refers to the user on the remote system, as the command itself will be run from the Docker build container as the root user (but with the invoking user's SSH keys).

Disabling the Ubuntu repo cache if rsync is not allowed

During the build phase, a local Ubuntu package repository is fetched from upstream in order to be added to the OPNFV Fuel ISO and for parts of this process rsync is used.

If neither of the two available methods for proxying rsync are available, the last resort is to turn off the caching of the Ubuntu packages in the build system. This is done by removing the "f_repobuild" from SUBDIRS in the beginning of the fuel/build/f_isoroot/Makefile.

Note! Doing this will require the Fuel master node to have Internet access when installing the ISO artifact built as no Ubuntu package cache will be on the ISO!

1.5.3 Configure your build environment

** Configuring the build environment should not be performed if building standard Colorado release **

Select the versions of the components you want to build by editing the fuel/build/config.mk file.

1.5.4 Non official build: Selecting which plugins to build

In order to cut the build time for unofficial builds (made by an individual developer locally), the selection if which Fuel plugins to build (if any) can be done by environment variable "BUILD_FUEL_PLUGINS" prior to building.

Only the plugin targets from fuel/build/f_isoroot/Makefile that are specified in the environment variable will then be built. In order to completely disable the building of plugins, the environment variable is set to " ". When using this functionality, the resulting iso file will be prepended with the prefix "unofficial-" to clearly indicate that this is not a full build.

This method of plugin selection is not meant to be used from within Gerrit!

1.6 Building

There are two methods available for building Fuel:

- A low level method using Make
- An abstracted method using build.sh

1.6.1 Low level build method using make

The low level method is based on Make:

From the <fuel/build> directory, invoke <make [target]>

Following targets exist:

- none/all - this will:
  - Initialize the docker build environment
  - Build Fuel from upstream (as defined by fuel-build/config-spec)
  - Build the OPNFV defined plugins/features from upstream
  - Build the defined additions to fuel (as defined by the structure of this framework)
  - Apply changes and patches to fuel (as defined by the structure of this framework)

  - Reconstruct a fuel .iso image

- clean - this will remove all artifacts from earlier builds.
- debug - this will simply enter the build container without starting a build, from here you can start a build by enter “make iso”

If the build is successful, you will find the generated ISO file in the <fuel/build/release> subdirectory!

1.6.2 Abstracted build method using build.sh

The abstracted build method uses the <fuel/ci/build.sh> script which allows you to:

- Create and use a build cache - significantly speeding up the build time if upstream repositories have not changed.
- push/pull cache and artifacts to an arbitrary URI (http(s):, file:, ftp:)

For more info type <fuel/ci/build.sh -h>.

1.7 Artifacts

The artifacts produced are:

- <OPNFV_XXXX.iso> - Which represents the bootable Fuel image, XXXX is replaced with the build identity provided to the build system
- <OPNFV_XXXX.iso.txt> - Which holds version metadata.

1.8 References

1. OPNFV Installation instruction for the Colorado release of OPNFV when using Fuel as a deployment tool
2. OPNFV Build instruction for the Colorado release of OPNFV when using Fuel as a deployment tool
3. OPNFV Release Note for the Colorado release of OPNFV when using Fuel as a deployment tool