Adding a minnow board in fuego

Prerequisites:

The essential and graphical support packages you need for a supported Ubuntu or Debian distribution are shown in the following command:

```
$ sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib \
 build-essential chrpath socat libsdl1.2-dev xterm cpio
```

Use a locale setting which supports UTF-8 (such as LANG=en_US.UTF-8). Python can't change the filesystem locale after loading so we need a UTF-8 when Python starts or things won't work.

To fix this issue:

sudo apt-get install locales
sudo dpkg-reconfigure locales
add this to ~/.bashrc:
export LC_ALL=en_US.UTF-8
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
source ~/.bashrc

Python version:

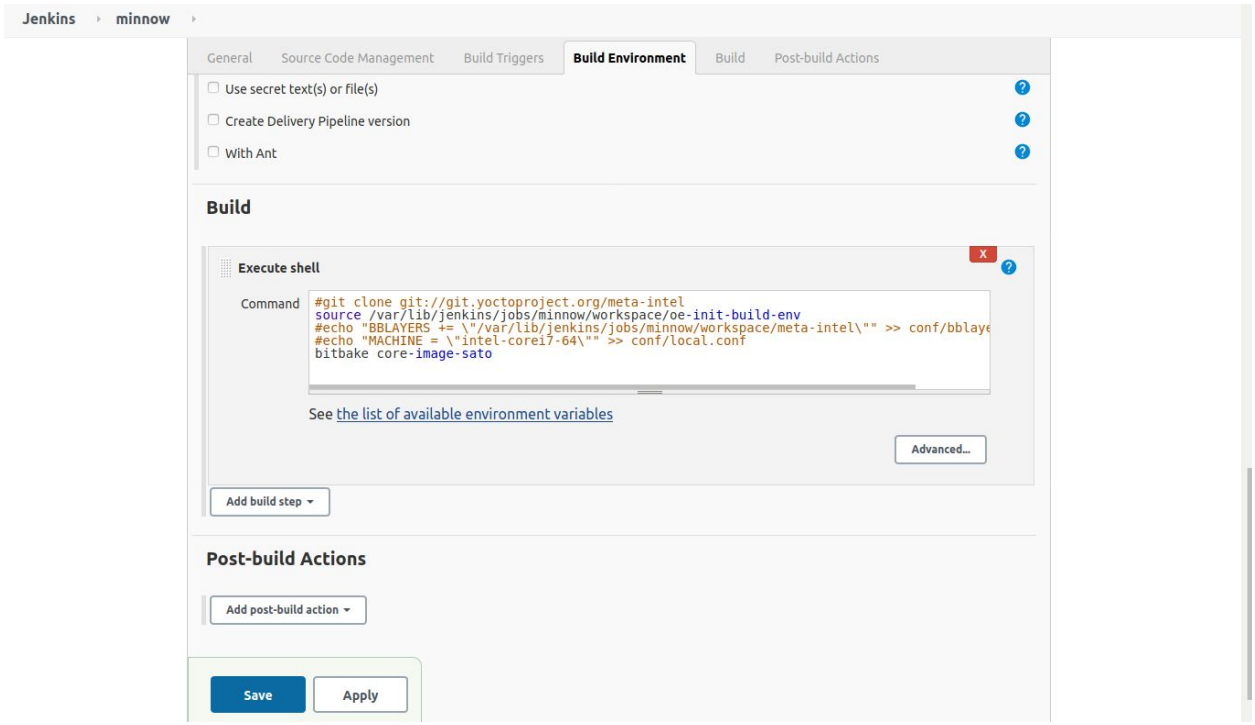
Python 3.5.0 or greater.

To build a image inside a container create new jenkins job through its UI as:

Jenkins Dashboard->New Item->(Job_Name)Freestyle Project->OK

This will create a workspace under /var/lib/jenkins/jobs/Job_Name/workspace.

Now you can configure a job through UI.
Write a commands required to build a image under execute shell option



Commands:

git clone git://git.yoctoproject.org/poky
git clone git://git.yoctoproject.org/meta-intel
source /var/lib/jenkins/jobs/minnow/workspace/oe-init-build-env
echo "BBLAYERS += \"/var/lib/jenkins/jobs/minnow/workspace/meta-intel\"" >> conf/bblayers.conf
echo "MACHINE = \"intel-corei7-64\"" >> conf/local.conf
bitbake core-image-sato

Once the image gets built successfully, image will be found under path

/var/lib/jenkins/jobs/minnow/workspace/build/tmp/deploy/intel-corei7-64/core-image-sato-intel-corei7-64.hddimg

Using the script mkefidisk.sh you can flash this image into sdcard

Command:

sudo ./mkefidisk.sh -v /dev/sdc build/tmp/deploy/images/intel-corei7-64/core-image-sato-intel-corei7-64.hddimg /dev/mmcblk2

Now the sd card is ready to boot the minnow board.
Insert the sd card into the minnow board and power it on, it will start booting from the sd card.

Adding a Minnow board:

Steps:

1. Make sure you can access the target via ssh, serial or some other connection
2. Decide whether to use an existing user account, or to create a user account specifically for testing
3. create a test directory on the target
4. create a board file (on the host)
5. add your board as a node in the Jenkins interface

3. create a test directory on the target

$ ssh root@your_target
<target>$ mkdir /home/fuego
<target>$ exit

4. create a board file (on the host)

The board files reside in <fuego-source-dir>/fuego-ro/boards, and each file has a filename with the name of the board, with the extension ".board".

$ cd fuego-ro/boards
$ cp template-dev.board MINNOW.board
$ vi MINNOW.board

Set board parameters: A board file has parameters which define how Fuego interacts with your board. Such as :

TRANSPORT, LOGIN and PASSWORD, IPADDR
ARCHITECTURE and TOOLCHAIN

MINNOW.board :

---

inherit "base-board"
include "base-params"

MACHINE=minnow

IPADDR="192.168.3.234"
SSH_PORT="22"
LOGIN="root"
BOARD_TESTDIR="/home/fuego/test"
PASSWORD=""
TOOLCHAIN="x86_64-poky-linux"
TRANSPORT="ssh"
ARCHITECTURE="x86_64"

---

5. add your board as a node in the Jenkins interface

To actually add the board as a node in jenkins, inside the docker container, run the following command at a shell prompt:

$ ftc add-nodes -b MINNOW

Adding a toolchain for minnow board:

Adding a toolchain to Fuego consists of these steps:

1. obtain (generate or retrieve) the toolchain

2. copy the toolchain to the container

3. install the toolchain inside the container

4. create a -tools.sh file for the toolchain

5. reference the toolchain in the appropriate board file

1. obtain (generate or retrieve) the toolchain

When you build an image in the Yocto Project, you can also build an SDK to go with that image using the '-c do_populate_sdk' build step with bitbake.

$ bitbake core-image-sato-sdk -c do_populate_sdk

This will generate sdk inside tmp/deploy/sdk folder

Install the SDK in the docker container:

To allow fuego to use the SDK, you need to install it into the fuego docker container. First, transfer the SDK into the container using docker cp.

With the container running, on the host machine do:

- docker ps (note the container id)
- docker cp tmp/deploy/sdk/<sdk-install-package> <container-id>:/tmp

sudo docker cp fuego-container:/var/lib/jenkins/jobs/minnow/workspace/build/tmp/deploy/sdk/poky-glibc-x86_64-core-image-sato-sdk-corei7-64-intel-corei7-64-toolchain-3.2+snapshot.sh

This last command will place the SDK install package into the /tmp directory in the container.

Now, install the SDK into the container, wherever you would like. Many toolchains install themselves under /opt.

At the shell inside the container, run the SDK install script (which is a self-extracting archive):

- /tmp/*poky-glibc-x86_64-core-image-sato-sdk-corei7-64-intel-corei7-64-toolchain-3.2+snapshot.sh*

  during the installation, select a toolchain installation location, like: /opt/poky/3.2+snapshot

Create a -tools.sh file for the toolchain:

The TOOLCHAIN variable is a string that is used to select the appropriate '<TOOLCHAIN>-tools.sh' file in /fuego-ro/toolchains.

x86_64-poky-linux-tools.sh:

```
# fuego toolchain script
# this sets up the environment needed for fuego to use a toolchain
# this includes the following variables:
# CC, CXX, CPP, CXXCPP, CONFIGURE_FLAGS, AS, LD, ARCH
# CROSS_COMPILE, PREFIX, HOST, SDKROOT
# CFLAGS and LDFLAGS are optional
#
# this script should be sourced by ${FUEGO_RO}/toolchains/tools.sh

POKY_SDK_ROOT=/opt/poky/3.2+snapshot
export SDKROOT=${POKY_SDK_ROOT}/sysroots/corei7-64-poky-linux

# the Yocto project environment setup script changes PATH so that python uses
# libs from sysroot, which is not what we want, so save the original path
# and use it later
ORIG_PATH=$PATH

PREFIX=corei7-64-poky-linux
source ${POKY_SDK_ROOT}/environment-setup-corei7-64-poky-linux

HOST=corei7-64-poky-linux

# don't use PYTHONHOME from environment setup script
unset PYTHONHOME
env -u PYTHONHOME >/dev/null
```

Reference the toolchain in a board file:

Edit the board file:

- *vi /fuego-ro/boards/MINNOW.board*

And add (or edit) the line:

- *TOOLCHAIN="x86_64-poky-linux"*