Continuous Integration for Building Software Infrastructure

Maciej Wyżliński

Outline

- Hello world pipeline
  - Basics of gitlab CI
- Automating build of a CentOS root file system
  - Configuring runners
  - Build a docker container
- Automating a cmake build
- Deployment to host
- Preparing a Petalinux docker image
- Automating build and deployment of Petalinux for zcu102
- Automating build of a custom kernel module

Repositories

https://gitlab.cern.ch/soc/ssh-private-key

https://gitlab.cern.ch/soc/ci-demo

https://gitlab.cern.ch/soc/centos-demo

https://gitlab.cern.ch/soc/petalinux-docker

https://gitlab.cern.ch/soc/petalinux-demo

https://gitlab.cern.ch/soc/petalinux-module-demo

CI hello world

Simple example:

```
# .gitlab-ci.yml

hello:
 stage: build
 script:
   - echo "Hello World"
```

CentOS root file system

```
#!/bin/bash
# install.sh
function dnf_cmd {
   dnf                              \
   -y                               \
   --forcearch=aarch64              \
   --releasever=7                   \
   --verbose                        \
   --installroot=/arm/sysroot       \
   --nogpgcheck                     \
   $@
}

dnf_cmd "groupinstall 'Minimal Install'"
dnf_cmd "install glibc-devel nfs-utils"

sed -e 's|root:\*|root:|g' -i
/arm/sysroot/etc/shadow
```

```
# .gitlab-ci.yml

build:
 stage: build
 tags:
   - shell
   - qemu
 script:
   - echo $CI_REGISTRY_PASSWORD | docker login -u $CI_REGISTRY_USER
--password-stdin $CI_REGISTRY
   - docker build -t $CI_REGISTRY_IMAGE/centos .
   - docker push $CI_REGISTRY_IMAGE/centos
```

```
# Dockerfile
FROM centos:7
RUN yum install -y dnf make rsync
# Install qemu
RUN curl -s -L
https://github.com/multiarch/qemu-user-static/releases/download/v4.0.0/
qemu-aarch64-static -o qemu-aarch64-static && chmod +x
qemu-aarch64-static
RUN mkdir -p /arm/sysroot/usr/local/bin && mv qemu-aarch64-static
/arm/sysroot/usr/local/bin
COPY install.sh /arm/
RUN /arm/install.sh
```

Configuring runner for building sysroot

```
mkdir -p /usr/local/bin && cd /usr/local/bin
wget https://github.com/multiarch/qemu-user-static/releases/download/v4.0.0/qemu-aarch64-static
chmod a+x qemu-aarch64-static
cat >  /etc/binfmt.d/qemu-aarch64.conf <<EOF
:qemu-aarch64:M::\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\xb7\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/local/bin/qemu-aarch64-static:
EOF
systemctl restart systemd-binfmt

# For CentOS8
:qemu-aarch64:M::\x7fELF\x02\x01\x01\x00\x00\x00\x00\x00\x00\x00\x00\x00\x02\x00\xb7\x00:\xff\xff\xff\xff\xff\xff\xff\x00\xff\xff\xff\xff\xff\xff\xff\xff\xfe\xff\xff\xff:/usr/local/bin/qemu-aarch64-static:FC

# see https://gitlab.cern.ch/atlas-tdaq-software/arm-cross-compiler
```

Adding cross compiler

```
# Extend Dockerfile
RUN mkdir -p /arm/compiler
RUN cd /arm/compiler && curl -s -L
https://developer.arm.com/-/media/Files/downloads/gnu-a/8.3-2019.03/binrel/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu.tar.
xz | tar Jxf -

ENV PATH=/cvmfs/sft.cern.ch/lcg/contrib/CMake/3.20.0/Linux-x86_64/bin:$PATH
ENV COMPILER=/arm/compiler/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu
ENV SYSROOT=/arm/sysroot

COPY aarch64-toolchain.cmake /arm
```

```
# aarch64-toolchain.cmake
set(CMAKE_SYSTEM_NAME Linux)
set(CMAKE_SYSTEM_PROCESSOR aarch64)
set(CMAKE_SYSROOT $ENV{SYSROOT})

set(tools $ENV{COMPILER})

set(CMAKE_C_COMPILER ${tools}/bin/aarch64-linux-gnu-gcc)
set(CMAKE_CXX_COMPILER ${tools}/bin/aarch64-linux-gnu-g++)

set(CMAKE_CXX_LINK_FLAGS " /cvmfs/sft.cern.ch/lcg/contrib/gcc/8/aarch64-centos7/lib64/libstdc++.so CACHE STRING "LDFLAGS" )
```

Building a c++ hello world

```cpp
// hello_world.cpp

#include <iostream>

int main() {
   std::cout<<"Hello World"<<std::endl;
}
```

```
# CMakeLists.txt

project(Test)

add_executable(hello-world hello_world.cpp)
```

```
# .gitlab-ci.yml
build:
 tags:
   - cvmfs
   - docker
 stage: build
 image: gitlab-registry.cern.ch/soc/centos-demo
 script:
   - mkdir build && cd build
   - cmake -D
CMAKE_TOOLCHAIN_FILE=/arm/aarch64-toolchain.cmake ..
   - make -j`nproc`
```

Deployment

- ssh-keygen -t ed25519 -f key
- Add the public key to /root/.ssh/authorized_keys
- Create a SSH_PRIVATE_KEY variable in CI/CD settings with the generated private key
- Create a HOSTS variable with a list of hosts you want to deploy to

```
.deploy:
 before_script:
   ##
   ## Install ssh-agent if not already installed, it is required by Docker.
   ##
   - 'command -v ssh-agent >/dev/null || yum install openssh-clients -y'

   ##
   ## Run ssh-agent (inside the build environment)
   ##
   - eval $(ssh-agent -s)

   ##
   ## Add the SSH key stored in SSH_PRIVATE_KEY variable to the agent store
   ## We're using tr to fix line endings which makes ed25519 keys work
   ## without extra base64 encoding.
   ## https://gitlab.com/gitlab-examples/ssh-private-key/issues/1#note_48526556
   ##
   - echo "$SSH_PRIVATE_KEY" | tr -d '\r' | ssh-add -

   ##
   ## Create the SSH directory and give it the right permissions
   ##
   - mkdir -p ~/.ssh
   - chmod 700 ~/.ssh
   - >
     for host in $HOSTS; do
       ssh-keyscan $host >> ~/.ssh/known_hosts
     done
   - chmod 644 ~/.ssh/known_hosts
```

Deploy

```
# ci hello world .gitlab-ci.yml
include:
'https://gitlab.cern.ch/soc/ssh-private-key/raw/master/ssh.yml'

build:
 tags:
   - cvmfs
   - docker
 stage: build
 image: gitlab-registry.cern.ch/mawyzlin/centos-minimal/centos
 script:
   - mkdir build && cd build
   - cmake -D CMAKE_TOOLCHAIN_FILE=/arm/aarch64-toolchain.cmake ..
   - make -j`nproc`
 artifacts:
   paths:
     - build/hello-world

deploy:
 stage: deploy
 extends:
   - .deploy
 script:
   - >
     for HOST in $HOSTS; do
       scp build/hello-world root@$HOST:/home/soc-workshop/software
     done
```

```
# centos .gitlab-ci.yml
include:
 - 'https://gitlab.cern.ch/soc/ssh-private-key/raw/master/ssh.yml'

build:
 stage: build
 tags:
   - shell
   - qemu
 script:
   - echo $CI_REGISTRY_PASSWORD | docker login -u $CI_REGISTRY_USER
--password-stdin $CI_REGISTRY
   - docker build -t $CI_REGISTRY_IMAGE/centos .
   - docker push $CI_REGISTRY_IMAGE/centos

deploy:
 stage: deploy
 extends:
   - .deploy
 tags:
   - docker
 image: $CI_REGISTRY_IMAGE/centos
 script:
   - >
     for HOST in $HOSTS; do
       rsync -a /arm/sysroot root@$HOST:/home/soc-workshop/
       rsync -a $COMPILER root@$HOST:/home/soc-workshop/
       rsync aarch64-toolchain.cmake root@$HOST:/home/soc-workshop/
     done
```

Petalinux

- Fork https://gitlab.cern.ch/soc/petalinux-docker
- Download petalinux installer and bsp https://www.xilinx.com/support/download/index.html/content/xilinx/en/downloadNav/embedded-design-tools/2020-2.html

```
docker login gitlab-registry.cern.ch
docker build --build-arg PETA_RUN_FILE=petalinux-v2020.2-final-installer.run -t
gitlab-registry.cern.ch/mawyzlin/petalinux-docker/petalinux-v2020.2 .
docker push gitlab-registry.cern.ch/mawyzlin/petalinux-docker/petalinux-v2020.2

petalinux-create -t project -s xilinx-zcu102-v2020.2-final.bsp
```

Petalinux

```
# Dockerfile
FROM gitlab-registry.cern.ch/soc/petalinux-docker/petalinux-v2020.2

RUN mkdir /home/vivado/petalinux-demo
COPY . /home/vivado/petalinux-demo

WORKDIR /home/vivado/petalinux-demo
RUN source /opt/Xilinx/petalinux/settings.sh && pwd && petalinux-build
```

```
# .gitlab-ci.yml
build:
 tags:
   - docker-image-build
 script:
   - echo $CI_REGISTRY_PASSWORD | docker login -u "$CI_REGISTRY_USER" --password-stdin $CI_REGISTRY
   - docker build -t $CI_REGISTRY_IMAGE .
   - docker push $CI_REGISTRY_IMAGE
```

Petalinux

- petalinux-build
- petalinux-boot --jtag --u-boot --tcl boot.tcl
- Add reset on top of the boot script
- Replace paths to point /home/soc-workshop/linux

```
# Add this on top of the boot.tcl script

targets -set -nocase -filter {name =~ "*PS TAP*"}
rst -srst
after 1000
```

```
include: 'https://gitlab.cern.ch/soc/ssh-private-key/raw/master/ssh.yml'

deploy:
 stage: deploy
 extends:
   - .deploy
 tags:
   - docker
 image: $CI_REGISTRY_IMAGE
 script:
   - >
     for HOST in $HOSTS; do
       rsync -a /home/vivado/petalinux-demo/images/linux root@$HOST:/home/soc-workshop/
       rsync -a /home/vivado/petalinux-demo/project-spec/hw-description/psu_init.tcl root@$HOST:/home/soc-workshop/linux
       rsync -a /home/vivado/petalinux-demo/images/linux/image.ub root@$HOST:/var/lib/tftpboot
       rsync boot.tcl root@$HOST:/home/soc-workshop/
     done
```

Petalinux

- Petalinux-config
  - Select nfs boot
- petalinux-config -c kernel
  - Disable initramfs
- Add CONFIG_BOOTCOMMAND to project-spec/meta-user/recipes-bsp/u-boot/files/bsp.cfg:

```
CONFIG_BOOTCOMMAND="setenv ipaddr 192.168.1.10 && setenv serverip 192.168.1.1 && setenv ethact ethernet@ff0e0000 && tftpboot 0x10000000 image.ub && setenv bootargs console=ttyPS0,115200 uio_pdrv_genirq.of_id=generic-uio clk_ignore_unused cpuidle.off=1 root=/dev/nfs rootfstype=nfs ip=192.168.1.10:::255.255.255.0:zcu102:eth0:on nfsroot=192.168.1.1:/home/soc-workshop/sysroot,nfsvers=3 nfsrootdebug rw rootwait && bootm"
```

- Set up nfs and tftp servers on the PC
- Then run

```
# On host
xsdb boot.tcl
# On zcu102
mount -t nfs 192.168.1.1:/home /home
/home/soc-workshop/software/hello-world
```

Petalinux kernel module

```
include: 'https://gitlab.cern.ch/soc/ssh-private-key/raw/master/ssh.yml'

build:
 image: gitlab-registry.cern.ch/mawyzlin/petalinux-demo:latest
 tags:
   - docker
 before_script:
   - source /opt/Xilinx/petalinux/settings.sh
 script:
   - mkdir -p /home/vivado/petalinux-demo/project-spec/meta-user/recipes-modules/hello-world
   - cp -r . /home/vivado/petalinux-demo/project-spec/meta-user/recipes-modules/hello-world
   - REPO_DIR=$PWD
   - cd /home/vivado/petalinux-demo
   - petalinux-build -c hello-world
   - cp /home/vivado/petalinux-demo/build/tmp/sysroots-components/zynqmp_generic/hello-world/lib/modules/5.4.0-xilinx-v2020.2/extra/hello-world.ko $REPO_DIR
 artifacts:
   paths:
     - hello-world.ko

deploy:
 stage: deploy
 extends:
   - .deploy
 script:
   - >
     for HOST in $HOSTS; do
       scp hello-world.ko root@$HOST:/home/soc-workshop/software/
     done
```

Local compilation

```
git clone https://:@gitlab.cern.ch:8443/soc/ci-demo.git
cd ci-demo
mkdir build && cd build
export COMPILER=/home/soc-workshop/gcc-arm-8.3-2019.03-x86_64-aarch64-linux-gnu
export SYSROOT=/home/soc-workshop/sysroot
cmake -D CMAKE_TOOLCHAIN_FILE=/home/soc-workshop/aarch64-toolchain.cmake ..
```