openSUSE Conference

Cross Building Factory & ALP

Using git, pbuild or OBS

adrian@opensuse.org

Cross Architecture Building

- No Emulator necessary (mostly)
- Build Host + Build Target environment needed
- Cross Compiler toolchain needed

Current State in Factory & ALP

- aarch64 and riscv64 toolchain exists
- kiwi live image creation works (via kiwi-crossprepare-plugin)
- Some(!) packages are working, eg. the kernel
- cross-aaa_base provides helper tooling

Example Setup

using an ALP prototype

Project setup in devel:ALP

- **NOTE: devel:ALP is a prototype only**

```
# osc meta prj devel:ALP
....
<scmsync>https://gitea.opensuse.org/adrianSuSE/Alp#factory</scmsync>
....
  <repository name="cross" rebuild="local" block="local">
    <path project="openSUSE:Factory:ARM" repository="standard"/>
    <path project="openSUSE:Factory:RISCV" repository="standard"/>
    <hostsystem project="openSUSE:Factory" repository="standard"/>
    <arch>aarch64</arch>
    <arch>riscv64</arch>
    <arch>x86_64</arch>
  </repository>
```

The git-way specials

- **package centric organisation**
  One git repo per package for all official revisions
- **Package sources come via submodules usually**
  Seperation of package source review and project aggregation
  binary rpm *is* reporting git resource via VCS header
  binary rpm *will be* reporting project source as well
- **Build config (prjconf) is part of git as _config file**
- **Only project meta is not part of git atm**
  project meta in OBS
  _pbuild in git for local builds using pbuild

pbuild 1/2

- part of build tool, install it via
  - zypper in build
  - git clone https://github.com/openSUSE/obs-build
- Works on
  - OBS project checkouts
  - Any other directory, eg. managed via git

pbuild 2/2

- It is a small OBS on your system
- The sources can be exchanged via OBS or any SCM
- Additional sources may get downloaded via the “Assets” mechanic
- Uses KVM when running as non-root

Cross Build using git & pbuild

- **Build a single rpm cross arch out of the large ALP project fast…**
- **Only one package source downloaded**

```
# git clone https://gitea.opensuse.org/adrianSuSE/Alp.git
# cd Alp
# git submodule init
# git submodule update xz
# cd xz

# pbuild –preset riscv64
```

Cross Build using git & pbuild

- **For the brave ones … building entire ALP also works…**

```
# git clone https://gitea.opensuse.org/adrianSuSE/Alp.git
# cd Alp
# git submodule init
# git submodule update

# pbuild –preset riscv64
```

Example Setup

Based on Leap and Factory

Cross Build using git & pbuild

- **It works across build types, eg. mixing rpm and kiwi build**

```
# git clone https://github.com/geckito/image-RaspBerryPi4-pi-hole
# cd image-RaspBerryPi4-pi-hole
# pbuild
```

Note: kiwi runs on x86_64, but uses qemu for executing rpm scripts

Build using osc & pbuild

- **How it almost already works ....**

```
# osc co openSUSE:Factory:ARM zstd

# cd openSUSE:Factory:ARM

# osc create-pbuild-config standard aarch64

WARNING: lacks still cross definition!

# pbuild –preset cross_aarch64
```

Adding cross build support to sources

On the example of spec files

Every package source is different...

cross-aaa_base-$arch provide generic helpers

- ENV: CC and CXX points to cross compiler
- rpm macros:
  %cross_sysroot directory
  %is_cross set to 1
  %_build %{_target_cpu}-suse-linux-gnu
  (also affects %configure and %cmake)
- Provides check for correct binary arch in build result

Add build config hints

Give OBS and pbuild a hint where a package is needed. Otherwise they get installed in target env. only. Eg:

#!OnlyNative: make

#!AlsoNative: Qt6

(also works via build config)

How to create an own setup

Building for a new device

Typical Requirements

1)Initialize new git repository

2)Create _pbuild

3)Pick to-be-rebuild or adapted source

4)Add own sources, eg. image description

5)Optional: add _config

Setup an own build - sources

```
# mkdir image-MyHardware

# cd image-MyHardware

# git init

# git submodule add https://gitea.opensuse.org/pool/kernel-source

# ln -sf kernel-source kernel-default
```

And steal a kiwi config as close as possible for your device

Also you most likely need to modify the kernel config

Setup an own build – build config

```
# cat > _pbuild <<EOF
<pbuild>
  <preset name="aarch64" default>
    <config>cross_aarch64</config>
    <config>tumbleweed</config>
    <hostrepo>https://download.opensuse.org/factory/repo/oss</hostrepo>
    <hostrepo>config:</hostrepo>
    <repo>https://download.opensuse.org/ports/aarch64/factory/repo/oss/</
repo>
    <arch>aarch64</arch>
  </preset>
</pbuild>
EOF

# pbuild
```

The pitfalls

pbuild hints

- Use latest version from openSUSE:Tools
- Results and logfiles are in _build.*/
- pbuild --single $package
  for live log debugging

Documentation: http://opensuse.github.io/obs-build/pbuild.html

Only aarch64 and riscv64 atm

- The other cross-*-gcc* do not support building against glibc atm!

Naming definitions

- **No common understanding of host, build and target...**

pbuild	host	target	-/-
cmake	HOST	default	-/-
GNU	build	host	target
rpm*	build	target	-/-

* Current state, older rpm versions have different definitions

Links

http://opensuse.github.io/obs-build/pbuild.html