FreeBSD on the Raspberry Pi ? Pt 3: How to host the package repository

Important Announcements on FreeBSD-armv6 packages

While initially writing this article, I had the idea to establish a service where packages can be selected to build for armv6. As of February 2016 this service is now online.
If you just need current FreeBSD packages for armv6, this is the place to visit. Otherwise, keep on reading.

Hi Folks, here's my writeup to conclude yesterdays post on crosscompiling arm6v packages for the Raspberry Pi.

Today I'm gonna quickly explain how to expose the packages repository via http from your build server.

This is part 3 of of my series of post related to FreeBSD on the Raspberry Pi.

So to summarize from my previous writeup, poudriere was used to crosscompile packages for armv6 platform. I used an x86 based build server to achieve this.

Here's the last output once again:

```
$ poudriere bulk -j arm102 -p ports_10_2_armv6 shells/bash
[....]
[02:40:23] ====>> Stopping 4 builders
[02:40:26] ====>> Creating pkgng repository
Creating repository in /tmp/packages: 100%
Packing files for repository: 100%
[02:40:33] ====>> Committing packages to repository
[02:40:33] ====>> Removing old packages
[02:40:33] ====>> Built ports: ports-mgmt/pkg print/indexinfo textproc/expat2 devel/m4 devel/gettext-runtime devel/gettext-tools lang/perl5.20 devel/bison shells/bash
[arm102-ports_10_2_arm6v] [2015-08-31_02h45m53s] [committing:] Queued: 9 Built: 9 Failed: 0 Skipped: 0 Ignored: 0 Tobuild: 0 Time: 02:40:31
[02:40:33] ====>> Logs: /build/poudriere//data/logs/bulk/arm102-ports_10_2_armv6/2015-08-31_02h45m53s
[02:40:33] ====>> Cleaning up
[02:40:33] ====>> Umounting file systems
```

Once this completes, poudriere will automatically generate a package repository.
The location for the repository will be set from the **poudriere.conf**'s POUDRIERE_DATA config option, which will be **/build/poudriere/data/**in this example. Poudriere will automatically create a subdirectory called **packages**. The repository will be created in a subdirectory derived from the jail and the ports tree used, here **arm102-ports_10_2_armv6**.

Let's have a quick look at this. Here's just to the top-level of the package repository. Poudriere will maintain a set of distinct release directories in there, designated by the **.real_timestamp** name. Also important to note is that there is a symlink called **.latest** which links to the latest release.
So in the end, **/build/poudriere/data/packages/arm102-ports_10_2_armv6/.latest** directory is the one which we will be serving up from a web-server.

```
[root@pbsp-bsd-001 /build/poudriere/data/packages/arm102-ports_10_2_armv6]# ls -la
total 8
drwxr-xr-x 4 root wheel 10 Aug 31 22:16 .
drwxr-xr-x 3 root wheel 3 Aug 31 22:01 ..
drwxr-xr-x 4 root wheel 8 Sep 1 23:17 .building
lrwxr-xr-x 1 root wheel 16 Aug 31 05:26 .latest -> .real_1440991586
drwxr-xr-x 4 root wheel 8 Aug 31 05:26 .real_1440991586
lrwxr-xr-x 1 root wheel 11 Aug 31 05:26 All -> .latest/All
lrwxr-xr-x 1 root wheel 14 Aug 31 05:26 Latest -> .latest/Latest
lrwxr-xr-x 1 root wheel 19 Aug 31 05:26 digests.txz -> .latest/digests.txz
lrwxr-xr-x 1 root wheel 16 Aug 31 05:26 meta.txz -> .latest/meta.txz
lrwxr-xr-x 1 root wheel 23 Aug 31 05:26 packagesite.txz -> .latest/packagesite.txz
```

Installing lighttpd Web-Server

Choosing a web-server is always a little of a debate. If you have python available, you can quickly and easily serve up files via http, also Apache and nginx are good choices.
I tend to use lighttpd for simple things, as it still sports everything a web-servers needs to have, while not beeing like a bloated monster as it's easy to setup and get going in under five minutes.

So if you want to build it from the ports, go ahead and install it like this on your build server:

```
cd /usr/ports/www/lighttpd
make config install clean
```

Then add the line **lighttpd_enable="YES"** to **/etc/rc.conf**.
But before you actually start lighttpd, open up **/usr/local/etc/lighttpd/modules.conf** in your editor of choice.
Look out for a section like this:

```
server.modules = (
"mod_access",
# "mod_alias",
# "mod_auth",
# "mod_evasive",
# "mod_redirect",
# "mod_rewrite",
# "mod_setenv",
# "mod_usertrack",
)
```

Uncomment the **mod_alias** keyword, then save the change and exit.

```
server.modules = (
"mod_access",
"mod_alias",
# "mod_auth",
# "mod_evasive",
# "mod_redirect",
# "mod_rewrite",
# "mod_setenv",
# "mod_usertrack",
)
```

Now edit **/usr/local/etc/lighttpd/lighttpd.conf**.
Go to the end of file and locate this section:

```
$SERVER["socket"] == "0.0.0.0:80" { }
```

Edit like this:

```
$SERVER["socket"] == "0.0.0.0:80" {
alias.url += ( "/FreeBSD:10:armv6" => "/build/poudriere/data/packages/arm102-ports_10_2_arm6v/" )

# optional: to enable dirlisting generation, set dir-listing.activate = "enable" below
$HTTP["url"] =~ "/FreeBSD:10:armv6($|/.*$)" {
dir-listing.activate = "disable"
}
}
```

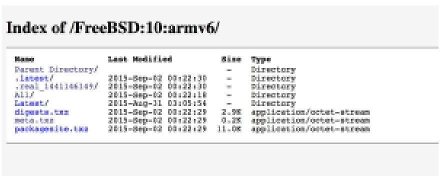
Now create the default document root and set default permissions.

```
mkdir -p /usr/local/www/data
chown www:www /usr/local/www/data
```

Now it's time to start lighttpd:

```
service lighttpd start
```

If all went well, you should be able to call up your web-server on the **http://IP-ADDRESS/FreeBSD:10:armv6 URL.**



Index of /FreeBSD:10:armv6/

```
Name                 Last Modified          Size  Type
Parent Directory/                            -    Directory
.latest/             2015-Sep-02 00:22:30    -    Directory
.real_1441146149/    2015-Sep-02 00:22:30    -    Directory
All/                 2015-Sep-02 00:22:18    -    Directory
Latest/              2015-Aug-31 03:05:54    -    Directory
digests.txz          2015-Sep-02 00:22:29   2.9K  application/octet-stream
meta.txz             2015-Sep-02 00:22:29   0.2K  application/octet-stream
packagesite.txz      2015-Sep-02 00:22:29  11.0K  application/octet-stream
```

The URL may look strange but that's exactly what you need. It's the ABI naming syntax that FreeBSD's pkg utility uses internally.

Configure pkg on the Raspi

So far, so good. Hop on to your Raspberry Pi and change to /etc/pkg directory.

If you previously configured the **uzsolt** repository as per my first post, open up that file and disable the repository by setting **enabled: no**. Alternatively you can also remove the file.

Register your new repository by adding a new file at **/etc/pkg/armv6.conf**.

```
armv6: {
url: "http://IP-ADDRESS/${ABI}/",
mirror_type: NONE,
signature_type: none,
enabled: yes
}
```

Update the repository to fetch the new package list.

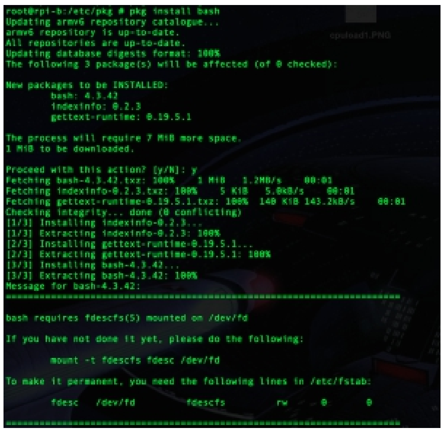
```
root@rpi-b:/etc/pkg # pkg update
Updating armv6 repository catalogue...
Fetching meta.txz: 100% 264 B 0.3kB/s 00:01
Fetching packagesite.txz: 100% 11 KiB 11.3kB/s 00:01
Processing entries: 100%
armv6 repository update completed. 30 packages processed.
```

Then all packages in the repository can be looked up by calling up **pkg search**.

```
pkg search -g '*'
```

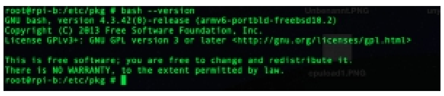
Remember we built **shells/bash**. It's about time to see if it actually worked.

```
pkg install bash
```



```
root@rpi-b:/etc/pkg # pkg install bash
Updating armv6 repository catalogue...
armv6 repository is up-to-date.
All repositories are up-to-date.
Updating database digests format: 100%
The following 3 package(s) will be affected (of 0 checked):

New packages to be INSTALLED:
        bash: 4.3.42
        indexinfo: 0.2.3
        gettext-runtime: 0.19.5.1

The process will require 7 MiB more space.
1 MiB to be downloaded.

Proceed with this action? [y/N]: y
Fetching bash-4.3.42.txz: 100%    1 MiB   1.2MB/s    00:01
Fetching indexinfo-0.2.3.txz: 100%    5 KiB   5.0kB/s    00:01
Fetching gettext-runtime-0.19.5.1.txz: 100%  140 KiB 143.2kB/s    00:01
Checking integrity... done (0 conflicting)
[1/3] Installing indexinfo-0.2.3...
[1/3] Extracting indexinfo-0.2.3: 100%
[2/3] Installing gettext-runtime-0.19.5.1...
[2/3] Extracting gettext-runtime-0.19.5.1: 100%
[3/3] Installing bash-4.3.42...
[3/3] Extracting bash-4.3.42: 100%
Message for bash-4.3.42:
======================================================================

bash requires fdescfs(5) mounted on /dev/fd

If you have not done it yet, please do the following:

        mount -t fdescfs fdesc /dev/fd

To make it permanent, you need the following lines in /etc/fstab:

        fdesc   /dev/fd         fdescfs         rw      0       0

======================================================================
```

Run the **bash --version** command to quickly see that it really works.



```
root@rpi-b:/etc/pkg # bash --version
GNU bash, version 4.3.42(0)-release (armv6-portbld-freebsd10.2)
Copyright (C) 2013 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>

This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.
root@rpi-b:/etc/pkg #
```

That's it! Now you are capable of cross-compiling packages for FreeBSD-armv6 and make them installable via a web repository the same way as on any FreeBSD tier 1 platform.

My next upcoming article will the _finally_ focus on how to actually build the WiFi access point on FreeBSD.
But guess what, in order to achieve this I first needed the packages ... ;-)