NAME

pb, aka project−builder.org − builds packages for your projects

DESCRIPTION

pb helps you build various packages directly from your project sources. Those sources could be handled by a CMS (Configuration Management System) such as Subversion, CVS, Git, Mercurial... or being a simple reference to a compressed tar file. It's based on a set of configuration files, a set of provided macros to help you keeping build files as generic as possible. For example, a single .spec file should be required to generate for all rpm based distributions, even if you could also have multiple .spec files if required.

SYNOPSIS

pb [−vhSq][−r pbroot][−p project][[−s script −a account −P port][−T VEtype][−t [os−ver−arch]][−m os−ver−arch[,...]]][−g][−i image] <action> [<pkg1> ...]

pb [−−verbose][−−help][−−man][−−quiet][−−snapshot][−−revision pbroot][−−project project][[−−script script −−account account −−port port][−T VEtype][−−target [os−ver−arch]][−−machine os−ver−arch[,...]]][−−nographic][−−image image][−−rebuild] <action> [<pkg1> ...]

OPTIONS

−v|−−verbose
Increase verbosity

−q|−−quiet
Do not print any output.

−h|−−help
Print a brief help message and exits.

−S|−−snapshot
Use the snapshot mode of VMs or VEs

−−man
Prints the manual page and exits.

−t|−−target os-ver-arch
Name of the target system you want to build for. All if none precised.

−m|−−machine os−ver−arch[,os−ver−arch,...]
Name of the Virtual Machines (VM), Virtual Environments (VE) or Remote Machines (RM) you want to build on (comma separated). All if none precised (or use the env variable PBV).

−T|−−vetype VEtype]
Type of Virtual Environments (VE) Can be chroot or docker.

−s|−−script script
Name of the script you want to execute on the related VMs/VEs/RMs.

−g|−−nographic
Do not launch VMs in graphical mode.

−i|−−image image
It could be either: − The name of the ISO image of the distribution you want to install on the related VMs − The name of the docker image of the distribution you want to install on the related VEs

−a|−−account account
Name of the account to use to connect on the related VMs/RMs.

−P|−−port port_number
Port number to use to connect on the related VMs/RMs.";

−p|−−project project_name
Name of the project you're working on (or use the env variable pb)

−r|−−revision revision
Path Name of the project revision under the CMS (or use the env variable PBROOT)

−V|−−version new_version
New version of the project to create based on the current one.

−k|−−keep
Keep the temporary dir where files have been created in or der to help debug

−−rebuild
Only valid with the checkssh action, it alllows to automatically relaunch the build of the failed packages

−−no−stop−on−error
Continue through errors with best effort.

ARGUMENTS

<action> can be:

sbx2build
Create tar files for the project under your CMS. Current state of the exported content is taken. CMS supported are SVN, SVK, CVS, Git and Mercurial parameters are packages to build if not using default list

cms2build
Create tar files for the project under your CMS. Current state of the CMS is taken. CMS supported are SVN, SVK, CVS, Git and Mercurial parameters are packages to build if not using default list

build2prep
Prepare the environment for build by installing required dependencies. Done once on the build system.

build2pkg
Create packages for your running distribution

cms2pkg
cms2build + build2pkg

sbx2pkg
sbx2build + build2pkg

sbx2pkg2ins
sbx2pkg + final install of packages

build2ssh
Send the tar files to a SSH host

sbx2ssh
sbx2build + build2ssh

cms2ssh
cms2build + build2ssh

pkg2ssh
Send the packages built to a SSH host

build2vm
Create packages in VMs, launching them if needed and send those packages to a SSH host once built VM type supported are QEMU and KVM

build2ve
Create packages in VEs, creating it if needed and send those packages to a SSH host once built

build2rm
Create packages in RMs, which should pre-exist, and send those packages to a SSH host once built RM means Remote Machine, and could be a physical or Virtual one. This is one buildfarm integration for pb.

prepvm
Prepare the VMs to have all requirements to build the project

prepve
Prepare the VEs to have all requirements to build the project

preprm
Prepare the RMs to have all requirements to build the project

sbx2vm
sbx2build + build2vm

sbx2ve
sbx2build + build2ve

sbx2docker
sbx2build + build2ve with a potential build of all necessary docker containers to perform it

sbx2rm
sbx2build + build2rm

cms2vm
cms2build + build2vm

cms2ve
cms2build + build2ve

cms2rm
cms2build + build2rm

launchvm
Launch one virtual machine

launchve
Launch one virtual environment

script2vm
Launch one virtual machine if needed and executes a script on it

script2ve
Execute a script in a virtual environment

script2rm
Execute a script on a remote machine

newvm
Create a new virtual machine

newve
Create a new virtual environment

setupvm
Setup a virtual machine for pb usage

setupve
Setup a virtual environment for pb usage

setuprm
Setup a remote machine for pb usage

sbx2setupvm
Setup a virtual machine for pb usage using the sandbox version of pb instead of the latest stable Reserved to dev team.

sbx2setupve
Setup a virtual environment for pb usage using the sandbox version of pb instead of the latest stable Reserved to dev team.

sbx2setuprm
Setup a remote machine for pb usage using the sandbox version of pb instead of the latest stable Reserved to dev team.

build2setupvm
Setup a virtual machine for pb usage using the build available Reserved to dev team.

build2setupve
Setup a virtual environment for pb usage using the build available Reserved to dev team.

build2setuprm
Setup a remote machine for pb usage using the build available Reserved to dev team.

snapvm
Snapshot a virtual machine for pb usage

snapve
Snapshot a virtual environment for pb usage

updatevm
Update the distribution in the virtual machine

updateve
Update the distribution in the virtual environment

updaterm
Update the distribution in the remote machine

test2pkg
Test a package locally

test2vm
Test a package in a virtual machine

test2ve
Test a package in a virtual environment

test2rm
Test a package in a remote machine

checkssh
Check the delivery of the packages on the repository

checkps
Check the process running the VM concerned

newver
Create a new version of the project derived from the current one

newproj
Create a new project and a template set of configuration files under pbconf

announce
Announce the availability of the project through various means

sbx2webssh
Create tar files for the website under your CMS. Current state of the exported content is taken. Deliver the content to the target server using ssh from the exported dir.

cms2webssh
Create tar files for the website from your CMS. Deliver the content to the target server using ssh from the DVCS.

sbx2webpkg
Create tar files for the website under your CMS. Current state of the exported content is taken.

cms2webpkg
Create tar files for the website under your CMS.

getconf
Print the full configuration parameters as found in the various configuration files. Help to debug conf issues. Also accepts a parameter to display only this value, and a VM/VE/RM

getvar
Print the full variables expanded based on the distrubution tuple. Help to debug conf issues. Also accepts a parameter to display only the values for this package, and a VM/VE/RM

clean
Purge the build and delivery directories related to the current project

cleanssh
Purge the ssh server of its packages (only for testver and test packages)

<pkgs> can be a list of packages, the keyword 'all' or nothing, in which case the default list of packages is taken (corresponding to the defpkgdir list of arguments in the configuration file).

WEB SITES

The main Web site of the project is available at <http://www.project−builder.org/>. Bug reports should be filled using the trac instance of the project at <http://trac.project−builder.org/>.

USER MAILING LIST

None exists for the moment.

ENVIRONMENT VARIABLES

PBACCOUNT
Login to use to connect to the VM/RM, undef by default

CONFIGURATION FILES

Each pb user may have a configuration in *$HOME/.pbrc*. The values in this file may overwrite any other configuration file value.

Here is an example of such a configuration file:

```
#
# Define for each project the URL of its pbconf repository
# No default option allowed here as they need to be all different
#
# URL of the pbconf content
# This is the format of a classical URL with the extension of additional schema suc
# svn+ssh, cvs+ssh, ...
#
pbconfurl linuxcoe = cvs+ssh://:ext:bcornec@linuxcoe.cvs.sourceforge.net:/cvsroot/l

# This is normaly defined in the project's configuration file
# Url of the project
#
pburl linuxcoe = cvs+ssh://:ext:bcornec@linuxcoe.cvs.sourceforge.net:/cvsroot/linux

# All these URLs needs to be defined here as the are the entry point
# for how to build packages for the project
#
pbconfurl pb = svn+ssh://svn.project−builder.org/mondo/svn/pb/pbconf
pbconfurl mondorescue = svn+ssh://svn.project−builder.org/mondo/svn/project−builder
pbconfurl collectl = svn+ssh://bruno@svn.mondorescue.org/mondo/svn/project−builder/
pbconfurl netperf = svn+ssh://svn.mondorescue.org/mondo/svn/project−builder/netperf

# Under that dir will take place everything related to pb
```

```
# If you want to use VMs/chroot/..., then use $ENV{'HOME'} to make it portable
# to your VMs/chroot/...
# if not defined then /var/cache
pbdefdir default = $ENV{'HOME'}/project-builder
pbdefdir pb = $ENV{'HOME'}
pbdefdir linuxcoe = $ENV{'HOME'}/LinuxCOE/cvs
pbdefdir mondorescue = $ENV{'HOME'}/mondo/svn

# pbconfdir points to the directory where the CMS content of the pbconfurl is check
# If not defined, pbconfdir is under pbdefdir/pbproj/pbconf
pbconfdir linuxcoe = $ENV{'HOME'}/LinuxCOE/cvs/pbconf
pbconfdir mondorescue = $ENV{'HOME'}/mondo/svn/pbconf

# pbdir points to the directory where the CMS content of the pburl is checked out
# If not defined, pbdir is under pbdefdir/pbproj
# Only defined if we have access to the dev of the project
pbdir linuxcoe = $ENV{'HOME'}/LinuxCOE/cvs
pbdir mondorescue = $ENV{'HOME'}/mondo/svn

# -daemonize doesn't work with qemu 0.8.2
vmopt default = -m 384
```

COMMAND DETAILS

newproj

The newproj command creates a new project-builder project. To run this command you first need to define two variables in your ~/.pbrc file:

```
pbconfurl I<project> = file:///home/anderse/.git/project-builder-config/I<project>
pbdefdir default = $ENV{HOME}/cache-project-builder
```

The first line defines the version controlled configuration information and the second defines the root directory for project-builder to use.

You can then run the command:

```
% pb -p I<$project> -r I<$version> newproj I<$pkg>
```

to create the new project. Running the newproj command will then generate the file `$pbdefdir`/$project/pbconf/$version/$project.pb, and the directory `$pbdefdir`/$project/pbconf/$version/$pkg. You will need to edit those files to make the later commands work.

cms2build

The cms2build command takes your files from the content management system and makes the two tar files that are necessary for building files. You need to have run the newproj command first. Then there are several steps for running this command:

Update your `$project`.pb configuration file.

You need to set the pburl, pbrepo, pbwf, pbpackager, projver, projtag, testver, delivery, and defpkgdir lines as described in the configuration file. The pburl entry is used to find the source for your package. The pbrepo entry is used to build the .repo or .sources.list files for use by downloaders of the package. The pbwf entry indicates that the source tar file is named by *package-name-version*. The pbpackager entry will be stored in the packages and should be you or your team. The projver/projtag entries indicate the version of the software and the version of the packaging scripts. The testver entry when true indicates that the package is in a test version, so no log file is computed (can be long), and version is made up using a timstamp. The delivery entry gives the subdirectory under which the packages will be delivered on the repository, and the defpkgdir entry corresponds to the local subdirectory hosting the package content.

For example:

```
pburl Lintel = file:///home/anderse/projects/Lintel-0.2012.02.28.tar.gz
pbrepo Lintel = http://tesla.hpl.hp.com/opensource
pbwf Lintel = 1
pbpackager Lintel = Eric Anderson <eric.anderson4@hp.com>
projver Lintel = 0.2012.02.28
projtag Lintel = 1
testver Lintel = false
delivery Lintel = production
defpkgdir Lintel = Lintel-0.2012.02.28
```

Create the build .tar.gz files:

Then you need to take those files and create the initial tar files. Run a command like:

```
% pb -p $project -r $version cms2build
```

To create the `$pbdefdir`/$project/pbdelivery/$project-$version.{,pbconf}.tar.gz files, the `$version`-$projtag.pb and pbrc files in the same directory.

build2pkg

The build2pkg command takes the tar files created in the cms2build step and attempts to build binary packages for your current operating system. There are two steps:

Update your filters and build files.

You probably need to edit the files describing the build steps in one of the `$pbdefdir`/$project/pbconf/$version/$project/{deb,rpm,pkg} directories and the filters in `$pbdefdir`/$project/pbconf/$version/pbfilter. Note that you can define additional filters and transforms in the *.pbf files. The build files will be filtered by the filters defined in the *.pbf files to generate the inputs to the build step. Therefore, if you change those files, you need to re-run the cms2build step.

Build the package.

Then you can run a command like:

```
% pb -p $project -r $version build2pkg
```

To create the files in `$project`/pbbuild that comprise your binary package(s).

newve

The newve command creates a new virtual environment, i.e. a chrooted OS for building packages. Using a virtual environment is an efficient way to build packages on a related set of operating systems. The OS's have to be related because the kernel will be shared. Steps:

Update ~/.pbrc

Update your ~/.pbrc file to specify the vepath, velist, velogin, and vetype variables, e.g.:

```
vepath default = $ENV{HOME}/cache-project-builder/chroot
velist default = debian-6.0-i386
velogin default = pb
vetype default = chroot
```

If you are building for rpm style OS's, update the verpmtype option, and install the appropriate tool.

```
verpmtype default = rpmbootstrap
```

You may also choose to specify a mirror for the OS packages, and optionally http/ftp proxies. You can specify the proxies either through environment variables ($http_proxy/$ftp_proxy) or in the configuration file. The configuration file will be used if no corresponding environment variable has been set. For example, for debian and with a local squid proxy:

```
rbsmirrorsrv debian = http://mirrors1.kernel.org/debian/
http_proxy default = http://localhost:3128/
ftp_proxy default = http://localhost:3128/
```

Run the cms2build command

If you have deleted your `$package`/pbdelivery directory, re-run the cms2build command as in the earlier step. This step is necessary to generate the *package*/pbdelivery/pbrc file.

Create the new virtual environment

Initialize the new operating system. This step will install the core OS packages for the virtual environment, e.g.:

```
% pb -v -p $project -m debian-6.0-i386 newve
```

setupve

The setupve command prepares a virtual environment for use by project builder. In particular it installs project-builder from the packages into the virtual environment. Two sub-steps are necessary:

Update `$project`.pb

You need to have a sshhost entry for setupve to work, so add one, even an invalid one, e.g.:

```
sshhost $project = foo.example.org
```

Setup the virtual environment

```
% pb -v -p $project -m debian-6.0-i386 setupve
```

If you prefer to install the current SVN version of project builder, you can substitute the setupve option by the sbx2setupv one.

build2ve

The build2ve command is similar to the build2pkg command in that it will take the sources created by cms2build and turn them into binary packages. The command has two differences. First, it creates the packages in a virtual environment, i.e. the one made by an earlier setupve setup. Second it copies the resulting packages to a repository and builds the repository meta-data needed.

Three sub-steps are needed:

Update `$project`.pb

You need to have a valid sshdir and sshhost entry for build2ve to work, so add them. Note that you need to be able to ssh from the host you run the command on to the repository host, preferably without needing to type in a password, so using ssh-agent or having a special passwordless project-builder ssh key will make this step easier.

```
sshhost $project = localhost
sshdir $project = $home/cache-project-builder/repos
```

You may also need to specify additional repository files to use or rpms to install. Note the URL for repositories is not the URL of the repository, but the URL of a file that can be put in the yum.repos.d or apt.sources.d directory.

```
addrepo centos-5-i386 = http://localhost/pb/centos-extras.repo,http://mirror.cer
```

Update your filters and build files

You may need to update your filter files (*.pbf) as in the build2pkg step if you are building for a new OS or architecture.

Build the packages and copy them to the repository

```
% pb -v -p $project -m debian-6.0-i386 build2ve
```

Debugging: If the build fails (and you did not specify the --no-stop-on-error) option, then the virtual environment and scripts should still be present and configured to build the package. You can run a command like 'sudo setarch i386 chroot `$path` bash' in order to get into the environment. In your log you should see a command like that. From there you can go into the /home/pb directory as the pb user and run the same style of pb commands as you did when doing build2pkg. This will help you figure out what has gone wrong in the build in the virtual environment.

AUTHORS

The Project−Builder.org team <http://trac.project−builder.org/> lead by Bruno Cornec <mailto:bruno@project−builder.org>.

COPYRIGHT

POD ERRORS

Hey! **The above document had some coding errors, which are explained below:**

Around line 475:

’=item’ outside of any ’=over’