NAME

poudriere - bulk package builder and port tester

SYNOPSIS

poudriere *poudriere-options* **command** [*command-options*]

DESCRIPTION

The **poudriere** tool is used to build packages from the ports tree. It can also be used to test a single port.

GLOBAL OPTIONS

poudriere accepts the following global options.

-e *etcdir* Path to the directory where **poudriere** will find its configuration data. See *FILES* and *ENVIRONMENT* for more information.

-A Force color support, even if not in a TTY.

-N Disable color support.

-v This will show more information during the build. Specify twice to enable debug output.

COMMANDS

The first argument to **poudriere** must be a **command** from the following list:

bulk (poudriere-bulk(8))
This command makes a ready-to-export package tree, and fills it with binary packages built from a given list of ports. During the build, hit ^T to send SIGINFO and show stats and progress about the build.

image (poudriere-image(8))
This command creates images.

jail (poudriere-jail(8))
This command manages the **poudriere** jails which are used as different building environments.

ports (poudriere-ports(8))
This command provides management of different portstrees which will be used by **poudriere**.

testport (poudriere-testport(8))
This command, mainly targeted at ports developers, launches a test on a given port (useful before

submitting/committing a port).

options (poudriere-options(8))
This command configures the options for a given port

logclean (poudriere-logclean(8))
This command will cleanup old logfiles

distclean (poudriere-distclean(8))
This command will cleanup old distfiles

pkgclean (poudriere-pkgclean(8))
This command will cleanup old and unwanted packages

queue (poudriere-queue(8))
This command allows a non-root user to queue **poudriere** commands. It is currently **EXPERIMENTAL**. Using it requires starting **poudriered** via the provided rc script.

status (poudriere-status(8))
This command shows status of current and previous builds

version (poudriere-version(8))
Show version of **poudriere**.

ENVIRONMENT

The **poudriere** command may use the following environment variables:

POUDRIERE_ETC If specified, the path to poudriere's config directory. Defaults to */usr/local/etc*.

UMASK The umask for files created by **poudriere**. Defaults to **022.**

The **jail** and **ports** subcommands may additionally use the following environment variables:

FETCH_BIND_ADDRESS The bind address used by fetch(1). See fetch(3) for more details.

HTTP_PROXY HTTP_* http_* FTP_PROXY FTP_* http_* SSL_* NO_PROXY no_proxy
The proxy configuration for fetch(1). See fetch(3) for other supported proxy environment variables.

The **jail** subcommand may additionally use the following environment variable:

MAKEOBJDIRPREFIX

FILES

POUDRIERE_ETC/poudriere.conf	See self-documented */usr/local/etc/poudriere.conf.sample* for example.
POUDRIERE_ETC/poudriere.d/poudriere.conf	The configuration can be stored in the poudriere.d directory as well.
POUDRIERE_ETC/poudriere.d	This directory contains various configuration files for the different jails.

EXIT STATUS

The **poudriere** utility exits 0 on success, and >0 if an error occurs.

EXAMPLES

Bulk build of specific binary packages

This first example provides a guide on how to use **poudriere** for bulk build packages.

[Prepare infrastructure]

First you have to create a jail, which will hold all the building infrastructure needs.

```
poudriere jail -c -v 8.2-RELEASE -a amd64 -j 82amd64
```

A jail will take approximately 3GB of space.

Of course you can use another version of FreeBSD, regardless of what version you are running. amd64 users can choose i386 arch like in this example:

```
poudriere jail -c -v 8.1-RELEASE -a i386 -j 81i386
```

This command will fetch and install a minimal jail, small (~400MB) so you can create a lot of them. It will install the jail under the pool you have chosen, at poudriere/jailname.

You also need to have at least one ports tree to build packages from, so let us take the default configuration and create a ports tree.

```
poudriere ports -c
```

A ports tree will take approximately 4GB of space.

[Specify a list of ports you want to build]

Create a flat text file in which you put the ports you want to see built by **poudriere**:

```
echo 'sysutils/screen' > ~/pkglist
echo 'editors/vim' >> ~/pkglist
```

Any line starting with the hash sign will be treated as a comment.

[Launch the bulk build]

Now you can launch the bulk build. At minimum the jail and list of packages to build must be specified.

```
poudriere bulk -f ~/pkglist -j 81i386
```

[Find your packages]

Once the bulk build is over, you can meet your shiny new packages here:

/usr/local/poudriere/data/packages/81i386

with 81i386 as the name of the jail.

Test a single port

This second example show how to use **poudriere** for a single port. Take the example of building a single port;

```
poudriere testport -o category/port -j myjail
```

all the tests will be done in myjail.

It starts the jail, then mount the ports tree (nullfs), then mounts the package dir (poudriere/data/packages/<jailname>-<tree>-<setname>), then it mounts the ~/ports-cvs/mybeautifulporttotest (nullfs) it builds all the dependencies (except runtime ones) and log it to poudriere/data/logs/testport/jailname/default/mybeautifulporttotest.log).

If packages for the dependencies already exist, then **poudriere** will use them.

When all the dependencies are built, packages for them are created so that next time it will be faster.

All the dependency phase is done with PREFIX == LOCALBASE.

After that it will build the port itself with LOCALBASE != PREFIX and log the build to poudriere/data/logs/testport/jailname/default/mybeautifulporttotest.log

Poudriere will try to: install it, create a package from it, deinstall it, check for cruft left behind and propose the line to add to pkg-plist if needed.

Poudriere is very easy to extend so that additional tests can be easily added.

FLAVORS

bulk -a will build all FLAVORS for each port. Otherwise **bulk** and **testport** use the following rules:

- A FLAVOR of **bar** for port *devel/foo* is specified as **devel/foo@bar**.

If *FLAVOR_DEFAULT_ALL* is not set, or is set to **no** (the default), in *poudriere.conf*, then:

- All FLAVORS for a port, without a specified FLAVOR, will be built using the FLAVOR **all**: **devel/foo@all**.
- The first (default) FLAVOR for a port is built by not specifying a FLAVOR: **devel/foo**.

If *FLAVOR_DEFAULT_ALL* is set to **yes** in *poudriere.conf*, then:

- All FLAVORS for a port, without a specified FLAVOR, will be built: **devel/foo**.
- The first (default) FLAVOR for a port is built by specifying the FLAVOR **-**: **devel/foo@-**

Known issues

- An invalid FLAVOR for a port will cause an error during dependency calculation.

- MOVED entries do not support a source FLAVOR, only a target one.

CUSTOMIZATION

For bulk building, you can customize binary packages produced by **poudriere** by changing build options port by port, and you can also specify building directives in a make.conf file.

Custom build options

Before building a package, **poudriere** can mount a directory containing option files if available. **poudriere** will check for any of these directories in this order:

/usr/local/etc/poudriere.d/<jailname>-<tree>-<setname>-options
/usr/local/etc/poudriere.d/<jailname>-<setname>-options

/usr/local/etc/poudriere.d/<jailname>-<tree>-options
/usr/local/etc/poudriere.d/<tree>-<setname>-options
/usr/local/etc/poudriere.d/<setname>-options
/usr/local/etc/poudriere.d/<tree>-options
/usr/local/etc/poudriere.d/<jailname>-options
/usr/local/etc/poudriere.d/options

If a directory with this name exists, it is null-mounted into the */var/db/ports/* directory of the jail, thus allowing to build package with custom OPTIONS.

The **options** subcommand can be used to easily configure options in the correct directory.

This directory has the usual layout for options: it contains one directory per port (the name of the port) containing an 'options' file with lines similar to:

WITH_FOO=true
WITHOUT_BAR=true

As a starter, you may want to copy an existing */var/db/ports/* to */usr/local/etc/poudriere.d/options*.

Blacklist ports

You can also specify a blacklist which will disallow the lists port origins from building on the matched jail. Any of the following are allowed and will all be used in the order shown:

/usr/local/etc/poudriere.d/blacklist
/usr/local/etc/poudriere.d/<setname>-blacklist
/usr/local/etc/poudriere.d/<tree>-blacklist
/usr/local/etc/poudriere.d/<jailname>-blacklist
/usr/local/etc/poudriere.d/<tree>-<setname>-blacklist
/usr/local/etc/poudriere.d/<jailname>-<tree>-blacklist
/usr/local/etc/poudriere.d/<jailname>-<setname>-blacklist
/usr/local/etc/poudriere.d/<jailname>-<tree>-<setname>-blacklist

If QEMU is being used then a special qemu blacklist is also loaded.
/usr/local/etc/poudriere.d/qemu-blacklist

Create optional poudriere.conf

You can also specify an optional poudriere.conf that is pulled in depending on the build. Any of the following are allowed and will all be used in the order shown:

/usr/local/etc/poudriere.d/poudriere.conf
/usr/local/etc/poudriere.d/<setname>-poudriere.conf
/usr/local/etc/poudriere.d/<tree>-poudriere.conf
/usr/local/etc/poudriere.d/<jailname>-poudriere.conf
/usr/local/etc/poudriere.d/<tree>-<setname>-poudriere.conf
/usr/local/etc/poudriere.d/<jailname>-<tree>-poudriere.conf
/usr/local/etc/poudriere.d/<jailname>-<setname>-poudriere.conf
/usr/local/etc/poudriere.d/<jailname>-<tree>-<setname>-poudriere.conf

Create optional make.conf

You can also specify a global make.conf which will be used for all the jails. Any of the following are allowed and will all be used in the order shown:

/usr/local/etc/poudriere.d/make.conf
/usr/local/etc/poudriere.d/<setname>-make.conf
/usr/local/etc/poudriere.d/<tree>-make.conf
/usr/local/etc/poudriere.d/<jailname>-make.conf
/usr/local/etc/poudriere.d/<tree>-<setname>-make.conf
/usr/local/etc/poudriere.d/<jailname>-<tree>-make.conf
/usr/local/etc/poudriere.d/<jailname>-<setname>-make.conf
/usr/local/etc/poudriere.d/<jailname>-<tree>-<setname>-make.conf
/usr/local/etc/poudriere.d/hooks/plugins/<plugin>/make.conf

Create optional src.conf

You can also specify a global src.conf which will be used for building jails with the **jail -c** subcommand. Any of the following are allowed and will all be used in the order shown:

/usr/local/etc/poudriere.d/src.conf
/usr/local/etc/poudriere.d/<setname>-src.conf
/usr/local/etc/poudriere.d/<jailname>-src.conf

Create optional src-env.conf

You can also specify a global src-env.conf which will be used for building jails with the **jail -c** subcommand. Any of the following are allowed and will all be used in the order shown:

/usr/local/etc/poudriere.d/src-env.conf
/usr/local/etc/poudriere.d/<setname>-src-env.conf
/usr/local/etc/poudriere.d/<jailname>-src-env.conf

Hooks

Hook scripts may be loaded in any of the following paths:

/usr/local/etc/poudriere.d/hooks/<hook>.sh
/usr/local/etc/poudriere.d/hooks/plugins/<plugin>/<hook>.sh

For specific hook documentation see: **https://github.com/freebsd/poudriere/wiki/hooks**

SEE ALSO

jail(8), poudriere-bulk(8), poudriere-distclean(8), poudriere-image(8), poudriere-jail(8), poudriere-logclean(8), poudriere-options(8), poudriere-pkgclean(8), poudriere-ports(8), poudriere-queue(8), poudriere-status(8), poudriere-testport(8), poudriere-version(8)

BUGS

In case of bugs, feel free to file a report:

https://github.com/freebsd/poudriere/issues

AUTHORS

Baptiste Daroussin <bapt@FreeBSD.org>
Bryan Drewery <bdrewery@FreeBSD.org>