

Managing your own Poudrière Repository

Matthew Seaman

BSDCan 2020 Eynsham

Who am I?

The very minimal biography

- FreeBSD Admin since the last millennium
- Ports committer since 2012
- pkg(8) developer (lapsed)
- Former core secretary

Who are you?

and what are you doing in my living room?

- Name — Introduce yourselves in the chat channel
- Where you're from?
- How you want to use Poudrière

Ground Rules

Ask questions

- — hands-up any time
- — or write in the chat channel

Stop me

- if you don't understand
- if you can't hear me
- if you're having problems with the practical bits

What are we doing today?

and other musings of a philosophical nature

Three parts:

- Set up — building a poudrière system
- Use — build a basic set of ports with that system
- Talk — Q & A, feedback, discussion

Set Up

What you need to participate in the class

1. Requirements:

- git
ansible-2.7 (or more recent)
dnspython (Ports: `py36-dnspython` **or** `pip3 install dnspython`)
ssh
a text editor

2. Check out git repository:

```
git clone --recurse-submodules https://github.com/infracaninophile/p4pm
```

Set Up

You will need a machine to install poudriere on

Running some version of FreeBSD

- At least as recent as anything you want to build packages for
- But I'm going to recommend **12.1-RELEASE**

Freshly installed is perfect — just add a user account and enable `sshd`

- SSH access to your target system:
`ssh username@hostname`

Set Up

Let's edit some text

- Edit ansible inventory: `hosts/poudriere`
change to your target system hostname
- Edit group variables: `hosts/group_vars/all.yaml`
create to settings for your own user account

Set up

Not usually necessary, but possibly useful

- (Optional) Run the keyscan playbook:
`ansible-playbook playbooks/keyscan.yaml`
Updates `~/.ssh/known_hosts`
- This does keep a backup of your current `known_hosts`

Set Up

Tweak the system to be a good ansible consumer

- We need to do some basic configuration to make them fully capable ansible clients
 - Install `python` and `sudo`
 - Create personal user accounts
 - Set up `pam_ssh_agent_auth` for `sudo`

Set Up

Passwords are great, until you have to use them

- Run the basics playbook:

```
ansible-playbook playbooks/basics.yaml -k -K
```

- You should now be able to log in as your own user, and sudo to root all authenticated by ssh keys — without being prompted for a password:

```
ssh -A username@hostname  
sudo -i
```

- Hint: add your user to the wheel group

Set Up

This is what we came here for

- The main event: run the poudriere playbook:
`ansible-playbook playbooks/poudriere.yaml`
- This will take some time...
- Live demo

Set Up

What just happened?

- What the playbook does:
 - Installs some useful packages
 - Installs and configures poudriere
 - Generates an RSA key used for package signing and website certificate
 - Builds filesystems for poudriere
 - Sets up cache
 - Sets up ports tree: Checks out <https://github.com/freebsd/freebsd-ports.git>
 - Sets up FreeBSD-12.1 jail installing from <http://ftp.freebsd.org>
 - Installs and configures nginx
 - Installs pkg(8) configuration to use the new poudriere repository

Set Up: poudriere

The main poudriere role in ansible

- Based on Vladimir Botka's

<https://github.com/vbotka/ansible-freebsd-poudriere>

- Fairly heavily modified

<https://github.com/infracaninophile/ansible-freebsd-poudriere>

Set Up: Useful Packages

Well, ‘useful package’ actually

- We need some trustworthy CA certificates:

`ca_root_nss`

- But this is a good place to install eg. development tools:

`tmux`

`emacs-nox`

`mtr`

`rsync`

`arcanist-php73`

- Customize the `standard_packages` array to your own requirements

`hosts/group_vars/poudriere.yaml`

Set Up: Poudriere itself

It's all (mostly) just shell scripts

- install packages
 poudriere
 ccache
- create RSA key pair and self-signed TLS certificate
- install `poudriere.conf`
- install `make.conf`
- create ZFSes used by poudriere (or UFS directory structure)
- configure ccache
- install ports tree
- install jails — FreeBSD-12.1-Release amd64

Set Up: Installing a ports tree

40,000 and counting

- The hardest part of the poudriere setup in terms of system requirements
- 1GiB RAM is typically too small for this step
- git is an arbitrary choice: any of the ways you could install a ports tree are supported
- ... or you can 'take over' a pre-existing ports tree
- ... or install several ports trees
- **Customize** `poudriere_ports` array in `hosts/group_vars/poudriere.yaml`

Set Up: Installing a jail

Just a clean copy of the system

- Installs FreeBSD from <http://ftp.freebsd.org/> — the same content as on the installation media
- Doesn't upgrade to the latest patch level. You can update if desired, but
 - Poudriere jails are not exposed to attack
 - Updating forces a rebuild of all packages
- Can install multiple jails eg. for *older* system versions
- Packages can be installed on the same *major* version + same or newer *minor* version (*)
- Can create jails via any mechanism, including self-build of `/usr/src`
- Build for `i386` on `amd64`
- Build for completely different architectures on `amd64` via `qemu`
<https://wiki.freebsd.org/Ports/BuildingPackagesThroughEmulation>

(*) Except for kernel modules

Set Up: ccache

Yes, autocorrect, that *is* how ccache is spelled

- D.R.Y. for compilers
- Generous 8GiB cache — trade disk space for time
- Poudriere builds as non-root user `nobody` — change ownership of ccache directories accordingly

Set Up: nginx

Watching the build progress

- Uses the same self-signed TLS certificate generated for poudriere
- Configuration based on <https://github.com/freebsd/poudriere/blob/master/src/share/examples/poudriere/nginx.conf.sample>
- Not immediately useable as a pkg repository: needs a proper certificate
- Mostly interested in the build logs right now

Set Up: pkg repo conf

So we can install the packages we just built

- Apply this on all of the machines you want to use your new repository
- Remember the comment about needing a real server certificate earlier?
- Recent OpenSSL means pkg(8) requires a recognised CA signature on the site certificate
- Only applies to the web-based downloads, not package signing
- Modify `nginx_ssl_certificate` by updating `hosts/group_vars/poudriere.yaml` to load a different certificate into `nginx`
- Getting your new cert onto your poudriere server is left as an exercise
- For the purposes of this class, fudge the issue by using `file:///` URLs on the poudriere server itself

Use

Putting it all to work

- Let's build some packages
- Not too many
- Change some default versions
- Global options settings
- Make the poudriere machine self-hosting

```
poudriere bulk -j 12_1a -f /usr/local/etc/poudriere.d/  
pkglist_amd64/packages
```

Use

Navigating the interface

- What does the poudriere web interface tell us?
 - What's building, already built and what's next to build
 - System load and throughput
 - Compilation success/failure
 - Diagnose most failures from the log file
 - eg. Easy fix for plist problems

Use

What poudriere does

- Builds all of the dependencies and build tools needed
- Only *rebuilds* dependencies when:
 - They are out of date
 - Options have changed
 - Jail updated
- Keeps the built packages even if you abort and restart a bulk build

Use

Don't do too much work

- We listed 10 packages to be built
- Which turned into 144 with build and runtime dependencies
- That's too many (at least, for the purposes of this class)
- `git` is largely to blame
- Changing options can almost halve the list

Use

Everything is optional

Setting options

- **Globally:** `poudriere options -c some/port`
- **Per ports tree:**
`poudriere options -p default -c some/port`
- **Per ports tree and package set:**
`poudriere options -p default -z development -c some/port`
- **Per ports tree, jail and package set:**
`poudriere options -p default -j 12_1a -z development -c some/port`

Use

- Options are stored in a directory tree, possibly labelled by jail, package set and ports tree:
 /usr/local/etc/poudriere.d/...
 default-development-options/
 default-options/
 options/
- Only the *first matching* directory tree is used

Use

- `make.conf` settings — hierarchy of files, also labelled by jail, package set and ports tree:
 - `/usr/local/etc/poudriere.d/...`
 - `default-development-make.conf`
 - `default-make.conf`
 - `make.conf`
- The result is the combination of all of these files

Use

Routine package building

- Typical command line:
`poudriere ports -u`
`poudriere bulk -j jailname -f packagelist`
- Only rebuilds what needs rebuilding
- Package repo is still usable during build
- New packages only published at the end of the `poudriere bulk`

Use

What to build?

- Specify a dictionary of packages in `hosts/group_vars/poudriere.yaml`
- Populates `/usr/local/etc/poudriere.d/pkglist_amd64/packages`

- Or install more than one list...
- Just list the packages you specifically want installed, not dependencies
- `pkg query -e '%a == 0' %O`
- Add more ports as required. Prune occasionally
- Can tag with `@flavor`

```
pkg_dict_amd64:
- pkglist: packages
  packages:
    - security/ca_root_nss
    - devel/ccache
    - devel/git
    - www/nginx
    - security/pam_ssh_agent_auth
    - ports-mgmt/pkg
    - ports-mgmt/poudriere
    - security/py-openssl
    - lang/python
    - security/sudo
```

Use

Again, from the top

- Should you rebuild everything from scratch at regular intervals?
 - Not actually necessary. Incremental builds work fine.
 - ... but your repo will contain some ‘orphaned’ packages
- How about if I change default values like `python37` \rightarrow `python38`?
 - Again, not a problem for incremental rebuilding
 - Although changing things like default `python` or `perl` versions mean so much of your repo will be rebuilt, you might just as well rebuild everything

Use

Resource Requirements

- System resource requirements
- Less than you might think
- Core2Duo with 8GB RAM and 250GB SSDs can update a repo of around 1000 packages within a hour or so each week
- Most modern desktop or laptop machines will be able to run a poudriere repo without problems

Use

Practical Considerations

- Some ports just take ages to build
`libreoffice`
- Worse: some are very early in the dependency tree
`llvmNN`
`gccN`
`openjdk`
- Just be patient

Use

Security Considerations

- If you update your build jails, poudriere will want to rebuild every package
- Port build jails are not an exposed security surface
- So don't be too religious about updating
- *Unless* you're building statically linked software and the vulnerabilities are in system libraries
- Keep your build box well updated and secured though
- Package signing to avoid impostors

Use

Sometimes things are not going to go smoothly

- What the build log tells you:
 - Port and build metadata
 - Dependencies
 - Options / make.conf settings
 - Build output
 - Staging / Packaging
 - PLIST testing

Use

How to get a clean build

- If it's broken upstream with the default settings, send patches. Or wait for someone else to fix it
- Otherwise, if it's broken with the particular combination of options you're using:
 - Fiddling with options settings will fix most problems
 - Sometimes you may come to regret tweaking default versions of ports
- These combinations are unlikely to be tested upstream, so probably won't be discovered or fixed promptly. Send bug reports (and patches)
- Especially if it builds and packages OK, but it's broken at runtime

Use

When it all goes a bit pear-shaped

- More complicated debugging
- Poudriere config specifically keeps WRKDIR from failed builds:
`SAVE_WRKDIR=yes`
- Good for:
 - fixing patches
 - autoconf problems
 - etc...

Use

Serious debugging

- But wait! There's more...

- Interactive build fixes

```
poudriere bulk -trk -C -j 12_0a -z development \  
-p default -i
```

- Rarely required

Talk

Feedback time

- Any questions?
- Anything you'd have liked me to cover?
-

Talk: why “poudrière”?

Previous software:

“Tinderbox”

Poudrière in French

but the word also translates

to:

Gunpowder Magazine



(Note the thin roof and thick walls so that explosions blow upwards rather than outwards)

**Thank You for
Attending
The End**

