

poudriere for Ports Maintenance

Matthew Seaman

EuroBSDCon 2019 Lillehammer

Who am I?

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

Who are you?

- Name
- ~~Rank~~ What do you do?
- ~~Serial Number~~ What do you want to learn?

Ground Rules

- Ask questions — hands-up any time
- 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?

- Three parts:
 - Set up — building a *poudriere* system
 - Use — build & debug ports with that system
 - Talk — further uses for *poudriere*

Set Up

1. Requirements:

- git
- ansible
- dnspython (Ports: py36-dnspython)
- ssh

2. Check out git repository:

```
git clone https://github.com/infracaninophile/p4pm
```

Set Up

- Take a slip with the hostname and access key
passphrase
- Gain access to your VM:
`ssh -i classN_ed25519 ec2-user@classN.black-earth.co.uk`

Set Up

- Edit ansible inventory: `hosts/poudriere`
change to your assigned host
- Edit group variables: `hosts/group_vars/all.yaml`
create your own user account

Set up

- (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

- VMs are `t2.small` instances installed using Colin Perceval's ZFS AMIs
<https://lists.freebsd.org/pipermail/freebsd-cloud/2019-February/000200.html>
- Essentially the same result as you'd get from FreeBSD installation media
- Differences:
 - Added First Boot actions to grow filesystem and apply system patches
 - `ec2-user` account

Set Up

- 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

- Run the basics playbook:

```
ansible-playbook playbooks/basics.yaml \
  -user ec2-user -private-key=keys/classN_ed25519
```
- You should be able to log in as your own user, and
sudo to root without being prompted for a password:

```
ssh -A username@classN.black-earth.co.uk
sudo -i
```

Set Up

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

Set Up

- What the playbook does:
 - Checks out <https://github.com/freebsd/freebsd-ports.git>
 - Installs some useful packages
 - Installs and configures poudriere
 - Installs and configures nginx
 - Installs a small script to run test builds

Set Up: Installing ports

- The hardest thing we're doing today in terms of system requirements
- `t2.micro` instance (1GB RAM) is too small
- `git` is an arbitrary choice: any of the ways you could install a ports tree are equally valid

Set Up: Useful Packages

- Development tools:
tmux
emacs-nox
ca_root_nss
mtr
rsync
arcanist-php73
- Customize this to your own requirements
hosts/group_vars/poudriere.yaml

Set Up: poudriere

- Based on Vladimir Botka's

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

- Fairly heavily modified

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

Set Up: poudriere

- install packages
 - poudriere
 - ccache
- create self-signed TLS certificate
- install poudriere.conf
- install make.conf
- create ZFSes used by poudriere
- configure ccache
- register ports tree created earlier
- install jails — FreeBSD 11, 12 Release; i386 and amd64

Set Up: nginx

- Uses the same self-signed TLS certificate generated by poudriere
- Configuration based on <https://github.com/freebsd/poudriere/blob/master/src/share/examples/poudriere/nginx.conf.sample>
- Useable as a pkg repository, but could be improved for that purpose
- Mostly interested in the build logs

Set Up: test-build.sh

- Builds the listed ports in each of the jails
- Builds all flavours
- Enables 'testing' (bulk -t option)

Use

- Let's build something
- Not too big
- Not too many dependencies

`textproc/jq`

Use

- What does the poudriere web interface tell us?
 - Dependencies
 - Compilation success/failure
 - Diagnose most failures from the log file
 - eg. Easy fix for plist problems

Use

- Builds all of the dependencies and build tools needed
- Only *rebuilds* dependencies when:
 - They are out of date
 - Options have changed
 - Jail updated
 - They're another specific build target

Use

- Setting options
- Globally: `poudriere options -c some/port`
- Per port:
`poudriere options -p development -c some/port`
- Per port and package set:
`poudriere options -p development -z development -c some/port`

Use

- Options are stored in a directory tree, possibly labelled by package set and ports tree:

```
/usr/local/etc/poudriere.d/...  
    development-development-options/  
    development-options/  
    options/
```

- Only the *first matching* directory tree is used

Use

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

Use

- Typical development cycle:
 - edit port
 - test build
 - fix problems
 - test build
 - repeat until clean result
 - (...other tests...)
 - commit

Use

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

Use

- But wait! There's more...

- Interactive build fixes

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

- Rarely required

Use

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

Use

- What the build log *doesn't* tell you
 - Does the ported software run correctly?
- But it will once port regression testing becomes standard
 - Too hit-and-miss to enable currently
 - Handling more complex CI requirements is hard

Use

- All updates to the ports should be run through poudriere
- Committers will do this by default
- ... but noting in a PR that changes pass poudriere testing always helps

Use

- What about other architectures?
- Assume everyone has access to amd64/i386
- Poudriere can cross build for various ARM and MIPS boards, but this is not a testing requirement
- You'll be notified by the package builders or by people that specifically test on alternate architectures if problems are found

Use

- What about Operating System Versions?
- Test on earliest supported version from each major branch
- Currently (2019-09-19) 11.1 and 12.0
- ABI compatibility guarantee means software that works on an early version of a branch will continue to work on all later ones
 - *Except* for loadable kernel modules
- Converse not necessarily true: newer packages may not work on older branches

Use

- Your build box needs to be newer than (or at least as new as) the latest branch you want to build packages for
- HEAD usually conforms, but it's a dev branch and there may be the odd bump in the road
- Running older poudriere jails on HEAD will work fine

Use

- Practical considerations
 - Some ports take ages to build
 - `libreoffice`
 - Worse: some are very early in the dependency tree
 - `llvm`
 - `gcc`
 - `openjdk`
- Just be patient

Use

- 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

Use

- We've talked about poudriere as a tool for ports maintenance
- Poudriere as a tool for generating your own repo is very similar
 - Build a whole list of packages
 - Customize port options / make.conf
 - Only build the flavours you need
 - Tweak nginx.conf to add alias matching the `ABI` setting `pkg(8)` generates
 - Custom `repo.conf` and repository keys

Use

- 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

Talk

- *Any questions?*

Talk: why “poudriere”?

Previous software: “Tinderbox”

Poudrière in French

but the word also translates to:

Gunpowder Magazine

