Puffalanche - OpenBSD by the busloads

OpenBSD and Vagrant: make autostart(8) by the busloads easy

Philipp Bühler <pb@sysfive.com> @pb_double

sysfive.com portfolio

- Continuous system and application operation
- Collaborations with Providers, Developers, Services and QA
- Hybrid cloud provisioning
- Cost efficient scaling on commodity HW
- Scale out to AWS/RS/GCE
- Incident, problem, disaster response

- Service availability independent of solution scenario
- Migrate from or to private/public cloud or own HW
- Robust, scalable technology portfolio
- Continuous improvements in security and server architecture
- Coherent provisioning across platforms (dev/stage/live)
- Vendor/provider independence, OSS focus
- ... and we’re hiring.
Solving what?

- Run multiple OpenBSD VMs on OpenBSD - w/o dealing with vm.conf(5)
- make inter/intra-networking "just work" - no bridge(8) "hassle"
- develop and TEST autoinstall at 30,000ft (or -50)
- create reproducible installs - even "me so unique" ones
- and also retrospective (live --> test)
- use the same configuration (Vagrantfile) on OpenBSD/OSX/Linux to get the same resulting VM package/network/setup
Puffy boxed on OpenBSD (Dev#1)

- Bootstrap-VM: might be based on manual install
  - better do it in ‘packer’
- vPuffy1+2: autoinstall from Bootstrap-VM directly
- vPuffy2a: autoinstall via dhcrelay on vPuffy2
Puffy boxed on Linux/OSX/.. (Dev#2-n)

Just run the SAME "infrastructure" on

- OSX (Virtualbox/VMware)
- Linux (Virtualbox/libvirt)
- Cloud (AWS/GCE/..)
- basically everything that Vagrant supports

Infrastructure going on a trainride or being airborne.
Puffy BREAKOUT to physical.

- Not impressed so far? Let’s go physical..
- Run the very SAME "infrastructure" on REAL puffymachines
- Develop + Test virtualized, use results for:
  - Confidence in rollouts
  - debug problems on Laptop, roll-out solution to Realpuffy
What did I need to work on? (ongoing)

- OpenBSD: VMM PXE enabled BIOS (hi Mike)
- Vagrant "Core" (plugin-loader) (+port)
- OpenBSD’s VMM as a Vagrant provider-plugin (+gem port)
- [Ruby development tools - only for plugin development (BUILD.md in repo)]
- integrated vether(8)/bridge(8)/dhcpd(8) setup (VM to VM)
- deeper knowhow on autoinstall(8) features
- [installer enhancements (pre-install.sh / softraid(4))]

Groundwork is done, but open points:
- Better automation in network lookup (no magic numbers/assumptions).
- VM-to-VM isolated networking (not via Host, bad for PXE)
- Gem of vagrant-openbsd (gemspec done)
- ports(7) of Vagrant and vagrant-openbsd. First one "complicated" for me, second should be a breeze after having a Gem on rubygems.org
- "synced folders" (NFS)
• suspend/resume/"package"
What’s already around?

- non-published PXE BIOS
- bundle(1) Vagrant 2.1 (but likely works with 1.5+)
- Vagrant provider-plugin: 0.3.0
  - box support (disk or [PXE-BIOS])
  - host OS detection by vagrant
  - VM lifecycle "import/up/halt/destroy"
  - Host-to-Guest networking + SSH
- autoinstall concepts:
  - PXE/DHCP/TFTP response steering
  - install.conf steering
  - disklabel templates
  - multiple set sources
  - siteNN.tgz
  - siteNN-hostname.tgz
  - install.site
Vagrant - Architecture

Naming - what’s in the bento?

Core
   plugin loader "framework" + utils
host
   capabilities (Linux, OSX, Free/OpenBSD, ..)
box
   Disk/BIOS image + metadata packed as tar.gz
guest
   capabilities (Linux, Free/OpenBSD, ..)
provider
   capabilities (vbox/VMM/bhyve/...) where the main show goes
communicators
   ssh/winssh/winrm to let Vagrant configure the guest
provisioner
   shell/ansible/chef/puppet/... run after the first ’up’ of the VM
Plugins

All plugins come along this layout:

e.g. Provider
  • lazy loader - overloading classes
  • Action (abstraction classes, workflow)
  • Driver (host integration, here mainly vmctl)
  • Templates (ERB) (vm.conf)
Networking capabilities

- **port-forward**: open arbitrary ports (default 127.0.0.1) on the host and ssh-forward it into the VM
- **bridged network**: reach out from VM to The Internet (Host might need to NAT)
- "private" network: VM to VM communication on isolated network (bridge(8) rdomain(4)?)
Provisioner - post-postinstall

Almost any automation stack can be included into a Vagrant based VM

- (inline) shell scripts
- ansible
- Chef
- Puppet
- Salt
- you-name-it, likely there’s a plugin
pf(4) integration

Still undecided, leave it to the user to adapt some pf.conf(5) or depend on an ‘anchor’ in it like relayd(8), authpf(8).
Leaning to anchor, which will make the experience likely better but requires more work in the plugin.
Minimum pf.conf(5) needed for bridged networking (VM to The Internet) on Host:

```
pass out on $ext_if from 100.64.0.0/10 to any nat-to ($ext_if)

pass in proto { tcp udp } from from 100.64.0.0/10 to any port 53 rdr-to 127.0.0.1
# and run unbound(1) or thelike. fix vmd-dhcpd?
# ..dont forget net.inet.ip.forwarding=1
```
Anatomy of an UP session

$ uname -a; bundle exec vagrant status; bundle exec vagrant up; \
bundle exec vagrant ssh -c "uname -a"

OpenBSD ssfnhv011.ham3.rootnexus.net 6.2 GENERIC.MP#134 amd64

Current machine states:
vagrobsd not_created (openbsd)

Bringing machine ‘vagrobsd’ up with ‘openbsd’ provider...

===> vagrobsd: Verifying VMM present and CPU capable...
===> vagrobsd: Importing an OpenBSD instance
  vagrobsd: Cloning virtual hard drive...
  vagrobsd: Successfully imported a VM image
  vagrobsd: Creating vmctl1 configuration
===> vagrobsd: Starting the machine...
===> vagrobsd: Waiting for the machine to report its IP address...
  vagrobsd: IP: 100.64.2.3
===> vagrobsd: Waiting for machine to boot. This may take a few minutes...
  vagrobsd: SSH address: 100.64.2.3:22
  vagrobsd: SSH username: root
  vagrobsd: SSH auth method: password
  vagrobsd: Inserting generated public key within guest...
  vagrobsd: Removing insecure key from the guest if it’s present...
  vagrobsd: Key inserted! Disconnecting and reconnecting using new SSH key...
===> vagrobsd: Machine booted and ready!

OpenBSD openbsd62.example.com 6.2 GENERIC#132 amd64
Connection to 100.64.2.3 closed.

$ cat Vagrantfile
Vagrant.configure("2") do |config|
  config.vm.box = "vagrobsd"
  config.ssh.shell = "ksh -l"
  config.ssh.sudo_command = "doas -n %c"
  config.vm.define "vagrobsd" do |v|
    v.vm.hostname = "openbsd-vagrant"
  end
end

Connection to 100.64.2.3 closed.
autoinstall(8)

Overview / Concept

- two shell scripts, common and MD (~3500 lines)
- install or upgrade
- simple answerfile (in ramdisk or download)
- answers consumed as they match, might be given several times
- https + signify
Anatomy of an installation

- bsd.rd, init and to /etc/rc
- dot.profile basic setup and launch installer
- choosing autoinstall if netboot (after 5s)
- sets mode and installsets
- configure network
- fetch official mirror list
- fetch answerfile
- disk config
- fetch+install sets
- system configuration, user setup
- relink kernel
- install bootblocks
- custom post-install
- /etc/rc.firsttime after reboot (sys{patch,merge}, fw_update)
Disks (amd64)

- fetch a disklabel(8) template
- OR calculate a root disk layout
- no softraid support YET - quirk: install twice
Network

- DHCP (inet4) or SLAAC (inet6)
- static configuration ("nope")
- can make use of http[s]_proxy
- ftplist.cgi (handling of mirrors)
Debugging

- bails to shell if errors occur
- /tmp/ai/ai.log # what happened with which values
- /tmp/ai/ai.conf # answerfile as provided (tag it!)
- /tmp/i/${MODE}.resp # logged answers
- /tmp/i/httplist,httpsec,wlanlist
- /tmp/i/cgiinfo # mirror information
- from shell: install -af $answerfile
base system settings

Generally order doesn’t matter - unless one uses same question multiple times, like installing from more than one source.

System hostname = myhost
Choose your keyboard layout = us
Start sshd(8) by default = yes
Do you expect to run the X Window System = no
Do you want the X Window System to be started by xenodm = no
Change the default console to = com0
Terminal type = vt220
speed should com0 use = 115200
What timezone are you in = Europe/Berlin
Sets location and Disk

It’s possible to repeat the question/answer tuples with differing values. So it’s possible to install the base OpenBSD from official mirrors, and subsequently pull siteNN.tgz from a different/local server.

Location of sets = h # http(s)
Set name(s)? = -x* +site*

Can be used multiple times, but (A)utolayout only for the rootdisk

disk do you wish to initialize = sd0
Which disk is the root disk = sd0
Use (A)uto layout, (E)dit auto layout, or create (C)ustom layout = A
URL to autopartitioning template for disklabel = https://10.1.1.100/disklabeltemplate
User

Password for root account = usekeyonly
Allow root ssh login = prohibit-password
Setup a user = toor
Password for user toor = **************** # 13 asterisks
Full name for user toor = Mr Toor
Public ssh key for user toor = ssh-rsa 909239234239490721349...
Public ssh key for root account = ssh-rsa 23674573423948902384...==
installtime networking

Time appears wrong. Set to = yes # off > 120s from HTTP

network interface should be used for the initial DHCP request = ix0
#defaults to netboot device

HTTP proxy URL? = none
HTTP Server? = [http[s]://]10.1.1.100 # also goes to installurl(5)
Unable to connect using https. Use http instead = no
Server directory? = pub/OpenBSD/6.2/amd64
DNS wont be asked when DHCP is used. Really?

DNS domain name = example.com
DNS nameservers = 1.1.1.1

network interface do you wish to configure = (phy0|vlan0) # hostname.if(5)

Symbolic (host) name for $_if = virtahost # only if > 1

IPv4 address for (em0|ix0|..) = (dhcp|10.1.1.1|10.2.2.2/24)
Netmask for for (em0|ix0|..) = 255.255.255.0 # if no CIDR above
Default IPv4 route? = 10.1.1.254 # static configuration if no dhcp, mygate(5)

IPv6 address for (em0|ix0|..) = (autoconf|fd8e:c35e:4631:0::1/64)
IPv6 prefix length for (em0|ix0|..) = 64 # if no prefix above

# vlan
Which interface:tag should $_if be on = em0 # any physical if, $_if like vlan0

Wireless

Access point? = any # 80211 setup, ESSID
Security protocol? = (O|W|P) # 80211 setup, answer means: Open/WEP/WPA-PSK

WEP key? = 13_characters # 80211 setup, see ifconfig(8) /nwkey
WPA passphrase? = longpassphrase # 80211 setup, see ifconfig(8) /wpakey
Checksum handling

These will happen for customized/additional sets like siteNN.tgz

Checksum test for $_f failed. Continue anyway = no
Unverified sets ... Continue without verification = no

$_f will be siteNN.tgz. For now there’s no way to properly signify(1) this (?).
Site packages / scripts

Installer will offer those for selection if present (index.txt!) and matches the hostname. Contents will be just be unpacked like

```
tar zxpf siteNN.tgz -C /mnt
```

install.site can be any arbitrary shell script that will be run chrooted in /mnt. Do not forget to set +x on it before tarring it up.

- siteNN.tgz : every host might select this
- siteNN-hostname.tgz : would only be selectable when hostname matches
- install.site / upgrade.site : be ran last before reboot
Decision making

tftp filename ’name’
installer will choose install or upgrade depending on the returned filename:
auto_install or auto_upgrade.

tftp filename download
installer will tftp download ’auto_install’ which shall be a symlink to the desired
bsd.rd. Note that tftpd(8) can deliver different files based on IP address (-i, since
6.3).

tftp next-server
installer will tftp download from this server (optional)

XXX-install.conf
installer will download MAC_Addr-install.conf or hostname-install.conf or
install.conf (same for update)

install.conf: ’HTTP Server’
as previous, other server(s) can be used for sets downloads

Some more fine grained DHCP options listed in autoinstall(8) manpage.
Ohai + Links + Thanks

- Revisit for EuroBSDCon Bucharest the latest (Tutorial!?)
- Any help/pull request very welcome (e.g. NFS)
- Vagrant plugins - https://github.com/hashicorp/vagrant/wiki/Available-Vagrant-Plugins
- Kickoff - Glarus, Switzerland / https://ungleich.ch

Questions?
BEER after the closing session and auction

DO NOT MISS - and see you at the Red Lion after it