

OpenBSD: VMM / packer / vagrant

puffy clouds?

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



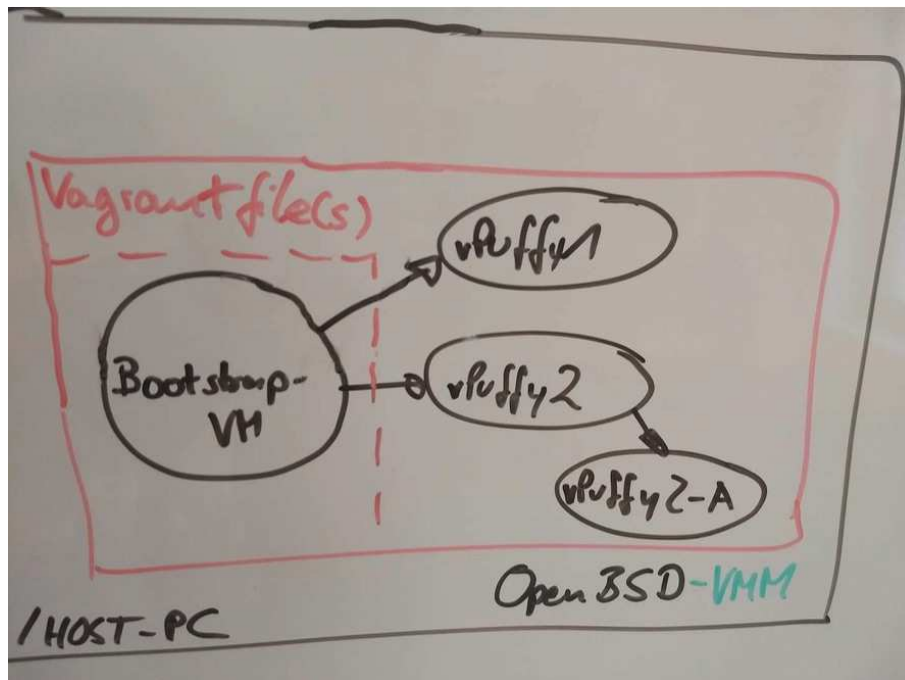
Ze problems I tackle

- 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) "infrastructure to go"
- 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

Solutions / Approaches

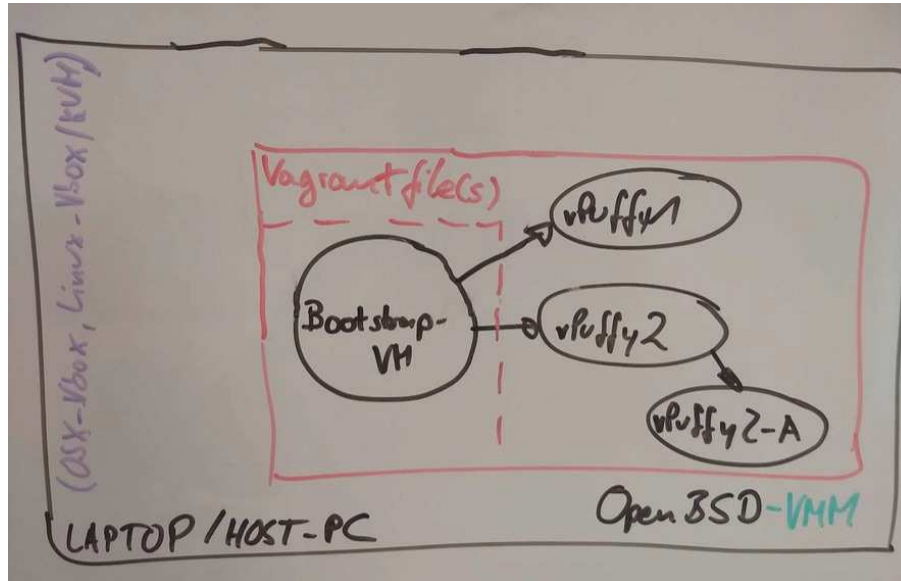
- my-imager.sh ; my-deploy.sh
- github | sh
- my-cold-hands ("VM guy is AWOL")
- vagrant / packer (o'rly)

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)

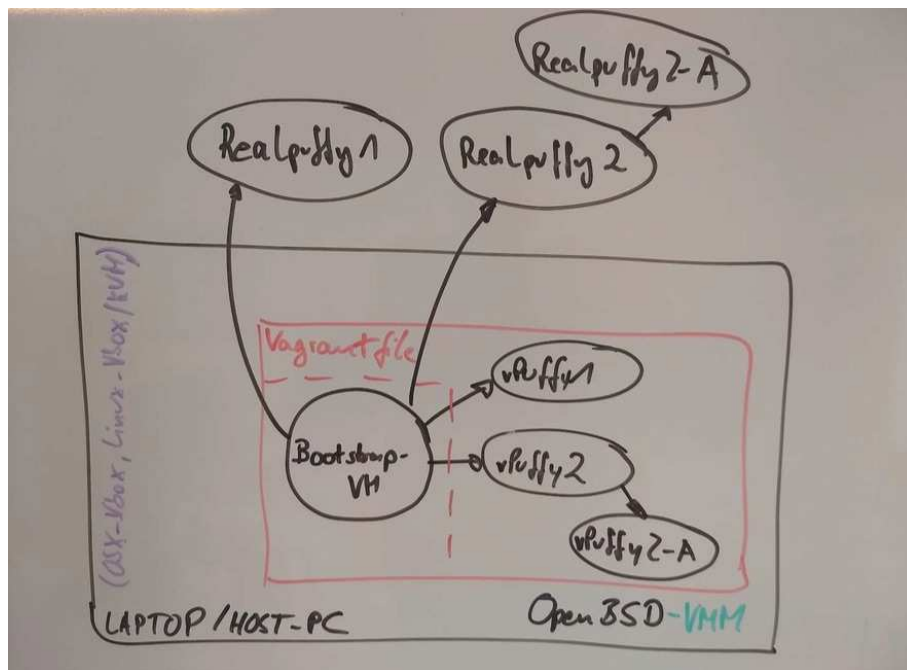


Just run the SAME
"infrastructure" on

- 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

OpenBSD VMM

vm(4)

virtual machine monitor (VMM) providing the required resources for the VMs (CPU, Disk, Net) and handles the necessary memory mapping (isolation).

vmd(8)

userland daemon to interact with vm(4) to create actual VMs and handle their lifecycle through:

vmctl(8)

administrative tool to create, start/stop, etc VMs. In this scope also the main 'interface' for the packer builder plugin.

vm.conf(5)

configuration file for vmd(8), persist VM/network configurations.

doas(1)

While most tasks the “builder” can (and should!) run as unprivileged user, some commands need to be run as root. The plugin does so automatically. **Caveat:** needs 'nopass' for now (no tty), 'persist' typically timeouts too early.

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

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

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 vmctl 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
```

What's packer anyway?

Packer is an open source tool for creating **identical** machine images for multiple platforms from a **single source** configuration. Packer is lightweight, runs on every major operating system, and is highly performant, creating machine images for multiple platforms **in parallel**.

Packer does not replace configuration management like Chef or Puppet. In fact, when building images, Packer is able to use tools like Chef or Puppet to install software onto the image.

- written in golang
- small core providing communications (rpc)
- everything else is a plugin (but linked into one binary)
- configuration in JSON (+ optional variables)

forms in the sandbox / Terminology

Artifacts

Outcome of a "Build", e.g. AMI, .vmdk, .box

Builds

The actual running task producing above artifacts

Builders

Code to steer the VM host, handle disk images, etc (see below)

Provisioners (optional)

Additional treatment, installation goes here and range from simple inline shell scripts to full-blown ansible, Chef, ..

Post-processors

Treat the Artifacts after creation, e.g. compress, upload AWS, ..

Templates

The JSON files defining all of the above (and some) - NOT a VM "template"

Builders + Post-Provisioning

By default the following “builder” engines are supported. Where needed the accompanying “post-processor” is typically available, too (e.g. EC2/AMI upload):

Alicloud ECS, Amazon EC2, Azure, CloudStack, DigitalOcean, Docker, File, Google Cloud, Hetzner Cloud, HyperOne, Hyper-V, Linode, LXC, LXD, NAVER Cloud, Null, 1&1, OpenStack, Oracle, Parallels, ProfitBricks, QEMU, Scaleway, Tencent Cloud, Triton, Vagrant, VirtualBox, VMware, Yandex.Cloud.

Further “builders” can be found in the wild and are just added as a single go binary in certain paths (e.g. ~/.packer.d/plugins/)

By default additional provisioning support for the following tools:

Ansible, Breakpoint, Chef, Converge, File, InSpec, PowerShell, Puppet, Salt Masterless, Shell.

OpenBSD dependencies / configuration

- /etc/pf.conf:

```
1 pass in quick proto { udp tcp } from 100.64.0.0/10 to any port domain \
2                               rdr-to $dns_server port domain
3 pass out quick on $ext_if from 100.64.0.0/10 to any nat-to $ext_if
```

- /etc/sysctl.conf

```
1 net.inet.ip.forwarding=1
```

- vmd(8)

```
1 rcctl enable vmd
2 rcctl start vmd
```

packer *DEMO TIME*

```
1 {
2   "builders": [
3     {
4       "type": "openbsd-vmm",
5       "name": "packer-obsd64-vmm-amd64",
6       "vm_name": "myvm",
7       "disk_size": "1500M",
8       "disk_format": "raw",
9       "output_directory": "images",
10      "http_directory": "./docroot",
11      "iso_image": "~/Downloads/install65.iso",
12      "bios": "/bsd.rd",
13      "boot_wait": "5s",
14      "boot_command": [
15        "A<enter>",
16        "http://{{ .HTTPIP }}:{{ .HTTPPort }}/packer-auto_install.conf<enter>",
17      ],
18      "ssh_username": "root"
19    }
20  ]
21 }
```

<https://asciinema.org/a/246654>

asciinema play ~/devel/presenter/presentations/h4g/2019/tl-demo.cast

What's already around?

vagrant

- fake netboot (vmctl -B)
- 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

packer

- plugin: all architectural integration (config/builder/driver)
- plugin_steps: create disk/VM/boot/provide auto_install infra

What's still to be done?

vagrant

- Networking VM-to-VM
- vagrant-binary (Go)
- ports(7)

packer

- GetTapIpAddress
- ports(7)

Ohai + Links + Thanks

- Code/Slides - <https://github.com/double-p>
- Kickoff - Glarus, Switzerland / <https://hack4glarus.ch>
- Thanks to Claudio for the idea of vmctl -B
- Thanks to grubernaut for go.mod, review, ISO+QCOW2 support and other spurious fixes
- Any help/pull request very welcome (e.g. multi-disk)

Questions?

