The FreeBSD Appliance

Leveraging FreeBSD and Strategic Scripting to Deliver Storage and Virtualization Services

Michael Dexter

editor@callfortesting.org

EuroBSDcon 2023

Please Read The Paper!

- It’s Short and Sweet
- Highlights many of these building blocks
- What I wish I was handed 20 years ago

My 2007 – 2023 paper trail leads to this point

callfortesting.org/log/TheFreeBSDAppliance-MichaelDexter.pdf

What is a Software Appliance?

“A software appliance is a **software application** combined with **just enough operating system** (JeOS) to run optimally on industry-standard hardware (typically a server) or in a virtual machine. It is a software distribution or firmware that implements a computer appliance.”

en.wikipedia.org/wiki/Software_appliance

What is a Software Appliance?

From General-Purpose Computing

to

Optimal Single-Purpose Computing

Any OS could be an appliance foundation

“Just Enough OS”

Some of the most significant software and hardware appliances have been built on BSD Unix

Many have stayed focused

Some have suffered from “feature creep”

“Just Enough OS”

Every appliance struggles to keep up with upstream

5 Year Hardware Warranty = 5 Year LTS Support?

Potential for competing motivations

Diagnosing an appliance will always be a question of operating system and application debugging

“Just Enough OS”

Forking is Expensive

Don’t add 64-bit support or ZFS on your own

Re-syncing is Expensive

The Secret: Engage Upstream!

“Just Enough” → “Had Enough”

Let’s explore
“Radically Just Enough”

“Just Enough” → “Had Enough”

Or “Twenty year of extreme patience in service of extreme impatience”

“Just Enough” → “Had Enough”

Or “Stop computing like it’s 2003”

Who remembers BSD Unix in 2003?

Continuously building world

FreeBSD Jail and NetBSD/Xen hinted at the future

Storage was terrible

That was then, this is now

Continuously building world…

Modern hardware obviously helps

Though Clang/LLVM came along…

The build options are working!

That was then, this is now

FreeBSD Jail and NetBSD/Xen
hinted at the future...

Hello bhyve and FreeBSD/Xen!

Thank you CPU hardware assistance!

That was then, this is now

Storage was terrible…

ZFS and 1TB HDDs changed everything

30TB NVMe drives are “affordable”

The OS is Half the Battle

FreeBSD, illumos, and GNU/Linux
can now all deliver ZFS,
containers, and hypervisors

How well is the question

Half the Battle: OpenZFS

illumos is not on OpenZFS

GNU/Linux cannot ship with OpenZFS

GNU/Linux suffers up to a 20% performance penalty at 100Gbps+

Half the Battle: Jails

illumos Zones are awesome

GNU/Linux does not have a true jail
and the performance depends
on who you ask

Half the Battle: Hypervisors

illumos imported FreeBSD bhyve…

GNU/Linux KVM is the most feature-rich free software hypervisor

If you can survive without `CTRL-T`…

Half the Battle: Honorable Mention

NetBSD technically has ZFS and a few hypervisors to choose from...

Xen, nvmm, HAXM

Fundamentally a Great Start

Out of the box, FreeBSD has significant advantages

Plus is a unified OS and that is permissively-licensed

Real and Perceived FreeBSD Complaints

- No nested virtualization
- No NFS Ganesha with SMB compatible locking
- No Docker – WE ARE TO BLAME
- Poor support for poor hardware
- Fewer GUI-based management tools

The subject of this talk

Optimal Single-Purpose Computing: JeOS

I have pursued “Just enough OS” since 2003 when the 5.1 jail tools arrived

“My web server jail does not need a toolchain or hypervisor”

JeOS: Packaged Base and Build Options

I confess I loved how
Red Hat Linux (5.2) was
inventoried with packages

Resulting in RPM Hell...
but the idea was right

JeOS: Packaged Base

FreeBSD Packaged Base will be here

Real Soon Now™

One of five implementations...

JeOS: Build Options

```
man src.conf ...

WITHOUT_BHYVE
        Do not build or install bhyve(8),
        associated utilities, and examples.

        This option only affects amd64/amd64.
```

Add `WITHOUT_BHYVE=YES` to `/etc/src.conf`

Anatomy of a Build Option

/usr/src/usr.sbin/Makefile.amd64

```
.if ${MK_BHYVE} != "no"
SUBDIR+=	bhyve
SUBDIR+=	bhyvectl
SUBDIR+=	bhyveload
.endif
```

Structured & Automated with OccamBSD

- *“An application of Occam's razor to FreeBSD”*
- An OS reduced to its minimum components
  - Minimum components to build
  - Minimum components to boot on a VM
  - Minimum components to boot on hardware
- Add networking and other features as needed

OccamBSD

- World and kernel build times in minutes
- Working OS in under 150 Megabytes
- Boot times in seconds
- Expected to be unrecognizable
- A flashback to 4.3BSD!
- *Very* educational…

OccamBSD: Immediate Benefits

- Reveal abandoned components
- Reveal undocumented components
- Reveal cross-building issues
- Produce a “Rescue” ISO
- Perfect classroom OS (Confirmed by Antranig V)
- Reproducible Builds are a perfect compliment

OccamBSD: START HERE

- No, Seriously, Literally...
- The core OS used by all uses at all times
- Where to begin documenting
- Where to begin auditing
- Where to begin fuzz testing
- Where to begin LEARNING

OccamBSD: 20 Years in the Making

`github.com/michaeldexter/occambsd`

```
sh occambsd.sh -v -z -p profile-amd64-zfs.txt
```

- `-v` VM image
- `-z` OpenZFS thanks to `mkimage -t zfs`!
- `-p` Profile

OccamBSD: 20 Years in the Making

`github.com/michaeldexter/occambsd`

Note the prior art in the README.md

NanoBSD, picobsd, TinyBSD, Crochet, Poudriere image.sh, mkjail

A *great* way to fall in love with FreeBSD again

Institutional Occamization: Boot Images

`/usr/src`

<Occamize... `buildworld` | `buildkernel`>

`installworld` | `VM-IMAGE`

Ode to the raw `(VM-)IMAGE`

Something something “cloud” “pets” “cattle”
“QCOW2” “VMDK” “VDI” “VHD(X)” “OCI”…

ZFS obsoletes most of these “solutions”

Virtualized Storage → Storage

Ode to the raw `(VM-)IMAGE`

Virtualized Storage → Storage

VM-IMAGE → BOOT IMAGE

FreeBSD Gets This Right

Ode to the raw `(VM-)IMAGE`

`makefs(8) -t zfs …`

Uses `libzfs`

Unprivileged Operation

Colin P says there might be an issue…

Ode to the raw (VM-)IMAGE

```
make -C /usr/src/release \
     SRCCONF=/etc/src.conf \
     KERNCONFDIR=/mydir \
     KERNCONF=MYKERNCONF \
     vm-image WITH_VMIMAGES=YES \
     VMFORMATS=raw \
     VMFS=raw 4G 1G
```

Ode to the raw `(VM-)IMAGE`

Imagine that!

Raw | Legacy/UEFI | OpenZFS

Hypervisor Boot | Hardware Boot

Ode to the raw `(VM-)IMAGE`

No really! Imagine that!

`github.com/michaeldexter/occambsd/imagine.sh`

Why install when you simply splat it down?

Ode to the raw `(VM-)IMAGE`

`imagine.sh`

Debian ships raw “nocloud” images…

`debagine.sh`!

Ode to the raw `(VM-)IMAGE`

Result: From Source to Installed

IN TWO STAGES

BUILDS → IMAGING → VM/HW BOOT

Did you say “Debian”?

Not another DOT COM Distro

“Zero Trust” build option via:

`bootstrapping.miraheze.org`

Note CHERI Build!

Did you say “Debian”?

Easily provisioned with `sysutils/debootstrap`

Supported with FreeBSD’s Linux emulation

Manageable with FUSE ext4, growpart, resize2fs

Raw images via FAI (Fully Automatic Installation)

Please build Devuan! Debian root on ZFS!

Net Result So Far: WE OWN THE STACK

Configurable “Just enough OS” (JeOS)

FreeBSD Native and Linux ABIs

Containers and VMs

Integrated OpenZFS doing HEAVY LIFTING

Net Result So Far: WE OWN THE STACK

Jail Innovations

`mount nullfs -f`

`The mount_nullfs utility supports mounting both directories` *`and single files`*

New `.include` functionality

A Platform But Not Yet an Appliance

“User Friendliness”

I’ve edited `/etc/ssh/sshd_config`
in a consistent way for 20+ years

My user “training” took a few minutes

ONCE

A Platform But Not Yet an Appliance

POLA: The Principle Of Least Astonishment

Multi-decade muscle memory

Something something old dogs, new tricks

I’m 250 years old by the new calculation!

A Platform… The Easy Buttons

“Ansible!”

“Webmin!”

Not wrong…

Infrastructure as a Service (IAAS)

“Ansible!”
“Puppet!”
“Chef!”
“Salt!”
“Terraform!”

```
pkg install terraform
New packages to be INSTALLED:
    terraform: 1.5.6_1
```

A Platform... The Easy Buttons

`pkg install py39-ansible`

py39-Babel: 2.12.1
py39-Jinja2: 3.1.2
py39-ansible: 8.2.0
py39-ansible-core: 2.15.2
py39-cffi: 1.15.1
py39-cryptography: 41.0.3_1,1
py39-markupsafe: 2.1.3
py39-packaging: 23.1
py39-pycparser: 2.21
py39-pytz: 2023.3,1
py39-resolvelib: 0.8.1_1
py39-setuptools: 63.1.0_1
py39-toml: 0.10.2
py39-yaml: 6.0
python39: 3.9.18

`pkg install puppet8`

augeas: 1.14.0_1
libunwind: 20211201_2
libxml2: 2.10.4
puppet8: 8.2.0
ruby: 3.1.4_1,1
ruby31-gems: 3.4.19
rubygem-concurrent-ruby: 1.2.2
rubygem-deep_merge: 1.2.2
rubygem-facter: 4.4.3
rubygem-fast_gettext: 2.3.0
rubygem-ffi: 1.15.5_1
rubygem-hocon: 1.4.0
rubygem-json_pure: 2.6.3
rubygem-locale: 2.1.3
rubygem-multi_json: 1.15.0
rubygem-puppet-resource_api: 1.9.0
rubygem-rexml: 3.2.6
rubygem-ruby-augeas: 0.5.0_4
rubygem-scanf: 1.0.0
rubygem-semantic_puppet: 1.1.0
rubygem-sys-filesystem: 1.4.3
rubygem-thor: 1.2.2

A Platform… The Easy Buttons

```
pkg install rubygem-chef
```

libunwind: 20211201_2
ruby: 3.1.4_1,1
ruby31-gems: 3.4.19
rubygem-addressable: 2.8.5
rubygem-aws-eventstream: 1.2.0
rubygem-aws-partitions: 1.820.0
rubygem-aws-sdk-core: 3.181.0
rubygem-aws-sdk-kms: 1.71.0
rubygem-aws-sdk-s3: 1.134.0
rubygem-aws-sdk-secretsmanager: 1.82.0
rubygem-aws-sigv4: 1.6.0
rubygem-builder: 3.2.4
rubygem-chef: 18.2.7
rubygem-chef-config: 18.2.7
rubygem-chef-telemetry: 1.1.1
rubygem-chef-utils: 18.2.7
rubygem-chef-vault: 4.1.11
rubygem-chef-zero: 15.0.11_2
rubygem-coderay: 1.1.3
rubygem-concurrent-ruby: 1.2.2
rubygem-corefoundation: 0.3.13
rubygem-date: 3.3.3
rubygem-diff-lcs: 1.5.0
rubygem-domain_name: 0.5.20190701
rubygem-erubi: 1.12.0
rubygem-erubis: 2.7.0_1
rubygem-faraday: 2.7.10
rubygem-faraday-follow_redirects: 0.3.0
rubygem-faraday-net_http: 3.0.2
rubygem-ffi: 1.15.5_1
rubygem-ffi-libarchive: 1.1.3
rubygem-ffi-yajl: 2.3.4
rubygem-fuzzyurl: 0.9.0
rubygem-gssapi: 1.3.1
rubygem-gyoku: 1.3.1_1
rubygem-hashie4: 4.1.0
rubygem-http-accept: 2.2.0
rubygem-http-cookie: 1.0.5
rubygem-httpclient: 2.8.3
rubygem-iniparse: 1.5.0
rubygem-inspec-core: 5.22.3
rubygem-ipaddress: 0.8.3
rubygem-jmespath: 1.6.2
rubygem-json: 2.6.3
rubygem-json_pure: 2.6.3
rubygem-libyajl2: 1.2.0
rubygem-license-acceptance: 2.1.13
rubygem-little-plugger: 1.1.4
rubygem-logging: 2.3.1
rubygem-method_source: 1.0.0
rubygem-mime-types: 3.5.1
rubygem-mime-types-data: 3.2023.0808
rubygem-mixlib-archive: 1.1.7
rubygem-mixlib-authentication: 3.0.7
rubygem-mixlib-cli: 2.1.8
rubygem-mixlib-config: 3.0.9
rubygem-mixlib-log: 3.0.9
rubygem-mixlib-shellout: 3.2.5
rubygem-multi_json: 1.15.0
rubygem-multipart-post: 2.3.0
rubygem-net-ftp: 0.2.0
rubygem-net-protocol: 0.2.1
rubygem-net-scp: 4.0.0
rubygem-net-sftp: 4.0.0
rubygem-net-ssh: 7.2.0,2
rubygem-net-ssh6: 6.1.0
rubygem-netrc: 0.11.0
rubygem-nori: 2.6.0
rubygem-ohai: 18.1.3
rubygem-parallel: 1.23.0
rubygem-parslet1: 1.8.2
rubygem-pastel: 0.8.0
rubygem-plist: 3.6.0
rubygem-proxifier2: 1.1.0
rubygem-pry: 0.14.2
rubygem-public_suffix: 5.0.1_8
rubygem-rack22: 2.2.8,3
rubygem-rest-client: 2.1.0
rubygem-rspec: 3.12.0
rubygem-rspec-core: 3.12.2
rubygem-rspec-expectations: 3.12.3
rubygem-rspec-its: 1.3.0
rubygem-rspec-mocks: 3.12.6
rubygem-rspec-support: 3.12.1
rubygem-ruby-termios: 1.1.0
rubygem-ruby2_keywords: 0.0.5
rubygem-rubyntlm: 0.6.3
rubygem-rubyzip: 2.3.2
rubygem-semverse: 3.0.2
rubygem-sslshake: 1.3.1
rubygem-strings: 0.2.1
rubygem-strings-ansi: 0.2.0
rubygem-syslog-logger: 1.6.8
rubygem-thor: 1.2.2
rubygem-time: 0.2.2
rubygem-timeout: 0.4.0
rubygem-tomlrb: 2.0.3
rubygem-tomlrb1: 1.3.0
rubygem-train-core: 3.10.8
rubygem-train-rest: 0.5.0
rubygem-train-winrm: 0.2.13
rubygem-tty-box: 0.7.0
rubygem-tty-color: 0.6.0
rubygem-tty-cursor: 0.7.1
rubygem-tty-prompt: 0.23.1
rubygem-tty-reader: 0.9.0
rubygem-tty-screen: 0.8.1
rubygem-tty-table: 0.12.0
rubygem-unf: 0.1.4
rubygem-unf_ext: 0.0.8.2
rubygem-unicode-display_width: 2.4.2
rubygem-unicode_utils: 1.4.0
rubygem-uuidtools: 2.2.0
rubygem-vault: 0.17.0
rubygem-webrick: 1.8.1
rubygem-winrm: 2.3.6
rubygem-winrm-elevated: 1.2.3
rubygem-winrm-fs: 1.3.5
rubygem-wisper: 2.0.1
rubygem-wmi-lite: 1.0.5
yajl: 2.1.0

Not to be confused with “chef”, the Swedish Chef

A Platform… Configuration “Management”

What do these tools
have in common?

What are they trying to achieve?

A Platform… Configuration “Management”

Idempotence

“A property of some operations such that no matter how many times you execute them, you achieve the same result.”

A Platform… Configuration “Management”

```
sysrc(8)

   sysrc -- safely edit system rc files

   sysrc hostname=current
   sysrc -c hostname=current ; echo $?
   0
```

However, it’s not idempotent

A Platform… Configuration “Management”

```
hostname="happyhost"
if [ "$( sysrc -c hostname=$hostname )" ] ; then
    echo "Hostname $hostname is correct"
    logger "Hostname $hostname is correct"
else
    echo ; echo "Setting hostname $hostname"
    logger "Setting hostname $hostname"
    sysrc hostname="$hostname"
    service hostname restart
fi
```

(See the sample rc.local in the OccamBSD repo)

A Platform… Configuration “Management”

Define a desired state, work to get there

This often involves doing nothing

`sysrc(8)` needs to learn to do nothing
if nothing needs to be done
`fetch(1)-i` needs to remember its job

“Own The Stack” Platform = Foundation

If you need a web|websocket|REST|etc. interface...

The OS is your friend

Human and Machine Read and Writability

“Own The Stack” Platform = Foundation

```
smartctl -a /dev/…
```

Epiphany: NEITHER HUMAN NOR MACHINE READABLE

Added JSON… Changed the schema…

BUT NOT THE SCHEMA VERSION

“Own The Stack” Platform = Foundation

wiki.freebsd.org/LibXo

Thank You Juniper!

In Base

arp df efitable iscsictl jls last lastlogin mount ndp
netstat nfsstat procstat ps savecore sesutil vmstat w wc

In Ports

sysutils/nsysctl sysutils/checkrestart sysutils/smart

“Own The Stack” Platform = Foundation

Thank You Klara!

Coming Soon

OpenZFS JSON Output

“Own The Stack” Platform = Foundation

`wiki.freebsd.org/UniversalConfigurationLanguage`

In Base: `ctld, iovctl`

In Ports: `devel/uclcmd`

“`uclcmd is a command line tool for working with UCL config files.`”

`bhyve_config(5)`: The plumbing for your format of choice!

Where Are We At For FreeBSD 14.0?

- Configurable “Just enough OS” via build options
- FreeBSD Native and Linux ABIs at our disposal
- Paths forward for zero trust and reproducible builds
- Containers and VMs via Jail, bhyve, and Xen
- nullfs file mounts for Jails, .include support
- Integrated OpenZFS with makefs “VM-IMAGE” support
- Paths forward for in-base idempotence
- Human and Machine-readability thanks to UCL/libxo for UIs

The OS is your friend and *is* the appliance

May 1000 Flowers Bloom

We’re in a University…

The OS can do some serious heavy lifting…

Every CS 201 class should be able to build a management interface each September

Rather than fork the OS!

Parallel Efforts: Production Users

Antranig Vartanian: PoC flua Jail front-end for jail API

“Crest”: CTL/virtio-scsi hot-pluggable VM storage

“Crest”: WireGuard RC script, s6rc experiments

jwd@: libnfs VM client, Jail/VM NAS for fast reboots

dch@: Countless things, watch “Immutable FreeBSD”

Weekly Jail, OpenZFS, and bhyve Production User Calls

Thank You Production Users!

Thank You Production Users!

Jail Call: jail.freebsd.am

bhyve call: bhyve.org

OpenZFS call: Deserves a home

Recordings: YouTube.com/@bhyvecon

All Are Welcome

Thank You Developers!

John Baldwin for bhyve Maintainership +++

Corvin Köhne for bhyve development

Jamie Gritton for jail Maintainership

Mark Johnston for `makefs -t zfs` +++

Kristof Provost for a faster bridge +++

The OpenZFS developers for non-suck storage

Everyone for making these tools so good!

Thank you!

Questions?

Michael Dexter

editor@callfortesting.org | @dexter@bsd.network

EuroBSDcon 2023