Jitsi on OpenBSD

Puffy presents video conferencing

Philipp Buehler
sysfive.com GmbH
Hamburg, Germany
pb-openbsd@sysfive.com

***Abstract*—This paper will cover all bits and bolts to fully understand the components at play, their intercommunications and how this knowledge can be used to create a Jitsi-on-OpenBSD setup that features a restricted (compartmentalized) setup using dedicated machines or -as shown- VMM based VMs, where each VM runs only one of the components.**

It'll be documented what's necessary to create a sensible pf.conf on each VM and how to add reverse proxy (relayd, haproxy) for distribution of workload.

Also covering pitfalls/hints along underlying components and what to lookout for on the client/browser side for interopability.

***Index Terms*—Jitsi, OpenBSD, VMM**

I. Introduction

Jitsi and OpenBSD are both not covered much as a documented setup. Installation documents for Jitsi are almost always about Linux OS (and there mostly Debian) and do not cover some internals. The reference documentation on the other hand can be very overwhelming.

There is some FreeBSD "all in one" port (package) with no explanation and it cannot be used to install core components (only) on different nodes.

This documentation is to show a distributed install on OpenBSD using pre-packages and need-to-function (minimum) firewall settings ('pf.conf').

II. Riddles

Both major players show obstacles that have to be overcome to gain a functioning installation.

A. Jitsi

A "full blown" Jitsi installation can consist over over a dozen components and all the necessary networking/firewalling configuration can be exhaustive. Any possible discovery magic is not documented.

Some configuration snippets are undocumented and tend to make the understanding poorer not better. Worst example are necessary DNS settings and 'nginx.conf.'

The typical answer to be found on asking question is to use the official 'all-in-one' Debian VM.

B. OpenBSD

This also leds to the question if it's possible to run a (core) Jitsi installation on OpenBSD only or if there's need be for Linux (VM).

Also it's a bit difficult to find example-based documentation on VMM, e.g. for using VMM as the core router, too. (Combination 'vm.conf'+'pf.conf's).

Can we scale the installation horizontally and how to use Java based applications with 'rcctl'.

III. Components

A. OpenBSD

In this example setup I make heavy use of 'VMM' eco system on OpenBSD which consists of:

- 'vmm(4) - virtual machine monitor:' kernel driver isolating/providing the required resources for the VMs (hypervisor)
- 'vmd(8):' userland daemon to interact with 'vmm'
- 'vmctl(8)': administrative tool to create, start/stop, .. VMs
- 'vm.conf(5)': persist VMs resource configuration

B. Jitsi

A 'core' (basic video conferencing) setup comprised by:

- 'nginx(8)' web: serving web assets and reverse proxy BOSH or websockets
- 'prosody(8)' xmpp: conference chat + internal components communication (esp.PubSub for health/discovery)
- 'jicofo' JItsi COnference FOcus: room+session handling in conferences (whos talking to whom and where)
- 'jvb' videobridge: mediastream (WebRTC) handlings between participants (SFU)
- 'jibri' JItsi BRoadcasting Infrastructure (optional): recording + streaming conferences

IV. Architecture

To host the Jitsi components in a VM each, this uses the following architecture (see Figure 1).

V. Communications

Communications between the components and the logical 'publication' + 'subscription' in Jitsi is as follows (needed in 'pf.conf' later on) (see Figure 2+3).

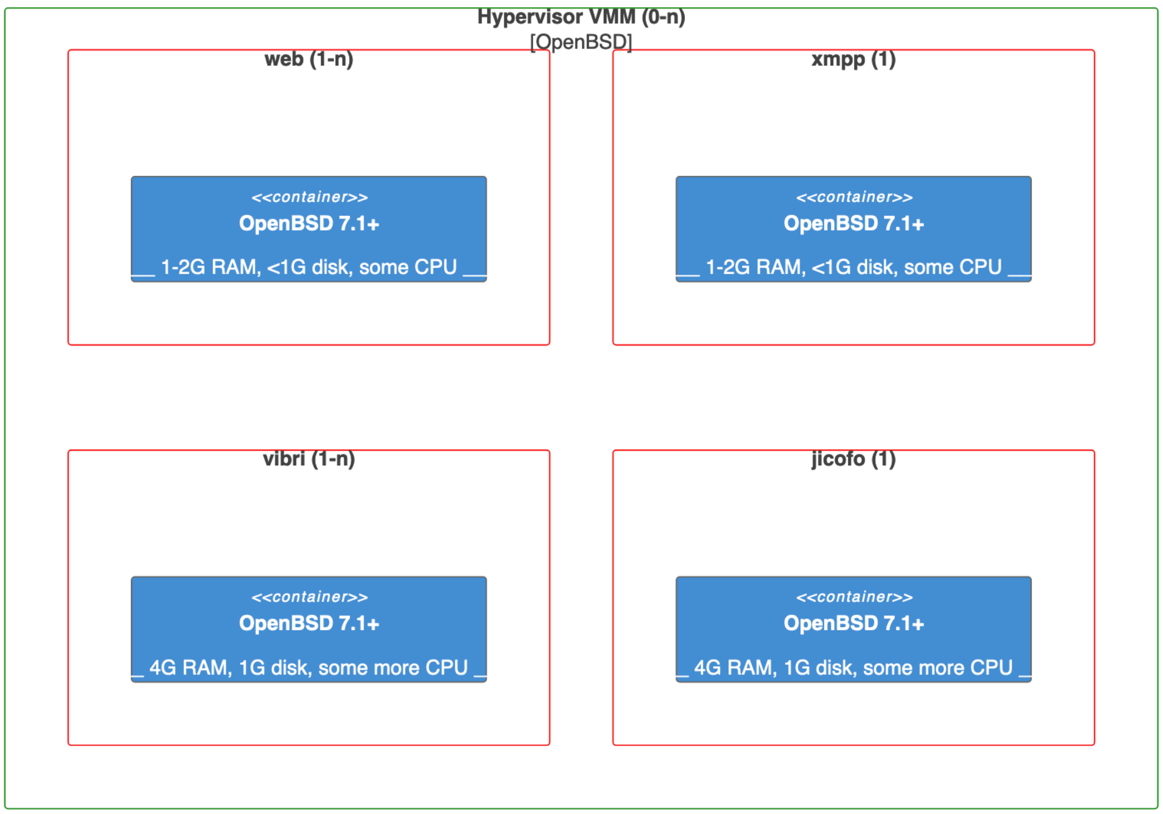

Fig. 1. OpenBSD VMM architecture

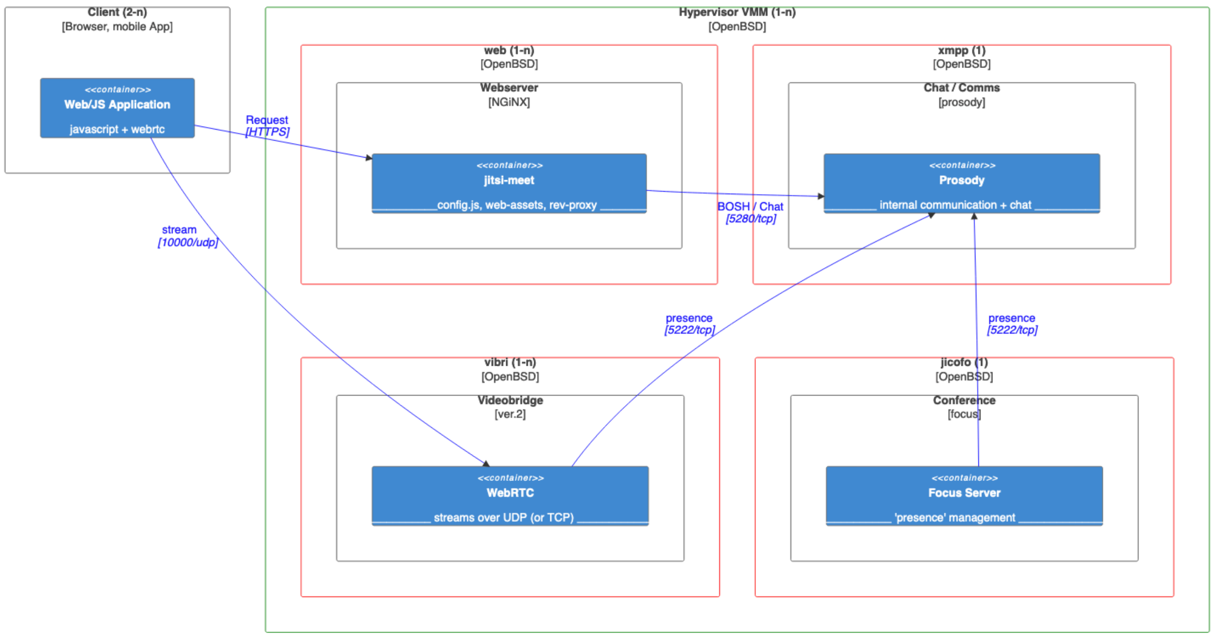

Fig. 2. OpenBSD VMM architecture

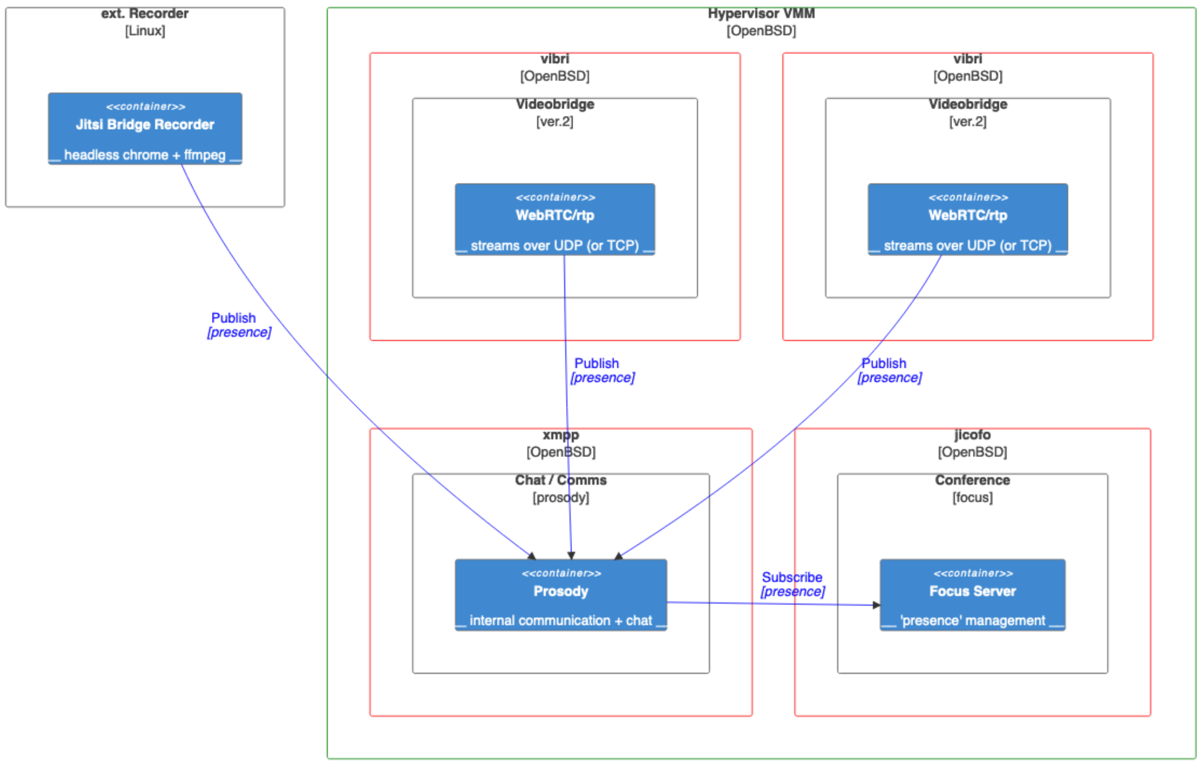

Fig. 3. OpenBSD VMM architecture

VI. Installation

The installation is structered in the following steps:

- create VM images
- construct /etc/vm.conf
- add hosts / DNS
- ‘nginx‘: install, config, certs
- ‘prosody‘: pkg install, config, certs, users
- ‘jicofo‘: pkg install, config
- ‘jvb‘: pkg install, config

A. *VM setup*

To create an VM image being used in the following setup:

```
rcctl enable vmd; rcctl start vmd
mkdir /home/vmm; cd /home/vmm
vmctl create -s 5G web.qcow2
ftp https://cdn.openbsd.org/pub/OpenBSD
/7.1/amd64/install71.iso
vmctl start -m 2G -L -i 1 -r install71.iso\
-d /home/vmm/web.qcow2 web
vmctl console web
## run the (I)nstaller, default options.
## only one 'a' slice on (w)hole disk
## halt -p (so new sshd_keys per VM)
vmctl stop web
for vm in xmpp jicofo jvb ; do
cp web.qcow2 \${vm}.qcow2; done
echo 'net.inet.ip.forwarding=1' >> \\
/etc/sysctl.conf
```

The VM definitions in ‘/etc/vm.conf‘ as follows. The ”instance” tells ‘vmctl‘ to use ”web”’s configuration as a template and only adapt changes like ”disk” or ”memory” to it.

```
vm "web" {
  enable
  memory 2G
  disk "/home/vmm/web.qcow2" format qcow2
  local interface { up }
}
vm "web" instance "xmpp" {
  disk "/home/vmm/xmpp.qcow2" format qcow2
}
vm "web" instance "jicofo" {
  memory 4G
  disk "/home/vmm/jicofo.qcow2" format qcow2
}
vm "web" instance "jvb" {
  memory 4G
  disk "/home/vmm/jvb.qcow2" format qcow2
}
```

B. *DNS + /etc/hosts*

DNS: ONE A-RR for jts.fips.de; but local hosts for jicofo (or split DNS).

The following ‘/etc/hosts‘ needs to be on each VM and on the VMM host. Use these names in ‘/etc/myname‘, too.

```
100.64.1.3    web
100.64.2.3    xmpp jts.fips.de
```

```
100.64.3.3    jicofo
100.64.4.3    jvb
```

VII. Firewalling

On each VM the following ‘pf.conf‘ is for good (admin) measure:

```
block return log
pass out quick on egress proto { tcp udp } to any port { 123 53 80 443 }
pass in quick on egress proto tcp from \$admin to port 22
```

A. VMM

I am assuming all traffic hits the VMM external IP-address (on egress) here.

```
pass in on egress proto tcp to any port { 80 443 } rdr-to web
pass in on egress proto udp to any port { 10000 } rdr-to jvb
pass in proto tcp from { jvb jicofo } to xmpp port 5222 # native
pass in proto tcp from web to xmpp port 5280 # http/BOSH
pass in on egress proto tcp to any port 5280 rdr-to xmpp # debug
# DNS
vms={ web xmpp jicofo jvb }
pass in proto { udp tcp } from \$vms to any port domain rdr-to \$resolver
```

B. web/nginx

C. prosody

D. jicofo

E. videobridge

VIII. Prosody

A. Users

B. TLS

IX. nginx

A. web

B. misc

X. webclient

XI. jicofo

A. Parameters

B. JVM

XII. videobridge

A. Parameters

B. JVM

XIII. Pitfalls

A. OpenBSD

B. Jitsi

XIV. Status

XV. Outlook

XVI. Acknowledgments

XVII. Availability

This paper, presentation slides and other directly related resources can be found on github: https://github.com/double-p/presentations/AsiaBSDCon/2022/

References

[1] OpenBSD project https://www.openbsd.org/
[2] Jitsi https://github.com/jitsi/