Leveraging Vagrant to Simulate Complex Systems for Network Application Development

Barry O'Donovan, INEX

Open Source WG, RIPE 89, Prague, Czechia

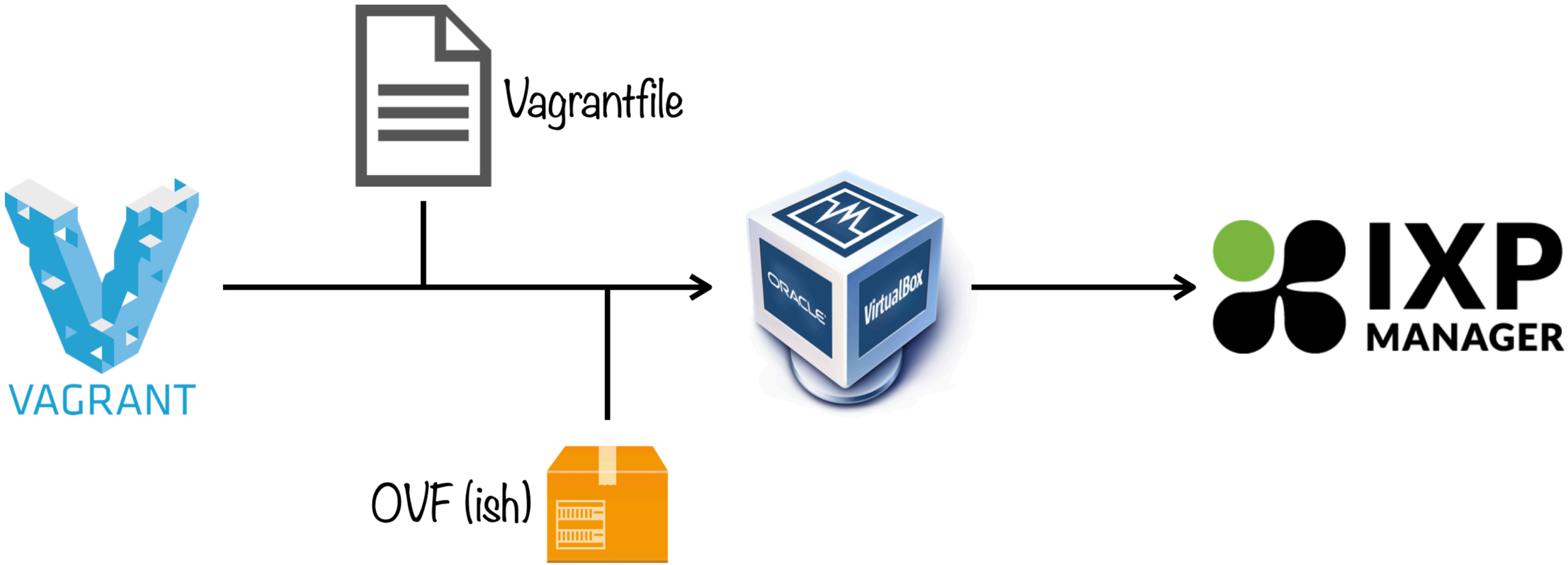
Introduction

- INEX: peering point Ireland; and home of IXP Manager
- Project manager / lead developer
- Created the dev and testing framework of IXP Manager
- Created related projects also, e.g., Bird's Eye, OSS_SNMP

Vagrant

"reproducible, and portable development environments"

Components of a Vagrant Environment



Vagrant Quick Start

```
$ vagrant init bento/ubuntu-24.04

$ vagrant up

$ vagrant ssh
```

Vagrant vs Docker

- Docker's not for me - you do you
- Mirrors the 'recommended' production environment
- Full development platform via `vagrant up`
- "Pure" developers require little sysadmin skills

IXP Manager Use Case #1

Creating a Complex Development Environment

On your laptop

- Code checked out locally
- IDE runs on your machine
- Source code management
- Run the basic application
- `vagrant up => /vagrant`

On Vagrant

- "Production" IXP Manager
- SNMP Simulator
- Bird route servers, etc.
- Bird client instances
- Birdseye looking glass
- mrtg for graphing

Development Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 80, host: 8088
  config.vm.network "forwarded_port", guest: 3306, host: 33061

  config.vm.provider "parallels" do |prl|
    prl.memory = 2048
    prl.cpus = 2
  end

  config.vm.provision :shell, path: "tools/vagrant/bootstrap.sh"
end
```

Development Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 80, host: 8088
  config.vm.network "forwarded_port", guest: 3306, host: 33061

  config.vm.provider "parallels" do |prl|
    prl.memory = 2048
    prl.cpus = 2
  end

  config.vm.provision :shell, path: "tools/vagrant/bootstrap.sh"
end
```

Development Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 80, host: 8088
  config.vm.network "forwarded_port", guest: 3306, host: 33061

  config.vm.provider "parallels" do |prl|
    prl.memory = 2048
    prl.cpus = 2
  end

  config.vm.provision :shell, path: "tools/vagrant/bootstrap.sh"
end
```

Development Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 80, host: 8088
  config.vm.network "forwarded_port", guest: 3306, host: 33061

  config.vm.provider "parallels" do |prl|
    prl.memory = 2048
    prl.cpus = 2
  end

  config.vm.provision :shell, path: "tools/vagrant/bootstrap.sh"
end
```

Development Vagrantfile

```
Vagrant.configure(2) do |config|
  config.vm.box = "bento/ubuntu-24.04"

  config.vm.network "forwarded_port", guest: 80, host: 8088
  config.vm.network "forwarded_port", guest: 3306, host: 33061

  config.vm.provider "parallels" do |prl|
    prl.memory = 2048
    prl.cpus = 2
  end

  config.vm.provision :shell, path: "tools/vagrant/bootstrap.sh"
end
```

bootstrap.sh

inex.ie/ripe89-bootstrap

bootstrap.sh

- Provisions new virtual machine
- On reboot, a service runs `tools/vagrant/startup.sh`
- `artisan` commands for more complex tasks
- NB: we're not 'installing' IXP Manager, just enabling it in Vagrant

Goal: enable you to create your own complex Vagrant environment.

Basic Set-up

```
/usr/bin/timedatectl set-timezone 'Europe/Dublin'

echo "Updating packages...."
apt-get update &>/dev/null
# apt-get dist-upgrade &>/dev/null

# Defaults for MySQL, etc.
echo 'mysql-server mysql-server/root_password ...' | debconf-set-selections
echo 'mysql-server mysql-server/root_password_again ...' | debconf-set-selections
```

Install Packages for IXP Manager

```
echo "Installing apache, php, etc..."
apt-get install -y apache2 php8.3 php8.3-intl php8.3-mysql php-rrd \
    php8.3-cgi php8.3-cli php8.3-snmp php8.3-curl php8.3-memcached \
    php8.3-cgi bash-completion php8.3-mysql memcached snmp          \
    php8.3-mbstring php8.3-xml php8.3-gd bgpq3 php8.3-memcache      \
    unzip php8.3-zip git php8.3-yaml php8.3-bcmath mrtg rrdtool     \
    mysql-server mysql-client joe screen phpmyadmin  &>/dev/null
```

- php-rrd, php-snmp not easily installed on MacOS

MySQL Database

- Vagrant installs development database and CI database
- Development database has:
  - Two infrastructures (IXPs)
  - Sample members: Eyeball, WISP, CDN, NREN, ...
  - Route servers, collectors, AS112
- Models various different states

bootstrap.sh

Bird / BGP Daemons

Install Bird2

```
$ apt-get -y install bird2 &>/dev/null
```

Pro tip: Multiple Bird instances on the loopback interface requires:

```
strict bind yes;
multihop;
```

Add router IP addresses to loopback

```
$ IPS=`mysql ... --skip-column-names --silent --raw ixp \
  -e 'SELECT DISTINCT ipaddr.address
        FROM ipv4address as ipaddr ...'`

$ for ip in $IPS; do /usr/sbin/ip address add $ip/24 dev lo; done
```

Configure and Start IXP Routers

```
$ /vagrant/tools/vagrant/scripts/rs-api-reconfigure-all.sh
$ /vagrant/tools/vagrant/scripts/rc-reconfigure.sh
$ /vagrant/tools/vagrant/scripts/as112-reconfigure-bird2.sh
```

- Downloads configurations using IXP Manager API
- Starts Bird daemons
- In dev environment, these start 16 routers
  - 2 infrastructures x ipv4/ipv6 x [RC+RS1+RS2+AS112]

Creating Member Routers

IXP Manager has the data to create dual-stack clients for testing:

- Peering IPs, AS number and MD5
- Routes to advertise via IRRDB (for route server config)
- All available routers (as112, collector, route servers)

```
$ artisan vagrant:generate-client-router-configurations
$ /srv/clients/start-reload-clients.sh
```

Creating Member Routers

```
$ birdc -s /var/run/bird/as25441-vix15.ctl
BIRD 2.14 ready.
bird> show protocols
Name              Proto      Table      State  Since         Info
device1           Device     ---        up     2024-10-25 15:24:59
static_bgp4       Static     master4    up     2024-10-25 15:24:59
rs1_vix1_ipv4     BGP        ---        up     2024-10-25 15:25:03  Established
rs2_vix1_ipv4     BGP        ---        up     2024-10-25 15:25:04  Established
rc1_vix1_ipv4     BGP        ---        up     2024-10-25 15:25:02  Established
as112_vix1_ipv4   BGP        ---        up     2024-10-25 15:25:03  Established
static_bgp6       Static     master6    up     2024-10-25 15:24:59
rs1_vix1_ipv6     BGP        ---        up     2024-10-25 15:25:02  Established
rs2_vix1_ipv6     BGP        ---        up     2024-10-25 15:25:04  Established
rc1_vix1_ipv6     BGP        ---        up     2024-10-25 15:25:02  Established
as112_vix1_ipv6   BGP        ---        up     2024-10-25 15:25:03  Established
```

bootstrap.sh

Birds Eye Looking Glass

Birds Eye

"A Simple Secure Micro Service for Querying Bird"

- Needs to be installed on the same server as Bird
- Needs http access from server running IXP Manager
- Needs per Bird instance configuration file

Birds Eye via bootstrap.sh

```
$ git clone https://github.com/inex/birdseye.git /srv/birdseye
$ cd /srv/birdseye
$ git checkout php83 &>/dev/null
$ COMPOSER_ALLOW_SUPERUSER=1 composer install &>/dev/null

$ cat >/etc/apache2/sites-enabled/birdseye.conf
...

$ php /vagrant/artisan vagrant:generate-birdseye-configurations
```

Looking Glass Route Collector #1 - VIX1 - IPv4

Route Collector #1 - VIX1 - IPv4

Bird v2 2.14 | API: 1.2.2 | Router ID: 192.0.2.126 | Uptime: 0 days. | Last Reconfigure: 2024-10-25 15:24:57 | JSON: [status] [bgp]

Search:

Neighbor	Description	ASN	Table	PfxLimit	State/PfxRcd	PfxExp	Actions
192.0.2.6	AS112 - AS112	112	master4		2	0	Details
192.0.2.10	AS1213 - NREN	1213	master4		55	0	Details
192.0.2.13	AS2906 - CDN	2906	master4		66	0	Details
192.0.2.11	AS25441 - Eyeball ISP	25441	master4		96	0	Details
192.0.2.12	AS25441 - Eyeball ISP	25441	master4		96	0	Details
192.0.2.14	AS39093 - Regional WISP	39093	master4		4	0	Details
192.0.2.8	AS65501 - VAGRANTIX Route Servers	65501	master4		223	0	Details
192.0.2.9	AS65501 - VAGRANTIX Route Servers	65501	master4		223	0	Details

bootstrap.sh

Modelling IXP Switches

SNMP Simulation

- Switch modelling only requires SNMP access
- We use the snmpsim package:
  - https://github.com/etingof/snmpsim

Sourcing Realistic SNMP Data

- Data from three real IXP switches

```
$ snmpsim-record-commands --agent-udpv4-endpoint=192.0.2.45 \
    --output-file=./data/public.snmprec
```

- Edited to model different states
  - Port speed; up/down; transceiver type; etc.
  - Matches port assignments in dev database

Querying the SNMP Switches

```
$ ls /vagrant/tools/vagrant/snmpwalks/
swi1-fac1-1.snmprec  swi1-fac2-1.snmprec  swi2-fac1-1.snmprec

vagrant:~$ sudo netstat -lpn | grep 161
udp  127.0.0.1:161  ...    24971/python3

$ snmpget -v 2c -c swi1-fac1-1 127.0.0.1 .1.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Arista Networks EOS version ..."
```

=> the community decides on the `.snmprec` file to query.

Querying the SNMP Switches

```
$ ls /vagrant/tools/vagrant/snmpwalks/
swi1-fac1-1.snmprec  swi1-fac2-1.snmprec  swi2-fac1-1.snmprec

vagrant:~$ sudo netstat -lpn | grep 161
udp  127.0.0.1:161  ...    24971/python3

$ snmpget -v 2c -c swi1-fac1-1 127.0.0.1 .1.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Arista Networks EOS version ..."
```

=> the community decides on the `.snmprec` file to query.

Querying the SNMP Switches

```
$ ls /vagrant/tools/vagrant/snmpwalks/
swi1-fac1-1.snmprec  swi1-fac2-1.snmprec  swi2-fac1-1.snmprec

vagrant:~$ sudo netstat -lpn | grep 161
udp  127.0.0.1:161  ...    24971/python3

$ snmpget -v 2c -c swi1-fac1-1 127.0.0.1 .1.3.6.1.2.1.1.1.0
iso.3.6.1.2.1.1.1.0 = STRING: "Arista Networks EOS version ..."
```

=> the community decides on the `.snmprec` file to query.

What does this mean for development?

- Enables Grapher development against a 'real' system
  - E.g., MRTG daemon can query switches and generate graphs
- IXP Manager's SNMP polling can be tested
  - IXP Manager's scheduler fully enabled in Vagrant
- New diagnostics functionality developed with snmpsim
- UI development and testing using OSS_SNMP

Transceiver diagnostics for: swi1-fac1-1 :: Ethernet49/1 [Physical Interface #10]

INFO

Transceiver information - FLEXOPTIX Q.161HG.10, serial #F79E8BW

Manufacturer: FLEXOPTIX

Model: Q.161HG.10

Serial: F79E8BW

INFO

Light levels - click info for switch-like output

L	Bias Current (mA)	Optical Tx Power (dBm)	Optical Rx Power (dBm)
1	+42.18	+0.01	-1.60
2	+47.26	-0.42	-0.14
3	+47.27	+0.48	-0.77
4	+44.08	+1.58	-1.15

Takeaways

Takeaways

- Typically developers are not sysadmins or netengs.
  - Easier to hire developers without these requirements!
- Complex environments as easy as `vagrant up`.
- Everything presented today in `release-v7` at github.com/inex/IXP-Manager

Thank You

barry.odonovan@inex.ie

IXP Manager Use Case #2

Testing the Automated Installer

The Automated Installer

We provide a complete script to install IXP Manager from a brand new minimal install of Ubuntu.

```
>>> download the installation script <<<
wget https://github.com/inex/IXP-Manager/.../ubuntu-lts-2004-ixp-manager-v6.sh

>>> and execute it: <<<
bash ./ubuntu-lts-2004-ixp-manager-v6.sh
```

We need to be able to efficiently test this.

Automated Installer Vagrantfile

- Shipped in `inex/IXP-Manager/tools/installers/`

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.provision "shell",
            path: "ubuntu-lts-2004-ixp-manager-v6.sh",
            args: "--no-interaction"
end
```

Automated Installer Vagrantfile

- Shipped in `inex/IXP-Manager/tools/installers/`

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.provision "shell",
            path: "ubuntu-lts-2004-ixp-manager-v6.sh",
            args: "--no-interaction"
end
```

Automated Installer Vagrantfile

- Shipped in `inex/IXP-Manager/tools/installers/`

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.provision "shell",
            path: "ubuntu-lts-2004-ixp-manager-v6.sh",
            args: "--no-interaction"
end
```

Automated Installer Vagrantfile

- Shipped in `inex/IXP-Manager/tools/installers/`

```
Vagrant.configure(2) do |config|
  config.vm.box = "ubuntu/focal64"
  config.vm.network "forwarded_port", guest: 80, host: 8080
  config.vm.provision "shell",
              path: "ubuntu-lts-2004-ixp-manager-v6.sh",
              args: "--no-interaction"
end
```

Testing the Automated Installer

```
$ mkdir test-installer && cd test-installer
$ cp .../Vagrantfile.ubuntu-lts-2004-ixp-manager-v6 .
$ vagrant up
$ open http://localhost:8080/
$ vagrant destroy
```

Testing the Automated Installer

```
$ mkdir test-installer && cd test-installer
$ cp .../Vagrantfile.ubuntu-lts-2004-ixp-manager-v6 .
$ vagrant up
$ open http://localhost:8080/
$ vagrant destroy
```

Testing the Automated Installer

```
$ mkdir test-installer && cd test-installer
$ cp .../Vagrantfile.ubuntu-lts-2004-ixp-manager-v6 .
$ vagrant up
$ open http://localhost:8080/
$ vagrant destroy
```

Thank You

barry.odonovan@inex.ie