Learning Automation Without Barriers Using Antidote and NRE Labs
Matt Oswalt - @Mierdin
Juniper Networks
About Me

Matt Oswalt
@mierdin

- 8 years in the industry
  - Network engineer
  - Consultant
  - Developer
  - Open Source Maintainer
- Currently at Juniper Networks
- Passionate about skills evolution
Core networking fundamentals still matter. It’s right in the name.

Represented a better way of doing things. Emphasizes the true goal of automation.

You can’t buy engineering - you DO it. Sidesteps the “productization” of automation.

Codify  Automate  Test

Monitor  Measure
Time Investment Minimum (TIM) - HIGH

Knowledge of a Subject

“This will be useful to me”

“I am competent with this”

Time and Energy Spent Learning Subject

NRE LABS
Time Investment Minimum (TIM) - LOW

Knowledge of a Subject vs. Time and Energy Spent Learning Subject

- "This will be useful to me" at the lower end
- "I am competent with this" at the higher end

NRE LABS
Community platform for learning and teaching automation and Network Reliability Engineering

- Totally browser-based
- Free - no login, paywall or creepy trackers
- Vendor-neutral
- Open Source (curriculum too!)
In this part (Part 2), our routers have been configured with the correct BGP peers. We can verify this by checking on the current BGP summary:

```
show bgp summary
```

It looks good, but as they say, “successful tests or it didn’t happen”. Let’s re-run JSNAPy to make sure our tests are passing with the new configuration:

```
jsnapy --snapshot -f jsnapy_config.yaml -v
```

This time, our network is behaving the way we’ve declared in the tests, so they pass. It’s important to note that our tests not only assert that the right configuration exists, but that the operational state of each router’s BGP peer status is correct. This is a nice feature of JSNAPy - it can make assertions over anything in the entire Junos data model.

This was a lightning-quick introduction to JSNAPy. Please see the wiki for more details - there’s a lot more capability than we covered here.
Today → Automated Workflow
Today

Fundamentals
- YAML
- Bash
- Jinja
- Python

Tools
- JSNAPy
- NAPALM

Workflow

Compliance Validation
Why should I care?

- Chance for the community to take back control of ops education
- Fairly new project - lots to do
- Covers a wide spectrum of disciplines
  - frontend, systems programming, ops, content
- Several cutting-edge technologies in use now or in the near future
Creating Lessons
Antidote Architectural Overview

Curriculum

Platform

Infrastructure

Kubernetes

BMaaS (Packet)
Scripts and Kubernetes Manifests for deploying Antidote on Minikube

https://github.com/nre-learning/antidote-selfmedicate/

Easily preview curriculum content locally before submitting a PR.
Flexible Presentation Layer

Learner’s Browser

Endpoints (any container)

- SSH
- Web (via iframe)
- VNC (soon)

Lab Guide

- Jupyter Notebook (full endpoint connectivity)
- Markdown to HTML
Inter-Stage Configuration

- Hands-Free environment prep while moving within a lesson
- Configurations present within lesson directory will be applied during stage transitions
- Currently supports NAPALM, Ansible, or Custom Python

Lesson Namespace

- Configurator
- Endpoints (any)
- Originally inspired by vrnetlab but currently a bit more bespoke. Hoping to get more standardized soon.
- Images packaged straight into docker and executed by the kubelet on the scheduled host
- ANY vendor is feasible as long as it runs in a VM and talks on a port

**Image Catalog**

**Current**
- vQFX
- vMX
- Cumulus VX

**Soon**
- VyOS
- ExtremeXOS
Diving Deeper...
Antidote as Deployed in Kubernetes

Kubernetes API Server

HTTP

REST

grpc

syringe

HTTP

nginx ingress-controller

HTTP

antidote-web

guacamole

guacd

HTTP

SSH or VNC

Namespace 12-abcdef

vqfx

Namespace 14-fedcba

vqfx

jupyter

Lesson Instances

Syringe Health Checks
## Anatomy of a Lesson

### Namespace: 12-abcdef

<table>
<thead>
<tr>
<th>Jobs</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>config-vqfx1</td>
</tr>
<tr>
<td></td>
<td>config-vqfx1</td>
</tr>
<tr>
<td></td>
<td>config-vqfx1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Networks (CRD)</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>net-vqfx1-vqfx2</td>
</tr>
<tr>
<td></td>
<td>net-vqfx1-vqfx2</td>
</tr>
<tr>
<td></td>
<td>net-vqfx1-vqfx2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ingresses</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>jupyter-iframe</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>NetworkPolicy</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>stoneage</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Services</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vqfx1</td>
</tr>
<tr>
<td></td>
<td>vqfx2</td>
</tr>
<tr>
<td></td>
<td>vqfx3</td>
</tr>
<tr>
<td></td>
<td>linux</td>
</tr>
<tr>
<td></td>
<td>jupyter</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Pods</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>vqfx1</td>
</tr>
<tr>
<td></td>
<td>vqfx2</td>
</tr>
<tr>
<td></td>
<td>vqfx3</td>
</tr>
<tr>
<td></td>
<td>linux</td>
</tr>
<tr>
<td></td>
<td>jupyter</td>
</tr>
</tbody>
</table>
Normal Kubernetes Pod Networking

Pod Network
(CNI plugin i.e. Linux Bridge)

eth0  eth0  eth0
vqfx1  vqfx2  vqfx3
Using Multus for Advanced Network Topologies

Alternatives:
- QEMU L2TP
- NetworkServiceMesh

https://github.com/intel/multus-cni
The Road to v1.0
The Road to v1.0

- v0.3.x - Q1 2019
  - Advisor

- v0.4.x - Q3 2019
  - Collections
  - BMaaS

- v0.5x
  - Syringe Redesign

- v0.6.x
  - Embeddable Lessons

- v0.7.x

- v0.8.x

- v0.9.x

- v1.0.0
Advantages:
- Single binary
- No external database to worry about
- Allowed us to get NRE Labs public quick

Disadvantages
- Single point of failure
- Everything is tightly coupled, harder to extend
- State is kept in-memory, so restart means state is lost
  - This means we need to kill all existing lessons on start
- Fairly opaque
Better resilience - no "one syringe"
Easier to extend - just pop a new service on the message bus
Easier to reason about, maintain, and contribute to individual services
User Experience

✔ Are users having problems?
  – Monitoring components is easy, monitoring the full thing is hard?

✔ If they are, what can we even do about it?
  – By definition, our users aren’t experts in Github
  – In the 0.01% of cases where users find a way to get feedback to us, all of the context is lost.

THIS JOB WOULD BE GREAT

IF IT WEREN'T FOR THE USERS
User feedback right in the UI. Click this button or type in this box to tell us about a problem. Responses go to some kind of queue for filtering and triage. Includes session and request IDs.

Better centralized and structured logging

System observability - Tracing from web front-end all the way through every syringe microservice. High cardinality based from initial session and lesson ID allows us to get to a specific interaction easily.
Target: security of VMs with developer experience of containers

Candidates:
- Weave Ignite
- Kata Containers
- Custom tooling

KubeVirt
- May help in future but this is mostly focused on ops-side stuff we don’t need.
✔ Hiring open source web dev firm to do a UX review and give us a new base
✔ Like......MOBILE SUPPORT
✔ Will still have a ton of work left for the community to do
Come full-circle and enable lesson contributions in the browser

Using Mozilla Janitor as inspiration (uses Amazon Cloud9)

Have to figure out a cost-effective deployment model
Resources

Labs - labs.networkreliability.engineering
Community - community.networkreliability.engineering
Open Source - github.com/nre-learning
Antidote Docs - antidoteproject.rtfd.io
Standups - Every Tuesday 8AM Pacific
Twitter - @NRELabs

No Contribution Too Small!
- Use NRE Labs and open issues!
- Lesson Contributions - new or existing
- Platform enhancements/fixes
Questions?