

VIRL CI Workbench

Using VIRL in a NetOps Tool-chain

Ralph Schmieder
Technical Leader
August, 2017

Agenda

- Introduction
- VIRL APIs, Jenkins, GitHub
- Python tool, YAML-config
- Tying it together: Demo
- Hands-on Lab
- Q&A

Pod Assignments

- GitHub Repo **<https://github.com/rschmied/gsx2018-virl>**
- Pods are hosted on packet.net (the easy and fast Virl cloud option)
- Type 0 servers (not very powerful but will do for the workbench)
- Have a **user 'guest'** with a **password 'gsx2018'**
- Pods go from **pod1.virl.info** to **pod8.virl.info**
- Make a note of the info above, log in from your workstation
- Don't mix up the pods ☺

Introduction

CI – Continuous Integration

- Why
- How
- Tools
- Specific Use Cases

VIRL APIs

- Control entire Sim Lifecycle via API
- Complete documentation is built into VIRL
- Lists each RESTful call, parameters, and behaviors

The screenshot shows the Cisco UWM (Universal Management) interface. The left sidebar has a navigation menu with items like Overview, My simulations, Project simulations, Projects, Users, VIRL Server, Connectivity, VM Control, Node resources, Repositories, Documentation, and three tabs at the bottom: STD API (selected), UWM API, and Routem. The main content area is titled "VIRL STD API". It lists several API endpoints under two sections: "admin" and "simengine". Under "admin", there are entries for autonetkit, catalog, links, openstack, and roster. Under "simengine", there are several RESTful calls with their methods, URLs, and descriptions:

- DELETE /simengine/rest/shaping/{simulation_id}**: Delete link-level traffic control settings for all links in simulation.
- GET /simengine/rest/shaping/{simulation_id}**: List link-level traffic control settings.
- GET /simengine/rest/shaping/{simulation_id}/{link_id}**: List link-level traffic control settings for a given link.
- PUT /simengine/rest/shaping/{simulation_id}/{link_id}**: Set link-level traffic control settings for particular link.
- GET /simengine/rest/link/{simulation_id}**: Get link identifiers for selected simulation.
- GET /simengine/rest/link/{simulation_id}/{link_id}**: Get info about a given link.
- GET /simengine/rest/systemlogs**: Get a zip-file of all current logs.

Required Tooling



- **GitHub**

Topologies, Configurations, Tests

- **Jenkins**

Gets notified, runs sim, reports success

- **VIRL**

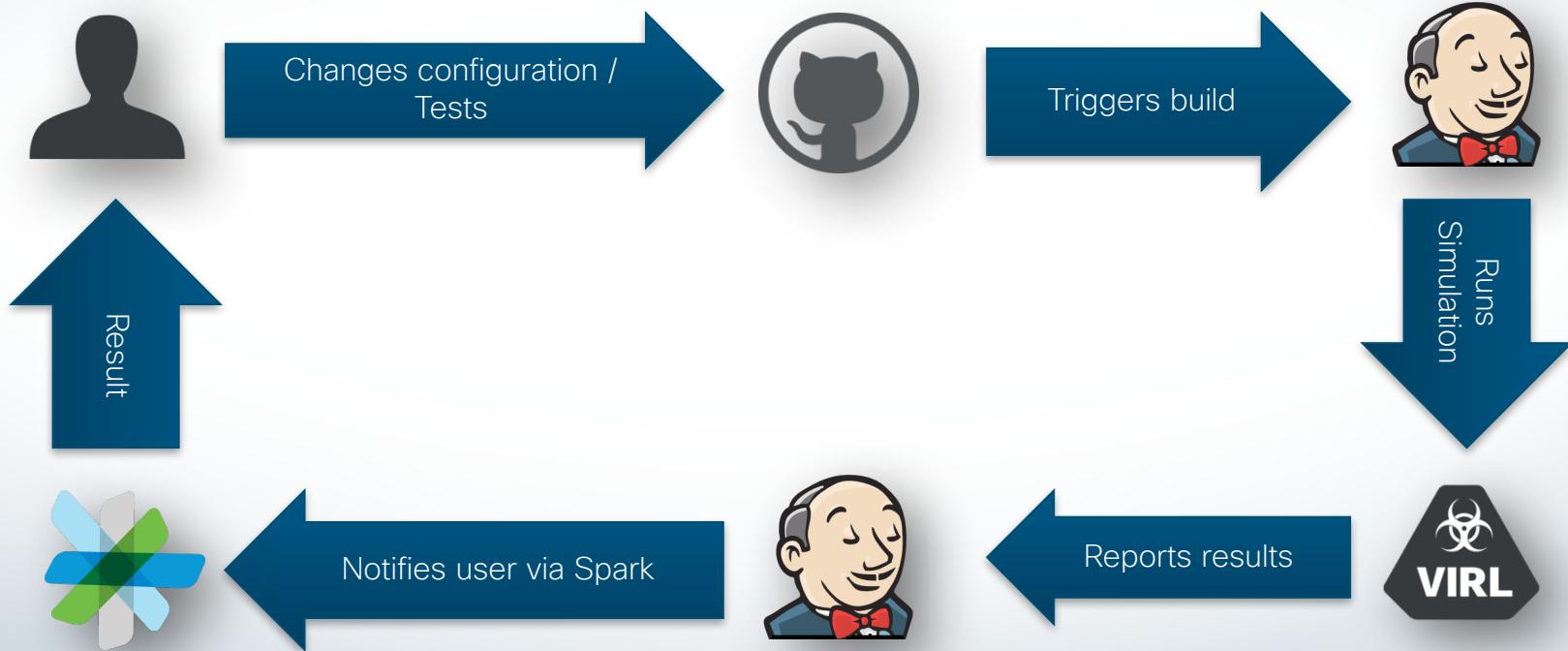
Simulates virtual network, executes tests

- **Cisco Spark and e-mail**

Notify about success / failure



CI Workflow



Sample Input / Output

```
5 projects/IOSv/iosv-1.cfg
6
7 @@ -81,6 +81,11 @@ interface GigabitEthernet0/2
81    81      no shutdown
82    82      !
83    83      !
84    84      +interface GigabitEthernet0/0.100
85    85      + encapsulation dot1Q 100
86    86      + ip address 172.100.100.100 255.255.255.0
87    87      + no cdp enable
88    88      +!
89    89      !
90    90      router ospf 1
91    91          network 192.168.0.2 0.0.0.0 area 0
92
93
```

GSX2018 CI VIRL API Demo

This starts the "GSX2018 CI VIRL API Demo" space. 9:09 AM



GitHubSparky (bot) 9:14 AM

Jenkins GSX 2018 build #134 SUCCESS <https://clus.virl.info:8443/job/clus/134/>



GitHubSparky (bot) 9:17 AM

Jenkins GSX 2018 build #135 SUCCESS <https://clus.virl.info:8443/job/clus/135/>

```
23
24 sims:
25   - topo: iosv-single-test.virl
26   nodes:
27     - lxc-1:
28       - type: command
29         transport: ssh
30         sleep: 10
31         in:
32           #- ping -W1 -c1 10.0.0.10
33           - ping -c5 10.0.0.10
34           - ping -c5 10.0.0.10
35         out:
36           - \ 0% packet loss
37     - lxc-2:
38       - type: command
39         transport: ssh
40         in:
41           - ping -W1 -c1 10.0.0.6
42           - ping -c5 10.0.0.6
43         out:
44           - \ 0% packet loss
```

Configure

GitHub Hook Log

GitHub

Build History

#122 Jun 26, 2017 6:20 PM

#121 Jun 26, 2017 5:35 PM

```
>>> ping -W1 -c1 10.0.0.6
<<< PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
 64 bytes from 10.0.0.6: icmp_seq=1 ttl=63 time=1.33 ms

--- 10.0.0.6 ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 1.334/1.334/1.334/0.000 ms

>>> ping -c5 10.0.0.6
<<< PING 10.0.0.6 (10.0.0.6) 56(84) bytes of data.
 64 bytes from 10.0.0.6: icmp_seq=1 ttl=63 time=1.28 ms
 64 bytes from 10.0.0.6: icmp_seq=2 ttl=63 time=0.829 ms
 64 bytes from 10.0.0.6: icmp_seq=3 ttl=63 time=1.63 ms
 64 bytes from 10.0.0.6: icmp_seq=4 ttl=63 time=2.70 ms
 64 bytes from 10.0.0.6: icmp_seq=5 ttl=63 time=1.78 ms

--- 10.0.0.6 ping statistics ---
5 packets transmitted, 5 received, 0% packet loss, time 4003ms
rtt min/avg/max/mdev = 0.829/1.647/2.708/0.625 ms
```

Demonstration



Hands-on Lab



Cursory Steps / Guideline

In a Terminal...

- `virl-tester.zip` file in ~/ (home directory), unzip (creates virl-tester dir)
- Create virtual environment (`python3 -m venv virl-ci`), source settings (`source virl-ci/bin/activate`)
- Install tool into venv (`pip install virl-tester`)
- Clone example (`git clone https://github.com/rschmied/gsx2018-virl.git`), cd into it (`cd gsx2018-virl`)
- Explore directory in Atom (or editor of choice)

What we'll do / suggestions

1. Create and run topology in VIRL Web Editor (three node triangle)
2. Save topology locally into directory
3. Check for routes, ping-ability in live sim. Make note of input to send and expected output
4. Stop sim manually (resources!)
5. Create new test file (.yaml) based on existing file in dir
`'gsx2018-virl'`
6. Run tester `'virltester -l4 yourtestfile.yml'`
7. Experiment ☺

Take-Away

- Automate with APIs!
- Repeatability!
- No fat-fingering
- When things fail, write a test

Resources

- Internal Repository for the Test Tool
 - <https://wwwin-gitlab-sjc.cisco.com/rschmied/virl-tester.git>
 - `git clone https://wwwin-gitlab-sjc.cisco.com/rschmied/virl-tester.git`
- Test Data Example
 - <https://github.com/rschmied/clus-virl>
 - <https://wwwin-gitlab-sjc.cisco.com/rschmied/virl-tests.git>
- Note: Better Documentation and Examples needed, contribute ☺

GSX *Own it.*