Automating IP networks with Python

Tomás Lynch
LACNOG 2019
What would you choose?

Life with or without automation?
Actually...

Life without automation

Life with automation
Standardization before automation

Automation is useless without standardized configuration

Naming convention, same OS version, etc. are automation fundamental pieces

Automation relies on regular expressions

Example: add a prefix list to all edge routers:

```
router.edge1.ar, router.edge1.br, router.edge1.co
vs.
diego10.gimnasia, router-garrincha, co5ar0_edge
```
Automation and Python
Python network element packages

ncclient
- Juniper, Cisco CSR, Cisco Nexus, Huawei, Alcatel Lucent, H3C, HP
- netconf only

PyEZ
- Juniper

netmiko
- Arista, Cisco IOS, Juniper, MikroTik, among others

And 6,594 projects more
### show lldp neighbors

<table>
<thead>
<tr>
<th>PyEZ - predefined table</th>
<th>netmiko - CLI</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>connect</strong></td>
<td></td>
</tr>
<tr>
<td>dev = Device(</td>
<td></td>
</tr>
<tr>
<td>host=router,</td>
<td></td>
</tr>
<tr>
<td>user=uname,</td>
<td></td>
</tr>
<tr>
<td>password=pw</td>
<td></td>
</tr>
<tr>
<td>)</td>
<td></td>
</tr>
<tr>
<td>dev.open()</td>
<td></td>
</tr>
<tr>
<td><strong>command</strong></td>
<td></td>
</tr>
<tr>
<td>router_lldp = LLDPNeighborTable(dev)</td>
<td></td>
</tr>
<tr>
<td>lldp_neighbors = router_lldp.get()</td>
<td></td>
</tr>
<tr>
<td><strong>disconnect</strong></td>
<td></td>
</tr>
<tr>
<td>dev.close()</td>
<td></td>
</tr>
<tr>
<td><strong>device</strong> =</td>
<td></td>
</tr>
<tr>
<td>'device_type': 'brocade',</td>
<td></td>
</tr>
<tr>
<td>'ip': router,</td>
<td></td>
</tr>
<tr>
<td>'username': uname,</td>
<td></td>
</tr>
<tr>
<td>'password': pw,</td>
<td></td>
</tr>
<tr>
<td>'port': port,</td>
<td></td>
</tr>
<tr>
<td>'secret': enablepass</td>
<td></td>
</tr>
<tr>
<td>ssh_connect=Netmiko(**device)</td>
<td></td>
</tr>
<tr>
<td>ssh_connect.enable()</td>
<td></td>
</tr>
<tr>
<td>ssh_connect.send_command('skip-page-display')</td>
<td></td>
</tr>
<tr>
<td>lldp_neighbors = ssh_connect.send_command('show lldp neighbors detail')</td>
<td></td>
</tr>
<tr>
<td>ssh_connect.disconnect()</td>
<td></td>
</tr>
</tbody>
</table>
Package performance

PyEZ

Predefined operational table
10 routers
15 seconds
Output: lldp_neighbors
Dictionary
Ready to use!

netmiko (ssh)

Command-line interface
10 routers
1 minute 38 seconds
Output: lldp_neighbors
Plain text
More processing
Automation and Python in use
The network

Cloud servers, bare metal, and storage

16 worldwide locations

1600 network elements in Clos topology

Automation using puppet, python, etc.
Example 1: update_bgp_peer

13 Public Peering Exchange Points
17 Private Peering facilities
1100 peers aprox.
Example 2: interface_description
Other developed scripts for BGP

configure_customer_bgp
remove_customer_bgp
get_bgp_summary
update_transit_config
enable_sflow_everywhere

and many more for maintenance, server activation, etc.
Conclusions, recommendations, and references
Conclusions

Standardization is the most important step before automation

Automate repetitive and boring tasks

    Peering information, standards verification, massive changes, etc.

Use complete commands: “show running-config” instead of “sh ru”
Recommendations

Do not spend time in once in a lifetime scripts

   Use your old friends: grep, awk, etc.

If no experience: start with non-disrupting commands

Use vendor specific packages if possible

Do not store passwords in scripts!
References

Network automation – juni.pr/2YVgjVj

netmiko platforms – bit.ly/2Tf6Oeo

PyEZ – juni.pr/2YSmf1g

BGP summary using PyEZ – www.inetzero.com/pyez

bgpq3 – github.com/snar/bgpq3

Use of BGP for Routing in Large-Scale Data Centers – RFC7938
Thank you!

Tomas Lynch

tlynch at vultr dot com
Backup slides

Tomas Lynch

tlynch@vultr.com
Automation
What is network automation?

Process of automating:

configuration,
management,
testing,
deployment, and
operations

Also called network programmability
Automation block diagram

1. Variables
   - Device name
   - ASN
   - IP address
   - Description
   - Etc.

2. Script

3. API
   - REST
   - XML
   - JSON
   - NETCONF

4. Infrastructure
   - Router
   - Switch
   - Server
   - Etc.
Also monitoring?
If it helps to make automated decisions

Variables

Script

API

Infrastructure
Standardization
Configuration standardization

Automation is useless without a configuration standard or naming convention

Automation relies on regular expressions:

\^TRANSIT.* = all transit interfaces

.*PRIV\_PEER = all private peers

.*(PUB|PRIV)\_PEER = all peers

router.cisco.*\.pa = Cisco routers in Panamá
Software version standardization

```python
junos.version_info(major=(15, 1))
{
    'community': [{
        'name': {
            'data': 'EXAMPLE_COMM'
        },
        'members': [{
            'data': '65536:1'
        }]
    }]
}
```

```python
junos.version_info(major=(18, 4))
{
    'community': [{
        'name': 'EXAMPLE_COMM',
        'members': ['65536:1']
    }]
}
```
PyEZ warning
Script

dev = Device(host=router, user=username, password=password)
dev.open()

cli = Config(dev, mode='private')

command = 'set interface et-0/0/0 description "A nice description"'

try:
    cli.load(command, format='set')
except (ConfigLoadError, Exception) as err:
    print ("Unable to load configuration changes: {0}".format(err))
Output

Unable to load configuration changes:
ConfigLoadError(severity: error, bad_element: interface, message: error: syntax error)
The problem?

set interface != set interfaces
Corrected script

dev = Device(host=router, user=username, password=password)
dev.open()

cli = Config(dev, mode='private')

command = 'set interfaces et-0/0/0 description "A nice description"'

try:
    cli.load(command, format='set')
except (ConfigLoadError, Exception) as err:
    print("Unable to load configuration changes: {0}".format(err))