NTS

RELAX, WE CARE

AUTOMATION FOR CISCO NETOPS

26-28.02.2019 | Innsbruck, Andrea Dainese

ABOUT

NTS

ANDREA DAINESE - SENIOR SYSTEMS ENGINEER

- Network and Security Architect (15+ years' exp.)
- Security Evangelist (Blue Team)
- Automation Addicted/Developer (UNetLab)
- Cisco CCIE #38620/VMware VCP/Red Hat RHCE

andrea.dainese@gmail.com

www.linkedin.com/in/adainese

@adainese

NOTES

NOTES

- Topics fit in a two-weeks course, but every topic needs to be at least shortly described.
- Part of each network automation engineer is to find doc (a.k.a. googling and/or reverse engineering).
- Network automation engineers needs to write code and write prototypes (a.k.a. spaghetti-code).
- Developers can turn prototypes into robust softwares. That's not a task for network automation engineers.
- Developers cannot write network automation softwares because they lack skills on networking, on vendor products. But they write better code.

GOAL, OBJECTIVES AND PREREQUISITES

GOAL AND OBJS

NTS

Course Goal

Upon completion of this course the students will be able to discuss automation frameworks available for Cisco devices, choose the right one and automate single or group of orchestrated tasks using the right set of tools.

Course Objectives

- Be familiar with automation terminology.
- Be able to approach an automation project.
- Understand automation frameworks available for Cisco devices.
- Be able to write automation scripts.

Out of scope: learning Linux, Python, details of each framework.

PREREQUISITES

Prerequisites

- Able to install and customize Linux (Ubuntu)
- Vi/Vim
- Python 3 (basics)
- PIP
- RegEx

PREREQUISITES

NTS

Software

Ubuntu 16.04 minimal with (minimal configuration):

- 6GB of RAM
- 2vCPU
- 40GB of disk (single filesystem + swap)
- openssh-server and python-2.7 installed
- reachable via SSH
- installed within a corporate server/laptop* on VMware ESXi, Workstation, Player or Fusion only.
- VT-x/EPT extensions must be enabled

*: be sure the VM is attached to a NAT adapter and configure NAT properly. Do not use bridge/shared network!

PREREQUISITES

Vim

- http://vimgenius.com/

Python

- https://learnpythonthehardway.org/python3/
- https://pynet.twb-tech.com/

RegEx

- https://regexone.com/
- https://regex101.com/

AGENDA

AGENDA

NTS

Agenda #1

- Automation
- Orchestration
- DevOps
- CI/CD: an enabler for DevOps
- Network DevOps
- Python 2.7 or 3?
- GIT (fundamentals)
- Templating
- Frameworks
  - Zero Touch Provisioning
  - Screen Scraping

AGENDA

NTS

Agenda #2

- Frameworks
  - Nornir
  - Ansible
  - Puppet
  - SaltStack
  - NetConf & RESTConf
  - Native API:
    - NX-API and NX-API REST
    - ASA
    - ACI
    - CUCM

AGENDA

Agenda #2

- Frameworks
  - SDK
  - Embedded Python
  - Cisco Embedded Event Manager
- Conclusions

NTS

AUTOMATION

Automation, the new hype

What is automation?

- backup network device configurations once an hour
- backup network device configurations after a configuration commit
- execute some troubleshooting tasks
- validate a new site
- upgrade hundreds of remote DMVPN routers
- deploy a new customer (VRF, networks, routing, load balancers, firewall rules…)
- …
- eliminate repeatable manual tasks

WE CAN AUTOMATE EVERY TASK WE CAN DEFINE

NTS

DEFINING A TASK #1

Task: upgrade hundreds of remote DMVPN routers. For each site:

1. connect to the secondary router
2. verify the current IOS version
3. verify that traffic is flowing through primary router
4. upload the image
5. verify the uploaded image
6. update the config to boot with the new image
7. save the config
8. reload the router

DEFINING A TASK #1

9. wait 5 minutes to get the secondary router back
10. verify the secondary router
11. make the secondary router active
12. verify the traffic is now flowing through the secondary router
13. repeat this process with the primary router

NTS

SINGLE TASK AUTOMATION OR PROCESS AUTOMATION?

AUTOMATION

NTS

Single task automation

- one shot
- designed for a specific task/event (tailored design)
- not reusable
- personal use
- does not involve colleagues/company
- out of procedures

Example: upgrade hundreds of remote DMVPN routers
Pro: easy and fast to implement
Cons: do not scale, script/framework proliferations, limited automation because the big picture is out of scope

Process automation

- based on an internal approved framework
- users cannot do things in a different way
- very limited user interaction
- changes on automation scripts go through an authorization process
- approved by the company and defined as a company standard

Example: upgrade hundreds of remote DMVPN routers
Pro: less error prone, well defined and documented
Cons: need to involve other people/groups/processes, slow to be approved/implemented/standardized

ORCHESTRATION

Orchestration

- a set of automated tasks grouped together in a coordinated workflow
- need to understand the business process (real world is usually different than expected)
- redesign the business process itself and all related processes
  - simplify
  - standardize
  - abstract

NTS

Orchestration: example

Daily check for IOS upgrades, if one is found request approval to upgrade. If upgrade is confirmed, provides a date to each site manager and request for confirmation. If date is confirmed, alert the monitoring team and upgrade the remote device. If upgrade fails, alert the site manager, the monitoring team and the on duty guy to fix the issue/replace the device.

DEVOPS

DevOps #1

The problem:

- developers want to write software (oftentimes on their laptop, testing in prod)
- operations folks care about stability (they want to sleep at night)

The idea: DevOps embraces the Lean philosophy, applies the Agile methodologies, and brings together Development and Operations.

DEVOPS

NTS

DevOps #2

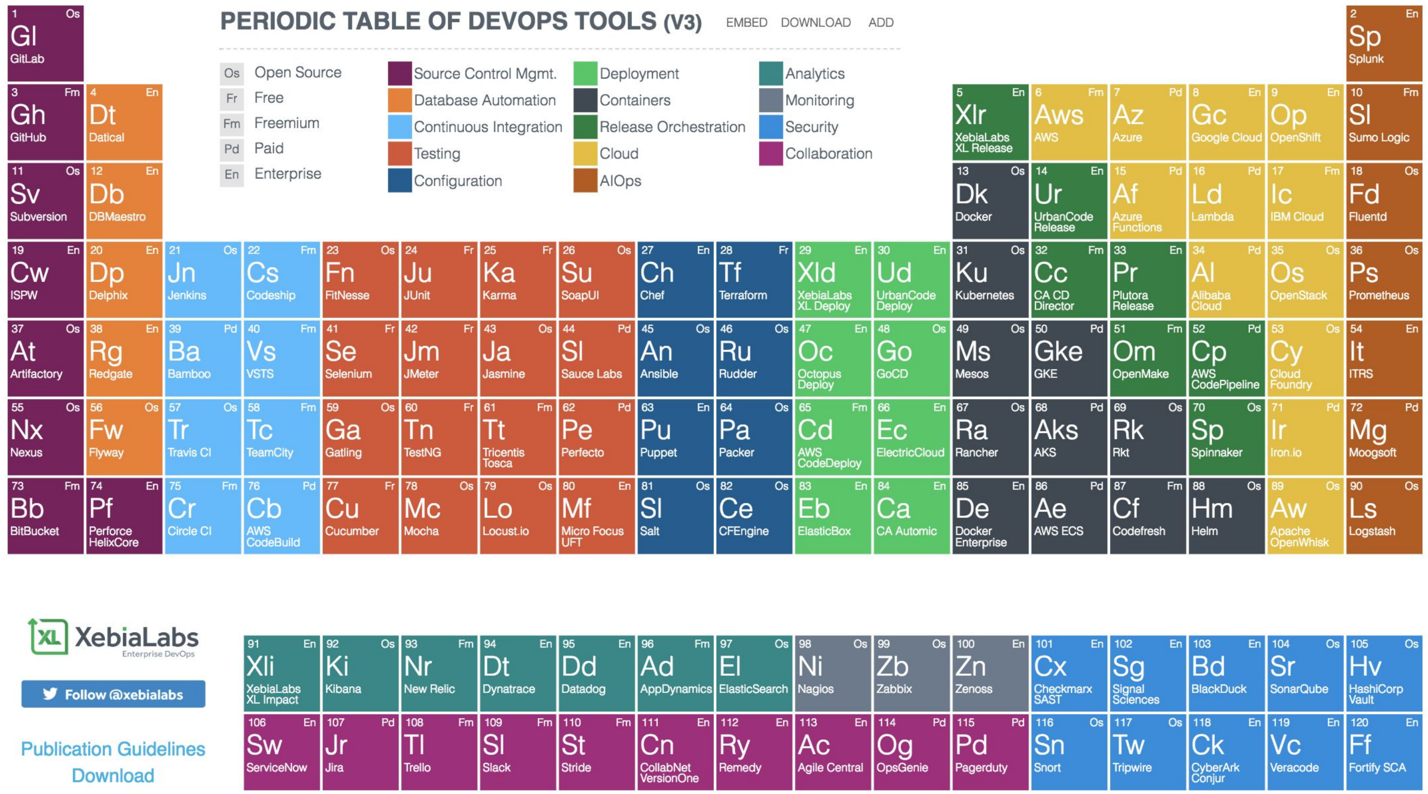
The result:
enhanced level of communication
deliver software, products, and services faster
based on CALMS (Culture, Automation, Lean, Measurement, Sharing)

DevOps #3

Culture: Trust, Supportive, Collaborative, No-blame / no victims, Common Goals
Automation: Puppet, Chef, Ansible
Lean: Identify wastes, Continuous learning, Focus on people, Optimize the whole
Measurement: outages/performance issues, **cost** of resources, release and deployment, **User Acceptance Testing**, Measure Everything
Sharing: code (git repositories) , ideas, and problems (ChatOps)

DEVOPS

NTS



PERIODIC TABLE OF DEVOPS TOOLS (V3)   EMBED   DOWNLOAD   ADD

Os Open Source	Source Control Mgmt.	Deployment	Analytics
Fr Free	Database Automation	Containers	Monitoring
Fm Freemium	Continuous Integration	Release Orchestration	Security
Pd Paid	Testing	Cloud	Collaboration
En Enterprise	Configuration	AIOps	

No.	Symbol	Name	Licence	Category
1	Gl	GitLab	Os	Source Control Mgmt.
2	Sp	Splunk	En	AIOps
3	Gh	GitHub	Fm	Source Control Mgmt.
4	Dt	Datical	En	Database Automation
5	Xlr	XebiaLabs XL Release	En	Release Orchestration
6	Aws	AWS	Fm	Cloud
7	Az	Azure	Pd	Cloud
8	Gc	Google Cloud	En	Cloud
9	Op	OpenShift	En	Cloud
10	Sl	Sumo Logic	Fm	AIOps
11	Sv	Subversion	Os	Source Control Mgmt.
12	Db	DBMaestro	En	Database Automation
13	Dk	Docker	Os	Containers
14	Ur	UrbanCode Release	En	Release Orchestration
15	Af	Azure Functions	Pd	Cloud
16	Ld	Lambda	Pd	Cloud
17	Ic	IBM Cloud	Fm	Cloud
18	Fd	Fluentd	Os	AIOps
19	Cw	ISPW	En	Source Control Mgmt.
20	Dp	Delphix	En	Database Automation
21	Jn	Jenkins	Os	Continuous Integration
22	Cs	Codeship	Fm	Continuous Integration
23	Fn	FitNesse	Os	Testing
24	Ju	JUnit	Fr	Testing
25	Ka	Karma	Fr	Testing
26	Su	SoapUI	Os	Testing
27	Ch	Chef	En	Configuration
28	Tf	Terraform	Fr	Configuration
29	Xld	XebiaLabs XL Deploy	En	Deployment
30	Ud	UrbanCode Deploy	En	Deployment
31	Ku	Kubernetes	Os	Containers
32	Cc	CA CD Director	Fm	Release Orchestration
33	Pr	Plutora Release	En	Release Orchestration
34	Al	Alibaba Cloud	Pd	Cloud
35	Os	OpenStack	Os	Cloud
36	Ps	Prometheus	Os	AIOps
37	At	Artifactory	Os	Source Control Mgmt.
38	Rg	Redgate	En	Database Automation
39	Ba	Bamboo	Pd	Continuous Integration
40	Vs	VSTS	Fm	Continuous Integration
41	Se	Selenium	Fr	Testing
42	Jm	JMeter	Fr	Testing
43	Ja	Jasmine	Os	Testing
44	Sl	Sauce Labs	Pd	Testing
45	An	Ansible	Os	Configuration
46	Ru	Rudder	Os	Configuration
47	Oc	Octopus Deploy	En	Deployment
48	Go	GoCD	Os	Deployment
49	Ms	Mesos	Os	Containers
50	Gke	GKE	Pd	Containers
51	Om	OpenMake	Fm	Release Orchestration
52	Cp	AWS CodePipeline	Pd	Release Orchestration
53	Cy	Cloud Foundry	Os	Cloud
54	It	ITRS	En	AIOps
55	Nx	Nexus	Os	Source Control Mgmt.
56	Fw	Flyway	Os	Database Automation
57	Tr	Travis CI	Os	Continuous Integration
58	Tc	TeamCity	Fm	Continuous Integration
59	Ga	Gatling	Os	Testing
60	Tn	TestNG	Fr	Testing
61	Tt	Tricentis Tosca	Fm	Testing
62	Pe	Perfecto	Pd	Testing
63	Pu	Puppet	En	Configuration
64	Pa	Packer	Os	Configuration
65	Cd	AWS CodeDeploy	Fm	Deployment
66	Ec	ElectricCloud	En	Deployment
67	Ra	Rancher	Os	Containers
68	Aks	AKS	Pd	Containers
69	Rk	Rkt	Os	Containers
70	Sp	Spinnaker	Os	Release Orchestration
71	Ir	Iron.io	Pd	Cloud
72	Mg	Moogsoft	Pd	AIOps
73	Bb	BitBucket	Fm	Source Control Mgmt.
74	Pf	Perforce HelixCore	En	Source Control Mgmt.
75	Cr	Circle CI	Fm	Continuous Integration
76	Cb	AWS CodeBuild	Pd	Continuous Integration
77	Cu	Cucumber	Fr	Testing
78	Mc	Mocha	Os	Testing
79	Lo	Locust.io	Os	Testing
80	Mf	Micro Focus UFT	En	Testing
81	Sl	Salt	Os	Configuration
82	Ce	CFEngine	Os	Configuration
83	Eb	ElasticBox	En	Deployment
84	Ca	CA Automic	En	Deployment
85	De	Docker Enterprise	En	Containers
86	Ae	AWS ECS	Pd	Containers
87	Cf	Codefresh	Fm	Containers
88	Hm	Helm	Os	Containers
89	Aw	Apache OpenWhisk	Os	Cloud
90	Ls	Logstash	Os	AIOps
91	Xli	XebiaLabs XL Impact	En	Analytics
92	Ki	Kibana	Os	Analytics
93	Nr	New Relic	Fm	Analytics
94	Dt	Dynatrace	En	Analytics
95	Dd	Datadog	En	Analytics
96	Ad	AppDynamics	Fm	Analytics
97	El	ElasticSearch	Os	Analytics
98	Ni	Nagios	Os	Monitoring
99	Zb	Zabbix	Os	Monitoring
100	Zn	Zenoss	En	Monitoring
101	Cx	Checkmarx SAST	En	Security
102	Sg	Signal Sciences	En	Security
103	Bd	BlackDuck	En	Security
104	Sr	SonarQube	Os	Security
105	Hv	HashiCorp Vault	Os	Security
106	Sw	ServiceNow	En	Collaboration
107	Jr	Jira	Pd	Collaboration
108	Tl	Trello	Fm	Collaboration
109	Sl	Slack	Fm	Collaboration
110	St	Stride	Fm	Collaboration
111	Cn	CollabNet VersionOne	En	Collaboration
112	Ry	Remedy	En	Collaboration
113	Ac	Agile Central	En	Collaboration
114	Og	OpsGenie	Pd	Collaboration
115	Pd	Pagerduty	Pd	Collaboration
116	Sn	Snort	Os	Security
117	Tw	Tripwire	Os	Security
118	Ck	CyberArk Conjur	En	Security
119	Vc	Veracode	En	Security
120	Ff	Fortify SCA	En	Security

XebiaLabs Enterprise DevOps

Follow @xebialabs

Publication Guidelines
Download

CI/CD: AN ENABLER FOR DEVOPS

CI/CD: an enabler for DevOps #1

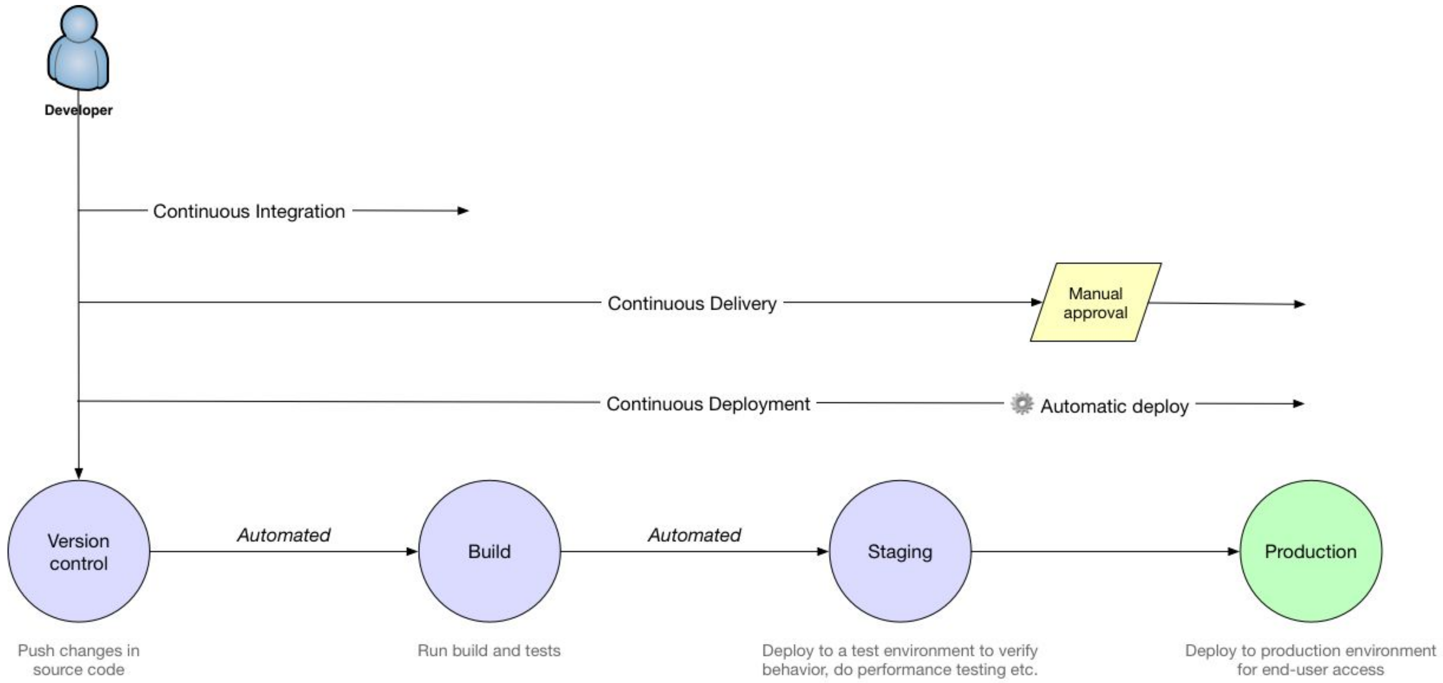
Continuous Integration: is about everyone merging code changes to a central repository multiple times a day. Each merge triggers an automated build and testing sequence for the given project.

Continuous Delivery: is a practice of automating the entire software release process. The idea is to do CI, plus automatically prepare and track a release to production. The desired outcome is that anyone with sufficient privileges to deploy a new release can do so at any time in one or a few clicks.

CI/CD: an enabler for DevOps #2

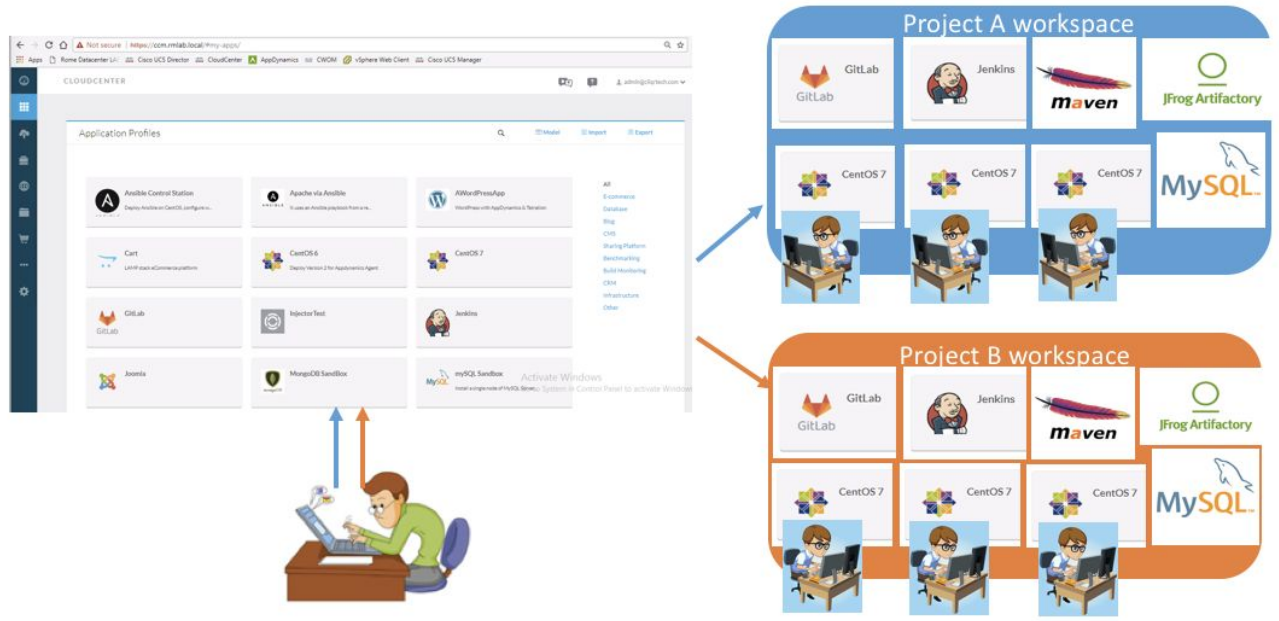
Continuous Deployment: is a step up, in which every change in the source code is deployed to production automatically, without explicit approval from a developer. Continuous deployment requires a highly developed culture of monitoring, being on call, and having the capacity to recover quickly.

CI/CD


Developer
Continuous Integration
Continuous Delivery
Manual approval
Continuous Deployment
Automatic deploy
Version control
Automated
Build
Automated
Staging
Production
Push changes in source code
Run build and tests
Deploy to a test environment to verify behavior, do performance testing etc.
Deploy to production environment for end-user access

NTS

CIaaS/CDaaS



https://blogs.cisco.com/cloud/have-you-ever-considered-ci-cd-as-a-service

NTS

NETDEVOPS

Network DevOps

- Treat the network like a software
- Implement DevOps, and especially CI/CD

An idea:

- configuration are built from a specific template and from a Python script + Jinja2
- after a commit, configurations are built then they are pushed to a test environment
- automatic tests are executed to verify if everything is working as expected
- configurations are now ready to be pushed to the production, with or without a manual approval.

NETDEVOPS

NTS

Network Automation: HowTo start

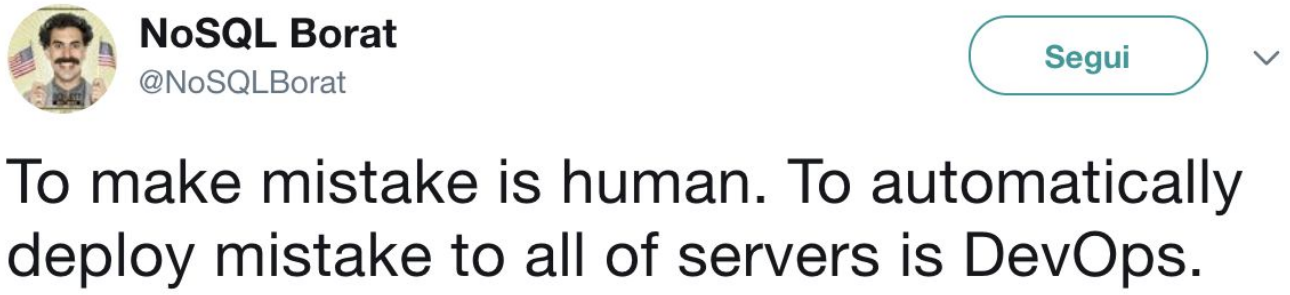
Single task automation:

- Use git to store the scripts and document them
- Choose a framework (see next part for suggestions)
- Choose how to push changes:
  - push a configuration delta
  - push the entire configuration
- Test your scripts on every expected and unexpected scenario
- Be sure you can roll-back at any time

NETDEVOPS

NTS

Network Automation: HowTo fail


NoSQL Borat
@NoSQLBorat

To make mistake is human. To automatically deploy mistake to all of servers is DevOps.

NETDEVOPS

NTS

Network Automation: HowTo start

Process automation:

- Analyze how the network operators and employees in general works (methodology, procedures, standards…).
- Simplify the infrastructure (cleaning up the existing mess).
- Standardize the infrastructure (no more tailored approach).
- Abstract creating templates and data structures.
- Automate the process.

NETDEVOPS

NTS

Network Automation: HowTo learn

- https://www.ipspace.net/Building_Network_Automation_Solutions
  - Transparent and vendor-neutral course, based on more than 25 years of design and deployment experiences
  - Focus on real-world problems and optimal solutions, not technologies and products promoted by individual vendors
- https://pynet.twb-tech.com/class-pyauto.html
  - Learn network automation tools by practicing

DEMO

Network CI/CD Demo

- Open http://<EVE-NG IP>:9080/root/net-ci-cdnet-ci-cd/tree/master
- Edit a file
- Check under CI/CD -> Pipelines what is going on

PYTHON

Python 2.7 or 3?

- Python seems the best choice to interact with network devices
- Many devices already have an embedded Python interpreter (NX-OS is using Python 2.7)
- "Being the last of the 2.x series, 2.7 will have an extended period of maintenance. Specifically, 2.7 will receive bugfix support until January 1, 2020. All 2.7 development work will cease in 2020."

GIT (FUNDAMENTALS)

GIT

“Git is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files. Git was created by Linus Torvalds in 2005 for development of the Linux kernel”

GIT

NTS

GIT: interesting links

- https://gerardnico.com/code/version/git/start
- https://git-scm.com/book/en/v2
- http://try.github.io/
- https://guides.github.com/activities/hello-world/
- https://tutorialzine.com/2016/06/learn-git-in-30-minutes
- https://my.ipspace.net/bin/list?id=NetAutSol

GIT

NTS

GIT: learning by examples

Initialize a repository (current directory):

```
$ git init
```

Add a file to be included in the next commit:

```
$ git add file1
```

Remove a file from git (also from the repository):

```
$ git rm -f file2
```

Commit a set of changes:

```
$ git commit -m "Fixed bug 4532"
```

See last commit:

```
$ git log --oneline
```

GIT

NTS

GIT: learning by examples

Check the status of the repository:

```
$ git status
```

See differences between now and last commit:

```
$ git diff
```

Revert changes of a previous commit:

```
$ git revert <first 7 char of the commit ID>
```

GIT: learning by examples

Global configuration:

```
$ git config --global user.name "Andrea
Dainese"
$ git config --global user.email
andrea.dainese@gmail.com
$ git config --global push.default simple
```

NTS

GIT: lab #1

- Create an empty GIT repository
- Configure global parameters
- Create a file and add and commit
- Delete the file and commit
- Recover the file

GIT: lab #1 (solution)

```
$ git config --global user.name "Andrea
Dainese"
$ git config --global user.email
andrea.dainese@gmail.com
$ git config --global push.default simple

$ git init git_lab_1
$ cd git_lab_1
$ ls /etc > file
$ git add file
$ git commit -m "Added file"
```

NTS

GIT: lab #1 (solution)

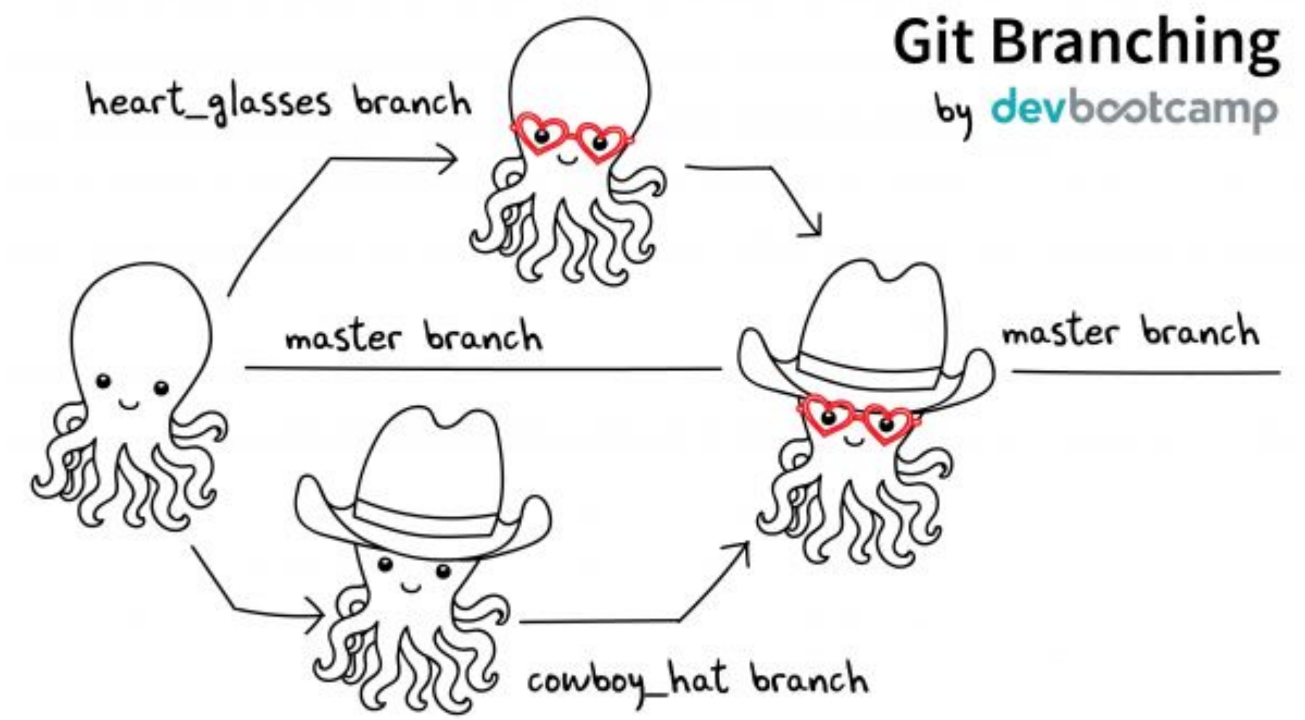
```
$ git rm -f file
$ git commit -m "Deleted file"
$ git log --oneline
c6b8116 Deleted file
646269c Added file

$ git revert c6b8116
```

or

```
$ git checkout 646269c file
```

GIT: branching



GIT

NTS

GIT

NTS

GIT: learning by examples

Create and change the current branch:

```
$ git checkout -b bugfixing_7340
```

Show branches:

```
$ git branch -a # or show-branch
```

Merge bugfixing_7340 into master:

```
$ git checkout master
$ git merge --commit bugfixing_7340
```

GIT

NTS

GIT: learning by examples

Cloning a remote repository:

```
$ git clone
https://github.com/dainok/course_netdevops
```

Update local branch with remote committed changes:

```
$ git pull
```

Update remote branch with local committed changes:

```
$ git push # or git push origin master
```

Delete a remote branch:

```
$ git push origin --delete wrong_branch
```

GIT

NTS

GIT: learning by examples

Get remote URLs and change protocol to SSH:

```
$ git remote -v
$ git remote set-url --push origin
git@github.com:dainok/unetlab
$ git remote set-url origin
git@github.com:dainok/course_netdevops
```

Special file: `.gitignore`

GIT: how to contribute to a GitHub project #1

1. Fork the project on GitHub (via web browser)
2. Clone the remote repository:

```
$ git clone
https://github.com/dainok/course_netdevops
```

3. Change the remote upstream:

```
$ git remote add upstream
https://github.com/<user>/course_netdevops
```

4. Create a new branch to store all changes:

```
$ git checkout -b NEW_FEATURE
```

GIT: how to contribute to a GitHub project #2

5. Make changes and test them (the following is specific for NAPALM):

```
$ TOXENV=py35 tox
test/ios/test_getters.py::TestGetter::test
_method_signatures
```

6. Add files, commit and push to the forked repository:

```
$ git add -A
$ git commit -m "Implemented new feature"
$ git push
```

NTS

GIT: how to contribute to a GitHub project #3

7. Ask to commit to the original repository (via web browser): go to the original repository -> Compare & pull request -> select original base/branch and custom fork/branch -> Create pull request

NTS

GIT: lab #2

- Create an account on GitHub
- Fork https://github.com/dainok/course_netdevops
- Clone https://github.com/<user>/course_netdevops
- Set the remote upstream using the original repository
- Create a new branch
- Modify or add some files
- Add changes and commit

GIT: lab #2 (solution)

Fork the repository via web

```
$ git clone
https://github.com/<user>/course_netdevops
$ git remote add upstream
https://github.com/dainok/course_netdevops
$ git checkout -b NEW_FEATURE
$ git add -A
$ git commit -m "Implemented new feature"
$ git push
```

Create the pull request

TEMPLATING

Jinja2

Jinja2 is a full featured template engine for Python. It has full unicode support, an optional integrated sandboxed execution environment, widely used and BSD licensed.

Jinja2: interesting links

- http://jinja.pocoo.org/
- https://www.ifconfig.it/hugo/post/2014-04-11-qpc_jinja2/

Jinja2: lab #1

- Create a template for a switch
- Apply the template on both switched defined into /root/scripts/devices.csv

Jinja2: lab #1 (solution)

Review /root/scripts/jinja2_lab1.py and /root/scripts/jinja2_lab1.template

Jinja2: conclusions

Jinja2 is useful to make device configurations. Configurations can be then pushed to the new device via POAP/AutoInstall or to running devices (replace config) via Pexpect/NAPALM/Netmiko/Ansible/…

FRAMEWORKS

Frameworks

In the real world we have the following ways to automatically interact with devices:

- Zero Touch Provisioning (ZTP)
- Screen Scraping
- NETCONF and RESTCONF
- REST API
- Automation tools (Ansible, Salt Stack, Puppet…)
- Software Development Kit (SDK)
- Embedded interpreters (Python, Cisco EEM...)

ZERO TOUCH PROVISIONING

Zero Touch Provisioning

ZTP should allow to configure devices without human intervention.

Cisco has two different technologies:

- AutoInstall (`service config`) for IOS and IOS-XE
- POAP for NX-OS

NTS

ZTP: AutoInstall

During the boot IOS and IOS-XE try to get an IP address from DHCP, and then try to get a configuration via TFTP:

```
%Error opening tftp://192.0.2.1/router-confg
(No such file or directory)
```

This behaviour can be disabled with:

```
no service config
```

A custom TFTP server should provide the proper configuration to each device looking up the MAC address and/or the IP address.

ZTP: interesting links

- https://www.ipspace.net/kb/CiscoAutomation/030-ztp.html
- http://www.nettinkerer.org/poap-and-ansible-integration-part-1/
- https://networklore.com/ztp-tutorial/
- https://github.com/CiscoSE/Cisco-POAP
- https://www.slideshare.net/CiscoDevNet/automating-with-nxos-lets-get-started

ZTP

NTS

ZTP: lab #1 (AutoInstall)

- Open ZTP.unl lab on EVE-NG
- Review `/usr/local/sbin/custom_tftp.py`
- Start `/usr/local/sbin/custom_tftp.py`
- Start the node “IOS”
- Verify that node “IOS” is getting the configuration
- Try to customize the TFTP server

NTS

ZTP: POAP (PowerOn Auto Provisioning)

When a NX-OS switch boots, the POAP process tries to download a **Python 2** script via TFTP. The script is then executed using the embedded Python 2 interpreter available in Nexus devices. POAP script can be used to:

- automatically install the right NX-OS version
- configure the whole switch
- do things supported by the embedded libraries

Cisco provides an official (discontinued?) customizable script, but there are many alternatives also.
POAP can also be implementing via USB storage.

ZTP LAB #2

NTS

ZTP: lab #2 (POAP)

- (not working yet)
- Open ZTP.unl lab on EVE-NG
- Review /opt/customtftp/tftproot/poap.py
- Start /usr/local/sbin/custom_tftp.py
- Start the node "NXOS"
- Verify POAP process

NTS

ZTP: conclusions

ZTP is an important feature to bring an out-of-the-box device connected to the network without human interaction. ZTP is an awesome feature if correctly used.

Important question: switches should be upgraded during provisioning?
Pro: provisioning assure switch are upgraded to the approved image
Cons: you have to maintain two different processes and related tools: the ones for the ZTP upgrade and the ones for the all running switches.

SCREEN SCRAPING

NTS

Screen Scraping

Screen scraping is the process of translating the displayed output of a software into a data structure. More generally I include in this category all automation script that try to emulate humans while they are interacting with a device. This is the most "rude" but effective way to automate many devices.
We can divide devices into two categories:

- Devices that can be configured using a terminal (telnet/serial or SSH).
- Devices that can be configured via web browser.

Web Scraping

When a device needs to be configured via browser only, a web scraping tool is required:

- WebBot
- Mechanize
- Selenium

Screen Scraping: frameworks

When a device can be configured via a terminal session or via an interactive (textual) script, a screen scraping approach must be used:

- Pexpect
- NetMiko
- NAPALM

NTS

Screen Scraping: Pexpect

Allows to interact with an interactive terminal session sending commands after receiving the expected output.

- Sends command after receiving expected output
- Useful for serial or telnet connection if ZTP is not supported
- Session is usually a command like **`telnet`**, **`ssh`** or a generic (textual and interactive) executable.
- Pexpect is the last resort option (compared to other tools)

Screen Scraping: NetMiko

NetMiko, compared to Pexpect, simplifies commands for supported devices:

- many supported devices (almost all Cisco devices)
- many supported commands
- translate output to data structure (with TextFSM and ntc-templates)
- easy to implement new devices or new commands
- “open” community

Screen Scraping: NAPALM

NAPALM is a vendor neutral cross platform automation tool:

- standard:
  - same method for all supported devices
  - same structured output from all supported devices
- very few supported devices
- very few supported commands
- “closed” community

Screen Scraping: conclusions

If there is no better way to interact with a device:

- use a web scraping if the device supports web only configuration;
- else use NAPALM if all required devices and commands are supported;
- else use NetMiko if all devices and commands are supported, or if can be developed;
- else use Pexpect (last resort).

Screen Scraping: interesting links

- https://www.seleniumhq.org/
- https://pexpect.readthedocs.io/en/stable/
- https://github.com/ktbyers/netmiko
- https://pynet.twb-tech.com/blog/automation/netmiko-textfsm.html
- https://github.com/networktocode/ntc-templates/
- https://napalm-automation.net/

NTS

Screen Scraping: suggestion for labs

- Working with network Cisco devices (routers, switches, wireless…)? Play with NetMiko.
- Working with devices not supported by NetMiko? Play with Pexpect.
- Already working with Git in CI/CD? Play with NAPALM (or NetMiko) replacing configuration.

Web Scraping: lab #1

- Review Selenium documentation
- Write a script to search something on Google

Web Scraping: lab #1 (solution)

Review /root/scripts/webscraping_lab1.py

Pexpect: lab #1

- Open ScreenScraping.unl lab on EVE-NG
- Start the node “IOS”
- Write a script to configure the node using the serial interface (telnet on port 32781):
  - wait until the boot process is completed
  - configure:
    - local admin
    - SSH server
    - DHCP on Ethernet0/3

Pexpect: lab #1 (solution)

Review /root/scripts/pexpect_lab1.py

Pexpect: lab #2

- Connect to a call manager using SSH
- Grab the output from `show version active`

Pexpect: lab #2 (solution)

Review /root/scripts/pexpect_lab2.py

NetMiko: lab #1

- Open ScreenScraping.unl lab on EVE-NG
- Start the nodes “R1” and “R2”
- Configure them to accept SSH connections
- Write a script to print the output of `show cdp neighbors detail`

NetMiko: lab #2 (solution)

Review /root/scripts/netmiko_lab1.py

NTS

NetMiko: lab #2

- Open ScreenScraping.unl lab on EVE-NG
- Start the nodes “R1” and “R2”
- Configure them to accept SSH connections
- Write a script to get the output of `show cdp neighbors detail` using TextFSM
- Print only the management IP addresses

NetMiko: lab #2 (solution)

Review /root/scripts/netmiko_lab2.py

NetMiko: lab #3

- Review `/usr/src/ntc-templates/templates/`
- Write a template for an unsupported command
- Add tests in `/usr/src/ntc-templates/tests`
- Commit and create a Pull Request

NetMiko: lab #3 (solution)

- Command: `show ap summary`
- Template: `/usr/src/ntc-templates/templates/cisco_wlc_ssh_show_ap_summary.template`
- Output: `/usr/src/ntc-templates/tests/cisco_wlc_ssh/show_ap_summary/cisco_wlc_ssh_show_ap_summary.raw`
- Parsed output: `/usr/src/ntc-templates/tests/cisco_wlc_ssh/show_ap_summary/cisco_wlc_ssh_show_ap_summary.parsed`

NetMiko: lab #3 (solution)

- Add the template to the index file (sorted) `/usr/src/ntc-templates/templates/index`
- Test the change before the commit: `tox -vv`
- See the pull request by Konrad Defranceschi: https://github.com/networktocode/ntc-templates/pull/272

NAPALM: lab #1

- Open ScreenScraping.unl lab on EVE-NG
- Start the node “R3”
- Configure it to accept SSH and SCP connections
- Write a script to get some “facts”
- Compare the output with NetMiko

NAPALM: lab #1 (solution)

Review /root/scripts/napam_lab1.py

NAPALM: lab #2

- Open ScreenScraping.unl lab on EVE-NG
- Start the node “R3”
- Configure it to accept SSH and SCP connections
- Write a script to configure something (merge)

Hints:

```
optional_args = {'dest_file_system': 'unix:'}
```

NAPALM: lab #2 (solution)

Review /root/scripts/napam_lab2.py

NAPALM: lab #3

- Open ScreenScraping.unl lab on EVE-NG
- Start the node “R3”
- Configure it to accept SSH and SCP connections
- Write a script to
  - get the running configuration
  - modify something
  - push the configuration to the router (replace)

Hints:

```
optional_args = {'dest_file_system': 'unix:'}
```

NAPALM: lab #3 (solution)

Review /root/scripts/napam_lab3.py

Review also /usr/src/net-ci-cd/scripts/get_templates*

NORNIR

NORNIR

Nornir is an automation framework written in python to be used with python. Nornir will take care of dealing with the inventory where you have your host information and it will take care of dispatching the tasks to your devices.

In short: Nornir + NAPALM ≊ Ansible within Python

NORNIR

NTS

NORNIR: interesting links

- https://networklore.com/introducing-brigade/
- https://github.com/nornir-automation/nornir
- http://brigade.readthedocs.io/en/develop/index.html

NORNIR

NTS

NORNIR: lab #1

- Open ScreenScraping.unl lab on EVE-NG
- Start the nodes “R1”, “R2” and “R3”
- Configure them to accept SSH connections
- Write a script to get the output of `show cdp neighbors detail` using NetMiko and TextFSM
- Print all data in a single table

NORNIR: lab #1 (solution)

Review /root/scripts/nornir_lab1.py, /root/scripts/hosts.yaml and /root/scripts/groups.yaml

NORNIR: lab #2

- Open ScreenScraping.unl lab on EVE-NG
- Start the nodes “R1”, “R2” and “R3”
- Configure them to accept SSH and SCP connections
- Add a new loopback interface using NAPALM (replace)

NORNIR

NORNIR: lab #2 (solution)

Review /root/scripts/nornir_lab2.py, /root/scripts/hosts.yaml and /root/scripts/groups.yaml

ANSIBLE

Ansible

"Ansible is radically simple IT (~~agentless~~) automation platform that makes your applications and systems easier to deploy."
It depends on Python 2.7 and on some related modules.

Ansible Network Engine Module is an automation tool:

- agentless for network devices (SSH is used)
- multi-threaded
- multi-OS (Linux, network devices, MacOS, ...)

Ansible: interesting links

- https://gitlab.com/rrlabs/ansible-components
- https://docs.ansible.com/ansible/devel/user_guide/intro_getting_started.html
- https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html
- https://gdykeman.github.io/2018/06/26/ios-upgrades/

Ansible: lab #1

- Check if all nodes are alive using Ansible

Ansible: lab #1 (solution)

```
ansible all -i ansible-hosts -m ping
```

Notes: ping in Ansible is a successful Ansible connection (via SSH) to the remote host, testing also the remote Python 2.7 installation. That’s the reason why this task cannot be used for network devices.

Ansible: lab #2

- Open Ansible.unl lab on EVE-NG
- Start all nodes (nodes are preconfigured)
- Send a raw command to all routers using Ansible

Ansible: lab #2 (solution)

```
ansible all -i ansible-hosts -u admin -m raw
-c paramiko -a "show ip alias"
```

Ansible: lab #3

- Open Ansible.unl lab on EVE-NG
- Start all nodes (nodes are preconfigured)
- Configure switches:
  - create VLAN2 and VLAN3 on both switches
  - SW1:e0/1 and SW2:e0/1 as access ports
  - SW1:e0/0, SW1:e0/2 and SW2:e0/0 as trunk ports

(Continue on the next slide)

Ansible: lab #3

- Configure routers:
  - R1:
    - e0/0: 192.168.12.1/24
    - lo: 1.1.1.1/32
  - R2:
    - e0/0.1: 192.168.12.2/24
    - e0/0.2: 192.168.32.2/24
    - lo: 2.2.2.2/32
  - R3:
    - e0/0.3: 192.168.32.3/24
    - lo: 3.3.3.3/32

Ansible: lab #3 (solution)

Review /root/scripts/ansible_lab_3_1.yaml and /root/scripts/ansible_lab_3_2.yaml

NTS

Ansible: conclusions

Ansible is good to **execute tasks (step by step)** on many systems, for example:

- add or remove a VirtualHost on Apache
- add or remove a VLAN

But Ansible also is:

- CPU intensive
- Hard to debug
- missing lot of operations for network devices

PUPPET

Puppet

Automatically deliver and operate all of your software across its entire lifecycle — simply, securely and at scale — with Puppet's modern automation.

A module “`puppetlabs-cisco_ios`” (version 0.2.0), exists, but it’s not so easy to make it work. And would you use a beta version on your infrastructure?

Puppet: interesting links

- https://www.digitalocean.com/community/tutorials/how-to-install-puppet-4-on-ubuntu-16-04
- https://forge.puppet.com/puppetlabs/cisco_ios

Puppet: conclusions

Puppet is an automation tool:

- with agent
- work very well with servers
- should work with IOS-XE devices, but the module seems not install properly

Puppet is good to **enforce a state** of a server (files configured as expected).

SALTSTACK

SALTSTACK

SaltStack

Salt is a new approach to infrastructure management built on a dynamic communication bus. Salt can be used for data-driven orchestration, remote execution for any infrastructure, configuration management for any app stack, and much more.

SaltStack: interesting links

- https://docs.saltstack.com/en/latest/topics/tutorials/salt_bootstrap.html#salt-bootstrap
- https://ripe74.ripe.net/presentations/18-RIPE-74-Network-automation-at-scale-up-and-running-in-60-minutes.pdf

SaltStack: lab #1

- Start the salt daemons
- Review the keys
- Add the key for localhost
- Test with a ping

SaltStack: lab #1 (solutions)

Review the configuration in /etc/salt/master and in /etc/salt/pillar, then start daemons with:

```
# salt-master -d
# salt-minion -d
```

Review authorized keys with:

```
# salt-key -L
# salt-key -A
```

Test with:

```
# salt '*' test.ping
```

SALTSTACK

NTS

SaltStack: lab #2

- Start all proxies
- Add all keys
- Test with a ping
- Send any show command

SaltStack: lab #2 (solutions)

Run all proxies with:

```
# salt-proxy -d --proxyid sw1
# salt-proxy -d --proxyid sw2
# salt-proxy -d --proxyid r1
# salt-proxy -d --proxyid r2
# salt-proxy -d --proxyid r3
```

Add all keys: # salt-key -A -y

Test with:

```
# salt '[rs]*' test.ping
# salt '[rs]*' net.cli "show ip alias"
```

SaltStack: lab #3

- Configure ntp servers

SaltStack: lab #3

```
# salt '[rs]*' ntp.set_peers 37.247.53.178
# salt '[rs]*' ntp.servers
```

SaltStack: conclusions

SaltStack is an automation tool:

- with agent
- work very well with servers (something like Ansible+Puppet)
- can work with network devices bringing up a proxy session **for each** network device
- faster than Ansible/NAPALM/Netmiko because SSH sessions are always active
- seems used more to read than to write

NETCONF & RESTCONF

NetConf & RESTConf

- NETCONF provides a mechanisms to install, manipulate, and delete the configuration of network devices. It uses RPC + XML. Default port for NETCONF agents is 830 using SSH.
- RESTCONF is NETCONF via HTTP using XML or JSON.
- YANG is how data are structured (like SMIv2+MIBs in SNMP)
- IOS-XR and NX-OS implement NETCONF
- IOS-XE implements both NETCONF and RESTCONF

NetConf & RESTConf

- get: retrieves running configuration and device state information.
- get-config: retrieves all or part of a specified configuration.
- edit-config: loads all or part of a specified configuration.
- copy-config: creates or replaces an entire configuration.
- delete-config: deletes a configuration.

(NET | REST)CONF

NTS

NetConf & RESTConf

- GET: retrieves data and metadata.
- PATCH: provides a framework for resource patching mechanisms, i.e., the equivalent of the NETCONF operation=merge.
- PUT: creates or replaces the target resource.
- POST: creates a data resource or invoke an operation resource.
- DELETE: deletes the target resource.

NetConf & RESTConf: interesting links

- https://www.cisco.com/c/dam/global/cs_cz/assets/ciscoconnect/2014/assets/tech_sdn10_sp_netconf_yang_restconf_martinkramolis.pdf
- https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/prog/configuration/167/b_167_programmability_cg/restconf_programmable_interface.pdf
- https://github.com/ncclient/ncclient
- https://www.codecademy.com/articles/what-is-rest
- https://github.com/YangModels/yang/

NetConf: lab #1

- Open NetRestConf.unl lab on EVE-NG
- Start node “CSR”
- Configure the node and enable NetConf
- Get the running config via NetConf

Print readable XML with ”`| xmllint --format -`”)

NetConf: lab #1 (solution)

Review /root/scripts/netconf_lab1.py

NetConf: lab #2

- Open NetRestConf.unl lab on EVE-NG
- Start node “CSR”
- Configure the node and enable NetConf
- Get the the config for interface Gi4 using the standard call and the Cisco proprietary call
- Compare the two results

(NET | REST)CONF

NetConf: lab #2 (solution)

Review /root/scripts/netconf_lab2.py

NTS

NetConf: lab #3

- Open NetRestConf.unl lab on EVE-NG
- Start node “CSR”
- Configure the node and enable NetConf
- Add a new loopback interface

NetConf: lab #3 (solution)

Review /root/scripts/netconf_lab3.py

RESTConf: lab #1

- Open NetRestConf.unl lab on EVE-NG
- Start node "CSR"
- Configure the node and enable RESTCONF
- Get the configuration of Gi4 via RESTConf

RESTConf: lab #1 (solution)

Review /root/scripts/restconf_lab1.py

NetConf & RESTConf: conclusions

Both NETCONF and RESTCONF try to achieve the same result: get a standard way to operate with devices.
Both fail because:

- even if standard schema exists, each vendor implements a custom schema
- different models from same vendors use different schema
- schemas change during time (vendor update without retro-compatibility)
- documentation is incomplete if it exists

NATIVE API

NX-API (CLI)

NX-API improves the accessibility of these CLIs by making them available outside of the switch by using HTTP/HTTPS. You can use the NX-API as an extension to the existing Cisco Nexus CLI.

NX-API REST

NX-API REST supports sending and receiving objects in the API payload:

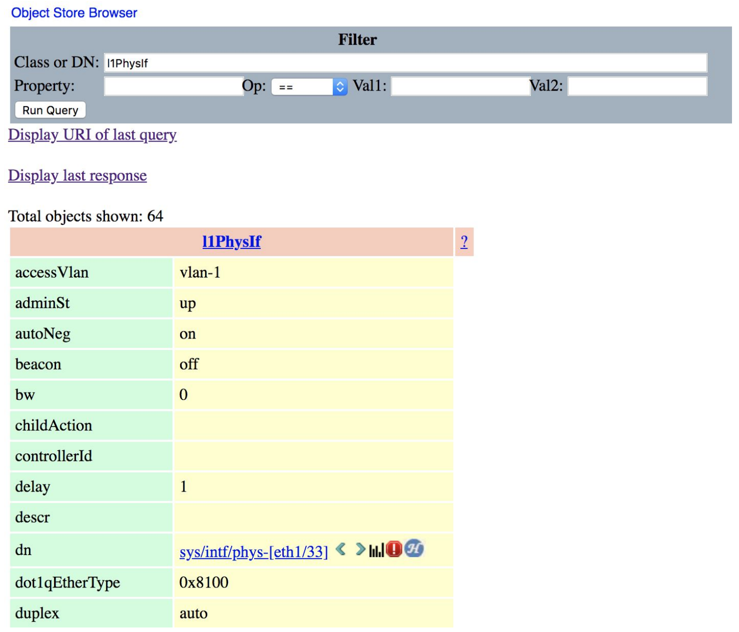
- NX-API (CLI): supports sending commands to the device while receiving structured data back (JSON, XML).
- NX-API REST: completely based on structured data meaning that JSON/XML payloads are sent to the device in the http request and received from the device in the response.

NATIVE API

NTS

NX-API REST: Visore

A sandbox (web page)called Visore help to play with REST API:



Object Store Browser

Filter

Class or DN: l1PhysIf

Property: Op: == Val1: Val2:

Run Query

Display URI of last query

Display last response

Total objects shown: 64

l1PhysIf	?
accessVlan	vlan-1
adminSt	up
autoNeg	on
beacon	off
bw	0
childAction	
controllerId	
delay	1
descr	
dn	sys/intf/phys-[eth1/33]
dot1qEtherType	0x8100
duplex	auto

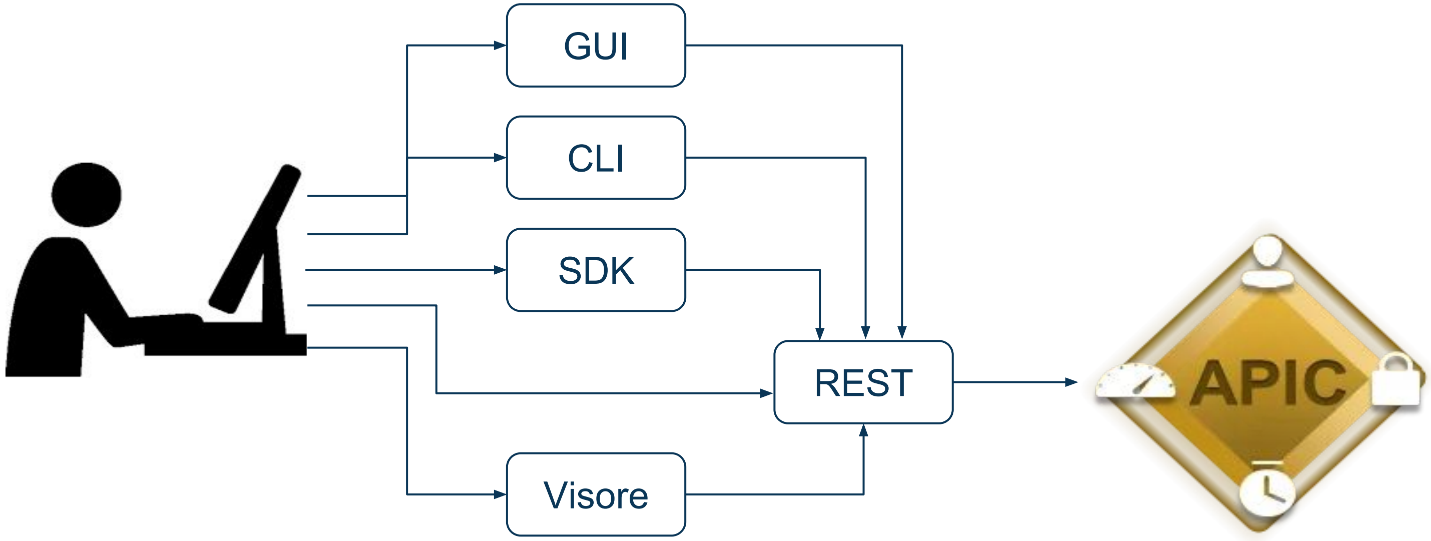
NATIVE API

ASA

- REST API
- Need to install an agent to ASA
- Documentation on the ASA itself

ACI

In Cisco ACI, REST API are the only way to operate.

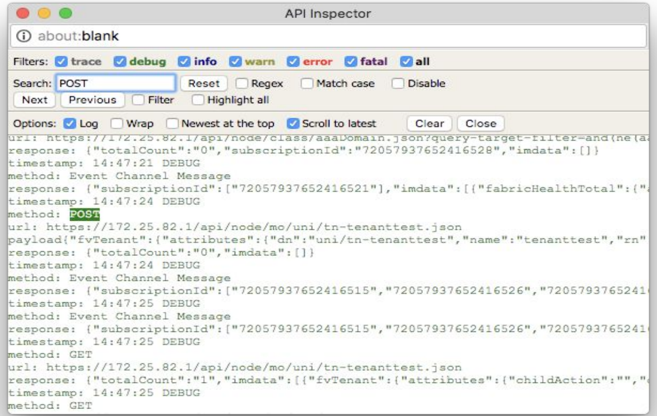


NTS

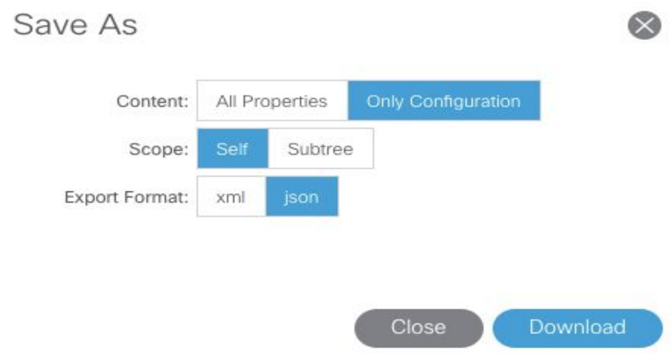
NATIVE API

ACI

Cisco ACI API are “self-documented”:


API Inspector
about:blank
Filters: trace debug info warn error fatal all
Search: POST Reset Regex Match case Disable
Next Previous Filter Highlight all
Options: Log Wrap Newest at the top Scroll to latest Clear Close

```
url: https://172.25.82.1/api/node/class/aaaDomain.json?query-target-filter=and(ne(a
response: {"totalCount":"0","subscriptionId":"72057937652416528","imdata":[]}
timestamp: 14:47:21 DEBUG
method: Event Channel Message
response: {"subscriptionId":["72057937652416521"],"imdata":[{"fabricHealthTotal":{"
timestamp: 14:47:24 DEBUG
method: POST
url: https://172.25.82.1/api/node/mo/uni/tn-tenanttest.json
payload{"fvTenant":{"attributes":{"dn":"uni/tn-tenanttest","name":"tenanttest","rn"
response: {"totalCount":"0","imdata":[]}
timestamp: 14:47:24 DEBUG
method: Event Channel Message
response: {"subscriptionId":["72057937652416515","72057937652416526","7205793765241
timestamp: 14:47:25 DEBUG
method: Event Channel Message
response: {"subscriptionId":["72057937652416515","72057937652416526","7205793765241
timestamp: 14:47:25 DEBUG
method: GET
url: https://172.25.82.1/api/node/mo/uni/tn-tenanttest.json
response: {"totalCount":"1","imdata":[{"fvTenant":{"attributes":{"childAction":"","
timestamp: 14:47:25 DEBUG
method: GET
```



Content: All Properties [✓ Only Configuration]
Scope: [✓ Self] Subtree
Export Format: xml [✓ json]

NTS

NATIVE API

CUCM

The Administrative XML Web Service (AXL) is a XML/SOAP based interface that provides a mechanism for inserting, retrieving, updating and removing data from the Unified Communication configuration database. Developers can use AXL and the provided WSDL to Create, Read, Update, and Delete objects such as gateways, users, devices, route-patterns and much more.

Native API: interesting links #1

- https://www.slideshare.net/CiscoDevNet/automating-with-nxos-lets-get-started
- https://github.com/datacenter/nexus9000/tree/master/nx-os/nxapi
- https://blog.ipspace.net/2017/04/lets-drop-some-random-commands-shall-we.html
- https://developer.cisco.com/docs/nx-os-n3k-n9k-api-ref/#!getting-started-with-the-cisco-nexus-3000-and-9000-series-nx-api-rest-sdk

Native API: interesting links #2

- https://www.cisco.com/c/en/us/td/docs/security/asa/api/qsg-asa-api.html
- https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/cucm/devguide/9_1_1/xmldev-911.html
- https://developer.cisco.com/site/axl/
- https://developer.cisco.com/docs/axl/#what-is-axl

NX-API (CLI): lab #1

- Open NativeAPI.unl lab on EVE-NG
- Start node “1NXOS”
- Configure the node and enable NX-API
- Get the interface status

NATIVE API

NTS

NX-API (CLI): lab #1 (solutions)

Review /root/scripts/nativeapi_lab1.py

NX-API REST: lab #1

Visore is a sandbox useful to play directly with REST API:

- Open NativeAPI.unl lab on EVE-NG
- Connect the node “NXOS” to a Cloud0 network
- Start node “NXOS”
- Connect via browser to https://<NX-OS IP>:8443/visore.html
- request a class (for example l1PhysIf)
- request a DN (for example sys/intf/phys-[eth1/1])

(mind that pnet0:7001 is a NAT for 192.0.2.30:8443)

NX-API REST: lab #2

- Open NativeAPI.unl lab on EVE-NG
- Start node “NXOS”
- Configure the node and enable NX-API REST
- Get the interface status

NX-API REST: lab #2 (solutions)

Review /root/scripts/nativeapi_lab2.py

ASA: lab #1

- Open NativeAPI.unl lab on EVE-NG
- Start node “ASAv”
- Upload the agent from /opt/customtftp/tftproot/asa-restapi-132200-lfbff-k8.SPA to the ASAv
- Install the agent
- Consult the doc page: https://192.0.2.31/doc/
- Get the physical interface status using REST API

(mind that pnet0:7000 is a NAT for 192.0.2.31:443)

ASA: lab #1 (solutions #1)

```
ssh 0.0.0.0 0 management
ssh scopy enable
http server enable
http 0.0.0.0 0 management
username admin password cisco privilege 15
aaa authentication ssh console LOCAL
interface Management0/0
 nameif management
 security-level 100
 ip address dhcp setroute
 no shutdown
```

ASA: lab #1 (solutions #2)

Upload from the EVE-NG VM to the ASAv:

```
$ scp
-oKexAlgorithms=+diffie-hellman-group1-sha1
/opt/customtftp/tftproot/asa-restapi-132200-l
fbff-k8.SPA
admin@192.0.2.31:disk0:asa-restapi-132200-lfb
ff-k8.SPA
```

ASA: lab #1 (solutions #3)

Install the agent:

```
rest-api image
disk0:asa-restapi-132200-lfbff-k8.SPA
rest-api agent
```

Get the status of physical interfaces:

```
$ curl -k -X GET -u "admin:cisco"
"https://192.0.2.31/api/interfaces/physical"
```

ACI: lab #1

- Login to an APIC controller
- Get some data via API

ACI: lab #1 (solutions)

Review /root/scripts/nativeapi_lab3.py

NATIVE API

NTS

CUCM: lab #1

- Review the AXL Schema reference: https://developer.cisco.com/docs/axl-schema-reference/
- Login to a CUCM
- Get configured users via SOAP API (AXL request)

NATIVE API

NTS

CUCM: lab #1 (solutions)

Review /root/scripts/nativeapi_lab4.py

CUCM: lab #2

- Review the CUCM data dictionary: https://developer.cisco.com/docs/axl/#!12-0-cucm-data-dictionary (also in /files/cucm-data-dictionary-12-0.zip of your EVE-NG VM)
- Login to a CUCM
- Get configured users via SOAP API (AXL SQL query)

CUCM: lab #2 (solutions)

Review /root/scripts/nativeapi_lab5.py

SDK

SDK

SDK

A software development kit (SDK or devkit) is typically a set of software development tools that allows the creation of applications for a certain software or hardware.

NTS

SDK

Cisco ACI SDK

- **ACI Toolkit:** exposes a subset of the ACI object model, it addresses **80%** of the common use cases , based on **Python 2.7**, unable to make it work.
- **Cobra:** covers all object types and attributes, based on **Python 2.7**. The documentation only covers the high-level classes.
- **Arya (APIC Rest Python Adapter):** given a JSON or XML, will generate Cobra code with the appropriate APIs and class types.

The real world is developing custom scripts with REST API calls directly (no SDK).

Cisco ACI SDK: interesting links

- https://cobra.readthedocs.io/en/latest/
- https://github.com/datacenter/arya
- https://APIC/cobra/_downloads/acicobra-3.1_1i-py2.7.egg
- https://APIC/cobra/_downloads/acimodel-1.1_1j-py2.7.egg

Cisco ACI SDK: lab #1

- Install the Cobra SDK to the EVE-NG VM

SDK

NTS

Cisco ACI SDK: lab #1 (solutions)

Download from the APIC:

- /cobra/_downloads/acicobra-3.1_1i-py2.7.egg
- /cobra/_downloads/acimodel-1.1_1j-py2.7.egg

Install then using `easy_install`:

- easy_install -Z acicobra-3.1_1i-py2.7.egg
- easy_install -Z acimodel-1.1_1j-py2.7.egg

Cisco ACI SDK: lab #2

- Using Cobra SDK, get all configured tenants

SDK

NTS

Cisco ACI SDK: lab #2 (solutions)

Review /root/scripts/sdk_lab1.py and /root/scripts/sdk_lab2.py

EMBEDDED PYTHON

Embedded Python

Cisco Nexus and Cisco Catalyst 9k have an embedded Python 2.7. Few libraries are included but generally cannot be installed additional modules.

On both platforms Embedded Python is used for ZTP (POAP) but can also be used to run custom scripts.

EMB. PYTHON

NTS

Embedded Python

- https://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus7000/sw/programmability/guide/b_Cisco_Nexus_7000_Series_NX-OS_Programmability_Guide/b_Cisco_Nexus_7000_Series_NX-OS_Programmability_Guide_chapter_0100.html
- https://github.com/datacenter/nexus7000
- https://learninglabs.cisco.com/modules/sdx-intro-nxos?utm_sq=fpzi4m7hxb
- https://github.com/datacenter/opennxos/tree/master/on-box
- https://tools.cisco.com/squish/2C58D

EMB. PYTHON

NTS

Embedded Python: lab #1

- Open NativeAPI.unl lab on EVE-NG
- Start node “NXOS”
- Display the status of the interfaces

Embedded Python: lab #1 (solution)

```
switch# python
>>> import cli
>>> help(cli)
>>> help(cli.clid)
>>> print(cli.clid('show interface brief'))
```

Embedded Python: lab #2

- Open NativeAPI.unl lab on EVE-NG
- Start node “NXOS”
- Configure an interface

Embedded Python: lab #3 (solution)

```
>>> import cisco
>>> help(cisco)
>>> help(cisco.interface)
>>> interface =
cisco.interface.Interface('Ethernet1/1')
>>> interface.set_state('up')
>>> interface.set_switchport()
>>> interface.set_mode('access')
```

NTS

CISCO EEM

Cisco EEM

Cisco Embedded Event Manager (EEM) is a powerful and flexible subsystem that provides real-time network event detection and onboard automation. It gives you the ability to adapt the behavior of your network devices to align with your business needs.

Cisco EEM: lab #1

- Create an EEM script to automatically add RSA keys for SSH.

Cisco EEM: lab #1 (solution)

```
event manager applet crypto_key
 event none sync yes
 action 1 cli command "enable"
 action 2 cli command "config terminal"
 action 3 cli command "crypto key generate rsa modulus
1024"
 action 4 cli command "end"
 action 5 cli command "write memory"
kron policy-list crypto_key
 cli event manager run crypto_key
kron occurrence crypto_key in 1 oneshot
 policy-list crypto_key
```

CONCLUSIONS

CONCLUSIONS

- POAP
- AutoInstall
- Pexpect
- Netmiko (+ Nornir)
- NAPALM (+ Nornir)
- Ansible (ios_config)
- ~~SaltStack~~ (my opinion because the lack of modules)
- ~~Puppet~~
- ~~NetConf and RestConf~~ (my opinion because the lack of documentation)
- Native API (~~NX-API CLI~~, NX-API REST, ASA, ACI, CUCM)
- ~~SDK (ACI)~~
- ~~Embedded Python (NX-OS)~~
- Cisco Embedded Event Manager (EEM)

NTS