auto_remediation_of_non_compliant_junos_c Documentation

Release 1

Khelil Sator

Apr 19, 2019

Contents

Contents:

CHAPTER 1

Repository description

This repository is about **auto remediation** of non compliant Junos configuration

This repository uses Junos devices, SaltStack, JSNAPy and Request Tracker.

At each Junos commit, SaltStack is notified with a syslog message, and runs a JSNAPy test to audit the new Junos configuration.If the Junos configuration is not compliant with the JSNAPy rules, SaltStack updates the ticketing system (Request Tracker) with this issue, and fixes the issue and reports this activity on the ticketing system.The ticket id is indicated in the Junos commit message.

Example:

configure telnet on a Junos device

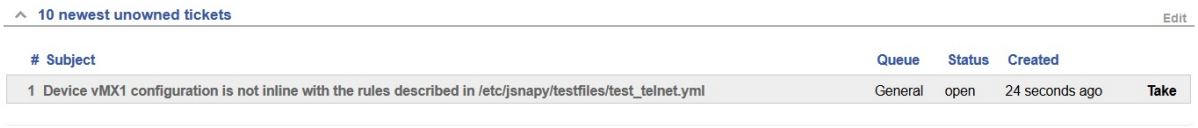
```
$ ssh jcluser@100.123.1.0
Password:
Last login: Sat Mar 30 13:44:53 2019 from 100.123.35.0
--- JUNOS 17.4R1-S2.2 Kernel 64-bit  JNPR-11.0-20180127.fdc8dfc_buil
jcluser@vMX1>

jcluser@vMX1> edit
Entering configuration mode

[edit]
jcluser@vMX1# set system services telnet

[edit]
jcluser@vMX1# commit and-quit
commit complete
Exiting configuration mode
```

SaltStack received the syslog commit message, and runs a JSNAPy test to audit the new Junos configuration.Telnet is not allowed. The new Junos configuration is not compliant with the JSNAPy rules. The JSNAPy test fails.SaltStack updates the ticketing system (Request Tracker) to report this issue.



new ticket

ticket update

Then, SaltStack fixes this issue, and reports this new activity on the ticketing system.

The ticket id is indicated in the Junos commit message.

```
jcluser@vMX1> show system commit
0   2019-03-30 14:30:31 UTC by jcluser via netconf
    configured with SaltStack using the delete_telnet.xml file to remove telnet␣
↪configuration due to ticket 1
1   2019-03-30 14:30:19 UTC by jcluser via cli
```

```
jcluser@vMX1> show configuration | compare rollback 1
[edit system services]
-    telnet;
```

So, in few seconds only, the new issue has been automatically detected, reported, and fixed

CHAPTER 2

Building blocks

2.1 SaltStack

Salt is a remote execution tool and configuration management system:

- remote execution: run commands on various machines in parallel with a flexible targeting system (salt execution modules)
- configuration management: establishes a client-server model to bring infrastructure components in line with a given policy (salt state modules in sls files)

SaltStack supports event driven infrastructure

SaltStack competes primarily with Puppet, Chef, StackStorm, and Ansible.

2.2 Request Tracker

Request Tracker (RT) is an open source issues tracking system.

2.3 JSNAPy

JSNAPy is an open source tool to automate verifications on Junos devices:

- operational state verifications
- configuration verifications

JSNAPy is supported in three modes:

- a command line tool
- a Python module
- An ansible module hosted on the Ansible Galaxy website

This repository uses the JSNAPy python module.

CHAPTER 3

Requirements to use this repository

You will need one Ubuntu VM and Junos devices.

3.1 install these dependencies on the Ubuntu VM

```
sudo apt-get update
sudo apt-get install python-pip -y
pip install pyyaml jinja2
```

```
pip list
```

3.2 install docker on the Ubuntu VM

Check if Docker is already installed

```
$ docker --version
```

If it was not already installed, install it. Here's how to install in on Ubuntu 16.04:

```
$ sudo apt-get update
```

```
$ sudo apt-get install \
    apt-transport-https \
    ca-certificates \
    curl \
    software-properties-common
```

```
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
```

```
$ sudo add-apt-repository \
   "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
   $(lsb_release -cs) \
   stable"
```

```
$ sudo apt-get update
```

```
$ sudo apt-get install docker-ce
```

```
$ sudo docker run hello-world
```

```
$ sudo groupadd docker
```

```
$ sudo usermod -aG docker $USER
```

Exit the ssh session to your ubuntu and open an new ssh session to your ubuntu and run these commands to verify you installed Docker properly:

```
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://hub.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/
```

```
$ docker --version
Docker version 18.03.1-ce, build 9ee9f40
```

3.3 install docker-compose on the Ubuntu VM

```
sudo curl -L https://github.com/docker/compose/releases/download/1.22.0/docker-
↪compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose
```

```
sudo chmod +x /usr/local/bin/docker-compose
```

```
docker-compose --version
```

CHAPTER 4

How to use this repository

4.1 clone the repository

```
git clone https://github.com/ksator/auto_remediation_of_non_compliant_configuration.
↪git
cd auto_remediation_of_non_compliant_configuration
```

4.2 Update the variables.yml file

The variables.yml file defines devices ip address, credentials, ...

```
vi variables.yml
```

4.3 Use the Makefile to start the setup

There is a Makefile at the root of the repository

Run this command to generate SaltStack files, instantiate a docker network and docker containers (Request Tracker, SaltStack master, SaltStack minion), start SaltStack services (master, minion) and daemons (one proxy for each Junos device)

```
make up
```

Run these commands to verify

```
docker-compose ps
docker ps
docker images
```

Run this commands to validate master <-> proxies <-> junos devices communication

```
docker exec -it master salt -G 'os_family:junos' junos.cli "show version"
```

4.4 Configure the Junos devices

Run this command to configure the Junos devices with the host-name and the syslog server indicated in the variables.yml file

```
docker exec -it master salt -G 'os_family:junos' state.apply syslog
```

Run these command to verify

```
docker exec -it master salt -G 'os_family:junos' junos.cli "show configuration system␣
↪host-name"
docker exec -it master salt -G 'os_family:junos' junos.cli "show configuration system␣
↪syslog"
```

4.5 Run the demo

configure telnet on a Junos device

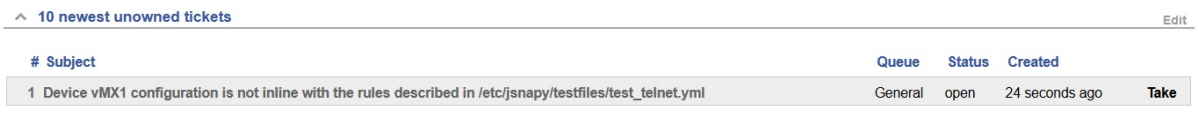
```
$ ssh jcluser@100.123.1.0
Password:
Last login: Sat Mar 30 13:44:53 2019 from 100.123.35.0
--- JUNOS 17.4R1-S2.2 Kernel 64-bit  JNPR-11.0-20180127.fdc8dfc_buil
jcluser@vMX1>

jcluser@vMX1> edit
Entering configuration mode

[edit]
jcluser@vMX1# set system services telnet

[edit]
jcluser@vMX1# commit and-quit
commit complete
Exiting configuration mode
```

SaltStack received the syslog commit message, and runs a JSNAPy test to audit the new Junos configuration.Telnet is not allowed. The new Junos configuration is not compliant with the JSNAPy rules. The JSNAPy test fails.SaltStack updates the ticketing system (Request Tracker) to report this issue.



new ticket

ticket update

Then, SaltStack fixes this issue, and reports this new activity on the ticketing system.

The ticket id is indicated in the Junos commit message.

```
jcluser@vMX1> show system commit
0   2019-03-30 14:30:31 UTC by jcluser via netconf
    configured with SaltStack using the delete_telnet.xml file to remove telnet␣
↪configuration due to ticket 1
1   2019-03-30 14:30:19 UTC by jcluser via cli
```

```
jcluser@vMX1> show configuration | compare rollback 1
[edit system services]
-    telnet;
```

So, in few seconds only, the new issue has been automatically detected, reported, and fixed

4.6 Use the Makefile to start a shell session in a container

To start a shell session in a container, run one of these commands:

```
make master-cli
```

```
make minion-cli
```

4.7 Use the Makefile to stop the setup

Run this command to stop docker containers, remove docker containers, remove docker network

```
make down
```

Verify

```
docker-compose ps
docker ps
docker ps -a
docker images
```

CHAPTER 5

Troubleshooting guide

5.1 Docker

Run this command to list Docker images

```
$ docker image
```

Run this command to list running containers

```
$ docker ps
```

Run this command to list all containers

```
$ docker ps -a
```

Run this command to list containers

```
$ docker-compose ps
```

Run this command to list networks

```
$ docker network ls
```

5.2 SaltStack

Run this command to check the salt-master service status

```
docker exec -it master service salt-master status
```

Run this command to check the salt-minion service status

```
docker exec -it minion1 service salt-minion status
```

Run this command to list the keys accepted by the master

```
docker exec -it master salt-key -L
```

Run this command to validate master configuration

```
docker exec -it master more /etc/salt/master
```

Run this command to check the other salt files on the master (pillar, runner, . . .)

```
docker exec -it master ls /srv/
```

Run this command to validate minion configuration

```
docker exec -it minion1 more /etc/salt/minion
```

Run this command to validate proxy configuration

```
docker exec -it minion1 more /etc/salt/proxy
```

Run these commands to validate master <-> minion communication

```
docker exec -it master salt minion1 test.ping
docker exec -it master salt "minion1" cmd.run "more /etc/salt/minion"
```

Run these commands to validate master <-> proxies communication

```
docker exec -it master salt -G 'os_family:junos' test.ping
```

Run these commands to validate master <-> proxies <-> junos devices communication

```
docker exec -it master salt -G 'os_family:junos' junos.cli "show version"
```

to watch the event bus, start a shell session on the master and run this command:

```
docker exec -it master bash
salt-run state.event pretty=True
```

5.3 JSNAPy

Run these commands

```
docker exec -it master jsnapy --version
docker exec -it master more /etc/jsnapy/testfiles/test_telnet.yml
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search