

# Starting with Ansible on Linux and Junos (and Cisco IOS)

## 1. Ansible setup on Debian 9

Contrary to what is claimed on the net about the environment setup of ansible, I did not find the startup configuration easy to setup as most enthusiasts claim, hence I thought I'd write a small article about this.

This is the setup:

Ansible runs on Debian Linux 9 → use 'apt-get' for this:

```
$ sudo apt-get install -i software-properties-common
$ sudo apt-add-repository ppa:ansible/ansible
$ sudo apt-get install -y ansible
```

Changes are that this will not install the latest release of ansible, so I used the following to upgrade ansible:

```
$ pip install ansible --upgrade
```

The versions of ansible and Python used within this document are:

```
$ ansible --version
ansible 2.9.5
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/home/john/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /home/john/.local/lib/python2.7/site-
packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.13 (default, Sep 26 2018, 18:42:22) [GCC 6.3.0
20170516]
```

-Install the Python / Ansible software and Junos modules from the galaxy server:

```
$ pip install ncclient
$ pip install junos-eznc
$ ansible-galaxy install Juniper.junos
$ pip install junos-netconf
```

To list the installed junos modules:

```
$ ansible-doc --list | grep -i juno
```

The version matching of ansible and Python and the modules etc. can really be a pain.. be prepared for this.

## 2. Junos SRX setup

We are using ansible for a SXR300 in this document. Public key authentication within SSH has been setup to login to the firewall without having to specify a password.

To allow ansible to work on the SRX, allow netconf over ssh:

```
set system services netconf ssh
```

This will open TCP.830 to enter the SRX with SSH(!). On this SSH connection you can now have a NETCONF dialogue.

If you would like to see a NETCONF conversation, then have a look at this:

## Starting with Ansible on Linux and Junos (and Cisco IOS)

[https://subscription.packtpub.com/book/networking\\_and\\_servers/9781788290999/1/ch01lv11sec8/junos-netconf-over-ssh-setup](https://subscription.packtpub.com/book/networking_and_servers/9781788290999/1/ch01lv11sec8/junos-netconf-over-ssh-setup)

Make sure you allow the NETCONF service on your zone:

```
set security zones security-zone dmz host-inbound-traffic system-services netconf
```

**First perform a check to verify that NETCONF works across TCP.830 with SSH:**

```
$ ssh john@srx300 -p 830 -s netconf
<!-- No zombies were killed during the creation of this user interface -->
<!-- user john, class j-local-super-user -->
<hello xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <capabilities>
    <capability>urn:ietf:params:netconf:base:1.0</capability>

<capability>urn:ietf:params:netconf:capability:candidate:1.0</capability>
  <capability>urn:ietf:params:netconf:capability:confirmed-commit:1.0</capability>

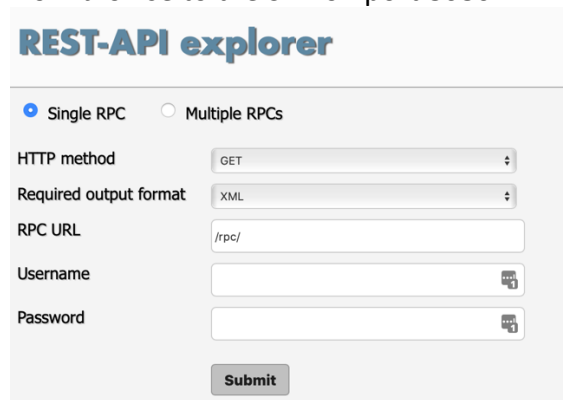
<capability>urn:ietf:params:netconf:capability:validate:1.0</capability>
```

If the command above fails, check with netcat or telnet whether the port has actually been opened on Junos. If not, then it might be.. that Junos is willing to play along but Linux' iptables @#\$%^& is not.. ☺.

Optional: You can browse the REST api explorer on the SRX as well once you enable it (not needed for ansible):

```
set system services rest enable-explorer
set system services rest http port 8080
set system services rest http addresses 192.168.3.254
set system services rest control allowed-sources 192.168.3.134
```

Now browse to the SRX on port 8080:



Once NETCONF works, attempt your ansible first playbook.. If you were UNABLE to get netconf working on Junos, fix this first..

### 3. First attempt at ansible for junos

Best is to start as simple as possible. Here is an example of your first Ansible "playbook" file:

```
---
- name: get Device Facts
```

## Starting with Ansible on Linux and Junos (and Cisco IOS)

```
hosts: all
roles:
  - Juniper.junos
connection: local

tasks:
  - name: retrieving information from devices running Junos OS
    juniper_junos_facts:
      user=john

  - name: Print version
    debug:
      var: junos.version
```

The indentation will probably kill you: the markup language format used here is “yaml”.

Make sure that when you indent you use TWO SPACES AND NEVER TABS!  
Be rigorous about your formatting.. it is a major pain in the beginning.

Now *without* creating an *inventory* file, run your first Ansible playbook, the firewall used here is called “srx300”:

```
$ ansible-playbook -i srx300, playbook2-srx300.yaml --ask-pass
SSH password:
```

```
PLAY [get Device Facts]
*****
*****

TASK [Gathering Facts]
*****
*****
[WARNING]: Platform linux on host srx300 is using the discovered Python
interpreter at /usr/bin/python, but future
installation of another Python interpreter could change this. See
https://docs.ansible.com/ansible/2.9/reference\_appendices/interpreter\_discovery.html
for more information.
ok: [srx300]

TASK [retrieving information from devices running Junos OS]
*****
ok: [srx300]

TASK [Print version]
*****
ok: [srx300] => {
  "junos.version": "18.2R3-S2.9"
}

PLAY RECAP
*****
srx300 : ok=3    changed=0    unreachable=0    failed=0    skipped=0
rescued=0    ignored=0
```

That seems to work. Now on to a playbook that will allow you to change a Junos setting..

Note: the “**--ask-pas**” refers to the password that is needed to DECRYPT THE PRIVATE SSH KEY! It is *not* the user password. And yes of course you can automate this, e.g. by specifying your password in the inventory file (see below).

# Starting with Ansible on Linux and Junos (and Cisco IOS)

## 4. Changing a configuration parameter with Ansible

Create your 2<sup>nd</sup> playbook:

```
---
- name: get Device Facts
  hosts: all
  roles:
    - Juniper.junos
  connection: local

  tasks:
    - name: retrieving information from devices running Junos OS
      juniper_junos_facts:

    - name: Print version
      debug:
        var: junos.version

    - name: Deploy Ansible User
      juniper_junos_config:
        load: "merge"
        comment: "Configuring a Junos parameter with Ansible"
        src: ./resources/config-user.set
      register: response
    - name: 'Print result'
      debug:
        var: response
```

The file `./resources/config-user.set` with the JUNOS command is very simple:  
`set system ntp server 1.2.3.4`

Run this:

```
$ ansible-playbook -i srx300, playbook2-srx300.yaml --ask-pass
```

Once that works.. you will have a baseline to start expanding upon.

If you run into problems with ansible and modules that are not recognized etc.. (OSX!) you are not the only one. Start with Linux, that usually works best. Changes are that you need a different Python version or instruct ansible to use a different Python version. See below how to do this in your *inventory* file.

## 5. More automation

Here we will craft an *inventory* file that will :

- list the hosts (or host groups) to connect to. The group of hosts will be referred to as `lab_juniper_ro` and the hosts will be `srx300`.
- define the ssh – username and password to decrypt the Public key.
- can specify other general parameters like the Python incarnation to use.

The *inventory* file looks as follows:

```
[lab_juniper_ro]
srx300 ansible_user=john ansible_ssh_pass=john123
```

## Starting with Ansible on Linux and Junos (and Cisco IOS)

Now kick of the playbook without specifying the host:

```
$ ansible-playbook -i inventory playbook2-srx300.yaml
```

To specify variables for all hosts, use:

```
[all:vars]
ansible_user=john
ansible_ssh_pass=john123
```

```
[lab_juniper_ro]
srx300
```

In the ansible.cfg file you can specify the inventory file. This file can be found under /etc/ansible/ansible.cfg or you can create it in the directory where you fire the playbook from.

You can also specify the python path in the inventory file just below the `all:vars` line:

```
ansible_python_interpreter=/usr/bin/python
```

Using the following inventory file, it now works on OsX as well, as long as you direct ansible to use the proper Python version:

```
[all:vars]
ansible_python_interpreter=/Users/john/anaconda2/bin/python
ansible_user=john
ansible_ssh_pass=john123
```

```
[lab_juniper_ro]
srx300
```

To refer to the hosts-group “lab\_juniper\_ro”, refer to this group from within the playbook file:

```
hosts: lab_juniper_ro
```

As opposed to having separate “set” file with the relevant commands, you can also run the Junos commands directly from within the playbook file:

```
- name: Enable Samsung Internet access
  juniper_junos_config:
    load: "set"
    comment: "Allowing Samsung internet access, no change to
whatsapp."
    lines:
      - "deactivate security policies global policy deny-samsungs6-no-
internet-yes-whatsapp"
    logfile: logs/allow-internet-samsung.log
    register: response

- name: Print the response
  debug:
    var: response
```

## 6. Ansible and Cisco IOS

Once we have setup ansible for Junos, its relatively easy to produce a playbook that will configure a setting on a Cisco IOS device, as ansible comes with modules for IOS by default. Many but not all Cisco devices support NETCONF. If your switch or router does not support

## Starting with Ansible on Linux and Junos (and Cisco IOS)

NETCONF (my trusted 3750's do not) then ansible can still be used by forwarding the playbooks through a regular SSH tunnel.

Here is an example used on a 3750 without NETCONF:

```
---
- name: Configure IOS
  hosts: cisco
  connection: local
  become_method: enable
  gather_facts: no

  tasks:
    - name: configure interface settings
      ios_config:
        lines:
          - description test interface
          - ip address 1.2.3.4 255.255.255.255
        parents: interface loopback0
```

By the way; If you want the ansible playbook to prompt for e.g. a username and password, then use the following:

```
vars_prompt:
- name: "mgmt_username"
  prompt: "Username"
  private: no
- name: "mgmt_password"
  prompt: "Password"
```

### 7. Help..

There seems to be loads of information on the internet and my document seems to simply replicate what others have written before me.. but I hope it points you in the right direction in case you get stuck.

The Juniper.junos role that contains all the modules, can be found under:  
~/.ansible/roles/Juniper.junos/

Have fun with answerbook and..

“may the odds be ever in your favor”..