Event driven Junos automation with SaltStack

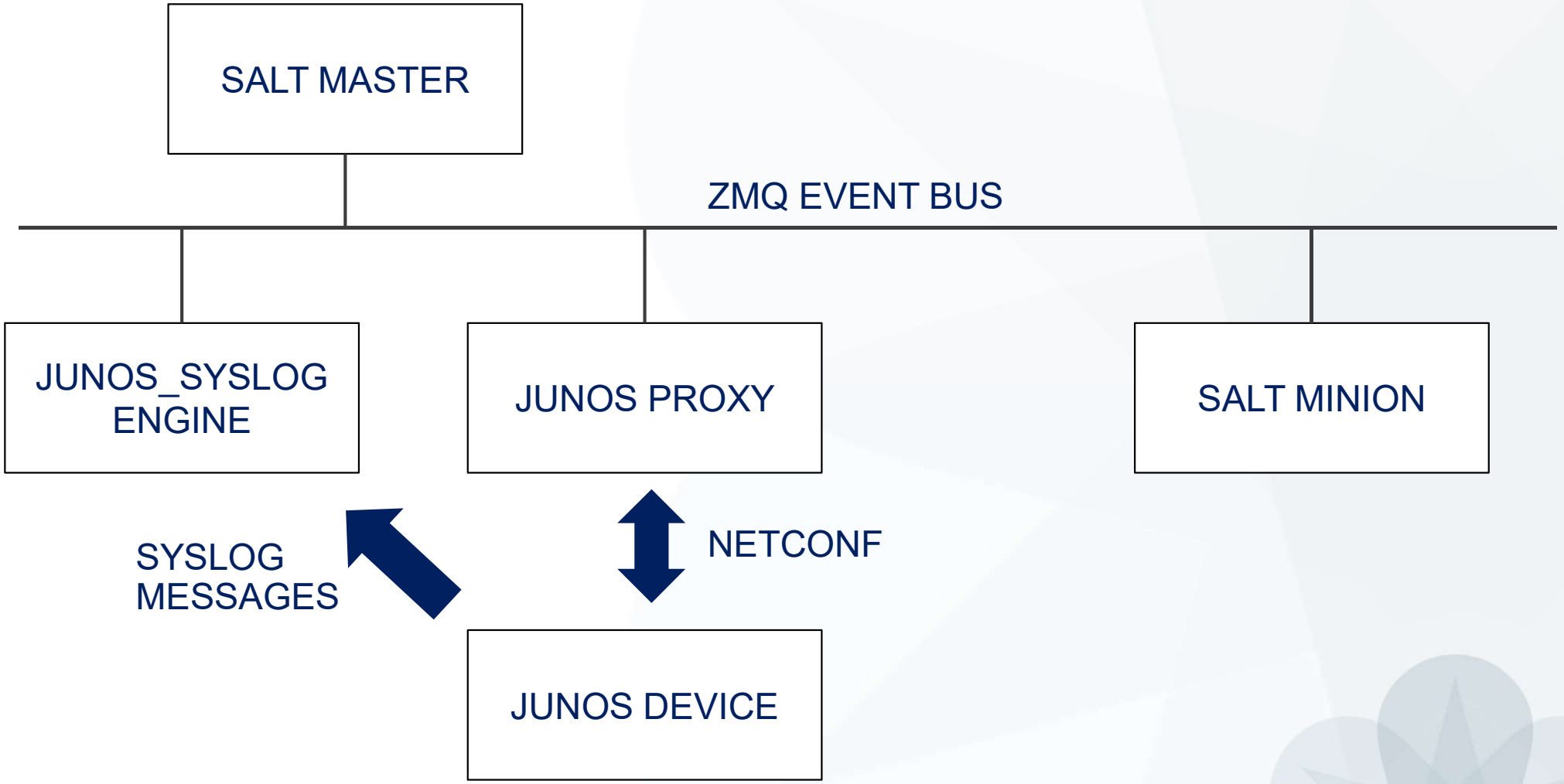
Khelil Sator
ksator@juniper.net
July 2017

JUNOS PROXY

- SaltStack supports Junos automation with a Salt proxy
- It provides execution modules for Junos so you can run commands on various machines in parallel with a flexible targeting system
  - https://docs.saltstack.com/en/latest/ref/modules/all/salt.modules.junos.html
- It provides state modules for Junos so you can apply sls files
  - https://docs.saltstack.com/en/latest/ref/states/all/salt.states.junos.html
- Junos facts are stored in salt grains.
- Junos proxy controls junos devices without installing salt on device.
- It uses Junos API: junos-eznc python library (pyez) and NETCONF on the device.

JUNOS SYSLOG ENGINE

- Listens to syslog events
- Extracts events information
- Sends information on the master/minion event bus.
- Control the type of events to be sent.
- Salt reactors has the ability to take actions according to these events (event driven automation).



Ansible orchestration live demo

- https://github.com/ksator/junos-automation-with-saltstack
- https://github.com/ksator/junos-automation-with-saltstack/wiki
- https://github.com/ksator/junos-automation-with-saltstack/wiki/17.-junos_syslog-engine-and-Salt's-reactor-system-end-to-end-demo#how-to-orchestrate-ansible-using-saltstack-event-driven-capabilities

SaltStack event driven capabilities

- A junos device (vqfx01) sends a syslog message UI_COMMIT_COMPLETED to the Junos_syslog Engine.
- The Junos_syslog Engine sends a ZMQ message jnpr/syslog/vqfx01/UI_COMMIT_COMPLETED to the event bus.
- The Salt Reactor component of the Salt Master daemon reacts to this event executing the reactor file /srv/reactor/on_commit.sls
- This sls file will have a minion to run an ansible playbook against the junos device that sent the syslog message (vqfx01)

Salt master configuration file

```
# more /etc/salt/master
file_roots:
 base:
   - /srv/salt
pillar_roots:
 base:
   - /srv/pillar
engines_dirs:
   - /srv/engines
engines:
   - junos_syslog:
       port: 516
reactor:
   - 'jnpr/syslog/*/UI_COMMIT_COMPLETED':
         - /srv/reactor/on_commit.sls
```

Reactor

- Reactor file:

```
# more /srv/reactor/on_commit.sls
Run ansible playbook:
  local.cmd.run:
    - tgt: minion_1
    - arg:
        - ansible-playbook /srv/ansible/junos_get_config/pb.2.yml --extra-
vars target={{ data['hostname'] }} -i /srv/ansible/hosts
```

- So the reactor:
  - watches for events jnpr/syslog/*/UI_COMMIT_COMPLETED
  - and runs the below command:

```
salt minion_1 cmd.run "ansible-playbook
/srv/ansible/junos_get_config/pb.2.yml --extra-vars target={{
data['hostname'] }} -i /srv/ansible/hosts"
```

Pass the value of the variable hosts to the playbook

- In the ansible playbook /srv/ansible/junos_get_config/pb.2.yml, the value for the hosts variable is the variable target, which has no default value.
  - We pass the value of the variable target to ansible using the flag extra-vars
    - `--extra-vars target={{ data['hostname'] }}`
  - If {{ data['hostname'] }} is equal to vqfx01, the master will actually run:

```
salt minion_1 cmd.run "ansible-playbook
/srv/ansible/junos_get_config/pb.2.yml --extra-vars target=vqfx01 -i
/srv/ansible/hosts"
```

  - So the minion_1 will run:

```
ansible-playbook /srv/ansible/junos_get_config/pb.2.yml --extra-vars
target=vqfx01 -i /srv/ansible/hosts
```

Junos device syslog configuration

- For junos_syslog engine to receive events, syslog must be set on the junos device:
  - The ip address is the one of the server running the syslog engine
  - The port is the port where the engine is listening for events.

```
vagrant@vqfx01> show configuration system syslog host 192.168.233.17
any any;
match UI_COMMIT_COMPLETED;
port 516;
```

Commit a configuration change on junos device

- Commit a configuration change on junos device:

```
vagrant@vqfx01# commit
```

- tcpdump output on junos_syslog engine :

```
22:48:53.957045 IP 192.168.233.158.59781 > 192.168.233.17.516: UDP, length 75
  0x0000:  000c 2911 2ecd 000c 2943 2de4 0800 4500  ..).....)C-...E.
  0x0010:  0067 cef6 0000 3f11 588e c0a8 e99e c0a8  .g....?.X.......
  0x0020:  e911 e985 0204 0053 8ea6 3c31 3838 3e4a  .......S..<188>J
  0x0030:  756e 2032 3720 3230 3a34 383a 3431 2076  un.27.20:48:41.v
  0x0040:  7166 7830 3120 6d67 645b 3137 3132 5d3a  qfx01.mgd[1712]:
  0x0050:  2055 495f 434f 4d4d 4954 5f43 4f4d 504c  .UI_COMMIT_COMPL
  0x0060:  4554 4544 3a20 636f 6d6d 6974 2063 6f6d  ETED:.commit.com
  0x0070:  706c 6574 65                             plete
```

Event publishes by junos_syslog engine

```
jnpr/syslog/vqfx01/UI_COMMIT_COMPLETED  {
    "_stamp": "2017-06-27T20:58:09.034524",
    "daemon": "mgd",
    "event": "UI_COMMIT_COMPLETED",
    "facility": 23,
    "hostip": "192.168.233.158",
    "hostname": "vqfx01",
    "message": "commit complete",
    "pid": "2945",
    "priority": 188,
    "raw": "<188>Jun 27 20:57:55 vqfx01 mgd[2945]: UI_COMMIT_COMPLETED:
commit complete",
    "severity": 4,
    "timestamp": "2017-06-27 22:58:09"
}
```

- So {{ data['hostname'] }} is equal to vqfx01

Event publishes by Salt master

```
salt/job/20170627225809048834/new {
    "_stamp": "2017-06-27T20:58:09.051006",
    "arg": [
        "ansible-playbook /srv/ansible/junos_get_config/pb.2.yml --extra-
vars target=vqfx01 -i /srv/ansible/hosts"
    ],
    "fun": "cmd.run",
    "jid": "20170627225809048834",
    "minions": [
        "minion_1"
    ],
    "tgt": "minion_1",
    "tgt_type": "glob",
    "user": "sudo_ksator"
}
```

Ansible playbook execution on the minion

- So the minion_1 runs:

```
ansible-playbook /srv/ansible/junos_get_config/pb.2.yml --extra-vars
target=vqfx01 -i /srv/ansible/hosts
```

- And returns the command output to the master using the event bus


Thank you