Automate SONiC Configuration using Ansible

Suhaib Ahmad
Technical Marketing Engineer

DEVNET-2990

Cisco Webex App

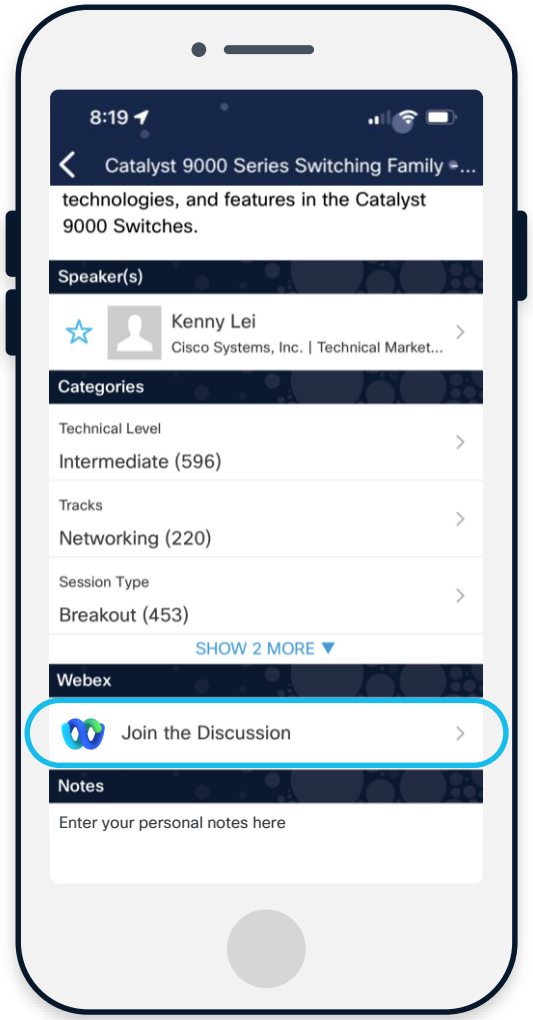
Questions?

Use Cisco Webex App to chat
with the speaker after the session

How

1. Find this session in the Cisco Live Mobile App
2. Click “Join the Discussion”
3. Install the Webex App or go directly to the Webex space
4. Enter messages/questions in the Webex space

Webex spaces will be moderated by the speaker until June 13, 2025.



https://ciscolive.ciscoevents.com/
ciscolivebot/**#DEVNET-2990**

Agenda

01 Introduction to SONiC

02 Configuration Modes

03 Ansible Automation

04 Cisco SONiC Ansible collection

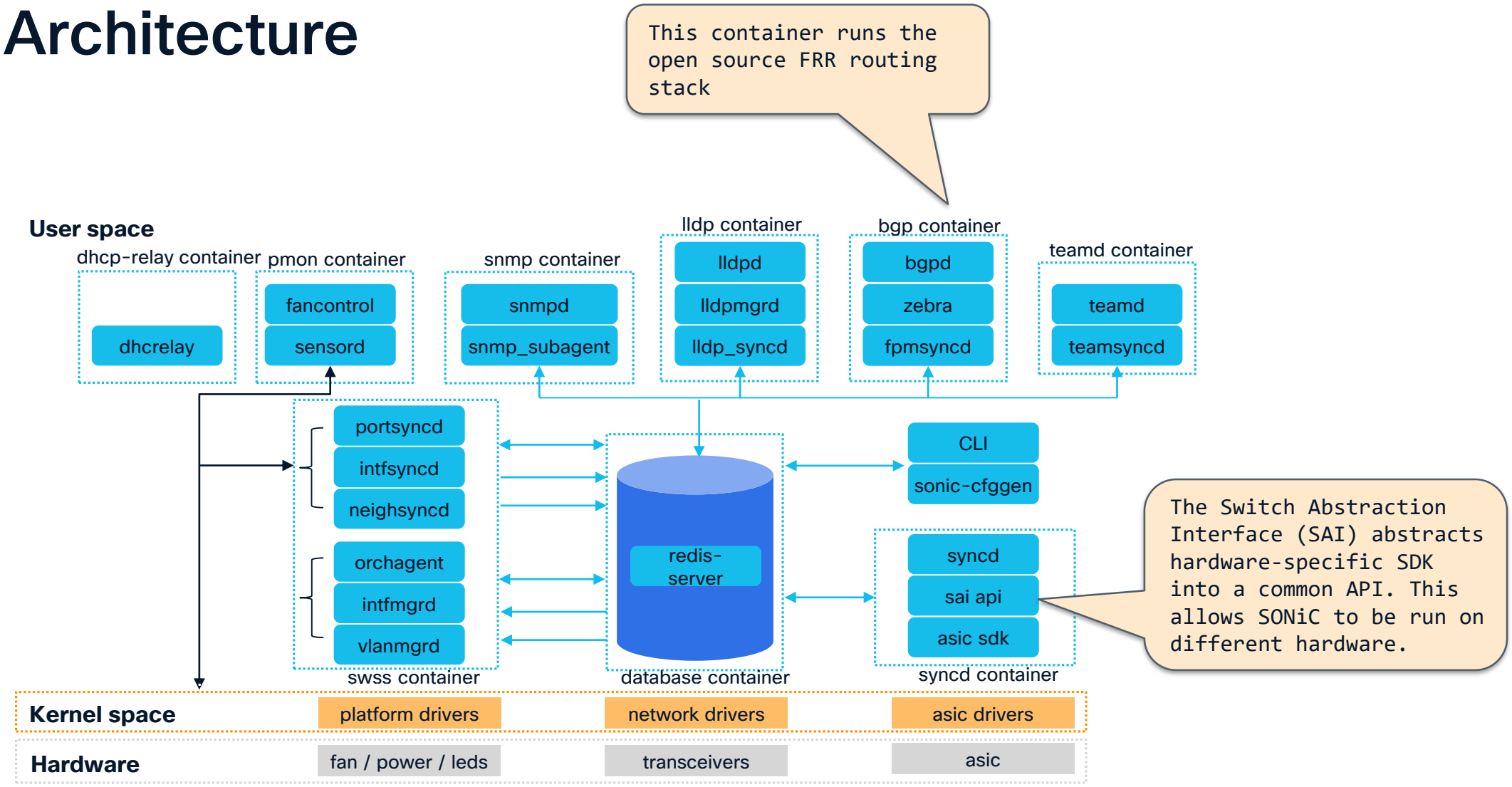
05 Conclusion

Introduction to SONiC

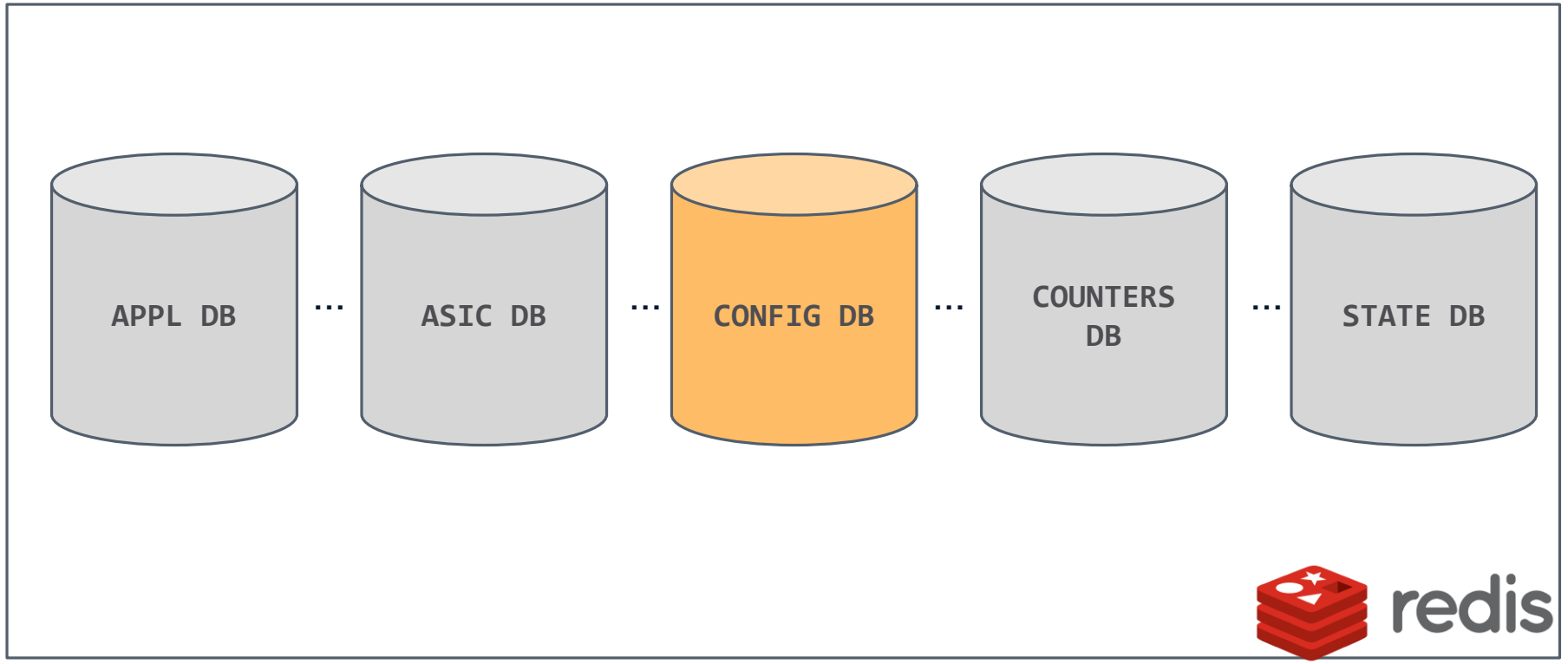
The SONiC Network Operating System

- SONiC – Software for Open Networking in the Cloud is an open-source network operating system based on Debian Linux.
- Initially created by Microsoft for use in their Azure datacenters. Open-sourced by Microsoft in 2016.
- It became part of the Linux Foundation in 2022 which focuses on the software components of SONiC
- Open Compute Platform (OCP) helps in aligning hardware and specifications like the Switch Abstraction Interface (SAI).
- https://github.com/sonic-net/SONiC/wiki
- https://sonicfoundation.dev/

SONiC Architecture



SONiC Redis Databases



SONiC Redis Container

SONiC Redis Databases

```
root@sonic:~# cat /var/run/redis/sonic-db/database_config.json
{
    "INSTANCES": {
        "redis": {
            "hostname": "127.0.0.1",
            "port": 6379,
            "unix_socket_path": "/var/run/redis/redis.sock",
            "persistence_for_warm_boot": "yes"
        }
    },
    "DATABASES": {
        "APPL_DB": {
            "id": 0,
            "separator": ":",
            "instance": "redis"
        },
        "ASIC_DB": {
            "id": 1,
            "separator": ":",
            "instance": "redis"
        },
        "COUNTERS_DB": {
            "id": 2,
            "separator": ":",
            "instance": "redis"
        },
        "LOGLEVEL_DB": {
            "id": 3,
            "separator": ":",
            "instance": "redis"
        },
        "CONFIG_DB": {
            "id": 4,
            "separator": "|",
            "instance": "redis"
        },
        "PFC_WD_DB": {
            "id": 5,
            "separator": ":",
            "instance": "redis"
        },
```

```
        "FLEX_COUNTER_DB": {
            "id": 5,
            "separator": ":",
            "instance": "redis"
        },
        "STATE_DB": {
            "id": 6,
            "separator": "|",
            "instance": "redis"
        },
        "SNMP_OVERLAY_DB": {
            "id": 7,
            "separator": "|",
            "instance": "redis"
        },
        "RESTAPI_DB": {
            "id": 8,
            "separator": "|",
            "instance": "redis"
        },
        "GB_ASIC_DB": {
            "id": 9,
            "separator": ":",
            "instance": "redis"
        },
        "GB_COUNTERS_DB": {
            "id": 10,
            "separator": ":",
            "instance": "redis"
        },
        "GB_FLEX_COUNTER_DB": {
            "id": 11,
            "separator": ":",
            "instance": "redis"
        },
        "APPL_STATE_DB": {
            "id": 14,
            "separator": ":",
            "instance": "redis"
        }
    },
    "VERSION": "1.0"
```

SONiC Configuration Database

```
root@sonic:~# redis-cli -n 4
127.0.0.1:6379[4]> KEYS *
  1) "LOGGER|SAI_API_BUFFER"
  2) "PORT|Ethernet176"
  3) "PORT|Ethernet508"
  4) "PORT|Ethernet388"
  5) "PORT|Ethernet84"
  6) "PORT|Ethernet436"
  7) "LOGGER|teammgrd"
  8) "PORT|Ethernet380"
  9) "LOGGER|SAI_API_TAM"
 10) "LOGGER|SAI_API_ARS_PROFILE"
 11) "LOGGER|SAI_API_BFD"
 12) "CRM|Config"
 13) "PORT|Ethernet484"
 14) "LOGGER|neighsyncd"
 15) "PORT|Ethernet500"
 16) "FEATURE|gnmi"
 17) "LOGGER|SAI_API_L2MC"
 18) "LOGGER|SAI_API_IPMC_GROUP"
 19) "LOGGER|SAI_API_DEBUG_COUNTER"
 20) "PORT|Ethernet168"
 21) "PORT|Ethernet260"
 22) "FLEX_COUNTER_TABLE|QUEUE"
 23) "PORT|Ethernet64"
 24) "SYSLOG_CONFIG_FEATURE|lldp"
 25) "LOGGER|orchagent"
 26) "SYSTEM_DEFAULTS|mux_tunnel_egress_acl"
 27) "PORT|Ethernet384"
 28) "FEATURE|dhcp_relay"
 29) "LOGGER|SAI_API_RPF_GROUP"
 30) "LOGGER|SAI_API_TWAMP"
 31) "PORT|Ethernet256"
 32) "PORT|Ethernet372"
 33) "FLEX_COUNTER_TABLE|RIF"
 34) "FEATURE|swss"
 35) "SYSLOG_CONFIG_FEATURE|gnmi"
 36) "SYSLOG_CONFIG_FEATURE|teamd"
 37) "LOGGER|SAI_API_DASH_VNET"
 38) "LOGGER|SAI_API_SCHEDULER_GROUP"
 39) "PORT|Ethernet104"
 40) "PORT|Ethernet432"
 41) "FLEX_COUNTER_TABLE|PORT_BUFFER_DROP"
 42) "AUTO_TECHSUPPORT_FEATURE|mux"
 43) "PORT|Ethernet360"
 44) "AUTO_TECHSUPPORT_FEATURE|syncd"
 45) "PORT|Ethernet340"
 46) "PORT|Ethernet292"
 47) "PORT|Ethernet368"
```

Accessing CONFIG_DB from Redis Shell

```
127.0.0.1:6379[4]> HGETALL PORT|Ethernet176
 1) "admin_status"
 2) "up"
 3) "alias"
 4) "etp22a"
 5) "index"
 6) "22"
 7) "lanes"
 8) "776,777,778,779"
 9) "mtu"
10) "9100"
11) "speed"
12) "400000"
13) "subport"
14) "1”

127.0.0.1:6379[4]> HGET PORT|Ethernet176 admin_status
"up"
127.0.0.1:6379[4]> HGET PORT|Ethernet176 speed
"400000"

127.0.0.1:6379[4]> HGETALL WRED_PROFILE|TEST_LOSSLESS
 1) "ecn"
 2) "ecn_green_yellow"
 3) "green_drop_probability"
 4) "5"
 5) "green_max_threshold"
 6) "10485760"
 7) "green_min_threshold"
 8) "2097152"
 9) "yellow_drop_probability"
10) "0"
11) "yellow_max_threshold"
12) "6144000"
13) "yellow_min_threshold"
14) "0”

127.0.0.1:6379[4]> HGET WRED_PROFILE|TEST_LOSSLESS green_drop_probability
"5”

127.0.0.1:6379[4]> HGETALL INTERFACE|Ethernet256
1) "vrf_name"
2) "Vrf-blue"

127.0.0.1:6379[4]> HGET INTERFACE|Ethernet256 vrf_name
"Vrf-blue"
```

SONiC Configuration Database

- Acts as the source of truth for SONiC configuration and exists as database 4 (`CONFIG_DB`) in Redis.
- SONiC applications subscribe to `CONFIG_DB` and generate their own running configuration.
- The Configuration Database a Redis key-value structure for all items in the Redis DB. But also maintains a json-based configuration in `/etc/sonic/config_db.json`. The Redis and JSON schema is maintained at https://github.com/sonic-net/sonic-buildimage/blob/master/src/sonic-yang-models/doc/Configuration.md
- The CONFIG_DB in Redis is analogous to the running configuration of the device. Whereas `/etc/sonic/config_db.json` is analogous to the startup configuration of the device.

```
127.0.0.1:6379[4]> HGETALL PORT|Ethernet176
 1) "admin_status"
 2) "up"
 3) "alias"
 4) "etp22a"
 5) "index"
 6) "22"
 7) "lanes"
 8) "776,777,778,779"
 9) "mtu"
10) "9100"
11) "speed"
12) "400000"
13) "subport"
14) "1”
```

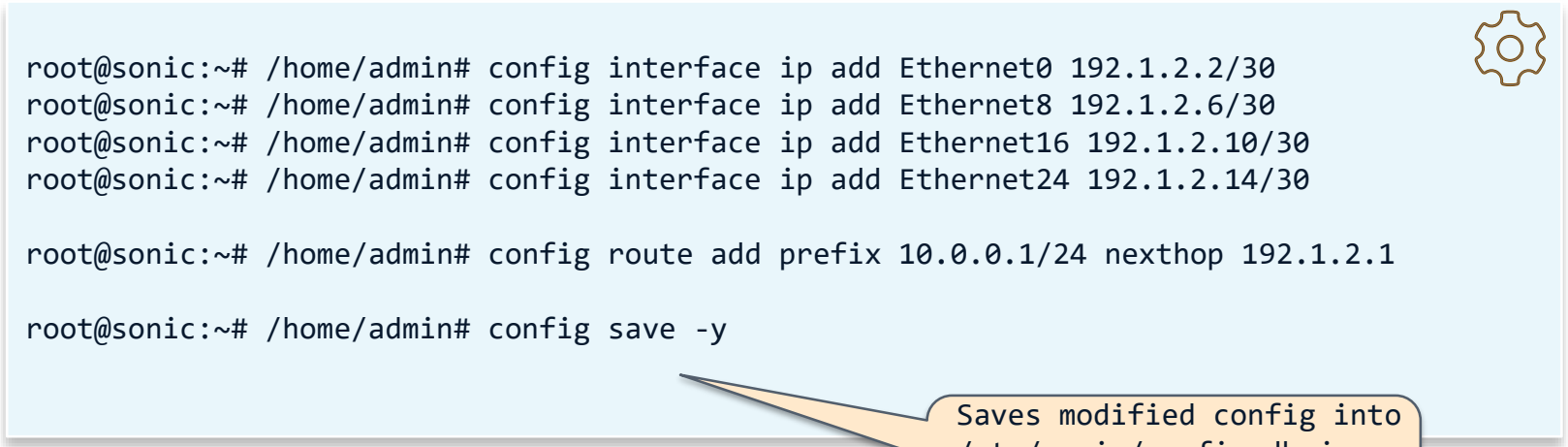
jq is a popular command-line tool for querying JSON files

```
root@sonic:~# jq '.PORT["Ethernet176"]' /etc/sonic/config_db.json
{
  "admin_status": "up",
  "alias": "etp22a",
  "index": "22",
  "lanes": "776,777,778,779",
  "mtu": "9100",
  "speed": "400000",
  "subport": "1"
}
root@sonic:~#
```

SONiC Configuration Modes

SONiC CLI

- SONiC also has a simple Python Click based CLI interface. The CLI interface supports show commands, config commands, and other “action” commands (e.g., sonic-installer). In addition to these commands via the Click CLI interface, SONiC supports various Debian Linux command-line utilities.
- show commands can be executed by any user whereas any config commands require sudo privileges.
- show commands pull data from appropriate SONiC databases and display it on the terminal screen
- config commands directly modify the `CONFIG_DB` in Redis. To make these commands persist, users must issue a `config save` command
- https://github.com/sonic-net/sonic-utilities/blob/master/doc/Command-Reference.md



JSON files: config load and sonic-cfggen

- SONiC users can express configuration as per SONiC's JSON schema and push it directly to the CONFIG_DB in Redis.
- The sonic-cfggen is a utility to read SONiC config from various input types and then write the config to the config database in Redis, print as a json or render a jinja2 config template. We can use it to write configuration to the CONFIG_DB in Redis
- Additionally, the `config load` command in SONiC CLI can load configuration into the Redis CONFIG_DB. By default, `config load` loads `/etc/sonic/config_db.json` into CONFIG_DB. However, users can specify their own configuration patches in other JSON files.

```
root@sonic:~# cat ip.json
{
"INTERFACE": {
  "Ethernet0": {},
  "Ethernet0|192.1.2.2/24": {},
  "Ethernet8": {},
  "Ethernet8|192.1.4.2/24": {},
  "Ethernet80": {},
  "Ethernet80|192.1.22.1/24": {}
 }
}

cisco@c1:~$ sonic-cfggen -j ip.json --write-to-db

OR

cisco@c1:~$ sudo config load ip.json
Load config from the file(s) ip.json ? [y/N]: y
Running command: /usr/local/bin/sonic-cfggen -j ip.json --
write-to-db
```

```
root@sonic:~# show ip interfaces
Interface    Master    IPv4 address/mask    Admin/Oper    BGP Neighbor    Neighbor IP
-----------  --------  -------------------  ------------  --------------  -------------
Ethernet0              192.1.2.2/24         up/up         N/A             N/A
Ethernet8              192.1.4.2/24         up/up         N/A             N/A
Ethernet40             192.1.22.1/24        up/up         N/A             N/A
Ethernet80             192.1.22.1/24        up/down       N/A             N/A
docker0                240.127.1.1/24       up/down       N/A             N/A
eth0                   1.18.1.1/16          up/up         N/A             N/A
lo                     127.0.0.1/16         up/up         N/A             N/A
```

Config load uses sonic-cfggen to write configuration to CONFIG_DB

JSON files: config reloads

- For Day 0 or config replace operations, users can reload SONiC with a new startup configuration (`/etc/sonic/config_db.json`) or a user specified config file.

```
root@sonic:~# sudo mv new_config.json /etc/sonic/config_db.json
root@sonic:~# sudo config reload -y
Acquired lock on /etc/sonic/reload.lock
Disabling container and routeCheck monitoring ...
Stopping SONiC target ...
Running command: /usr/local/bin/sonic-cfggen -j /etc/sonic/init_cfg.json -j /etc/sonic/config_db.json --write-to-db
Running command: /usr/local/bin/db_migrator.py -o migrate
Running command: /usr/local/bin/sonic-cfggen -d -y /etc/sonic/sonic_version.yml -t /usr/share/sonic/templates/sonic-environment.j2,/etc/sonic/sonic-
environment
Restarting SONiC target ...
Enabling container and routeCheck monitoring ...
Reloading Monit configuration ...
Reinitializing monit daemon
Released lock on /etc/sonic/reload.lock
```

JSON files: config reloads

Config reload causes key SONiC containers to restart

```
root@sonic:~# docker container ls -a

CONTAINER ID   IMAGE                                COMMAND                  CREATED        STATUS                      PORTS     NAMES
9050488b3dec   docker-snmp:latest                   "/usr/bin/docker-snm…"   2 months ago   Exited (0) 44 seconds ago             snmp
695fee499783   docker-platform-monitor:latest       "/usr/bin/docker_ini…"   2 months ago   Exited (0) 49 seconds ago             pmon
4922a1d0664f   docker-sonic-mgmt-framework:latest   "/usr/local/bin/supe…"   2 months ago   Exited (0) 51 seconds ago             mgmt-framework
1449d599d2a2   docker-lldp:latest                   "/usr/bin/docker-lld…"   2 months ago   Exited (0) 45 seconds ago             lldp
b8d431d4ac1d   docker-sonic-gnmi:latest             "/usr/local/bin/supe…"   2 months ago   Exited (0) 51 seconds ago             gnmi
48ec7b1c2940   bd474e1c14fe                         "/usr/bin/docker_ini…"   2 months ago   Up 13 seconds                         dhcp_relay
64b2b2ee6396   docker-fpm-frr:latest                "/usr/bin/docker_ini…"   2 months ago   Up 7 seconds                          bgp
1bf9403740d4   docker-router-advertiser:latest      "/usr/bin/docker-ini…"   2 months ago   Up 13 seconds                         radv
4eacf05d1b24   docker-syncd-cisco:latest            "/usr/local/bin/supe…"   2 months ago   Up 14 seconds                         syncd
95b7a6270dfa   docker-teamd:latest                  "/usr/local/bin/supe…"   2 months ago   Up 14 seconds                         teamd
f3c2420c1338   docker-orchagent:latest              "/usr/bin/docker-ini…"   2 months ago   Up 16 seconds                         swss
806862ade41d   docker-eventd:latest                 "/usr/local/bin/supe…"   2 months ago   Up 17 seconds                         eventd
5806236beada   docker-database:latest               "/usr/local/bin/dock…"   2 months ago   Up 2 weeks                            database
```

JSON files: configuration patches

- SONiC also allows users the option to “patch” the running configuration supporting ADD, REPLACE, and REMOVE.

```
root@sonic:~# show runningconfiguration ports | grep Ethernet0 -A 8
    "Ethernet0": {
        "admin_status": "up",
        "alias": "eth0a",
         "index": "0",
        "lanes": "1536,1537,1538,1539",
        "mtu": "9100",
        "speed": "400000",
        "subport": "1"

root@sonic:~# show runningconfiguration all | grep -w VLAN -A 5
    "VLAN": {
        "Vlan10": {
            "vlanid": "10"
        },
        "Vlan20": {
            "vlanid": "20"
```

```
root@sonic:~# cat patch.json
[
  {
    "op": "remove",
    "path": "/VLAN/Vlan20"
  },
  {
   "op": "add",
   "path": "/PORT/Ethernet0/fec",
   "value": "rs"
  },
  {
   "op": "replace",
   "path": "/PORT/Ethernet0/mtu",
   "value": "9000"
  }
]
```

```
root@sonic:~# show interfaces status Ethernet0
  Interface                Lanes    Speed    MTU    FEC    Alias    Vlan    Oper    Admin                                          Type    Asym PFC
-----------  ------------------  -------  -----  -----  -------  ------  ------  -------  ---------------------------------------------  ----------
  Ethernet0  1536,1537,1538,1539     400G   9100    N/A  eth0a    routed      up       up  QSFP-DD Double Density 8X Pluggable Transceiver         N/A

root@sonic:~# show vlan brief
+-----------+-------------+---------+----------------+------------+-----------------------+
|   VLAN ID | IP Address  | Ports   | Port Tagging   | Proxy ARP  | DHCP Helper Address   |
+===========+=============+=========+================+============+=======================+
|        10 |             |         |                | disabled   |                       |
+-----------+-------------+---------+----------------+------------+-----------------------+
|        20 |             |         |                | disabled   |                       |
+-----------+-------------+---------+----------------+------------+-----------------------+
```

JSON files: configuration patches

```
root@sonic:~# sudo config apply-patch patch.json
Patch Applier: localhost: Patch application starting.
Patch Applier: localhost: Patch: [{"op": "remove", "path": "/VLAN/Vlan20"}, {"op": "add", "path": "/PORT/Ethernet0/fec", "value": "rs"}, {"op": "replace", "path":
"/PORT/Ethernet0/mtu", "value": "9000"}]
Patch Applier: localhost getting current config db.
Patch Applier: localhost: simulating the target full config after applying the patch.
Patch Applier: localhost: validating all JsonPatch operations are permitted on the specified fields
Patch Applier: localhost: validating target config does not have empty tables,
                          since they do not show up in ConfigDb.
Patch Applier: localhost: sorting patch updates.
Patch Applier: The localhost patch was converted into 3 changes:
Patch Applier: localhost: applying 3 changes in order:
Patch Applier:   * [{"op": "remove", "path": "/VLAN/Vlan20"}]
Patch Applier:   * [{"op": "replace", "path": "/PORT/Ethernet0/mtu", "value": "9000"}]
Patch Applier:   * [{"op": "add", "path": "/PORT/Ethernet0/fec", "value": "rs"}]
Patch Applier: localhost: verifying patch updates are reflected on ConfigDB.
Patch Applier: localhost patch application completed.
Patch applied successfully.

root@sonic:~# show interfaces status Ethernet0
  Interface                Lanes    Speed    MTU    FEC    Alias    Vlan    Oper    Admin                                          Type    Asym PFC
-----------  ------------------  -------  -----  -----  -------  ------  ------  -------  ---------------------------------------------  ----------
  Ethernet0  1536,1537,1538,1539     400G   9000     rs  eth0a_1  routed      up       up  QSFP-DD Double Density 8X Pluggable Transceiver         N/A


root@sonic:~# show vlan brief
+-----------+--------------+---------+----------------+-------------+-----------------------+
|   VLAN ID | IP Address   | Ports   | Port Tagging   | Proxy ARP   | DHCP Helper Address   |
+===========+==============+=========+================+=============+=======================+
|        10 |              |         |                | disabled    |                       |
+-----------+--------------+---------+----------------+-------------+-----------------------+


root@sonic:~# config save -y
Running command: /usr/local/bin/sonic-cfggen -d --print-data > /etc/sonic/config_db.json
```

Special Consideration: FRR

- The routing stack in SONiC is built on the open-source FRR project. FRR exists in SONiC as a docker container named bgp.
- The bgp container (FRR) has a native shell (vtysh) for configuration that has a user-friendly IOS-like interface. Community SONiC does not have any other CLI for configuring routing features.
- However, using vtysh means configuration is maintained inside the bgp docker container instead of the configuration database.
- Users can choose whether routing configuration is to persist in FRR's domain or inside SONiC's /etc/sonic/config_db.json

```
root@sonic:~# vtysh

Hello, this is FRRouting (version 8.5.4).
Copyright 1996-2005 Kunihiro Ishiguro, et al.

sonic# show bgp summary

IPv4 Unicast Summary (VRF default):
BGP router identifier 192.1.22.2, local AS number 65000 vrf-id 0
BGP table version 400000
RIB entries 0, using 0 bytes of memory
Peers 2, using 1449 KiB of memory

Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc
172.20.166.52   4      65000      1146       193        0    0    0 5d00h45m       Active        0 N/A
192.1.22.1      4      65000         0         4        0    0    0    never       Active        0 N/A

Total number of neighbors 2
sonic#
```

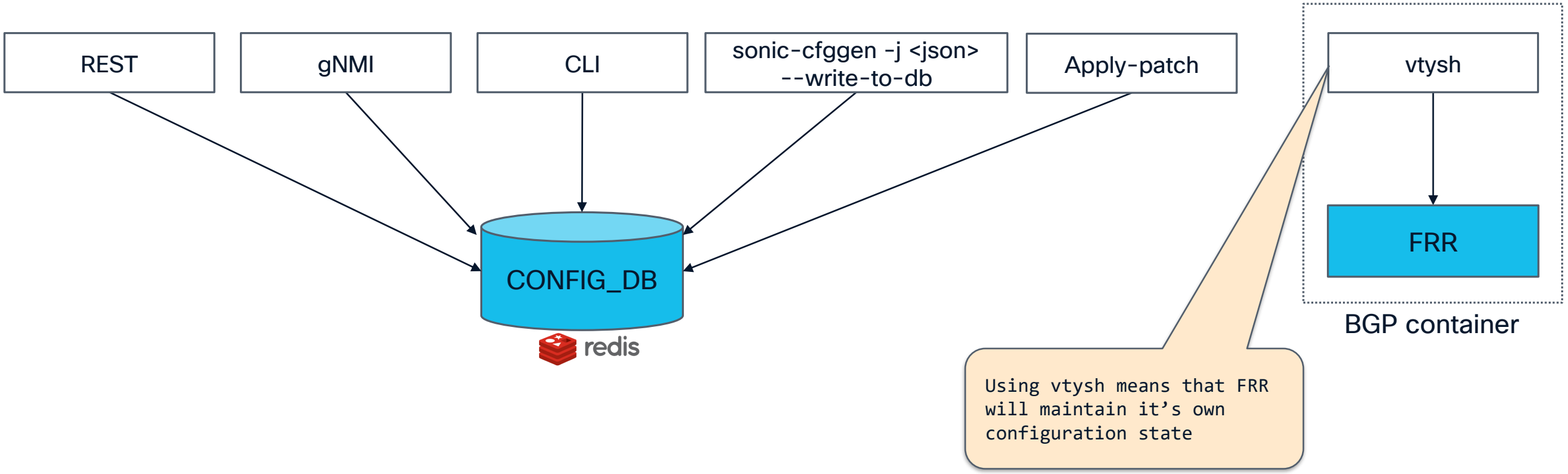
Programmatic APIs: gNMI and REST

- gNMI is a widely used protocol based on gRPC that allows users to configure, manage, and stream telemetry data from network devices.
- SONiC also supports HTTP based REST APIs to configure network state in devices.
- SONiC maintains native YANG models for configuration DB at https://github.com/sonic-net/sonic-buildimage/tree/master/src/sonic-yang-models/yang-models

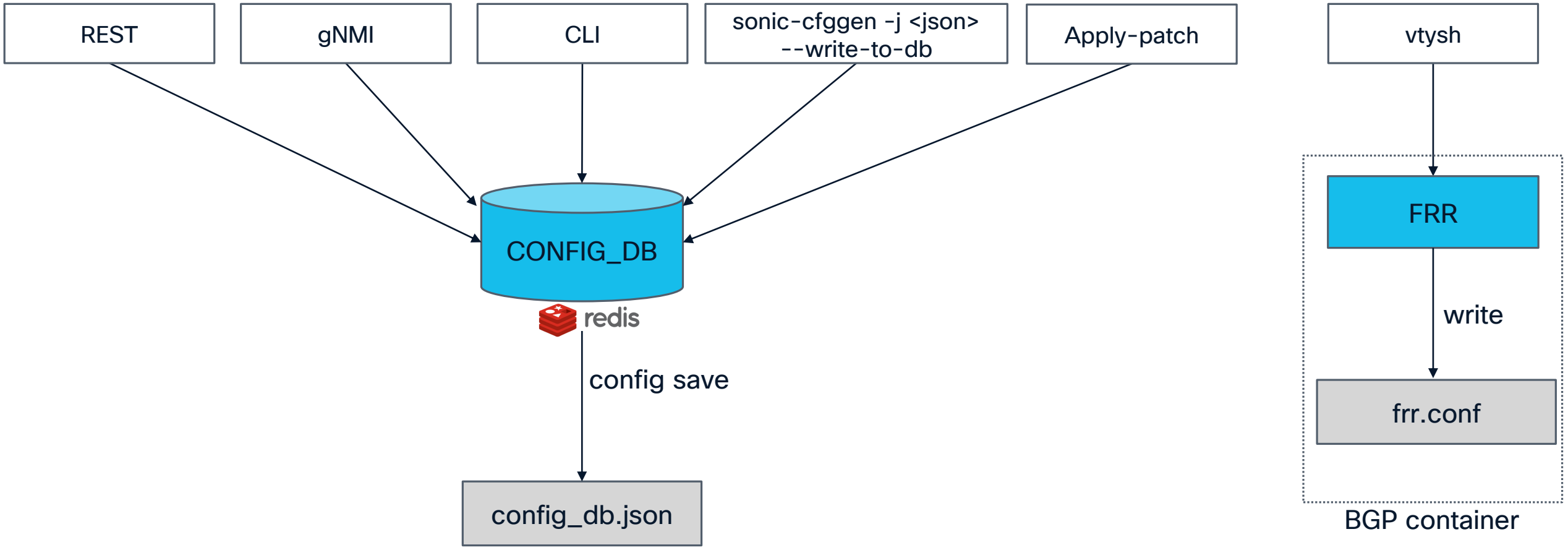
```
root@8ktme:/home/cisco# pygnmicli -t 1.18.1.1:50051 \
    -r "./certs/RootCA.crt" \
    -c "./certs/c1/c1.sonic.cisco.com.cer" \
    -k "./certs/c1/c1.sonic.cisco.com.key" \
    -u cisco -p cisco123 \
    --gnmi-path-target CONFIG_DB \
    -o get -x PORT/Ethernet0
```


```
{
    "notification": [
        {
            "timestamp": 1731590130064395032,
            "prefix": null,
            "alias": null,
            "atomic": false,
            "update": [
                {
                    "path": "PORT/Ethernet0",
                    "val": {
                        "admin_status": "up",
                        "alias": "etp0",
                        "index": "0",
                        "lanes": "2304,2305,2306,2307,2308,2309,2310,2311",
                        "mtu": "9100",
                        "speed": "400000"
                    }
                }
            ]
        }
    ]
}
```

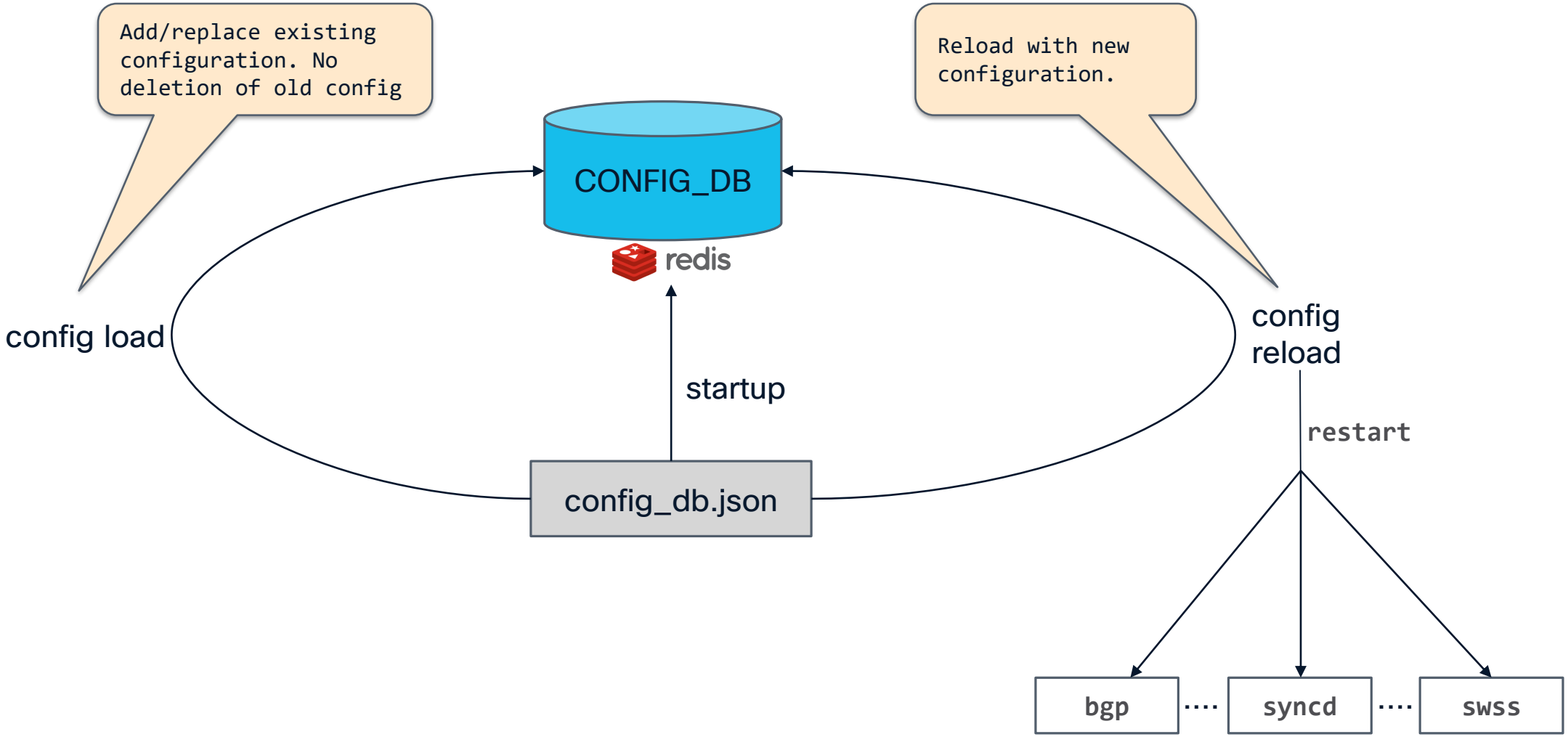

Configuration Management: config options



Configuration Management: config save operations



Configuration Management: config load operations



Ansible Automation

SONiC and Ansible

- Ansible is an open-source automation tool developed by RedHat.
- The Ansible platform allows users to develop playbooks that allow them to perform tasks on a set of targeted devices.
- Widely used in IT infrastructure for automating deployment and management of compute servers, network devices, and other infrastructure.
- Pre-defined modules can be created and shared across the community using Ansible galaxy.

- SONiC runs on Debian Linux and allows users to access the Linux shell. It can utilize several pre-existing Ansible modules built for Linux based systems.
- SONiC's native support JSON based configuration actions fit with Ansible-style automation actions.
- Common Ansible transport mechanisms (SSH, REST, etc.) are already supported by SONiC

SONiC CLI using ansible.builtin.shell

```
- name: Leaf configuration
  gather_facts: false
  hosts: leaf_switches
  tasks:
   - name: Configure IP address for Ethernet0
     ansible.builtin.shell: config interface ip add Ethernet0 192.1.1.1/24
     become: True
   - name: Save config to /etc/sonic/config_db.json
     ansible.builtin.shell: config save -y
     become: True
```

Day 0 config using Ansible

```
- name: Leaf configuration
  gather_facts: false
  hosts: leaf_switches
  tasks:
   - name: Copy config
     copy: src=leaf_config_db.json
           dest=/etc/sonic/config_db.json
     become: true
   - name: Reload config
     ansible.builtin.shell: config reload -y
     become: True
```

Using config load or sonic-cfggen

```
- name: Leaf configuration
  gather_facts: false
  hosts: leaf_switches
  tasks:
   - name: Copy config
     copy: src=config.json
           dest=/home/cisco/config.json
     become: true
   - name: Load config
     ansible.builtin.shell: config load config.json -y
     become: True
```

```
- name: Leaf configuration
  gather_facts: false
  hosts: leaf_switches
  tasks:
   - name: Copy config
     copy: src=config.json
           dest=/home/cisco/config.json
     become: true
   - name: Load config
     ansible.builtin.shell: sonic-cfggen -j
      /home/cisco/config.json --write-to-db
     become: True
```

```
cisco@8ktme:~/suhahmad/tools/sonic/clus25$ cat config.json
{
"INTERFACE": {
  "Ethernet0": {},
  "Ethernet0|192.1.2.2/24": {},
  "Ethernet8": {},
  "Ethernet8|192.1.4.2/24": {},
  "Ethernet80": {},
  "Ethernet80|192.1.22.1/24": {}
 }
}
```

Using apply-patch for defined operations

```
cisco@8ktme:~/suhahmad/tools/sonic/clus25$ cat patch.json
[
  {
    "op": "replace",
    "path": "/PORT/Ethernet0/alias",
    "value": "eth0a_1"
  },
  {
   "op": "add",
   "path": "/PORT/Ethernet0/fec",
   "value": "rs"
  }
]
```

```
- name: Leaf configuration
  gather_facts: false
  hosts: leaf_switches
  tasks:
   - name: Copy config
     copy: src=patch.json
           dest=/home/cisco/patch.json
     become: true
   - name: Apply config patch
     ansible.builtin.shell: config apply-patch patch.json
     become: True
```

Getting Better: Generating JSON files using templates

```
cisco@8ktme:~$ cat interface_config.json.j2
{
  "INTERFACE": {
    "Ethernet0": {},
    "Ethernet0|{{ ethernet0_ip }}": {},
    "Ethernet8": {},
    "Ethernet8|{{ ethernet8_ip }}": {},
    "Ethernet80": {},
    "Ethernet80|{{ ethernet80_ip }}": {}
  }
}

cisco@8ktme:~$ cat host_vars/r1.yml
ethernet0_ip: 192.1.2.2/24
ethernet8_ip: 192.1.4.2/24
ethernet80_ip: 192.1.22.2/24

cisco@8ktme:~$ cat host_vars/r2.yml
ethernet0_ip: 10.1.1.2/24
ethernet8_ip: 10.1.4.2/24
ethernet80_ip: 10.1.22.2/24
```

Getting Better: Generating JSON files using templates

```
- name: Generate interface configuration
  hosts: all
  become: yes
  vars:
    json_file_destination: "/home/cisco/config.json"
  tasks:
    - name: Generate JSON file from Jinja2 template
      template:
        src: "templates/interface_config.json.j2"
        dest: "{{ json_file_destination }}"
        mode: '0644’

    - name: Load config
      ansible.builtin.shell: config load /home/cisco/config.json -y
```

The cisco.sonic Ansible collection

- This collection includes Ansible core modules, and plugins to provision and manage devices running community SONiC.
- Readily available for use at https://galaxy.ansible.com/ui/repo/published/cisco/sonic/
- Current implementation uses network_cli connections (SSH based)

Name	Description	Connection Type
[sonic_command]	Run commands through the Management Framework CLI	network_cli
[sonic_config]	Run configuration commands through the Management Framework CLI	network_cli
[sonic_frr_command]	Run FRR show commands through the Management Framework CLI	network_cli
[sonic_frr_config]	Run FRR configurations through the Management Framework CLI	network_cli

cisco.sonic.collection SONiC command

```yaml
- name: Test custom command module
  hosts: all
  connection: network_cli
  gather_facts: false
  vars:
    FACTS_MODULES: ["cisco.sonic.sonic"]
  tasks:
    - name: Run command and capture output
      cisco.sonic.sonic_command:
        commands:
          - "show aaa"
        retries: 1
        interval: 1
      register: command_output_1

    - name: Run command and capture output 2
      ignore_errors: yes
      cisco.sonic.sonic_command:
        commands:
          - "show aaa"
          - "show platform summary"
        match: any
        wait_for:
          - result[0] contains AAA authentication login local
          - result[1] contains x86_64-8101-32FH-r0
      register: command_output_2  # Save the output for later use
    - name: Display the output from the first command
      debug:
        msg: "{{ command_output_1.stdout_lines }}"

    - name: Display the output from the second command
      debug:
        msg: "{{ command_output_2.stdout_lines }}"
```

Accepted values: “any” or “all”

```
root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25# ansible-playbook sonic_show.yaml -i routers.yaml

PLAY [Test custom command module] ***********************************************************************************************************************************

TASK [Run command and capture output] *******************************************************************************************************************************
ok: [m1]

TASK [Run command and capture output 2] *****************************************************************************************************************************
ok: [m1]

TASK [Display the output from the first command] *********************************************************************************************************************
ok: [m1] => {
    "msg": [
        [
            "AAA authentication login local (default)",
            "AAA authentication failthrough False (default)",
            "AAA authorization login local (default)",
            "AAA accounting login disable (default)"
        ]
    ]
}

TASK [Display the output from the second command] ********************************************************************************************************************
ok: [m1] => {
    "msg": [
        [
            "AAA authentication login local (default)",
            "AAA authentication failthrough False (default)",
            "AAA authorization login local (default)",
            "AAA accounting login disable (default)"
        ],
        [
            "Platform: x86_64-8102_64h_o-r0",
            "HwSKU: Cisco-8102-C64",
            "ASIC: cisco-8000",
            "ASIC Count: 1",
            "Serial Number: FLM28030AQ1",
            "Model Number: 8102-64H-O",
            "Hardware Revision: 1.0"
        ]
    ]
}

PLAY RECAP **********************************************************************************************************************************************************
m1                         : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25#
```

cisco.sonic.collection SONiC command

```yaml
- name: Test custom command module
  hosts: all
  connection: network_cli
  gather_facts: false
  vars:
    FACTS_MODULES: ["cisco.sonic.sonic"]
  tasks:
    - name: Run command and capture output
      cisco.sonic.sonic_command:
        commands:
          - "show aaa"
        retries: 1
        interval: 1
      register: command_output_1

    - name: Run command and capture output 2
      ignore_errors: yes
      cisco.sonic.sonic_command:
        commands:
          - "show aaa"
          - "show platform summary"
        match: all
        wait_for:
          - result[0] contains AAA authentication login local
          - result[1] contains x86_64-8101-32FH-r0
      register: command_output_2  # Save the output for later use
    - name: Display the output from the first command
      debug:
        msg: "{{ command_output_1.stdout_lines }}"

    - name: Display the output from the second command
      debug:
        msg: "{{ command_output_2.stdout_lines }}"
```

```
root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25# ansible-playbook sonic_show.yaml -i routers.yaml

PLAY [Test custom command module] ***********************************************************************************************************************************

TASK [Run command and capture output] *******************************************************************************************************************************
ok: [m1]

TASK [Run command and capture output 2] *****************************************************************************************************************************
fatal: [m1]: FAILED! => {"changed": false, "failed_conditions": ["result[1] contains x86_64-8101-32FH-r0"], "msg": "One or more conditional statements have
not been satisfied"}

PLAY RECAP **********************************************************************************************************************************************************
m1                         : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

cisco.sonic.collection SONiC config

```
- name: Test custom config module
  hosts: all
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Add VLANs globally
      ignore_errors: yes
      cisco.sonic.sonic_config:
        all_or_none: 'all'
        lines:
          - "vlan add 10"
          - "vlan add 20"
          - "vlan asd 30"
          - "vlan member add 10 -u Ethernet252"

    - name: Asign interfaces to VLAN
      ignore_errors: yes
      cisco.sonic.sonic_config:
        all_or_none: 'none'
        lines:
          - "vlan member add 20 -u Ethernet252"
          - "vlan member add 20 -u Ethernet248"
```

Accepted values: “all” or “none”

```
root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25# ansible-playbook sonic_config.yaml -i routers.yaml

PLAY [Test custom config module] ***********************************************

TASK [Add VLANs globally] ******************************************************
fatal: [m1]: FAILED! => {"changed": true, "commands": ["sudo config vlan add 10", "sudo config vlan add 20", "sudo config vlan asd 30", "sudo config vlan member add 10 -u Ethernet252"], "failed_cmds": ["sudo config vlan asd 30"], "passed_cmds": ["sudo config vlan add 10", "sudo config vlan add 20", "sudo config vlan member add 10 -u Ethernet252"], "stderr": ["Usage: config vlan [OPTIONS] COMMAND [ARGS]...\nTry \"config vlan -h\" for help.\n\nError: No such command \"asd\"."], "stderr_lines": [["Usage: config vlan [OPTIONS] COMMAND [ARGS]...", "Try \"config vlan -h\" for help.", "", "Error: No such command \"asd\"."]], "stdout": ["", "", ""], "stdout_lines": [[""], [""], [""]], "updates": ["sudo config vlan add 10", "sudo config vlan add 20", "sudo config vlan member add 10 -u Ethernet252"]}
...ignoring

TASK [Asign interfaces to VLAN] ************************************************
fatal: [m1]: FAILED! => {"changed": false, "failed_cmds": ["Usage: config vlan member add [OPTIONS] <vid> port\nTry \"config vlan member add -h\" for help.\n\nError: Ethernet252 is already untagged member!"], "msg": "Command in playbook failed. Rollback initiated. Device state back to pre-task state. Rollback complete."}
...ignoring

PLAY RECAP *********************************************************************
m1                         : ok=2    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=2
```

```
root@m1:/home/cisco# show vlan brief
+-----------+--------------+-------------+----------------+-------------+--------------------------+-----------------------+
|   VLAN ID | IP Address   | Ports       | Port Tagging   | Proxy ARP   | Static Anycast Gateway   | DHCP Helper Address   |
+===========+==============+=============+================+=============+==========================+=======================+
|        10 |              | Ethernet252 | untagged       | disabled    | disabled                 |                       |
+-----------+--------------+-------------+----------------+-------------+--------------------------+-----------------------+
|        20 |              |             |                | disabled    | disabled                 |                       |
+-----------+--------------+-------------+----------------+-------------+--------------------------+-----------------------+
root@m1:/home/cisco#
```

Setting this to none will initiate a config rollback in case of failed commands

cisco.sonic.collection SONiC config from template

```
tacacs authtype {{ var }}
{% if var == "pap" %}
tacacs timeout 10
{% else %}
tacacs timeout 5
{% endif %}
```


```
- name: Test custom config module
  hosts: all
  connection: network_cli
  tasks:
    - name: Set variable for TACACS+ auth type
      ansible.builtin.set_fact:
        var: "pap"
    - name: Run file with var input to apply TACACS+ configuration
      ignore_errors: yes
      cisco.sonic.sonic_config:
        src: templates/j2file.in
```

```
root@m1:/home/cisco# show tacacs
TACPLUS global auth_type pap
TACPLUS global timeout 10
TACPLUS global passkey <EMPTY_STRING> (default)

root@m1:/home/cisco#
```

cisco.sonic.collection FRR command

```
- name: Test custom config module
  hosts: all
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Show BGP summary
      ignore_errors: yes
      cisco.sonic.sonic_frr_command:
        commands:
          - "show router-id"
          - "show bgp summary"
        retries: 2
        wait_for:
          - result[0] contains 2.2.2.2
          - result[1] contains 1.18.170.156
        match: all
      register: bgp_summary_output

    - name: Print BGP summary output to screen
      debug:
        msg: "{{ bgp_summary_output.stdout_lines }}"
```

cisco.sonic.collection FRR command

```
root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25# ansible-playbook frr_show.yaml -i bgp.yaml

PLAY [Test custom config module] ***********************************************************************************************************************************************************************

TASK [Show BGP summary] ********************************************************************************************************************************************************************************
ok: [c2]

TASK [Print BGP summary output to screen] **************************************************************************************************************************************************************
ok: [c2] => {
    "msg": [
        [
            "zebra:",
            "     router-id 2.2.2.2 vrf default"
        ],
        [
            "IPv4 Unicast Summary (VRF default):",
            "BGP router identifier 2.2.2.2, local AS number 4200112195 vrf-id 0",
            "BGP table version 1006008",
            "RIB entries 7, using 1568 bytes of memory",
            "Peers 2, using 41 KiB of memory",
            "",
            "Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc",
            "1.18.170.156    4 4200112195      5464       844        0    0    0 08w2d14h       Active        0 N/A",
            "4.2.0.4         4 4200129999         0         0        0    0    0    never      Connect        0 N/A",
            "",
            "Total number of neighbors 2",
            "",
            "L2VPN EVPN Summary (VRF default):",
            "BGP router identifier 2.2.2.2, local AS number 4200112195 vrf-id 0",
            "BGP table version 0",
            "RIB entries 2003, using 438 KiB of memory",
            "Peers 2, using 41 KiB of memory",
            "",
            "Neighbor        V         AS   MsgRcvd   MsgSent   TblVer  InQ OutQ  Up/Down State/PfxRcd   PfxSnt Desc",
            "1.18.170.156    4 4200112195      5464       844        0    0    0 08w2d14h       Active        0 N/A",
            "4.2.0.4         4 4200129999         0         0        0    0    0    never      Connect        0 N/A",
            "",
            "Total number of neighbors 2"
        ]
    ]
}

PLAY RECAP *********************************************************************************************************************************************************************************
c2                         : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25#
```

cisco.sonic.collection FRR config

```
- name: Test custom config module
  hosts: all
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Start BGP process
      ignore_errors: yes
      cisco.sonic.sonic_frr_config:
        lines:
          - "router bgp 64000"

    - name: Add BGP neighbor
      ignore_errors: yes
      cisco.sonic.sonic_frr_config:
        lines:
          - "neighbor 192.1.22.4 remote-as 64000"
        parents: ["router bgp 64000"]

    - name: Set Keepalive
      ignore_errors: yes
      cisco.sonic.sonic_frr_config:
        lines:
          - "bgp tcp-keepalive 5 5 5"
        parents: ["router bgp"]
```

cisco.sonic.collection FRR config from template

```
router bgp {{ as }}
 bgp tcp-keepalive 5 5 5
 neighbor {{ neighbor_id }} remote-as {{ remote_as }}
```

```
- name: Test frr config module with j2 template
  hosts: all
  gather_facts: false
  connection: network_cli
  tasks:
    - name: Set BGP AS
      ansible.builtin.set_fact:
        bgp_as: "64000"
    - name: Set neighbor
      ansible.builtin.set_fact:
        neighbor_id: "192.1.22.4"
    - name: Set remote AS
      ansible.builtin.set_fact:
        remote_as: "64000"
    - name: FRR Config Test 1
      ignore_errors: yes
      cisco.sonic.sonic_frr_config:
        src: templates/j2_frr.in
```

```
root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25# ansible-playbook frr_template_config.yaml -i routers.yaml

PLAY [Test frr config module with j2 template] *********************************************************************************************************************************************

TASK [Set BGP AS] **************************************************************************************************************************************************************************
ok: [m1]

TASK [Set neighbor] ************************************************************************************************************************************************************************
ok: [m1]

TASK [Set remote AS] ***********************************************************************************************************************************************************************
ok: [m1]

TASK [FRR Config Test 1] *******************************************************************************************************************************************************************
changed: [m1]

PLAY RECAP *********************************************************************************************************************************************************************************
m1                         : ok=4    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0

root@8ktme:/home/cisco/suhahmad/tools/sonic/clus25#
```

Conclusion

Conclusion

- SONiC has multiple modes of configuration which makes it critical to automate configuration workflows using consistent mechanisms
- SONiC's Linux native infrastructure makes Ansible a natural choice for building automation pipelines
- Users can leverage SONiC's support for CLI and JSON files for configuring network devices using built-in Ansible modules. Additionally, Jinja2 templating makes it easier to scale configuration automation.
- The cisco.sonic Ansible collection provides direct access to CLI commands in both SONiC and FRR along with error handling options such as "wait_for" and "all_or_none".
- Future work for this collection includes:
  - Additional transport mechanism such as gNMI or REST
  - Defined API list corresponding to different configuration actions (e.g., BGP, ACL, QoS, etc.)
  - Sample playbooks and roles targeted towards different use-cases (e.g., IP-BGP fabric, EVPN/VXLAN fabric)

Complete your session evaluations

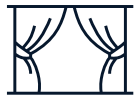
Complete a minimum of 4 session surveys and the Overall Event Survey to be entered in a drawing to win 1 of 5 full conference passes to Cisco Live 2026.

Earn 100 points per survey completed and compete on the Cisco Live Challenge leaderboard.

Level up and earn exclusive prizes!

Complete your surveys in the Cisco Live mobile app.

Continue your education



Visit the Cisco Showcase for related demos

Book your one-on-one Meet the Engineer meeting

Attend the interactive education with DevNet, Capture the Flag, and Walk-in Labs

Visit the On-Demand Library for more sessions at www.CiscoLive.com/on-demand

Contact me at: suhahmad@cisco.com

Thank you

cisco Live !


CISCO