Management and Operations of Networks, Services, and Systems

Device API and Automation Tools

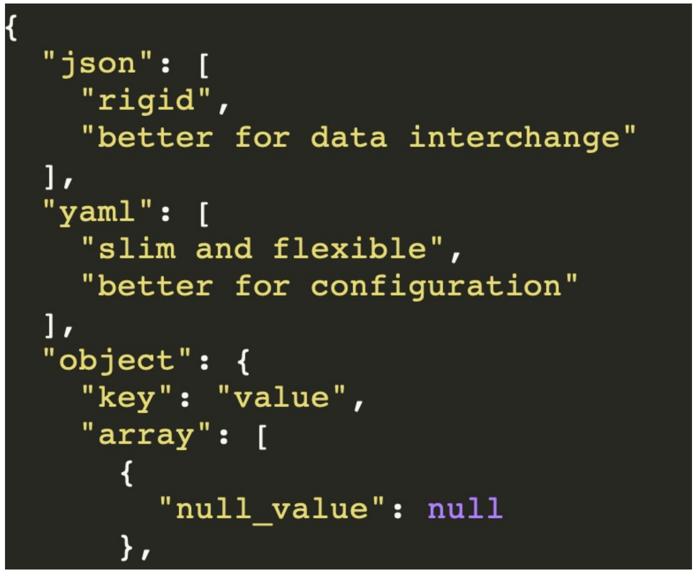
Ricardo Morla

FEUP – GORS/M.EEC, GRS/M.EIC

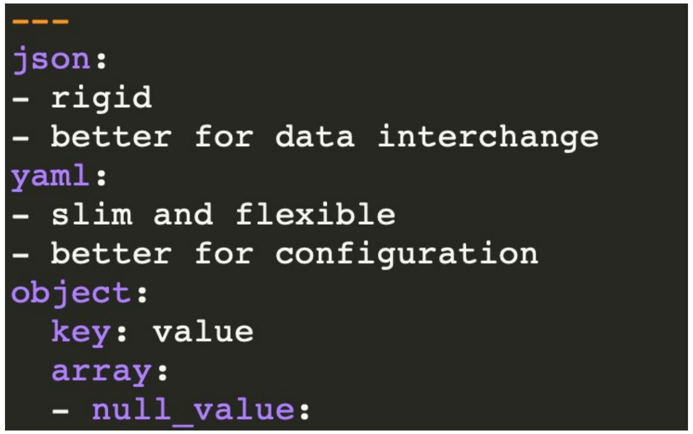
Data formats

- JSON, YAML, XML

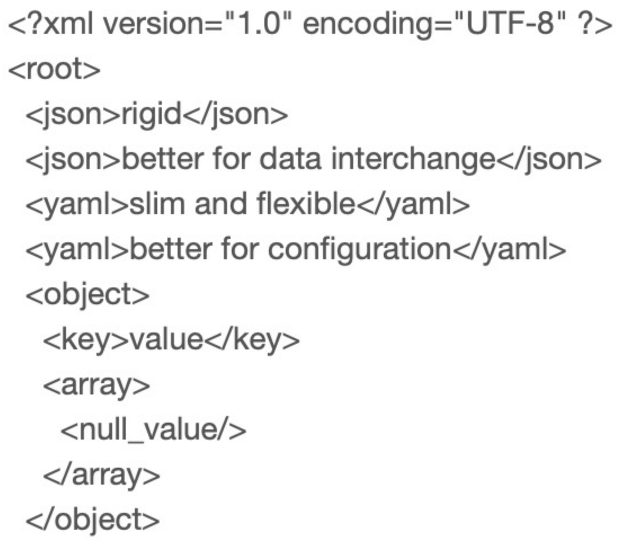
- Types of Objects
  - String, integer, boolean
  - Lists and dictionaries



```json
{
  "json": [
    "rigid",
    "better for data interchange"
  ],
  "yaml": [
    "slim and flexible",
    "better for configuration"
  ],
  "object": {
    "key": "value",
    "array": [
      {
        "null_value": null
      },
```



```yaml
---
json:
- rigid
- better for data interchange
yaml:
- slim and flexible
- better for configuration
object:
  key: value
  array:
  - null_value:
```



```xml
<?xml version="1.0" encoding="UTF-8" ?>
<root>
  <json>rigid</json>
  <json>better for data interchange</json>
  <yaml>slim and flexible</yaml>
  <yaml>better for configuration</yaml>
  <object>
    <key>value</key>
    <array>
      <null_value/>
    </array>
  </object>
```

YANG, data modeling language

- Defines a structure for the data
- Building blocks:
  - **module** – top of the hierarchy of nodes
  - **containers** – related nodes
  - **lists** – identifies nodes
  - **leaf** – individual attributes of a node
  - **type** – every leaf has an associated type

https://en.wikipedia.org/wiki/YANG

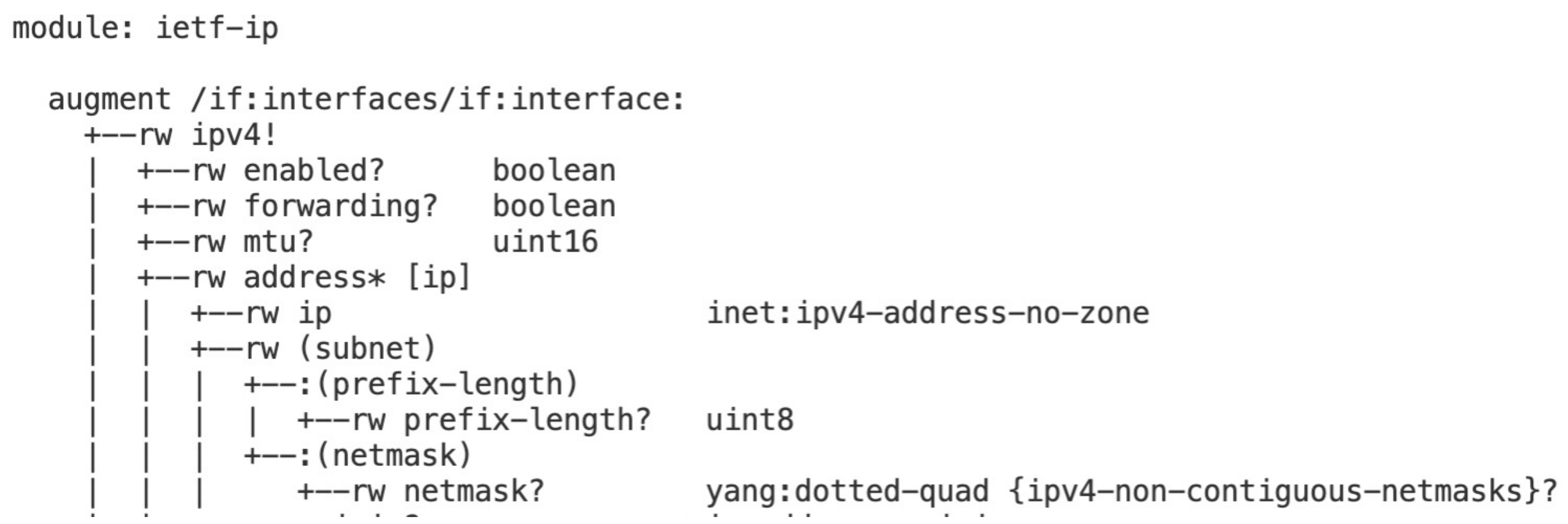
```
list person {
  key name;
  leaf name { type string; }
  leaf birthday { type yang:date-and-time; mandatory true; }
}
```

```
<person>
  <name>Cristiano Ronaldo</name>
  <birthday>1985-02-05T00:00:00-00:00</birthday>
</person>
```

YANG for networking

https://github.com/YangModels/yang

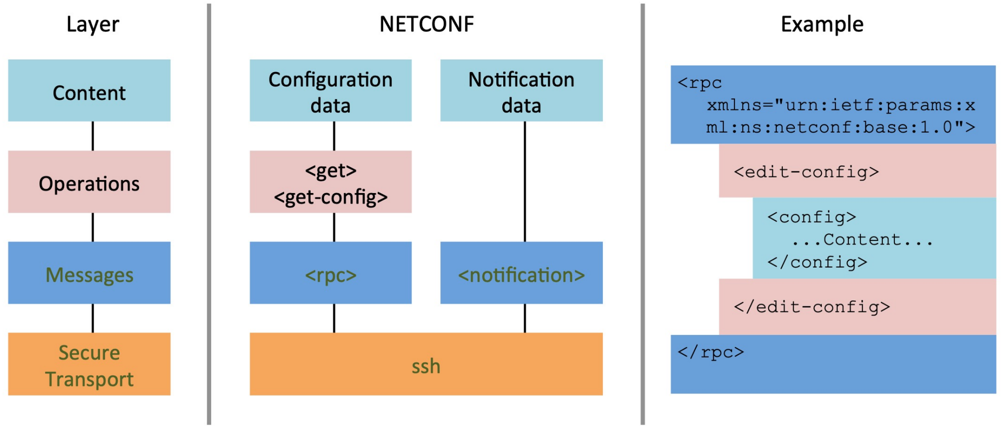
- pyang -f tree ietf-ip.yang



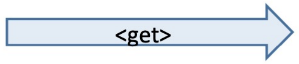
```
module: ietf-ip

  augment /if:interfaces/if:interface:
    +--rw ipv4!
    |  +--rw enabled?      boolean
    |  +--rw forwarding?   boolean
    |  +--rw mtu?          uint16
    |  +--rw address* [ip]
    |  |  +--rw ip                    inet:ipv4-address-no-zone
    |  |  +--rw (subnet)
    |  |  |  +--:(prefix-length)
    |  |  |  |  +--rw prefix-length?   uint8
    |  |  |  +--:(netmask)
    |  |  |     +--rw netmask?         yang:dotted-quad {ipv4-non-contiguous-netmasks}?
```

netconf RFC6241

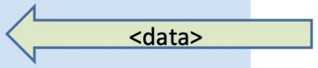
https://trac.ietf.org/trac/netconf/wiki



```
<rpc
  xmlns="urn:ietf:params:x
  ml:ns:netconf:base:1.0">
    <edit-config>
      <config>
        ...Content...
      </config>
    </edit-config>
</rpc>
```

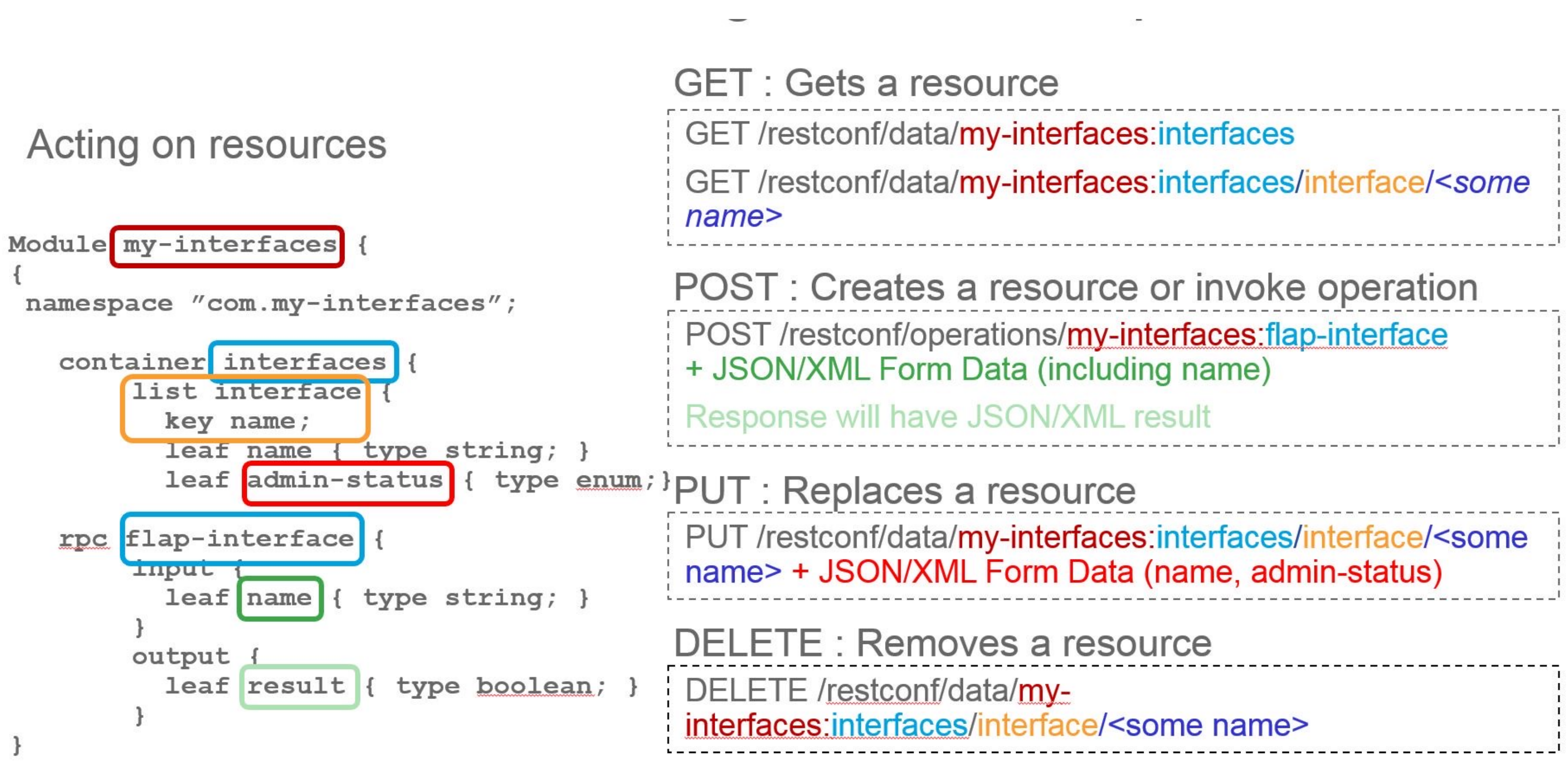


```
<rpc message-id="1"
xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
   <get/>
</rpc>
```



```
<rpc-reply message-id="1" xmlns="urn:ietf:params:xml:ns:netconf:base:1.0">
  <data>
    <interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
      <interface>
        <name>eth0</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:1::3</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
      </interface>
      <interface>
        <name>eth1</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
          <address>
            <ip>2001:db8:c18:2::1</ip>
            <prefix-length>128</prefix-length>
          </address>
        </ipv6>
      </interface>
    </interfaces>
  </data>
</rpc-reply>
```

restconf
RFC8040



Acting on resources

```
Module my-interfaces {
{
 namespace "com.my-interfaces";

    container interfaces {
         list interface {
           key name;
           leaf name { type string; }
           leaf admin-status { type enum;}

    rpc flap-interface {
         input {
           leaf name { type string; }
         }
         output {
           leaf result { type boolean; }
         }
}
```

GET : Gets a resource

GET /restconf/data/my-interfaces:interfaces

GET /restconf/data/my-interfaces:interfaces/interface/<*some name*>

POST : Creates a resource or invoke operation

POST /restconf/operations/my-interfaces:flap-interface
+ JSON/XML Form Data (including name)

Response will have JSON/XML result

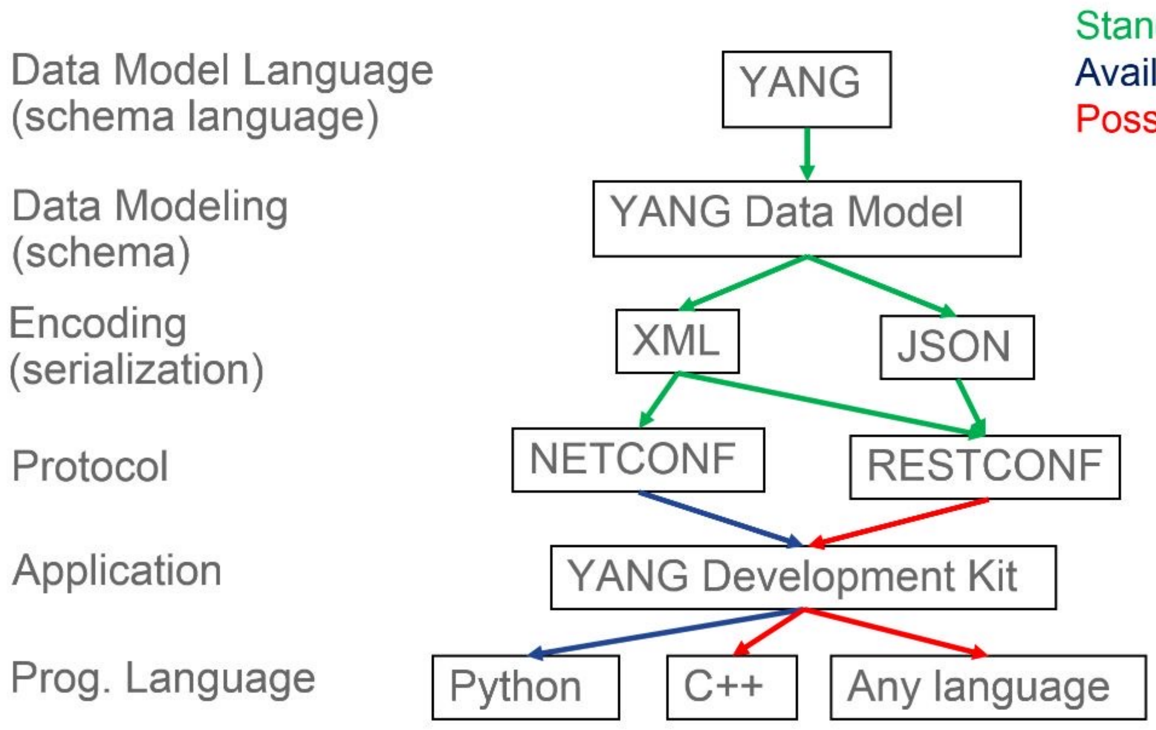
PUT : Replaces a resource

PUT /restconf/data/my-interfaces:interfaces/interface/<some name> + JSON/XML Form Data (name, admin-status)

DELETE : Removes a resource

DELETE /restconf/data/my-interfaces:interfaces/interface/<some name>

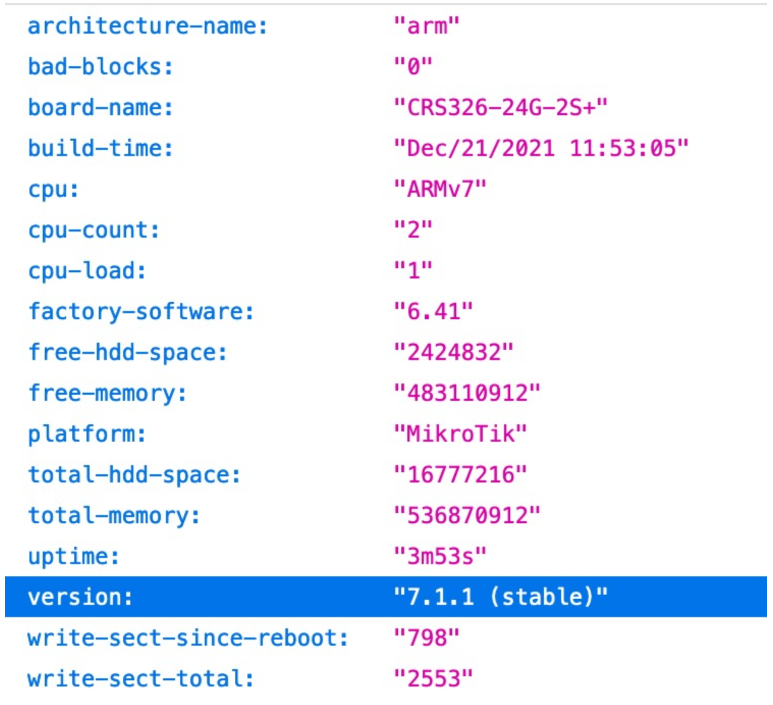
Birds'eye view



Mikrotik REST API

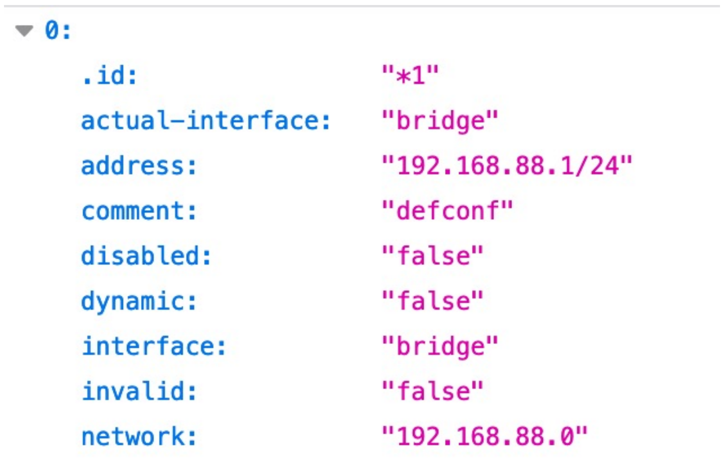
https://help.mikrotik.com/docs/display/ROS/REST+API

curl -k -u USER:PWD https://192.168.88.1/rest/system/resource



```
architecture-name:          "arm"
bad-blocks:                 "0"
board-name:                 "CRS326-24G-2S+"
build-time:                 "Dec/21/2021 11:53:05"
cpu:                        "ARMv7"
cpu-count:                  "2"
cpu-load:                   "1"
factory-software:           "6.41"
free-hdd-space:             "2424832"
free-memory:                "483110912"
platform:                   "MikroTik"
total-hdd-space:            "16777216"
total-memory:               "536870912"
uptime:                     "3m53s"
version:                    "7.1.1 (stable)"
write-sect-since-reboot:    "798"
write-sect-total:           "2553"
```

curl -k -u USER:PWD https://192.168.88.1/rest/ip/address

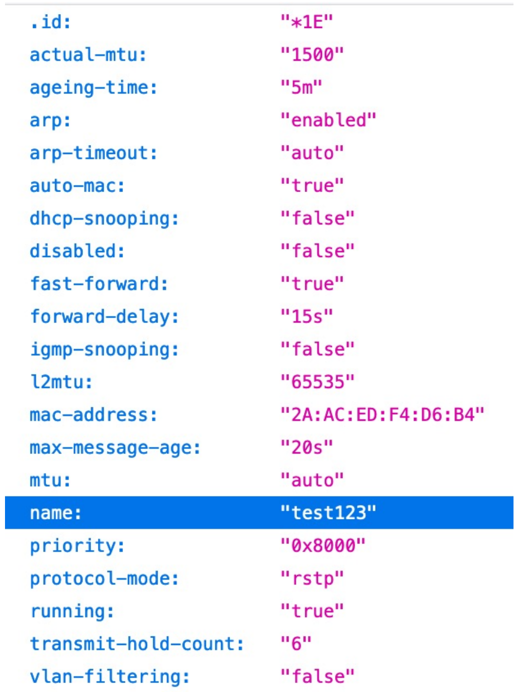


```
▼ 0:
    .id:                  "*1"
    actual-interface:     "bridge"
    address:              "192.168.88.1/24"
    comment:              "defconf"
    disabled:             "false"
    dynamic:              "false"
    interface:            "bridge"
    invalid:              "false"
    network:              "192.168.88.0"
```

Add bridge, set IP address

curl -k -u USER:PWD -X PUT https://192.168.88.1/rest/interface/bridge --data '{"name": "**test123**"}' -H "content-type: application/json"

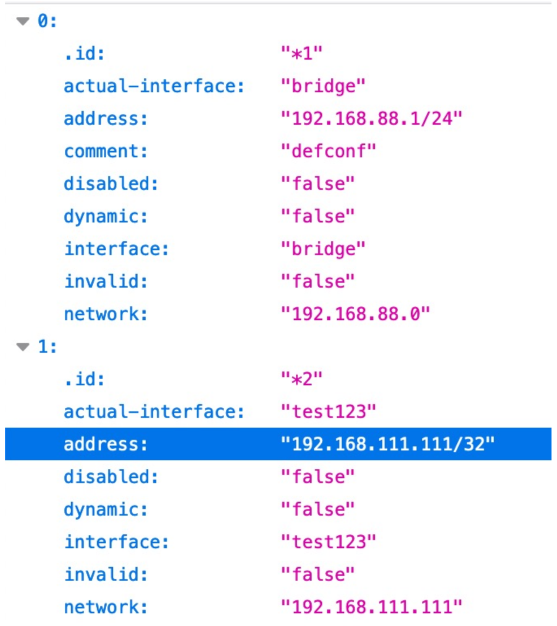
https://192.168.88.1/rest/interface/bridge/test123



```
.id:                  "*1E"
actual-mtu:           "1500"
ageing-time:          "5m"
arp:                  "enabled"
arp-timeout:          "auto"
auto-mac:             "true"
dhcp-snooping:        "false"
disabled:             "false"
fast-forward:         "true"
forward-delay:        "15s"
igmp-snooping:        "false"
l2mtu:                "65535"
mac-address:          "2A:AC:ED:F4:D6:B4"
max-message-age:      "20s"
mtu:                  "auto"
name:                 "test123"
priority:             "0x8000"
protocol-mode:        "rstp"
running:              "true"
transmit-hold-count:  "6"
vlan-filtering:       "false"
```

#curl -k -u USER:PWD -X PUT https://192.168.88.1/rest/ip/address --data '{"address": "**192.168.111.111**", "interface": "test123"}' -H "content-type: application/json"

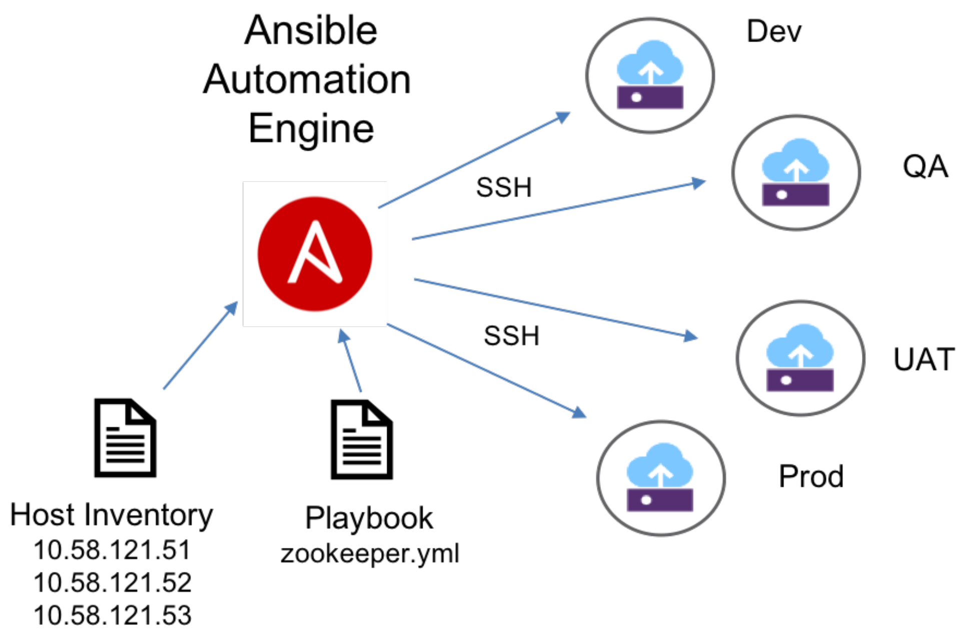
https://192.168.88.1/rest/ip/address



```
▼ 0:
    .id:                "*1"
    actual-interface:   "bridge"
    address:            "192.168.88.1/24"
    comment:            "defconf"
    disabled:           "false"
    dynamic:            "false"
    interface:          "bridge"
    invalid:            "false"
    network:            "192.168.88.0"
▼ 1:
    .id:                "*2"
    actual-interface:   "test123"
    address:            "192.168.111.111/32"
    disabled:           "false"
    dynamic:            "false"
    interface:          "test123"
    invalid:            "false"
    network:            "192.168.111.111"
```

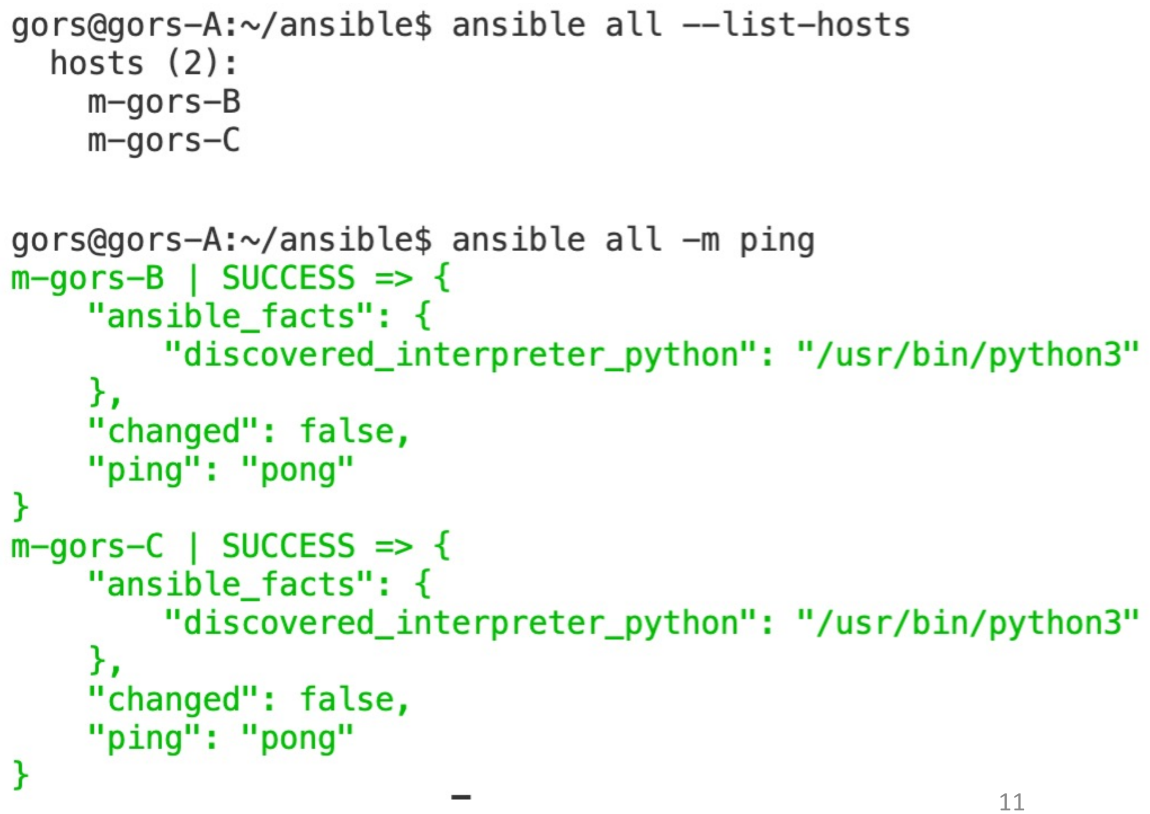
Automation tools

- Ansible, Salt, Puppet, Chef
- Host inventory
- Playbook



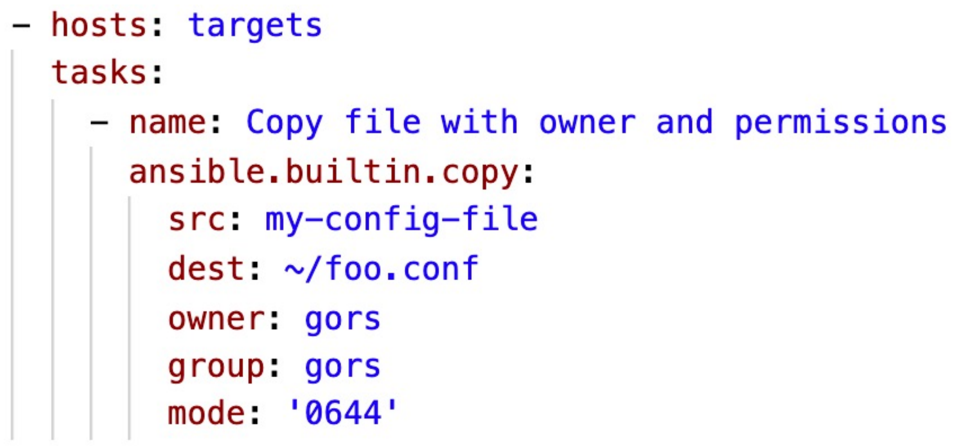
Ansible quick start

- On the config host
  - `apt install ansible`
  - ansible.cfg
    - [defaults]
    - inventory = /home/gors/ansible/hosts
  - hosts
    - [targets]
    - m-gors-B
    - m-gors-C

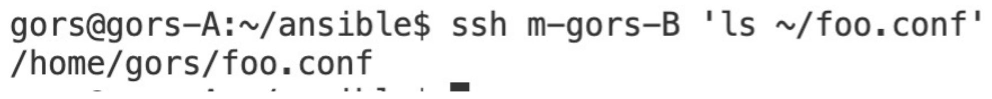


```
gors@gors-A:~/ansible$ ansible all --list-hosts
  hosts (2):
    m-gors-B
    m-gors-C

gors@gors-A:~/ansible$ ansible all -m ping
m-gors-B | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
m-gors-C | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": false,
    "ping": "pong"
}
```

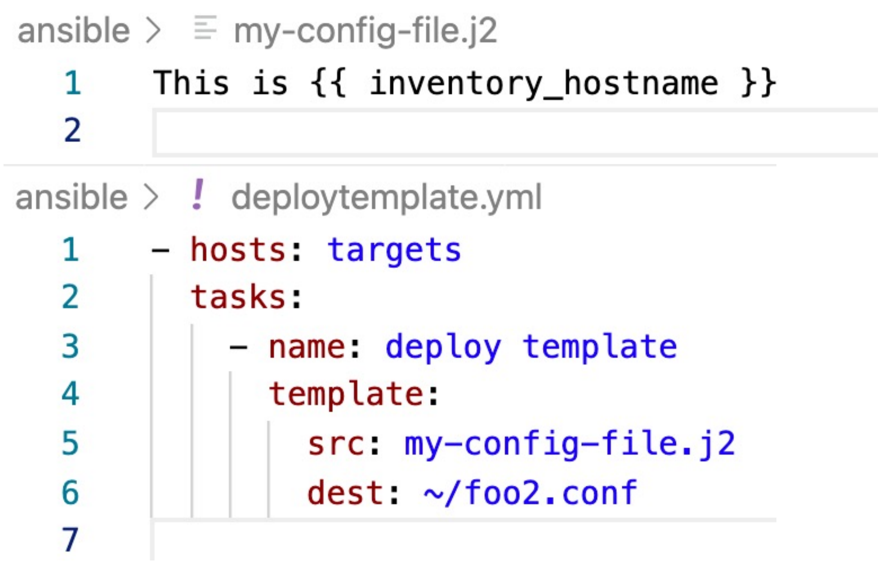
Ansible upload configuration files



```yaml
- hosts: targets
  tasks:
    - name: Copy file with owner and permissions
      ansible.builtin.copy:
        src: my-config-file
        dest: ~/foo.conf
        owner: gors
        group: gors
        mode: '0644'
```

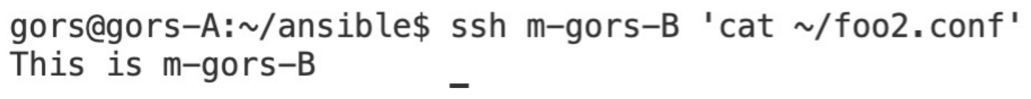


```
gors@gors-A:~/ansible$ ssh m-gors-B 'ls ~/foo.conf'
/home/gors/foo.conf
```



```
This is {{ inventory_hostname }}

```

```yaml
- hosts: targets
  tasks:
    - name: deploy template
      template:
        src: my-config-file.j2
        dest: ~/foo2.conf

```



```
gors@gors-A:~/ansible$ ssh m-gors-B 'cat ~/foo2.conf'
This is m-gors-B
```

Ansible napalm

- pip install napalm-ansible
- pip install napalm-ros
- ansible.cfg
    - [defaults]
        - library = LIBHOME/napalm_ansible/:LIBHOME/napalm_ros/

```
- hosts: switches
  connection: local
  gather_facts: no
  tasks:
  - name: get facts from device
    napalm_get_facts:
        hostname: "{{ inventory_hostname }}"
        username: "USER"
        dev_os: "ros"
        password: "PWD"
        filter: "facts"
    register: result
  - name: print results
    debug: msg="{{ result }}"
```

```
TASK [print results] *********************************************************************
ok: [m-sw1] => {
    "msg": {
        "ansible_facts": {
            "discovered_interpreter_python": "/usr/bin/python3",
            "napalm_facts": {
                "fqdn": "",
                "hostname": "MikroTik",
                "interface_list": [
                    "br-client",
                    "br-server",
                    "bridge",
                    "ether1",
                    "ether2",
```

Management and Operations of Networks, Services, and Systems

`Device API and Automation Tools`

Ricardo Morla

FEUP – GORS/M.EEC, GRS/M.EIC