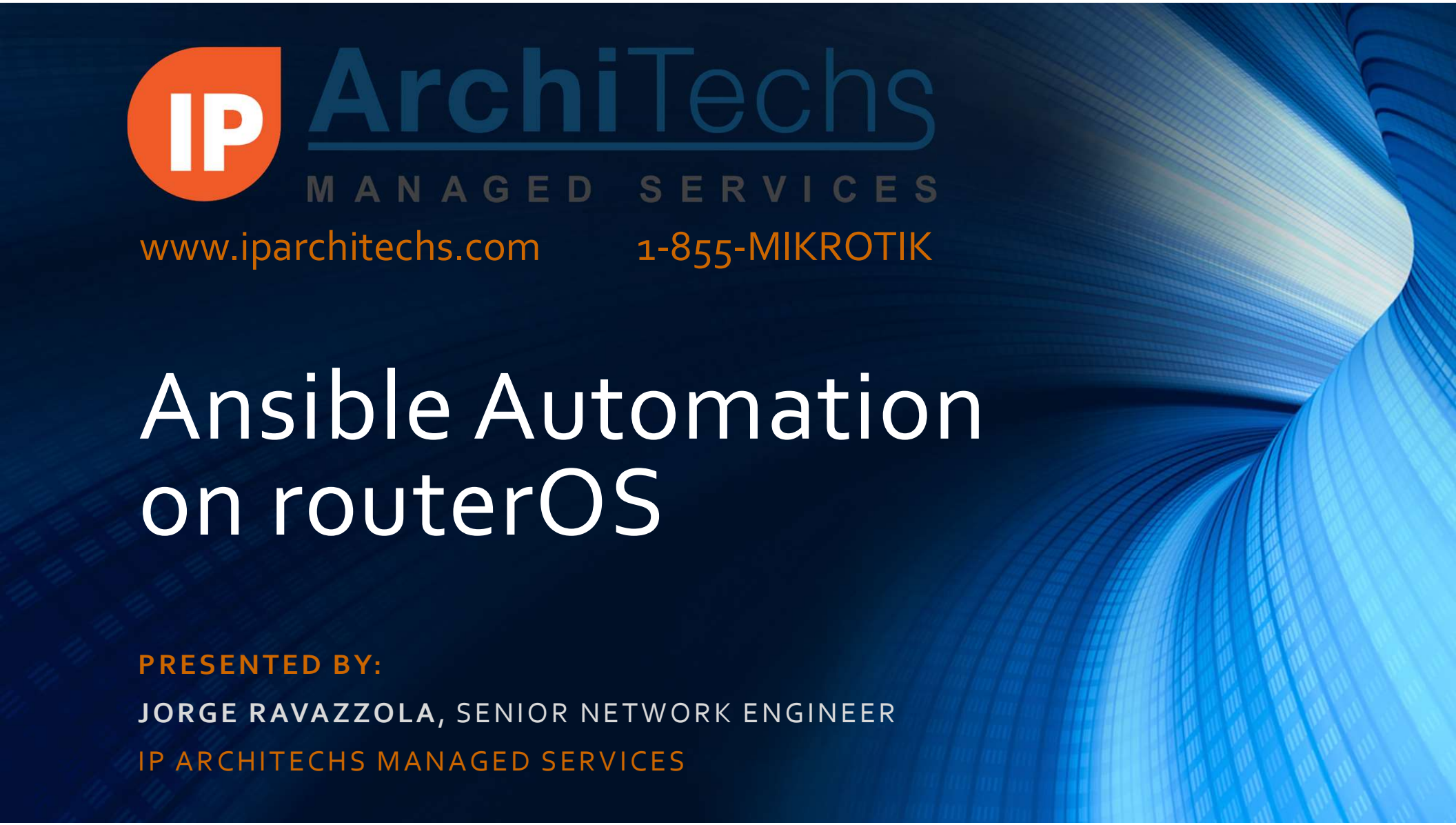

IP
ArchiTechs
MANAGED SERVICES
www.iparchitechs.com
1-855-MIKROTIK
Ansible Automation on routerOS
PRESENTED BY:
JORGE RAVAZZOLA, SENIOR NETWORK ENGINEER
IP ARCHITECHS MANAGED SERVICES

Background

- **Jorge Ravazzola**
  - 10 + years in IT/Network Engineering
  - Designed and implemented networks in Service Provider, and Power distribution grid environments
  - Areas of Design Focus:
    - Juniper/MikroTik/Cisco integration
    - Design of BGP/MPLS/OSPF Service Provider Triple-Play networks
    - Design of Enterprise Data Center networks
  - Certifications
    - Pursuing JNCIE-SP
    - Certified – JNCIP-SP,JNCIS-SP, JNCIA

IP ArchiTechs MANAGED SERVICES

1-855-MIKROTIK

www.iparchitechs.com

IP ArchiTechs Managed Services

- Exhibitor at 2019 MUM –
- The first Carrier-Grade 24/7/365 MikroTik TAC (Technical Assistance Center)
  - Three tiers of engineering support
  - Monthly and per incident pricing available
  - 1-855-MIKROTIK or support.iparchitechs.com
- Private Nationwide 4G LTE MPLS backbone
  - Partnership with Verizon Wireless - available anywhere in the Verizon service area
  - Not Internet facing – privately routed over our MPLS infrastructure
  - Point-to-Point or Point-to-MultiPoint
- Proactive Monitoring / Ticketing / Change Control / IPAM (IP Address Management)
- Carrier-Grade Network Engineering / Design in large (10,000+ nodes) environments

24/7/365 MikroTik TAC **Nationwide Private 4G LTE MPLS** **Proactive Network Monitoring** **Design / Engineering / Operations**

Introduction – About Ansible

- It's a Framework
  - Based on Python
  - Designed to simplify and automate massive operation on servers, now becoming popular for networking automation.
  - Recently purchased by Red Hat, but open source version is available and maintained by ansible community
- Some Features
  - It's agent-less. Ansible does not need any specific piece of software or code running in the network device.
  - It's idempotent. Does not run any modification if not necessary.

Why would a network engineer use ansible?

- Relatively simple introduction to coding for network devices
- Code Re-use is encouraged among the community
- No need to learn Python (at least at the beginning)
- It's open source and run in almost any OS.
- It's multi vendor.
- It allows progressive learning and progressive shift to automation
  - Use cases: From simple to complex.
  - Specific modules can be created as needed
  - It is a tool. There are other similar frameworks as well.

How does it work anyway?

- Ansible processes text-coded files named playlists
  - Playlists are coded in YAML format (---)
  - Playlists are made of ,well, plays
- Each play is a list of tasks.
- Each task can use one or more modules, written in Python
- A task can create a configuration file based on a given template
- Templates are written in Jinja2 format .j2

A quick Example: Check OS Version

```
---

- name: "Check OS version"
  hosts: mum-r1
  gather_facts: no


  tasks:
    - name: Check OS version on device mum-r1
      routeros_command:
        commands: /system package print
      register: version_output

    - name: Display facts variables
      debug:
        var: version_output

...
```

- YAML files start with --- and end with …
- This playbook has a single play called "Check OS version"
- The play is applied to "mum-r1" device hardcoded in the playlist
- The play has two tasks with its own names
- First task uses two modules: routeros_command and register
- Second task uses debug module to display output.
- All these modules were re-used and are available on ansible open-source distribution

Can "Check OS Version" scale?

```
---

- name: "Check OS version"
  hosts: MUM-routers
  gather_facts: no


  tasks:
    - name: Check OS version from Mktik devices
      routeros_command:
        commands: /system package print
      register: version_output

    - name: Display facts variables
      debug:
        var: version_output

...
```

- Hardcoding does not scale, nor is it a good coding practice
- Check out for "hosts" value at play statement.
- Rest of the code is still the same
- By changing hosts from a hardcoded value to a group value we run the same code in several devices
- Devices are listed in a separate file independent from code itself

So, How does Ansible sort all this out?

```
[defaults]
inventory = ./hosts
host_key_checking = False
timeout = 5
log_path = ./ansible.log
```

```
[MUM-routers]
mum-r[1:4]
```

- Directory structure is fundamental
  - /etc/ansible/ansible.cfg is the default config file, can be overridden by local config file
  - Local "hosts" file can be specified at local config file
  - Inventory is the key concept here
    - Ansible can check on its inventory file to know to which devices the tasks must be applied on
    - Inventory files can include specific host related info and/or groups related info
    - where's the rest of the info?
    - Keep in mind that duplicating code is not a good practice, nor does it scale

Enter host_vars and group_vars directory:

```
root@NetworkAutomation-1:~/ansible-MUM# ls ./host_vars
mum-r1  mum-r2  mum-r3  mum-r4
root@NetworkAutomation-1:~/ansible-MUM# ls ./group_vars
MUM-routers
root@NetworkAutomation-1:~/ansible-MUM# cat ./hosts


[MUM-routers]
mum-r[1:4]
root@NetworkAutomation-1:~/ansible-MUM# cat ./group_vars/MUM-routers
---
ansible_user: admin
ansible_ssh_pass: admin
ansible_connection: network_cli
ansible_network_os: routeros

root@NetworkAutomation-1:~/ansible-MUM# cat ./host_vars/mum-r1
---
ansible_host: 192.168.122.253
root@NetworkAutomation-1:~/ansible-MUM# cat ./host_vars/mum-r2
---
ansible_host: 192.168.122.214
root@NetworkAutomation-1:~/ansible-MUM#
```

- (Local) Directory structure is (still) fundamental
  - Ansible looks into these two (sub) directories in order to find the information common to a group or a specific host listed on inventory
  - Main concept is to avoid duplicating data
    - These directories contain specific host files and group related files
    - Each file contains info related to a specific host or common information related to a specific group respectively
    - Keep in mind that duplicating code is not a good practice, nor does it scale

Ok, we have a start of scalable code…can we re-use?

```
- name: "Get ARP status"
  hosts: MUM-routers
  gather_facts: no


  tasks:
    - name: Include Check arp code
      include: ./check_arp_status.yml
```

```
---

#- name: "Check ARP Status"
#  hosts: MUM-routers
#  gather_facts: no


#  tasks:
    - name: Check ARP status from Mktik devices
      routeros_command:
        commands: /ip arp print
      register: arp_output

    - name: Display facts variables
      debug:
        var: arp_output

...
```

- Several ways
  - Already re-using: By using Ansible modules
  - Generate hosts and var structures that can allow you to use them in all your playbooks
  - Several vendors include their own roles
  - Use include: module
    - Re-use code must be a set of tasks listed in a YAML file

Use Cases: Provisioning, configuring, monitoring

- Provisioning: Tunnels , queues, circuits etc.
- Configuration file generation for massive deployment using templates to create

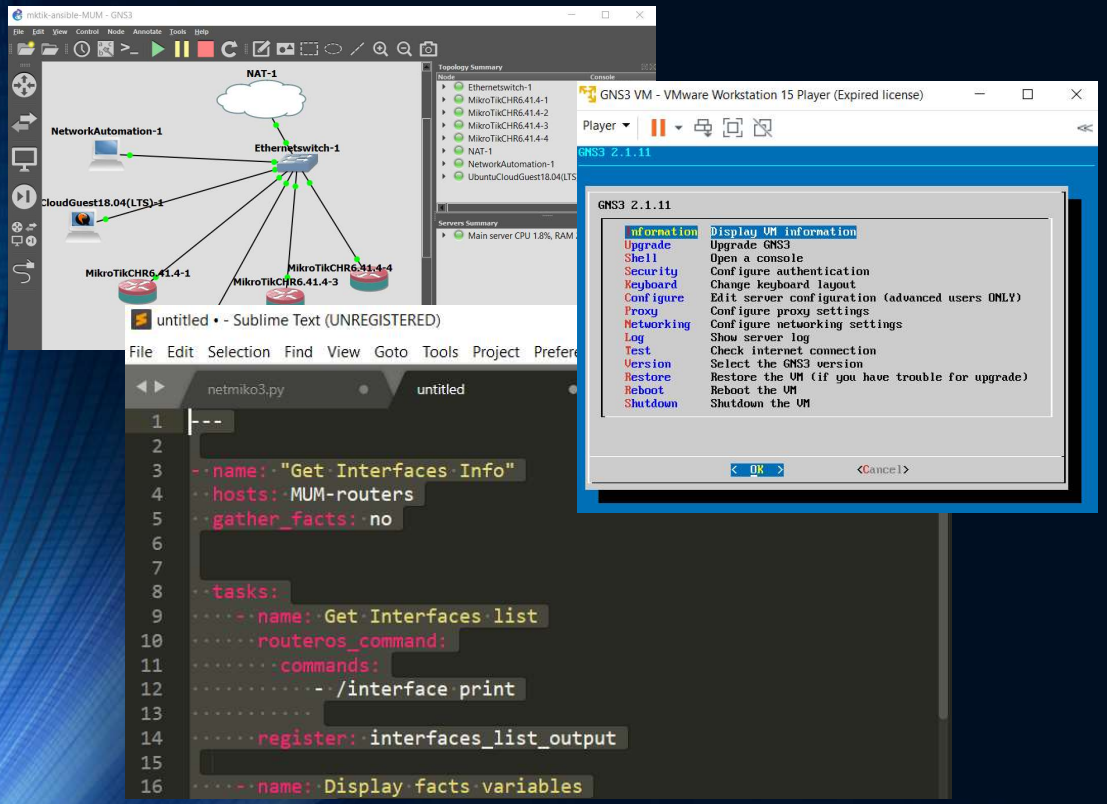
  the corresponding file for each specific device
- Traffic , SNR, Power, CPU etc .monitoring
- Key concept: avoid repetitive tasks and save time.

Further steps: templates, roles, modules

```
roles/
├── myfirewall
│   ├── defaults
│   │   └── main.yml
│   ├── handlers
│   │   └── main.yml
│   └── tasks
│       └── main.yml
└── myvhost
    ├── files
    │   └── html
    │       └── index.html
    ├── handlers
    │   ├── main.yml
    │   └── maun.yml
    ├── meta
    │   └── main.yml
    ├── tasks
    │   └── main.yml
    └── templates
        └── vhost.conf.j2
```

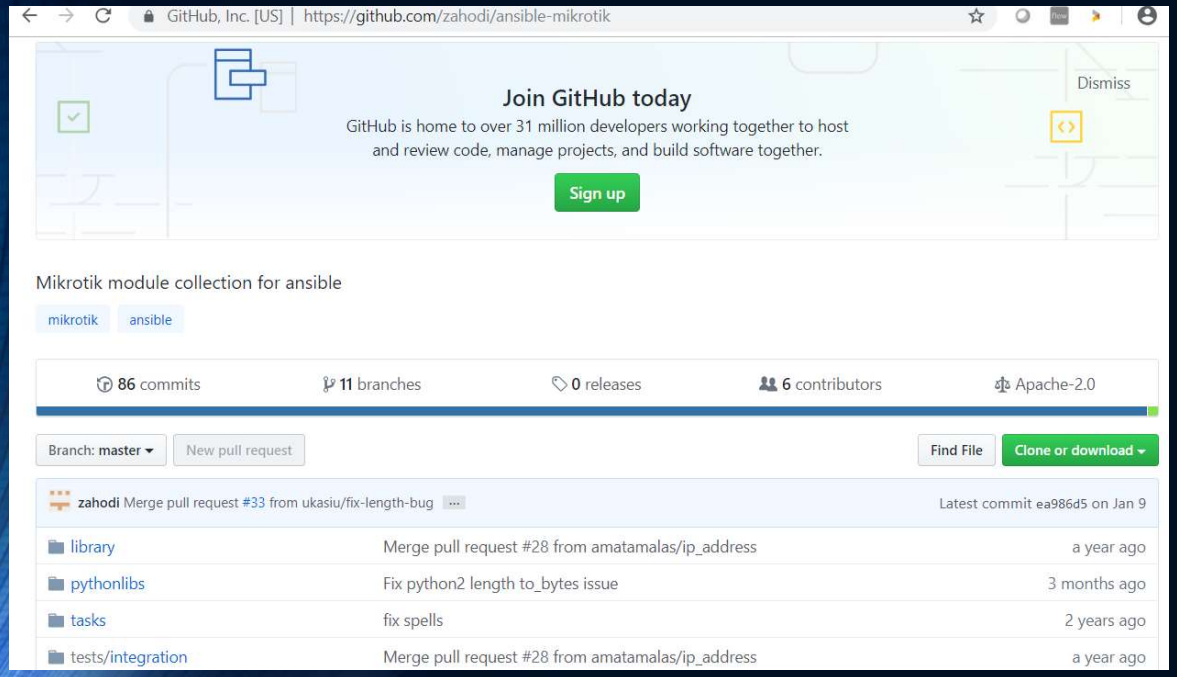
- Templates written in jinja2 format used to create configuration filesLos templates
  - Contain variables {{ }}
  - Can generate files in several formats (HTML, XML, JSON)
- Roles are collections of code including tasks, templates, variables, files and modules.
  - Fundamental mechanism to divide a playbook in various files
  - Simplifies complex playbook creation and facilitates code re use
  - Still: directory structure is fundamental
  - Many vendors offer their own generated and tested roles
- Modules are code used to perform some function in ansible
  - A.K.A task or library plugins
  - written en python

IDE : Integrated Developing environments



- GNS3 with network automation appliance
  - Linux based Appliances are launched as containers in GNS3 server
  - Must be connected to internet to update linux packages and be able to leverage git. (use NAT in GNS3 project to connect)
  - GNS3 server can run as a VM on vmware workstation. Be sure to enable the virtual NAT interface (different from NAT cloud in GNS3 project) .
- Using sublime text 3 with ansible syntax plugin. Helps with indentation and other stuff (CRITICAL)

Get on GIT



- Distributed Control version system
  - Free open-source
  - Very popular amog Python and Ansible developers
  - Well documented and easy to use
  - You can "clone" a GIT repository to re use the included code.
  - Fundamental tool for Ansible users and developers

Links & References

- https://docs.ansible.com/ansible/latest/network/user_guide/platform_routeros.html#
- https://docs.ansible.com/ansible/latest/modules/routeros_command_module.html
- https://github.com/CFSworks/ansible-routeros
- https://codeburst.io/jinja-2-explained-in-5-minutes-885484868834e
- https://netapp.io/2017/01/30/getting-started-ansible-playbooks-can-addictive-heres-can-get-hooked/
- https://dev4devs.com/2018/10/31/ansible-understanding-roles/

IP ArchiTechs MANAGED SERVICES

1-855-MIKROTIK

www.iparchitechs.com

Questions?

- The content of this presentation will be available at mum.iparchitechs.com
- Please come see us at the IP ArchiTechs booth in the Exhibitor Hall
- Email: jorge.Ravazzola@iparchitechs.com
- Office: (303) 590-9946
- Web: www.iparchitechs.com
- Thank you for your time and enjoy the MUM!!