cisco Live!

Let's go

The bridge to possible

Automating Cloud Networking with Ansible, Python and the Meraki API

John Shea Meraki TSA
CCIE 51399, DevNet Pro

BRKOPS-2243

Cisco Webex App

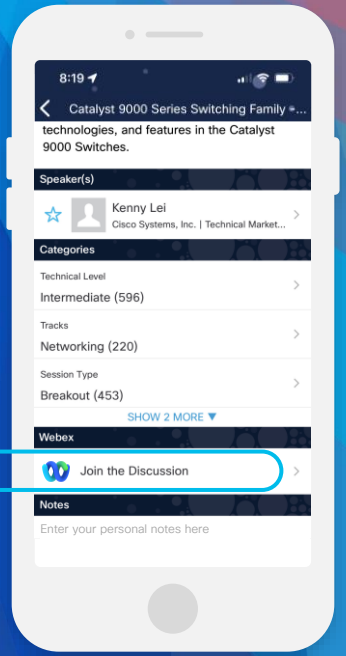
Questions?

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 9, 2023.



https://ciscolive.ciscoevents.com/ciscolivebot/#BRKOPS-2243

Agenda

- Introduction – Automation in Multi Platform Deployments
- The Power of Normalized Data and APIs
- Multi-Platform automation with Ansible
- Multi-Platform automation with Python
- Monitoring our Meraki Networks Programmatically
- Q&A

The Power of Normalized Config Data and APIs

“A Multi Domain/Platform network can be complex to design and manage, and requires careful planning and coordination to ensure that each domain is properly secured and isolated”

The Challenges of Multi Domain/ Platform Environments

- Skillset and knowledge gaps
- Complexity
- Integration issues
- Change management

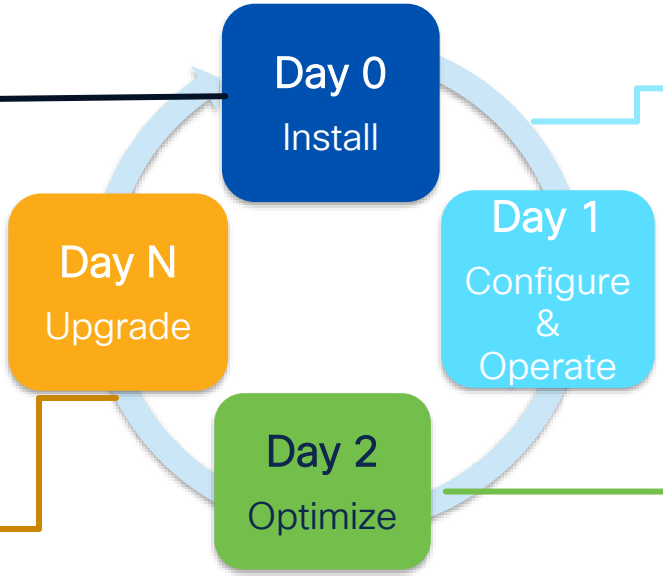
The Challenges of Multi Domain Environments

- Skillset and knowledge gaps: Managing a multi-domain environment often requires a diverse set of skills and knowledge.
- Complexity: Multi-domain environments are often complex, with many different types of devices and configurations to manage
- Integration issues: Multi-domain environments may use different tools and technologies. This can lead to gaps in visibility and increase the risk of errors.
- Change management: Changes to configurations or policies can have a significant impact on the network, especially in multi-domain environments.

Automate the Network Lifecycle

GOAL:
Get device/s into an operational state

Provisioning Automation Tools: ZTP, PnP, Python Scripting



GOAL:
Apply configuration to the device

Tools:
Orchestration Tools / Python

GOAL:
Continuously upgrade network, incrementally and safely

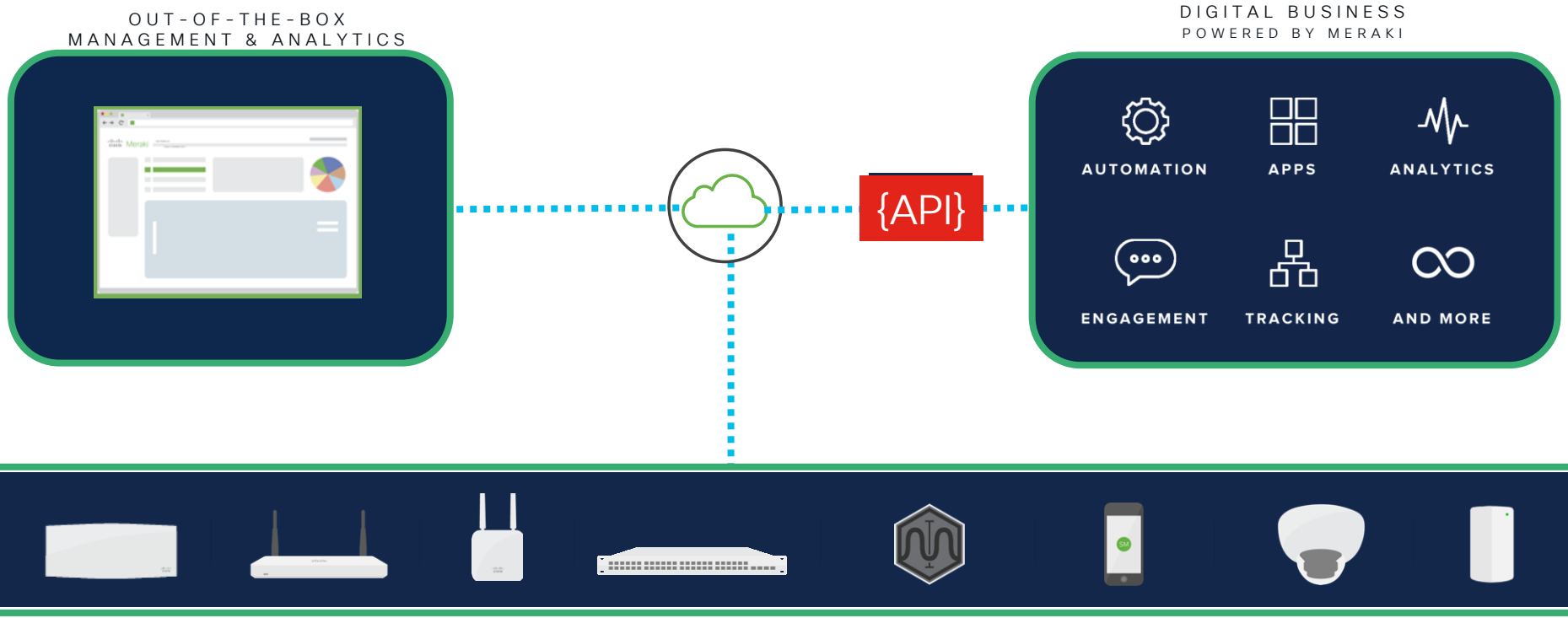
Tools:
Patching, Config / Replace

GOAL:
Add dynamic services, optimize behavior and trouble shooting

Tools:
REST / RESTCONF, Telemetry

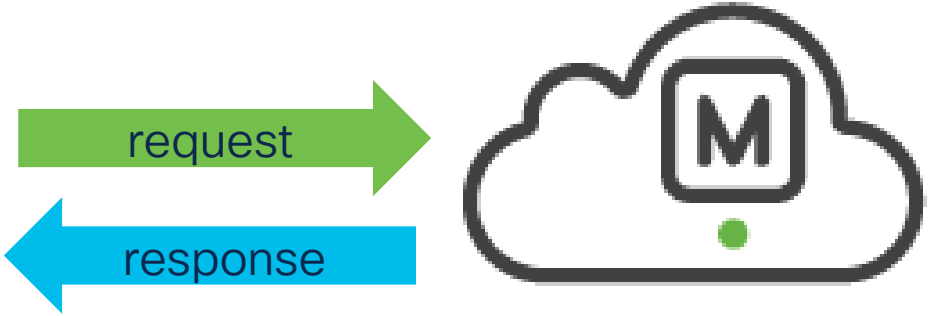
“An API is simply a way for 2 or more applications to talk to each other”

The Meraki Cloud Platform



The Value Prop for Automation Via APIs

```
>>> do("repetitious
work...")
done.
do it over, and over
and over again ...
```



- ✓ Request actions be performed
- ✓ Get information
- ✓ Store information
- ✓ Modify Information
- ✓ Submit information

Why do we work with Structured Data?

Machines have trouble interpreting formatted text

Its easy to load into and process in memory.

Allows us to separate our data from our code.

Formatted Text

Meraki

Configuration

Port status	Enabled
Type	Access
VLAN	817
Voice VLAN	--
Access policy	Open
Link negotiation	Auto negotiate (1 Gbps)
RSTP	Enabled (Forwarding)
Port schedule	Unscheduled

IOS-XE

```
Catalyst(config)
 Interface GigabitEthernet1/0/16
  switchport access vlan 817
  switchport mode access
  speed auto
  duplex auto
  no shutdown
```

Structured Data

JSON

```
[{
"id": " device-1",
    ”network_name": ”CLUS-2023",
    ”template_name": "RemoteBranch",
    "device_name ": "MX68",
    " serial_no ": "XXXX-XXXX-XXXX",
    "vlan_id": 1,
    "ip_address”: “10.168.1.1”,
    “subnet_mask”: “255.255.255.0”
    “gateway_ip”: n/a,
    “dns_server1”: “208.67.222.222”
    “dns_server2”: “208.67.220.220”
}]
```

YAML

```
device-1:
  network_name: CLUS-2023
  template_name: RemoteBranch
  device_name: MX68
  serial_no: XXXX-XXXX-XXXX
  vlan_id: 1
  ip_address: 10.168.1.1
  subnet_mask: 255.255.255.0
  gateway_ip: n/a
  dns_server1: 208.67.222.222
  dns_server2: 208.67.220.220
```

What Can We do to Normalize Config Data and Why should I?

- Choose a standard format
- Consistent Structure
- Automate configuration deployment
- Centralize management
- Test and validate configurations

What Can We do to Normalize Config Data?

- Choose a standard format: NetOps teams should choose a standard format for normalizing configurations, such as YAML or JSON. The format should be easy to read, write, and parse, and should be compatible with the tools and technologies used in the environment.
- Use templates: Templates are pre-configured files that contain configuration data for specific devices or domains. Using templates helps to ensure that configurations are consistent and reduces the risk of errors.
- Automate configuration deployment: Automating configuration deployment helps to ensure that configurations are applied consistently and accurately across all domains in the network.

Normalized Config

- Expressed in YAML
- Common elements standardized as a template
- Consumable by common automation tools and deployable to multiple platforms
- Easy to read and validate

```yaml
---
  interfaces:
    interface: '16'
    description: 'access-template'
    mode: 'access'
    vlan: '997'
    portstate: 'enabled'
    speed: 'speed'
    duplex: 'duplex'
```

Multi-Domain/Platform Automation with Ansible, Meraki and Catalyst

Why Ansible?

- Agentless
  - Push Model
- Modular
- Open-Source Community and Vendor Commercial Support
- Multi Platform / Domain
  - Meraki
  - Catalyst
  - Many Others
- Idempotent

Demo
Ansible
Documentation

Ansible Components

- **Playbooks** where you define the steps required to achieve a particular configuration or state. Each playbook consists of one or more plays.
- A **Play** is a set of tasks to be performed against a targets.
- A **Task** consists of a module and its parameters. Ansible modules are pre-defined units of code that perform specific actions on the target.

Breaking down our Playbook

- Use Prompts to Get API Key and Network Name

```
vars_prompt:
  - name: auth_key
    prompt: "Enter your API Key: "
    private: yes

  - name: network_name
    prompt: "Enter your Network Name: "
    private: no
```

- Load our variables from our YAML config data and assign to a variable

```
tasks:

  - name: include network vars
    include_vars:
      file: ./{{ network_name|lower }}/net-vars.yaml
      name: netvars

  - name: include variables for devices
    include_vars:
      file: ./{{ network_name|lower }}/devices.yaml
      name: devices
```

Breaking down a Playbook

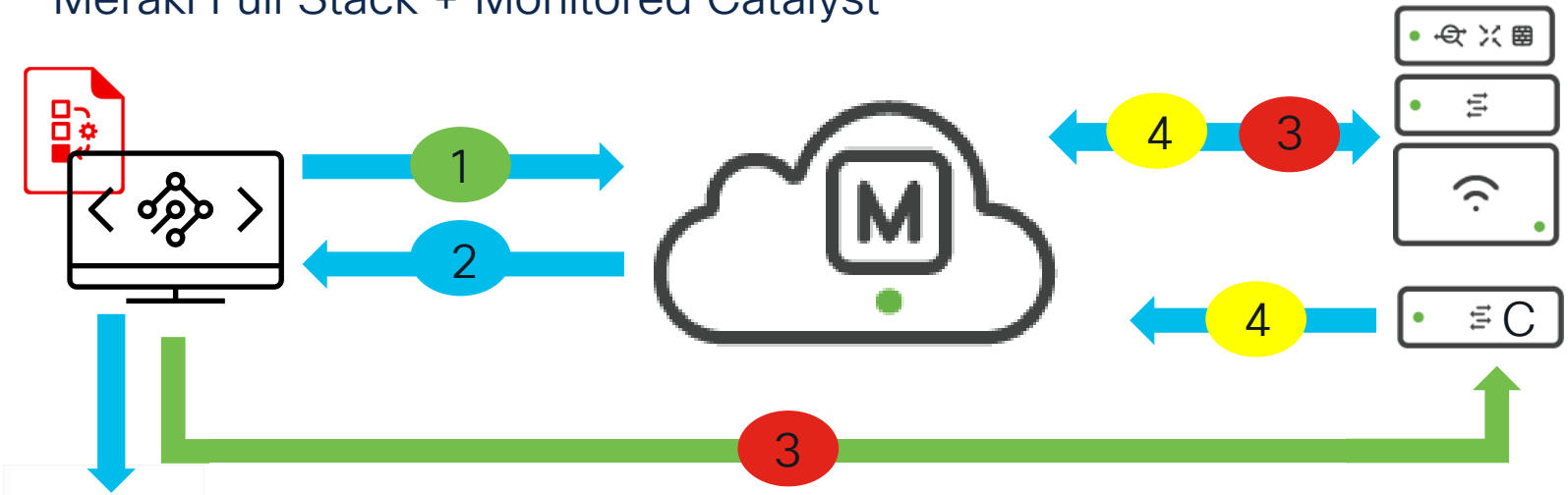
- Task Name
- Module Name
- Authentication
- Register Variable
- Ignore Errors
- Loop through dictionary Vars
- Debug Output (register)

```
- name: Configure - add ms switch port vlans
  cisco.meraki.meraki_ms_switchport:
    auth_key: "{{ auth_key }}"
    state: present
    serial: "{{ item.value.serial }}"
    number: "{{ item.value.number }}"
    enabled: "{{ item.value.enabled }}"
    name: "{{ item.value.name }}"
    tags: "{{ item.value.tags }}"
    type: "{{ item.value.type }}"
    vlan: "{{ item.value.vlan }}"
  delegate_to: localhost
  register: add_ms_swports
  ignore_errors: yes
  loop: "{{ lookup('dict', m_swports, wantlist=True) }}"
  when: item.value.platform != "catalyst"


#- debug:
    #var: add_ms_swports
```

Multi Platform Orchestration with Ansible

Meraki Full Stack + Monitored Catalyst



1. Ansible parses and executes the Playbook
2. Executes the Tasks against specific Endpoints
3. Push Config to Meraki and Monitored Catalyst
4. Dashboard Telemetry Updated
5. Communicate Changes / Modifications

Install Ansible and Meraki Collections

Ansible* installation:

- With Python PIP - ”pip install ansible-core” from the command line, verify with ansible -version
- For more information: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

Ansible Meraki Collection:

- From the CLI “ansible-galaxy collection install cisco.meraki”
- For more information: https://galaxy.ansible.com/cisco/meraki

* Ansible can be run on any Unix-like host machine , this includes Red Hat, Debian, Ubuntu, macOS. For Windows it must be run under the Windows Subsystem for Linux (WSL).

Configure RESTCONF on IOS-XE

Authentication

RESTCONF connections should be authenticated using AAA credentials. RADIUS, TACACS+ or local users defined with privilege level 15 access are allowed. AAA impacts other systems, such as administrator access to the CLI. Here is a lab configuration example from an ISR-4451 using local credentials (without any AAA enabled):

```
username admin privilege 15 secret cisco123
```

HTTP/HTTPS

RESTCONF runs over HTTPS. The following commands must be enabled to support RESTCONF over port 443:

```
ip http secure-server
```

Demo
Automation with Ansible Playbooks

Visible Success!

```
Enter your API Key: :
Enter your Network Name: : devnet-2022

PLAY [meraki deployment] ******************************************************

TASK [Gathering Facts] ********************************************************
ok: [localhost]

TASK [include network vars] ***************************************************
ok: [localhost]

TASK [include variables for devices] ******************************************
ok: [localhost]

TASK [include variables for addresses] ****************************************
ok: [localhost]

TASK [include variables for network specific ssids] ***************************
ok: [localhost]

TASK [include variables for switchports] **************************************
ok: [localhost]

TASK [include Common clients for all networks] ********************************
ok: [localhost]
```

Visible Success!

```
TASK [Create org network] ******************************************************
changed: [localhost] => (item={'key': 'device-1', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-mx-1',
'device_type': 'MX68', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.128.1', 'subnet_mask': '255.255.255.0',
'gateway_ip': 'n/a', 'dns_server1': '208.67.222.222', 'dns_server2': '208.67.220.220', 'enable_vlans': True}})
skipping: [localhost] => (item={'key': 'device-2', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-SW-1',
'device_type': 'MS220-24P', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.2', 'subnet_mask': '255.255.255.0',
'gateway_ip': '10.168.1.1'}})
skipping: [localhost] => (item={'key': 'device-3', 'value': {'network_name': 'devnet-2022', 'device_name': 'ap-1', 'device_type':
'devnet2022-MR76-1', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.3', 'subnet_mask': '255.255.255.0',
'gateway_ip': '10.168.1.1'}})

TASK [Add devices to Network] **************************************************
ok: [localhost] => (item={'key': 'device-1', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-mx-1', 'device_type':
'MX68', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.128.1', 'subnet_mask': '255.255.255.0', 'gateway_ip': 'n/
a', 'dns_server1': '208.67.222.222', 'dns_server2': '208.67.220.220', 'enable_vlans': True}})
ok: [localhost] => (item={'key': 'device-2', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-SW-1', 'device_type':
'MS220-24P', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.2', 'subnet_mask': '255.255.255.0', 'gateway_ip':
'10.168.1.1'}})
ok: [localhost] => (item={'key': 'device-3', 'value': {'network_name': 'devnet-2022', 'device_name': 'ap-1', 'device_type':
'devnet2022-MR76-1', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.3', 'subnet_mask': '255.255.255.0',
'gateway_ip': '10.168.1.1'}})

TASK [WebexTeams - Text Message by an Individuals ID] **************************
ok: [localhost]

TASK [Update device Information] ***********************************************
changed: [localhost] => (item={'key': 'device-1', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-mx-1',
'device_type': 'MX68', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.128.1', 'subnet_mask': '255.255.255.0',
'gateway_ip': 'n/a', 'dns_server1': '208.67.222.222', 'dns_server2': '208.67.220.220', 'enable_vlans': True}})
changed: [localhost] => (item={'key': 'device-2', 'value': {'network_name': 'devnet-2022', 'device_name': 'devnet2022-SW-1',
'device_type': 'MS220-24P', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.2', 'subnet_mask': '255.255.255.0',
'gateway_ip': '10.168.1.1'}})
changed: [localhost] => (item={'key': 'device-3', 'value': {'network_name': 'devnet-2022', 'device_name': 'ap-1', 'device_type':
'devnet2022-MR76-1', 'serial_no': 'xxxx-xxxx-xxxx', 'vlan_id': 1, 'ip_address': '10.168.1.3', 'subnet_mask': '255.255.255.0',
'gateway_ip': '10.168.1.1'}})
```

Visible Success!

```
TASK [Enable Vlans in Networks] ************************************************
ok: [localhost]

TASK [Create a VLAN.] **********************************************************
changed: [localhost] => (item={'key': 'subnet-1', 'value': {'network_name': 'devnet-2022', 'name': 'Management', 'vlan_id': 1,
'subnet': '10.0.1.0/24', 'default_gw': '10.0.1.2'}})
changed: [localhost] => (item={'key': 'subnet-2', 'value': {'network_name': 'devnet-2022', 'name': 'devnet-Corp', 'vlan_id': 10,
'subnet': '10.10.1.0/24', 'default_gw': '10.10.1.1'}})
changed: [localhost] => (item={'key': 'subnet-3', 'value': {'network_name': 'devnet-2022', 'name': 'devnet-User', 'vlan_id': 20,
'subnet': '10.20.1.0/24', 'default_gw': '10.20.1.1'}})
changed: [localhost] => (item={'key': 'subnet-4', 'value': {'network_name': 'devnet-2022', 'name': 'devnet-IOT', 'vlan_id': 30,
'subnet': '10.30.1.0/24', 'default_gw': '10.30.1.1'}})

TASK [WebexTeams - add vlans] **************************************************
ok: [localhost]

TASK [Delete Default SSID] *****************************************************
ok: [localhost]

TASK [add ssids] ***************************************************************
ok: [localhost] => (item={'key': 'ssid-2', 'value': {'ssid_name': 'IOT', 'net_name': 'devnet-2022', 'enabled': True, 'auth_mode':
'psk', 'psk': '12345abc', 'encryption_mode': 'wpa', 'ip_assignment_mode': 'Bridge mode', 'number': 2, 'splash_page': 'none',
'use_vlan_tagging': True, 'vlan_id': 11}})
ok: [localhost] => (item={'key': 'ssid-5', 'value': {'ssid_name': 'devnet-2022', 'net_name': 'devnet-2022', 'enabled': True,
'auth_mode': 'psk', 'psk': 'abc12345', 'encryption_mode': 'wpa', 'ip_assignment_mode': 'Bridge mode', 'number': 2, 'splash_page':
'none', 'use_vlan_tagging': True, 'vlan_id': 11}})

TASK [WebexTeams - add SSID] ***************************************************
ok: [localhost]

TASK [insert a pause for API to catchup] ***************************************
Pausing for 5 seconds
(ctrl+C then 'C' = continue early, ctrl+C then 'A' = abort)
ok: [localhost]
```

Visible Success!

```
TASK [add switch vlans] ********************************************************
ok: [localhost] => (item={'key': 'switchport-10', 'value': {'serial': 'xxxx-xxxx-xxxx', 'number': 10, 'enabled': True, 'name':
'uplink-port', 'tags': 'uplink', 'type': 'trunk', 'vlan': 1}})
ok: [localhost] => (item={'key': 'switchport-1', 'value': {'serial': 'xxxx-xxxx-xxxx', 'number': 1, 'enabled': True, 'name': 'Client-
PC-1', 'tags': 'POS', 'type': 'access', 'vlan': 10}})
ok: [localhost] => (item={'key': 'switchport-2', 'value': {'serial': 'xxxx-xxxx-xxxx', 'number': 2, 'enabled': True, 'name': 'AP-1',
'tags': 'AP', 'type': 'trunk', 'vlan': 1}})
ok: [localhost] => (item={'key': 'switchport-3', 'value': {'serial': 'xxxx-xxxx-xxxx', 'number': 3, 'enabled': True, 'name': 'Client-
PC-1', 'tags': 'clientPC', 'type': 'access', 'vlan': 11}})

TASK [WebexTeams - report on switch ports] *************************************
ok: [localhost]

TASK [WebexTeams - network created] ********************************************
ok: [localhost]

PLAY RECAP *********************************************************************
localhost                  : ok=21   changed=3    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

- Runs through each task.
- Let’s you know how many tasks were ok, changed, failed, etc.
- To see more output use “-v”, “-vvv”, or “-vvvv”

Deploy-Net Playbook with Verbose Output

% ansible-playbook deploy-net.yaml -vvv


```
    "ansible_loop_var": "item",
    "changed": false,
    "invocation": {
        "module_args": {
            "msg": "Catalyst m_cat9200l-1  Switchport 1/0/4 in Network CiscoLive have been Configured.",
            "msg_type": "text",
            "personal_token": "VALUE_SPECIFIED_IN_NO_LOG_PARAMETER",
            "recipient_id": "VALUE_SPECIFIED_IN_NO_LOG_PARAMETER",
            "recipient_type": "roomId"
        }
    },
    "item": {
        "key": "Switchport-1/0/4",
        "value": {
            "enabled": true,
            "hostname": "m_cat9200l-1",
            "name": "uplink-port to MX Spare",
            "number": "1/0/4",
            "platform": "catalyst",
            "rport": "1%2F0%2F3",
            "tags": "uplink",
            "type": "access",
            "url": "192.168.1.208",
            "vlan": 1
        }
    },
    "message": "OK (unknown bytes)",
    "status_code": 200
}
```

Ansible Tags for Running Specific PB Tasks

If you have a comprehensive playbook and only want to execute a specific task in a run, you can use tags.

```
Ansible-playbook deploy-net.yaml --tags “meraki,devices”
```

Format:

```
- name: Configure - Add devices to Network
    cisco.meraki.meraki_device:
      auth_key: "{{ auth_key }}"
      org_name: "{{ org_name }}"
      net_name: "{{ network_name }}"
      serial: "{{ item.value.serial_no }}”
    tags:
      - meraki
      - devices
```

Multi-Domain/Platform Automation with Python, Meraki and Catalyst

Why Python?

- Easy to learn and use
- Wide range of libraries and frameworks
- Cross-platform compatibility
- Integration with other tools
- Scalability
- Community Support

Using Dashboard Tags to Deploy Config to Meraki and Monitored Catalyst Endpoints

208.67.222.222

LAN IPV6

Not configured

TERMINAL

Launch Terminal

SERIAL NUMBER

J EG (Catalyst)

Q '-8ZMP (Meraki)

TAGS

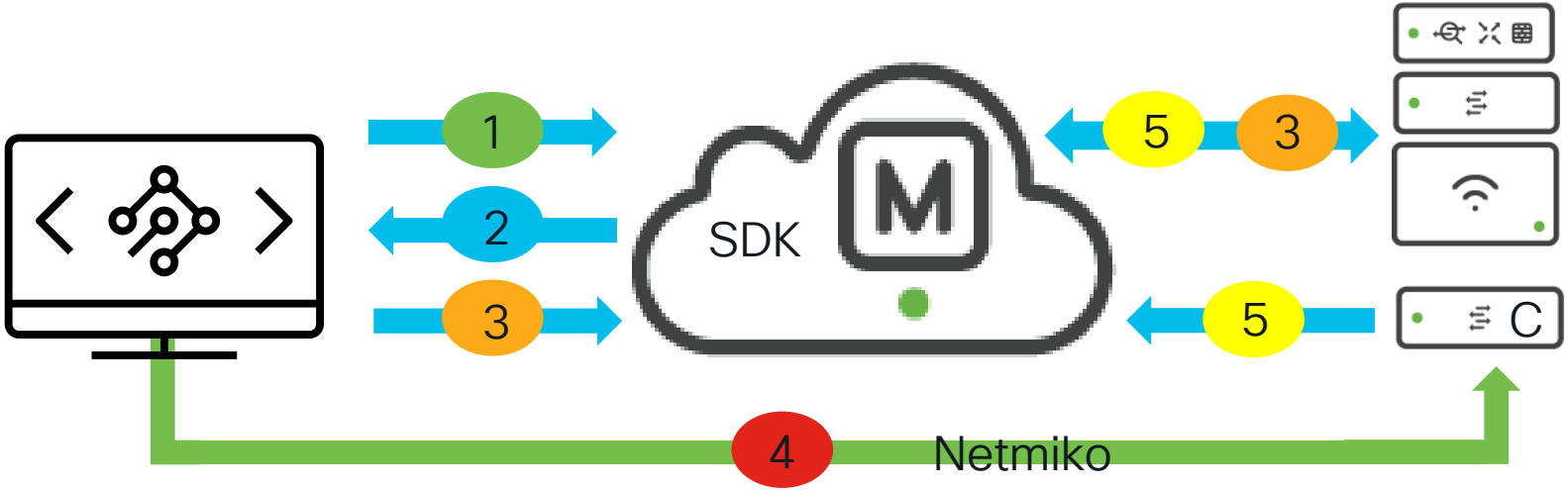
interfaces ntp recently-added routes snmp stp svi

Python Libraries

- Meraki SDK
- Netmiko
- PathLib
- Jinja2

Multi Platform Orchestration with Python

Meraki Full Stack + Monitored Catalyst



1. GET ORG Devices (Meraki & Catalyst)
2. Get data and parse for Tags
3. PUT Meraki configs based on Tag
4. PUT Catalyst config based on Tags
5. Dashboard Telemetry Updated

Let The Meraki SDK do the work for you!

- The SDKs are built from the Meraki API OpenAPI specification
- The library can take care of error handling, logging, retries, and other convenient processes and options for you automatically.
- To install pip install meraki and then simply import it as any other python library.

Connect to Catalyst with Netmiko

```
#Instantiate netmiko connector
from netmiko import ConnectHandler

#Create connection object
Iosxe_17 = {
        'device_type': 'cisco_ios'
        'ip': <deviceIP>
        'username': <username>
        'password': <password>

connect = ConnectionHandler(**iosxe17)
connect.enable()
```

Configure Catalyst with Jinja2

```
{% for loopback in loopbacks %}
    {% set address = loopback.address %}
    {% set netmask = loopback.netmask %}
    {% set ipaddress = address ~ " " ~
        netmask %}
    interface  {{ loopback.int }}
    description {{ loopback.description }}
    ip address {{ ipaddress }}
{% endfor %}
```

Demo
Python Automation

Monitoring our Network Programmatically (NO CODE)

Key features of the Meraki Dashboard for monitoring include:

- Device-specific dashboards: You can view the performance and statistics of each individual device, such as access points, switches, and security appliances.

- Real-time data: The dashboard provides real-time information on network traffic, bandwidth usage, and device status.

- Alerts and notifications: You can set up alerts and notifications to be notified when certain events occur, such as when a device goes offline or when there is a security threat.

Reporting with Meraki Tools and Google Sheets

Dashboard Reports

File Edit View Insert Format Data Tools Extensions Help Meraki-Reports Last edit was seconds ago

C2 | 1 NOCODE WAY, Meraki USA

	A	B	C	D	E	F	G	H	I
1	lat	lng	address	serial	mac	tags.0	tags.1	url	networkI
2	38.2329912	-85.4280639	1 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:76:68:07	MT	recently-added	https://n111.mera	L_62543
3	38.23192709	-85.42862535	2 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	f8:9e:28:7e:2b:4b	MT		https://n111.mera	L_62543
4	0	0	3 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	f8:9e:28:86:62:73	recently-added		https://n111.mera	L_62543
5	38.23219677	-85.43047071	4 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	f8:9e:28:77:9a:5f	MT		https://n111.mera	L_62543
6	38.23187652	-85.42789578	5 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	2c:3f:0b:ff:eb:0f	MT		https://n111.mera	L_62543
7	0	0	6 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:1a:db:70	MT		https://n111.mera	L_62543
8	0	0	7 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	f8:9e:28:78:51:a0	MT	MT12	https://n111.mera	L_62543
9	0	0	8 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	f8:9e:28:78:51:95	MT	MT12	https://n111.mera	L_62543
10	0	0	9 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:1a:ee:ea			https://n111.mera	L_62543
11	38.23298	-85.42807	10 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	38:84:79:00:c2:d7	MT-30	MT30	https://n111.mera	L_62543
12	38.23298	-85.42807	11 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	38:84:79:00:c4:7f	MT-14	MT14	https://n111.mera	L_62543
13	37.4180951	-122.0985317	12 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	c4:8b:a3:fc:60:b6			https://n111.mera	L_62543
14	37.4180951	-122.0985317	13 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:1b:12:7f	recently-added		https://n111.mera	L_62543
15	0	0	14 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:be:00:87			https://n111.mera	L_62543
16	38.23299739	-85.42934418	15 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	ac:17:c8:0c:b6:0d	MV	MV12	https://n111.mera	L_62543
17	0	0	16 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	98:18:88:00:f1:4b	MV	camera	https://n111.mera	L_62543
18	38.23243274	-85.42816401	17 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	34:56:fe:a3:cf:e7	MV		https://n111.mera	L_62543
19	38.23335134	-85.42997718	18 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	2c:3f:0b:f2:0c:13	MV		https://n111.mera	L_62543
20	0	0	19 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	a8:46:9d:c4:e8:86			https://n111.mera	L_62543
21	38.2319018	-85.42836785	20 NOCODE WAY, Meraki USA	xxxx-xxxx-xxxx	2c:3f:0b:ff:6e:2b	MV	MV2	https://n111.mera	L_62543

https://developer.cisco.com/meraki/build/meraki-dashboard-reports-with-google-sheets/

Demo
Meraki Tools & Google Sheets

Proactive Alerting with Webex

- Use predefined templates to send alert data to Webex (and others)
- Customizable with Liquid Templates
- Secure
- Deploy in Minutes

T500 9:05 AM

Power supply went down

Alert ID:

Alert level: critical

Occurred at: 2023-03-13T13:05:32.826192Z

Organization name: DevNet

Organization ID: 365279

Network name: Dev-Catalyst

Network ID: L_770678486233778274

Network Tags: cat

Device name: m_cat9200l-1

Device serial: Q: MP

Device MAC: 08:ec:f5:d6:ca:80

Device tags: interfaces, ntp, recently-added, routes, snmp, stp, svi

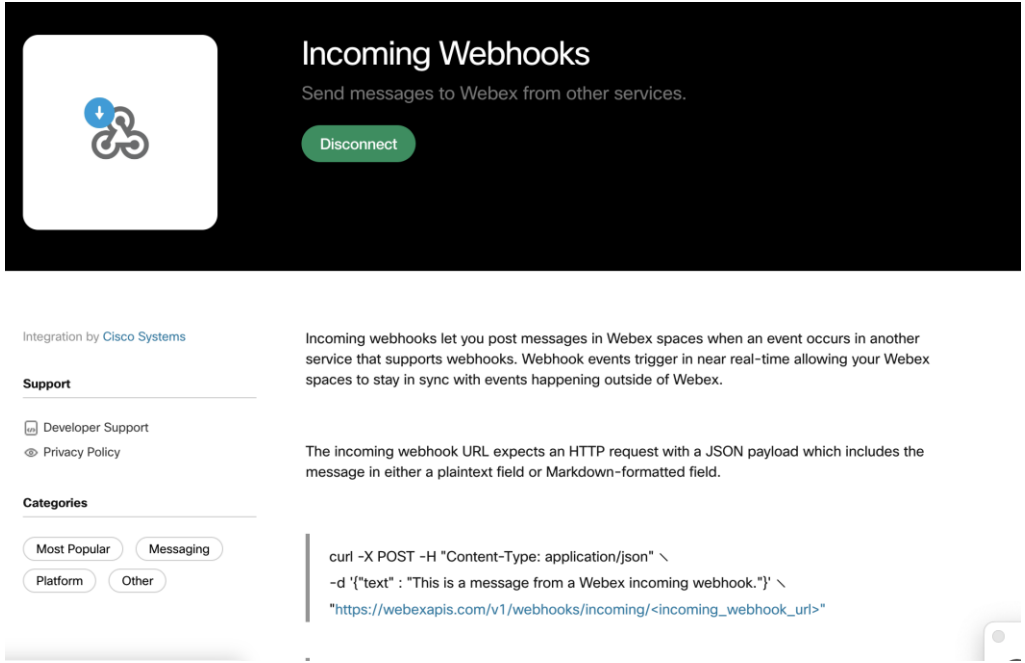
Device model: C9200L-24P-4X

Alert Data

num: 2

Setting up Webex for Webhooks

Read the Webex [Incoming Webhooks Guide] to enable the service on your Webex account and receive an HTTP POST URL.



Setting up Webex for Webhooks

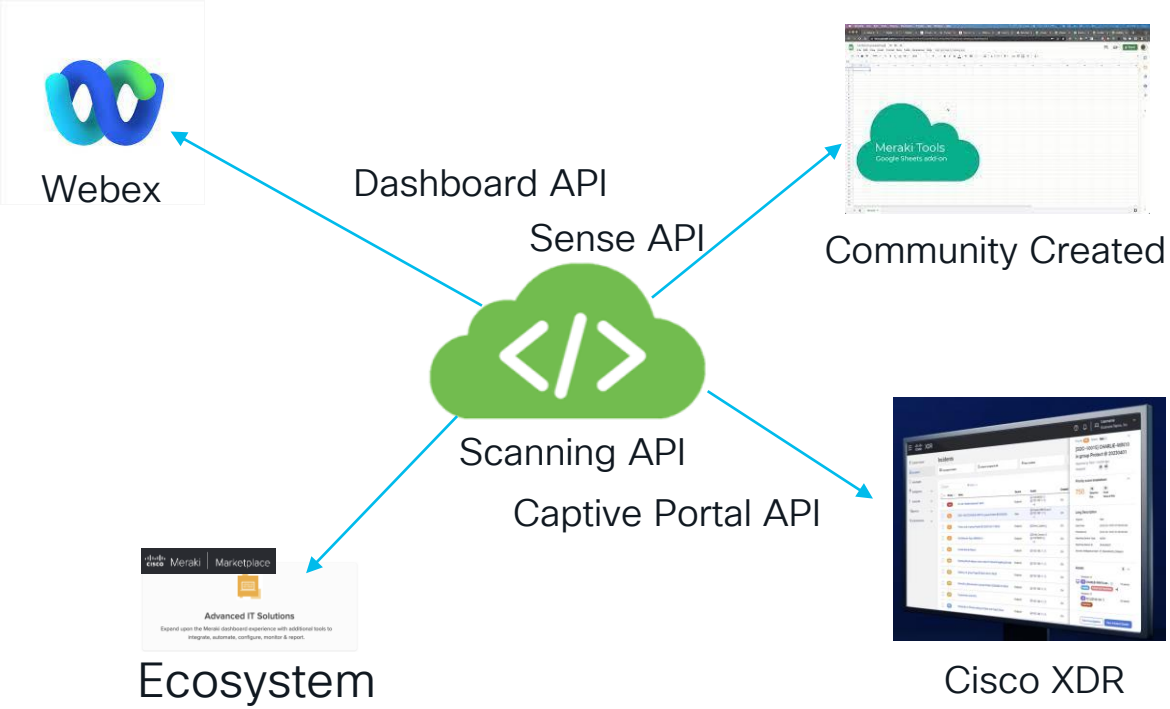
Set Dashboard to use your Webex Reciever

- Add/Update a webhook HTTP Server
  - URL: *(i.e.* `https://webexapis.com/v1/webhooks/incoming/<incoming_webhook_url>` *)*
  - Secret: `blank`
  - Payload Template: `Webex`
- Verify
  - Push the **Test** button to verify that it works.
  - Verify that you received an alert in your channel.

Demo
Dashboard Webhooks

The Power of the Platform

Integrations, Ready for Your Data



Session Recap

- During this session we discussed the operational challenges of Multi Platform / Multi domain environments
- Normalizing our configurations into a common language allows you to consume it universally across your footprint
- Using Automation tools, we can consume our normalized configurations to deploy them evenly, minimizing administrative touch, and reducing risk of error
- With Meraki APIs we can export our Meraki and Catalyst telemetry and alerts for consumption and enrichment in other platforms and tools.

Resources

- Session Source Code - https://github.com/johshea/BRKOPS-2243
- Meraki SDK - https://developer.cisco.com/meraki/api-latest/#!python/meraki-dashboard-api-python-library
- Ansible Modules- https://docs.ansible.com/ansible/2.9/modules/list_of_network_modules.html#meraki
- Meraki Tools for Google Sheets - https://developer.cisco.com/docs/meraki-tools-google-sheets-add-on/#!introduction
- Webex Integrations - https://developer.cisco.com/meraki/webhooks/#!webex-included/webex

Fill out your session surveys!

Attendees who fill out a minimum of four session surveys and the overall event survey will get **Cisco Live-branded socks** (while supplies last)!

Attendees will also earn 100 points in the **Cisco Live Challenge** for every survey completed.

These points help you get on the leaderboard and increase your chances of winning daily and grand prizes

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

Thank you

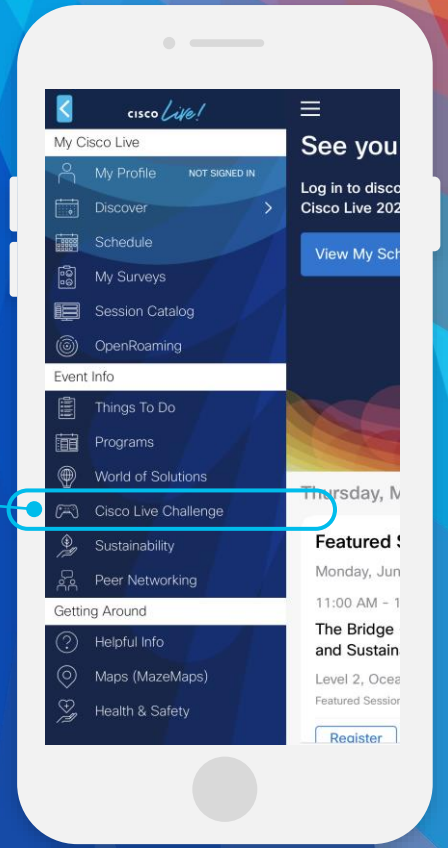
Cisco Live
Challenge

Gamify your Cisco Live experience!

Get points for attending this session!

How:

1. Open the Cisco Events App.
2. Click on 'Cisco Live Challenge' in the side menu.
3. Click on View Your Badges at the top.
4. Click the + at the bottom of the screen and scan the QR code:



cisco Live!

Let's go