



The bridge to possible

# Simplifying network management with Meraki API

Mikael Fredriksson, Technical Solution Architect, Meraki Channel  
CCIE #9861

# 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 until February 24, 2023.





# Agenda

- The History of Network management
- Why should we change
- How do we start the journey to API's
- What can we change
- See what can be done for real

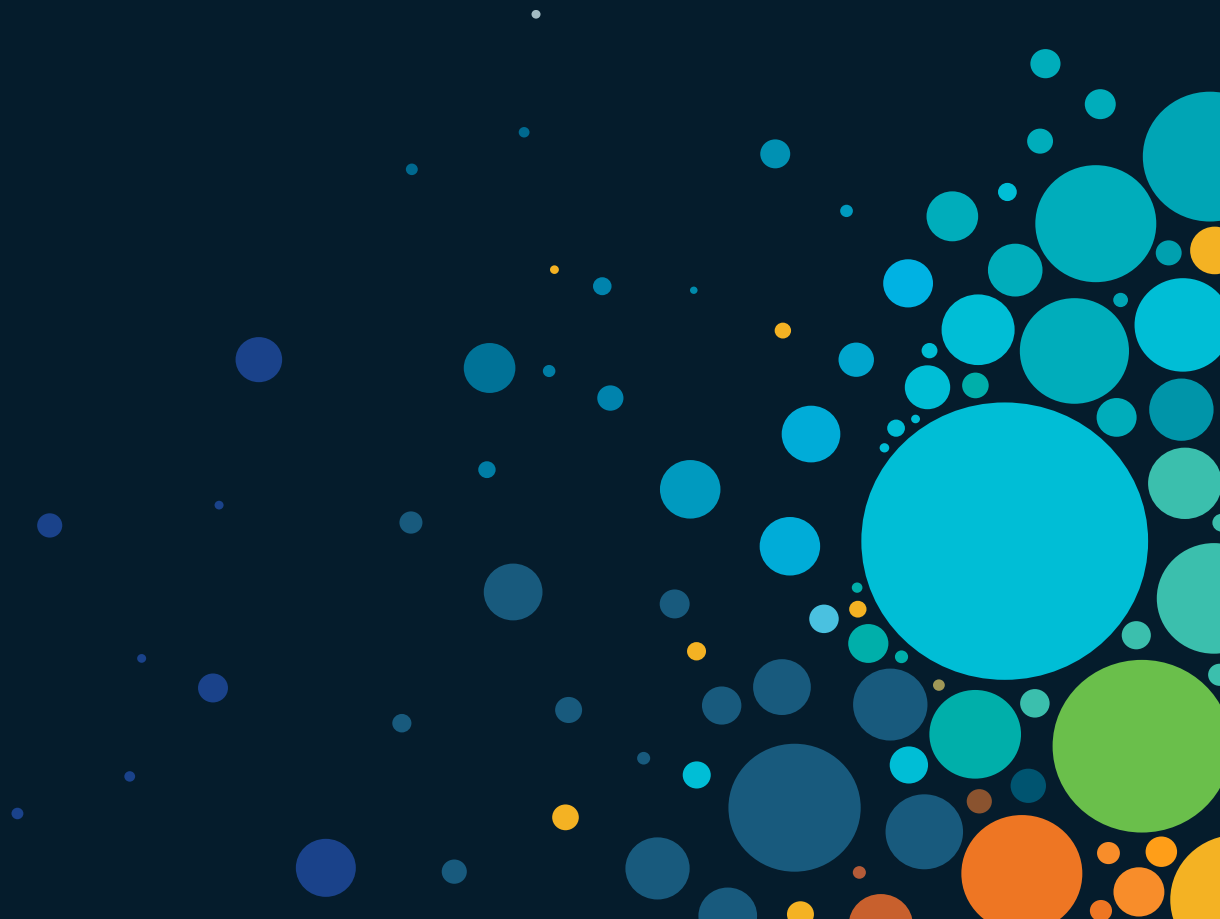
# Who am I?

Mikael Fredriksson



- TSA with the Meraki BU
- 2xCCIE of +20 Years
- Started with IT communication mid 90th
- Father of 3 children
- Happily married for +30 year
- First time speaker
- Passionated skier

# The history of network management



# Most of us remember this I hope

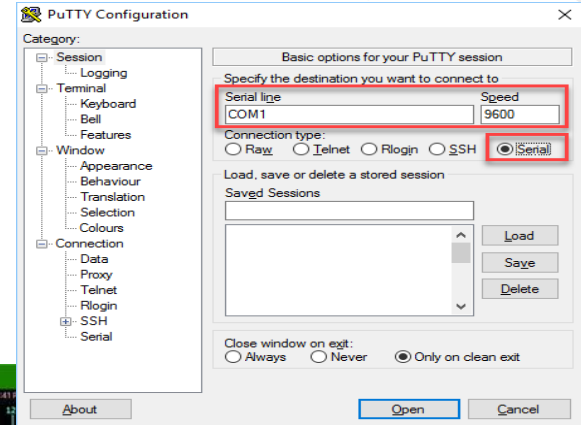


```
ssh sales@10.10.0.13
cisco C9208L-24PXG-2Y (ARM64) processor with 528021K/3871K bytes of memory.
Processor board ID JAE231488Q2
2 Virtual Ethernet interfaces
16 Gigabit Ethernet interfaces
8 Ten Gigabit Ethernet interfaces
2 Twentyfive Gigabit Ethernet interfaces
2648K bytes of non-volatile configuration memory.
1984768K bytes of physical memory.
415284K bytes of Crash Files at crashinfo.
1941584K bytes of Flash at flash:.

Base Ethernet MAC Address       : dc:8c:37:38:ef:b9
Motherboard Assembly Number     : 73-18783-85
Motherboard Serial Number       : JAE231488Q2
Model Revision Number           : A8
Motherboard Revision Number     : A8
Model Number                    : C9208L-24PXG-2Y
System Serial Number            : JAE231488Q2
CLEI Code Number                : INNM08AARA

Switch Ports Model          SW Version  SW Image        Mode
-----
* 1 26  C9208L-24PXG-2Y    17.04.01   CAT9K_LITE_IOSXE  INSTALL

Configuration register is 0x1802
VMD#_0002#
```



# Configuration database/Golden Config

- Individual config principles
- Uppercase/Lowercase
- Scripts
- Risk for typing error

```

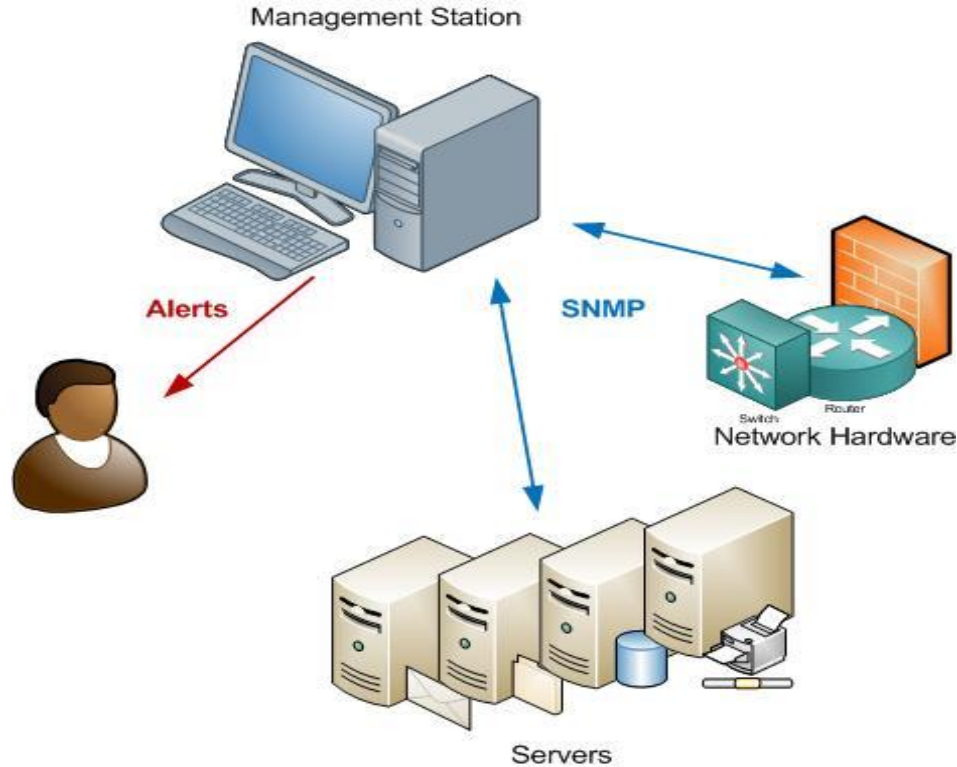
File Edit View Options Transfer Script Tools Window Help
Cisco IOS XE CLI
Please register Cisco Nexus7000 Family devices promptly with your
supplier. Failure to register may affect response times for initial
service calls. Nexus7000 devices must be registered to receive
entitled support services.

Press Enter at anytime to skip a dialog. Use ctrl-c at anytime
to skip the remaining dialogs.

Would you like to enter the basic configuration dialog (yes/no): no
Cisco Nexus Operating System (NX-OS) Software
The support: http://www.cisco.com/err
Copyright (c) 2000-2014, Cisco Systems, Inc. All rights reserved.
The copyrights to certain works contained in this software are
owned by either third parties and used and distributed under
license. Certain components of this software are licensed under
the GNU General Public License (GPL) version 2.0 or the GNU
Lesser General Public License (LGPL) Version 2.1. A copy of each
such license is available at
http://www.opensource.org/licenses/gpl-2.0.php and
http://www.opensource.org/licenses/lgpl-2.1.php
DC-W7K1-VMC1
DC-W7K1-VMC1#
DC-W7K1-VMC1 conf t
Enter configuration commands, one per line. End with CNTL/Z.
DC-W7K1-VMC1(config)# featu
feature
feature-set
DC-W7K1-VMC1(config)# feature-set fax
DC-W7K1-VMC1(config)# int port-channel1
DC-W7K1-VMC1(config-if)# switchport
DC-W7K1-VMC1(config-if)# switchport mode fax-fabric
DC-W7K1-VMC1(config-if)# fax associate 101
DC-W7K1-VMC1(config-if)# no shut
DC-W7K1-VMC1(config-if)# exit
DC-W7K1-VMC1(config)# int eth4/12,eth7/12
DC-W7K1-VMC1(config-if-range)# switchport
DC-W7K1-VMC1(config-if-range)# switchport mode fax-fabric
DC-W7K1-VMC1(config-if-range)# fax associate 101
DC-W7K1-VMC1(config-if-range)# no shut
DC-W7K1-VMC1(config-if-range)# channel-group 101
DC-W7K1-VMC1(config-if-range)# exit
2014.03.13 13:07:49 DC-W7K1-VMC1 45 vnc-2 45 45vnc-2-0-nexus7000_row_fex_mgmt: DC-W7K1-VMC1 Module 1: Check environment alarms.
2014.03.13 13:07:49 DC-W7K1-VMC1 45 vnc-2 45 45vnc-2-nexus7000_row_fex_mgmt: fex-101 on-line (Serial Number: F0C14640J9J)
DC-W7K1-VMC1(config-if-range)# exit
DC-W7K1-VMC1(config)# exit
DC-W7K1-VMC1# show f

```

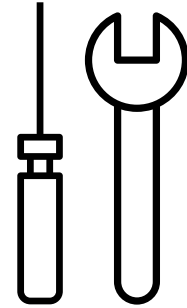
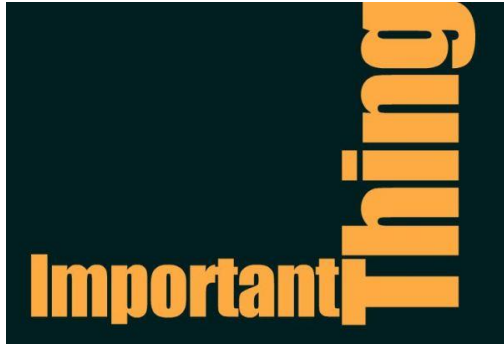
# Proactive monitoring and management





Why do we use  
API's?

# How programmability is a competitive advantage



# The potential benefits to use API for your management



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)



[This Photo](#) by Unknown Author is licensed under [CC BY-NC-ND](#)



[This Photo](#) by Unknown Author is licensed under [CC BY](#)



[This Photo](#) by Unknown Author is licensed under [CC BY-SA-NC](#)

# Ever had to...

Deploy 1,000 sites  
install 5,000 switches and  
configure 170K switch ports?

Let's do the math:

Running many devices in parallel give us an  
average of 10 min/device.

This include:

Unpack/repack

Powerup/Firmware/Configuration load

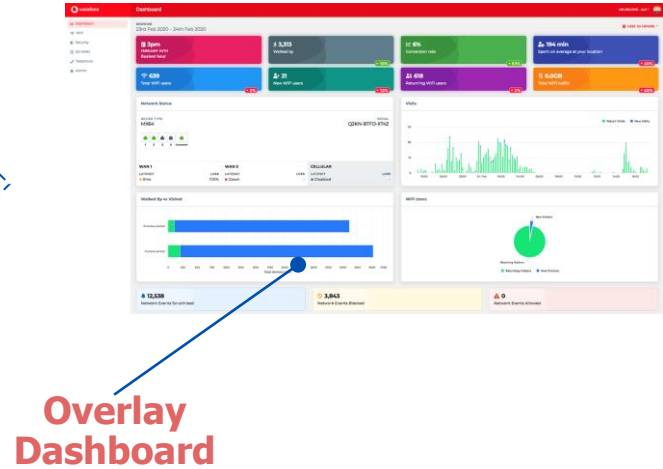
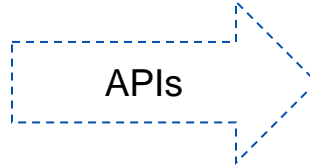
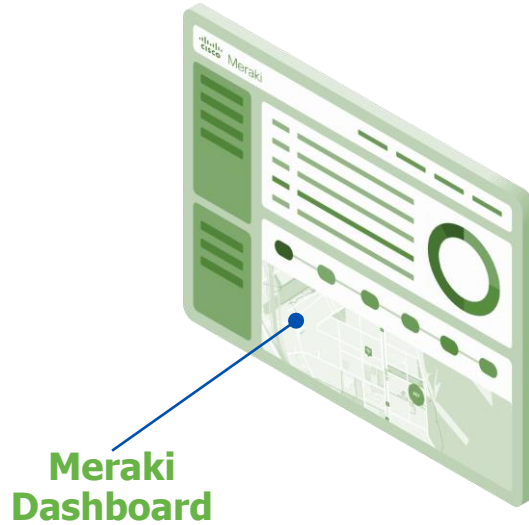
Prepare shipping

Total time  $10\text{min} \times 5,000 \text{ switches} =$   
833 hours/100 working days (8hour days)

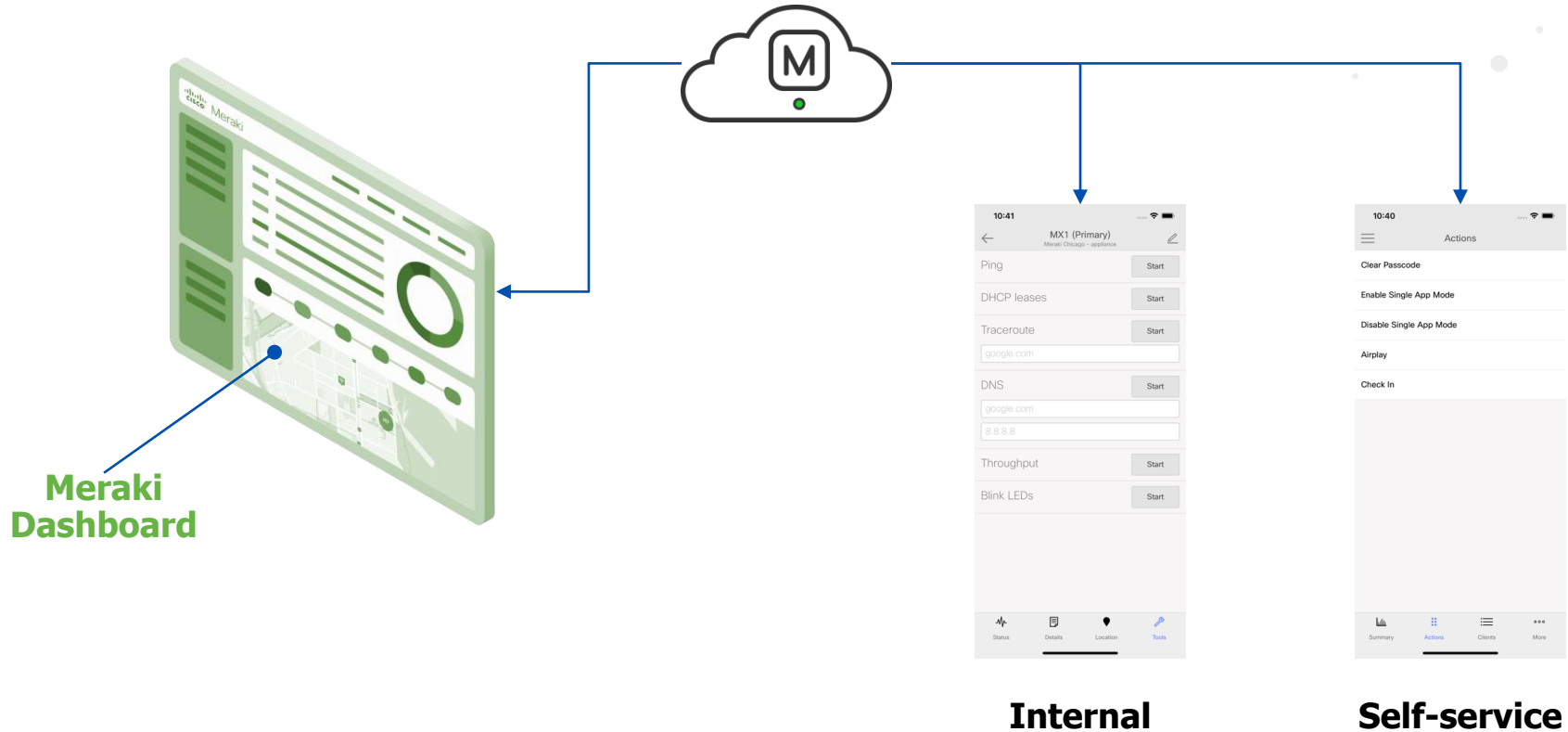
**CISCO** *Live!*



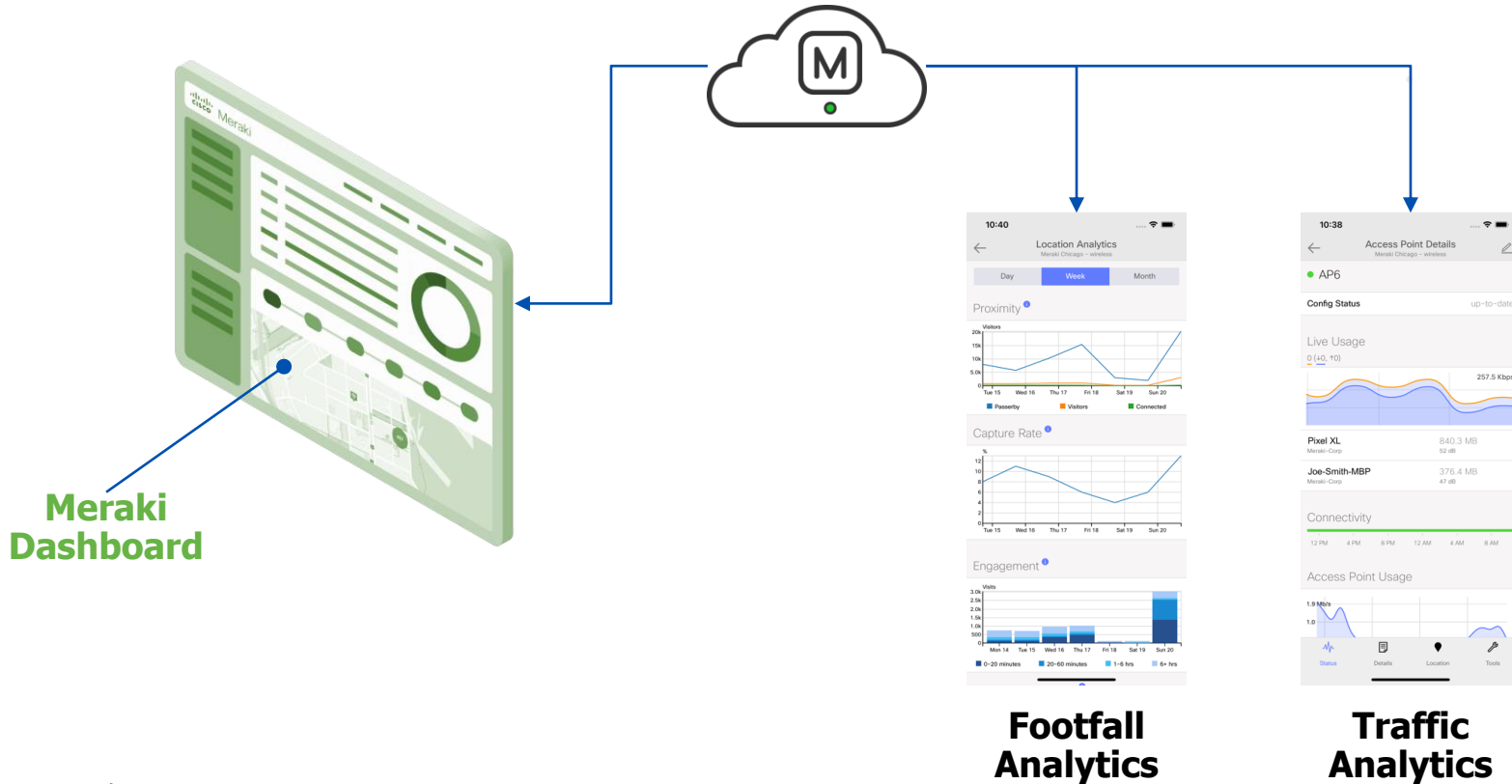
# Offer customer view in your control of Single Pane



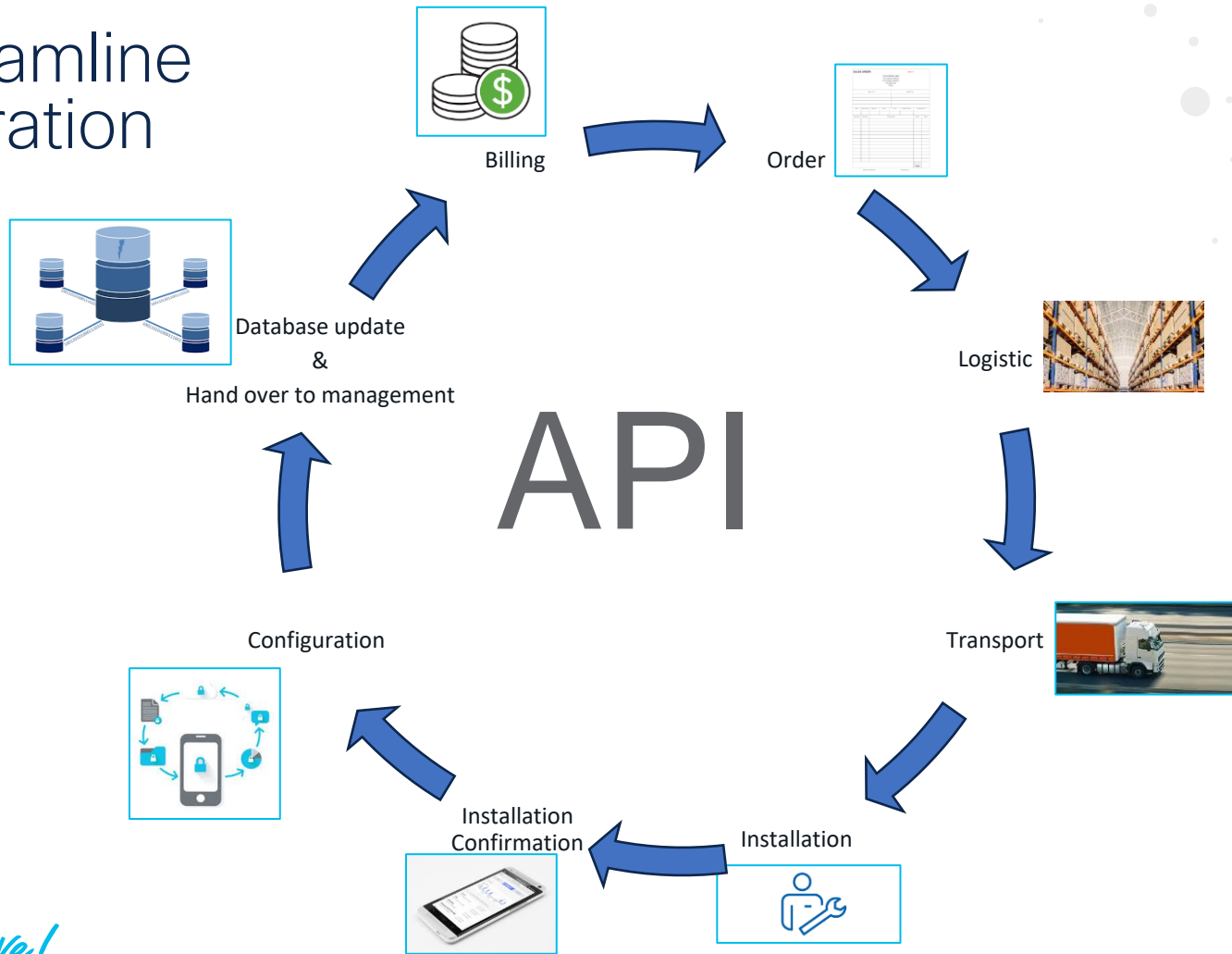
# Offer dedicated application based on use case



# Analytics



# Streamline operation





How do we start  
the journey to  
API's

# How to start the API journey?

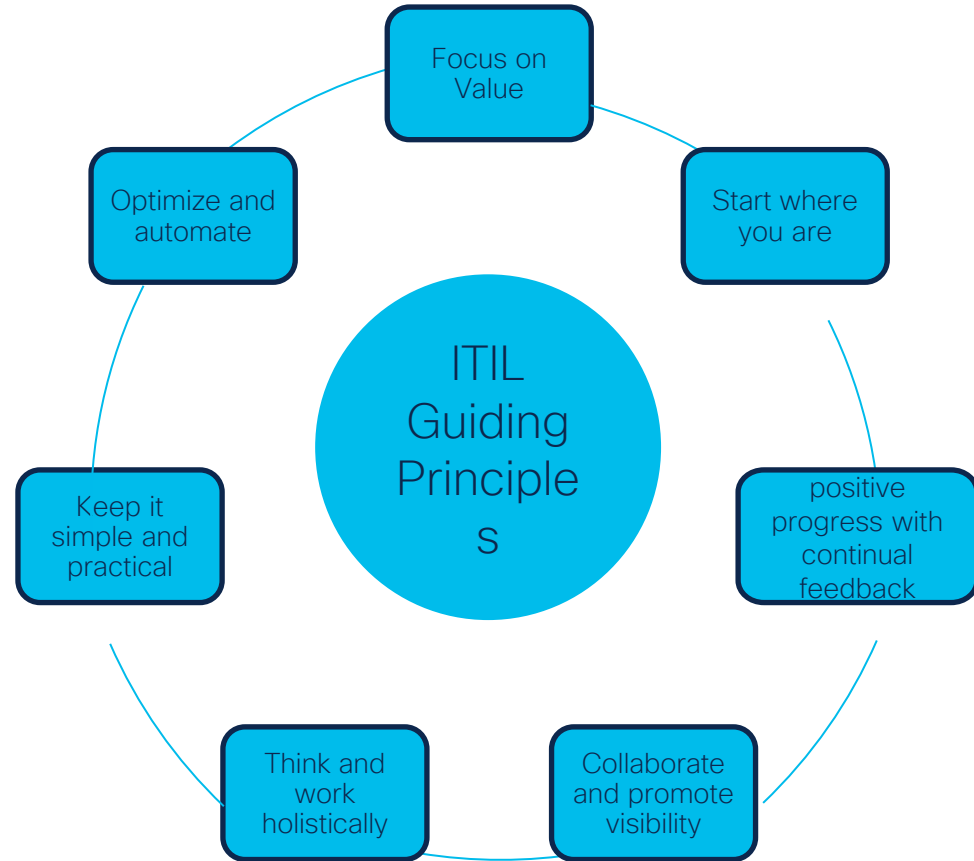
“What process do I start with?”

“What is the most important thing to consider when building a service?”

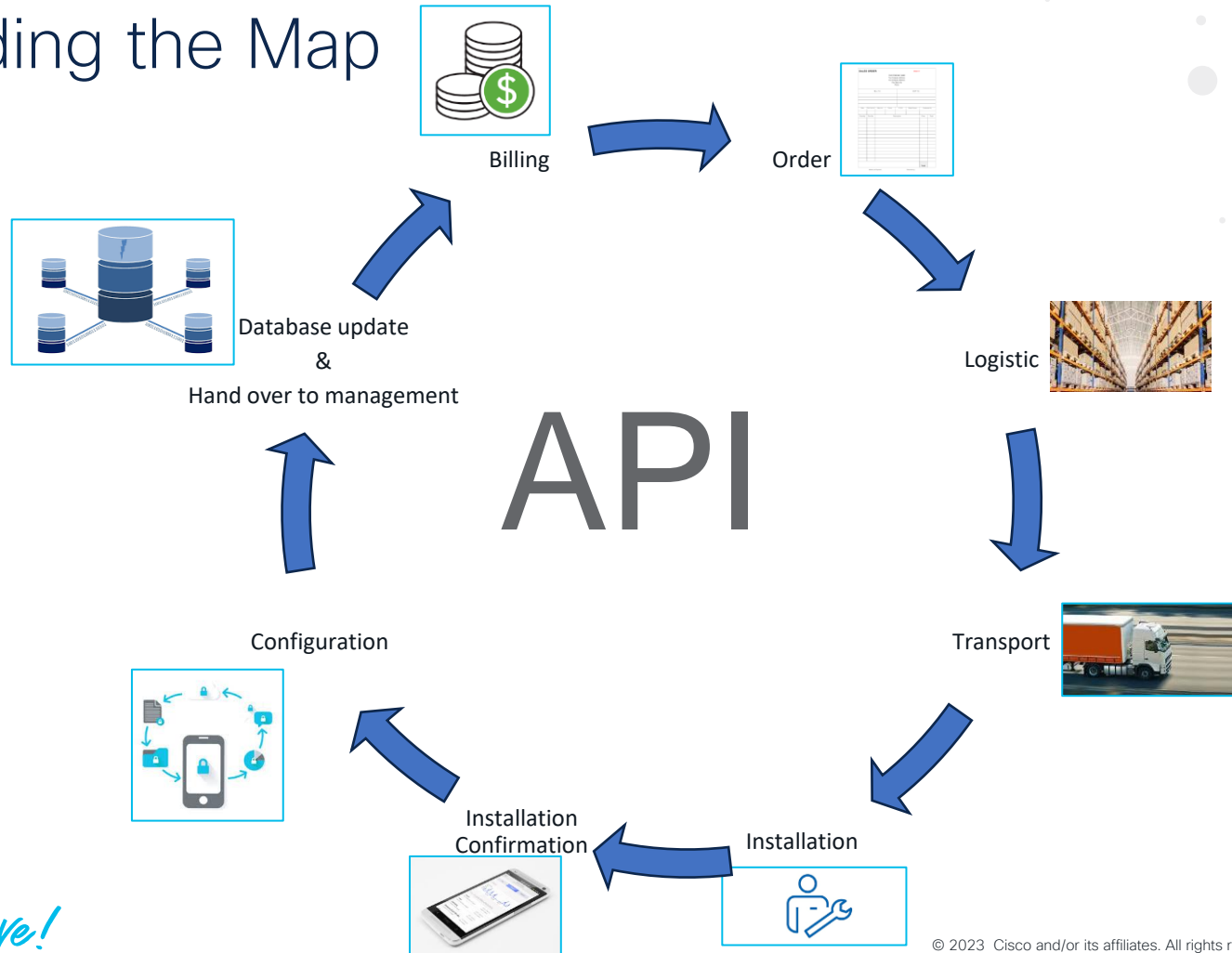
“How much time and resources do I require to build the service?”

“How do I differentiate?”

“I created an offer, but I receive only few requests”



# Building the Map



# Ecosystem integrations. Rich API tool kit.

BUILD



{APIs and more}

## Dashboard API

- Programmability
- Automation
- Monitoring
- Reporting
- Data insights
- Snapshot API

## Webhook API

- Event stream
- Automation trigger

## Scanning API

- Asset tracking
- Location analytics
- Wayfinding

## MQTT wireless

- Real-time location services
- Wayfinding\*

## Captive portal API

- Guest Wi-Fi
- Secure Onboarding

## MV Sense API

- Real-time (4 Hz) data stream
- Historical time-series via REST
- Current snapshot

\*Application firmware deployed on Meraki hardware

# Top Use Cases for Meraki APIs

## BULK PROVISIONING & CONFIGURATION



Setup 10k networks  
across 5 time zones

## AUTOMATION



Auto-configure a  
switch port when a  
printer is plugged in

## CLOUD INTEGRATIONS



Integrate with custom  
and 3<sup>rd</sup> party  
applications

## ADVANCED MONITORING & ALERTING



Monitor all devices and  
trigger a workflow if a  
site goes down

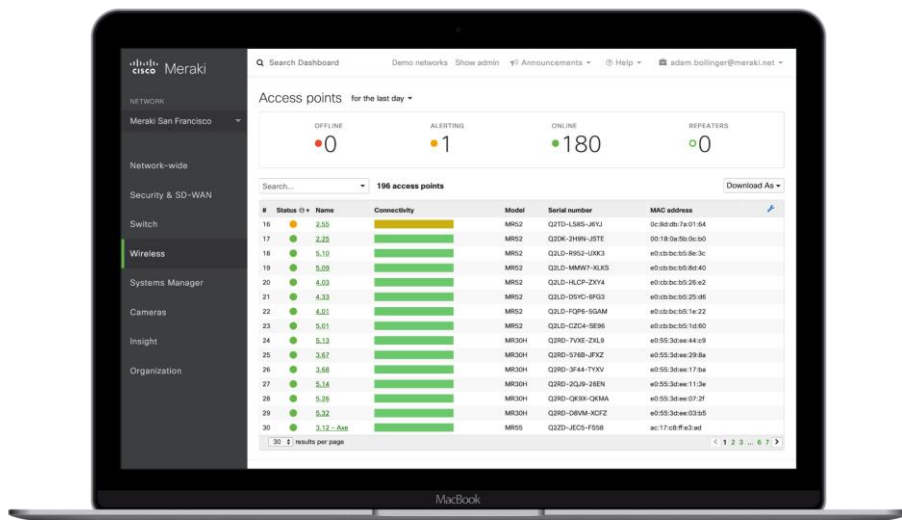
## ADVANCED REPORTING & ANALYTICS



Produce a report that  
shows % of clients on  
Wi-Fi vs wired

# Dashboard API

- Example | Meraki device info and status



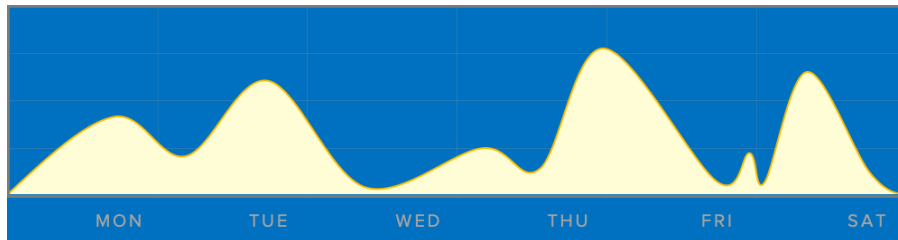
```
{
  "name": "2.55",
  "serial": "Q2TD-LS8S-J6YJ",
  "mac": "0c:8d:db:7a:01:64",
  "publicIp": "67.188.23.251",
  "networkId": "L_599541700393699125",
  "status": "alerting",
  "lanIp": "192.168.1.174"
},
{
  "name": "2.25",
  "serial": "Q2DK-2H9N-JSTE",
  "mac": "00:18:0a:5b:0c:b0",
  "publicIp": "67.188.23.251",
  "networkId": "L_599541700393699125",
  "status": "online",
  "lanIp": "172.16.0.5"
},
}
```

Dashboard → API

# Monitoring vs Alerting

## Monitoring Over Time – Dashboard REST API

E.g. latency trend over the last week



## Real-Time Alerting – Webhooks

E.g. alert if the latency is greater than 200ms

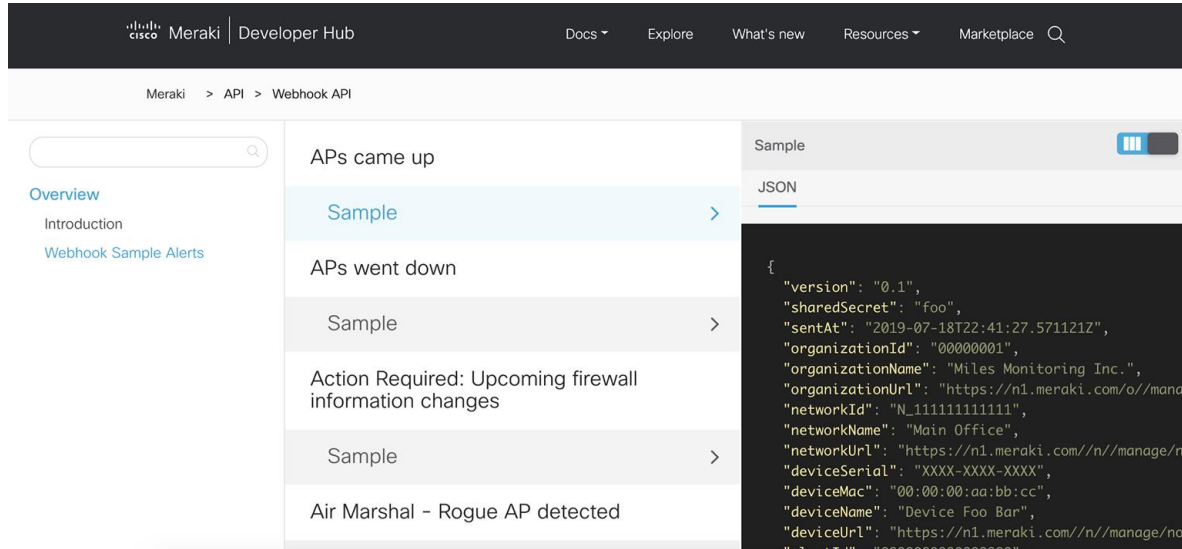
ALERT 1

Thu, 7:53am: Latency for access point in [conference room 3](#) exceeded 200ms

ALERT 2

Fri, 2:10am: Latency for access point in [conference room 3](#) exceeded 200ms

# Cloud Events via Webhook to any Web Client



Meraki | Developer Hub

Docs ▾ Explore What's new Resources ▾ Marketplace 🔍

Meraki > API > Webhook API

Overview  
Introduction  
Webhook Sample Alerts

APs came up  
Sample >

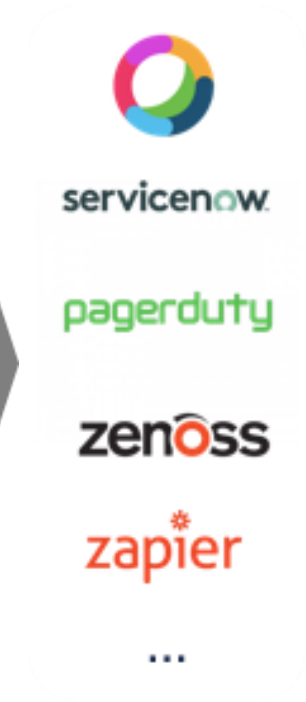
APs went down  
Sample >

Action Required: Upcoming firewall information changes  
Sample >

Air Marshal - Rogue AP detected

Sample  
JSON

```
{
  "version": "0.1",
  "sharedSecret": "foo",
  "sentAt": "2019-07-18T22:41:27.571121Z",
  "organizationId": "00000001",
  "organizationName": "Miles Monitoring Inc.",
  "organizationUrl": "https://n1.meraki.com/o//manage/n1",
  "networkId": "N_111111111111",
  "networkName": "Main Office",
  "networkUrl": "https://n1.meraki.com/n//manage/nod",
  "deviceSerial": "XXXX-XXXX-XXXX",
  "deviceMac": "00:00:00:aa:bb:cc",
  "deviceName": "Device Foo Bar",
  "deviceUrl": "https://n1.meraki.com/n//manage/nod",
  "deviceType": "XXXXXXXXXXXXXXXXXXXX"
}
```





# So what steps to take?

Think algebra and math

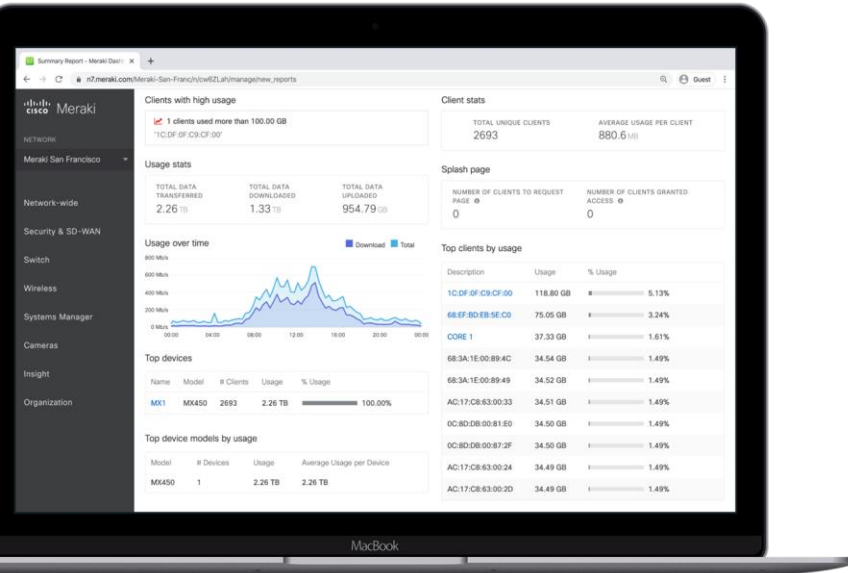
Constants – Partly coded in  
(or a definition file)

- User access (SSO)
- Security Settings
- Branding
- Wireless/Wired settings
- Templates
- Special settings

Variables – Input fields

- Naming
- IP schema
- SSID/VLANs
- Inventory
  - Device serials
  - Licenses
- Special access users

# Dashboard as template



- Think of the workflow you use in the dashboard.
- Common settings
- Clone/Copy
- Naming standards

What can we  
change



Starting to use API is like  
starting to learn alpine skiing

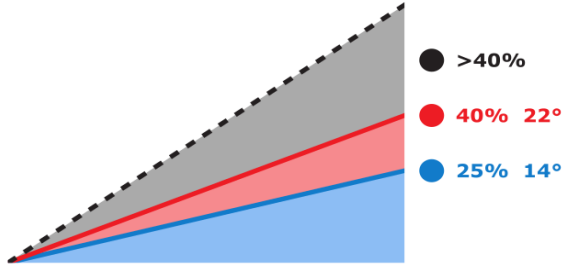
[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)

# Where to start

- Start small
- Find the low hanging fruits
- Test and Learn
- Expand
- DIY or BUY

```
7
8
9
10 function getDevJob(studying, hardWork, luck) {
11     var isPrepared = studying && hardWork && luck;
12     if (isPrepared) {
13         return true;
14     } else {
15         return false;
16     }
17 }
18
19
```

# What do I need to get started?



[This Photo](#) by Unknown Author is licensed under [CC BY-SA](#)



[Devnet - Sandbox](#)



[This Photo](#) by Unknown Author is licensed under [CC BY-ND](#)

# Access to a Complete Ecosystem



Select the  
**outcome** for your  
business

[apps.meraki.io](https://apps.meraki.io)

## BUSINESS OUTCOME SOLUTIONS

- ML Video Analytics
- Asset & People Tracking
- Real-time, Interactive Customer Experience
- POS & Wi-Fi Ordering
- Physical Security & Access Control
- Dwelling & Dormitory Experience
- Wayfinding & Mapping
- Wi-Fi Analytics & Engagement
- [Smart Campus & Workplace](#)
- [Video Security](#)

## ADVANCED IT SOLUTIONS

- Automation
- Identity Services
- IT Service Management Tools
- IOT Security
- Managed Service Solutions
- Application & Performance Monitoring
- Network Monitoring & Logging
- SD-WAN Monitoring & Performance
- Network Security

[apps.meraki.io](https://apps.meraki.io)



# Demo



# API Demo CLI Style!

[https://youtu.be/\\_igfuAqxo\\_U](https://youtu.be/_igfuAqxo_U)

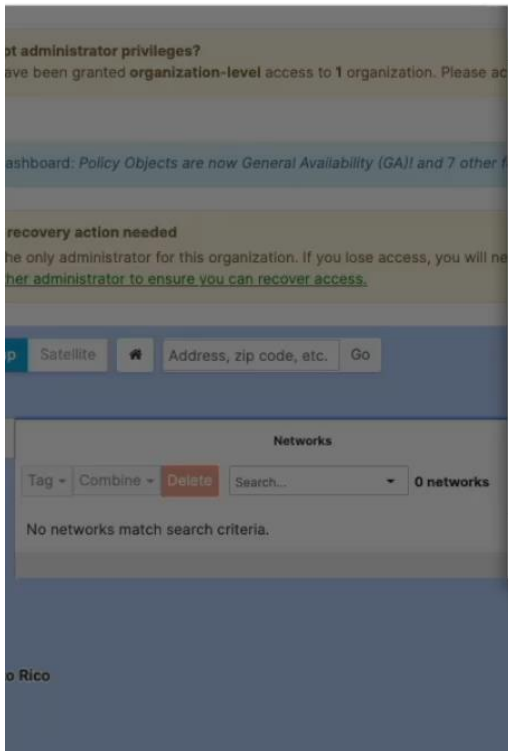
```
Mikael-Fredriksson-MBP:~ mikael.fredriksson$
```

# API-Demo with an UI!

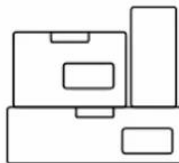
[https://youtu.be/THrU\\_MDZmmA](https://youtu.be/THrU_MDZmmA)

h Dashboard

Announcements



## Welcome to Meraki Dashboard



### Register Meraki devices

Add your devices to a network to monitor and configure them through the Meraki cloud.



### Set up Systems Manager

Enroll client devices in Meraki's EMM to manage apps, settings, and security.

Next

After some practise you are  
ready for the red slops



# Complete your Session Survey

- Please complete your session survey after each session. Your feedback is very important.
- Complete a minimum of 4 session surveys and the Overall Conference survey (open from Thursday) to receive your Cisco Live t-shirt.
- All surveys can be taken in the Cisco Events Mobile App or by logging in to the Session Catalog and clicking the "Attendee Dashboard" at <https://www.ciscolive.com/emea/learn/sessions/session-catalog.html>



# Additional learning

- [Cisco Meraki: Enabling Infrastructure as Code - BRKMER-2663](#)
- [Meraki Dashboard Automation using Python - LABMER-2405](#)
- [Enhanced Automation of a Meraki-based Environment - DEVNET-2200](#)
- [Meraki APIs 101 - DEVNET-1314](#)
- [Innovate with Meraki APIs and its Partner Marketplace - DevNet-1338](#)
- [Meraki 202 - Programmatic Action: Simplicity at Scale - DEVNET-2177](#)
- [Automate Meraki Alerting & Response with Webhooks and AWS Lambda - DE VWKS-2845](#)
- [Introduction to the Meraki Dashboard API - DE VWKS-1494](#)
- [Intro to Meraki Dashboard API Python Library - DEVNET-1303](#)
- [Cloud Management and Monitoring for Catalyst Walk-in Lab – LABCOC-2236](#)
- [World of Solution to meet our technology partners and the Meraki team](#)

Thank you



CISCO *Live!*





# Appendix

# Code for the CLI demo

```
import meraki
import time

API_KEY = 'API-Key2'
dashboard = meraki.DashboardAPI(API_KEY,
    suppress_logging=True
)
organization_id = 'Org-id'
name = 'Big-City'
product_types = ['appliance', 'switch', 'wireless',
    cellularGateway', 'sensor']
network_id = ""
print ('Create the network')
response =
dashboard.organizations.createOrganizationNetwork(
    organization_id, name, product_types,
    timeZone='Europe/Stockholm',
    notes='API demo'
)
network_id = response['id']
print ('Add devices to the Network')
serials = ['Qxxx-xxxx-xxxx', 'Qxxx-xxxx-xxxx', 'Qxxx-xxxx-
xxxx', 'Qxxx-xxxx-xxxx']
response = dashboard.networks.claimNetworkDevices(
    network_id, serials
)
print ('Create additional Administrators')
email = 'jondoe@acme.com'
name = 'Super-Admin'
org_access = 'full'
response =
dashboard.organizations.createOrganizationAdmin(
    organization_id, email, name, org_access,
)

print ('Enable VLANs')
response =
dashboard.appliance.updateNetworkApplianceVlansSettings(
    network_id,
    vlansEnabled=True
)
print ('Update VLAN-1')
vlan_id = '1'
response =
dashboard.appliance.updateNetworkApplianceVlan(
    network_id, vlan_id,
    name='Management',
    subnet='192.168.1.0/24',
    applianceIp='192.168.1.1',
    dhcpHandling='Run a DHCP server',
    dhcpLeaseTime='1 day',
    dnsNameservers='opendns',
)
print ('Create VLAN-10 Corp')
name = 'Kista-Data'
id_ = '10'
response =
dashboard.appliance.createNetworkApplianceVlan(
    network_id, id_, name,
    subnet='192.168.110.0/24',
    applianceIp='192.168.110.1',
)
vlan_id = id_
response =
dashboard.appliance.updateNetworkApplianceVlan(
    network_id, vlan_id,
    dhcpHandling='Run a DHCP server',
    dhcpLeaseTime='1 day',
    dnsNameservers='opendns',
)

print ('Update Guest SSID')
response = dashboard.wireless.updateNetworkWirelessSsid(
    network_id, 1,
    name='Branch-Guest',
    enabled=True,
    authMode='open',
    ipAssignmentMode='Bridge mode',
    splashPage='Click-through splash page',
    useVlanTagging=True,
    defaultVlanId=20,
)
print ('Update Splashpage')
response =
dashboard.wireless.updateNetworkWirelessSsidSplashSettin
gs(
    network_id, 1,
    redirectUrl='https://dn.se',
    useRedirectUrl=True,
    welcomeMessage='Välkommen!',
)
print ('Updating Device name and installation address')
address = ('Big-Place Big-City Big Country')
serials = [' Qxxx-xxxx-xxxx', 'Qxxx-xxxx-xxxx', 'Qxxx-xxxx-
xxxx', 'Qxxx-xxxx-xxxx']
devices = ['MX', 'MS', 'MR', 'MG']
for serial in serials:
    for devicename in devices:
        response = dashboard.devices.updateDevice(
            serial,
            name = devicename,
            address = address,
            moveMapMarker = True
        )
```

# Code for the UI demo part 1 of 2

```
#!/usr/bin/ python3
import meraki
import json
import sys
import time

API_KEY = 'Your API-Key'
dashboard = meraki.DashboardAPI(API_KEY)

def create_network():
    response = dashboard.organizations.createOrganizationNetwork(
        organization_id, nw_name, product_types,
        timeZone='Europe/Amsterdam'
    )

def get_network_id():
    response = dashboard.organizations.getOrganizationNetworks(
        organization_id, total_pages='all'
    )
    for nw in response:
        if (nw["name"]) == nw_name:
            nw_id = (nw["id"])
            return nw_id

def claim_device(nw_id):
    network_id = nw_id
    response = dashboard.networks.claimNetworkDevices(
        network_id, [serials]
    )

def update_device(serials):
    serial = serials
    response = dashboard.devices.updateDevice(
        serial,
        name = devicename,
        address = address,
        moveMapMarker = True
    )
```

```
def add_ssid(nw_id,ssid_name,ssid_key):
    network_id = nw_id
    response = dashboard.wireless.updateNetworkWirelessSsid(
        network_id,0,
        name = 'Corp-Data',
        enabled = True,
        vlanId = 10,
        authMode = '8021x-radius',
        ipAssignmentMode ='Bridge mode',
        wpaEncryptionMode = 'WPA3 Transition Mode',
        lanIsolationEnabled = False,
        radiusServers = [{'host':'192.168.1.1','port':1812,'secret':'secret'}, {'host':'192.168.2.2','port':1812,'secret':'secret'}
    )
    response = dashboard.wireless.updateNetworkWirelessSsid(
        network_id, 1,
        name = ssid_name,
        enabled = True,
        vlanId = 100,
        authMode = 'psk',
        ipAssignmentMode ='Bridge mode',
        psk = ssid_key,
        encryptionMode = 'wpa',
        wpaEncryptionMode = 'WPA3 Transition Mode',
        lanIsolationEnabled = False
    )
    response = dashboard.wireless.updateNetworkWirelessSsid(
        network_id, 1,
        name = ssid_name,
        enabled = True,
        vlanId = 100,
        authMode = 'psk',
        ipAssignmentMode ='Bridge mode',
        psk = ssid_key,
        encryptionMode = 'wpa',
        wpaEncryptionMode = 'WPA3 Transition Mode',
        lanIsolationEnabled = False
    )
```

# Code for the UI demo part 2 of 2

```
response = dashboard.wireless.updateNetworkWirelessSsid(
    network_id, 2,
    name = 'Guest',
    splashPage = 'Click-through splash page',
    enabled = True,
    authMode = 'open',
    ipAssignmentMode = 'NAT mode',
    splashTimeout = '240 minutes'
)

response = dashboard.wireless.updateNetworkWirelessSsidSplashSettings(
    network_id, 2,
    splashMethod = 'Click-through splash page',
    splashTimeout = 240,
    welcomeMessage = 'This is our Company Guest WiFi,\nDo not use this network for any illegal activities'
)

if __name__ == "__main__":

    print('File: ',str(sys.argv[0]))
    print('Input')
    jsonData = json.loads(sys.stdin.readline())
    print(jsonData)

    nw_name = jsonData['nw_name']
    address = jsonData['address']
    ssid_name = jsonData['ssid_name']
    ssid_key = jsonData['ssid_key']
    serials = jsonData['serials']
    devicename = jsonData['devicename']
```

```
# Creation of the network
organization_id = 'Org-ID'
product_types = ['appliance', 'switch', 'wireless']
create_network()
time.sleep(1)

# Add device to the network
# Grab Network-ID
nw_id = get_network_id()
time.sleep(1)
# Claim device to network
claim_device(nw_id)
time.sleep(1)

# Create SSID
#print(SSID_Info)
add_ssid(nw_id,ssid_name,ssid_key)
time.sleep(1)
# Update name and address to device
update_device(serials)
time.sleep(1)
```

# Use Cases



# Automation at Nationwide Rental Company

## CUSTOMER PROFILE

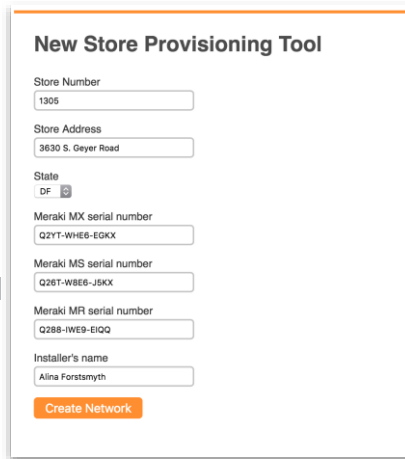
\$4B 2006 revenue	18,000 employees	2,000 retail locations
----------------------	---------------------	---------------------------

## CHALLENGE

- Automate deployment of 2,000+ nation-wide locations over 4 months
- Streamline and simplify operations to reduce overhead and costs

## SOLUTION

- Rapid, fully automated network setup and provisioning with the Meraki dashboard API
- Automate setup of all networks, bind them to configuration templates via an elegant web tool built for the field deployment team



The screenshot shows a web form titled "New Store Provisioning Tool". It contains several input fields: "Store Number" (1305), "Store Address" (3630 S. Geyer Road), "State" (DF with a dropdown arrow), "Meraki MX serial number" (Q2YT-WHE6-EGKX), "Meraki MS serial number" (Q26T-WBE6-JSKX), "Meraki MR serial number" (Q2B8-WE9-EIQQ), and "Installer's name" (Alina Forstsmyth). At the bottom is an orange "Create Network" button.

## Products used

- MS | MR | MX

## APIs used

- Dashboard API

## Built by

- Customer

# “At-Home” IoT Wireless Experience for Dorms

## CUSTOMER PROFILE

Private research university	17,500+ students	4 campus locations
-----------------------------	------------------	--------------------

## CHALLENGE

- Easily onboard student IoT devices that speak to each other in dorm with registration portal
- Provide seamless IoT support with Tier 1 management portal

## SOLUTION

- Tech partner developed device registration and Tier 1 management portals utilizing the Meraki dashboard and captive portal APIs
- Students easily onboard and manage their IoT devices with devices grouped to communicate with each other
- Tier 1 management portal for Georgetown network services team to support/troubleshoot



## Products used

- MR

## APIs used

- Dashboard API
- Captive portal API

## Built by

- Technology partner

# The Meraki Digital Workplace

## CHALLENGE

- Meraki IT wanted a fully automated provisioning solution for Teleworker kits for Meraki employees
- Hundreds of devices need to be provisioned
- The dashboard GUI is great, however the task would take too many clicks

## SOLUTION

- Automated provisioning using the dashboard API
  - Meraki devices need to be added to inventory
  - Each teleworker kit requires its own dashboard network
  - Lock down the network config. by binding to a template
  - Bind the network to a specific device from the inventory



```
118 # Claim device into network
119 response = conn.post do |request|
120   request.url "/api/v0/networks/#{network_id}/devices/claim"
121   request.headers['X-Cisco-Meraki-API-Key'] = "#{dash_api_key}"
122   request.headers['Content-Type'] = 'application/json'
123   request.body = '{"serial\\":\\"#{serial}\\"}'
124 end
```

### Products used

- Z1 / Z3 teleworker

### APIs used

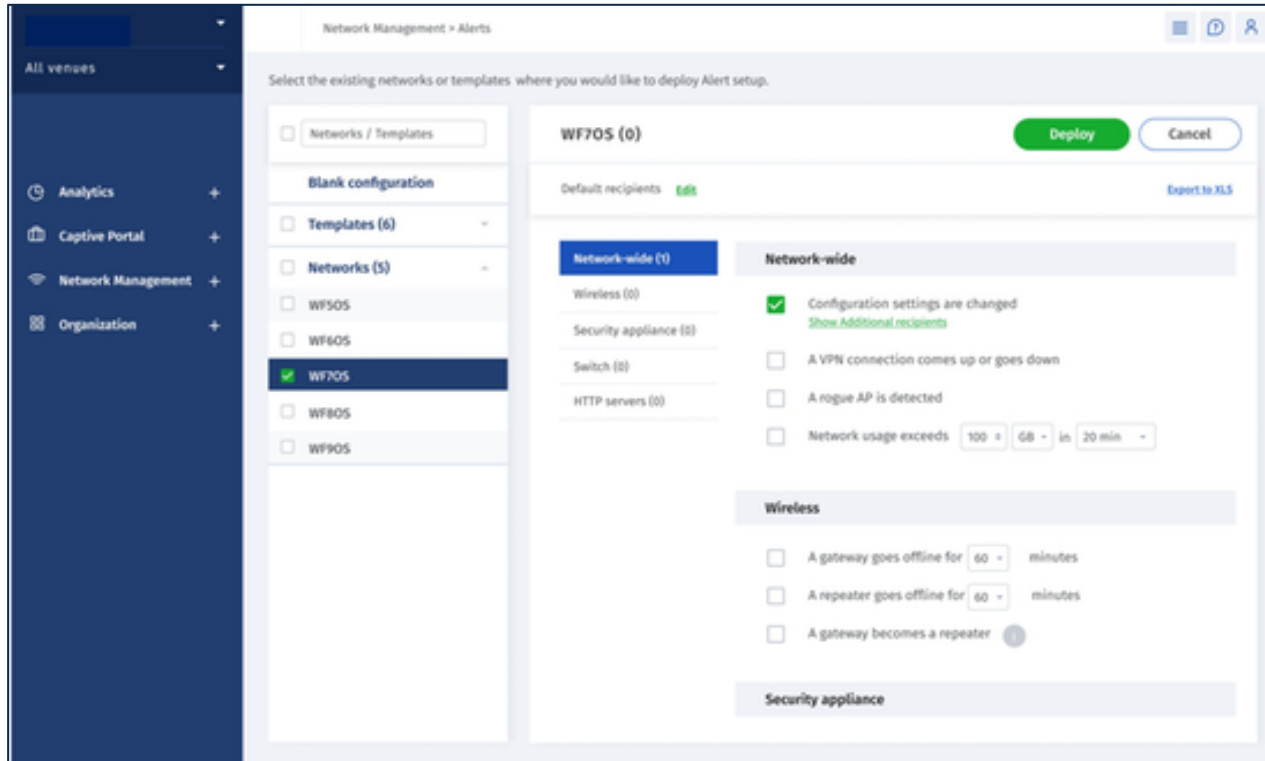
- Dashboard API

### Built by

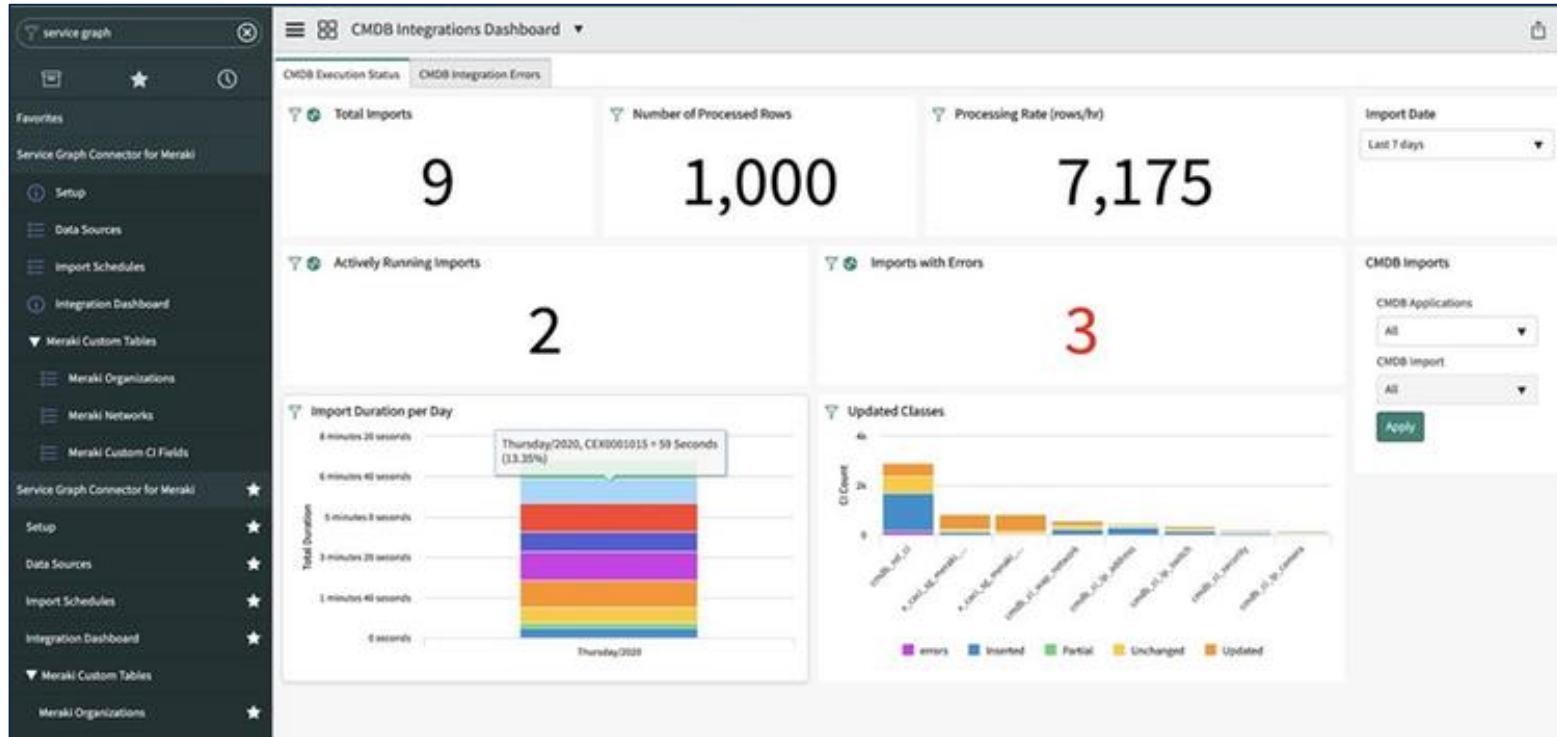
- Meraki IT



# Boundless Automation



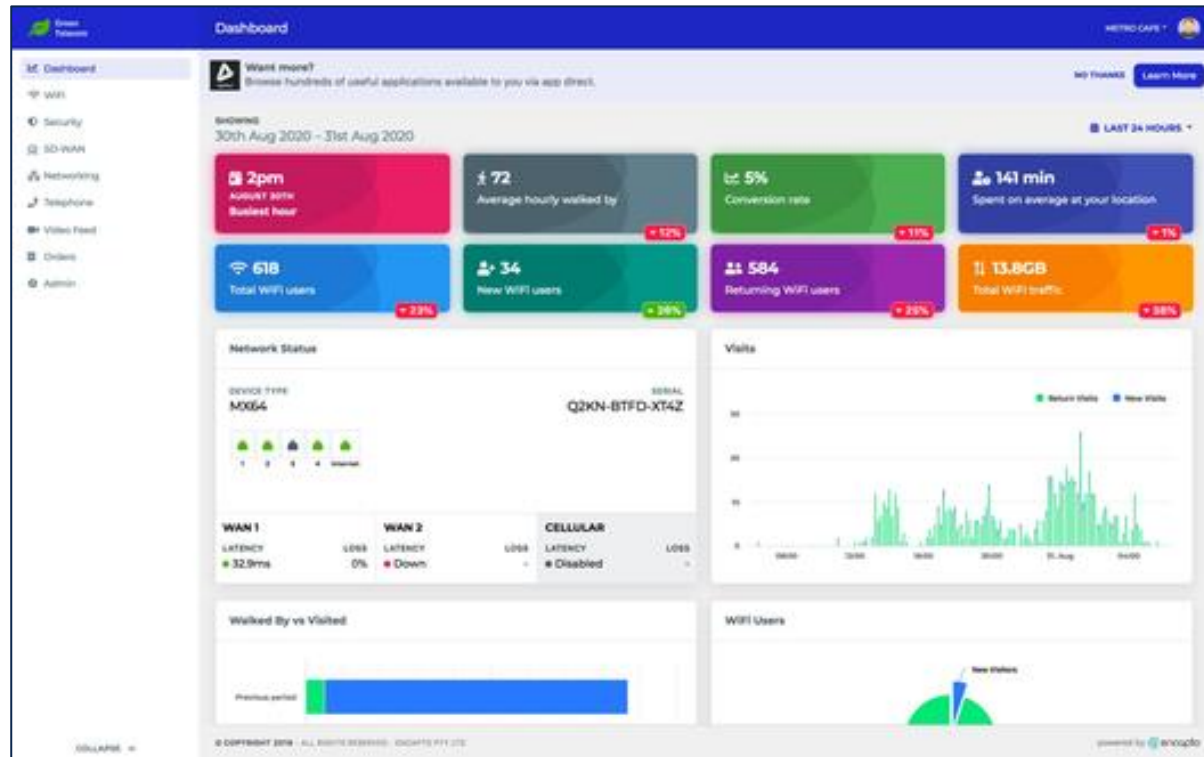
# Service Now Connector



# Auvik Networks



# Encapto



CISCO *Live!*

ALL IN