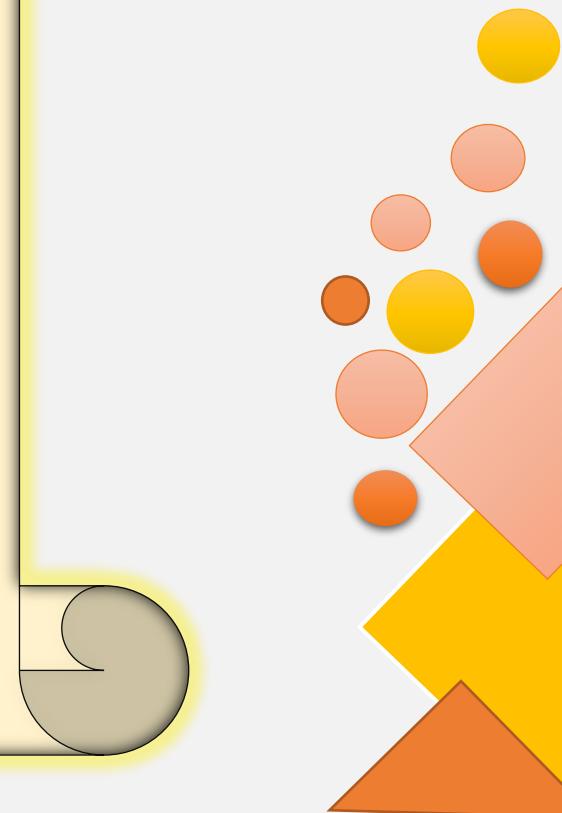


# **Software Defined Networking {SDN}**

**with Demonstration of Ansible**

**Automation Tool**



# الفهرس:

- 2 - فكرة المشروع (Project Idea)
- 3 - أهداف المشروع (Project goals)
- 3 - مقدمة
- 4 - المهام الأساسية (المتطلبات الوظيفية) التي ستقدمها تقنية SDN باستخدام أداة Ansible
- 5 - بعض الخصائص والتعديلات التي يمكن إضافتها لاحقاً في الإصدارات القادمة من هذه التقنية
- 5 - الدراسة التحليلية والتصميمية للمشروع
- 6 - Introduction to Controller-based Networking
- 8 - The Data, Control, and Management Planes
- 12 - Data Serialization / Modeling Language
- 22 - APIs Application Programming Interface
- 26 - (Restful API) Representational State Transfer API
- 33 - SOAP – Simple Object Access Control
- 31 - Examples of Network programmability and SDN
- 34 - How Automation Impacts Network Management
- 38 - SDN Solution
- 40 - Cisco Application Centric Infrastructure – ACI

- 43 -	Cisco APIC Enterprise Model
- 44 -	SD – WAN
- 45 -	SD-WAN Architecture
- 46 -	SD-ACCESS Fabric
- 53 -	Summary Of SDN Examples
- 54 -	Configuration Management
- 62 -	Comparing Ansible, Puppet, Chef
- 62 -	التطبيق العملي لتقنية SDN باستخدام أداة ANSIBLE

## :Project Idea

تقوم فكرة هذا المشروع على مناقشة تقنية الشبكات المعرفة ببرمجياً (SDN)، التي منحت إمكانيات كبيرة في إدارة الإعدادات ومراقبة موارد التجهيزات الشبكية وخاصة تجهيزات شركة CISCO، مع شرح عن استخدام الأدوات المناسبة لكل نوع من أنواع الشبكات الداخلية أو الخارجية مع مراعاة حجم الشبكة. الأمر الذي يسمح لمدير الشبكة بأتمتة عمليات يدوية دورية وتفادي أي أخطاء في الشبكات الضخمة بسبب الاعتماد على العنصر البشري.

## أهداف المشروع : (Project goals)

- تقديم شرح بسيط وشامل لتقنيات الشبكات المعرفة برمجياً مع ذكر البروتوكولات التي يتم الاعتماد عليها في هذه التقنية وطريقة ربط الأجهزة الشبكية بالمتحكمات وأشهر الأدوات المستخدمة في كل نوع من أنواع الشبكات القابلة لتطبيق هذه التقنية.
- المقارنة بين طريقة إدارة الشبكات التقليدية (يدوياً - CLI)، والطريقة الحديثة المعرفة برمجياً والمعتمدة على المركزية في الإدارة باستخدام متحكمات (Controllers).
- ذكر متطلبات استخدام المتحكمات في كل نوع من أنواع الشبكات.

توضيح كل ما سبق بمثال عملي باستخدام برنامج GNS3، عن طريق إنشاء سيناريو لربط عدة تجهيزات شبكة ومكتبة مع أداة Ansible التي ستستخدم كمتحكم في الشبكات الصغيرة والمتوسطة الداخلية والتي تعتمد على لغة Yaml.

## مقدمة:

في الوقت الحالي نشهد تطوراً كبيراً ومتسراً في مجال التكنولوجيا وذلك لمواكبة المتطلبات المتزايدة والمختلفة في شتى المجالات التي أصبحت معتمدة اعتماد كلي أو شبه كلي على التكنولوجيا.

لذلك كان من الضروري لمجال الشبكات أن يواكب هذا التطور والعمل على تحسين وزيادة كفاءة عمل الشبكات من خلال السعي لتطوير بنيتها وتجهيزاتها وجعلها أكثر سهولة من حيث الإدارة والبرمجة، حيث منذ عام 1980 لم يطرأ أي تغيير على البنية التحتية للشبكات لتتلاءم مع التطور الكبير الحاصل في مجال تكنولوجيا المعلومات حتى عام 2004 بدأت تظهر المساعي للتغيير البنية التحتية للشبكات التقليدية.

منذ حوالي العشر سنوات تقريباً ظهر مفهوم **SDN (Software Defined Network)** بشكله الكامل والذي يسعى من خلاله المهندسين إلى إعادة ترتيب أجزاء ومكونات البنية التحتية للشبكات بحيث يتم فصل **Data plane** عن **Control plane**.

## فيما يلي سنعرض المهام الأساسية (المطلبات الوظيفية) التي ستقدمها تقنية SDN باستخدام أداة ..Ansible

{Odom, 2019 #1}

- i. تجزأ الشبكة إلى ثلات أجزاء (طبقات) رئيسية وهي .  
  - . **Data Plan** : تضم أجهزة الشبكة (Switches and Routers).
  - . **Control Plan** : وهي الطبقة المسؤولة عن إعطاء الأوامر والتحكم بالشبكة.
  - . **Application Plan** : وهي الطبقة الخاصة بالتطبيقات التي تقوم بعمل implementation of services وت تكون من الخدمات والتطبيقات التي تقدمها إلى المستخدم (network admin).
- ii. جعل مهمة الراوتر والسويتش هي توجيه البيانات فقط (forward) مما يساعد على تخفيف أعباء الإدارة والتحكم بعمليات التوجيه (routing) من على الراوتر.
- iii. سهولة وдинاميكية التعديل وإضافة أي إعدادات جديدة من خلال وضعها على Controller ثم إرسالها للأجهزة المطلوبة.
- iv. حل مشكلة ال Configuration Drift والتي يقصد بها ابتعاد الإعدادات الحالية للأجهزة الشبكية بشكل كبير عن الإعدادات الأصلية التي تم الاتفاق عليها من قبل مديرى الشبكة عند بدء وضع الجهاز في الخدمة.
- v. اختصار الكثير من الوقت في عمليات ال Troubleshooting الخاصة بالشبكة.
- vi. الاعتماد على بروتوكول SSH في الاتصال بين الأجهزة الشبكية والمتحكم Controller, الذي يسمح بنقل آمن للبيانات ويوفر التعامل مع كافة التجهيزات الشبكية بغض النظر عن إصداراتها.
- vii. إمكانية اختيار الجهاز أو مجموعة الأجهزة المراد تطبيق الإعدادات الجديدة عليها عوضاً عن إرسالها لجميع الأجهزة في الشبكة.

- viii. التأكد من تقبيل الأجهزة للإعدادات المدفوعة إليها قبل تطبيقها.
- ix. عدم تطبيق الإعدادات المدفوعة إلى الأجهزة المختارة بحال كانت الإعدادات مطبقة سابقاً.
- x. المقدرة على حفظ ملفات الإعدادات في قوالب (Templates) واستخدام متغيرات (Variables) من أجل تطبيقها على نفس الأجهزة المتشابهة في الشبكة (حيث يتم استخدام قالب موحد مع تغيير قيم المتغيرات بما يناسب التجهيزات المراد تطبيق القالب عليها).

## تاليًا سنقدم بعض الخصائص والتعديلات التي يمكن إضافتها لاحقًا في الإصدارات القادمة من هذه التقنية... {Odom, 2019 #1}

- ✓ القدرة على جدولة تنفيذ أو دفع الإعدادات إلى التجهيزات الشبكية ضمن أوقات زمنية معينة وجعلها مؤتمته بدون تدخل مهندس الشبكة.
- ✓ إضافة خاصية التحقق من المفاتيح المستخدمة في اتصال ال SSH بين المتحكم والتجهيزات الشبكية لإضافة أمان أكبر لعمليات الاتصال.
- ✓ دمج أعمال ال programing routing and IP security and networking responsibility مهندس الشبكات مسؤول عن عمليات addressing .firewall rules
- ✓ برمجة Control server على إعطاء تقارير دورية عن Transaction of network .
- ✓ ضبط routers and switches لإصدار تنبيةات عند حصول أي تعديل عليها غير صادر من Controller .

الدراسة التحليلية والتصميمية للمشروع

## *1- Introduction to Controller-based Networking:* {Mobley., 2022 #5}

في شبكات ال SDN تقوم ال Switches, Routers بنفس عمليات التوجيه والاستلام و ... ولكن تختلف الطريقة والسبب بحيث يصبحون أسرع وأكثر فعالية في شبكات ال SDN، حيث يتم الاعتماد على ال Controllers لضبط التجهيزات الشبكية، يعتمد على {APIs – Application Programming Interfaces} للتواصل مع التجهيزات الشبكية بينما يتم استخدام SSH, SNMP للتواصل بين ال Admin والتجهيزات الشبكية.

### - *Controllers and Software Defined Architecture*: {Lessons., 2016 #8}

في الحالة التقليدية كان كل جهاز يعمل بشكل مستقل بأجزائه الرئيسية الثلاثة (Data- Control-Management Plane)، تسمى هذه الحالة ب **Distributed Control Plane** لذلك غير مناسب للشبكات الكبيرة للتغلب على هذه المشكلة يتم استخدام جهاز مركزي واحد أو أكثر يسمى **SDN Controller**.

مهمته التحكم بالأجهزة الأخرى والأشراف عليها حيث أن ال Controller هو عبارة عن Software منصب على VM، أو هذا الجهاز يتعامل بشكل مباشر مع قسم Date Plane الموجود في جهاز السويفت او الراوتر من أجل تحديث جداول Physical Server .TCAM Tables

عملية التواصل بين جهاز SDN Controller وأجهزة السويفت والراوتر يتم عن طريق وسيط **SBI SouthBound Interface** التي تحتوي على <APIs – Application Programming Interface>.

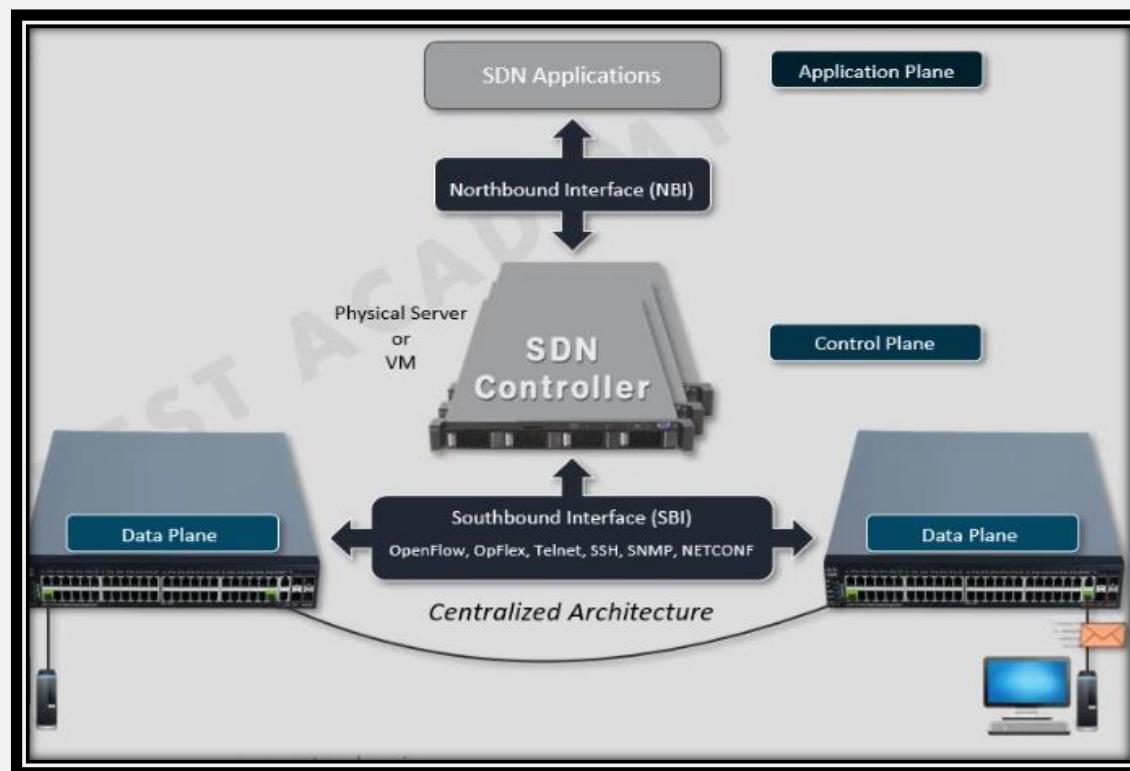
حيث تسمح ال APIs بتواصل البرنامجين (Software) Network Devices بالController (بيئتين مختلفتين).

{WENDELL, 2020 #3} **من أشهر ال APIs المستخدمة ضمن ال SB**

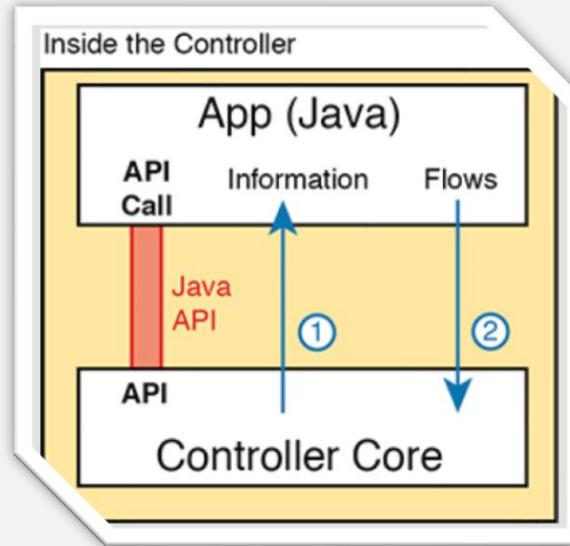
- OpenFlow: (from ONF).

- OpFlex: (from Cisco, used with ACI).
- CLI (SSH/Telnet) and SNMP (Used with Cisco APIC-EM).
- CLI (SSH/Telnet) and SNMP, and NETCONF (Used with Cisco SD-Access).

في حال نجاح عملية التواصل فإنه يقوم بجمع معلومات كأسماء وعناوين الأجهزة والمنافذ الموجودة فيها وحالات المنافذ وكيفية اتصال المنافذ مع بعضها ويرسم صورة عن الشبكة بعد جمع هذه المعلومات يمكن له ان يوفرها لمجموعة من التطبيقات الموجودة فيه من اجل اجراء برمجة لأجهزة الشبكة او لتطبيقات منفصلة عنه متصلة معه عبر منفذ يسمى (NBI NorthBound Interface)، حيث عند اتصالها وحصولها على المعلومات تستطيع اجراء برمجة للشبكة التي تحتوي على APIs تسمح بالتواصل بين ال Controller و Software Management الخاصة به.



مثال: في حال كان الـ Software الخاص بال Controller مكتوب بلغة Java ويحتوي على API مكتوب بلغة Java. يعمل على أي تطبيق Java يتعامل مع أجهزة مدير الشبكة، مما يتيح التبادل البينات مع ال Controller.



يمكن أن تحتوي التجهيزات الشبكية على جزء من ال Control Plane خاص بها، ليصبح مخطط الشبكة من نوع Hybrid SDN ويستخدم هذا النمط لإبقاء التجهيزات قابلة للتحديث динамически لـ Tables الخاصة بال Control Plane وقادرة على اتخاذ قرارات بنفسها في حال انقطاع الاتصال بين التجهيزات وال Controller.

## 2- The Data, Control, and Management Planes: {WENDELL, 2020 #3}

الأجزاء الرئيسية الموجودة في الشبكات التقليدية:

## The Data Plane

يطلق عليها أيضا Forwarding Plane تتضمن هذه الطبقة جميع مهام التجهيزات الشبكية المتعلقة بإرسال الرسائل، معالجتها، واستلامها (من جميع الطبقات، L2 – L3)

Data-link frame ضمن ال De-Encapsulating, Encapsulating Packets (1)

Adding, Removing 802.1Q Header (2)

Matching Destination MAC Address in Datalink to the MAC Address table (3)

Matching Destination IP Address in Packets to IP Address in Routing Table (4)

Encrypting Data, and Adding new VPN Header (5)

Changing Source or Destination IP Address in NAT Processing (6)

(ACL, Port Security) Discarding a message due to filters (7)

## The Control Plane

يعتبر الجزء الذي يلعب دور الدماغ لجهاز الراوتر او السويفت يتضمن هذا الجزء جميع العمليات الخاصة بتشكيل ال Tables التي تعتمد عليها ال Data Plane ذلك بعد تبادل المعلومات عن الراوتر والأجهزة الموجودة في الشبكة بشكل تلقائي ذلك باستخدام أحد برامج التوجيه ويحدد المنافذ من نوع forwarding والمنافذ من نوع blocking (مثل... Routing Table, MAC Table, ARP Table...) في هذا الجزء وكل العمليات التي تتم فيه يعتمد بشكل أساسى على المعالج والذاكرة الموجودين في كل راوتر او سويفت في الحالة التقليدية كانت كل تجهيزه تعمل بالطبقتين (Data, Control)، تسمى هذه الحالة ب **Distributed Control Plane**

من أهم البروتوكولات التي تعمل ضمن هذه الطبقة:

- Routing Protocols (OSPF, EIGRP, RIP, BGP)
- IPv4 ARP
- IPv6 NDP

- Switch MAC Learning
- STP

## *The Management Plane*

تتضمن بروتوكولات تستخدم لإدارة التجهيزات الشبكية والتحكم في الأجهزة والاشراف عليها، مثل: SSH, Telnet, SNMP, Syslog



### - Cisco Switch Data Plane Internals: {CISCO., 2022 #6}

تحتاج الـ Switch لعتاد قوي لمعالجة الكم الكبير من الـ Frames، لحساب عدد الـ Frames التي ممكن معالجتها في الـ Switch في الثانية الواحدة:

نقوم بحساب عدد ال ports الممكن حسابها لكل port ثم ضربها بعدد ال ports الموجودة كما يلي:

حيث كمثال في حال وجود Switch تحتوي على 24 port ذات سرعة (1,000,000,000 bits/sec = 1Gbps)

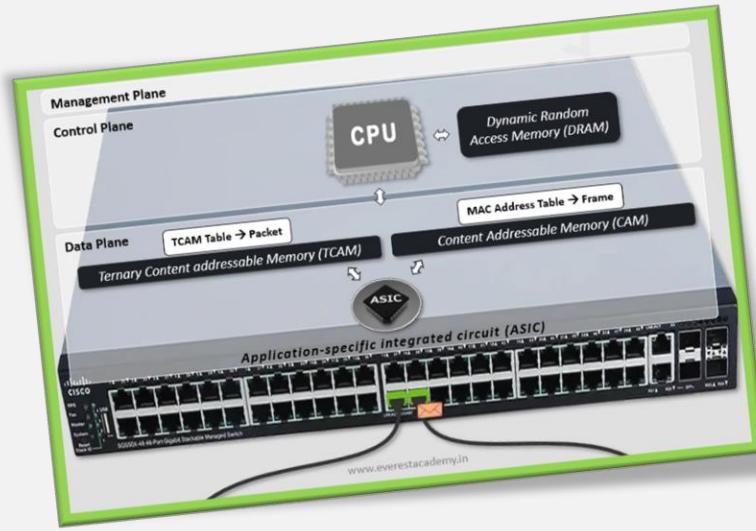
ففي حال كان المعدل الوسطي لحجم ال Frame 1,000 Bit = 125 Byte ، يكون بالإمكان معالجة 1,000 fps على كل Port وبشكل إجمالي 24 million fps في ال Switch على جميع Ports ، لذلك يتم الاستعانة ب switches خاص بال Hardware للقيام بهذه العمليات، حيث أن الاعتماد على المعالجة الخاصة بال CPU (Software) غير كافية.

## ASIC – Application-Specific Integrated Circuit:

أي جهاز سويفت يحتوي على شرائح الكترونية قريبة من المنافذ تسمى ASIC مخصصة لمعالجة البيانات في طبقة data plane بسرعات عالية وتعامل مع ذواكر خاصة قابلة لتخزين معلومات على شكل جداول تساعد في تحديد المنفذ التي يجب إرسال البيانات منها عند استقبالها من قبل جهاز السويفت، وهذه الذواكر تصنف إلى قسمين:

TCAM – Ternary Content Addressable Memory	CAM – Content addressable Memory
يتم بناء جدول يسمى TCAM TABLE من أجل تخزين عناوين IP – PORT NUMBER – QS حيث يساعد في تمرير PACKET CONTROL PLANE	يتم بناء جدول MAC ADDRESS من أجل تخزين عناوين MAC يتم بنائه بشكل تلقائي لأجهزة الكمبيوتر التي ترسل بيانات عبر جهاز السويفت حيث يستخدم لتمرير FRAME

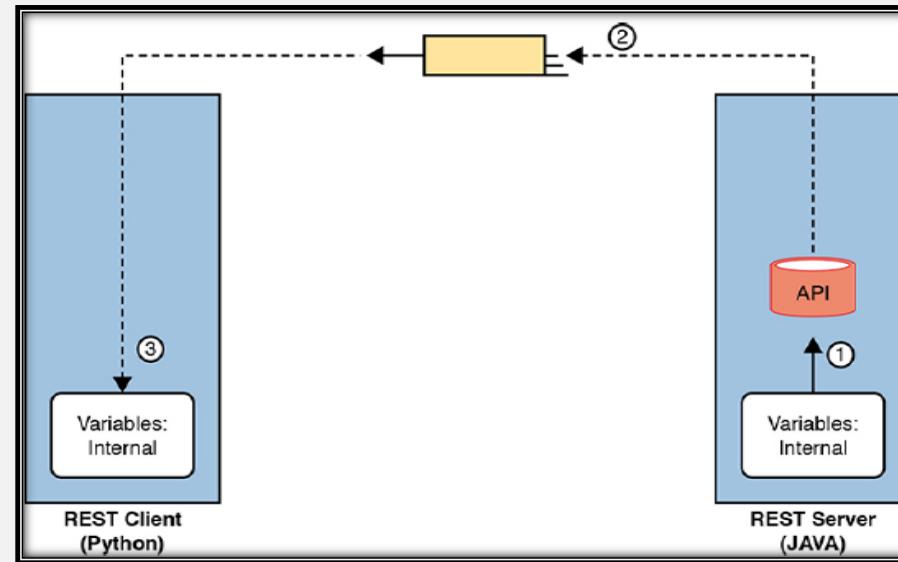
حيث يتم استثمار ال ASIC لتوفير معالجة سريعة ل Data Plane وال TCAM لتوفير استعلام أسرع من ال Tables المحفوظة بداخلها .(ARP,MAC-address, Routing Table..)



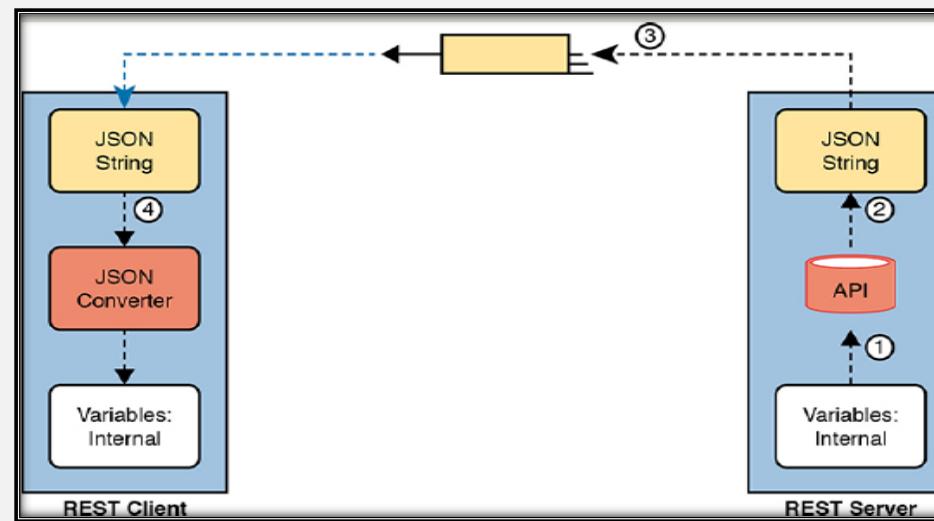
### 3- Data Serialization / Modeling Language: {Odom, 2019 #1}

من أجل نقل معلومات الإعدادات بين جهاز Controller وأجهزة الشبكة والتطبيقات يتم استخدام لغات خاصة تسمى (DSL) من أهم الـ Data serialization Language هي: (XML, Yaml, Json) حيث كل لغة لها طريقة خاصة لتمثيل المعلومات بناء على القواعد الموجودة في كل لغة، كل لغة من هذه اللغات توفر **Structure**، **Methods** خاصة بها لتمثيل الـ **Data Variables** هي عملية تحويل الـ Data إلى نمط قابل للمشاركة والتخزين وبنفس الوقت قابل للاسترجاع لنمطه الأصلي وتستخدم للتواصل بين بيئتين مختلفتين مثل: .Python - Java

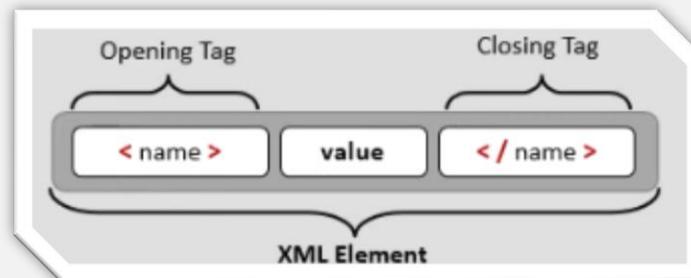
حيث من دونها لن يكون بالإمكان لتطبيقيين مبني على لغتين مختلفتين التواصل مع بعضهما البعض مثل:



ولكن استخدام إحدى الـ Data Serialization Languages (مثل ال Json) سيحل المشكلة:



- ↳ تستخدم بشكل أساسي في موقع ال Web وليست سهلة القراءة من البشر.
- ↳ تقوم بتمثيل المعلومات باستخدام عناصر تسمى Element حيث كل عنصر مؤلف من ثلاثة أجزاء:
  - 1: يحتوي على اسم العنصر يحجب أن يكون له معنى حسب نوع البيانات التي يحملها.
  - 2: يحتوي على المعلومات التي سيتم نقلها.
  - 3: يدل على نهاية العنصر.



مثال عن خرج تعليمية تستخدم لغة XML

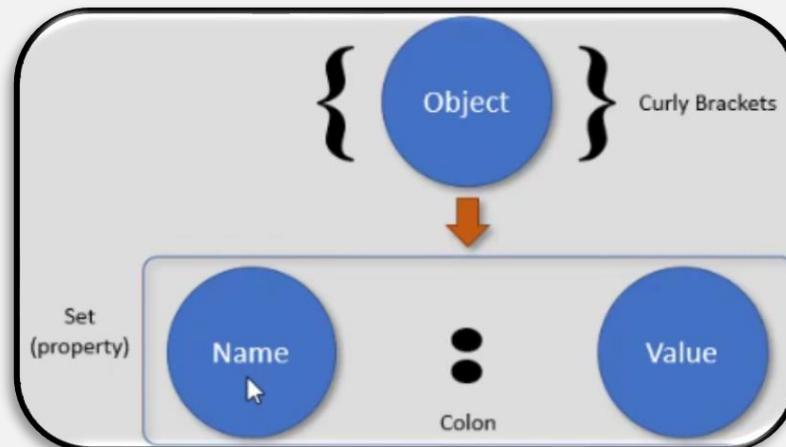
```
<?xml version="1.0" encoding="UTF-8"?>
<ShowIpInterfaceBrief xmlns="ODM://built-in//show_ip_interface_brief">
  <SpecVersion>built-in</SpecVersion>
  <IPInterfaces>
    <entry>
      <Interface>FastEthernet0/0</Interface>
      <IP-Address>192.168.1.1</IP-Address>
      <OK>YES</OK>
      <Method>NVRAM</Method>
      <Status>up</Status>
      <Protocol>up</Protocol>
    </entry>
    <entry>
      <Interface>FastEthernet0/1</Interface>
      <IP-Address>192.168.2.1</IP-Address>
      <OK>YES</OK>
      <Method>NVRAM</Method>
      <Status>up</Status>
      <Protocol>up</Protocol>
    </entry>
  </IPInterfaces>
</ShowIpInterfaceBrief>
```

{WENDELL, 2020 #3} **JavaScript Object Notation (JSON)**

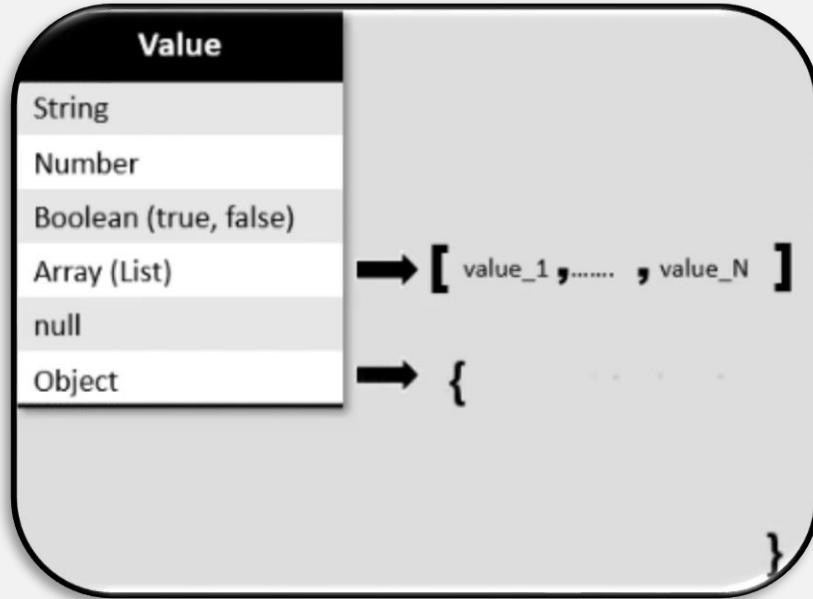
تستخدم غالبا مع ال JavaScript وال APIs لأنها سهلة للقراءة من البشر وسهلة التحويل ل Variables يفهمها الكمبيوتر أكثر من ال XML ، مع تطور استخدامات الويب وظهور الحاجة لإيجاد أسلوب لنقل البيانات من وإلى مستعرض الويب من دون استخدام الملحقات ك ( Adobe Flash Player ) و ( Java Applets ) والذي كان الأسلوب الشائع لإنجاز هذا التبادل.

فتم إيجاد صيغة JSON في بدايات عام (2000) وهو تنسيق نصي مستقل للبيانات مستمد من لغة JavaScript، وفي الوقت الحالي تتضمن العديد من لغات البرمجة الشائعة طرق وشفرات لإنشاء بيانات بصيغة JSON وتحويلها.

ال (JSON) اختصار (JavaScript Object Notation) هي صيغة لهيكلة البيانات بطريقة أكثر اختصار وتكون - البيانات - سهلة القراءة والفهم بالنسبة للإنسان، عادة ما يتم استخدام هذه الطريقة (JSON) لتسهيل نقل البيانات بين جهاز الخادم والعميل وبالعكس تستخدم في عملية التمثيل وتعتمد على **Object** يتم وضعه ضمن قوسين كبيرين ويتألف من يتألف من جزء يسمى **Set** كل **Set** يتتألف من جزئين **Name**: يعطى كاسم للمعلومات او البيانات ونستطيع تكرار الاسم، **Value**: تحمل البيانات ذاتها يمكن ل **Object** ان يحتوي أكثر من قيمة بوضع فاصلة بين كل قيمتين باستثناء آخر قيمة.



بالنسبة للقيم التي يمكن ان يحملها **Value** ( String – Name – Boolean – Array – Null – Object)



مثال لخرج تعليمية من نوع :Json



```

"ShowIpInterfaceBrief": {
    "SpecVersion": "built-in",
    "IPInterfaces": [
        "entry": [
            {
                "Interface": "FastEthernet0/0",
                "IP-Address": "192.168.1.1",
                "OK": "YES",
                "Method": "NVRAM",
                "Status": "up",
                "Protocol": "up"
            },
            {
                "Interface": "FastEthernet0/1",
                "IP-Address": "192.168.2.1",
                "OK": "YES",
                "Method": "NVRAM",
                "Status": "up",
                "Protocol": "up"
            }
        ]
    }
}

```

المسافات الفارعة (Whitespaces) ليس لها معنى ولا تشكل تغيير وعند استخدام مسافات فارغة (Spaces) في ملف Json يطلق على نسخته اسم (Pretty, beautiful, spaced) تعتبر النسخة الأسهل للقراءة من قبل المستخدم وعند عدم استخدام مسافات في ملف Json، يطلق على نسخته اسم (minified, raw)، Json لا تدعم التعليقات.

### {yamlchecker., #11} *Ain't Markup Language (YAML)*

بسطة جداً بعكس ال XML حيث لا تحدد ال Font, Color, Size...، تستخدم بشكل أساسى مع ال Perl, Python، تستخدم مع ال Network Automation Tool، يتم تعريف ال Variables داخل double curly braces ( { }{}) سهلة القراءة جداً من Ansible

البشر ( تستخدم Wide Spaces – فراغات لها معنى, مسافات من بداية كل سطر لتمييز الفقرات عن بعضها ) حيث تبدأ ملفات الـ YAML بـ (---) و (-) واحدة تعني بداية List.

مثال عن YAML file :

```
root [1]
ShowIpInterfaceBrief:
SpecVersion: built-in
IPInterfaces:
entry:
- Interface: FastEthernet0/0
  IP-Address: 192.168.1.1
  OK: 'YES'
  Method: NVRAM
  Status: up
  Protocol: up
- Interface: FastEthernet0/1
  IP-Address: 192.168.2.1
  OK: 'YES'
  Method: NVRAM
  Status: up
  Protocol: up

root [1]
ShowIpInterfaceBrief [2]
SpecVersion: built-in
IPInterfaces [1]
entry [2]
  0 [6]
    Interface: FastEthernet0/0
    IP-Address: 192.168.1.1
    OK: YES
    Method: NVRAM
    Status: up
    Protocol: up
  1 [6]
    Interface: FastEthernet0/1
    IP-Address: 192.168.2.1
    OK: YES
    Method: NVRAM
    Status: up
    Protocol: up
```

وللمقارنة بين اللغات الثلاثة:

NETCONF XML -1

API Jason -2

ANSIBLE YAML -3

```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <ShowIpInterfaceBrief xmlns="ODM://built-in//"
3   show_ip_interface_brief>
4     <SpecVersion>built-in</SpecVersion>
5     <IPInterfaces>
6       <entry>
7         <Interface>FastEthernet0/0</Interface>
8         <IP-Address>192.168.1.1</IP-Address>
9         <OK>YES</OK>
10        <Method>NVRAM</Method>
11        <Status>up</Status>
12        <Protocol>up</Protocol>
13      </entry>
14      <entry>
15        <Interface>FastEthernet0/1</Interface>
16        <IP-Address>192.168.2.1</IP-Address>
17        <OK>YES</OK>
18        <Method>NVRAM</Method>
19        <Status>up</Status>
20        <Protocol>up</Protocol>
21      </entry>
22    </IPInterfaces>
23  </ShowIpInterfaceBrief>
24
25
26
27
28

```

```

1 {
2   "ShowIpInterfaceBrief": {
3     "SpecVersion": "built-in",
4     "IPInterfaces": {
5       "entry": [
6         {
7           "Interface": "FastEthernet0/0",
8           "IP-Address": "192.168.1.1",
9           "OK": "YES",
10          "Method": "NVRAM",
11          "Status": "up",
12          "Protocol": "up"
13        },
14        {
15          "Interface": "FastEthernet0/1",
16          "IP-Address": "192.168.2.1",
17          "OK": "YES",
18          "Method": "NVRAM",
19          "Status": "up",
20          "Protocol": "up"
21        }
22      ]
23    }
24  }
25 }

```

```

1 ShowIpInterfaceBrief:
2   SpecVersion: built-in
3   IPInterfaces:
4     entry:
5       - Interface: FastEthernet0/0
6         IP-Address: 192.168.1.1
7         OK: "YES"
8         Method: NVRAM
9         Status: up
10        Protocol: up
11       - Interface: FastEthernet0/1
12         IP-Address: 192.168.2.1
13         OK: "YES"
14         Method: NVRAM
15         Status: up
16         Protocol: up

```

## - Comparison of Json and XML: {Edgeworth, 2019 #2} {WENDELL, 2020 #3}

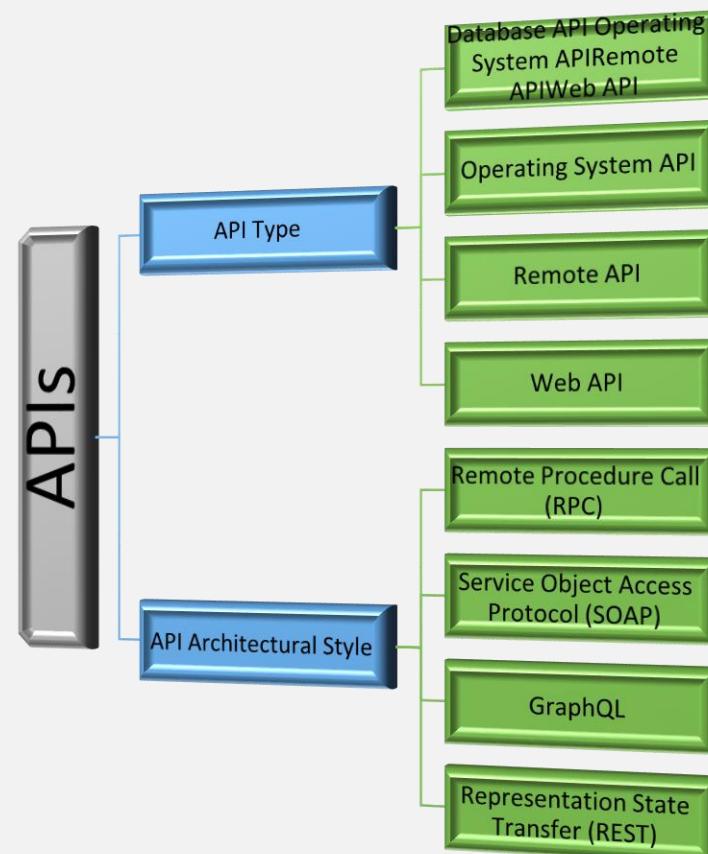
XML	JSON	
Extensible Markup Language	JavaScript Object Notation	اختصار

(SGML) Standard Generalized Markup Language	JavaScript	امتدت من
صيغة تحتوي على مجموعة من القواعد لترميز المستندات القابلة لكل من الإنسان والآلة، تم تطويرها بواسطة.	من التنسيقات المستندة لمعايير تستخدم لتبادل البيانات، تم تطويرها بواسطة (Douglas Crockford).	الغاية
لها وسوم بداية ونهاية وتحتاج لبنية أكبر من JSON) لتمثيل نفس البيانات.	صيغتها أخف من XML) نظراً لأن تنسيق البيانات فيها يحتوي أقل قدر من التكرار فهي لا تحتوي على وسوم البدء والانتهاء.	الصيغة
ليست كسرعة JSON	أسرع في النقل من XML	السرعة
لا توفر أي نوع بيانات لذلك يجب تحليلها كنصوص وتحويلها للنوع.	تدعم نوع البيانات (Data type) بما في ذلك عدد صحيح وسلسل ومصفوفات ولكن يبقى الدعم لأنواع محددة.	دعم أنواع البيانات
يمكن لها الحصول على دعم الكائن من خلال الاستخدام المختلط للسمات والعناصر. يدعم التعليقات.	لديها دعم كائن. لا يدعم التعليقات.	دعم كائن التعليقات
لا يوجد دعم لنطاق الأسماء.	لا يوجد دعم لنطاق الأسماء.	نطاق الأسماء
تعتمد على مفهوم المستند لذلك تحتاج لعمليات متعددة.	تستند لمفهوم الكائن لذلك هي أسرع.	الهيكلة
Extensible Stylesheet Language يمكن تغيير بياناتها إلى تنسيق آخر مثل النص العادي أو JSON	لا يمكنك تغيير بياناتها إلى تنسيق آخر.	تحويل لصيغ أخرى
باستخدام XPath من الممكن الحصول على الوصول المباشر إلى جزء معين من بيئة البيانات.	لا يوجد الوصول المباشر إلى جزء معين في بيئة البيانات.	الوصول المباشر للبيانات

## *4 APIs Application Programming Interface:* {Edgeworth, 2019 #2} {Lessons., 2016 #8}

هي نافذة برمجية مبرمجة بلغة معينة تساعدنا بنقل معلومات معينة بين البرامج وبين الأجهزة المختلفة التي يجب ان تكون متصلة فيما بينها وتقسم الى قسمين:

- 1- الموصفات التي يشرح ويوضح طريقة عمل API
- 2- الكود البرمجي الذي يتم كتابته بشكل يتوافق مع هذه الموصفات ويكتب من قبل مبرمجين مختصين في APIs.



## **:REST-Base APIs**

هي طريقة تسمح بتشغيل برامج وتطبيقات موجودة في أجهزة مختلفة ومنفصلة ولكن متصلة فيما بينها تستخدم بروتوكولات معينة لنقل البيانات فيما بينها ولبناء المخطط العام ل API يتتألف من:

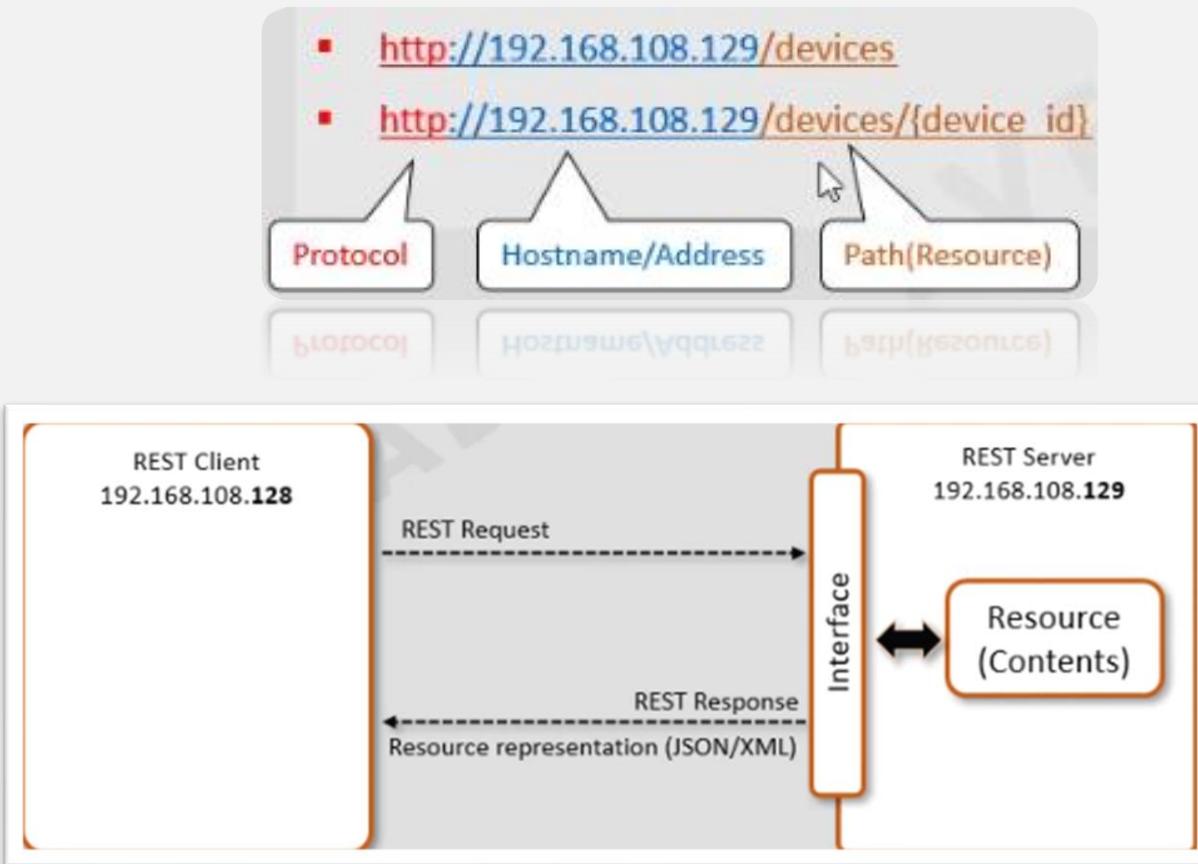
- 1 **Rest Server**: يجب وجود جهاز سيرفر عادة يسمى Rest Server.
- 2 **Rest Client**: يسمى Rest Client.

-3 **Resource**: البيانات التي تكون في جهاز سيرفر ويمكن لاي جهاز Client الحصول عليها من السيرفر بالتواصل مع Interface.

يرسل جهاز Client بما يسمى REST Request ويりد عليه السيرفر بما يسمى REST Response تكون محمولة ببيانات على شكل (JSON or XML) هذه العملية كلها تسمى REST API.

ولبناء نافذة برمجية API من نوع REST يجب ان نحدد الطريق للوصول للمعلومات الموجودة في جهاز السيرفر بإعطاء عنوان يسمى URIs ويتألف من ثلاث أقسام:

- 1 يدل على اسم البروتوكول المستخدم للتواصل.
- 2 اسم أو عنوان السيرفر أو اسم الدومين.
- 3 لتمييز Resource داخل السيرفر لأنه يمكن أن يكون أكثر من Resource.



يمكن ل REST API أن تستخدم بروتوكول HTTP للتواصل بين الأجهزة في هذه الحالة يرسل جهاز Client رسالة HTTP Request ويرد جهاز السيرفر رسالة HTTP Response.

✓ رسالة HTTP Request تتتألف من:  
**Verb**: الفعل التي يمكن لجهاز ال Client القيام به و يطلق عليها مصطلح **CRUD**  
 فعندما نريد إنشاء معلومة ما (Create) نستخدم Post، عندما نريد الاستعلام عن معلومة ما (Read) نستخدم Get، عندما نريد تحديث معلومة ما (Update) نستخدم Patch

.Delete معلومة ما (Delete) نستخدم Put، عندما نريد حذف معلومة ما (Update) نستخدم .  
عندما نريد استبدال معلومة ما (Update) نستخدم Put، عندما نريد حذف معلومة ما (Delete) نستخدم .  
Resource: يمثل عنوان URI  
HTTP Version: يمثل إصدار HTTP  
Request Header: يحتوي على المعلومات التي يجب أن تكون (Key: Value Pairs)  
Body: يحتوي على المعلومات في هذه الرسالة التي يرسلها جهاز Client.  
ملاحظة: رسالة Get لا تحتوي على Body



✓ رسالة HTTP Response تتتألف من:  
Response Code: يحتوي على رقم معين لمعرفة حالة السيرفر  
هناك 5 أنواع أساسية من ال Response Code المستلمة عند التعامل مع ال API:  
Informational: تكون 1xx  
2xx: مثل 200 تعني تم بنجاح, 201 تعني Created, 204 تعني Deleted.  
3xx: تعني Redirection  
4xx: تعني وجود مشكلة من جهة ال Client, مثل 404 تعني عدم وجود الطلب الخاص بالمستخدم, 403 تعني Forbidden, 401 تعني Unauthorized.  
5xx: تعني وجود مشكلة من جهة ال Server

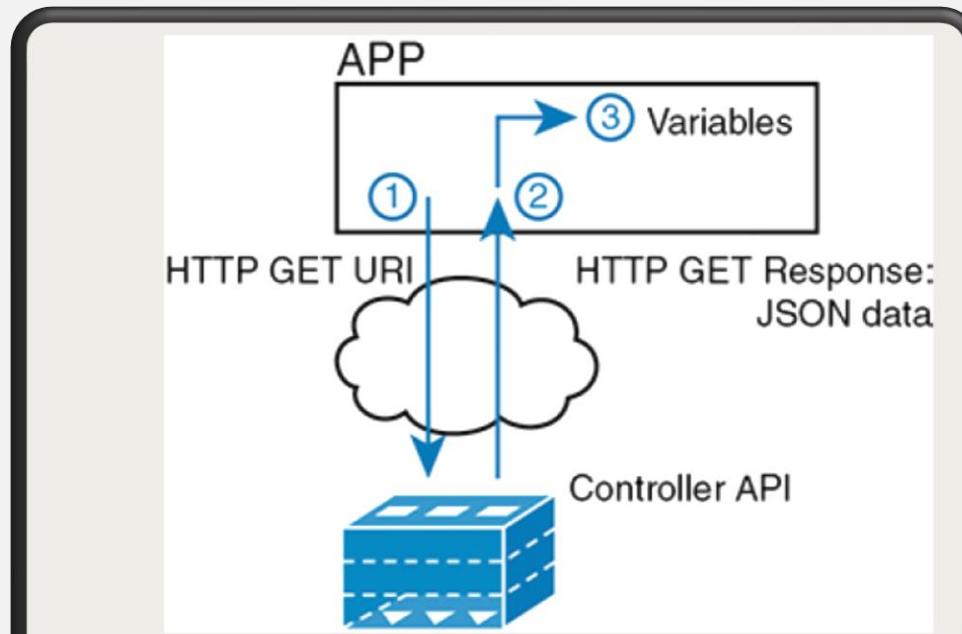
.HTTP Version: خاص بإصدار

.Request Header: يحتوي على المعلومات التي يجب أن تكون (Key: Value Pairs).

.Body: يحتوي على المعلومات في هذه الرسالة التي يرسلها جهاز Client.

## 5- *Representational (State Transfer API) Restful API:* {CISCO., 2019 #9}

هي Architect وليس Protocol خاص بال API يسمح بتوزيع تطبيق معين على عدة Hosts, وتستخدم رسائل HTTP لنقل الرسائل عبر ال API، تستخدم عادة عند تنصيب ال Management Software الخاص بال جهاز غير الجهاز المنصب عليه ال Software الخاص بال Controller، حيث ستحتاج ال API لطريقة لإرسال الرسائل بينهما، وهنا يأتي دور ال Restful API.



**Figure 16-7 Process Example of a GET Using a REST API**

أغلب ال APIs Restful تقوم بطلب وتبادل Structured Data (باستخدام رسائل ال HTTP) عوضاً عن Web Pages، حيث تحتوي ال Software على Variables, their Values على Structured Data.

يقوم ال Software الخاص بال Controller باستلام Variables ومعالجتها ليتم إرسالها كأوامر معينة إلى التجهيزات الشبكية عبر ال SBIs، من أشهر التنسيقات المستخدمة من قبل ال Restful APIS هي Network Programmability في ال JSON, XML.

Client-Server Architecture: يجب وجود Client, Server, Client، حيث يقوم ال Client ب إرسال Request إلى ال server Rest تحوي معينة، ثم يقوم ال Rest Server بإستلام ال Request و ترجمتها ليحدد ال Variable المناسب ل Replay

State-less: لا تقوم باتباع المحادثات السابقة وجهاز السيرفر لا يقوم بتسجيل أي معلومات التي أرسلها لل Client بعد انتهاء عملية التبادل.

Cashable (or not): تستطيع القيام ب Cash (حفظ) لمعلومات معينة لأي Resource بشكل صريح عنها يخبر جهاز Client بان Resource من نوع Cashable

Uniform Interface: هو إحتواء ال API على URI خاص بها

Layard: يمكن إضافة أكثر من جهاز سيرفر بين جهاز السيرفر وجهاز ال client

Code-On-Demand

وفي حال عدم تحققها فإن ال API لن تكون Rest API (تعتبر ال Rest API أسرع وأسهل من ال SOAP) في ال APIs تتم المصادقة باستخدام Token في طلبات ال Get, Request يكون للرد عبارة عن File Replay, Video, ينبع من Web Server من Get, Request ولكن عند الطلب من API يكون الرد JSON (تستخدم أكثر مع ال Restful API)، أو XML غالباً لإدارة DNA Center بواجهة CLI يقوم باستخدام تطبيق اسمه Postman حيث يتم عن طريقه إرسال API Calls تحتوي Request معينة (Get, Update, delete..) إلى ال DNA Center

مثال: عن إرسال طلب Get عن طريق تطبيق Postman إلى ال DNA Center يتم فيه طلب الاستعلام عن ال Network كما مبين في ال parameter في آخر ال URL، ونلاحظ الرد من ال DNA Center كما مبين أسفل ال URL كنتيجة: نرى أن خرج ال API Call Replay الخاصة بال API، هو من نمط JSON (Key : Value)

The screenshot shows the Postman application interface. At the top, there are tabs for 'New', 'Import', 'Runner', and 'My Workspace'. Below that, there are three requests listed: a POST request to 'https://sandboxdnac2.d...', a GET request to 'Intro DNAC APIs - Lab 2, API ...', and the current GET request to 'Intro DNAC APIs - Lab 2, API ...'. The current request is selected and shows the URL 'https://sandboxdnac2.d.../api/v1/network-device'. The response status is '200 OK' with a time of '4239ms' and a size of '17.06 KB'. The response body is displayed in JSON format, showing a large object with many properties, each preceded by a line number from 100 to 310. The properties include 'macAddress', 'opManagerInterfaceIp', 'associatedDlLclp', 'bootDateTIme', 'collectionStatus', 'errorDescription', 'interfaceCount', 'lastUpdated', 'lineCardCount', 'lineCardId', 'locationName', 'managementIpAddress', 'memorySize', 'platformId', 'reachabilityFailureReason', 'reachabilityStatus', 'series', 'snmpContact', 'snmpLocation', and 'tenantName'. The JSON structure is nested, with some properties containing objects or arrays.

**Figure 18-9** URI Structure for REST GET Request

مثال آخر عن خرج من نمط JSON : Replay

**Example 18-3 JSON Output from a REST API Call**

Click here to view code image

```
{
    "response": {
        "type": "Cisco Catalyst 9300 Switch",
        "family": "Switches and Hubs",
        "role": "ACCESS",
        "macAddress": "f8:7b:20:67:62:80",
        "hostname": "cat_9k_1",
        "serialNumber": "FCW2136L0AK",
        "softwareVersion": "16.6.1",
        "upTime": "17 days, 22:51:04.26",
        "interfaceCount": "41",
        "lineCardCount": "2",
        "managementIpAddress": "10.10.22.66",
        "series": "Cisco Catalyst 9300 Series Switches",
        "softwareType": "IOS-XE"
    }
}
```

## CI/CD – Continuous Integrity / Continuous Delivery ↗

هي مجموعة من الضوابط والقيود التي تسمح بتطوير البرامج بطريقة آمنة وعلى عدة مراحل، حيث يعتبر تطوير البرامج على مراحل أفضل من دفع نسخة واحدة كبيرة من التحديثات. ويطبق نفس المبدأ على ال Network Automation، ويطلق على تطبيق هذه التقنية: CI/CD Pipeline

## *6- SOAP – Simple Object Access Control :*

هو بروتوكول من بروتوكولات ال API (مثل ال RestAPI) يسمح لنظامين مختلفين بالتواصل (مثلاً ويندوز - لينكس)، يتم نقله باستخدام XML ونمط ال Data HTTP/HTTPS

## *7- Examples of Network programmability and SDN:* {WENDELL, 2020 #3}

### *OpenFlow Protocol*

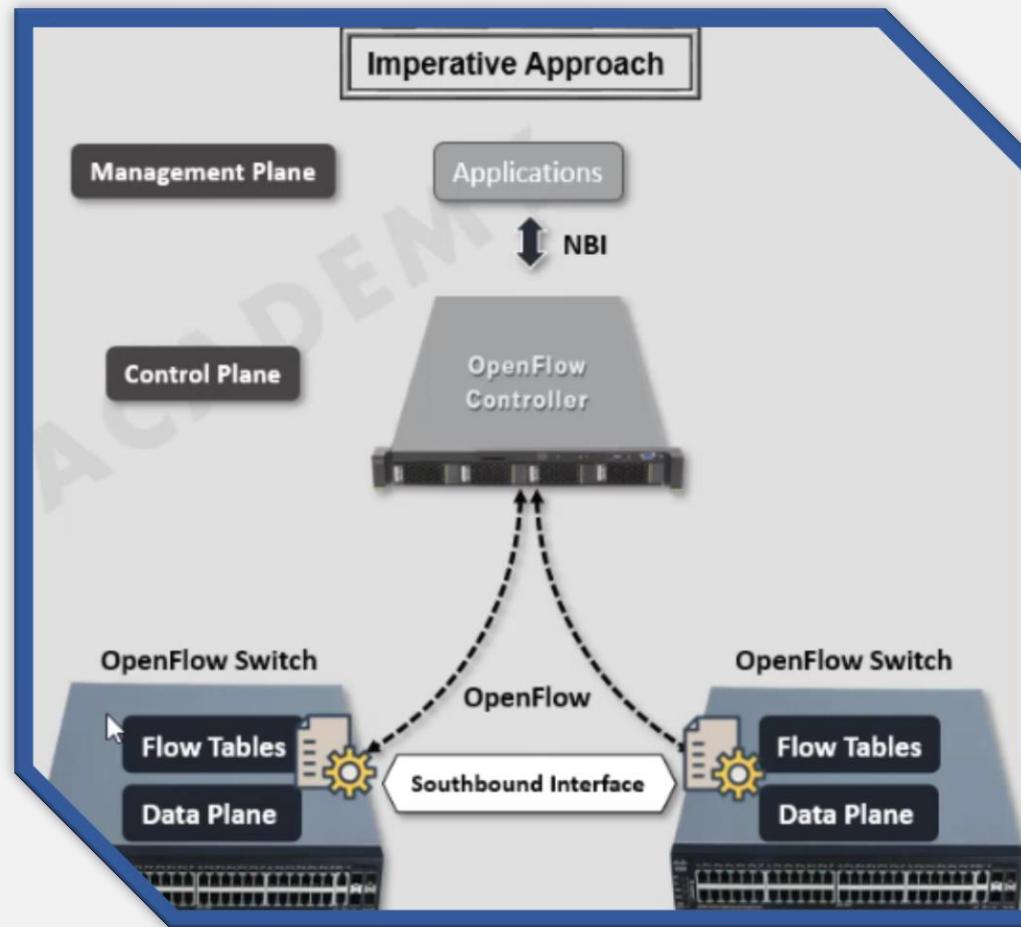
هو بروتوكول **OpenFlow** من **Open Flow**، يستخدم لضبط ال Switches في بيئات ال SDN، تم إطلاق ال SDN قبل Controller، Network IP-based SBI يقوم ال Open Flow بعمله بتحديد ONF – Open networking Foundation، تحديد قدرات ال Switch بناء على ال ASICs، TCAM الخاصين بها، تجميع (Centralize) أغلب ال Devices ضمن ال Control Plane، تحديد جهة التحكم بال Controller باتصالات ال NBI.

يمكن لهذا البروتوكول:

- يساعد أجهزة الشبكة في بناء الجداول Flow Tables بمعلومات المساعدة في طريقة تمرير البيانات التي تستقبلها.
- عند استقبال جهاز السويفت بيانات يقوم بمطابقتها بالبيانات الموجودة في Flow Table.
- في حال فشل في إجراء أي مطابقة يقوم بأعلام جهاز Controller.
- يقوم Controller بإرسال معلومات من أجل تحديث Flow Table.

في الـ OpenFlow Model, يتم استخدام أي نوع API في الـ NBI للاتصال مع الـ Controller ولكن تستخدم فقط بروتوكول OpenFlow ضمن الـ SBIs، يمكن استخدام تطبيقات منفصلة للتحكم في أجهزة الشبكة باستخدام SBI، يجب أن تدعم الـ Switches بروتوكول OpenFlow.

**ملاحظة:** هذا النوع في شبكات SDN يجب وجود Controller ليستطيع التحكم بكل أجهزة الشبكة بدونه لا تستطيع أجهزة الشبكة تمرير البيانات تسمى هذه الطريقة Imperative Approach.



## OpFlix Protocol

هو بروتوكول يعتمد في إنشاء مجموعة من الإعدادات معدة مسبقاً في جهاز Controller وأرسالها لأجهزة الشبكة باستخدام لغات XML or Jason. يتم ارسال هذه الإعدادات لأجهزة الشبكة من أجل التحكم الموجود فيها Controller Plane التي لم تعد موجودة في Control Plane.

ملاحظة: هذا النوع في شبكات SDN حال توقف Controller عن العمل لا يؤثر على أجهزة الشبكة لأن الأجهزة تحتوي Control Plane . Declarative Approach يوصي and Data Plane

## 8- How Automation Impacts Network Management

لإدارة الشبكات تقليديا يتم استخدام اتصال SSH لكل جهاز على حدي لضبطه واستخدام طريقة copy/paste كثيراً، ثم ظهرت أدوات SolarWinds, Cisco NMS – Network Management System وال NetFlow في برامج مثل: .... Prime, Cisco Works

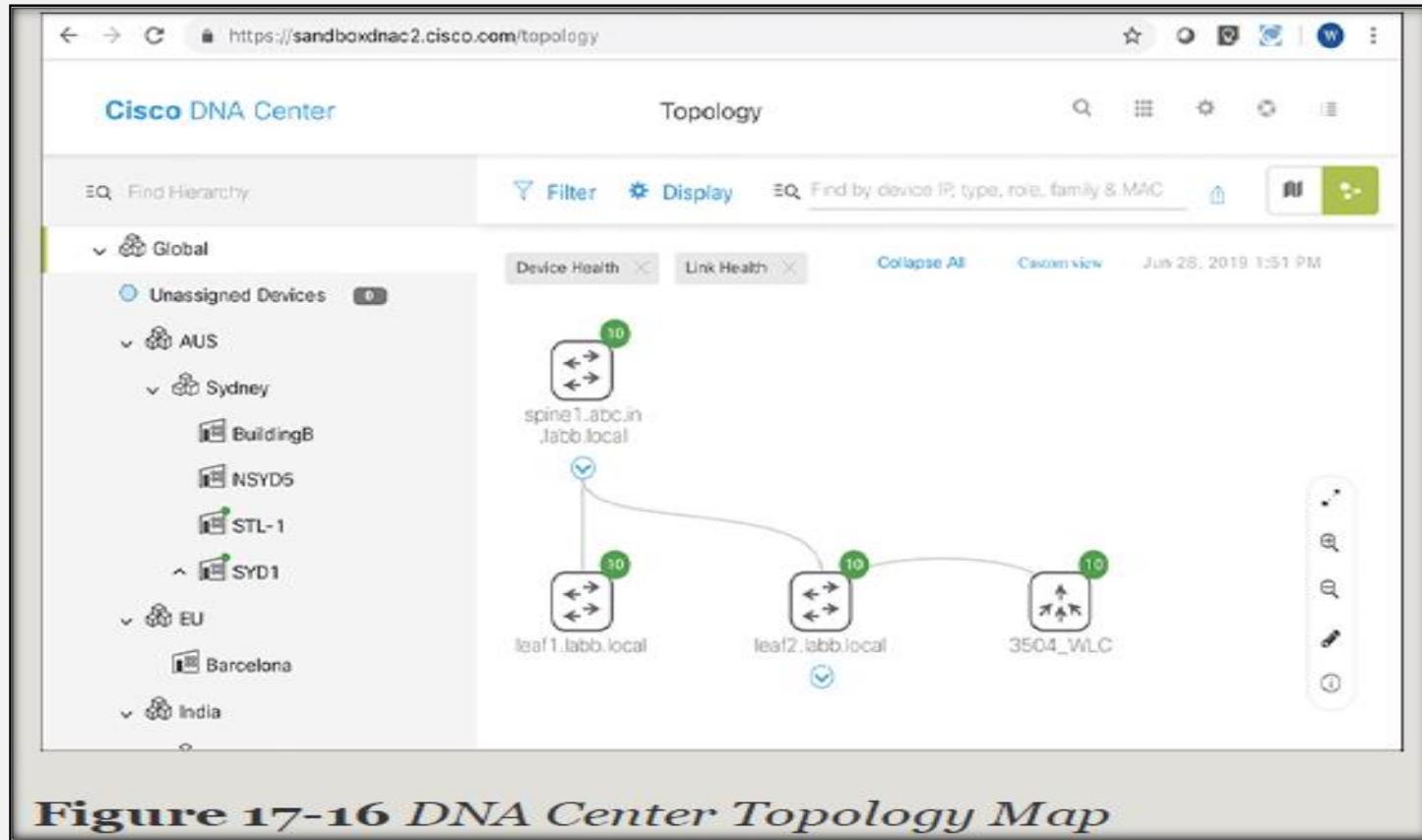
مساوئ استخدام الطريقة التقليدية في ضبط الشبكة:

Time Consuming ←  
Inefficient ←  
Typos ←  
احتمال حدوث ←  
Configuration Drift ←  
admins لضبط الشبكة يستخدمه جميع ال ←  
Inefficient Troubleshooting ←

يستخدم ال Network Automation للمزايا التالية:

- |  |  |
|--|--|
| Device Configuration (1)   | Initial Device Provisioning (2)  |
| الجديدة بالبحث عن Controller أول وصلها بالشبكة، ووصولها له عن طريق ال DNS, DHCP Option 43, أو ال .DNS. | للتتأكد من أن جميع التجهيزات تعمل على نفس النسخة. Software Version Control (3) |
| Collect Statistics from Devices (4)  |  |

- Compliance Verification (5)  
Reports (6)
- تقليل التدخل من العنصر البشري مما يقلل الأخطاء مثل ال Typos. (7)
- Scalable (8): يمنح استقرار أكبر للشبكة و لنموها من ضبط كل جهاز على حدٍ.
- يمنح القدرة على مراجعة التعليمات التي تم ضبطها مسبقاً والعودة إليها في حال الحاجة. (9)
- Reduce Operating Expense – OPEX (10): زيادة كفاءة ال Troubleshooting مما يتاح تخفيف تكاليف و نفقات العمليات.
- Increase Capital Expenses - CAPEX (11). Automation
- Topology Map (12): يقوم ال Controller (مثل DNA Center) بتوفير رسم مخطط شامل للشبكة مع جميع التجهيزات الشبكية المتصلة.



**Figure 17-16 DNA Center Topology Map**

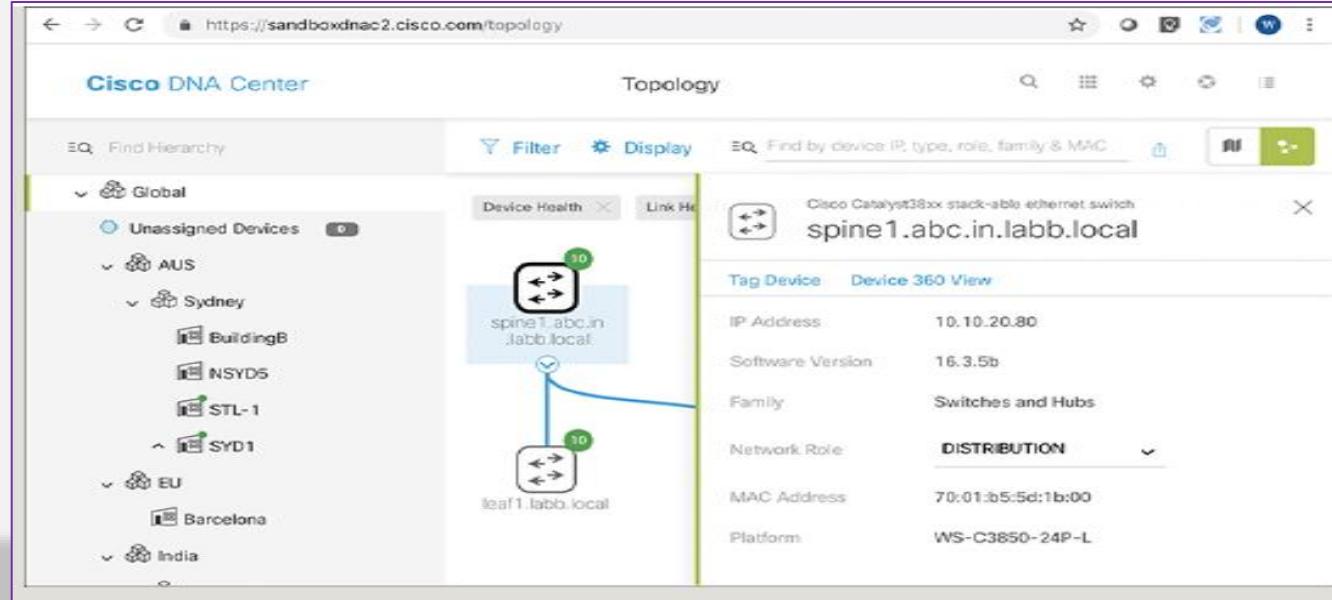
مع إمكانية الحصول على معلومات إضافية عن كل عقدة عند الضغط عليها عوضاً عن استخدام الطريقة التقليدية في ضبط التجهيزات الشبكية باستخدام إرسال تعليمات، سيتم في ال SDN API Call إرسال API Call تحتوي على الطلب المناسب (Get, Update, Delete ...). (Trunking mode, telnet password...). مثلاً الـ Variable Sets

## **Example 16-2** Python Dictionary with Variables Set to Needed Values

Click here to view code image

file:///tmp/calibre\_4.3.0\_tmp\_EFmwkj/KUhePy\_pdf\_out/OEB  
PS/Images/ch16\_images.xhtml#exm16\_02

```
>>> interface1
{'trunk-config': 'dynamic auto', 'trunk-status': 's1
>>>
```



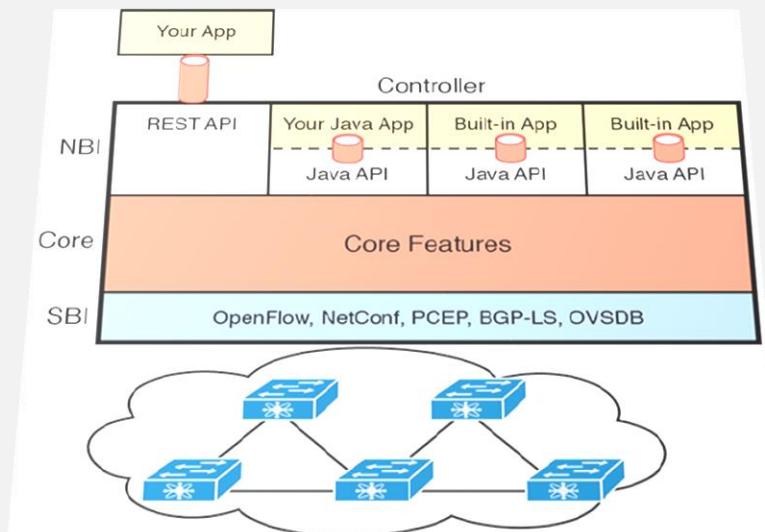
**Figure 17-17 Hover and Click Details About One Cisco 9300 Switch from DNA Center**

## 9- SDN Solution: {Edgeworth, 2019 #2} {CISCO., 2022 #6}

- ❖ Open Daylight Controller.
- ❖ Cisco Application Centric Infrastructure – ACI.
- ❖ Cisco APIC Enterprise Module – APIC-EM.
- ❖ SD Access.
- ❖ SD WAN.

## *(ODL) Open Daylight Controller*

يعتبر ال Open Daylight Controller مفتوح المصدر Open Source ومن أنجح ال Controllers التي تم تجربتها مع ال OpenFlow ويلعب دور رئيسي في الشبكة ويستخدم في شبكات Data Center . تم بناء ال ODL بالاعتماد على Linux ، ويمكنه التعامل مع عدة APIs ضمن ال SBI مثل ال (LS, OVSDDB .).



## *The Cisco Open SDN Controller (OSC)*

قامت Cisco بتطوير ال OSC كنسخة تجارية خاصة بها مبنية على ال ODL يعتمد ال OSC على طريقة ال Intent-Based-Networking (IBN)، حيث يأخذ البنية الخاصة بنا (مثلاً ربط Users خاصين بال HR مع سيرفر الدوام)، ويقوم هو بتطبيق ال Policies, Routing, ACLs, VLAN .. اللازمة لوحده. لم يتم الاستثمار ب OSC من قبل Cisco كثيراً في السوق، حيث توقف إنتاجه ولكن تم الاعتماد عليه في بناء عدة Controllers آخرين خاصين ب Cisco مثل ال ACI .

## **10- Cisco Application Centric Infrastructure – ACI:** {CISCO., 2022 #6}

بعد الأبحاث والتطوير الذي قامته به Cisco على تقنيات ال SDN, قامت بتحديد ثلاثة تقنيات لاستخدام ال SDN حسب بيئه العمل والشبكة كالتالي:

- .Data Centers :ACI (1)
- .Enterprises Campus :SDA (2)
- .Enterprise WAN :SD-WAN (3)

### **:ACI Physical Design (Spine and Leaf)**

يستخدم في ال Data Centers الحديثة تصميم ال Spine/Leaf (يطلق عليه أيضا Clos Network (بناء على اسم أحد مخترعيه)، حيث يتم وصل كل التجهيزات مثل (Endpoints, Servers, VMs..) مع كل Leaf، ويتم وصل كل Leaf مع كل Spine، وكل Spine مع كل ال Leaves، ولا يمكن وصل أي Leaf Switch مع Spine Switch أخرى، ولا وصل Spine Switch مع Leaf Switch أخرى يتم وصل ال叶 switches (بمختلف أنواعها – ...,Containers, VMs, Routers Physical Servers –) بشكل مباشر مع ال Endpoints، ولا يتم وصلهم مع ال Spine Switches نهائيا. ممكن أن تتصل Leaf Switch مع أكثر من Endpoint كنوع من ال Redundancy في حال الحاجة لذلك.

### **:ACI Operating Model with Intent-Based Networking**

يطلق على ال SDN Controller المستخدم في ال Spine/Leaf Design اسم <APIC – Application Policy Infrastructure>، الذي يكون مسؤوال عن إنشاء وتطبيق ال Overlay Controller

نرى في ال Data Centers ثلاثة أنواع من ال Servers يطلق عليهم <EPG – Endpoint Group> :

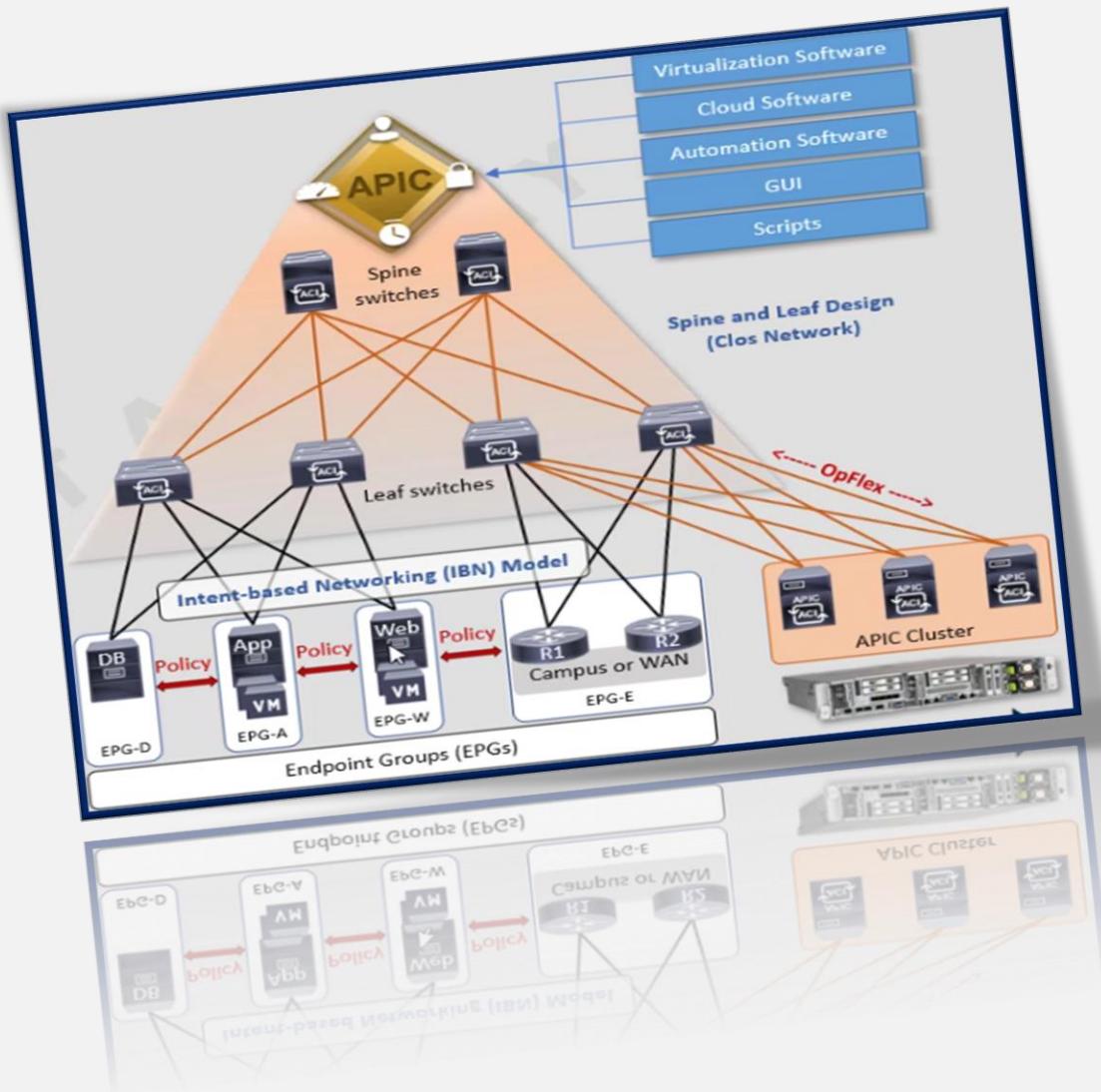
- Web Servers (1)
- Apps Servers (2)
- DB Servers (3)

ويربط بينهم Access Policy تحدد سماحيات اتصالهم مع بعض وطريقة اتصالهم بعد أن يتم وصل جميع ال Endpoints وتحديد ال Data Center، يتم استعمال ال APIC Controller حيث يكون مسؤوال عن إنشاء ال Application Policies الخاصة بال Infrastructure Overlay، يتم استخدام تقنية ال Vxlan للتواصل بين ال

يطلق على ال SBI في ال ACI اسم OFFLEX (اسم البروتوكول المستخدم)، يطلق على ال NBI في ال ACI اسم GUI، أو API.

في بيئه ال ACI يتم ضبط ال APIC (عن طريق GUI...) بجميع الإعدادات الخاصة بال Switches بحيث لا يتم هناك أي حاجة لضبط ال Switches بشكل فردي عن طريق ال CLI، يستخدم ال APIC لإدارة ال Data Centers التي تحتوي على Nexus Switches، في ال DNA Center ينصح باستخدام IS-IS ك Routing Protocol

يطلق مصطلح {Routed Access Layer Design} على البنية التي لا تستخدم بها تقنيات Layer2 (مثل ال Underlay) حيث كل Link يكون Routed.



## 11- Cisco APIC Enterprise Model: {Edgeworth, 2019 #2} {WENDELL, 2020 #3}

APIC-Enterprise Environment مصمم لإدارة الـ APIC-EM (Branch, WAN, Campus) مثل النسخة المحدثة من الـ APIC-EM هي الـ DNA Center.

### **:APIC-EM Basics**

من أهم خواص الـ APIC-EM انه يقوم بدعم خواص الـ SDN Controller القديمة بدون حاجة لاستبدالها بتجهيزات جديدة.

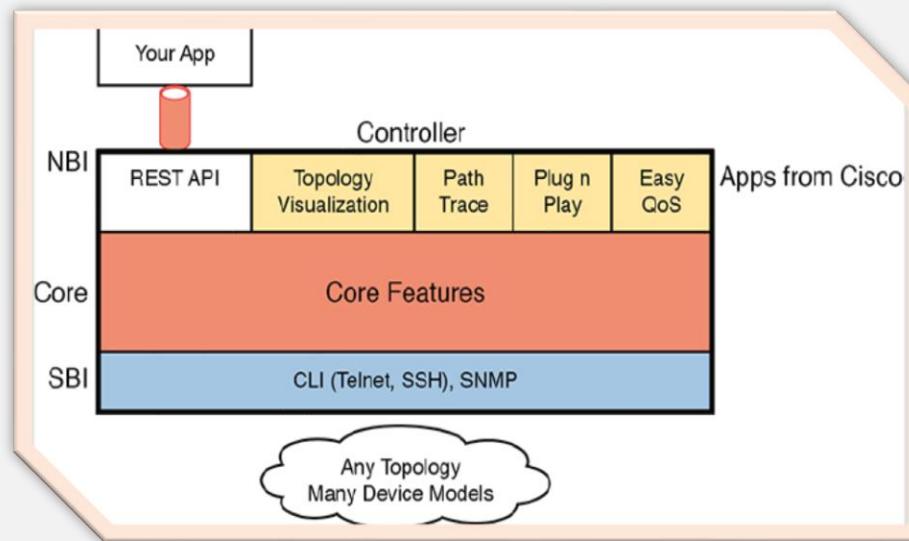
بالرغم من تعامل الـ APIC-EM مع التجهيزات نفسها الموجودة في الشركات (بدون استبدالها بأحدث تدعم الـ SDN)، يبقى لديه عدة خواص هامة يوفرها عن طريق الـ NBIs والـ SBIs مثل:

(1) Topology Map: يوفر الكشف عن كامل مخطط الشبكة وعرضه

(2) Path Trace: يقوم المستخدم بتحديد Application, Source/Destination Addresses ليقوم الـ Forwarding Decisions بعرض مخطط مسیر هذا الاتصال مع تفاصيل عن الـ Forwarding Decisions التي تم اتخاذها في كل Step.

(3) Plug and Play: يقوم بتوفير Day 0 Installation Support، مما يوفر دعم أي تجهيز شبکیة جديدة لها عنوان IP للشبکة يتم وصله مباشرة للشبکة بدون الحاجة لأي ضبط لها.

(4) Easy QOS: ممكن ضبط قواعد QOS معقدة لكل Node في الشبکة عن طريق خطوات بسيطة في الـ Application.



## APIC-EM Replacement

تم توقيف إنتاج ال APIC-EM من Cisco في ال 2019، واستبداله بال DNA Center، ولكن وجب وجود أفكار أساسية لدينا عن ال APIC-EM بسبب استخدام أغلب خواصه الرئيسية في بناء ال DNA Center (DNAC).

### 12- SD – WAN:

في الطريقة التقليدية لتطبيق ال WAN كان من الصعب الانتقال إلى خدمة WAN جديدة وكان التركيز على تحقيق ال Connectivity وليس على الاستخدام الأفضل للبرام吉. ولم يكن هناك Stander معين لتطبيق ال Configurations، عند تطبيق ال SD-WAN نحصل على:

- 1) اتصال بين التجهيزات يكون Automated ,Standardized
- 2) يعمل مع جميع أنواع ال WAN (مثل ...)(MPLS, BGP, ...).

- (3) بسيط ويتاح إمكانية التحديث أو الانتقال ل WAN Service جديدة.
- (4) يتيح التعامل مع أحدث تقنيات ال Cloud.
- (5) تكلفة أقل.

### 13- SD-WAN Architecture: {CISCO., 2019 #9}

تكون توسعتها Horizontal Scaling لدينا 4 مكونات رئيسية ابتداء من الأسفل:

التي تحتوي على ال Edge Routers، حيث ممكن أن تكون ال Edge Routers من نوع Virtual/Physical (مثل Cisco ISR)، وتكون ال Vedge Routers مسؤولة عن ال Forwarding، وتقوم بإنشاء طبقة من اتصال ال IPsec المشفر بين بعضها لتبادل المعلومات حيث كل Site ممكن أن يحتوي على 2 Vedge Routers كنوع من ال Redundancy. التي تحتوي على Controller يتحكم بال Edge Routers ويطلق على ال Vsmart الذي يتم تنسييه على Vedge (مثلاً وضع ال Virtual Machine)، يعتبر ال Vsmart العقل المدبر في ال SD-WAN ويقوم بإرسال ال OMP – Overlay Policies ومعلومات ال Forwarding إلى ال Vedge Routers عن طريق TLS Tunnels باستخدام بروتوكول Management Protocol وممكن لكل Vedge Router أن يتصل ب 2 Vsmart Routers كنوع من ال Redundancy.

التي تتيح إدارة ووصول ل Vsmart عبر استخدام ال Vmanage (نوع من أنواع تطبيقات ال NMS)، يوفر ال Vmanage NMS واجهة GUI لإدارة ال Management Plane، وتم تشغيله أيضاً على Virtual Machine، ويحتوي على خاصية Alerting للتنبيه في حال أي إشعار يتم عادة تجميع عدة Vmanage Machines ضمن Cluster واحد ل Redundancy.

التي يتم بها استخدام ال VBond، الذي ممكن تنصيبه أيضاً على VM أو على Router في التصاميم الصغيرة ومن الممكن ربط عدة Vbond Orchestrators باستخدام Round Robin DNS حيث يقوم ال Vbond بمصادقة جميع ال Vedge، Vmanage NMS وال Vsmart في الشبكة ويتيح ل باكتشاف بعضها واكتشاف ال DMZ ويوضع في منطقة ال Public IP ب Vbond يضبط ال IP.

**:Zero Touch Provisioning – ZTP**

VBond هي منصة Cisco Cloud Based Service (CBSS) التي تستخدمها الـ Vedge Routers عند إقلاعها لتوجيهها إلى الـ Cloud الذي ينظم انضمامها للشبكة ممكناً امتلاكه وتنصيب الـ Premises أو في استضافة على الـ VManage, VBond, VSmart (وهي مستخدمة أكثر).

### - Building the Data Plane in SD-WAN: {CISCO., 2019 #9}

يقوم الـ VSmart Controller بتوجيه الـ Vedge Routers لبناء Full Mesh من قنوات IPsec VPN بين بعضهم (هذا يكون خيار الـ Default). ويقوم بتزويدهم بالـ IPs المتاحة في الـ Sites الخاصة بهم، يتم إرسال Packets في الـ VPN Tunnel بـ SLA (Service Level Agreement) الخاصية لكل Link.

### - Traffic Forwarding Option: {Edgeworth, 2019 #2}

في حال وجود عدة Tunnels (مثل MPLS, Internet...) يمكن القيام بـ Load Balancing بينها حيث يمكن استخدام خوارزمية Application Pinning Active / Standby (الأفضلية لأحد أنواع الـ Tunnels) أو Weighted Active / Active أو Active/Active (كإرسال مثلاً الفيديوهات والصوت على الـ Tunnel... والإيميلات والملفات على الـ Tunnel...) أو Application Aware Routing.

Developers يستخدمون SDK – Software Development Kit لـ DNA Center ليمنحوا زيادة في الـ Flexibility، حيث يستخدمون الـ Cisco Vendors لإتاحة التعامل مع آخرين.

## 14. SD-ACCESS Fabric: {WENDELL, 2020 #3}

هي الـ Physical Network التي توفر الـ Connectivity لـ Overlay. وتعتمد على إنشاء قدر الإمكان من Redundant L3 Connectivity .Routed Access Layer Design (تحتوي على الـ L3 Switches) ليتم تشكيل ما يسمى بـ

ممكن أن يتم بها استخدام التجهيزات القديمة الخاصة بالشركة، مع إضافة إعدادات جديدة لبناء ال Underlay Network **أو:** يتم شراء تجهيزات Switches جديدة لبناء ال Underlay Network، ومن ثم القيام ب Migrate للشبكة القديمة إلى الجديدة بالتسلسل مع الوقت، وفي هذه الحالة يتم ضبط ال DNA Center لضبط التجهيزات الجديدة فقط، بحيث لا يؤثر على عمل الشبكة المستخدمة حالياً (القديمة) إلى أن يتم نقلها ودمجها بشكل كامل مع شبكة ال Underlay الجديدة، ويجب أن تدعم التجهيزات القديمة خاصية ال SDA (حسب نسخة ال IOS) وإن يكون بالإمكان دمجها مع الشبكة الجديدة.

يطلق على ال Switch المواجهة للتجهيزات المستخدم في بيئة SD-Access مصطلح **Fabric Edge Node** (مثل ال Catalyst 3850,3650 أو سلسلة ال 9k).

يطلق على ال Switch المتصلة بتجهيزات غير منضمة ل SDA Network (مثل ال Routers المتصلة بال WAN, او شبكة ACI) بال **Fabric Border Node**.

يطلق على ال Switch المسئولة عن بعض عمليات ال Underlay Control Plane الخاصة بال Fabric Control Node وهي تحتاج ل CPU, Memory إضافية.

يتم حفظ جميع المعلومات الخاصة بكل جهاز متصل ب Edge Node داخل Map وهذه ال Map تحفظ داخل Fabric Edge Node، حيث سيحتوي على ما يشابه ب Small Routing Table .Fabric Control Plane Node

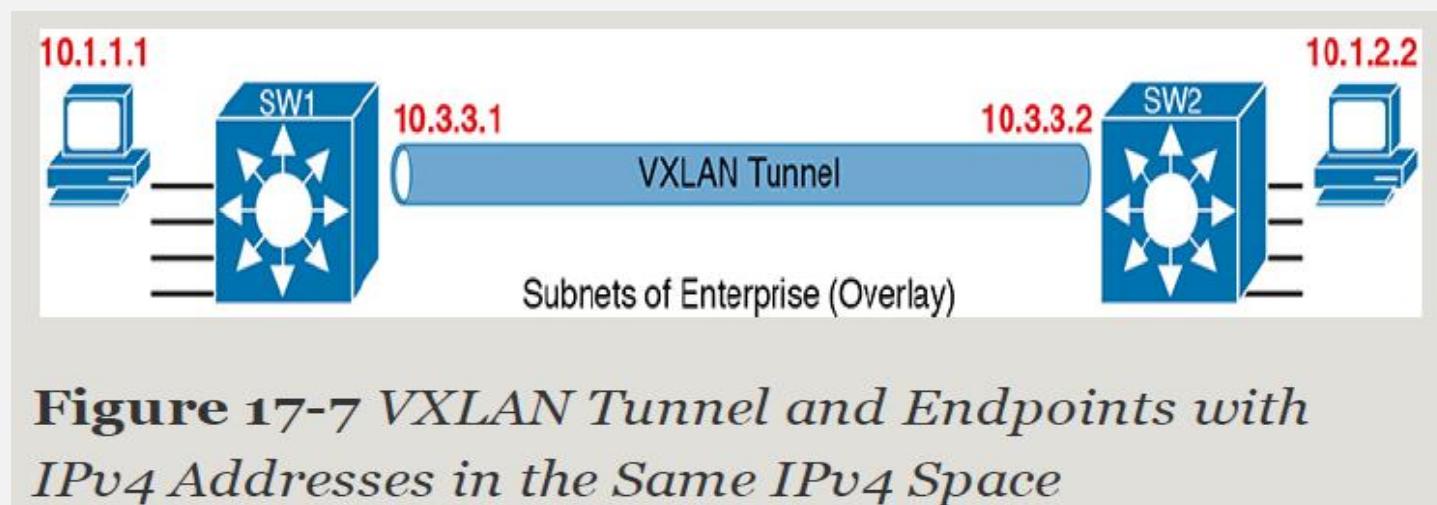
### **يتميز ال Routed Access Layer Design بالخصائص التالية:**

- 1) كل ال Switches تكون L3
- 2) تستخدم ال switches بروتوكول ال IS-IS ك Routing Protocol
- 3) جميع ال Links بين ال Switches تكون Routed L3 Links (Single Links, EtherChannel Links)
- 4) ليس هناك حاجة ل STP/RSTP بما أن جميع ال Links من نوع L3
- 5) يتم استخدام ال Default Gateway (ال Access Layer Switch ) ك SDA Edge Node عوضاً عن استخدام Switch في ال Distribution Layer كما في التصاميم التقليدية
- 6) ليس هناك أي داعي لاستخدام HSRP, FHSRP أو أي بروتوكول مشابه في تصميم ال SDA Routed Access Layer

**Overlay (OV)**: هي الشبكة الافتراضية المبنية على أساس ال Underlay و التي تقوم ب tunneling (باستخدام تقنية ال Vxlan) بين الأجهزة المراد إنشاء Connection بينها، حيث عند استلام أي رسالة من End point، تقوم ال SDA Node بتغليف هذه الرسالة Fabric (encapsulate it) بخلاف جديد خاص بال Vxlan Tunnel وإرسالها إلى ال Fabric، عند وصولوها ل Fabric ستقوم باقي ال Nodes لتحويلها وإرسالها حسب معلومات طبقة ال Vxlan داخلها عبر ال Vxlan Tunnels ، وعند وصول الرسالة لهدفها، يتم فتح غلاف ال Tunnel للوصول لمعلومات الرسالة الأساسية.

يتم الاعتماد على ال ASIC داخل ال Switches للقيام بمهام طبقة ال المعقدة، مثل مهام ال Encapsulation، De-encapsulation .ASIC الخاصة بال Vxlan حيث من أهم النقاط الواجب توفرها في أي Switch لانضمامها لشبكة SDA هو احتوايتها على

يتم استخدام Subnet بشبكة ال Overlay التي يعمل بها ال Vxlan مختلف عن ال Subnet الخاص بال Underlay (مختلف عن ال المستخدم لعناوين ال الفعلية) ويكون ال Subnet المستخدم في شبكة ال Overlay غالبا من نفس ال Subnet الخاص بال End-points.

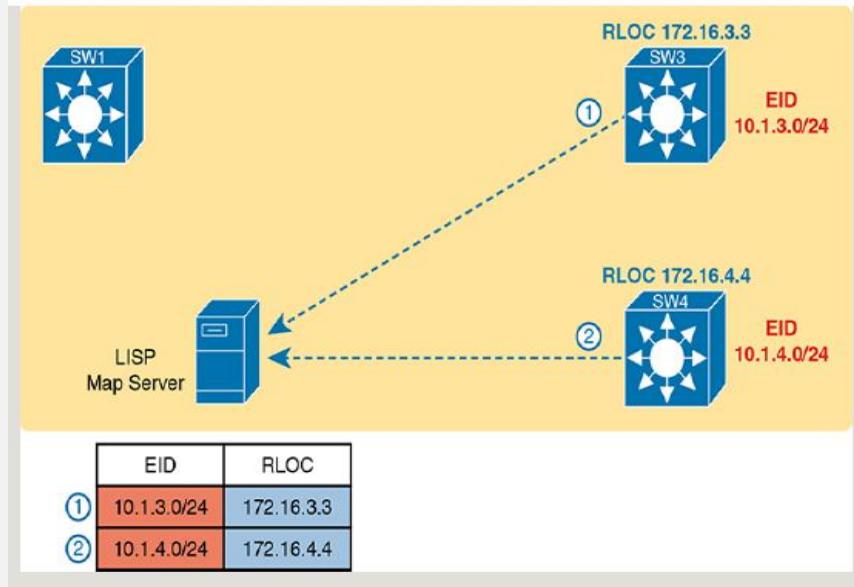


**Figure 17-7 VXLAN Tunnel and Endpoints with IPv4 Addresses in the Same IPv4 Space**

: هو مصطلح يطلق على جميع الأجهزة في ال SDN Design وهو يعبر عن كل من ال Overlay + Underlay ، تضم منطقة ال Fabric . SD-Access Center Controller يتم استخدام ال Underlay، Overlay، Fabric في ال DNA SDA Southbound شبكات ال

تحتوي ال SDA على 3 طبقات: {Lessons., 2016 #8}

(1) **Control Plan**: (تعمل في ال Overlay - مثل ال DNS) ، ولا تقوم بنفس وظائف ال Control Plane الاعتيادية (بناء-  
LISP – Location Identity address table, Routing Table, Arp Table  
Separation التي تقوم بتعريف كل جهاز متصل ب Switch لدينا ب EID – Endpoint Identifier حيث إن كان لدينا موبايل (أو أي End Point) متصل بالشبكة، سيحظى هذا الموبايل ب EID خاص به يتم حفظ جميع ال EID ضمن Lisp Map Server Database التي تضم أيضا ال RLOC وهو عبارة عن طريقة الوصول لهذا الموبايل من الشبكة (أي جهاز شبكي لهذا الموبايل متصل به، يعني أي fabric Edge Node متصل بها الموبايل) وكلا ال ROLC وال EID موجودان ضمن **Master Map**.



**Figure 17-8 Edge Nodes Register IPv4 Prefixes (Endpoint IDs) with LISP Map Server**

في حال وصول أي رسالة موجّهة لعنوان محدد، تقوم الـ Map Server بسؤال الـ Edge Node ليبحث عن هذا العنوان ضمن الـ Map Database الخاصة به ثم يقوم الـ Map Server بالتحقق من الـ ROLC الخاص بهذا العنوان عبر تجربة التواصلي معه، ثم يقوم بإرسال الـ ROLC الخاص بالـ EID المستلم إلى الـ Edge Node التي أرسلت الاستعلام.

إن هذا التصميم له مزايا مهمة:

- ✓ **Smaller Routing Table**: وكل ما ازداد حجم الشبكة سنقوم بزيادة مواصفات الـ Fabric Control Plane Node.
- ✓ **Device Mobility**: في حال انتقال User أو Endpoint من Switch لـ Switch بموقع آخر فإن جميع خصائصه ستبقى معه.
- ✓ **Data Plane(2 Routed Access Layer**: تستخدم تقنية Vxlan التي تقوم بـ Encapsulate لـ L2 frame داخلاً لـ L3 UPD Packet ليتم تشكيل ما يسمى بـ

SGT – Scalable Group Tags على DNA Center, حيث يعتمد ال CTS – Cisco Trust Set على المصادقة عن بعضها بواسطة هذا ال Tag وبناء على هذا ال Tag الخاص بكل مجموعة تقوم بتطبيق ال Policies التي تسمح ل Groups معينة بالاتصال مع بعضها البعض.

حيث باستخدام تقنية ال Scalable Groups Policies تم التخلص من المشاكل التي كانت تواجهنا أثناء التعامل مع ال ACL مثل:

- صعوبة التخلص من ACE (Access List Entry) من أحد ال ACLs, لصعوبة تحديد الأجهزة وبقى ال ACE المتاثرة بها.
- إضافة ACE جديدة ل ACL تزداد صعوبة مع الوقت بسبب زيادة طول محتوى ال ACL.
- تحديد تسلسل معالجة إرسال الرسائل من قبل ال ACE يصبح أصعب مع الوقت بسبب زياتها.

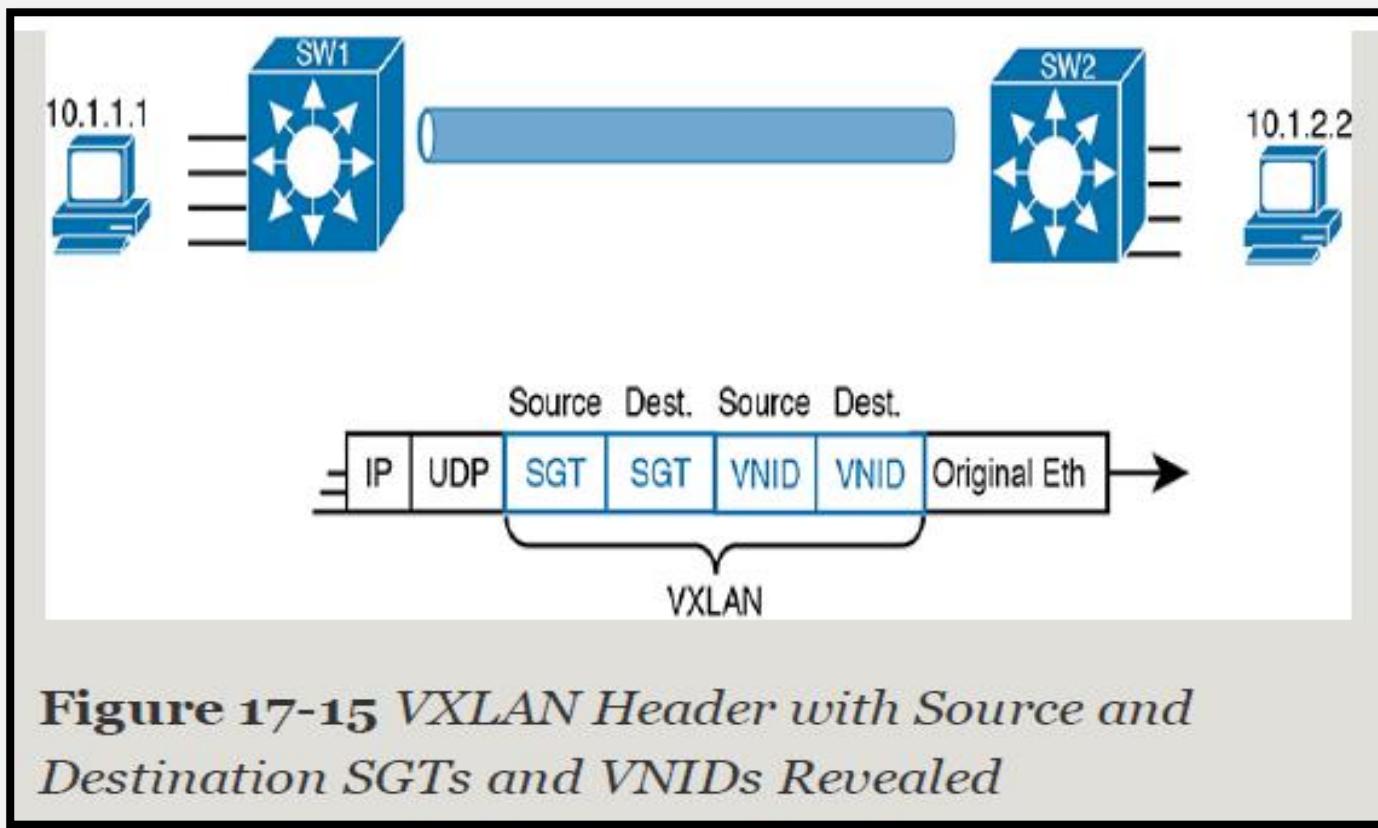
❖ هناك عنصرين هامين في تكوين ال SD-Access :

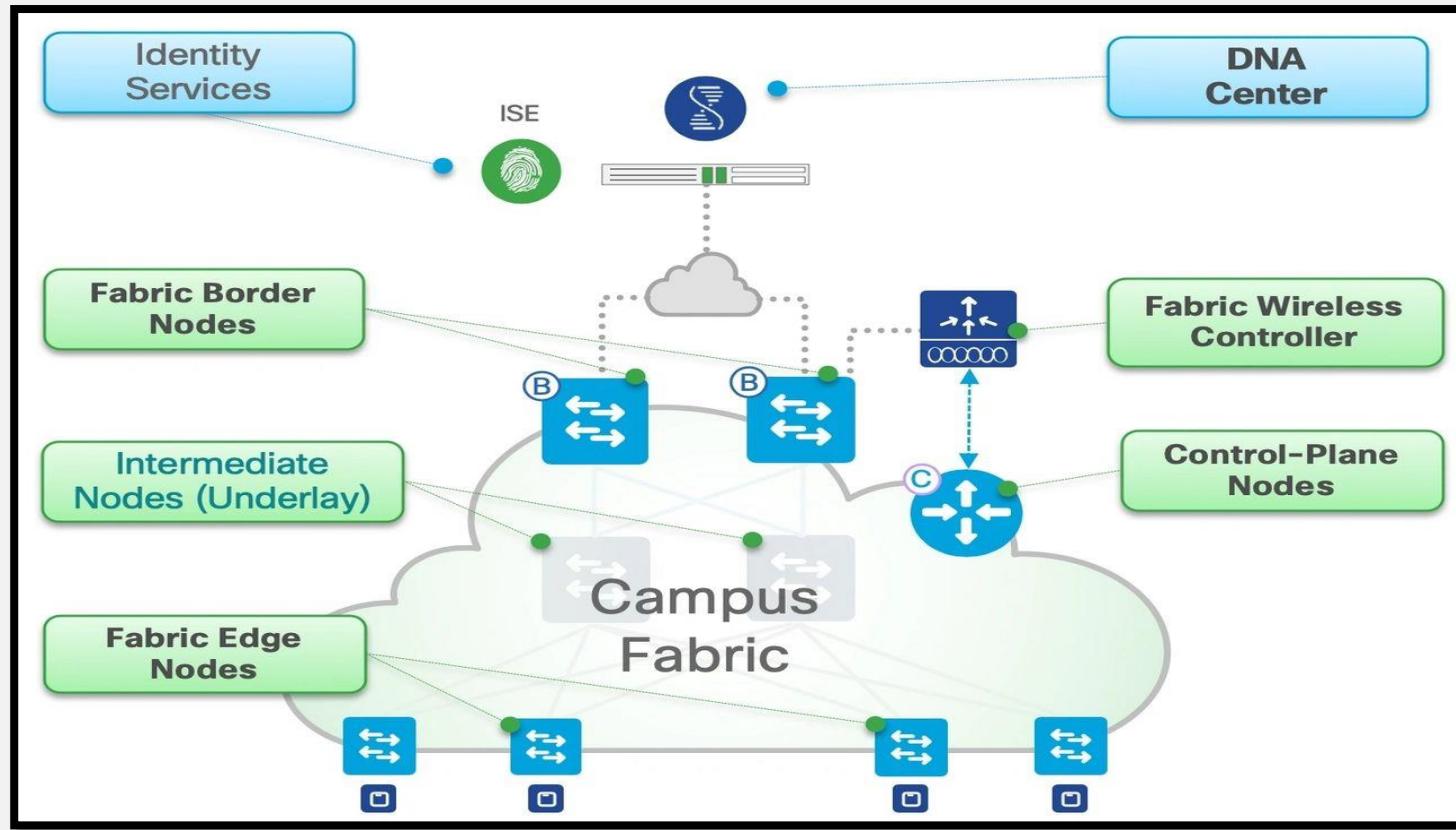
Identity Service Engine :ISE ➤ Security group Tag الذي يستخدم للمصادقة (مثل ال AAA), ولتمييز ال Tag الذي ينتمي له User المرسل للرسالة.

Security Policy ➤ DNA Center: التي تسمح و تمنع مرور الاتصالات بين ال Groups ويتم ضبطها في ال DNA Center.

بعد استيعاب مفهوم ال SGTs, نستنتج أنه أي رسالة جديدة تصل ل Edge Node يتم أولاً التعرّف على هوية مرسليها لمطابقتها مع ال SGT الموافق لها عن طريق ال ISE

ثانياً يتم معرفة ال Policies الخاصة بهذه ال Group, وعلى أساسها يتم تغليفها ب Vxlan وإرسالها عبر ال Vxlan Tunnel المناسبة، يقوم ال DNA Center باستخلاص ال Policy من إعدادات الأجهزة التي يديرها، تستخدم للمصادقة مع AAA Server وتضبط على ال Postman . غالباً لإدارة DNA Center تقوم باستخدام أداة اسمها Center





## 15- Summary Of SDN Examples : {Odom, 2019 #1}

مقارنة بين ال 3 مخططات الخاصة بشبكات ال SDN (OpenFlow, ACI, APIC-EM)

	OpenFlow	ACI	APIC-EM
Organization that is the primary definer/owner	ONF	Cisco	Cisco
Degree to which the architecture centralizes the control plane	Mostly	Partially	None

SBI used Controllers mentioned in this chapter	OpenFile OpenDaylight	OpFlix APIC	<b>CLI, SNMP</b> <b>APIC-EM</b>
---	--------------------------	----------------	------------------------------------

## 16- Configuration Management:

{CNNA., #7}

بالانتقال لإدارة الشبكة يوجد عدة طرق تختلف عادتاً

باختلاف عدد الأجهزة الموجودة في الشبكة **Switches and Routers**

في الشبكات الصغيرة والطريقة التقليدية يمكن ضبط إعدادات الشبكة باستخدام **CLI** بشكل منفرد لكل جهاز على حده باستخدام إحدى المنافذ (**Console – Telnet – SSH**) حيث تكتب الأوامر لكل جهاز ويتم حفظ الإعدادات في النهاية، ثم ظهرت أدوات ال – NMS – SolarWinds, Cisco Prime, NetFlow وال SNMP في برامج مثل: Network Management System .... Cisco Works

مساوٍ استخدام الطريقة التقليدية في ضبط الشبكة:

Time Consuming

Inefficient

احتمال حدوث Typos

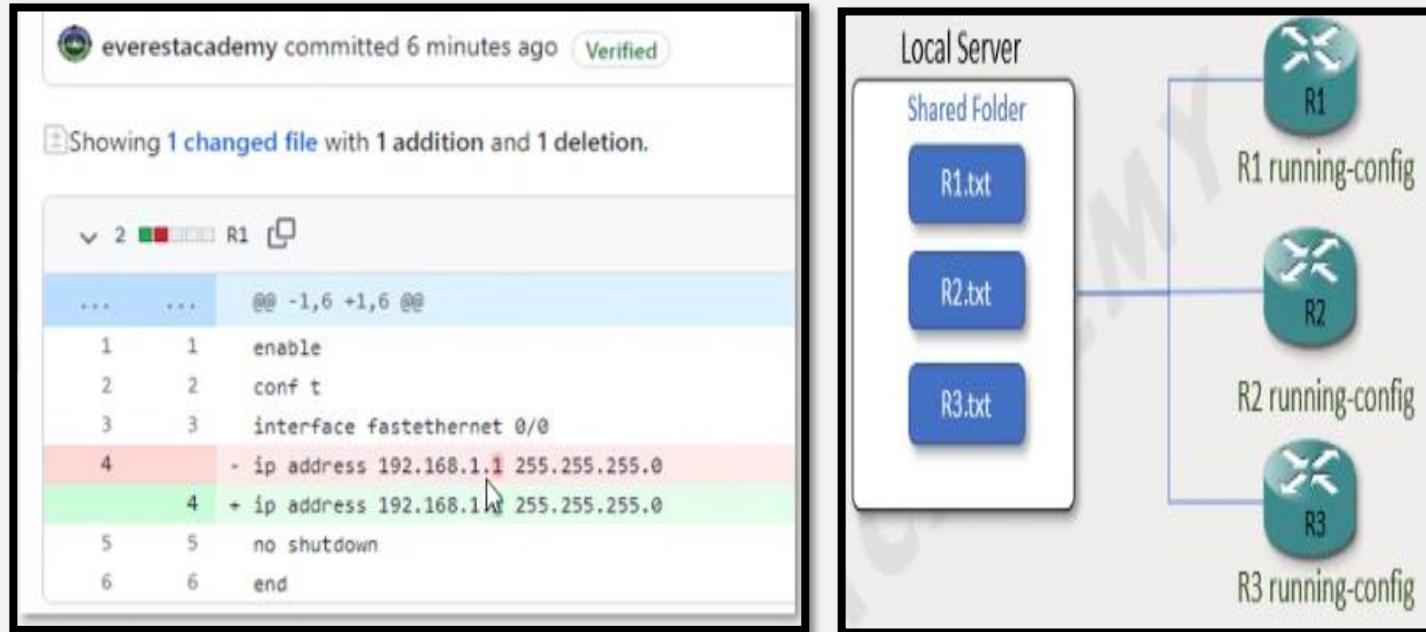
admins: عدم وجود Stander Configuration Drift

Inefficient Troubleshooting

باستخدام الطريقة التقليدية واضافة تعديلات عليها يؤدي الى ما يسمى **Configuration Drift** خروج ضبط وإعدادات أحد التجهيزات عن اتساق (**Consistency**) باقي إعدادات أجهزة الشبكة بدون إعلامه لباقي الفريق لانشغاله وهذه الطريقة لا تسجل أي معلومات عن التغييرات التي تم إعطائها للإعدادات ومن قام بتعديلها والحل يتم تخزين الإعدادات في جهاز مركزي وتتبع التعديلات مثل: Git: هي **Access Control System** تستخدم لتتبع تغييرات الإعدادات الخاصة في الشبكة وهو مناسب للشبكات متعددة الأجهزة.

**GitHub** هي **Hosting Service** تحفظ جميع ملفاتها داخل **Repository**, حيث ممكن أن تكون الـ **Public/Private Repositories** (جمهور/خاص).  
أن تنسخ بين الـ **Users**, وتملك **Tools** لمنع حدوث **Conflict** عند الكتابة عليها من قبل عدة مستخدمين في الشبكات الكبيرة يتم استخدام أدوات خاصة في اتمتة الإعدادات مثل (**Ansible – CHEF – Puppet**).  
.

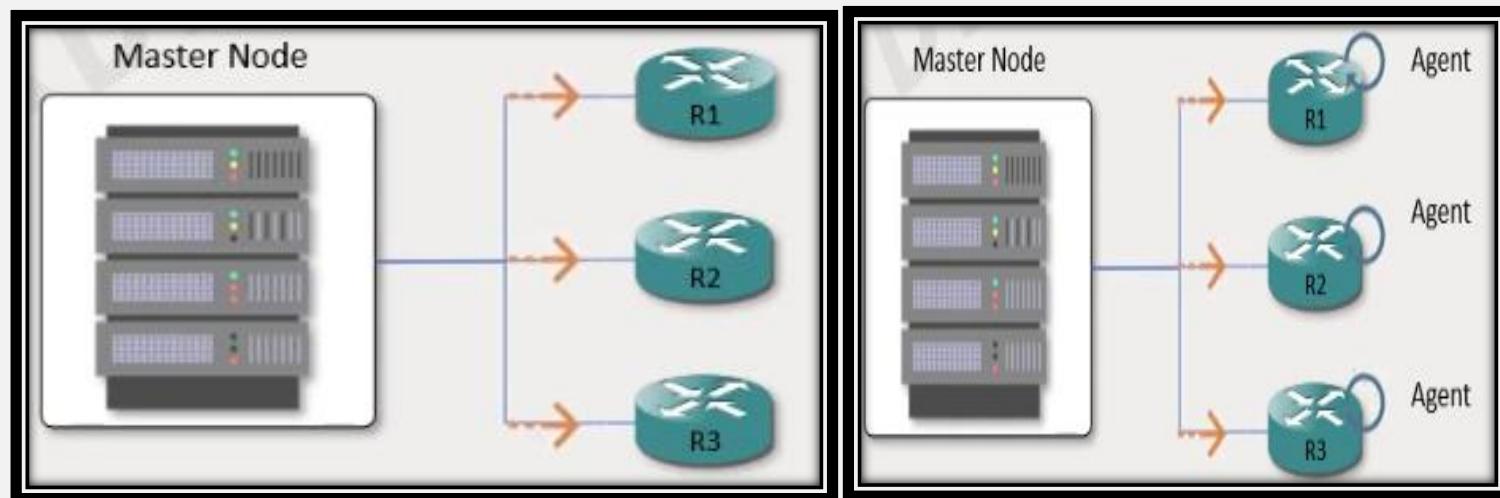
في هذه الطريقة يتم تخزين ملفات الإعدادات لأجهزة الشبكة في جهاز رئيسي يسمى **Shard Folder** ويكون لدينا نسخة من الإعدادات موجودة في الأجهزة ونسخة موجودة في الجهاز المركزي ويمكن استخدام برنامج لتسجيل كل التعديلات التي تحدث على ملفات الإعدادات مثل برنامج **GET** ويمكن رفع ملفات الإعدادات لموقع يوفر هذه الخدمة مثل **GitHub** يسمح بتعديل الملفات باستخدام **GUI** التي من شأنها ان تظهر التعديلات التي حصلت ضمن الملفات وتاريخ التعديلات.  
.



**Configuration Management Tools** .ii  
يوجد نوعين لضبط إعدادات الأجهزة:

**Pull Mode**: يتم تنصيب سوالفتوير يسمى **Agent** في جميع أجهزة الشبكة ويتوافق بشكل دوري مع جهاز **Master** الذي يحتوي على الأداة من أجل اكتشاف أي تعديلات على الإعدادات وضبطها وفقاً لهذه الإعدادات ومن الأمثلة التي تستخدم هذا النمط - **Puppet** .SHEF

**Push Mode**: في هذا النمط لا يتم استخدام **Agent** وأنماء يقوم جهاز **Master** بإرسال هذه الإعدادات لأجهزة الشبكة دون الحاجة لطلب منها وتقوم بضبط الإعدادات التي تصلها من جهاز **Master** ومن الأمثلة التي تستخدم هذا النمط **Ansible**.



### iii. **Configuration Templates and Variables**

تعد من أهم خواص ال **Configuration Management Tools** يتم استخدام ثلاثة ملفات في جهاز **Master** من أجل ضبط إعدادات الأجهزة...  
في هذا الملف يتم تحديد أسماء وعنوانين للأجهزة التي سيتم تطبيق الإعدادات عليها ويمكن وضعها ضمن مجموعات. **Inventory File**

DNS: يتم وضع القيم التي يمكن أن تتغير من جهاز إلى آخر مثل IP and Name .Server .  
 يتم وضع الأوامر التي يجب تنفيذها في الأجهزة مع أسماء المتغيرات وليس القيم.

Hosts (Inventory) File	R1 Variables File	R2 Variables File
<pre>[Routers] R1 ansible_host=192.168.122.101 R2 ansible_host=192.168.122.102</pre>	<pre>name: Router1 ip: 192.168.1.1 mask: 255.255.255.0 pool_name: POOL1 network: 192.168.1.0 access_list_name: ACL wild_card_mask: 0.0.0.255 next_hop_address: 192.168.122.1</pre>	<pre>name: Router2 ip: 192.168.2.1 mask: 255.255.255.0 pool_name: POOL2 network: 192.168.2.0 access_list_name: ACL wild_card_mask: 0.0.0.255 next_hop_address: 192.168.122.1</pre>
Routers Variables File	Jinja2 Template with Variables	
<pre>ansible_ssh_user: "admin" ansible_ssh_pass: "Cisco123" ansible_network_os: "ios" ansible_connection: "network_cli" dns_server: 8.8.8.8</pre>	<pre>hostname {{name}} interface Fastethernet0/1   ip address {{ ip }} {{ mask }}   ip nat inside   no shutdown interface Fastethernet0/0   ip nat outside   ip dhcp pool {{ pool_name }}   network {{ network }} {{ mask}}   default-router {{ ip }}   dns-server {{ dns_server }}   ip access-list standard {{ access_list_name }}   permit {{ network }} {{ wild_card_mask }}   ip nat inside source list {{ access_list_name }} interface Fastethernet 0/0 overload   ip route 0.0.0.0 0.0.0.0 {{ next_hop_address }}</pre>	

- 1 : **Ansible** يتوفر منها نسخة مدفوعة و نسخة مجانية، من خواص أداة ال **Ansible** أنها: (تعتبر الأسهل)
- (1) : ليست بحاجة لتنصيب Agent على الجهاز الهدف للتواصل معه.
- (2) : مبنية على لغة Python Based (Python2, Python3)
- (3) : تستخدم ال SSH أو ال NETCONF لتنصل و تطبق التعليمات (الكودات) على التجهيزات.
- (4) : يمكن استعمالها من أنظمة ال Linux/mac
- (5) : تستخدم هذه اللغة لتعريف ال Yaml Syntax

بعد تنصيبه، يقوم بإنشاء عدة ملفات نصية خاصة به وهي:

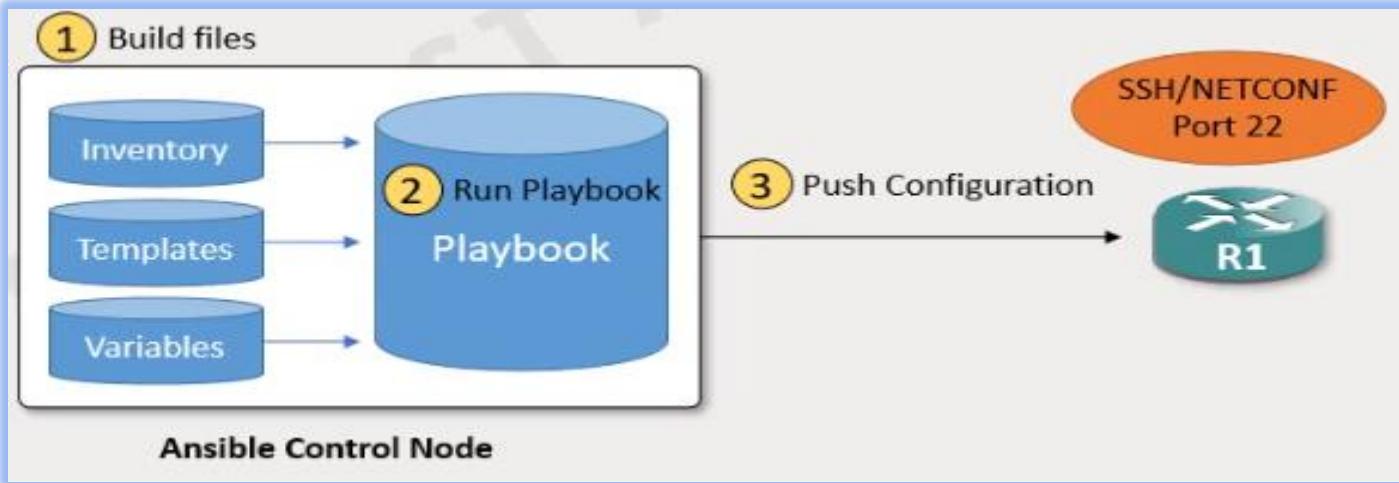
Playbooks : يتم بها جمع ال Actions التي سيقوم Ansible بتنفيذها.

Inventory : تحتوي على ال Hostnames الخاصة بالأجهزة (يمكن تقسيم الأجهزة إلى مجموعات لتسهيل التعامل معها، مثل مجموعة Access Switches أو مجموعة خاصة بال core أو حسب المناطق الجغرافية ...), إضافة لمعلومات مختلفة عنها.

Templates : تستخدم لغة Jinja2 في ال Templates التي تحتوي إعدادات عامة عن الأجهزة مع Variables متعلقة بها.

Variables : ملفات YML تحتوي Variables خاصة بملفات ال Templates.

يتم تجميع تعليمات ال Ansible في Playbook (يستخدم لغة Yaml) ثم ترسل دفعه واحدة في حال أردنا إضافة تعليمات كثيرة، تستخدم ال Ansible وظيفة Push (تدعمها وتعتمد عليها) لإيصال تعليمات للجهازه وتدعم وظيفة ال Pull (لا تعتمد عليها جدا).



**ZTP**: للاستفادة من ميزة **Ansible** Zero Touch Provisioning نقوم بضبط التجهيزات الشبكة لتعمل في **DHCP** مع أداة **DHCP option** حيث يتم ضبط **DHCP Server** عن طريق **ZTP Script** يقوم بمنع عنوان **DHCP Server** عن طريق **ZTP Script** عن طريق **DHCP Server** لتقديم التجهيزات الشبكة عن استلامها لـ **DHCP Server** والذهاب إلى **DHCP Server** وتحميم وتنفيذ **ZTP Script**.

- **Puppet**: هي أداة تستخدم لضبط وإدارة التجهيزات الشبكة (تستخدم غالباً من قبل Sys Admins أيضاً), ممكن تنصيبها على **أنظمة لينكس** ومن خواصها:

**Require Agent (1)**: تحتاج لتنصيب **Agent** على الجهاز الهدف, لذلك من ناحية **Cisco** لا يمكن استخدامها إلا على سويتشات **Nexus 9K** بعد تفعيل خاصية **Guest Shell** لهم والتي تعتمد على **Linux**

**Ruby-based (2)**: على عكس **Ansible** فإنها لا تعتمد على لغة **Ruby**, بل تعتمد على لغة **Python**

**Pull-model (3)**: تدعم وتعتمد على الـ **Pull** أكثر من الـ **Push**

**Declarative Structure Language (DSL) Domain-Specific-Language (4)**: أو كما تعرف بـ **Syntax** وهي اللغة التي يتم بها تعريف

يتم استخدام **Puppet Master** (برنامج يعمل على نظام Linux) للتحكم بجميع ال Agents حيث أن نسخة ال Puppet Master المجانية تتيح التحكم ب 10 Agent.

يتم في اتصالات Puppet استخداماً بورت **TCP 8140**, بعد تنصيب ال Puppet Master, وفي جهاز Master يوجد عدة ملفات:

1 : **Manifests** : تحتوي على الإعدادات الخاصة بالتجهيزات الشبكية وتضم العديد من ال Classes ,Resources ,Classes

2 : **Module** : هي أكبر محتويات ال Manifest وتحتوي على العديد من ال Classes

3 : **Classes** : التي تنظم البيانات داخل ال Manifest وتحتوي على ال Resources

4 : **Resources** : التي تحدد ما الوظيفة التي يتم استخدامها (تحديد IP جديد لمنفذ معين مثلاً)

5 : **Templates** : يقوم Puppet Master باستخدامها لإنشاء ال manifests, variables و ذلك عبر إدخال إليها.

يتم تجميع تعليمات ال Manifest (ال DSL Syntax ) داخل Playbook (ملف مثل ال Ansible Playbook في ال Puppet Manifest ) ثم إرسالها.

حيث كل فترة زمنية (Default time = 30 min) وهي خاصية قوية جداً يقوم ال Agent بالتواصل مع ال Master للتأكد من تزامن تعليماته وصحتها، وفي حال وجد أي تغيير يقوم بإرسال طلب Pull من ال Master ليأخذ هذه التعليمات الجديدة.



**Chef -3:** هي أداة تستخدم غالبا لضبط التجهيزات الشبكية برمجيا و من خواصها:

*Loved by Developers /1*

*Ruby-Based /2*: تعتمد على لغة Ruby مثل ال Puppet ، و تعتبر الأصعب من بين ال IAC

*Agent-Based /3*: تحتاج لتنصيب Agent على الجهاز الهدف حيث يكون لدينا Central Server الذي يحتوي على جميع الإعدادات المحفوظة داخل Cook Book ، وهذه ال Recipes (مجموعة من ال Resources التي تحدد ال Config Policy الخاص بال Node ) تكتب من قبل ال Developer على ال Node Workstation هو الجهاز الذي تحكم به والمنصب عليه ال Agent

للوصول لداخل ال Cook Book ، يوجد مجلد اسمه Chef-repo على سطح المكتب الخاص بال Workstation Node للدخول إلى ال Cook Book وإدخال تعليمات جديدة (Recipes)

يحتوي ال Central Server على مجموعة من ال Cook Books وكل مجموعة تكون متضمنة داخل Book Shelf ، حيث يحتوي ال Central Server على العديد من ال Book Shelves ويقوم ال Workstation Node برفع ال Cook Book إلى ال Central Server بواسطة أداة تسمى Knife ، مثل ال Puppet ، تقوم ال Node التي تستخدم ال Chef ب Pull للإعدادات من ال Central Server وباستخدام بورت TCP 10002 يقوم ال Push Central Server

تم تصميم Chef, Puppet, Ansible بشكل أساسي لإدارة ال Server System وليس الشبكة، تعتبر ال Ansible الأنسب لإدارة الشبكة لعدم الحاجة ل Agent.

## 17- Comparing Ansible, Puppet, Chef: {CNNA., #7}

Comparing Ansible, Puppet, and Chef			
	Ansible	Puppet	Chef
Managed Node Requirements	Agentless	Agentless and Agent Based	Agent Based
Deployment Method	Push Model	Pull Model	Pull Model
Master Server	Linux Only	Linux/Windows	Linux/Windows
File that lists actions	Playbook	Manifest	Recipe/Runlist
Transport Mechanism	SSH/NETCONF	REST	REST
Port Number	22	8140	10002

{Documentation., 2020 #12} {Ansible., 2022 #10} **التطبيق العملي لتقنية SDN باستخدام أداة ANSIBLE:**

↳ IAC = Infrastructure as code

هو شكل من أشكال الشبكات، يعتمد على أدوات مثل Ansible, Puppet, Chef ومن فوائده تفادي المشاكل التالية:

**Configuration Drift .1**: خروج ضبط وإعدادات أحد التجهيزات عن اتساق (Consistency) باقي إعدادات أجهزة الشبكة، أو عن الإعدادات التي تم الاتفاق عليها مسبقاً من قبل إدارة الفريق عندما تم وضع الجهاز في البداية في الشبكة نتيجة لحدث طاري، أو تعديل أحد المسؤولين له بدون إعلامه لباقي الفريق ...

حيث في حال استخدام طريقة الضبط التقليدية، لا يمكن معرفة من قام بهذه التعديلات ولا يمكن استرجاع الإعدادات القديمة، ولا معرفة أي إعدادات بالضبط تم تعديلها. خاصة في حال تم تطبيق عدة تعديلات في فترات زمنية قريبة ومتفاوتة.

ولحل مشكلة ال Configuration Drift بشكل نهائي، يتم وصل جميع التجهيزات بالإضافة إلى Configuration Management Tool مع GitHub مثل Centralized repository يتم رفع كل الإعدادات الخاصة بالأجهزة إليه. بحيث تكون هذه الإعدادات هي ال Ideal وعند حدوث أي تعديل يدوي (ليس عن طريق ال Management Tool على أي جهاز شبكي، سيتم مقارنة الإعدادات الحديثة للجهاز مع الموجودة في ال Centralized repository وإرسال أشعار لفريق مديرى الشبكة في حال عدم توافقهما.

```

WendellOdom committed 6 days ago Verified

Showing 1 changed file with 2 additions and 2 deletions.

v 4 BR1.txt

  @@ -5,6 +5,6 @@ router ospf 1
  5   5     router-id 1.1.1.1
  6   6     !
  7   7     interface gigabitethernet0/0
  8   -   description connected to SW1
  9   -   ip address 10.1.1.1 255.255.255.0
  8   +   description connected to SW2
  9   +   ip address 10.1.22.1 255.255.255.0
  10  10

```

**Figure 19-3** Showing File Differences in GitHub

.2 **Idempotence**: تطبيق تعليمية مطبقة مسبقا، حتى لو كانت نفسها تماما، فمن الطرق التقليدية التي كانت تستخدم لضمان استرجاع الإعدادات الخاصة بالتجهيزات، هي نسخ الإعدادات الخاصة بالتجهيزات الشبكية إلى موقع شبكي مشارك مع كل فريق بشكل دوري أو عند حدوث أي تعديل على أي من التجهيزات (مثلا TFTP Server Admins).

## من أهم النقاط التي تقوم بتوفيرها أدوات ال Configuration Management :Configuration Templates and Variables

تعد من أهم خواص ال Configuration Management Tools، حيث تسمح بإنشاء Template خاص بالإعدادات التي تكون عادة شبه متماثلة في التجهيزات الشبكية.

حيث يتم إنشاء Template تحتوي الإعدادات المراد تطبيقها على التجهيزات، مع تحديد القيم الممكن اختلافها بين التجهيزات ك Variables قبل الإدخال يدوياً:

**Example 19-2** *Jinja2 Template with Variables Based on Example 19-1*

**Key Topic**

[Click here to view code image](#)

```
hostname {{hostname}}
!
interface GigabitEthernet0/0
    ip address {{address1}} {{mask1}}
    ip ospf {{OSPF_PID}} area {{area}}
!
interface GigabitEthernet0/1
!
interface GigabitEthernet0/1/0
    ip address {{address2}} {{mask2}}
    ip ospf {{OSPF_PID}} area {{area}}
!
router ospf {{OSPF_PID}}
    router-id {{RID}}
```

حيث يتم استخدام هذه ال **Template** من قبل ال **Configuration Tools** مع قيم ال **Variables** التي يقوم بتحديدها لكل تجهيزه على حدا لتسهيل ضبط التجهيزات بإعدادات كبيرة:

**Example 19-3 YAML Variables File Based on Example**  
19-2

Click here to view code image

```
---  
hostname: BR1  
address1: 10.1.1.1  
mask1: 255.255.255.0  
address2: 10.1.12.1  
mask2: 255.255.255.0  
RID: 1.1.1.1  
area: '11'  
OSPF_PID: '1'
```

فيقوم ال **Configuration Management Tool** بمعالجة ال **Template** مع ال **Variables** التي تم تحديد قيمها من قبل **Admins** وإرسالها للتجهيز المراد ضبطها بها.



{Documentation., 2020 #12}

يتوفر منها نسخة مدفوعة ونسخة مجانية (Open Source) ويشرف على تطويرها شركة Red Hat.

من خواص أداة ال **Ansible** أنها: (تعتبر الأسهل)

.i. **Agent-less**: ليست بحاجة لتنصيب Agent على الجهاز الهدف للتواصل معه.

.ii. **Python Based**: مبنية على لغة Python (Python 2, Python3).

.iii. **SSH / NETCONF**: تستخدم ال SSH أو ال NETCONF لتتصل وتطبق التعليمات (الأكواد) على التجهيزات.

.iv. **Linux-mac**: يمكن استعمالها من أنظمة ال Linux/mac.

.v. **Yaml Syntax**: تستخدم هذه اللغة لتعريف ال.

بعد تنصيبه، نقوم بإنشاء عدة ملفات نصية خاصة به وهي:

☞ **Playbooks**: يتم بها جمع المهام (Tasks) التي سيقوم Ansible بتنفيذها.

☞ **Inventory**: تحتوي على ال Hostnames الخاصة بالأجهزة الشبكية التي ستقوم الأداة بضبط التعامل معها (يمكن تقسيم الأجهزة إلى مجموعات لتسهيل التعامل معها، مثل مجموعة بال Access Switches و مجموعة خاصة بال core أو حسب المناطق الجغرافية ...)، إضافة لمعلومات مختلفة عنها.

وعند تطبيق المعلومات يتم تطبيقها على مجموعة كاملة أو على جهاز معين من أحد المجموعات.

**Templates** : تستخدم لغة **Jinja2** في ال **Templates** التي تحتوي على الإعدادات المراد تنفيذها على الأجهزة مع أسماء المتغيرات (وليس قيم التغييرات)، حيث أن قيم المتغيرات تكون معرفة داخل ملفات المتغيرات.

**Variables** : ملفات بصيغة **YML** تحتوي على **Variables** خاصة بملفات ال **Templates**.

يتم فيها وضع القيم التي يمكن أن تتغير من جهاز إلى آخر. مثل (الاسم، عنوان ال IP)، ومنها ما يتم فيه وضع قيم مشتركة بين جميع الأجهزة. مثل (عنوان ال DNS، اسم المستخدم وكلمة المرور الخاصة باتصال ال SSH).

#### Hosts (Inventory) File

```
[Routers]
R1 ansible_host=192.168.122.101
R2 ansible_host=192.168.122.102
```

#### R1 Variables File

```
name: Router1
ip: 192.168.1.1
mask: 255.255.255.0
pool_name: POOL1
network: 192.168.1.0
access_list_name: ACL
wild_card_mask: 0.0.0.255
next_hop_address: 192.168.122.1
```

#### R2 Variables File

```
name: Router2
ip: 192.168.2.1
mask: 255.255.255.0
pool_name: POOL2
network: 192.168.2.0
access_list_name: ACL
wild_card_mask: 0.0.0.255
next_hop_address: 192.168.122.1
```

#### Routers Variables File

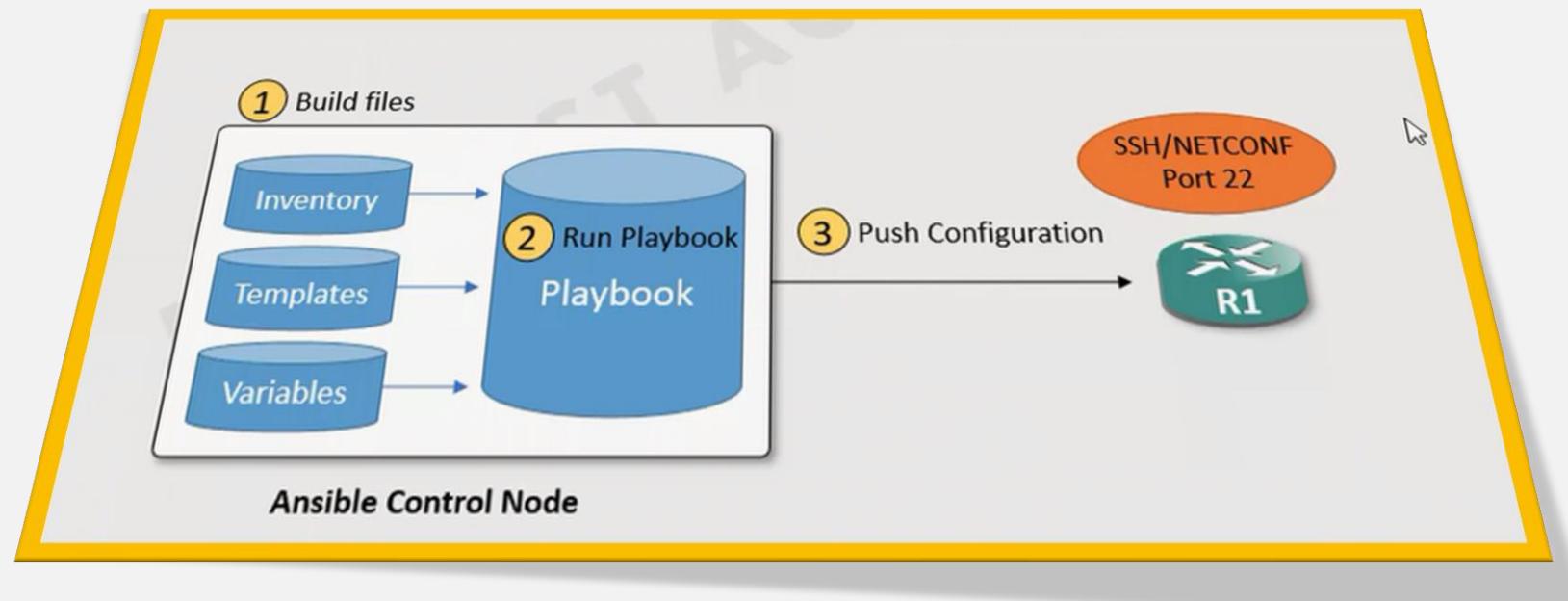
```
ansible_ssh_user: "admin"
ansible_ssh_pass: "Cisco123"
ansible_network_os: "ios"
ansible_connection: "network_cli"
dns_server: 8.8.8.8
```

#### Jinja2 Template with Variables

```
hostname {{name}}
interface Fastethernet0/1
  ip address {{ ip }} {{ mask }}
  ip nat inside
  no shutdown
interface Fastethernet0/0
  ip nat outside
  ip dhcp pool {{ pool_name }}
    network {{ network }} {{ mask}}
    default-router {{ ip }}
    dns-server {{ dns_server }}
  ip access-list standard {{ access_list_name }}
    permit {{ network }} {{ wild_card_mask }}
  ip nat inside source list {{ access_list_name }} interface Fastethernet 0/0 overload
  ip route 0.0.0.0 0.0.0.0 {{ next_hop_address }}
```

يتم تجميع تعليمات ال Ansible في **Playbook** (يستخدم لغة Yaml) بشكل مهام (Tasks)، ثم ترسل دفعه واحدة في حال أردنا إضافة تعليمات كثيرة باستخدام بروتوكول SSH والمنفذ TCP22 أو باستخدام ال .NETCONF

تستخدم الـ Ansible وظيفة Push (تدعمها وتعتمد عليها) لإيصال التعليمات للتجهيزات وتدعم وظيفة ال Pull (لا تعتمد عليها).



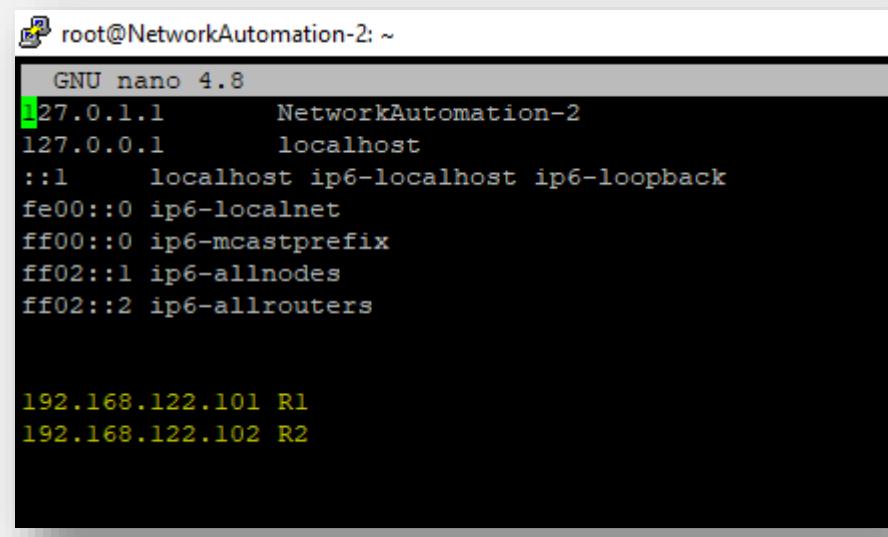
## مثال عن العمل على أداة Ansible :

يجب ألا ننسى أبداً، انه حتى لو تبيّن لنا أن ضبط جهاز بالطريقة التقليدية (CLI) سيأخذ نفس الوقت أو حتى أقل من الطريقة التي تعتمد على أداة Ansible فإن هدفنا هو النظر لما أكبر من ذلك، فبالتأكيد عند العمل في شبكة مؤلفة من عشرات أو حتى من مئات الأجهزة الشبكية (التي تحوي على الآلاف من أسطر التعليمات) فإن Ansible ستقوم باختصار الكثير من الوقت.

لنبدأ بالعمل، كل ما يجب فعله من جهة التجهيزات الشبكية هو التأكد من تفعيل اتصال ال SSH وتحديد اسم مستخدم وكلمة مرور عليها، كون أداة Ansible تعتمد في اتصالها وعملها (Push Mode) مع جميع الأجهزة على بروتوكول ال SSH، أما بالنسبة للجهاز المركزي المنصّبة عليه أداة Ansible فلدينا عدة خطوات يجب القيام بها:

**أولاً:** سنقوم بالتعديل على ملف etc/hosts (ملف إعدادات ال DNS) الخاص بجهاز Linux المنصّبة عليه أداة Ansible وذلك لكي ندخل أسماء وعنوانين الأجهزة الشبكية إلى ال DNS الخاص بهذا الجهاز.

**مثال** عن إدخال اسم وعنوان راوترين من الشبكة لدينا:

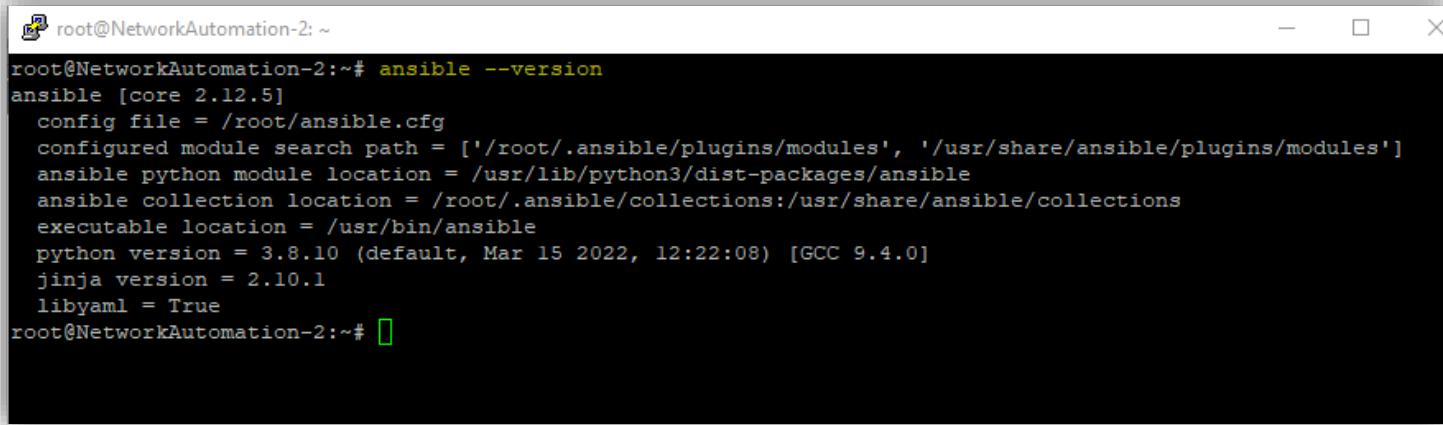


The screenshot shows a terminal window titled "root@NetworkAutomation-2: ~". The command "nano /etc/hosts" is running. The file contains the following content:

```
GNU nano 4.8
127.0.1.1      NetworkAutomation-2
127.0.0.1      localhost
::1            localhost ip6-localhost ip6-loopback
fe00::0         ip6-localnet
ff00::0         ip6-mcastprefix
ff02::1         ip6-allnodes
ff02::2         ip6-allrouters

192.168.122.101 R1
192.168.122.102 R2
```

**ثانياً:** التأكد من تثبيت ال Package الخاصة بال Ansible لدينا وذلك باستخدام التعليمة: <Ansible --version>



```
root@NetworkAutomation-2:~# ansible --version
ansible [core 2.12.5]
  config file = /root/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3/dist-packages/ansible
  ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
  executable location = /usr/bin/ansible
  python version = 3.8.10 (default, Mar 15 2022, 12:22:08) [GCC 9.4.0]
  jinja version = 2.10.1
  libyaml = True
root@NetworkAutomation-2:~#
```

حال لم تكن أداة Ansible مثبتة على جهاز Linux لدينا، نقوم بتنسيتها عبر إتباع الخطوات التالية:

- ☞ تحديث معلومات الحزم لدينا باستخدام التعليمية: <sudo apt update>
- ☞ إضافة ال Repository الخاص ب Ansible إلى مجموعة ال Repositories الخاصة بحزم لينكس في الجهاز باستخدام التعليمية:  
`<Sudo apt-add-repository –yes –update ppa:ansible/ansible>`
- ☞ تثبيت حزمة Ansible <code><Sudo apt install ansible></code>

**ثالثاً:** البدء بضبط الملفات الخاصة ب Ansible والتي ذكرناها سابقاً:

- ملف الإعدادات الرئيسية لأداة Ansible :Ansible.cfg (ملف إجباري)
- ملف الأجهزة التي سيتم التعامل معها من قبل أداة Ansible :Ansible hosts (ملف إجباري)

- ملفات ال **Variables** التي سنقوم بإنشائها لتعرف ال **Variables** المشتركة بين جميع الأجهزة وال **Variables** الخاصة بكل تجهيزه مع قيمها واستخدامها لاحقاً في ملفات ال **Playbooks** (ملف اختياري)
  - ملف **Template** الذي سيحتوي على الأوامر المراد تطبيقها على الأجهزة باستخدام المهام المحددة في ملفات ال **Playbooks** (ملف اختياري)
  - ملف أو ملفات ال **Playbooks** التي ستحتوي على المهام التي نريد تنفيذها على التجهيزات الشبكية (ملف اختياري)
- تقوم أداة Ansible بحفظ ملفات الإعدادات الخاصة بها وملفاتها الأساسية بشكل افتراضي في المسار `/etc/ansible` / بشكل بنية شجرية حيث يكون لدينا بشكل افتراضي ملفين أساسين هما: (hosts - Ansible.cfg)

```
root@NetworkAutomation-1:/etc/ansible# tree
.
|-- ansible.cfg
-- hosts
    |
    |-- hosts
0 directories, 2 files
root@NetworkAutomation-1:/etc/ansible#
```

ملف `hosts` الذي يطلق عليه اسم ملف ال **Inventory** والذي يتم بداخله تعريف الأجهزة التي سيتم التخاطب والتعامل معها بواسطة Ansible ملف `ansible.cfg` الذي يعد ملف الإعدادات الخاص ب Ansible ضبط الملفات الخاصة بأداة **:Ansible**

### (host file) :Inventory file

يوجد هذا الملف بشكل افتراضي عند تثبيت حزمة Ansible على الجهاز، ويكون ضمن المسار `/etc/ansible/hosts`

وهذا المسار يكون معزف ضمن ملف الإعدادات الرئيسية الخاصة ب Ansible (الملف ansible.cfg) ضمن القسم default وممكن التعديل على المسار الموجود فيه ملف hosts من هنا.

يتم في ملف ال hosts تحديد أسماء وعناوين الأجهزة التي سيتم التعامل معها من قبل الأداة Ansible، ويمكن تعريف هذه الأجهزة ضمن مجموعات (تعرف المجموعات ضمن []) وفصلها عن بعض بما يناسب مدير الشبكة.

كما يمكن تعريف **Ranges** **Variables** في ملف ال Inventory هي لغة Yml . حيث أن البنية التي يقوم عليها ملف ال Inventory هي لغة Yml . وعند تطبيق الإعدادات المحددة ضمن المهام في ملف ال Playbook ، يتم تطبيقها على المجموعة المختارة (بما يتضمن كل الأجهزة ضمن هذه المجموعة) أو على جهاز أو مجموعة أجهزة محددة.

NetworkAutomation-1 x ● R1 ● R2

GNU nano 4.8 ansible.cfg

```
config file for ansible -- https://ansible.com/
# -----
# nearly all parameters can be overridden in ansible-playbook
# or with command line flags. ansible will read ANSIBLE_CONFIG,
# ansible.cfg in the current working directory, .ansible.cfg in
# the home directory or /etc/ansible/ansible.cfg, whichever it
# finds first

[defaults]

# some basic default values...

#inventory      = /etc/ansible/hosts
#library        = /usr/share/my_modules/
#module_utils   = /usr/share/my_module_utils/
#remote_tmp     = ~/.ansible/tmp
#local_tmp      = ~/.ansible/tmp
#plugin_filters_cfg = /etc/ansible/plugin_filters.yml
#forks          = 5
#poll_interval  = 15
#sudo_user      = root
#ask_sudo_pass  = True
#ask_pass       = True
#transport      = smart
#remote_port    = 22
#module_lang    = C
#module_set_locale = False

# plays will gather facts by default, which contain information about
# the remote system.
#
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
[ Read 490 lines ]
```

^G Get Help ^O Write Out ^W Where Is ^K Cut Text ^J Justify ^C Cur Pos M-U  
^X Exit ^R Read File ^A Replace ^U Paste Text ^T To Spell ^Y Go To Line M-E

Solar-PuTTY free tool © 2013

## Ansible Inventory



```
ansible_host=IP ADDRESS/DOMAIN NAME
ansible_user=USERNAME
ansible_password=PASSWORD
ansible_port=PORT NUMBER
ansible_connection=CONNECTION PROTOCOL
ansible_become=TRUE/FAKE
ansible_become_user=USERNAME
ansible_become_pass=PASSWORD
ansible_become_method=PRIVELEGED METHOD
```

## مثال عن محتوى ملف hosts

حيث تم هنا تعريف المجموعات Aleppo\_Routers, Damascus\_Router التي تتضمن الأجهزة: R1,R2,R3,R4,R5,R6 مع تعريف المجموعة Routers التي تتضمن جميع التجهيزات للمساعدة في أي عمليات Troubleshooting لتجهيزات الشبكة بشكل كامل مع استخدام عدة variables مع كل الأجهزة لتوضيح عناوينهم ومعلومات المصادقة الخاصة بهم وغيرها..

```
root@NetworkAutomation-2: ~
GNU nano 4.8                               hosts22
[Aleppo_Routers]

R1 ansible_host=10.252.1.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R2 ansible_host=10.252.2.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R3 ansible_host=10.252.3.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true

[Damascus_Routers]

R4 ansible_host=10.253.1.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R5 ansible_host=10.253.2.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R6 ansible_host=10.253.3.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true

[Routers]
R1 ansible_host=10.252.1.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R2 ansible_host=10.252.2.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R3 ansible_host=10.252.3.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R4 ansible_host=10.253.1.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R5 ansible_host=10.253.2.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
R6 ansible_host=10.253.3.1 ansible_user=admin ansible_password=Cisco!23 ansible_become=true
```

وهنا قمنا بشكل آخر بتعريف جميع الأجهزة في ملف hosts يجمع المعلومات المشتركة ضمن مجموعة واحدة عوضاً عن تكرارها في كل سطر خاص بجهاز:

```
root@NetworkAutomation-2: ~
GNU nano 4.8
hosts22

[Aleppo_Routers]
R1 10.252.1.1
R2 10.252.2.1
R3 10.252.3.1

[Damascus_Routers]
R4 10.253.1.1
R5 10.253.2.1
R6 10.253.3.1

[Routers]
R1 10.252.1.1
R2 10.252.2.1
R3 10.252.3.1
R4 10.253.1.1
R5 10.253.2.1
R6 10.253.3.1

[all:vars]
ansible_connection=ssh
ansible_user=admin
ansible_password=Cisco!23
ansible_become=true
```

للاستعلام عن جميع الـ hosts الموجود داخل المعرفة hosts نقوم باستخدام التعليمية: {Ansible –list-hosts all}

```

root@NetworkAutomation-2: ~
root@NetworkAutomation-2:~# ansible --list-hosts all
hosts (2):
  R1
  R2
root@NetworkAutomation-2:~#

```

## :Variable files

هي ملفات اختيارية تكون بصيغة YML، ممكن أن نقوم ضمنها بتعريف متغيرات خاصة بقيم مشتركة بين جميع التجهيزات (مثل ال DNS Server, DHCP server, NTP server, SSH username, SSH password ..... خاصية به مع قيمها (مثل: عنوان ال IP ,Hostname ,OSPF area ....)

مثال عن بعض ملفات ال Variables الخاصة بمتغيرات خاصة بكل جهاز (R1.yml, R2.yml)

```

File Edit Selection View Go Run Terminal Help
R1.yml - Visual Studio Code
Restricted Mode is intended for safe code browsing. Trust this window to enable all features. Manage Learn More
! R1.yml X
C: > Users > Ali > Desktop > ! R1.yml > ...
1  ---
2  name: Router1
3  ip: 192.168.1.1
4  mask: 255.255.255.0
5  pool_name: POOL1
6  network: 192.168.1.0
7  access_list_name: ACL
8  wild_card_mask: 0.0.0.255
9  next_hop_address: 192.168.122.1
10

```

R2.yml - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Learn More](#)

R2.yml X

C: > Users > Ali > Desktop > ! R2.yml > ...

```
1 ---  
2 name: Router2  
3 ip: 192.168.2.1  
4 mask: 255.255.255.0  
5 pool_name: POOL2  
6 network: 192.168.2.0  
7 access_list_name: ACL  
8 wild_card_mask: 0.0.0.255  
9 next_hop_address: 192.168.122.1  
10  
11
```

وملف [Routers.yml](#) الذي يحتوي على متغيرات مشتركة بين جميع الأجهزة وقيمها:

Routers.yml - Visual Studio Code

Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Learn More](#)

Routers.yml X

C: > Users > Ali > Desktop > ! Routers.yml > dns\_server

```
1 ---  
2 ansible_ssh_user: "admin"  
3 ansible_ssh_pass: "Cisco123"  
4 ansible_network_os: "ios"  
5 ansible_connection: "network_cli"  
6 dns_server: 8.8.8.8
```

حيث يتم استخدام المتغيرات المعرفة داخل هذه الملفات داخل Templates الخاصة بالإعدادات المراد تطبيقها على الأجهزة، وتقوم Ansible حينئذ بقراءة قيم هذه المتغيرات من ملفات ال Variables

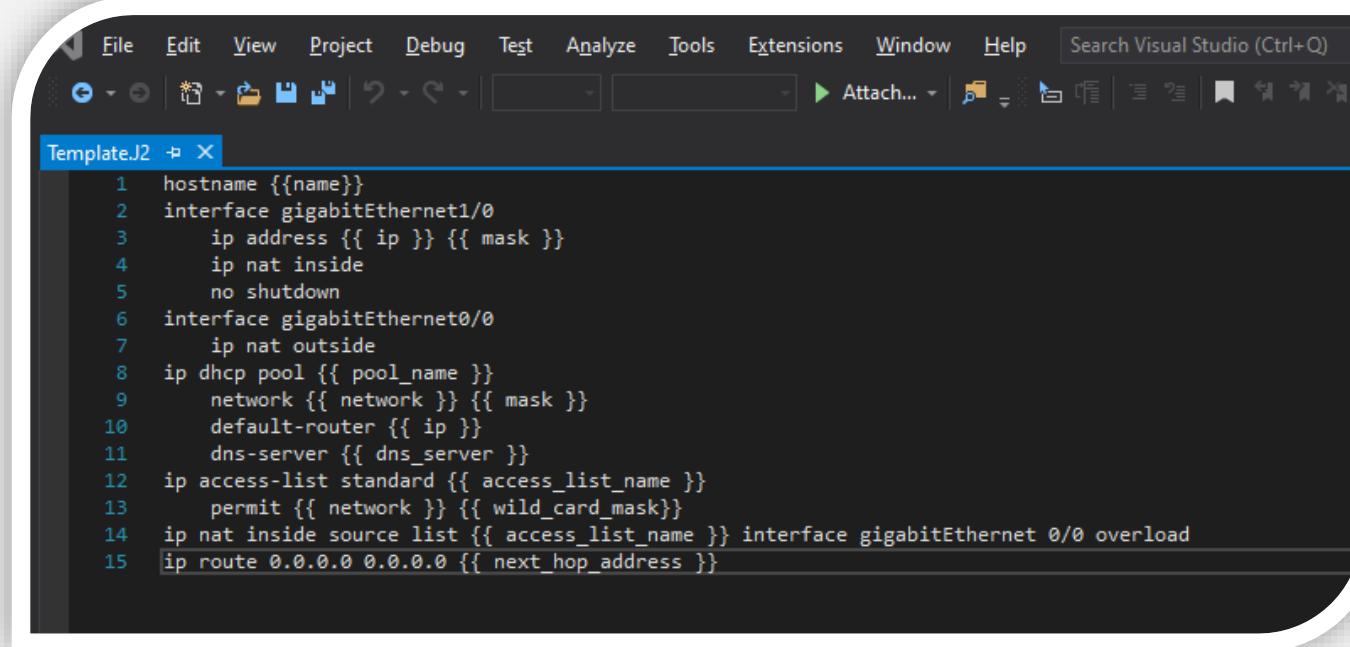
☞ يجب الإنتباه لوجود مجموعة من ال Variables المعرفة بشكل افتراضي لدى أداة Ansible، تدعى ب **Magic Variables** مثل: ....hostvars, groupnames, groups, environment

حيث يجب الإنتباه لعدم تعريف متغيرات باسمها لعدم ظهور أي أخطاء في ملفات ال Playbooks التي سنستعمل هذه المتغيرات ضمنها.  
لتتعرف أكثر عن ال Magic Variables، نستطيع زيارة الموقع [Docs.ansible](#) وقراءة المزيد عن جميع ال Variables المعرفة بشكل تلقائي لدى Ansible.

## :Template Files

هي ملفات مكتوبة بصيغة Ninja2، يتم فيها وضع الأوامر المراد تنفيذها على الأجهزة الشبكية مع أسماء المتغيرات المعرفة ضمن ملفات ال Variables ويتتم استخدامها من قبل ال Tasks المراد تنفيذها في ملفات ال Playbook.

**مثال** عن ملف Template الذي تم ضمه استخدام ال Variables المعرفة ضمن ملفات ال Variables:



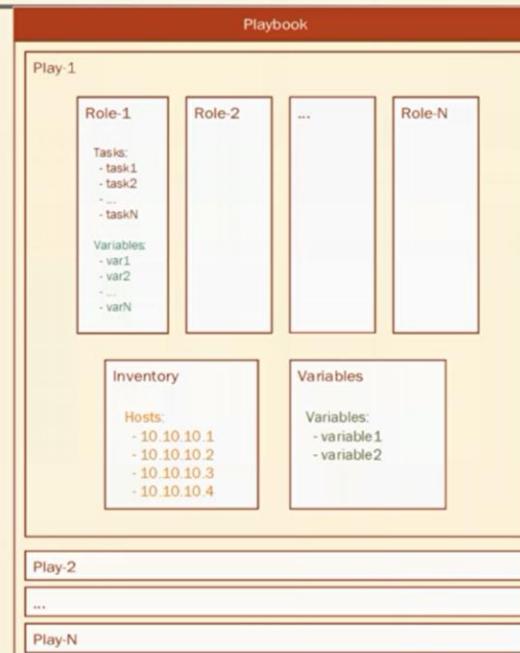
```
1 hostname {{name}}
2 interface gigabitEthernet1/0
3   ip address {{ ip }} {{ mask }}
4   ip nat inside
5   no shutdown
6 interface gigabitEthernet0/0
7   ip nat outside
8   ip dhcp pool {{ pool_name }}
9     network {{ network }} {{ mask }}
10    default-router {{ ip }}
11    dns-server {{ dns_server }}
12   ip access-list standard {{ access_list_name }}
13     permit {{ network }} {{ wild_card_mask}}
14   ip nat inside source list {{ access_list_name }} interface gigabitEthernet 0/0 overload
15   ip route 0.0.0.0 0.0.0.0 {{ next_hop_address }}
```

## :Playbook Files

يتم فيها تحديد Play أو أكثر، حيث كل Play ممكن أن تتضمن مهمة أو عدة مهام (Tasks) التي يجب على الأداة القيام بها كما يتم بها تحديد الأجهزة الهدف عن طريق اختيار جهاز أو مجموعة أجهزة من ملف hosts، بحيث أن البنية الأساسية لملف Playbook تكون بالشكل التالي:

Playback paused

# Playbook Structure



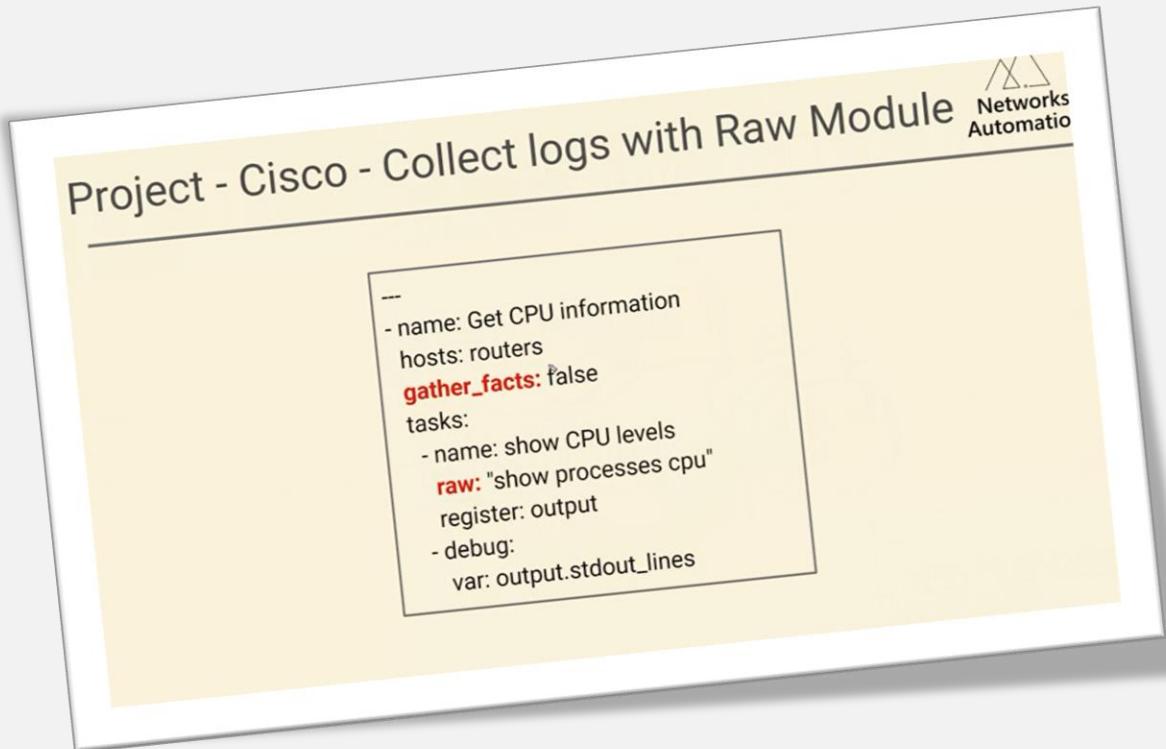
مثال عن ملف Playbook يحتوي على 2 Tasks واحدة تتضمن 2 باسم (Save running to startup when) configure Routers

The screenshot shows a Visual Studio Code window with a dark theme. The title bar says "Playbook.yml - Visual Studio Code". A message at the top of the editor area reads: "Restricted Mode is intended for safe code browsing. Trust this window to enable all features. [Manage](#) [Learn More](#)". The editor pane displays the following YAML code:

```
1 ---  
2   - hosts: Routers  
3     gather_facts: false  
4     tasks:  
5       - name: configure Routers  
6         ios_config:  
7           src: ./templates/template.j2  
8       - name: save running to startup when modified  
9         ios_config:  
10            save_when: modified
```

هنا قمنا بالاستفادة من ملف ال Template الذي قمنا بإنشائه سابقاً لكي يتم تنفيذ التعليمات الموجودة داخله عن طريق ال Task المعرفة باسم .configure Routers

مثال آخر عن ملف Playbook



هنا لم يتم الاستعانة بملف Template ضمن ال Task, بل تم استخدام إحدى ال Modules الخاصة بالتعامل مع تجهيزات cisco من قبل أداة Ansible والذي هو "Raw Module".

:Ansible.cfg File 

هو الملف الأساسي لإعدادات Ansible لا ينصح بالتعديل عليه إلا في حال كان مدير الشبكة يمتلك خبرة كافية للتعامل معه.

## طرق لإرسال التعليمات من أداة Ansible إلى التجهيزات الشبكية:

{Ansible., 2022 #10} هناك طريقتين لإرسال التعليمات من أداة Ansible إلى التجهيزات الشبكية (كلاهما يعتمد على بروتوكول ال SSH):

### *:Add-hoc commands* ⊕

هي تعليمات بسيطة تتيح تنفيذ غرض معين بسرعة وبدون الحاجة لاستعمال Template أو Playbook، ولا يتم حفظ هذه التعليمات لاتاحة استعمالها مجدداً.

تعتمد في تنفيذ عملها على Modules خاصية بنوع التجهيز المراد التعامل معها (Cisco, Juniper, Huawei..)، حيث يتم اختيار ال Module من مجموعة ال Modules المتوفرة ضمن ال Module Library التي تتيحها Ansible لكل نوع من التجهيزات.

الشكل العام لتعليمات add-hoc يكون كالتالي:

## Ad-Hoc Usage



**ansible [pattern] -m [module] -a "[module options]"**



مثال عن استخدام لتعليمية :add hoc

## Ad-Hoc Usage



```
ansible host-1 -m shell -a "free -m" -u networksautomation -k
```

حيث هنا تم تحديد ال host (host-1) من الملف hosts لكي يتم تنفيذ التعليمية عليه مع استخدام ال Module (shell) وال Argument (free -m) و تحديد اسم المستخدم المراد استعماله للاتصال مع الجهاز عن طريق الخيار u ، واستخدام الخيار k- لإدخال كلمة المرور الخاصة بالمستخدم عند الاتصال.

للقيام ب لأي Debug Ad-hoc command نستخدم الخيار **-vvv** :

```

bot@NetworkAutomation-2:~# ansible R1 -m raw -a "show version" -u admin -k -vvv
ansible [core 2.12.5]
config file = /root/ansible.cfg
configured module search path = ['/root/.ansible/plugins/modules', '/usr/share/ansible/plugins/modules']
ansible python module location = /usr/lib/python3/dist-packages/ansible
ansible collection location = /root/.ansible/collections:/usr/share/ansible/collections
executable location = /usr/bin/ansible
python version = 3.8.10 (default, Mar 15 2022, 12:22:08) [GCC 9.4.0]
jinja version = 2.10.1
libyaml = True
using /root/ansible.cfg as config file
SH password:
st_list declined parsing /root/hosts as it did not pass its verify_file() method
script declined parsing /root/hosts as it did not pass its verify_file() method
to declined parsing /root/hosts as it did not pass its verify_file() method
rsed /root/hosts inventory source with ini plugin
tipping callback 'default', as we already have a stdout callback.
tipping callback 'minimal', as we already have a stdout callback.
tipping callback 'oneline', as we already have a stdout callback.
IA: ran handlers
I: ESTABLISH SSH CONNECTION FOR USER: admin
I: SSH: EXEC sshpass -d10 ssh -C -o ControlMaster=auto -o ControlPersist=60s -o StrictHostKeyChecking=no -o "User='admin'" -o ConnectTimeout=5 -o 'ControlPath /root/.ansible/tmp/28022897b9c' -tt R1 'show version'
R1> (0, b'\r\nCisco IOS Software, 7200 Software (C7200-ADVISORIESK9-M), Version 15.2(4)S5, RELEASE SOFTWARE (fc1)\r\nTechnical Support: http://www.cisco.com/echosupport\r\nCopyright (c) 1986-2014 by Cisco Systems, Inc.\r\nCompiled Thu 26-Feb-14 06:51 by prod_rel_team\r\nROM: ROMMON Emulation Microcode\r\nC7200 Software (C7200-ADVISORIESK9-M), Version 15.2(4)S5, RELEASE SOFTWARE (fc1)\r\n\r\nRI uptime is 8 minutes\r\nSystem returned to ROM by unknown reload cause - suspect boot_data[BOOT_COUNT] 0x0, BOOT_COUNT 0, BOOTDATA 19\r\nSystem image file is "tftp://255.255.255.255/unknown"\r\nLast reload reason: unknown rel d cause - suspect boot_data[BOOT_COUNT] 0x0, BOOT_COUNT 0, BOOTDATA 19\r\n\r\nThis product contains cryptographic features and is subject to United States and local country laws governing import, export, transfer and use. Delivery of Cisco cryptographic products does not imply third-party authority to import, export, distribute or use encryption. Importers, exporters, distributors and users are responsible for compliance with U.S. and local country laws. By using this product you agree to comply with applicable laws and regulations. If you are unable to do so, return this product immediately.\r\nA summary of U.S. laws governing Cisco cryptographic products may be found at: https://www.cisco.com/w处处出口/exports/crypto/tool/seccgr.html\r\nIf you require further assistance please contact us by sending email to flexport@cisco.com.\r\nCisco 7206VXR (INPE400) processor (revision A) with 8520K/32768K bytes of memory.\r\nProcessor board ID 4273256517\r\nR7200 CPU at 150MHz, Implementation 39, Rev 2.1, 256KB L2 Cache\r\nOne slot VME midplane, Version 2.1\r\nLast reset from power-on\r\nPCI bus mb0_mbi (Slots 0, 1, 3 and 5) has a capacity of 600 bandwidth points.\r\nCurrent configuration on bus mb0_mbi has a total of 600 bandwidth points.\r\nThe set of PA-2FE, PA-POS-2OC3, and I/O-2FE qualify for "half bandwidth points" consideration, when full bandwidth point counting results in oversubscription, under the condition that only one of the two ports is used. With this adjustment, current configuration on bus mb0_mbi has a total of 600 bandwidth points.\r\nThis configuration has oversubscribed the PCI bus and is not a supported configuration.\r\nPCI bus mb2 (Slots 2, 4, 6) has a capacity of 600 bandwidth points.\r\nCurrent configuration on bus mb2 has a total of 400 bandwidth points.\r\nThis configuration is within the PCI bus capacity and is supported.\r\nPlease refer to the following document "Cisco 7200 Series Port Adaptor Hardware Configuration Guidelines" on Cisco.com <http://www.cisco.com>\r\nfor c7200 bandwidth points oversubscription and usage guidelines.\r\nWARNING: PCI bus mb0_mbi Exceeds 600 bandwidth points.\r\nEthernet interface\r\nGigabit Ethernet interfaces\r\nSerial interfaces\r\n509K Bytes of NVRAM\r\n8192K Bytes of Flash internal memory (Sector size 256K)\r\nConfiguration register is 0x2102\r\nb'Shared connection to R1 closed.\r\n'
| CHANGED | rc=0 >

```

حيث أن جميع الخرج الذي باللون الأزرق هو عبارة عن **debugging** لكل الأحداث والعمليات التي تمت إلى أن تم تنفيذ الـ **argument المختار على الجهاز الهدف وإرجاع النتيجة إلى الجهاز المتحكم ليتم عرضها كما مبين في اللون الأصفر.**

## **:Playbooks files⊕**

يتم بها تحديد ال hosts المراد تنفيذ ال tasks عليهم، وتعتمد أيضاً في عملها على ال Modules المحددة ضمن ال Tasks الخاصة بال Play ، مع إمكانية الاستفادة من ملفات ال Variables Template وال .

يتم تنفيذ ال Tasks المعروفة داخل ال Playbook على التسلسل.

الشكل العام لتنفيذ تعليمية Playbook هو: **(ansible-playbook playbook-name)**

للحصول على صحة ال Syntax الخاص ب Playbook نستخدم الخيار **<--syntax-check>**

```
root@NetworkAutomation-2: ~
root@NetworkAutomation-2:~# ansible-playbook rip_with_many_lines_playbook.yml --syntax-check
playbook: rip_with_many_lines_playbook.yml
root@NetworkAutomation-2:~# [ ]
```

في حال تم العثور على خطأ ما في ال Syntax ، سيتم إخبارنا عنه وعن موقعه في الملف.

كما يمكننا الاستفادة من الموقع: [Yamllint.com](https://yamllint.com) في التحقق من صحة ال Syntax الخاص بأي ملف من صيغة YAML.

من أهم الخيارات المستخدمة مع تعليمية تنفيذ ال Playbook :

### **1) -check:**

للحصول على إمكانية تطبيق ال Tasks المحددة في ال Playbook على ال host المختار (توافق ال Modules المستخدمة ضمن ال Playbook مع ال host ) ، نستخدم الخيار **-check** مع تعليمية **ansible-playbook** حيث سيتم فقط اختبار تنفيذ ال Tasks بدون تطبيقها بشكل "فعلي"

التالي:

# Verifying Playbooks



```
root@ubuntu:/# ansible-playbook test.yml --check
PLAY [Get CPU information]
*****
TASK [show CPU levels]
*****
skipping: [10.10.10.1]
skipping: [10.10.10.2]
TASK [debug] *****
ok: [10.10.10.1] => { "print_output.stdout_lines": "VARIABLE IS NOT DEFINED!"}
ok: [10.10.10.2] => { "print_output.stdout_lines": "VARIABLE IS NOT DEFINED!" }
PLAY RECAP *****
10.10.10.1 : ok=1    changed=0   unreachable=0  failed=0  skipped=1  rescued=0   ignored=0
10.10.10.2 : ok=1    changed=0   unreachable=0  failed=0  skipped=1  rescued=0   ignored=0
```

## :**-dif** الخيار 2

من أهم الخيارات التي يمكن استخدامها مع تعليمية تنفيذ ملفات ال Playbook هو الخيار **-dif** الذي يتيح التخلص من مشكلة Configuration Drifting التي تنتج عن تطبيق تعليمات مطبقة مسبقا، أو تطبيق تعليمات بشكل متسرّع يسبب الخروج عن المسار المحدد لوظيفة التجهيزه الشبكيه، فعند استخدام هذا الخيار سيتم عرض الفروقات التي ستنتج عن تنفيذ ال Tasks الموجودة في ال Playbook المختار:

# Verifying Playbooks



```
root@ubuntu:/# ansible-playbook test.yml --diff // ansible-playbook test.yml --diff --check
PLAY [Get CPU information]
*****
TASK [Interface Configuration]
*****
-- before
+++ after
@@ -131,7 +131,7 @@
interface Ethernet1/10
- description test1
+ description test2
ok: [10.10.10.1] => { "print_output.stdout_lines": "VARIABLE IS NOT DEFINED!"}
PLAY RECAP *****
10.10.10.1 : ok=1  changed=1  unreachable=0  failed=0  skipped=0  rescued=0  ignored=0
```



[الخبار :list-tasks](#) (3)

الذي يقوم بعرض جميع ال Tasks الموجودة ضمن أي ملف `.playbook`.

## References:

1. Odom, W., *CCNA 200-301 Official Cert Guide, Volume 2*. 2019; Cisco Press.
2. Edgeworth, B., et al., CCNP and CCIE Enterprise Core ENCOR 350-401 Official Cert Guide. 2019; Cisco Press.
3. WENDELL, O., *CCNA 200-301 Official Cert Guide, Volume 2, San Jose*. 2020, CA: Cisco Press.
4. Apostolopoulos., J. *Software defined networking (SDN)* June 1, 2018; Available from: <https://blogs.cisco.com/analytics-automation/why-is-intent-based-networking-good-news-for-software-defined-networking>.
5. Mobley., J. *Welcome to the 2022 Global Networking Trends Report: The Rise of Network as a Service (Naas)*. 2022; Available from: [https://www.cisco.com/c/en\\_uk/solutions/enterprise-networks/2022-networking-report-preview.html](https://www.cisco.com/c/en_uk/solutions/enterprise-networks/2022-networking-report-preview.html).
6. CISCO. *Cisco DNA Center At-a-Glance*. 2022; Available from: <https://www.cisco.com/c/en/us/products/collateral/cloud-systems-management/dna-center/nb-06-cisco-dna-center-aag-cte-en.html?oid=aagen000249>.
7. CNNA. *Configuration Management Tools – Ansible, Chef, Puppet*. Available from: <https://study-ccna.com/configuration-management-tools-ansible-chef-puppet/>.

8. Lessons., N. *Introduction to SDN (Software Defined Networking)*. 2016; Available from: [https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-software-defined-networking#REST\\_API](https://networklessons.com/cisco/ccna-routing-switching-icnd2-200-105/introduction-to-sdn-software-defined-networking#REST_API).
9. CISCO. *Cisco SD-WAN*. 2019; Available from: [https://www.cisco.com/c/en\\_uk/solutions/enterprise-networks/sd-wan/index.html#~products](https://www.cisco.com/c/en_uk/solutions/enterprise-networks/sd-wan/index.html#~products).
10. Ansible. *Ansible Template Tester*. 2022; Available from: <https://ansible.sivel.net/test/#>.
11. yamlchecker. *What is YAML*. Available from: <https://yamlchecker.com/>.
12. Documentation., A. 2020; Available from: <https://docs.ansible.com/>.