Network Programmability Fundamentals

S/w Development and Information technology operations (DevOps):

A collection of procedures where s/w developers work with IT operations professionals to automate n/w changes and s/w development.

SDN (s/w defined n/w):

A collection of tools that can be used for DevOps, allowing an entire n/w to be managed as one entity, rather than configuring individual devices.

Data plane: in charge of forwarding packets in the frames to appropriate interface(ingress, egress, MAC address)

Control plane: Tacking care of protocols. How CAM and routing protocols getting populated. Routing protocols, like RIP,OSPF,EIGRP

Management plane: In charge of how we administratively attach in to device. we can do telnet or ssh

Managing n/w as entity rather than maintaining individual devices.

Distributed Control Plane: A n/w architecture where control plane functions (e.g routing protocols) reside in the n/w devices.

SDN Controller: Take control plane from devices and consolidates them in to SDN controller. It can keep track of everything happening in the n/w. and push changes to individual devices.

This type of communication goes on b/w SDN controller and one of end devices is called API.

Application Programming Interface (API):

Allows an application on one device to communicate application with another device.

South Bound Interface (SBI):

An API used for Communication Between an SDN controller and a N/w device(e.g a router , a switch).

Centralized Control Plane:

An/w architecture where control plane functions reside in a centralized controller.

e.g : open day light is a protocol.(open flow)

North Bound Interface (NBI) :

An API used for communication b/w an SDN controller and an application.

Here we are using REST API also called Restful API.

REST API: (Representational state Transfer (REST) API

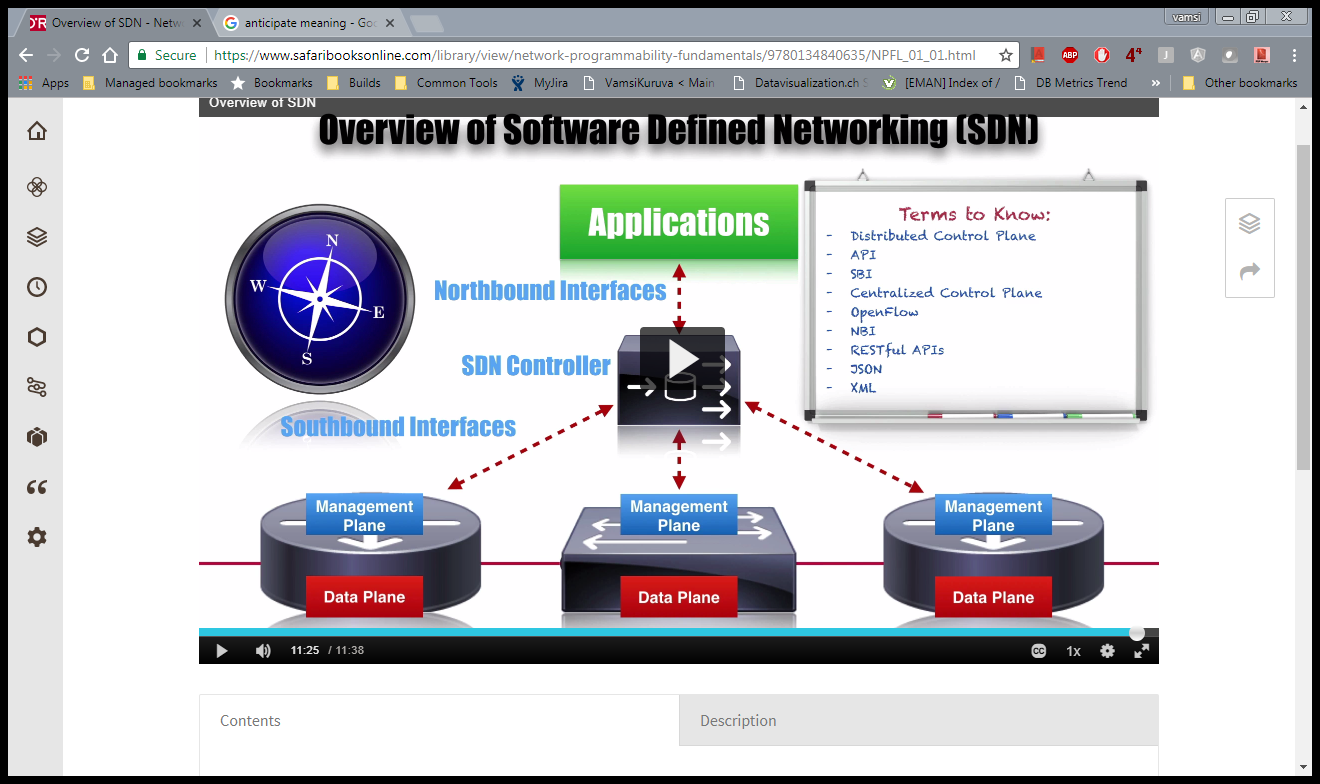
An API, commonly written as RESTful API, that uses HTTp msgs(e.g GET or POST) to send information from an application to SDN controller.

🡪Application could be Python program.

🡪Data formats that could be used by RESTful APIs.

JavaScript Object notation (JSON)

Extensible Markup Language (XML)

```

Comparing the Cisco APIC and APIC-EM

CISCO APIC : (application policy infrastructure controller)

Control plane is in devices only

Stateless Controller:

An SDN controller where each n/w device has its own contol plane. The SDn controller sends policies to the n/w devices, which are individually responsible for translating policies into commands.

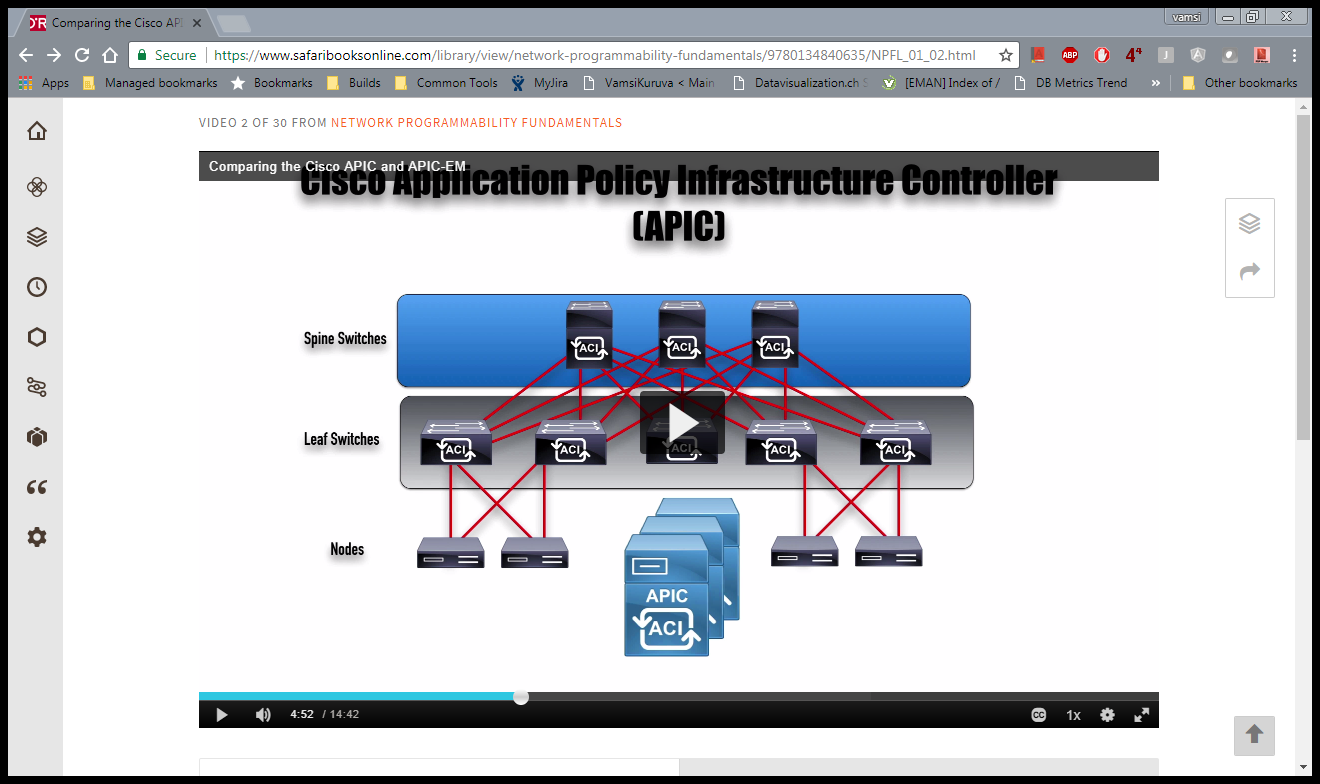
Typically see in data centers.

APIC-EM: APIC enterprise model

It can interact with treditional devices.

Also called restful APIs

Cisco Data Center:



APIC will communicate with Nexus switches by using ACI.

Cisco application centric Infrastructure (ACI):

Cisco’s approach to SDN in a data center environment, where application policy profiles can automatically configure infrastructure devices to support specific application needs.

🡪ACI allows multiple (multi-tendency environment) users to use same switch racks for their applications but, they are logically separated.

🡪we can also dynamically move that VM to another switch rack or data centers.

Specific characteristics of ACI:

For use in Data centers

Applications

Policy manager

Topology manager

Observer(monitoring)

Boot Director(firmware and booting diff Spain leaf switches)

Application Director (setting and controlling cluster)

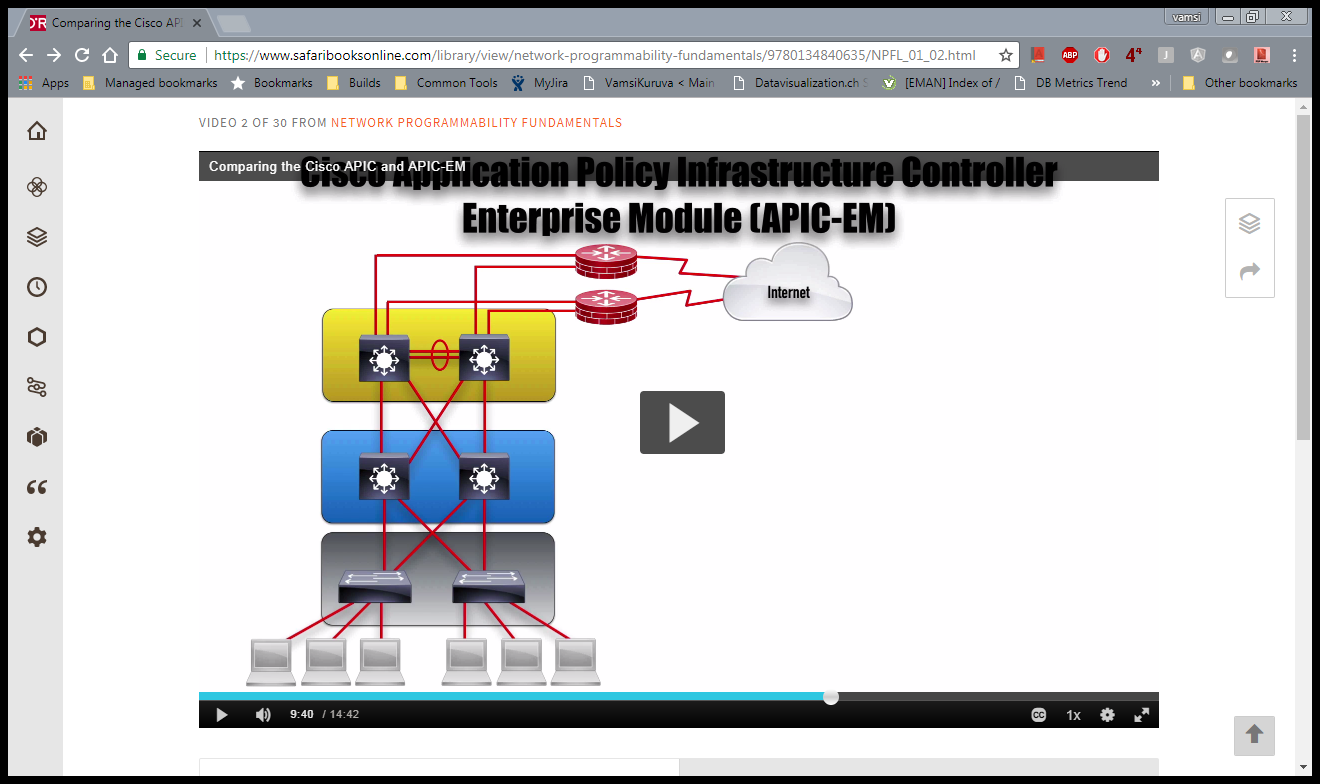
Virtual Machine Manager (VMM)

Event manager( store all events or faults coming from nodes or APICs)

Appliance Element

APIC dashboard:

CISCO APIC-EM



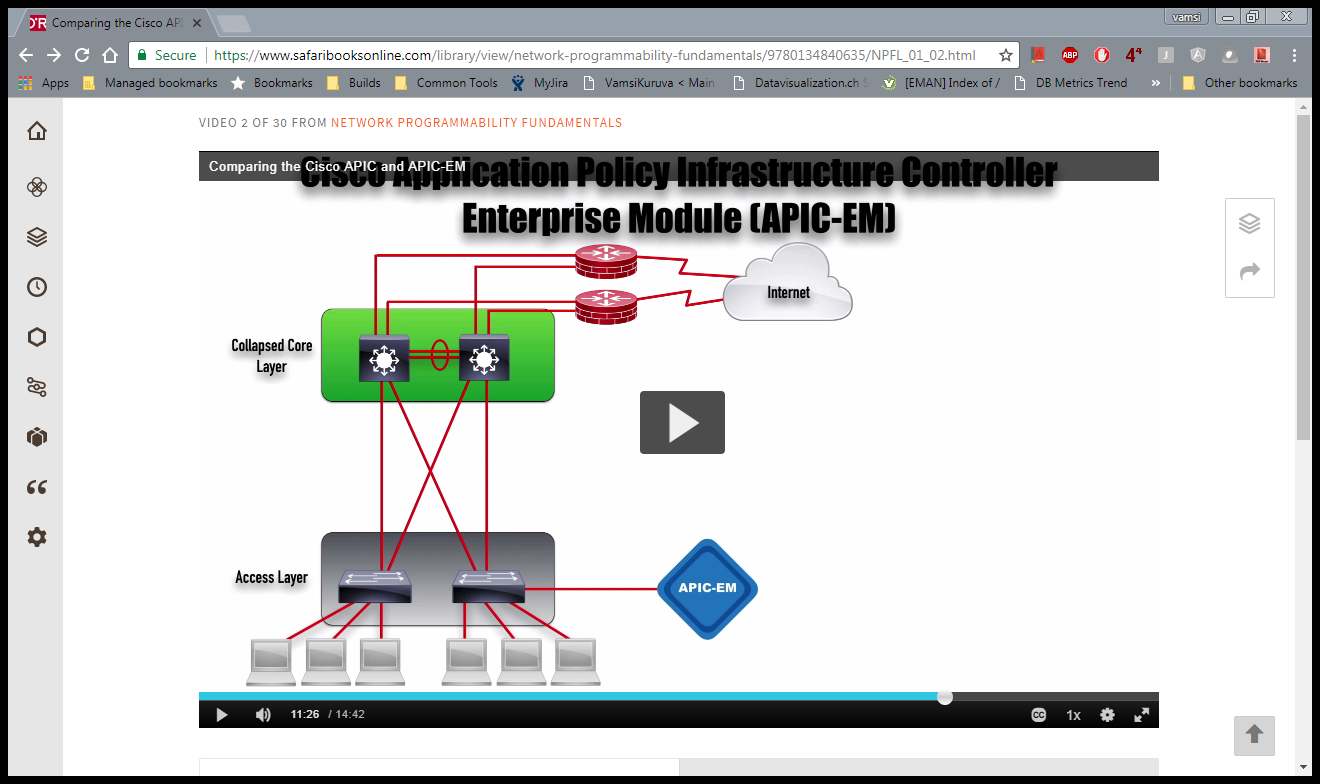
🡪Three-Tire Architecture

Access layer, Distribution layer, core layers

🡪More use in branch office or campus.

For branch offices and campuses sometimes we don’t need 3 tire architecture so, will consolidates distribution and core layers and term it as a collapsed core layer.

Collapsed core Layer Design:



Here we can use APIC-EM to connect with traditional devices.

It can use telnet, ssh, SNMP to connect with devices.

APIC-EM characteristics

For use in campus, branch or WAN

Applications:

N/w topology visualization

Cisco Wan

Path Trace application

JSON(JAVAScript Object Notation) Format:

2 basic structures

A collection of name/value pairs (like dictionary in python)

An ordered list of values (like array list in python)

Object:

An unordered set of name/values pairs

Enclosed in curly brackets

Ex:{“firstname”:”uma”,”lastName”:”g”}

{

“firstname”:”uma”

“lastname”:”g”

}

Array

An ordered set of comma-separated values.

Enclosed in straight brackets

Ex: [“AA”,”BB”,”CC”]

Whitespace:

[

“AA”

“BB”

“CC”

]

Value:

Can be a string, number, object,array,null,true,false.

<http://jsonlint.com> : An online JSON format validator

XML Formatting ( extensible markup language):

Extensible: we can add to its capabilities by creating our own tags

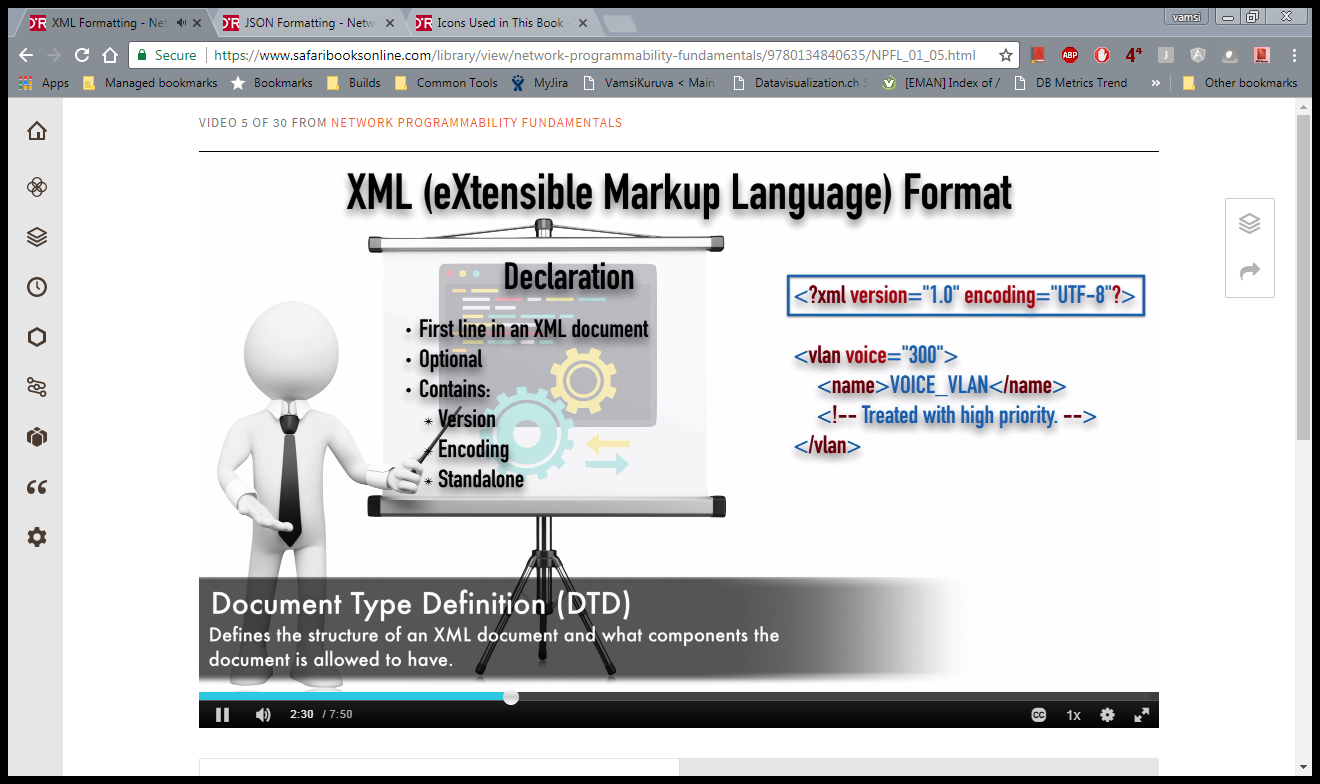
Markup: Similar to HTML, tags are used for formatting.

HTML Example

<h4> Header text </h4>

XML format:

Declaration



Default version is UTF-8

Components of XML

DTD

Tag : A string of text inside < and > signs.

Has both a start and end tag

Data contained between the start and end tags.

EX: <name> </name>

<vlan …

………….

</valn>

Elemnet :

Data located bw start and end tags.

Can have nested elements.

Ex: VOICE VLAN

Attribute:

Gives more detail about an element

Value of attribute must appear in quotes.

EX: voice=300 (in this case)

Comment:

Used to proide documentation within the file

Is not executed

Enclosed in a start tag of <!- -and and end tag of - ->

EX: <!- - Treated with high priority. - -> (in this case)

XML JSON

<?xml version =”1.0” encoding=”UTF-8”?> {

<vlan voice=”300”> “vlan:{

<name> VOICE\_VLAN </name> -voice”:”300”,

<!- - Treated with high priority.- - > “name”:“VOICE\_VLAN”

</vlan> }

YANG Data Modeling:

Commonly used with NETCONF.

Data modeling

Developed by the IETF

RFCs 6020 and 6021

For YangModel examples : <https://github.com/YangModels/yang>

Example: like

Iphone 7 plus,5.5” Display Size, Black, 256 GB

Yang Data Model of a Network Interface:

