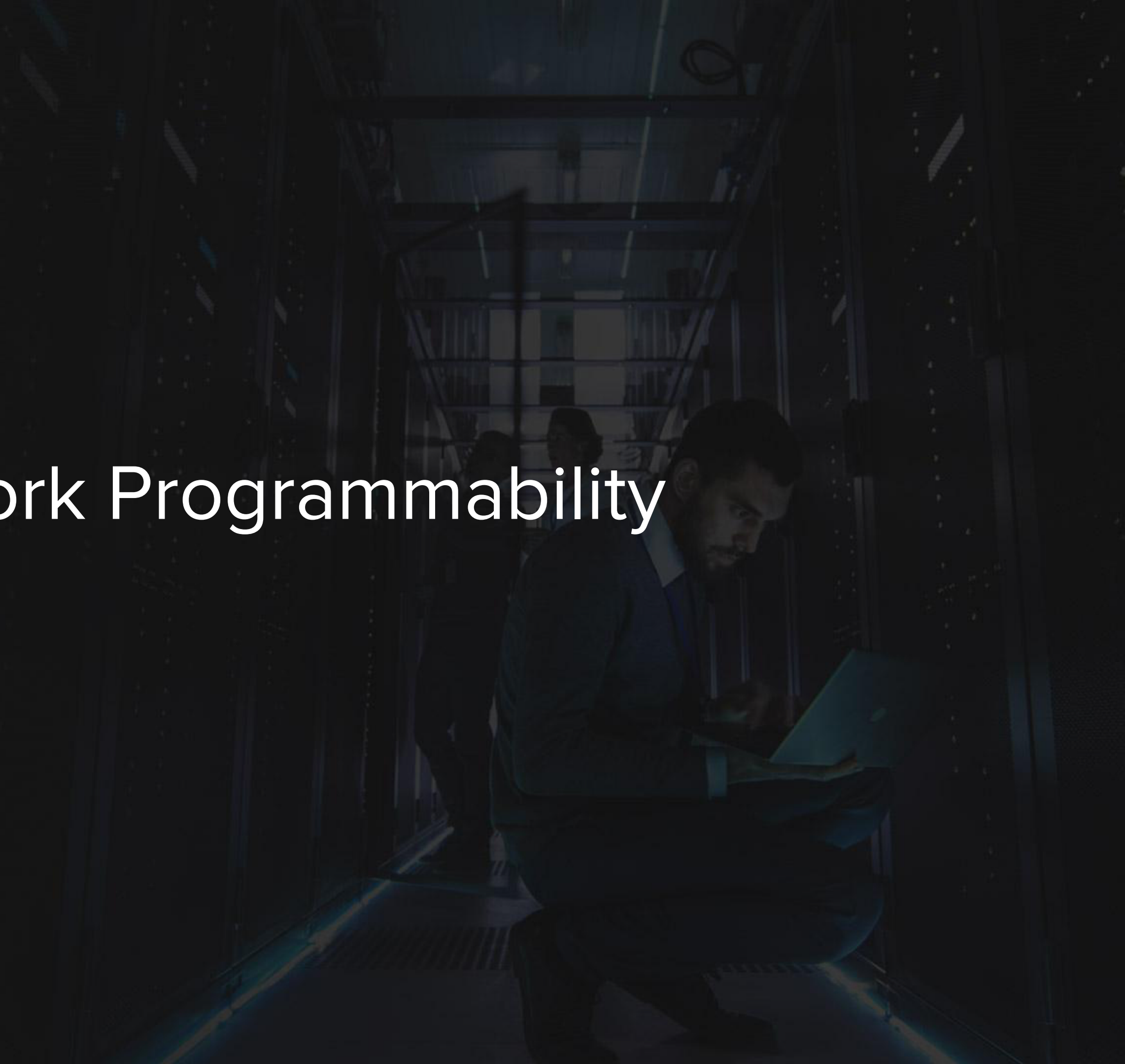
Introducing Network Programmability And Automation



ine.com

Keith Bogart

CCIE #4923

kbogart@ine.com

@keithbogart1

linkedin.com/in/keith-bogart-2a75042

CCIE Routing & Switching

Course Prerequisites

- Basic understanding of the roles of network infrastructure equipment
- Experience with configuring network equipment via a command-line.
- High-level understanding of the usage of scripts
- Understanding of basic IP packet routing concepts.

Course Objectives

- To give you exposure to various elements of network automation and programmability including:
  - Network automation tools like Chef, Ansible and Puppet
  - The function of use of APIs
  - The role of SDN controllers
  - The Imperative vs Declarative models
  - Concepts of Underlay and Overlay Networks
  - Introduction to Cisco DNA Center
  - Interpreting JSON encoded data

INE

Historical-To-Current Methods Of Network Management

Topic Overview

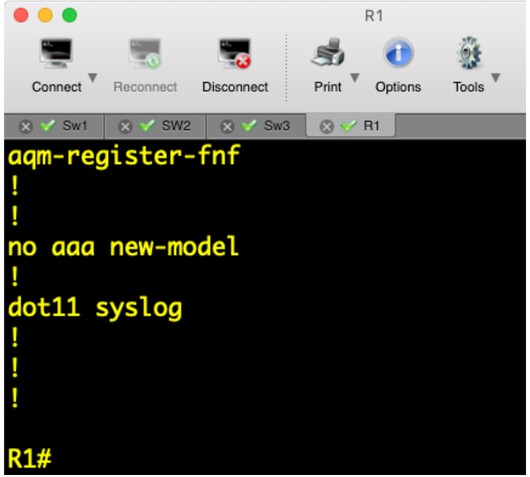
+ Definitions Of Network Management
+ Historical Methods
+ Past & Current Challenges

What Encompasses “Network Management”?

- Physical installation of new equipment
- Initial configuration of equipment (i.e. “provisioning”)
- Monitoring/Testing
- Software upgrades and patches
- Configuration tuning and enhancements

Past & Present Methods Of Management

- Configuration/Troubleshooting/Software upgrades
  - SSH/Telnet/Console
  - Limited SNMP
  - Scripts
  - Notepad
- Network Monitoring
  - SNMP
  - Netflow



Challenges With Traditional Methods

- Large Network = Large IT staff
- Frequently Requires Knowledge of multiple network Operating Systems
  - Silos of expertise
  - Huge learning curves
- Knowledge of SNMP configuration and management
- Box-by-box management
- Notepad...the engineer's favorite tool
  - Easy to make mistakes or lose documents

Thanks for Watching!

Introduction To Network Management Automation

ine.com

Topic Overview

- Goals Of Automation
- What Can Be Automated?

Goals Of Network Automation

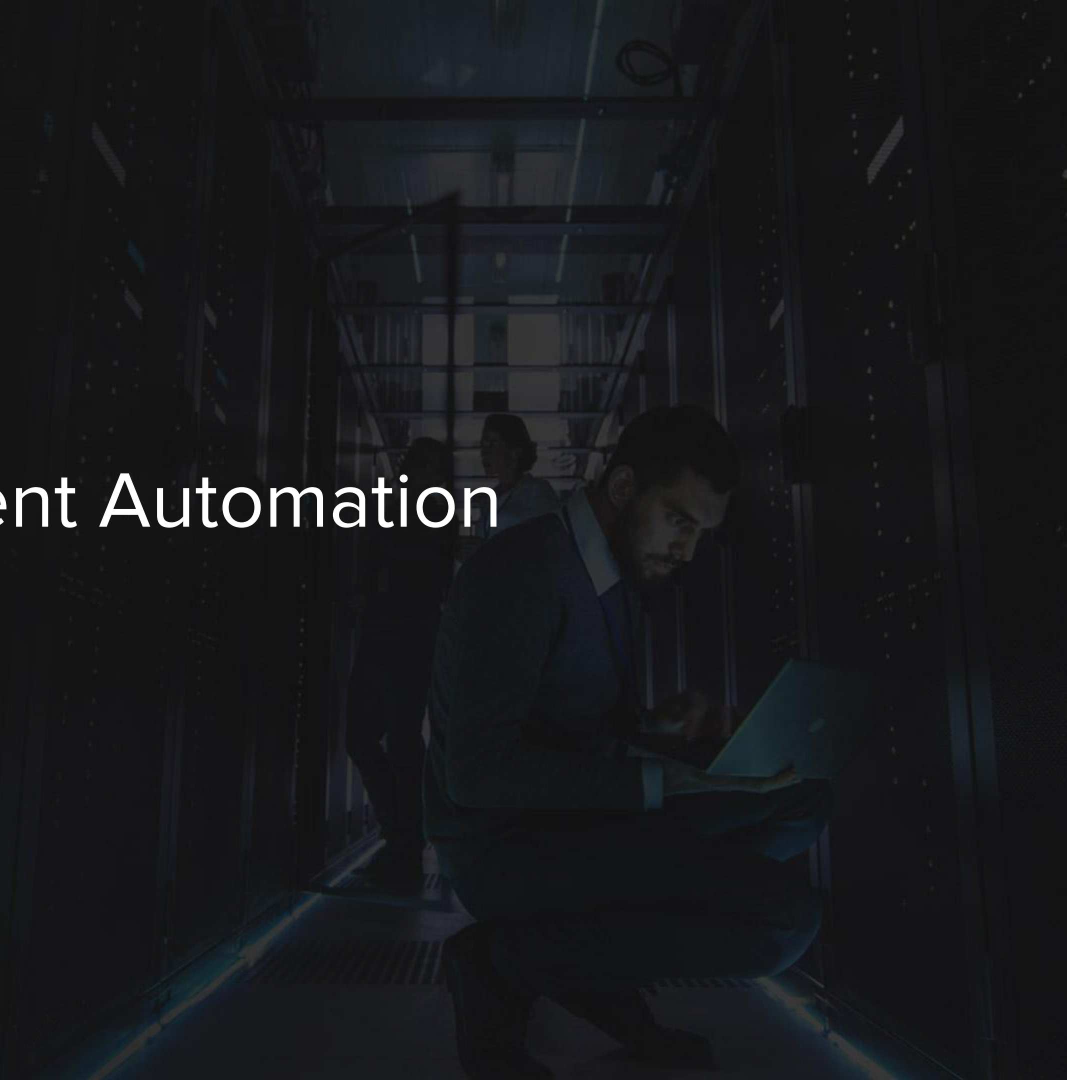
- Reduce box-by-box management model
- Eliminate repetitive tasks
- Standardize software types and procedures
  - Identify "Golden Images"
  - Standard upgrade procedures
- Utilize scripts and tools to perform mass upgrades/changes
- Apply consistent policy across the network
- Reduce time spent troubleshooting

What Can Be Automated?

- Plug-and-play initial provisioning
- Path segregation via dynamic overlay networks
- Automated and dynamic QoS policies
- Dynamic security policies
- Scheduled software deployments
- Topology visualizations
- Intelligent and automated solutions to troubleshooting problems

Thanks for Watching!

Network Management Automation Origination Points



ine.com

Topic Overview

+ Automation Origination Points

Automation Origination Points

+ Network Management Automation can happen from three different origins:
    + SDN Controllers
    + Servers running Network Management protocols
    + On-the-box automation using built-in scripts

Ansible Tower

TCL Script

```
R1#
R1#tclsh
R1(tcl)#foreach ip {
+>(tcl)#192.168.1.2
+>(tcl)#192.168.1.3
+>(tcl)#} { puts [ exec "ping $ip"]
+>(tcl)#}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/2/4 ms

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/4 ms

R1(tcl)#
```

SDN Controllers

+ SDN = Software Defined Networking
+ A “Controller” is the integral part of SDN
+ Two form factors:
  + Software pre-installed on a physical chassis (i.e. “Appliance”)
  + Software installed on your own server (or in the Cloud)
+ Examples of Cisco SDN Controllers;
  + Cisco ACI/APIC
  + Cisco APIC-EM

CISCO APIC Appliance

Network Automation Software Tools

- Network Configuration Tools that can assist SDN Controllers
  - Ansible
  - Chef
  - Puppet
  - Others

Ansible Tower

Scripting For Network Automation

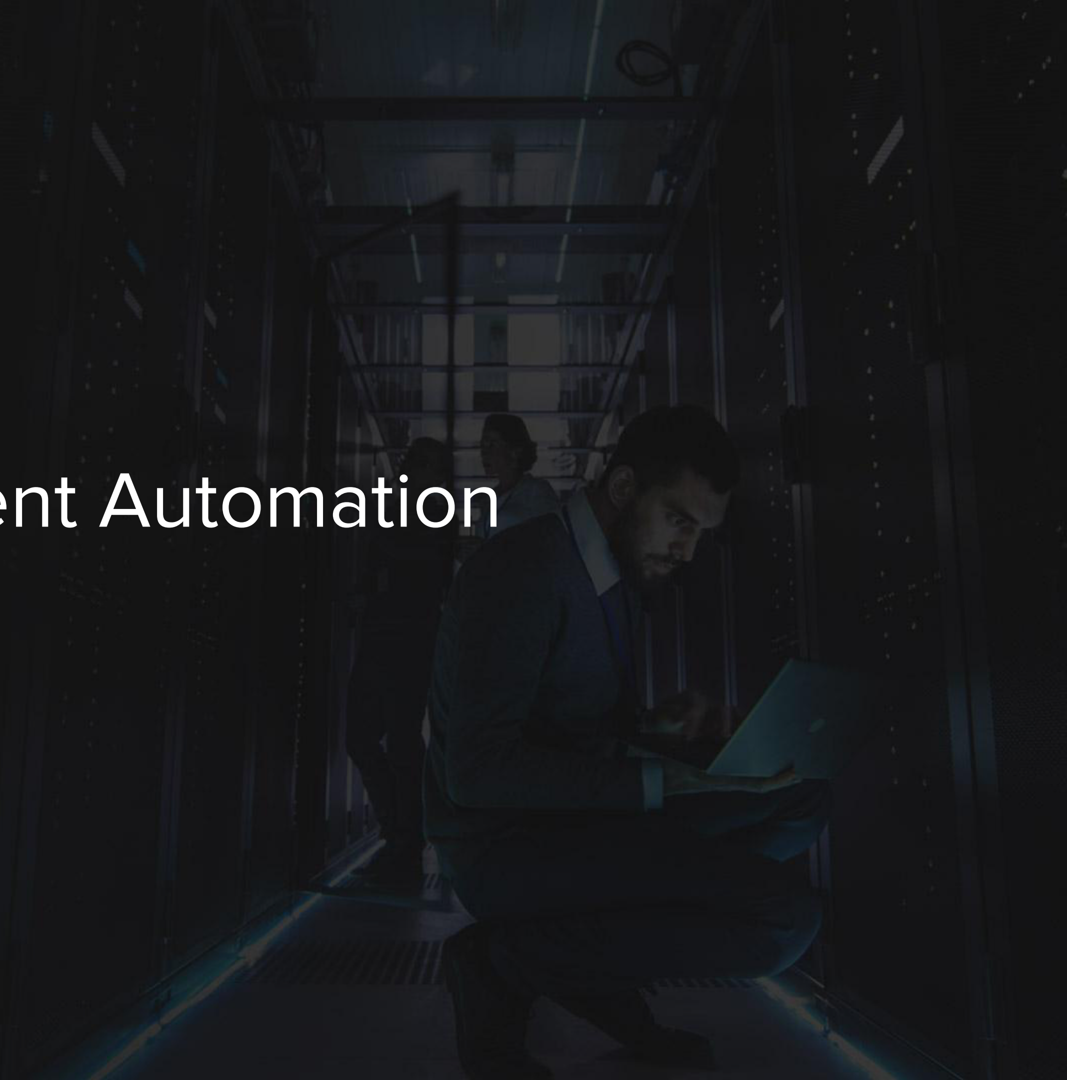
- Many available scripting languages
- Origination points for scripts
  - Script initiated on remote device and commands sent over an IP connection.
  - Script built-into device software and invoked on-the-box
- Popular examples;
  - TCL & Python scripts

TCL Script

```
R1#
R1#tclsh
R1(tcl)#foreach ip {
+>(tcl)#192.168.1.2
+>(tcl)#192.168.1.3
+>(tcl)#} { puts [ exec "ping $ip"]
+>(tcl)#}

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.2, timeout is 2 seconds:
..!!!
Success rate is 60 percent (3/5), round-trip min/avg/max = 1/2/4 ms

Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.3, timeout is 2 seconds:
.!!!!
Success rate is 80 percent (4/5), round-trip min/avg/max = 1/1/4 ms

R1(tcl)#
```

Thanks for Watching!

Network Management Automation Protocols & Impact



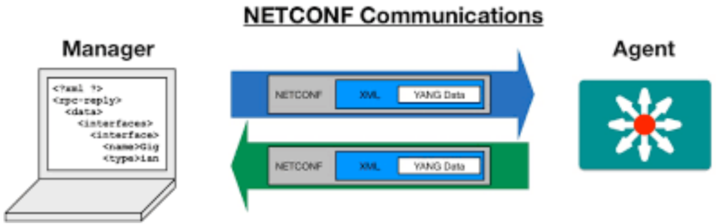
ine.com

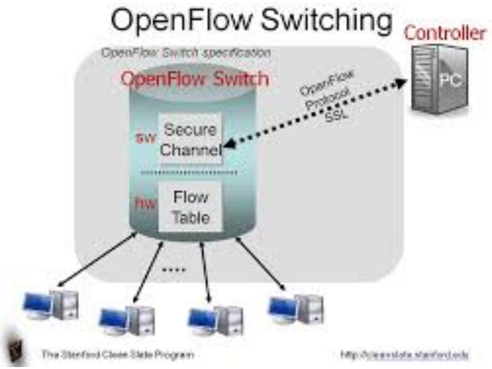
Topic Overview

- Common Languages & Protocols
- Impact Of Automation On Network Management

Common Languages For Network Automation

- Network Monitoring
  - SNMP Managers
  - Netflow Collectors
- Common languages and protocols;
  - CLI carried over SSH
  - SNMP
  - NETCONF/YANG
  - RESTCONF/YANG
  - OpenFlow
  - Cisco OpFlex
  - REST APIs





The Impact Of Network Automation

- How is network management impacted by automation?
  - Cost reduction
  - Time savings and elimination of repetition
  - Configuration consistency
  - Elastic scaling

- Network Admins will need to become familiar with Server OS, installation, patching and troubleshooting.

Thanks for Watching!

Comparing Traditional Networks With Controller-Based Networking

ine.com

Topic Overview

- Management Of Traditional Networks
- Controller-Based Networks
- Imperative & Declarative Approaches
- The Design Impact Of SDN

Managing Traditional Networks

- Box-by-box management
- CLI-driven
- Extensive use of Telnet/SSH/HTTP or SNMP
- Networking functions implemented in individual devices using vendor-proprietary ASICs
- Devices start with minimal (or no) initial configs.
  - Complexity and usefulness added via complex CLI commands or box-by-box GUI implementation.
- Multiple, disparate servers for network management

Networks Managed By SDN Controllers

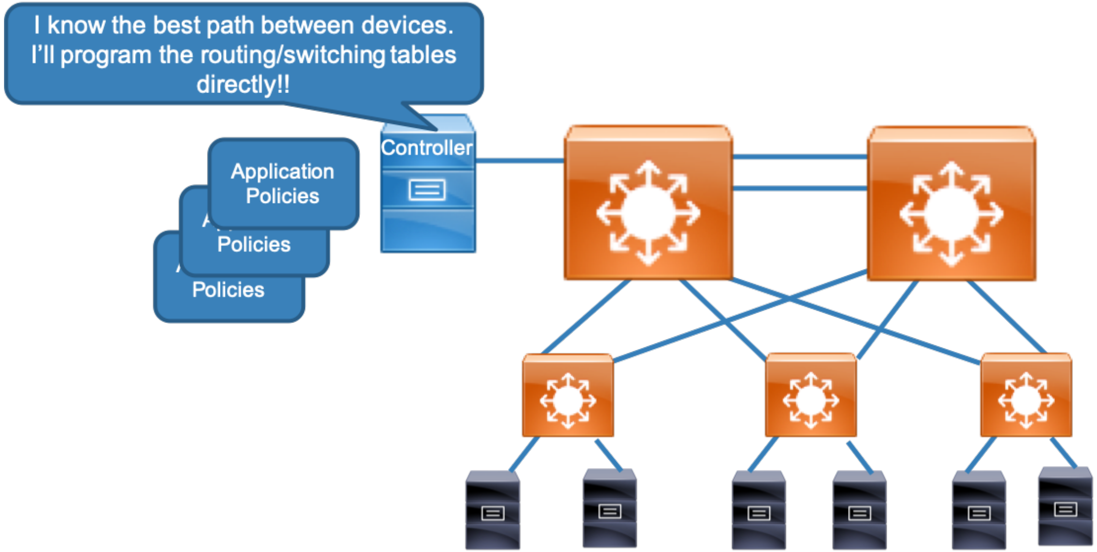
- Dynamic implementation of initial configurations
  - Plug-and-play
  - Zero-touch provisioning
- Dynamic and automatic updates/changes to configurations based on pre-configured policies
- Relocation of Control Plane functionality to a central SDN Controller
- Controllers can consolidate multiple management services into one box.

Imperative & Declarative Approaches

- Two approaches for Controller implementation;
  - Imperative approach
  - Declarative approach
- Let’s look at each of these..

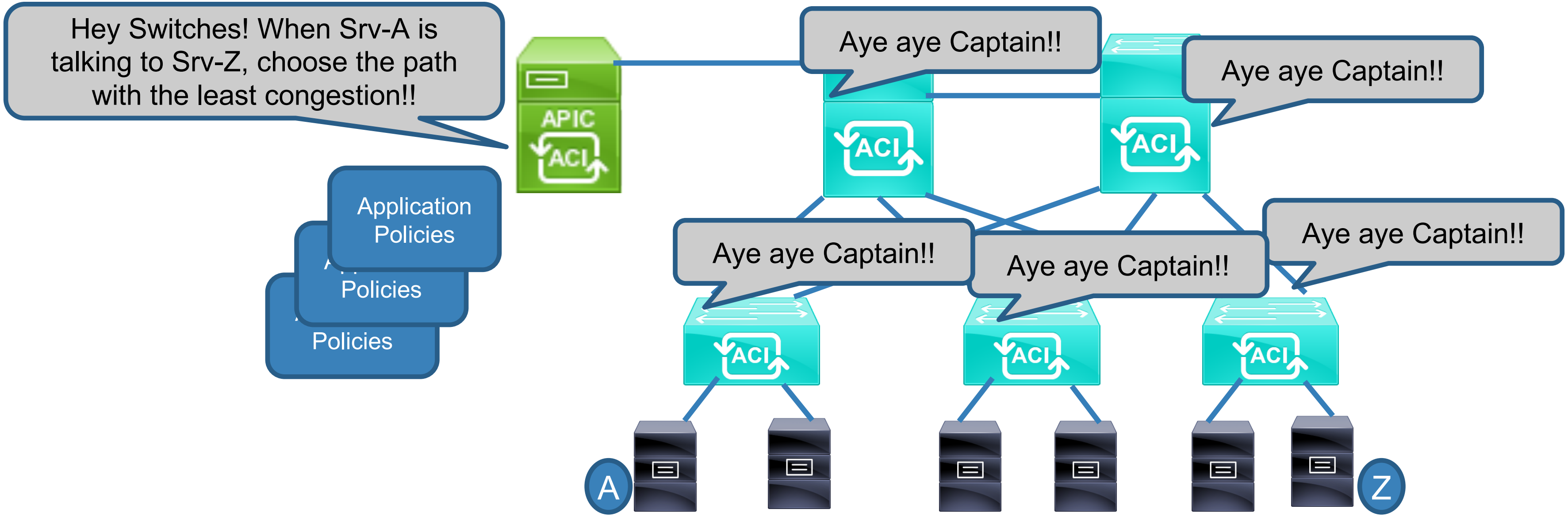
Imperative Approach To Controllers

- Control Plane logic resides in Controller
- Controller has complete control over programing the forwarding path of devices



Declarative Approach To Controllers

- Control Plane resides within networking devices
- Controller “declares” the requirements of applications
- Network devices decide how to translate that into functional actions.



SDN Controller’s Impact On Network Design

- Hardware selected must understand the Controller’s protocols
- Redundant paths to/from the Controller should exist
- If datacenters are geographically dispersed, one must plan for Controller reachability.
- Controllers should be configured in clusters for redundancy
- Security is critical!
- Is training available for I.T. staff on new protocols and software?

Thanks for Watching!

Controller-Based SDN Architectures

Topic Overview

- What Is An Underlay Network?
- What Is An Overlay Network?
- What Is The SDN Fabric?

Architectural Concepts

- Underlay Network
- Overlay Network
- SDN Fabric

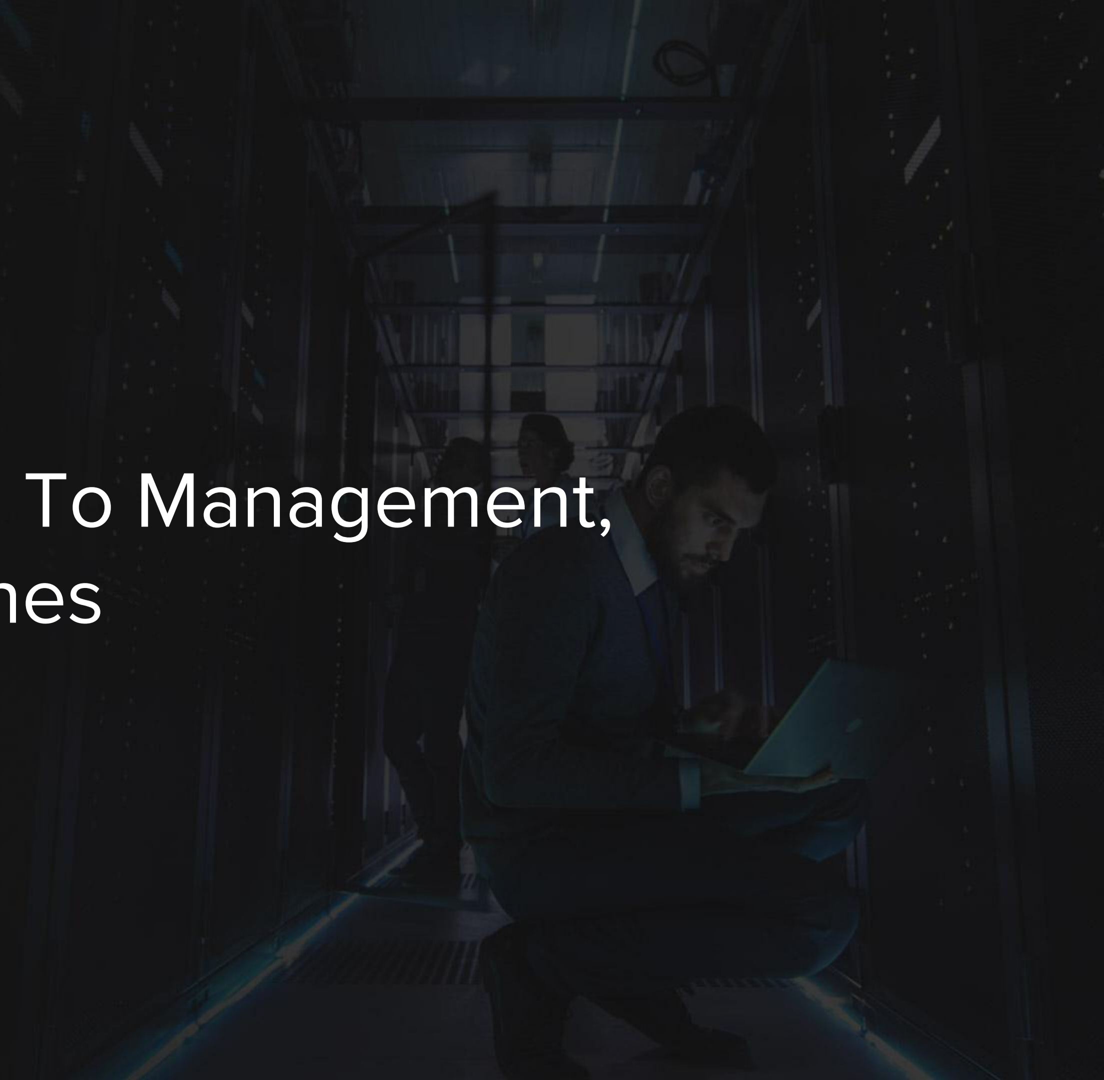
Underlay & Overlay

- Underlay Network
  - Protocols and features used to establish full IP reachability between endpoints
  - All links typically configured as Layer-3, point-to-point
  - Common, industry-standard Routing Protocols used (OSPF or IS-IS)
  - Network Engineers have been building/maintaining underlay networks for years…we just haven't called them that.
- Overlay Network
  - Virtual networks that are created by software and implemented by the Underlay Network.
  - Practically implemented via VRFs, MPLS-VPNs, VxLAN or other technologies.

The Fabric

- Physical infrastructure used to build the Underlay Network (actual switches, routers, cables, and internal switching paths)
- Typically used to describe ONLY those devices (in the Underlay Network) that can be programmed/controlled by the SDN controller
- A full-mesh of devices with multiple, equal-cost paths between destinations

Thanks for Watching!



SDN’s Relationship To Management, Control & Data Planes

ine.com

Topic Overview

+ Management Plane Defined
+ How SDN Affects Management Plane
+ Data Plane Defined
+ Population Of The Data Plane
+ Control Plane Defined
+ How SDN Affects Control & Data Planes

Management, Control & Data Planes

- All of the things a network device can do, can be categorized as residing in one-of-three logical places:
  - Management Plane
  - Control Plane
  - Data Plane / Forwarding Plane
- Let’s talk about all three and how they relate to Software Defined Networking…

The Management Plane

- The Management Plane is responsible for giving you access to control the device (configure, monitor, troubleshoot).
- Any feature or protocol that exists to give you this ability, resides in the Management Plane.
  - Console access
  - Telnet
  - SSH
  - HTTP/HTTPS
  - SNMP

The Management Plane & SDN

- How is the Management Plane affected by SDN?
- Many SDN Controllers rely on existing Management Plane mechanisms (such as SSH or HTTPS)
- Some new mechanisms have been developed for new types of access:
  - NETCONF/YANG
  - XML-based commands

The Data Plane

- The Data Plane is responsible for transporting data through a network
  - Also called the “Forwarding Plane”
- Any entity (logical or physical) that exists to give a device this ability, resides in the Data Plane
  - MAC Address Tables
  - Routing Tables (“Forwarding Tables”)
  - Cables
  - NICs
  - Packet buffers and queues

Population Of The Data Plane

- Typically, memory structures that house Data Plane information, start out empty
  - MAC tables have no MAC addresses
  - Routing tables have no routes
- The Data Plane is not responsible for the learning of forwarding information…that is the job of the Control Plane
- Whatever the Control Plane learns (that is relevant to packet handling) is downloaded into Data Plane structures

The Control Plane

- The Control Plane is responsible for teaching the device HOW to forward (or otherwise act upon) traffic.
- Any feature or protocol that exists to provide this ability, resides in the Control Plane
  - Dynamic Routing Protocols
  - The process of dynamic MAC address learning
  - Interactions between a network device and a AAA server
  - DHCP transactions

How SDN Affects The Control & Data Planes

- Separation of the Control and Data Planes
- Imperative approach:
  - Also called "Stateful SDN"
  - All functions of the Control Plane centrally reside at SDN Controller
  - Controller can directly program the Data Plane of devices
- Declarative Approach:
  - Also called, "Stateless SDN"
  - Both Control and Data Planes reside within individual network devices
  - Controller "declares" how it wishes the network to function
  - Network devices translate that declaration into actions for programming their own, individual Data Plane constructs

Thanks for Watching!

Introducing Northbound & Southbound APIs

ine.com

Topic Overview

+ What Is An API?
+ Southbound APIs
+ Northbound APIs

What Is An API?

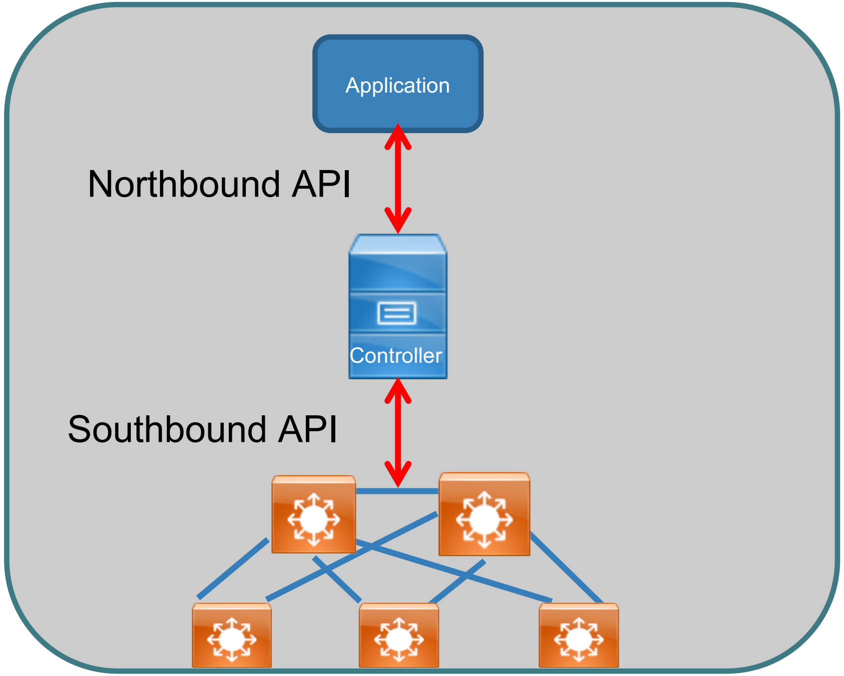
- Application Programming Interface
- A piece of code to allow different applications to talk to each other.
- Two generic types of APIs
  - Those that allow internal applications in your local system to exchange data.
  - Those that use IP networking to exchange data between remote applications.

API Usage In SDNs

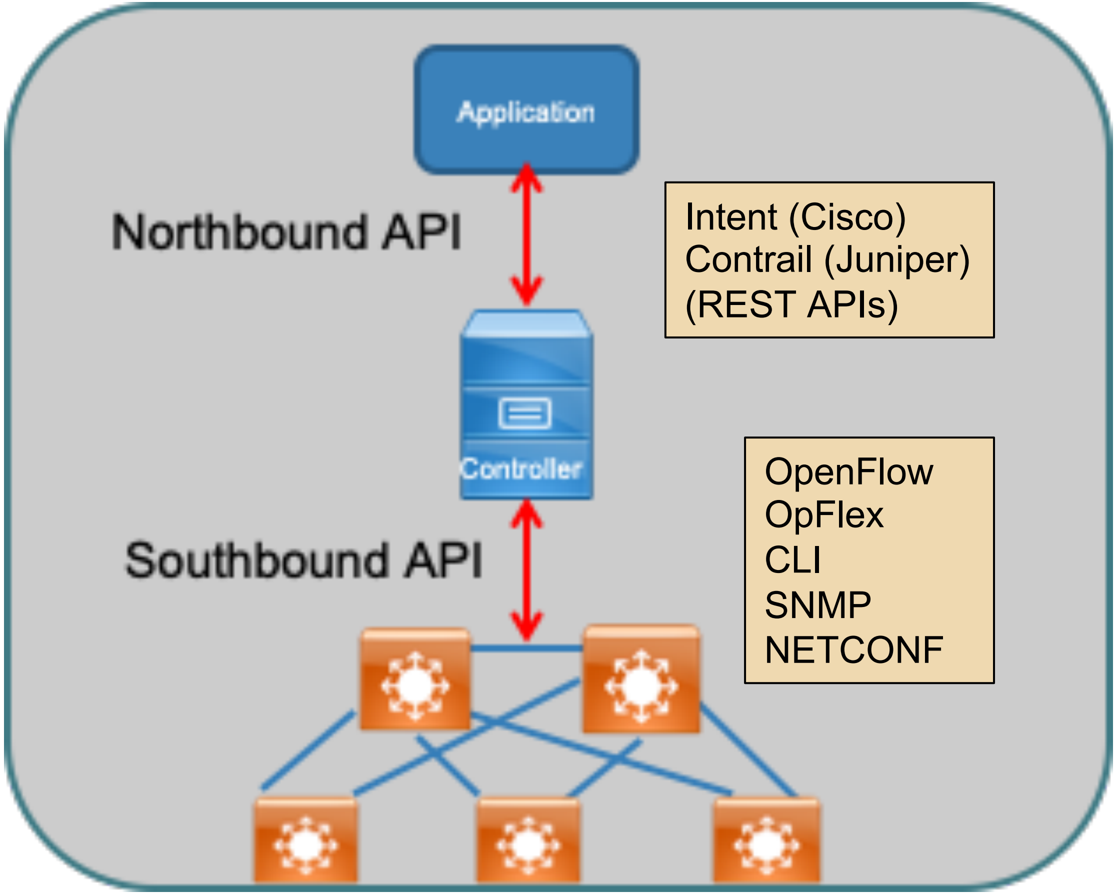
- Two primary uses of APIs in the world of SDN;
  - Applications connecting to Controllers
  - Controllers connecting to network devices
- Both of these types of connections utilize the Internet Protocol (IP), frequently over HTTP
- APIs use a Client/Server model
  - Application (API Client) communicates with SDN Controller (API Server)
  - SDN Controller (API Client) communicates with Switch/Router (API Server)

Northbound & Southbound APIs

- With reference to SDN, APIs are considered either Northbound or Southbound.
  - This is all in relationship to the position of the Controller in the topology.



Examples Of Common North/South APIs



Thanks for Watching!

Introduction To Cisco DNA Center

Topic Overview

+ The Problems Defined
+ Introduction To Cisco DNA Center
+ DNA Center Components

The Problems Defined

- Applying consistent configurations to newly-provisioned devices.
- Applying consistent security policies to users and devices
- Segmenting the network as needed…dynamically
- Applying QoS policies to enhance QoE
- Ensuring all devices are running consistent software images
- Providing useful data analytics
- Providing all of this from a “single pane of glass”

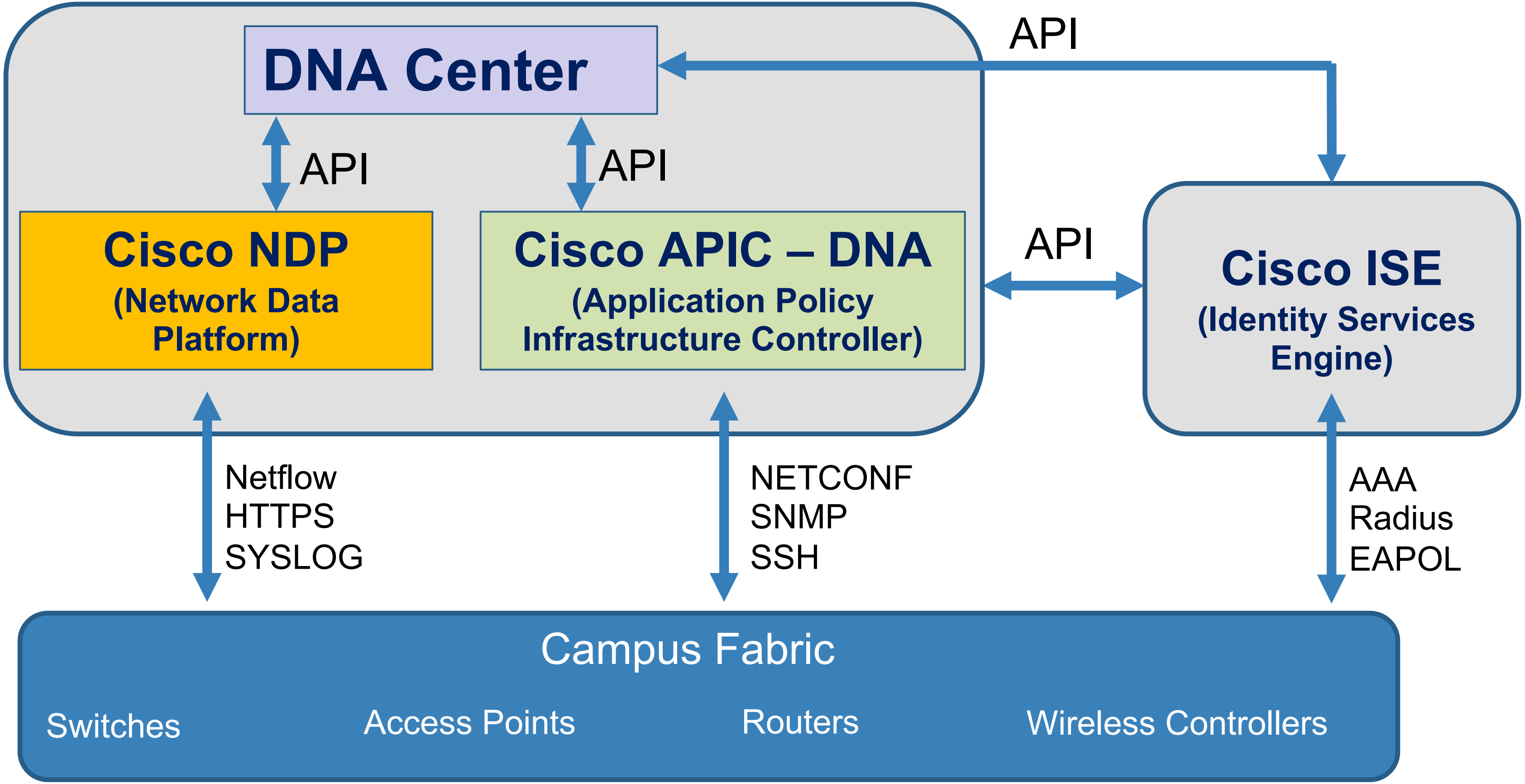
Introducing Cisco DNA Center

- DNA = Digital Network Architecture
- Cisco DNA Center is - a centralized management dashboard for complete control of a network
- Provides a central automation and analytics platform to facilitate “Intent-Based Networking”

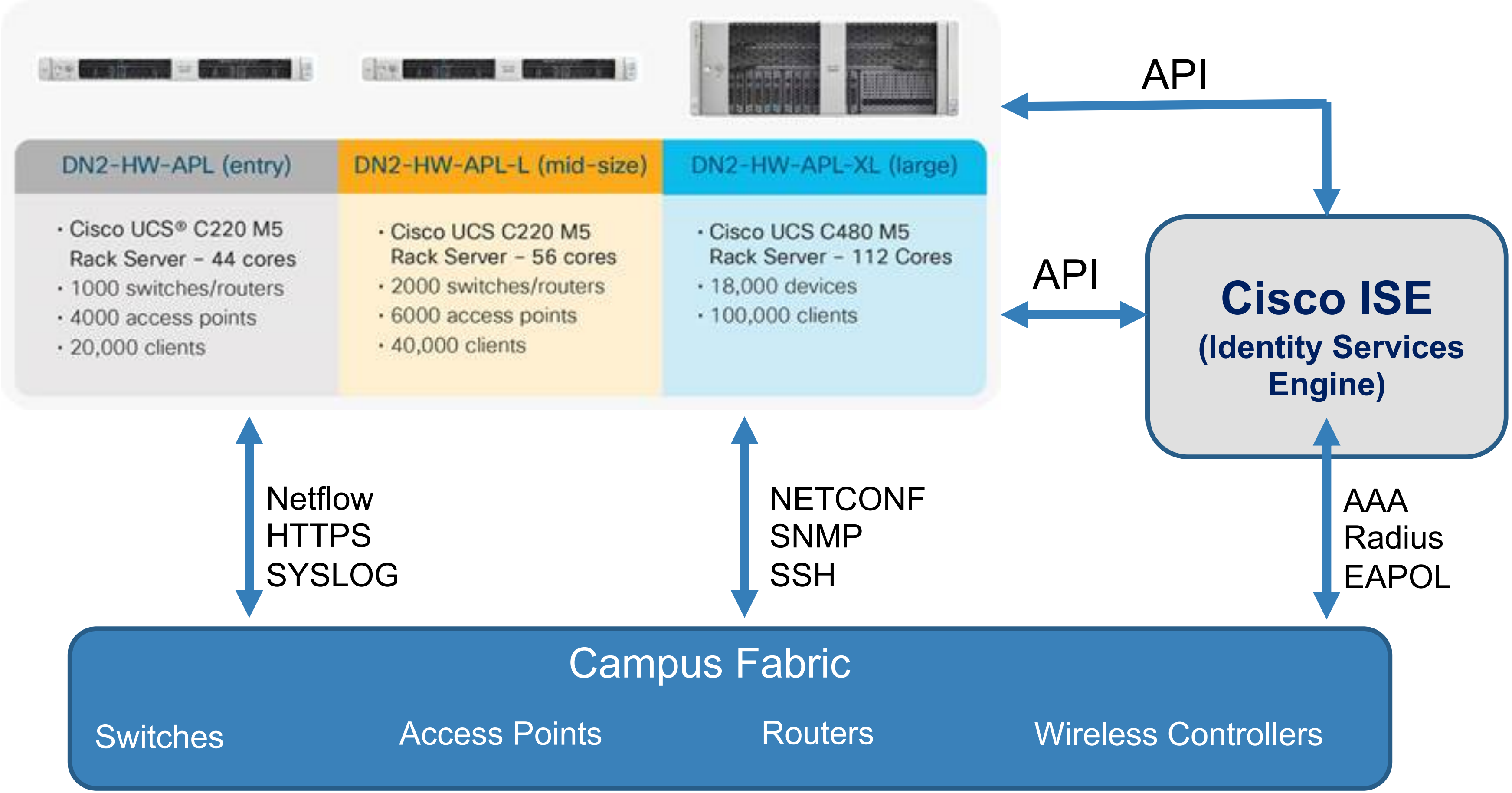
Cisco DNA Center

- Appliance pre-built with Cisco DNA Center software.
- A controller and analytics platform
- Central point of GUI-based network control allowing:
  - Design your network
  - Create topology maps and diagrams
  - Identify/list "Golden Images" for software deployments
  - Create wireless profiles and SSIDs
  - GUI-based configuration of network devices.

Cisco DNA Center Components



Cisco DNA Center Components



Thanks for Watching!

Architectural Elements Of Intent-Based Networking

ine.com

Topic Overview

- Architectural Elements Of Intent-Based Networking

Intent-Based Networking: Architectural Elements

- In order to get as close to intent-based networking as possible, the following architectural elements must be in place:
  - Instrumentation
  - Distributed on-device analytics
  - Telemetry
  - Scalable Storage
  - Analytics Engines
  - Machine Learning
  - Guided Troubleshooting & Remediation
  - Automated Troubleshooting & Remediation

Instrumentation

- Software and hardware elements designed to measure and collect data/statistics
- Cisco Catalyst 9000 switching family with the Cisco Unified Access™ Data Plane (UADP) 2.0 Application-Specific Integrated Circuit (ASIC).
- Client instrumentation
  - Apple and Cisco iOS analytics

- Sensors such as Aironet wireless sensors

On-Device Analytics

+ Data gathered during Instrumentation populates Key Performance Indicators (KPI)
+ On-device analytics enables prioritization of these KPIs without having to send ALL data gathered over the network to a central location.
+ A method of distributing analytics and processing

Telemetry

- Getting data off-the-box and over to the server.
- Legacy forms of telemetry include:
  - SNMP
  - Syslog
  - Netflow
- DNA Center approach for telemetry = model-based streaming telemetry
  - Data is “pushed” from the device at any time
  - Individual metrics can be streamed to Collectors

Scalable Storage

- Networks and hosts can generate well over 1TB of data per day.
- Scalable storage required
- Forms of scalable storage:
  - Centralized collectors
  - Distributed collectors
  - Cloud-based collectors

Analytics Engines

- Cisco DNA Center analytics engines.
- Provides automated context surrounding trouble issues such as:
  - Type of device experiencing the issue
  - IP and MAC addresses of the device
  - Security policy (via Cisco ISE) granted to the device
  - Network connection of the endpoint
  - Geographical detail and context

Machine Learning/Troubleshooting/Remediation

- Machine learning (ML) is the ability to "statistically learn" from data without explicit programming
- Traditional network management relies on silos of expertise among engineers
- Cisco DNA Center can utilize AI/ML to automate the baselining of network performance, identify problem areas and suggest remedies

Thanks for Watching!

Comparing Traditional Campus Networks Against DNA Center-Enabled Networks

ine.com

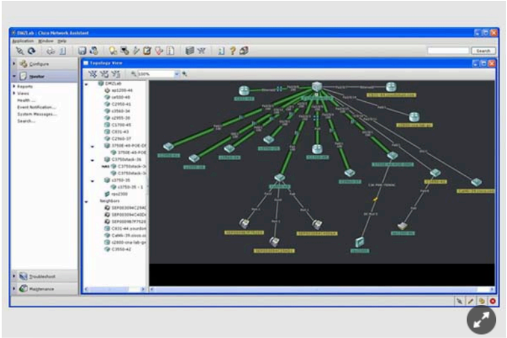
Topic Overview

- Traditional Device Management
- Similarities In Device Management
- Differences in Device Management

Traditional Device Management

- Devices managed via CLI
  - Box-by-box management
  - CLI-driven
  - Console/Telnet/SSH access
- Requires knowledge of hundreds of CLI commands
- Current (and legacy) GUI-based systems;
  - SNMP Management Stations
  - Cisco Works (Legacy)
  - Cisco Network Assistant
  - Many others

Cisco Network Assistant



Traditional vs DNA Center-Enabled: Similarities

- Both require that devices have configurations with full IP reachability
- Both typically require that devices have credentials configured;
  - SNMP
  - SSH/Telnet
- When using SNMP, both traditional campus and DNA Center-enabled topologies will have a central point (SNMP Manager) which collects and displays statistics.

Traditional vs DNA Center-Enabled: Differences

- Traditional campus
  - No topology visibility. Must create manually (PowerPoint, Visio, etc)
  - Box-by-box management when it comes to;
    - Updating configurations
    - Updating software
- DNA Center-enabled campus
  - Dynamic Topology Visualization
  - Path Trace and easy ACL analysis
  - Centralized management of Software updates and version control
  - Centralized control of initial configurations for plug-and-play/zero-touch devices
  - AI/ML to assist with identifying and resolving problem areas.

Thanks for Watching!

Characteristics Of REST-based APIs

Topic Overview

- API Definition
- What Are Web-Service APIs
- Types & Similarities Among Web-Service APIs
- Introducing REST
- REST Architectural Constraints
- HTTP Verbs & CRUD
- REST API Data Encoding

What Is An API?

- Application Programming Interface
- A piece of code to allow different applications to talk to exchange data.
- Two generic types of APIs
  - Those that allow internal applications in your local system to exchange data.
  - Those that use IP networking to exchange data between remote applications.

API Usage In SDNs

- Two primary uses of APIs in the world of SDN;
  - Applications connecting to Controllers
  - Controllers connecting to network devices
- Both of these types of connections utilize the Internet Protocol (IP), frequently over HTTP
- APIs use a Client/Server model
  - Application (API Client) communicates with SDN Controller (API Server)
  - SDN Controller (API Client) communicates with Switch/Router (API Server)

Web-Service APIs

- A common type of API to access data on a remote device over an IP-based network
- Data is referenced in the API via URIs or URLs
  - URI = Uniform Resource Identifier
  - A string of characters used to identify a resource on a computer network, of which the best known type is the web address or URL
  - Example: /dna/intent/api/v1/network-device/{id}/vlan
- Web-service APIs can;
  - Add new data (create)
  - Ask for data (read)
  - Modify existing data (update)
  - Destroy/erase data (delete)

Common Web-Service APIs

- SOAP (Simple Object Access Protocol)
- XML-RPC
- JSON-RPC
- REST

Similarities Among Web-Service APIs

- What do they all have in common?
  - Designed to allow different applications to share data.
  - Designed to operate across an IP network
- How are they different?
  - Some (like SOAP) are protocols with distinct rules to follow.
  - Others (like REST) are more generic “Architectural Guidelines” about how the API should function

REST

- Representational State Transfer
- An architectural style for **distributed hypermedia systems developed by Roy Fielding in his 2000 dissertation.**
- REST APIs act on “Resources”
- REST API commands utilize standard HTTP “verbs” (GET, PUT, POST, DELETE)
- For an API to be considered “RESTful” it must meet the “6 Guiding Constraints” of the REST Architecture (see next slide)

Architectural Constraints Of REST APIs

+ Uniform interface
+ Client-server model
+ Statelessness
+ Cacheable
+ Layered system
+ Code on demand (optional)

https://restfulapi.net/rest-architectural-constraints/

HTTP Verbs & CRUD

- As previously mentioned, REST APIs communicate using HTTP as the transport mechanism.
- HTTP verbs are used to perform actions on REST API resources
- For database developers (familiar with CRUD) the HTTP verbs used by REST map nicely to CRUD:

HTTP Verb Used By REST	CRUD Database Commands
Post	Create
Get	Read
Put	Update
Delete	Delete

REST API Data Encoding

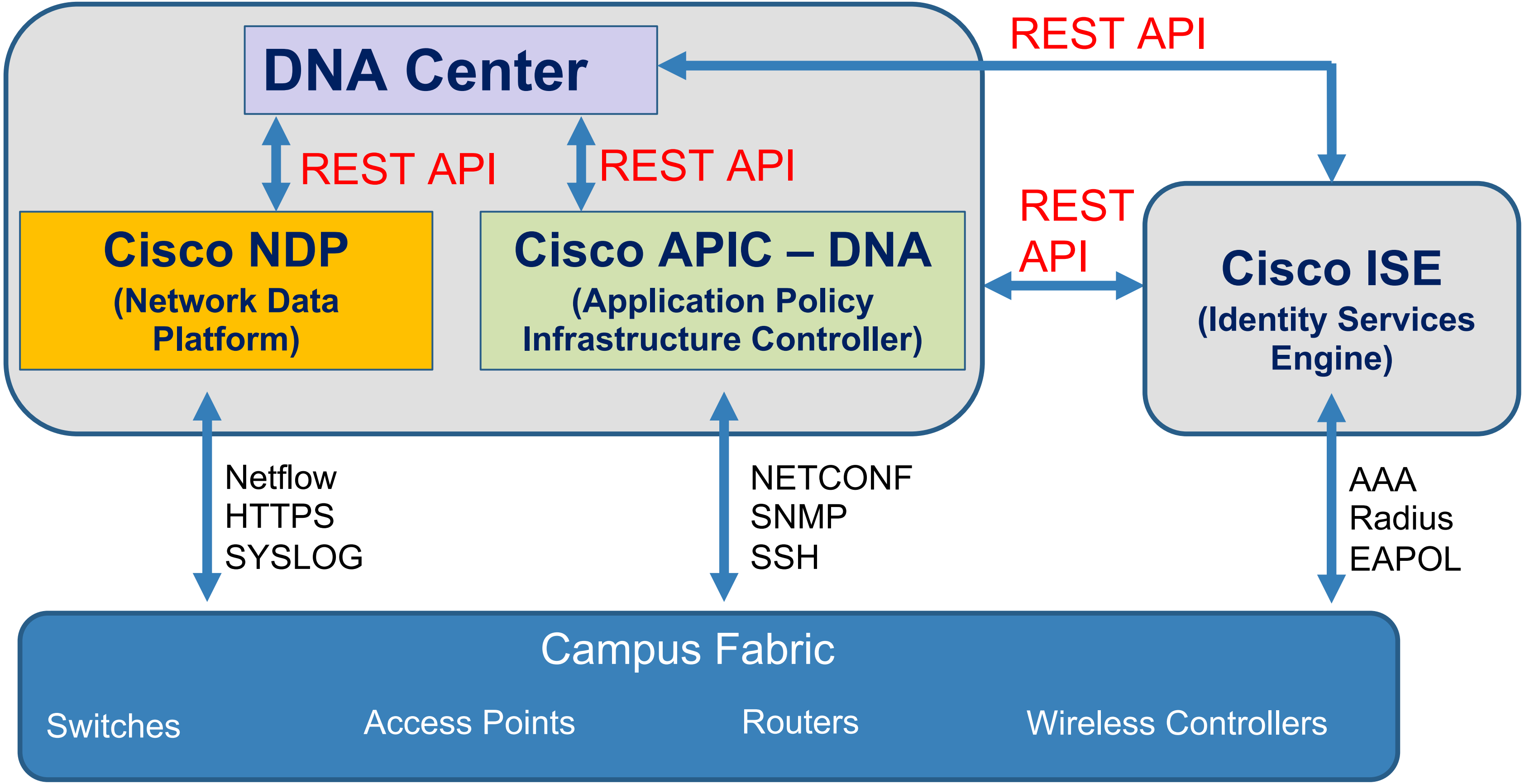
- Standard encoding required for REST API objects/resources
- REST APIs strive for;
  - Quick serialization and deserialization of objects
  - A compact format for accessing data
  - Minimize the data transfer required
  - Offer broad language support
- REST APIs typically encode data in JSON or XML format

REST APIs & Cisco DNA Center

+ Cisco has created their own REST API called the "Intent API"
+ Developers can utilize this API to create custom applications that interact with Cisco DNA Center
+ This would be considered a Northbound API from the perspective of DNA Center.
+ For more information;

https://developer.cisco.com/docs/dna-center/#!cisco-dna-center-platform-overview/intent-api

Cisco DNA Center Components



Thanks for Watching!

Overview Of Network Automation Tools

Topic Overview

- Capabilities Of Configuration Management Tools
- Common Config Mngt Tools & Their Similarities
- Masters & Agents
- Push & Pull Models
- What Are Configuration Files
- Puppet Terminology & Concepts
- Chef Terminology & Concepts
- Ansible Terminology & Concepts

Capabilities Of Configuration Management Tools

- Remove dependencies of box-by-box CLI management
- Centralize configuration and software management tasks onto a single Controller
- Allow for Day-0 device provisioning
- Create resources that can be applied against a single node, or groups of nodes
- Automate deployment of changes, either by scheduled process or manual deployment

Common Configuration Management Tools

SALTSTACK

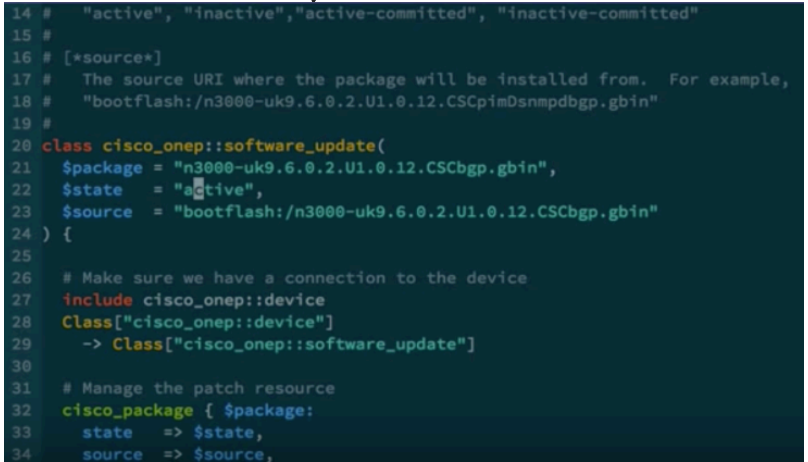
ANSIBLE

puppet

CHEF

Config Management Tools Similarities

- All tools require some CLI/scripting knowledge
  - Some CLIs resemble Cisco IOS
  - Some are totally different (like YAML or RUBY)
- Many tools include a GUI used to;
  - Schedule automated tasks
  - Manually instantiate events
- Any associated GUI references items/resources you created with the associated CLI/script



Masters & Agents

- Some types of configuration management mechanisms require two pieces, a "master" (installed in your Server) and an "agent" (code installed in the network device that responds to the Master). Puppet, Chef and SaltStack are examples of this.
- Depending on the software, the component installed in the server, and the component installed in the Router/Switch have different names/terminology:
  - Puppet: Master-Agent
  - CHEF: Master-Agent
  - SALTSTACK: Master-Minions

Push & Pull Models

- Configuration management platforms vary in their use of a Push or Pull model.
- The Push Model
  - Master pushes a configuration (or other change) down to the agents/clients
  - Can be invoked manually, or after a defined schedule
  - Good method for tools that require no Agent component
- The Pull Model
  - Agents responsible to frequently poll the Master to detect changes
  - Upon detection of a change, Agent will automatically "pull" relevant information to itself.

Creating Configuration Files

- The complexity of creating device configuration files is one differentiator among these Configuration Automation Tools
- Ansible & SaltStack utilize YAML
- Puppet & Chef utilize Ruby (or Ruby-derivatives)
  - Called “Domain-Specific Languages (DSLs)
  - Puppet configuration file is called a “Manifest”

https://www.networkworld.com/article/2172097/puppet-vs--chef-vs--ansible-vs--salt.html

Puppet Terminology & Concepts

- Puppet Master
- Puppet Agents
- Puppet Modules which give, “Providers” and “Types”
- Puppet Manifest
- Puppet Forge

Example Puppet Manifest

```
cisco_ospf {"Sample":
    ensure => present,
}
```

```
cisco_ospf_vrf {"Sample default":
    ensure => 'present',
    default_metric => '5',
    auto_cost => '46000',
}
```

```
cisco_interface_ospf {"Ethernet1/2 Sample":
    ensure => present,
    area => 200,
    cost => "200",
}
```

Chef Terminology & Concepts

- Core components
  - Chef Server
  - Workstations
  - Nodes
- Changes pushed from workstation to server…then pulled from server to node
- Cookbooks (found in the “Chef Supermarket”)
- Recipes

https://www.linode.com/docs/applications/configuration-management/beginners-guide-chef
https://supermarket.chef.io/cookbooks/cisco-cookbook#type-cisco_vlan

Example Chef Recipes

```
cisco_ospf 'Sample' do
  action :create
end
```

```
cisco_ospf_vrf 'dark_blue vrf1' do
  auto_cost 46000
  default_metric 10
  log_adjacency 'log'
  timer_throttle_lsa_start 8
  timer_throttle_lsa_hold  5600
  timer_throttle_lsa_max   5800
  timer_throttle_spf_start 277
  timer_throttle_spf_hold  1700
  timer_throttle_spf_max   5700
end
```

```
cisco_interface_ospf 'Ethernet1/2' do
  action :create
  ospf 'Sample'
  area 200
  cost 200
  dead_interval 200
  hello_interval 200
  message_digest true
  message_digest_encryption_type 'cisco_type_7'
  message_digest_algorithm_type 'md5'
  message_digest_key_id 7
  message_digest_password '088199c89d4a5ee'
  passive_interface true
end
```

Ansible Terminology & Concepts

- Ansible Master
- Utilizes SSH to connect to managed devices
- Ansible Playbooks
- Ansible Modules
  - Make API calls to managed nodes
  - Apply configurations
  - Example: ios_config module

https://docs.ansible.com/ansible/latest/modules/ios_config_module.html

Ansible Module Example

```yaml
- name: configure top level configuration
  ios_config:
    lines: hostname {{ inventory_hostname }}

- name: configure interface settings
  ios_config:
    lines:
      - description test interface
      - ip address 172.31.1.1 255.255.255.0
    parents: interface Ethernet1

- name: configure ip helpers on multiple interfaces
  ios_config:
    lines:
      - ip helper-address 172.26.1.10
      - ip helper-address 172.26.3.8
    parents: "{{ item }}"
  with_items:
    - interface Ethernet1
    - interface Ethernet2
    - interface GigabitEthernet1
```

INE

Thanks for Watching!

Encoding Data With JSON

ine.com

Topic Overview

+ JSON Overview
+ JavaScript Foundational Overview
+ Benefits Of JSON
+ JSON Value Types
+ JSON Syntax Rules

JSON Overview

- Pronounced “Jay Sahn”
- JavaScript Object Notation
- A subset of JavaScript syntax
- JSON “Objects” used for representing data that is transferred between server and client.
  - Simply put, it is a method of generically describing data.
- Used extensively by web-service APIs (such as REST APIs)

Javascript Foundational Overview

- One of several core languages used for designing websites (also included are HTML and CSS)
  - A scripting language
  - Defines variables, objects, functions, etc
  - Identified in raw HTML with the <script> and </script> tags
- Adds behaviors and interactivity to websites
- Javascript only runs on the client-side.
- Javascript Objects
  - A container that encloses one-or-more **name:value** pairs
  - Sometimes called a **key-value pair**
    - Example of Javascript object: **{"name" : "Keith", "employer": "INE", "salary": 100};**

Benefits Of JSON

- It is light-weight
- It is language independent
- Easy to read and write
- Text based, human readable data exchange format

XML


```xml
<?xml version="1.0" encoding="UTF-8"?>
<response uri="http://fake-url.com">
   <result>
      <Accounts>
         <row no="1">
            <FL val="id">1242160000000072037</FL>
            <FL val="phone"><![CDATA[null]]></FL>
            <FL
val="website"><![CDATA[www.joshuawyse.com]]></FL>
            <FL val="employees"><![CDATA[0]]></FL>
            <FL val="billingStreet"><![CDATA[null]]></FL>
            <FL val="shippingStreet"><![CDATA[null]]></FL>
            <FL val="billingCity"><![CDATA[null]]></FL>
            <FL val="shippingCity"><![CDATA[null]]></FL>
            <FL val="billingState"><![CDATA[null]]></FL>
            <FL val="shippingState"><![CDATA[null]]></FL>
            <FL val="billingCode"><![CDATA[null]]></FL>
            <FL val="shippingCode"><![CDATA[null]]></FL>
            <FL val="billingCountry"><![CDATA[null]]></FL>
            <FL val="shippingCountry"><![CDATA[null]]></FL>
            <FL val="description"><![CDATA[null]]></FL>
         </row>
      </Accounts>
   </result>
</response>
```


JSON


```json
{
    "id": "1242160000000072038",
    "description": "3",
    "website": "3",
    "numberOfEmployees": "3",
    "phone": "3",
    "name": "account3",
    "shippingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    },
    "billingAddress": {
        "country": "3",
        "stateOrProvidence": "3",
        "city": "3",
        "postalCode": "3",
        "street1": "3"
    }
}
```

JSON Value Types

- Objects
  - Always surrounded by curly brackets
  - Composed of one-or-more name-value pairs
  - Example:

    {“Department”:”Payroll”, “VLAN”:300, “Manager”:”Bob”}
- String – must be enclosed in double quotes
  - Example: {“name”: “John”}
- Numbers – integers or floats
  - Example: {“age”: 5}

JSON Value Types (contd)

- Arrays
  - Similar to what Python calls a “list”
  - Comma, separated list of values enclosed by square brackets
  - Example: {“classAges” : [5 , 8, 9, 10]}
- Booleans
  - A true or false statement
  - Example: {“sale”: true}
- Null
  - Example: {“route”: null}

JSON Value Types (contd)

- JSON does not allow the following values:
  - Functions
  - Dates
  - Undefined

JSON Syntax Rules

+ Data is in name/value (key-value) pairs
+ Multiple name-value pairs within a single object are separated by commas
+ Curly braces hold objects
+ Square brackets hold arrays
+ Spaces and line breaks don't matter.

This…

```
{"instructor": [{"name":"Keith", "employer":"INE", "salary":70}]}
```

Or this…

```
{
  "instructor":[
    {
      "name":"Keith",
      "employer":"INE",
      "salary":70
    }
  ]
}
```

Interpreting JSON Data

- Lab demonstration
- After viewing JSON data from a router, answer the following question;
  - Where would this router send a packet going to 44.1.1.55?
  - Will the FastEthernet0/1 interface be allowed to transmit a Telnet packet, sourced from 1.1.1.1 and destined to 20.20.30.1?

Thanks for Watching!

INE