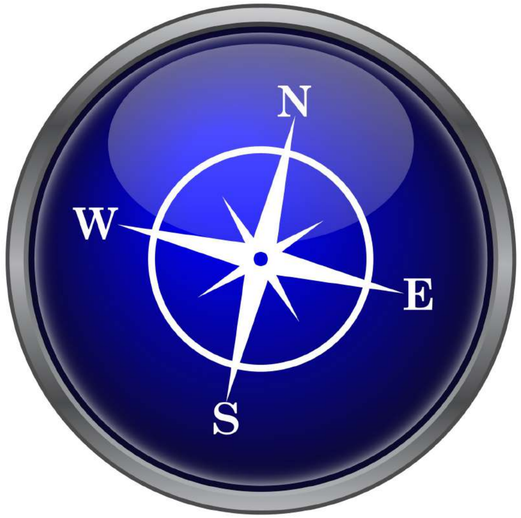
Module 6
Network Automation

Overview of SDN

Overview of Software Defined Networking (SDN)



Applications

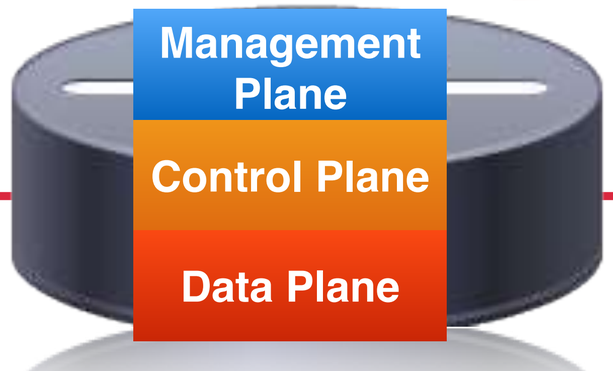
Northbound Interfaces

SDN Controller

Southbound Interfaces

Terms to Know:

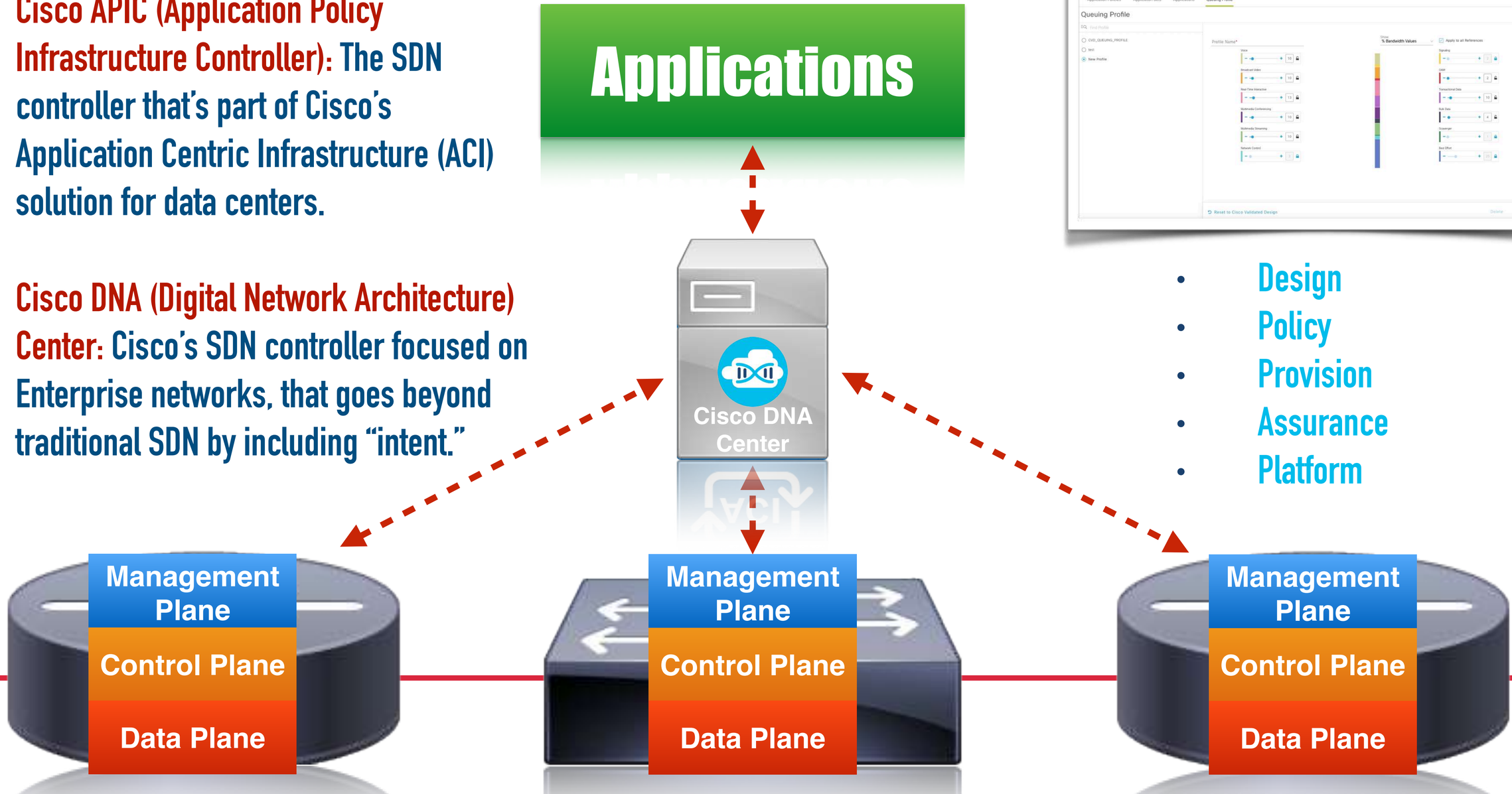
- Distributed Control Plane
- API
- SBI
- Centralized Control Plane
- OpenFlow
- NBI
- RESTful APIs
- JSON



Cisco SDN Controllers

- **Cisco APIC (Application Policy Infrastructure Controller):** The SDN controller that's part of Cisco's Application Centric Infrastructure (ACI) solution for data centers.

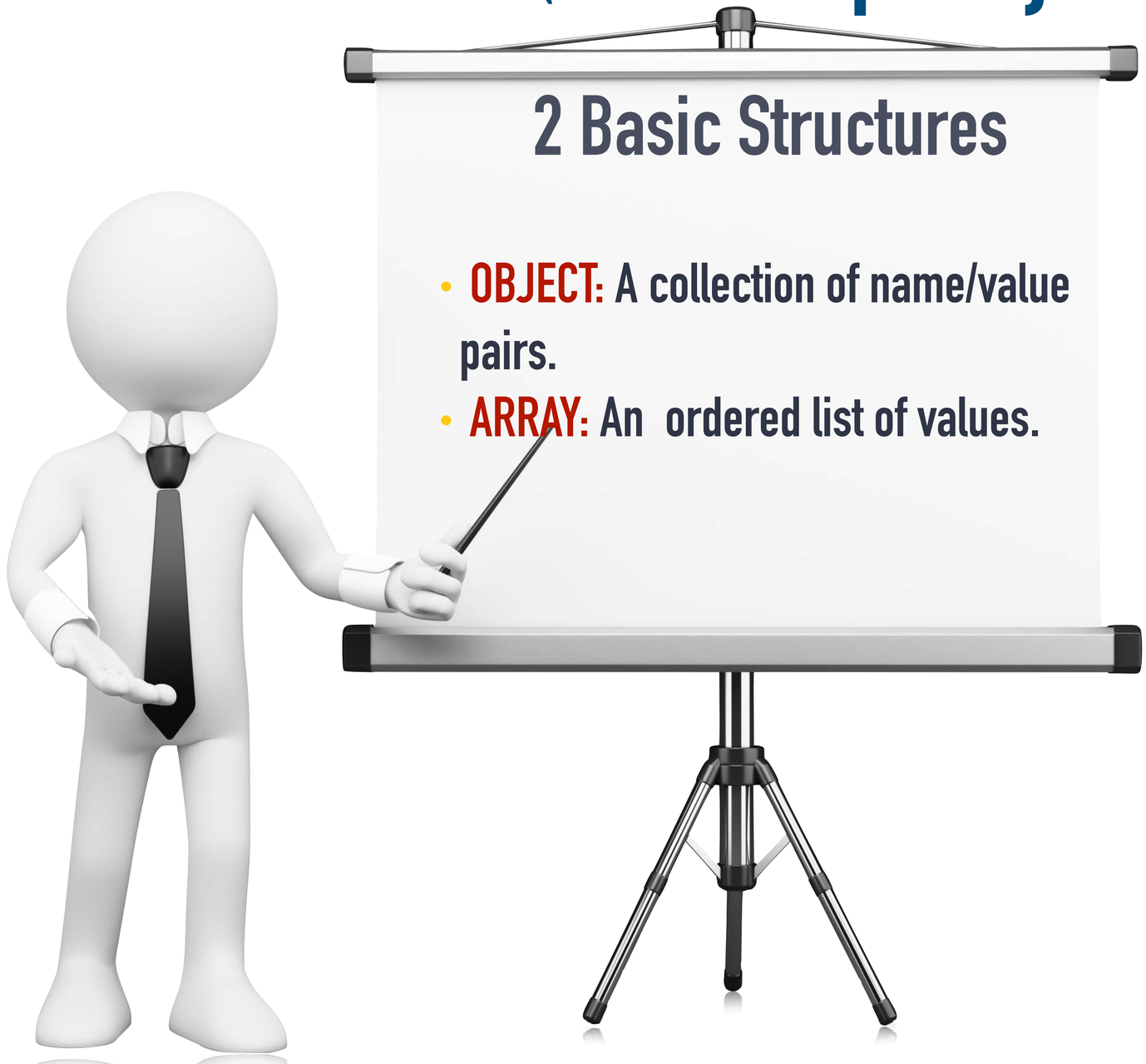
- **Cisco DNA (Digital Network Architecture) Center:** Cisco's SDN controller focused on Enterprise networks, that goes beyond traditional SDN by including "intent."



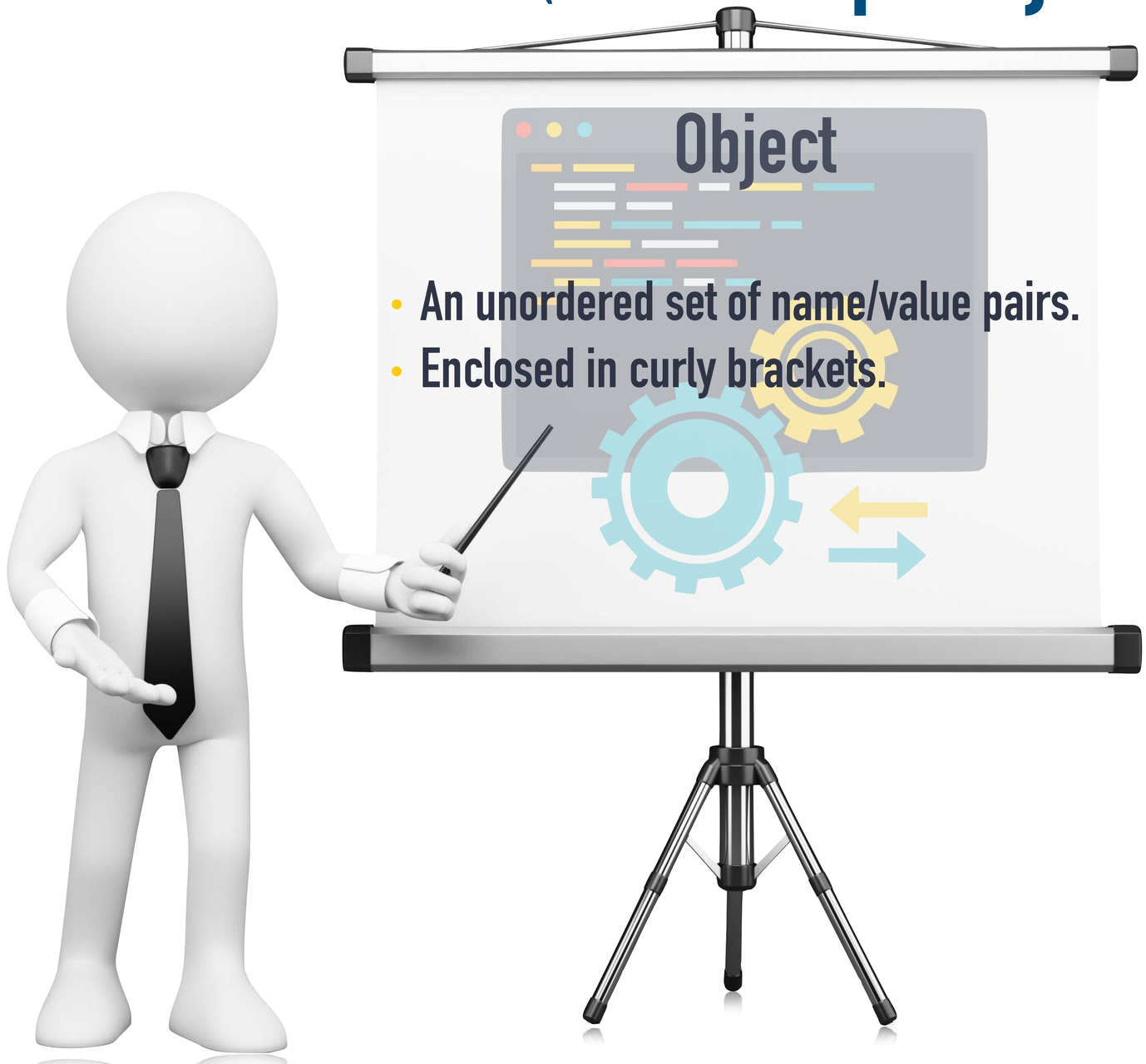
- Design
- Policy
- Provision
- Assurance
- Platform

JSON Formatting

JSON (JavaScript Object Notation) Format



JSON (JavaScript Object Notation) Format

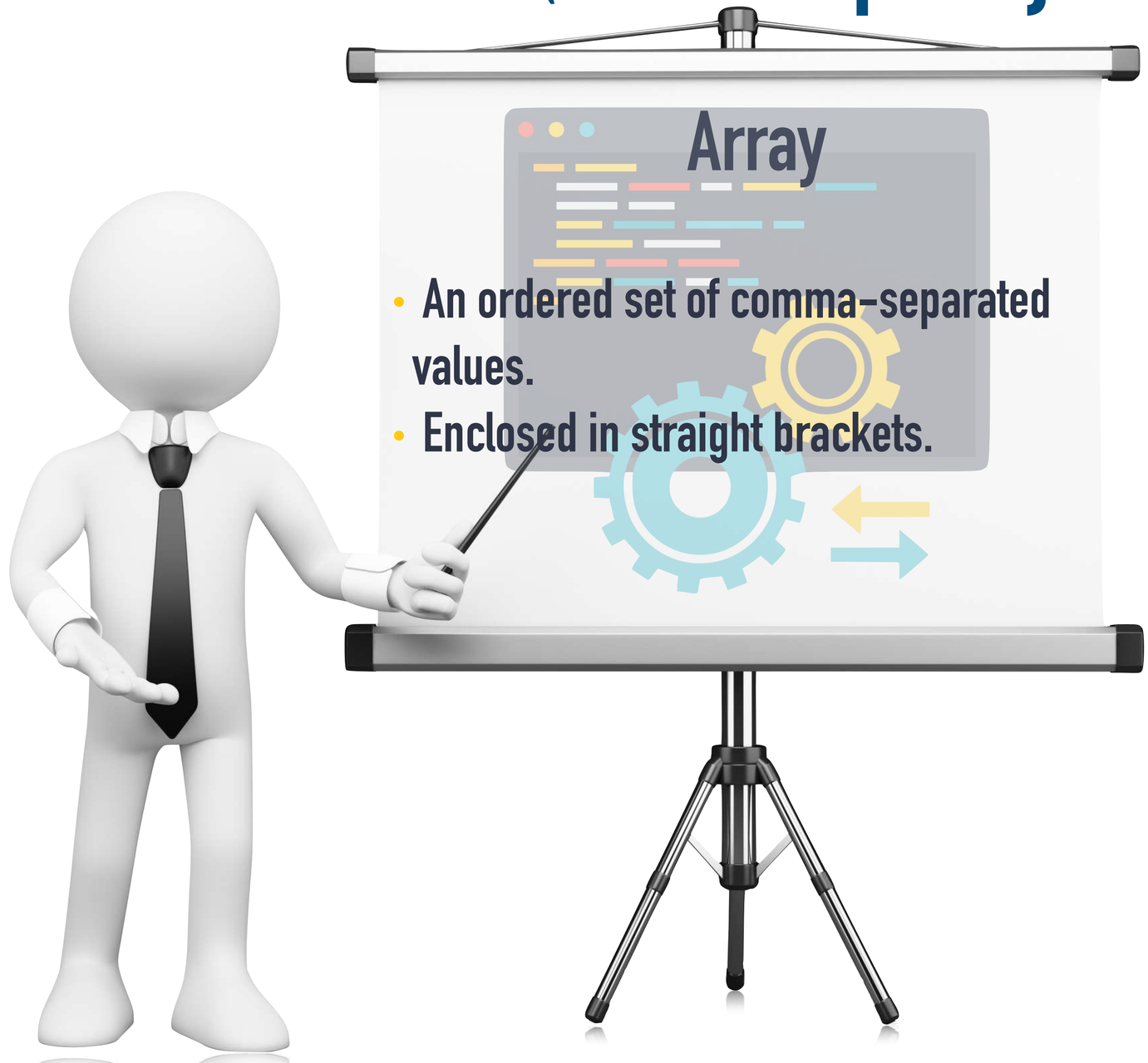



```
{"firstName":"Kevin","lastName":"Wallace"}
```

With whitespace:

```
{
    "firstName": "Kevin",
    "lastName": "Wallace"
}
```

JSON (JavaScript Object Notation) Format



["CCNA","CCNP Enterprise","CCIE Enterprise Infrastructure“]

With whitespace:

```
[
    "CCNA",
    "CCNP Enterprise",
    "CCIE Enterprise Infrastructure”
]
```

JSON (JavaScript Object Notation) Format

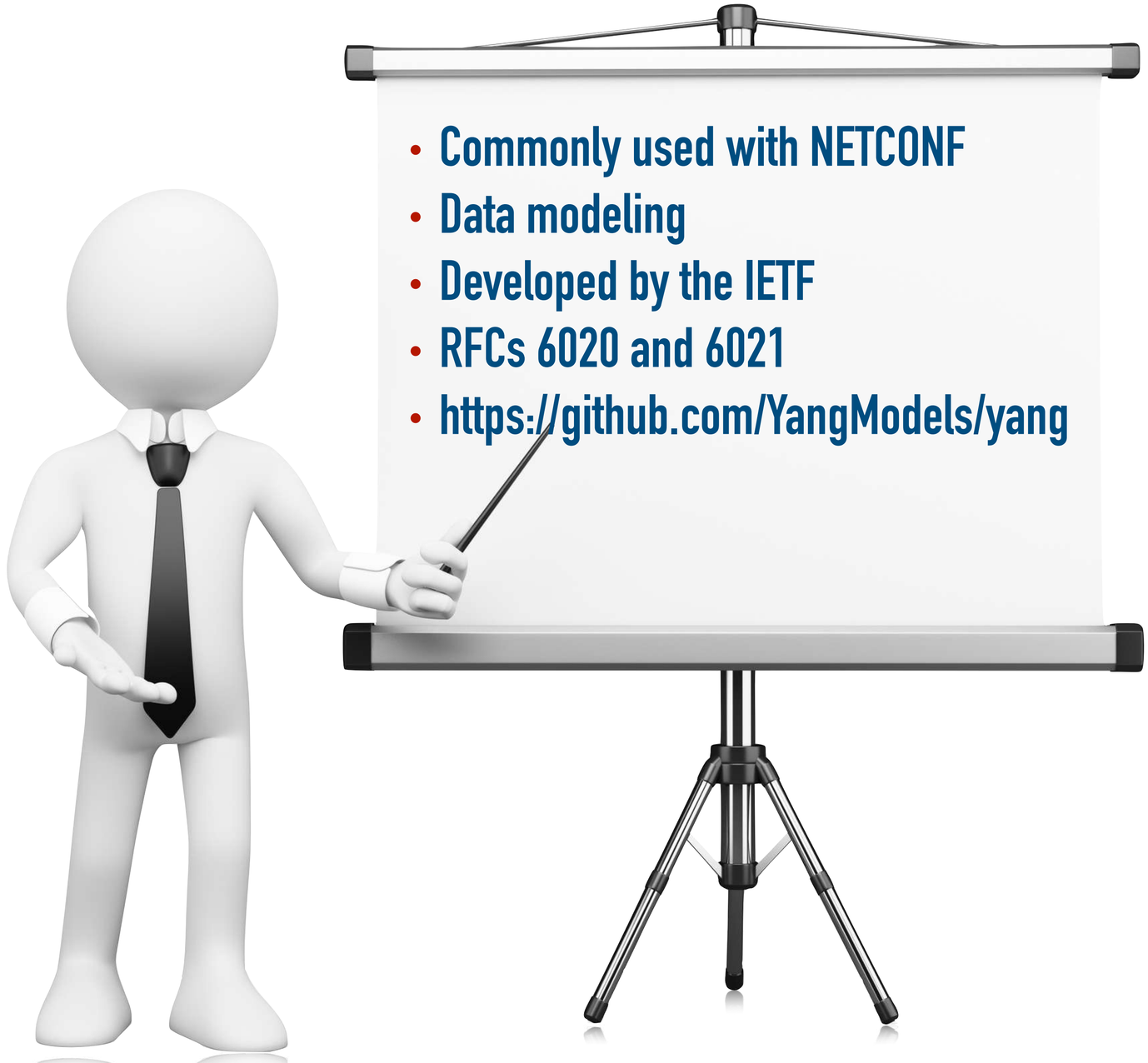
Value

- Can be a string, number, object, array, null, true, or false.
- Example of a JSON validator:
  https://jsonlint.com

```
{"Name":"Kevin Wallace","CCIE #":7945,"Tracks":["Enterprise Infrastructure","Collaboration"]}
```

YANG Data Modeling

YANG Data Modeling



- Commonly used with NETCONF
- Data modeling
- Developed by the IETF
- RFCs 6020 and 6021
- https://github.com/YangModels/yang

Data Modeling Example

Apple iPhone

Model: 11, 11 Pro, 11 Pro Max, Other

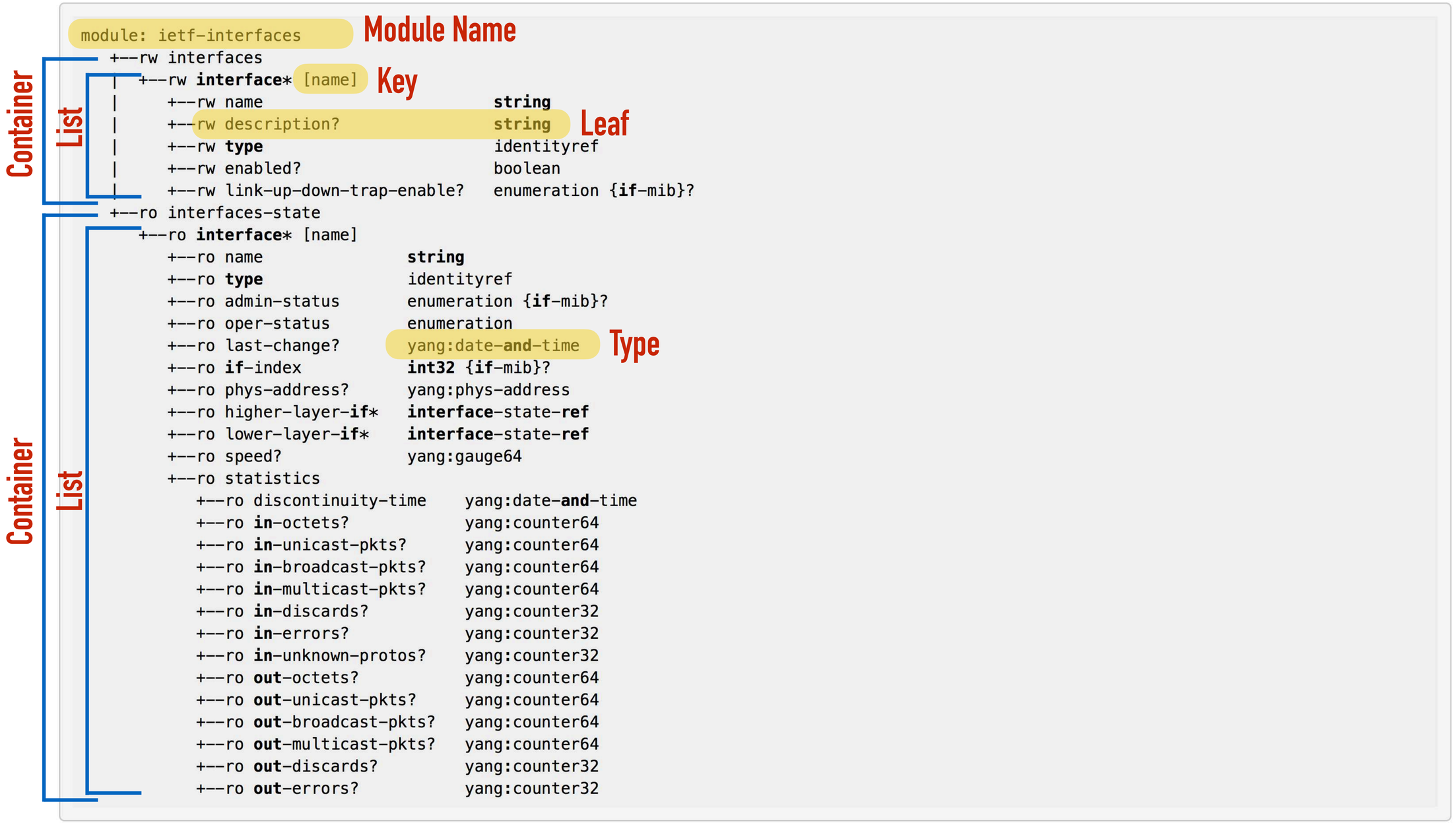
Display Size : 5.8”, 6.1”, 6.5”, Other

Color: Midnight Green, Silver, Space Gray, Gold, Purple, Yellow, Green, Black, White, (PRODUCT)RED ,Other

Capacity: 64 GB, 128 GB, 256 GB, 512 GB, Other

11 Pro Max, 6.5”, Space Gray, 256 GB

YANG Data Model of a Network Interface



YANG Data in XML Format

```
<interfaces xmlns="urn:ietf:params:xml:ns:yang:ietf-interfaces">
    <interface>
        <name>GigabitEthernet1</name>
        <description>DONT'T TOUCH ME</description>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip">
            <address>
                <ip>10.10.10.10</ip>
                <netmask>255.255.255.0</netmask>
            </address>
        </ipv4>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
    </interface>
    <interface>
        <name>GigabitEthernet2</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
    </interface>
    <interface>
        <name>GigabitEthernet3</name>
        <type xmlns:ianaift="urn:ietf:params:xml:ns:yang:iana-if-type">ianaift:ethernetCsmacd</type>
        <enabled>true</enabled>
        <ipv4 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
        <ipv6 xmlns="urn:ietf:params:xml:ns:yang:ietf-ip"/>
    </interface>
    .
    .
    .
</interfaces>
```

Namespace

Leaf

Interface Node

IPv4 Node

Interfaces Container

NETCONF

NETCONF

CLI

(Telnet, SSH)

SNMP

NETwork CONFiguration

NETCONF

Request

Reply

Server

NETCONF

NETCONF

Manager

Agent

NETCONF

NETCONF

Manager

Agent

NETCONF

NETCONF

Manager

NETCONF

Agent

NETCONF DEMO

RESTCONF

RESTCONF

Client

HTTPS

Server

RESTCONF

- Uses the same YANG Data Model definitions used by NETCONF
- Supports JSON and XML formatting

Representational State Transfer (REST)

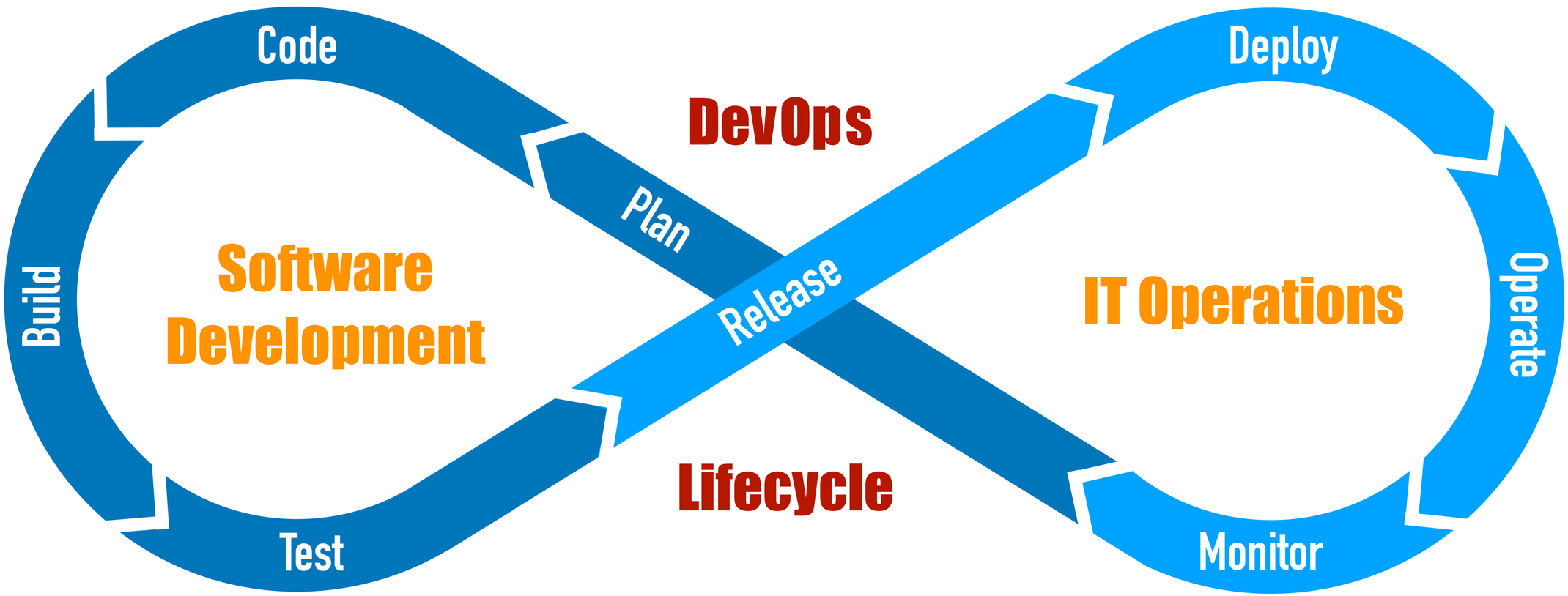
Create
Read
Update
Delete

CRUD Function	HTTP Verb	NETCONF Operation
Create	POST	<edit_config> (operation="create")
Read	GET	<get> , <get_config>
Update	PUT or PATCH	<edit_config> (operation="create/replace" or "merge")
Delete	DELETE	<edit_config> (operation="delete")

RESTCONF DEMO

Orchestration Tools

Configuration Management Tools

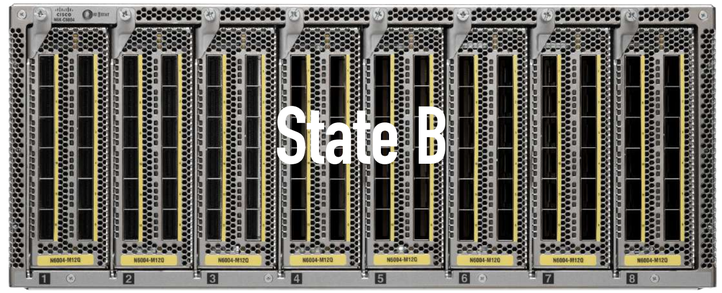


Puppet

Intent = State B

Cisco Nexus 6000

Cisco Nexus 6000



Cisco Nexus 5600

- Written in Ruby
- Resource Declaration: Identifies a resource in a node that is manageable by Puppet and the type of that resource
- Class: A collection of common configuration settings (contains Resources)
- Manifest: A file that contains Puppet code (contains Classes)
- Module: A grouping of files and directories that can contain Puppet manifests (contains Manifests)

Puppet

```
#Configuring Interface eth1/4
     cisco_interface { "Ethernet1/4" :
        shutdown             => true,
        switchport_mode      => disabled,
        description          => 'managed by puppet',
        ipv4_address         => ‘172.16.1.24’,
        ipv4_netmask_length => 19,
      }
```

Chef

Cisco Cookbook

Chef Client

Cisco Nexus 9300

Cisco Nexus 9300

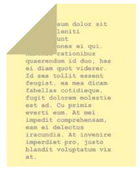
Cisco Nexus 3100

- Written in Ruby
- Recipes: A set of instructions for a specific task
- Cookbook: A collection of recipes

Chef

```
cisco_interface 'Ethernet1/1' do
  action :create
  ipv4_address '192.168.1.1'
  ipv4_netmask_length 24
  ipv4_proxy_arp true
  shutdown false
  switchport_mode 'disabled'
end

cisco_interface 'Ethernet1/2' do
  action :create
  access_vlan 200
  shutdown false
  switchport_mode 'access'
  switchport_vtp false
end
```

Ansible

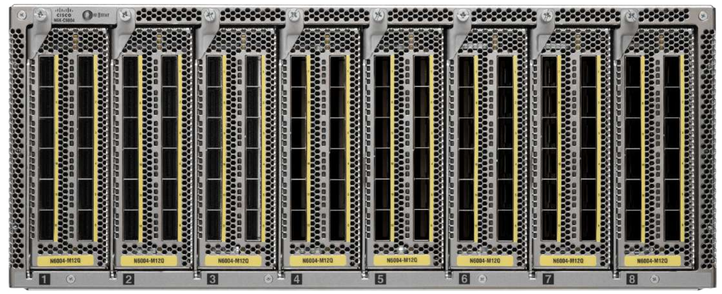


SSH

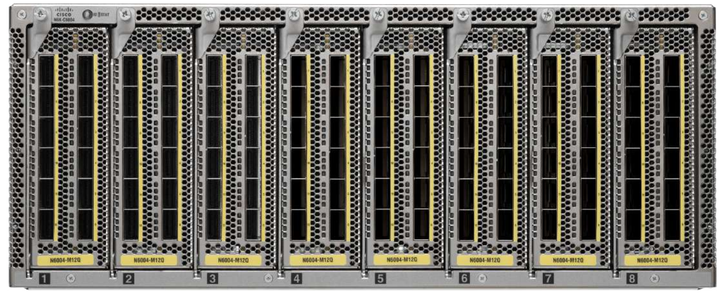
SSH

SSH

Cisco Nexus 6000



Cisco Nexus 6000



Cisco Nexus 5600

- **Playbook:** Configuration instructions
- **YAML (YAML Ain't Markup Language):** Configuration instructions
- **Inventory:** Contains a list of devices
- Playbook is run against the Inventory
- No Agent required on device

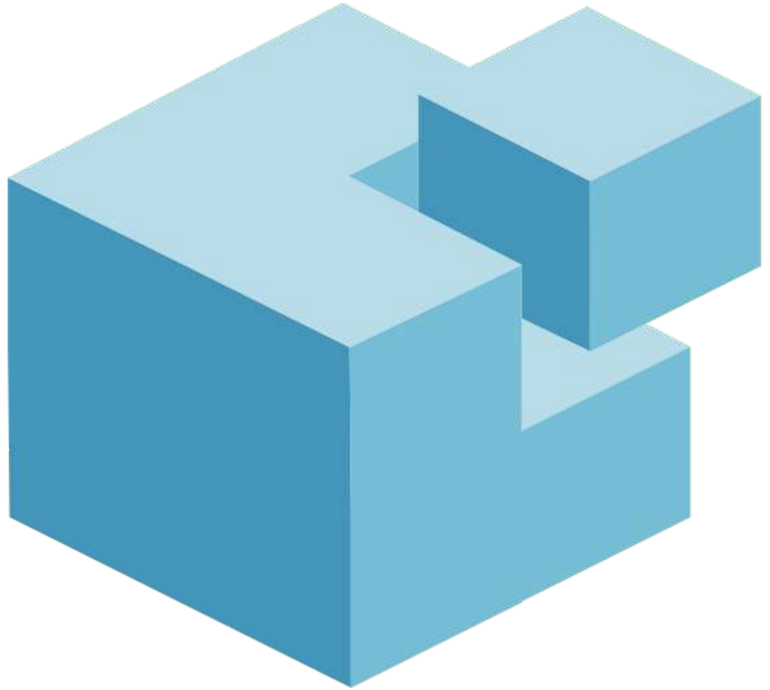
Ansible

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

- name: configure interface settings
  ios_config:
    lines:
      - description Engineering_Interface
      - ip address 10.5.5.1 255.255.255.192
    parents: interface GigabitEthernet1

- name: configure ip helpers on multiple interfaces
  ios_config:
    lines:
      - ip helper-address 10.1.1.100
    parents: "{{ item }}"
  with_items:
    - interface GigabitEthernet1
    - interface GigabitEthernet2
```

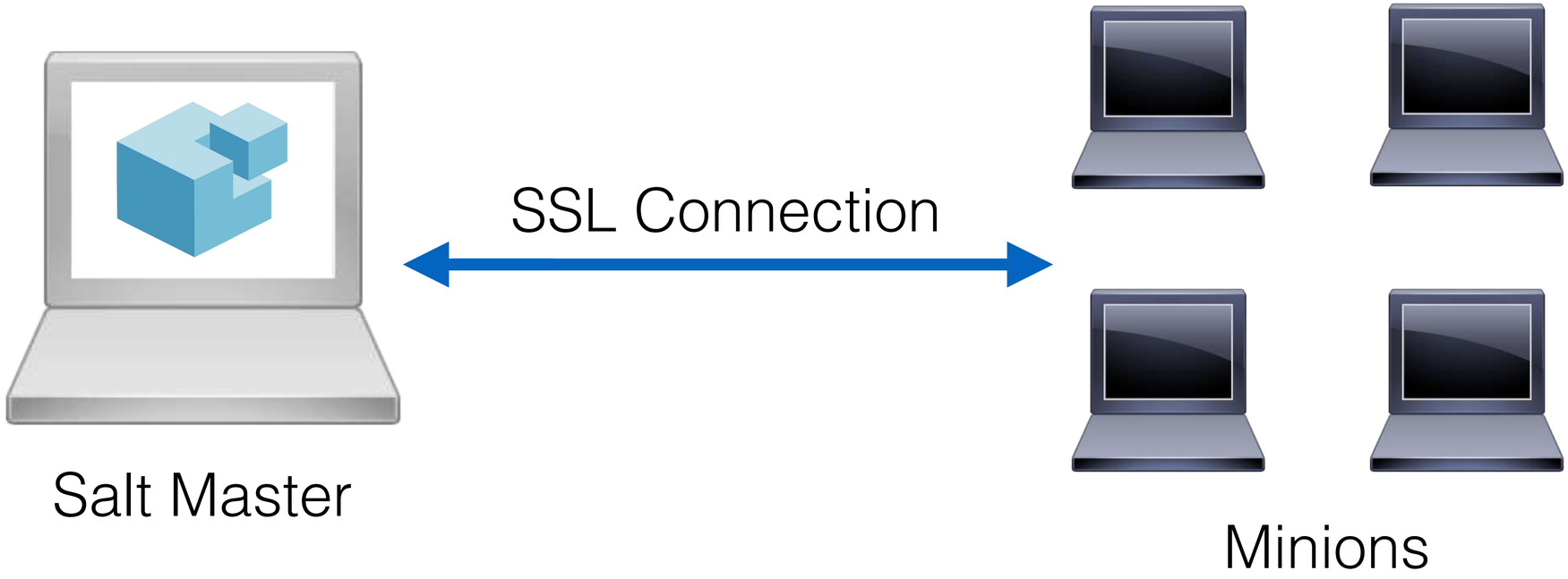
SaltStack

Components:

- Works as both a push and pull model
- Build on the Python programming language
- Instructions pushed out in YAML



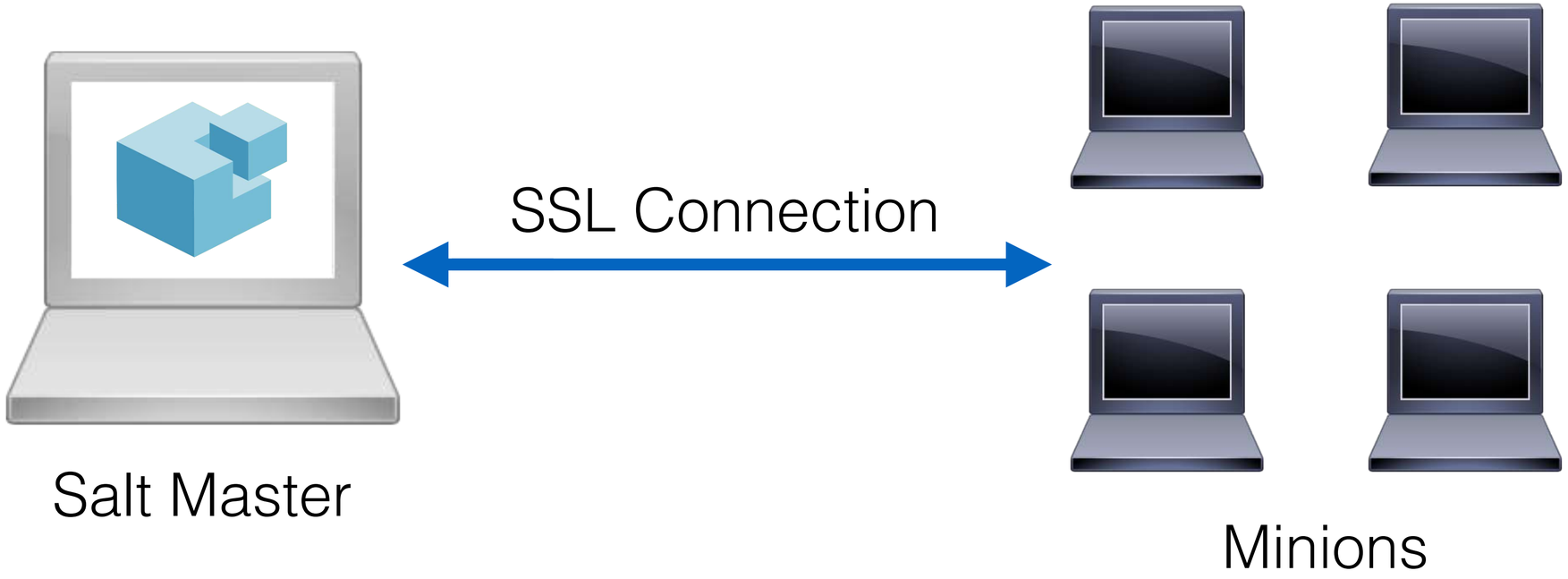
SaltStack



Salt Master:

- Main hub for configuration
- Instructions and configurations pushed out to Minions

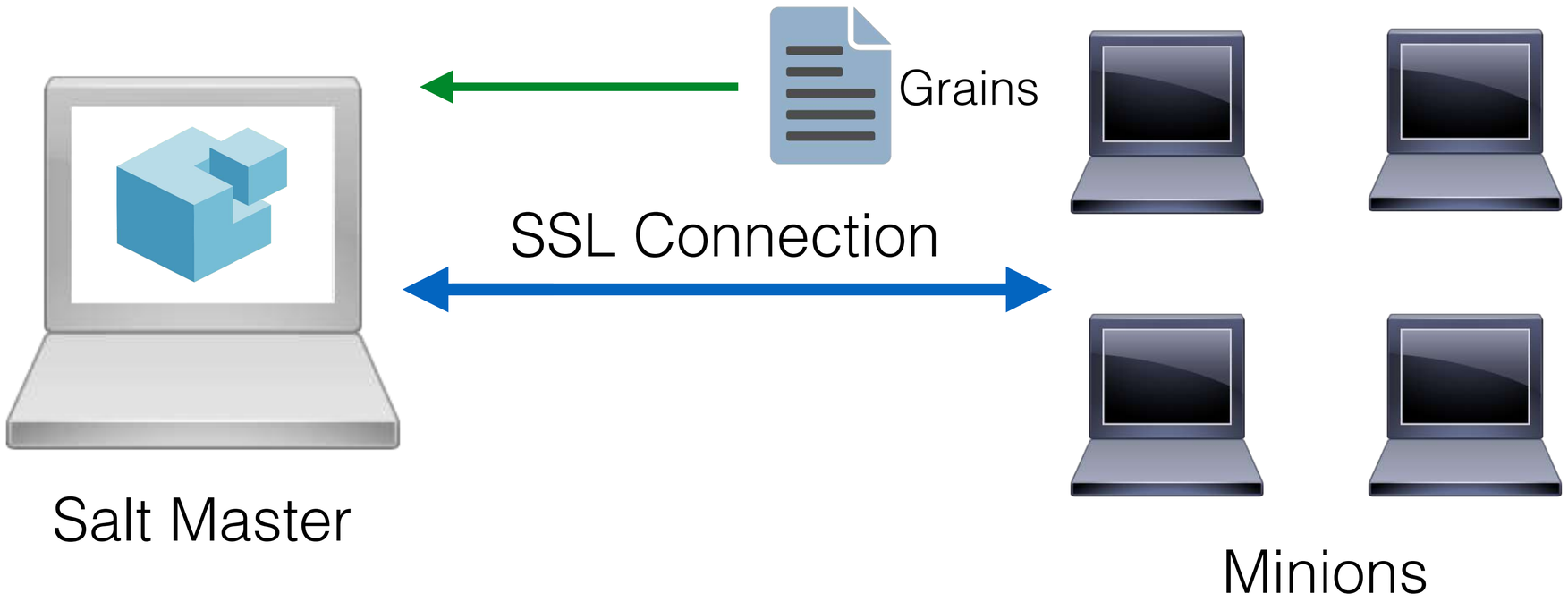
SaltStack



Minions:

- Runs as agent software installed on managed nodes
- Used to receive and execute commands, and report information back to the Salt Master

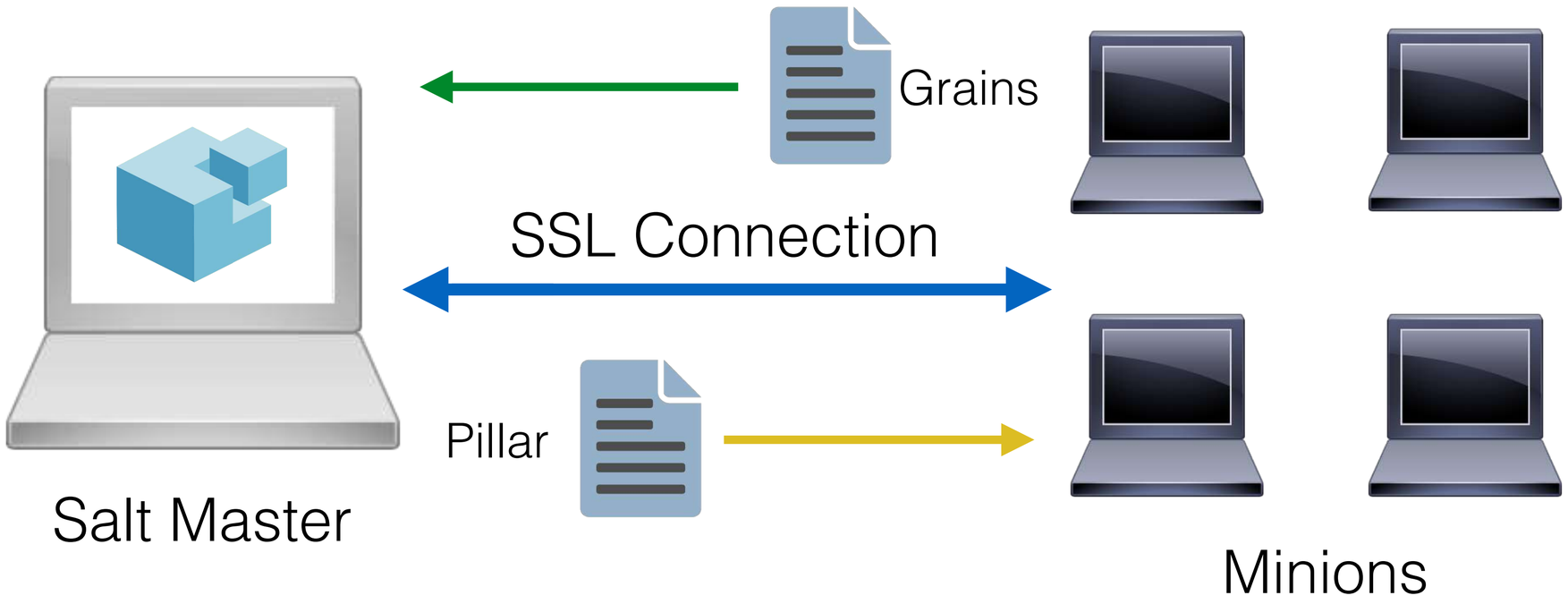
SaltStack



Grains:

- Information about managed nodes sent to Salt Master
- Network information, operating system, hardware details, etc.
- Information is static and not real-time

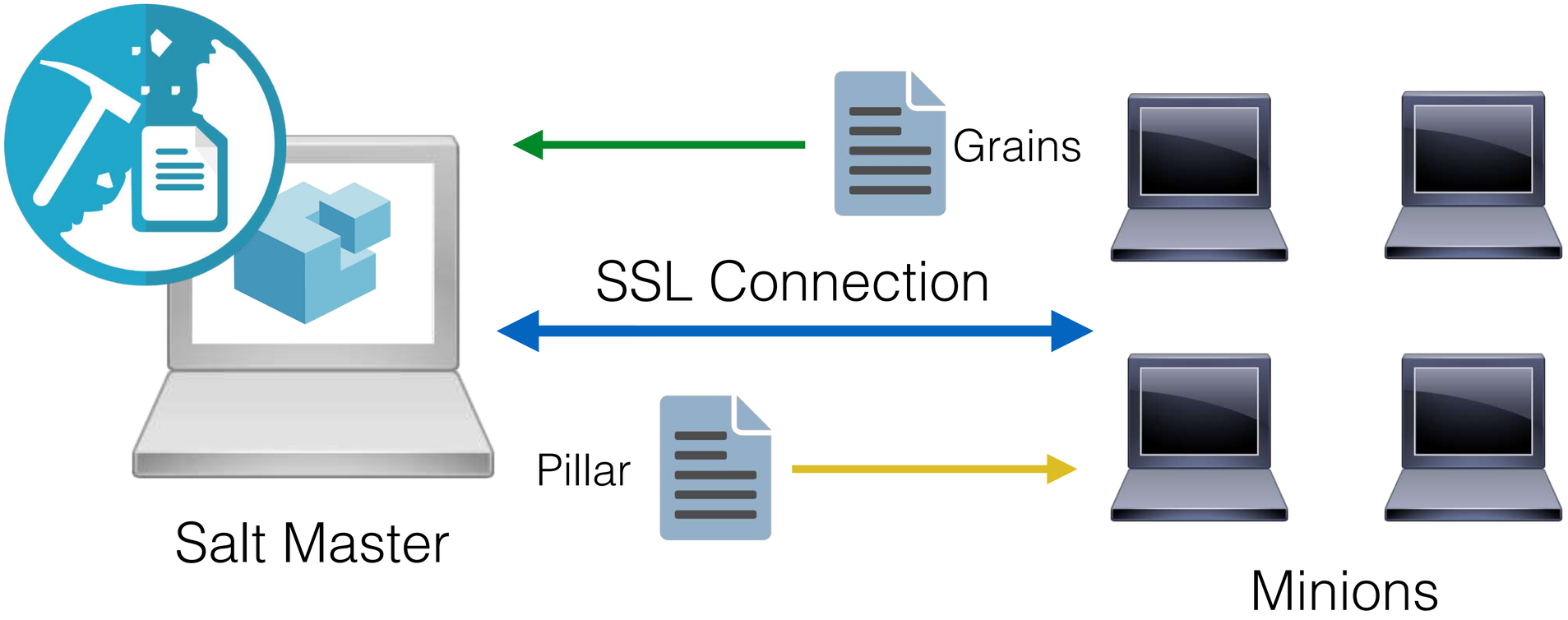
SaltStack



Pillar:

- Stores data that Minions can retrieve
- Contains minion-specific sensitive data
- Cryptographic keys, passwords, etc.

SaltStack



Salt Mine:

- Captures arbitrary information from managed Minions
- Information is made available to all of the Minions
- Salt Mine data is much more up-to-date than Grain information

Python Demo

Cisco DNA Center and vManage Demos

Cisco DNA Center

https://sandboxdnac2.cisco.com

Username: devnetuser

Password: Cisco123!

Cisco vManage

https://cisco.com/go/sdwandemos

Username: demo

Password: demo1234!

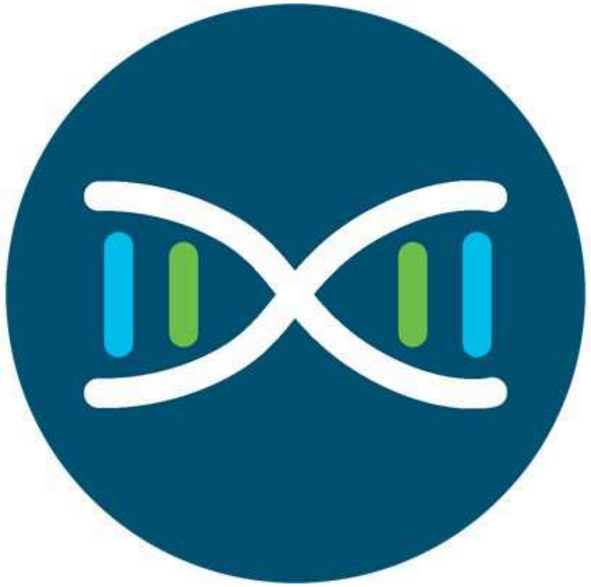
Cisco DNA Center and vManage APIs

Cisco DNA Center and vManage APIs

Application Programming Interface (API):

- Building block for inter-application communication
- Blocks of code that create software
- Pre-built APIs save time during development

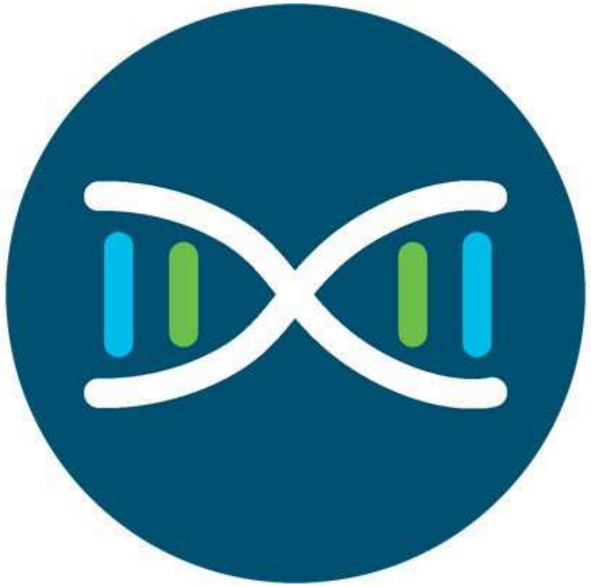
Cisco DNA Center and vManage APIs



Intent (Northbound) APIs:

- Representational State Transfer (REST) APIs
- Common in web services
- Use HTTP requests for data transfer
- GET, PUT, POST, and DELETE requests

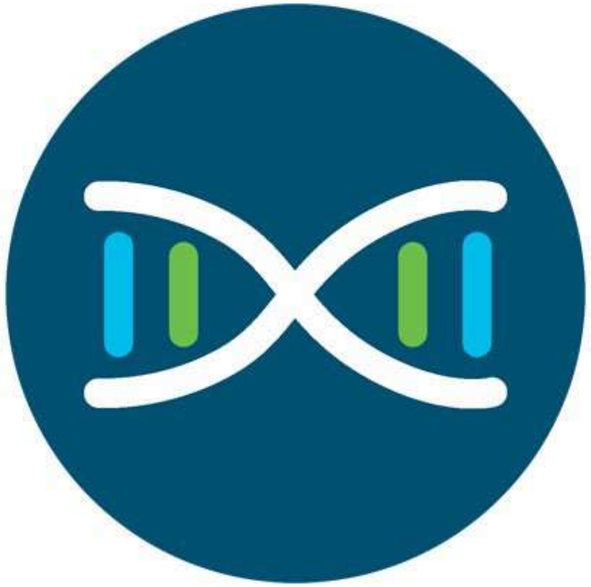
Cisco DNA Center and vManage APIs



Intent (Northbound) APIs:

- Creating and managing sites
- Retrieving network health information
- Device onboarding and provisioning
- Troubleshooting commands
- Policy creation

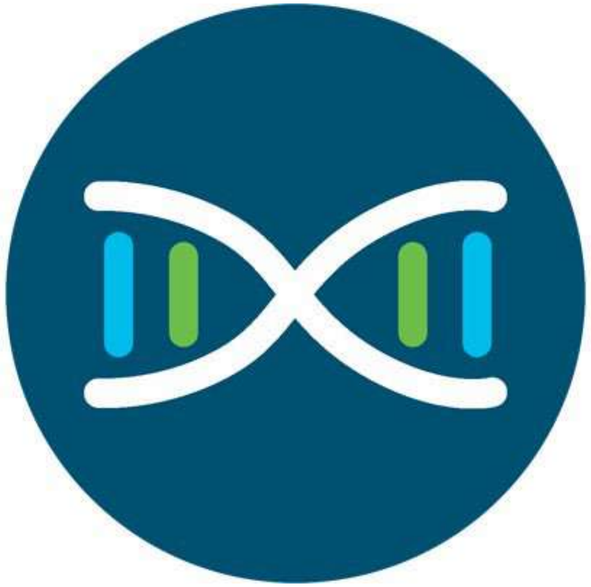
Cisco DNA Center and vManage APIs



Multivendor Support (Southbound) APIs:

- Multivendor Software Development Kit (SDK)
- SDKs can include multiple APIs
- SDKs allow for management of non-Cisco devices

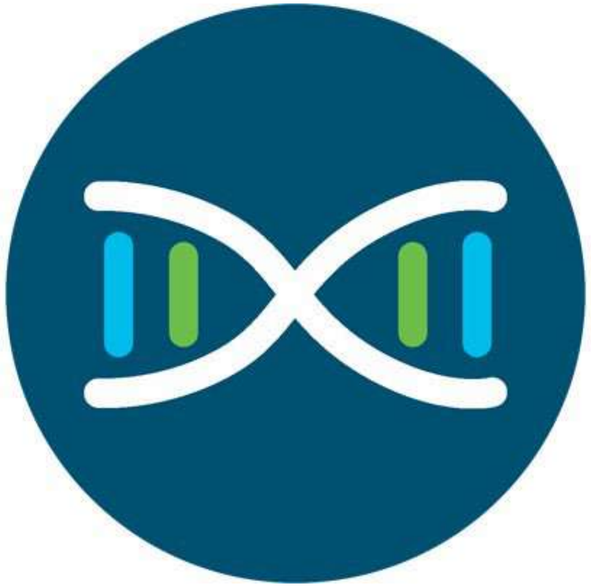
Cisco DNA Center and vManage APIs



Integration (Westbound) APIs:

- Integrate Cisco DNA Center with other platforms
- Communicate with third-party IT service management solutions
- Ticket and request automation

Cisco DNA Center and vManage APIs



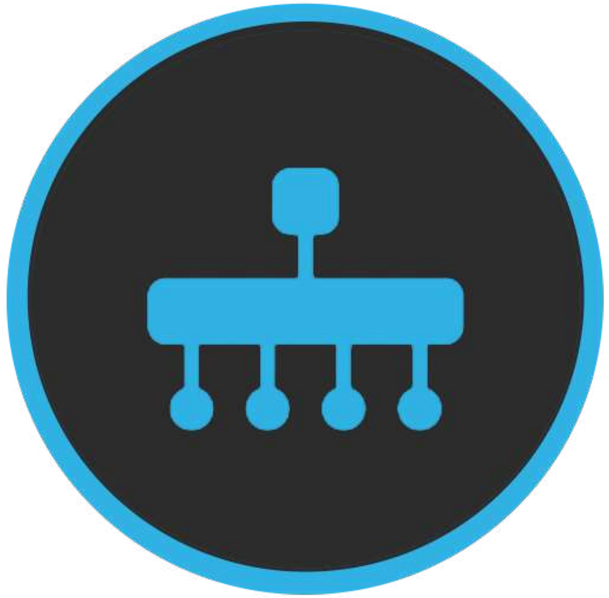
Events and Notifications (Eastbound) APIs:

- Allow external systems to take action against notifications
- Especially useful for security compliance

Cisco DNA Center and vManage APIs

vManage API Resource Collections:

- Administrative APIs
- Certificate Management APIs
- Configuration APIs
- Device Inventory APIs
- Monitoring APIs
- Real-Time Monitoring APIs
- Troubleshooting APIs



REST API Response Codes

REST API Response Codes

Extensible Markup Language (XML):

- Easy to interpret by both humans and software
- Indentation is not mandatory but helps readability

```
<note>
  <to>John</to>
  <from>Susan</from>
  <heading>Reminder</heading>
  <body>Don't forget to buy eggs!</body>
</note>
```

REST API Response Codes

JavaScript Object Notation (JSON):

- Gaining popularity in network automation
- Cisco DNA Center expects incoming JSON data

REST API Response Codes

Common Response Codes:

- *200 OK* = Successful GET or POST command
- *201 CREATED* = Resource was created
- *400 BAD REQUEST* = Client syntax issue
- *401 UNAUTHORIZED* = User authentication needed
- *403 FORBIDDEN* = Request received but refused by server
- *404 NOT FOUND* = No resource available to return

REST API Security Considerations

REST API Security Considerations

REST API Security:

- Should be stateless authentication
- Authentication and authorization should not be cached
- Every request made to server should require validation

REST API Security Considerations

REST API Security:

- Users should only have enough privileges to do their job
- Actions should be denied without explicit permission
- Security design should be simple and intuitive
- Cached credentials should not be allowed
- User privileges shouldn’t be solely based on a role
- REST APIs should be shielded from unnecessary access

REST API Security Considerations

REST API Security Best Practices:

- Always use HTTPS
- Use strongly hashed passwords
- Immediate input parameter validation
- Consider using OAuth over basic authentication