Network Automation: DevOps, Python, and More

Ila Gokarn
Software Solutions Lead, APAC

About Arista Networks

15%
Market Share

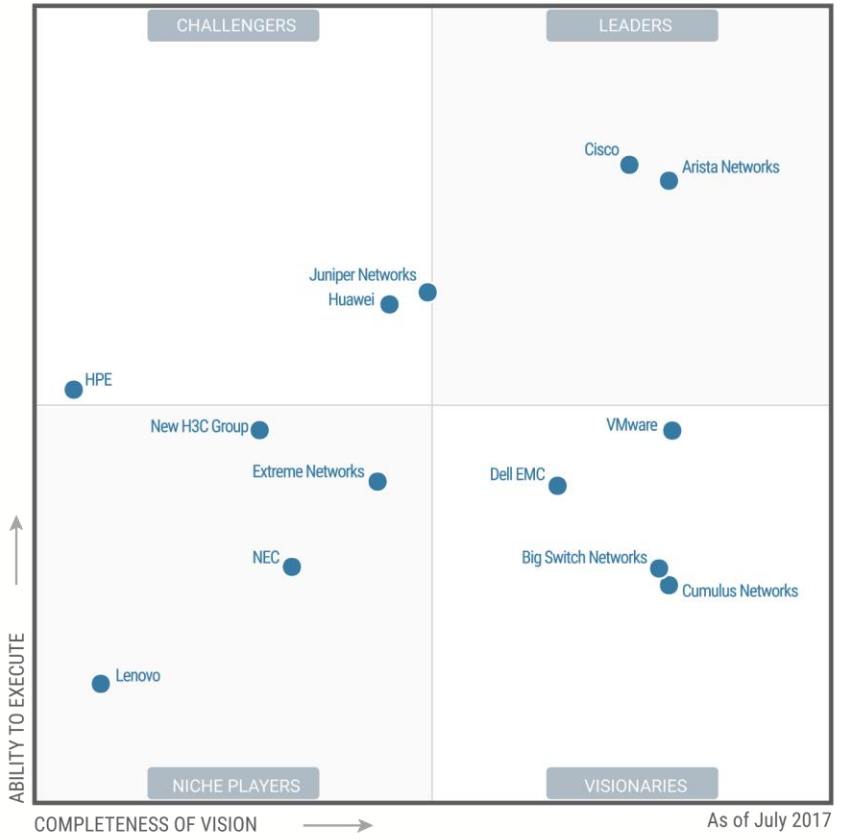
15+ Million
Ports Shipped

5000+
Customers

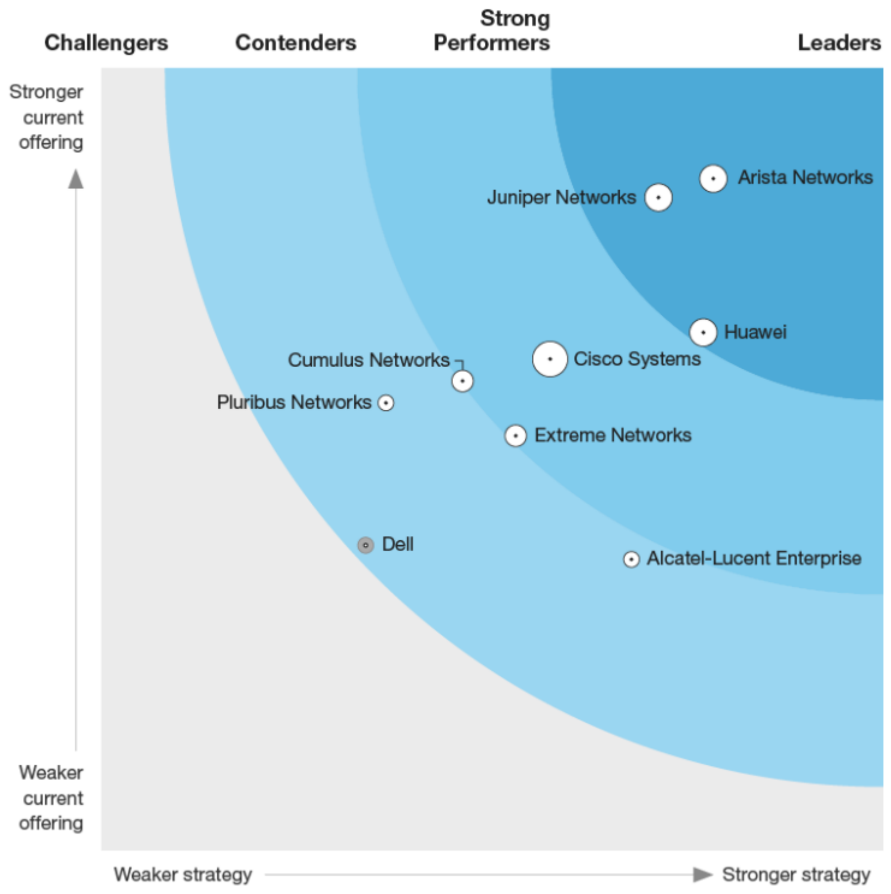
26+
New Products

1
Operating System

Gartner ™:
Magic Quadrant for Data Center Networking, 03 July 2017

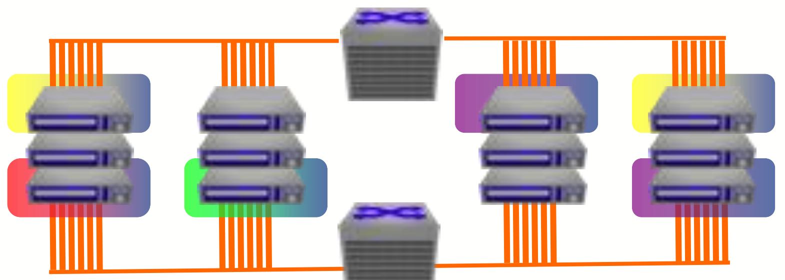


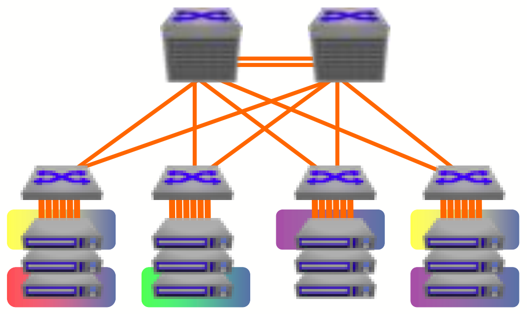
The Forrester Wave™: Hardware Platforms For Software-Defined Networking, Q1 2018

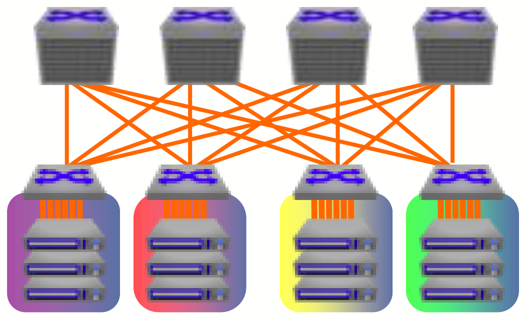


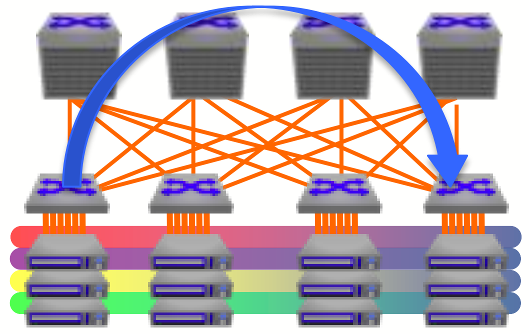
The Software Defined Data Centre

Vertically integrated, proprietary stacks	Open technologies, maximum generalization
Infrastructure specific to specific apps	Applications abstracted from infrastructure
Vendor lock-in, Forklift refreshes	Best-of-breed, continuous innovation
Multiple management domains	Homogenous, universal automation
Complex and custom architectures	Simple, repeatable and scalable architectures


Spline™


Layer 2 / MLAG


Layer 3 / ECMP

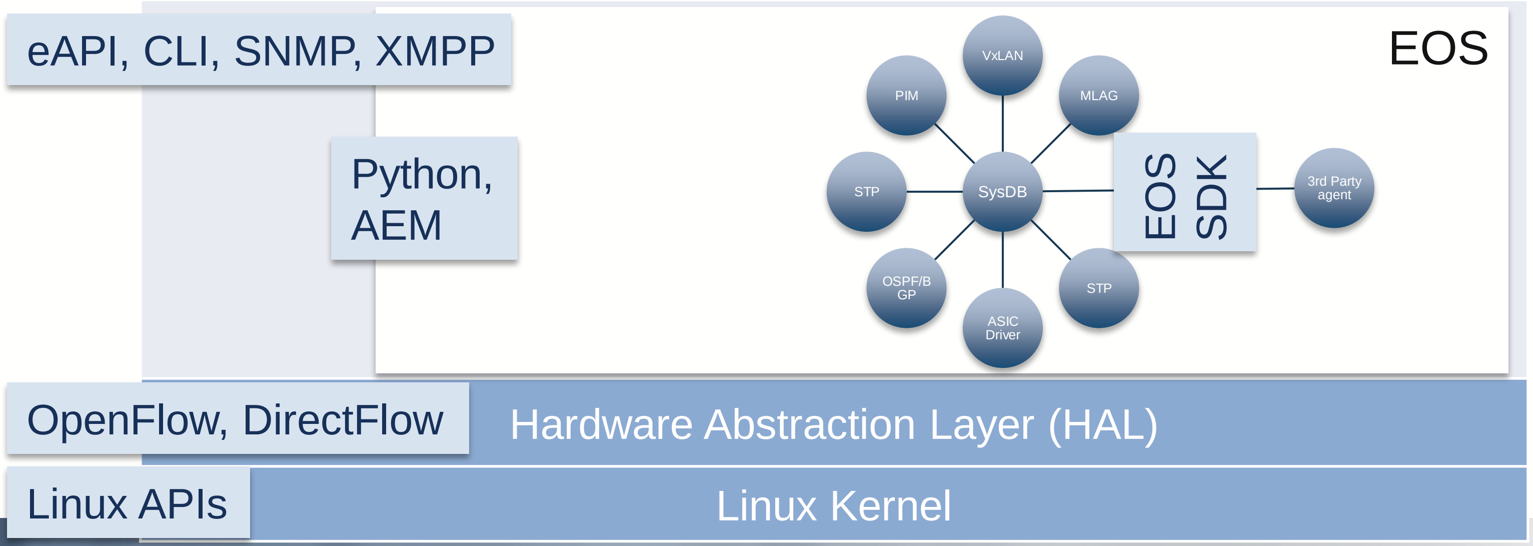

L2 over Layer 3 VXLAN

ARISTA

EOS: Software Driven Foundation Like No Other

EOS Network Services

Network Virtualization VxLAN gateway, VMTracer for VxLAN	**SDN Controller Integration** OpenStack, VMWare NSX, Nuage VSP, Plumgrid, OpenDaylight
Application and Network Visibility LANZ, Network Tracers	**Automation** ZTP, ZTR, DevOps, (Chef, Puppet, Ansible)
Network Telemetry DANZ, Tap Aggregation, sFlow, Splunk Forwarder, Timestamping	**Network Operations** Smart System Upgrade, Configuration & Image Management



ARISTA

Approaches to Cloud Network Automation

Customer Spectrum

D.I.Y.	**DevOps**	**Turnkey**
Custom development for integration	Leveraging existing tools (Puppet, Chef, etc)	Limited solutions today, focus on legacy models
Software development resources & approach	Integration & customization via scripts	Customers need turnkey solution for cloud-automation!
ARISTA	ARISTA	ARISTA
EOS SDK **eAPI**	**EOS DevOps Toolkit** **Ruby/Python Object Model**	**CloudVision**

DIY

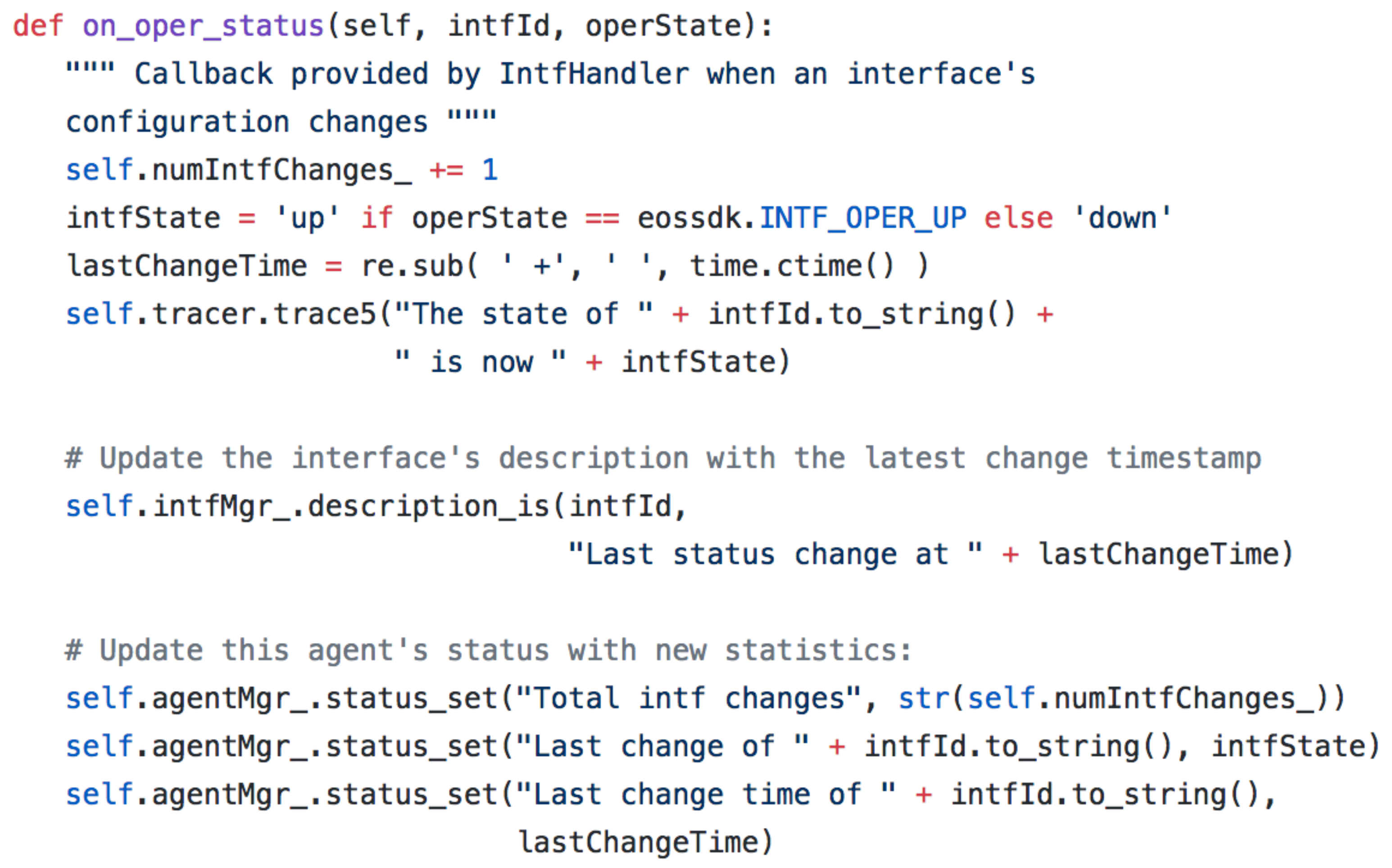
DevOps

Turnkey

DIY Approach: EOS Software Development Kit

- EOS SDK let’s you write native apps for your switch.
- Well documented and versioned, available on GitHub
- High performance, so agents can operate on large scale networks
- Low-level integration, so agents can receive notifications instantly
- APIs in both Python and C++
- Over 35 modules, including access to interfaces, the FIB, route configuration, the MAC and ARP tables, LLDP, system information, hardware capacity monitoring, ACLs, policy maps, etc.
- Can develop agents on any 32bit Linux system, no heavy dependencies needed.

DIY Approach: SDK InterfaceMonitor.py



```
def on_oper_status(self, intfId, operState):
   """ Callback provided by IntfHandler when an interface's
   configuration changes """
   self.numIntfChanges_ += 1
   intfState = 'up' if operState == eossdk.INTF_OPER_UP else 'down'
   lastChangeTime = re.sub( ' +', ' ', time.ctime() )
   self.tracer.trace5("The state of " + intfId.to_string() +
                      " is now " + intfState)

   # Update the interface's description with the latest change timestamp
   self.intfMgr_.description_is(intfId,
                                "Last status change at " + lastChangeTime)

   # Update this agent's status with new statistics:
   self.agentMgr_.status_set("Total intf changes", str(self.numIntfChanges_))
   self.agentMgr_.status_set("Last change of " + intfId.to_string(), intfState)
   self.agentMgr_.status_set("Last change time of " + intfId.to_string(),
                             lastChangeTime)
```

See full example at https://github.com/aristanetworks/EosSdk/blob/master/examples/InterfaceMonitor.py

DIY

DevOps

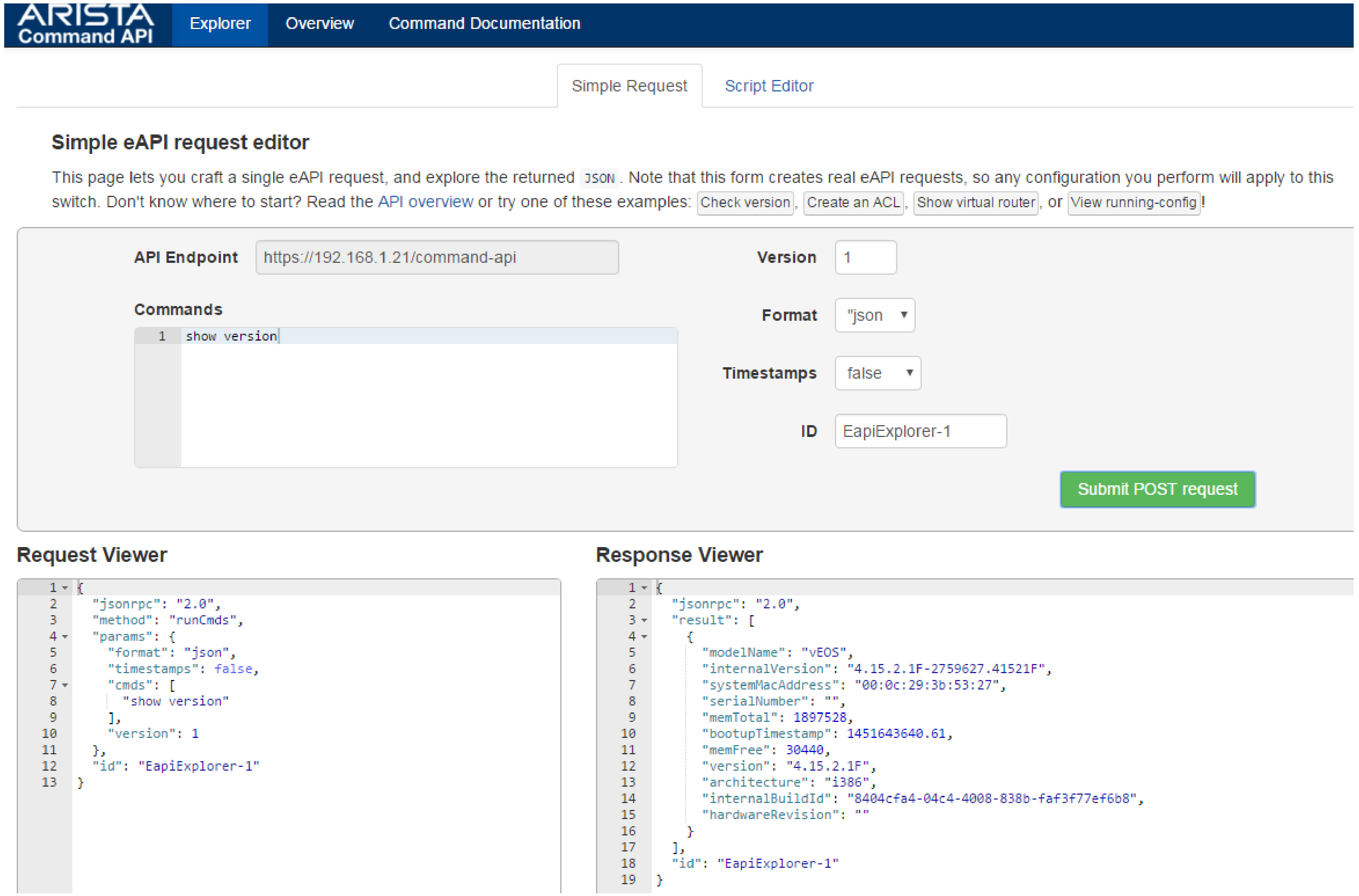
Turnkey

DIY Approach: eAPI

- eAPI is a simple method of remotely interacting with an Arista switch without screen scraping
- HTTP or HTTPS and uses JSON (JavaScript Object Notation)
- Full configuration supported – many show commands supported
- eAPI allows CLI Commands to be issued remotely
- eAPI returns the output in a programmable-friendly format (JSON) and generally in key-value pairs
- Useful when you need to automatically read or control a remote switch (automation)!

ARISTA

DIY Approach: eAPI Web Interface



Simple eAPI request editor

This page lets you craft a single eAPI request, and explore the returned `JSON`. Note that this form creates real eAPI requests, so any configuration you perform will apply to this switch. Don't know where to start? Read the API overview or try one of these examples: Check version, Create an ACL, Show virtual router, or View running-config!

API Endpoint	https://192.168.1.21/command-api	**Version**	1
		Format	"json
		Timestamps	false
		ID	EapiExplorer-1

Commands

```
show version
```

Request Viewer

```
{
  "jsonrpc": "2.0",
  "method": "runCmds",
  "params": {
    "format": "json",
    "timestamps": false,
    "cmds": [
      "show version"
    ],
    "version": 1
  },
  "id": "EapiExplorer-1"
}
```

Response Viewer

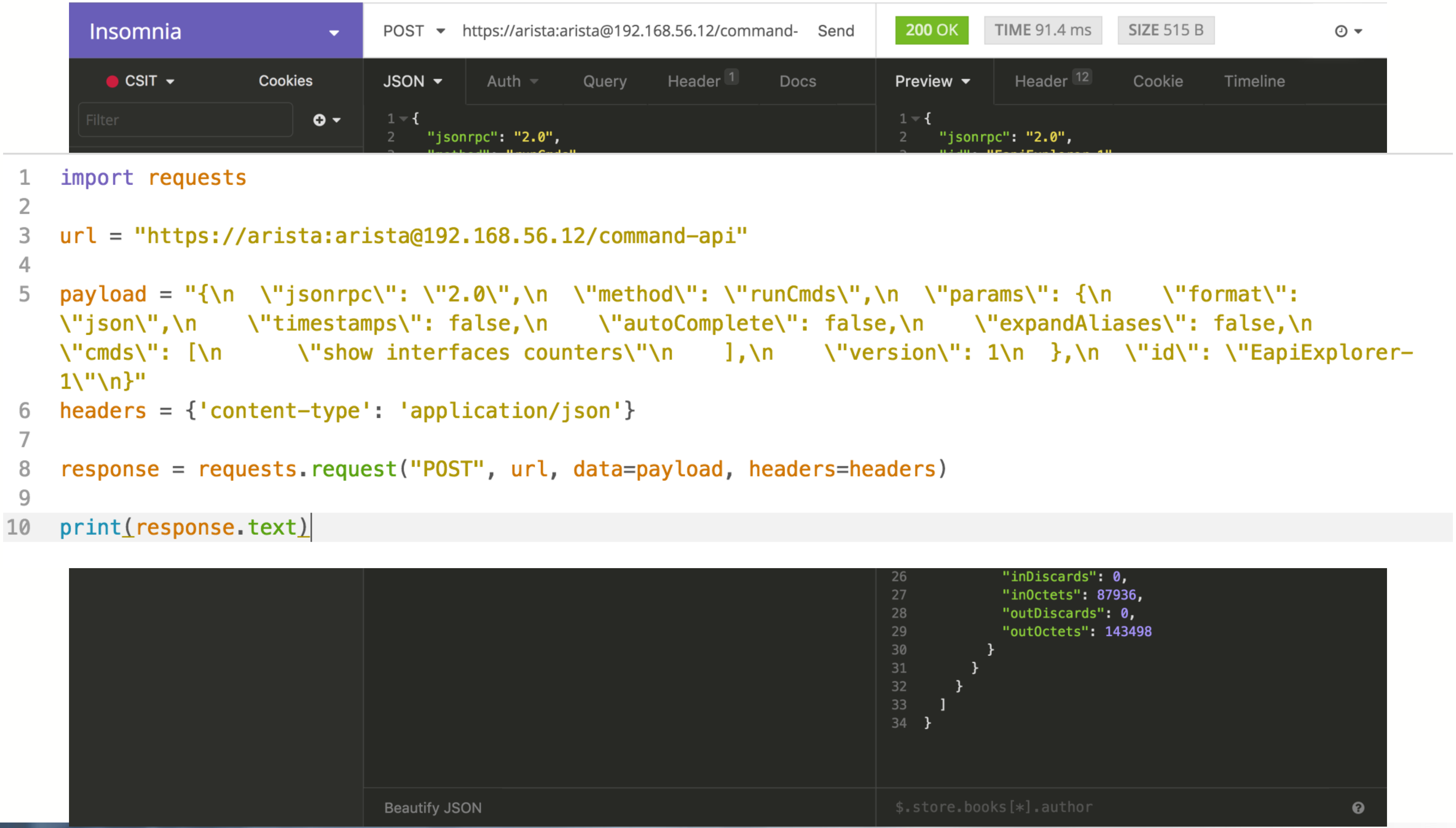
```
{
  "jsonrpc": "2.0",
  "result": [
    {
      "modelName": "vEOS",
      "internalVersion": "4.15.2.1F-2759627.41521F",
      "systemMacAddress": "00:0c:29:3b:53:27",
      "serialNumber": "",
      "memTotal": 1897528,
      "bootupTimestamp": 1451643640.61,
      "memFree": 30440,
      "version": "4.15.2.1F",
      "architecture": "i386",
      "internalBuildId": "8404cfa4-04c4-4008-838b-faf3f77ef6b8",
      "hardwareRevision": ""
    }
  ],
  "id": "EapiExplorer-1"
}
```

DIY

DevOps

Turnkey

DIY Approach: eAPI Example



```
{
  "jsonrpc": "2.0",
```

```
{
  "jsonrpc": "2.0",
```

```python
import requests

url = "https://arista:arista@192.168.56.12/command-api"

payload = "{\n  \"jsonrpc\": \"2.0\",\n  \"method\": \"runCmds\",\n  \"params\": {\n    \"format\": \"json\",\n    \"timestamps\": false,\n    \"autoComplete\": false,\n    \"expandAliases\": false,\n    \"cmds\": [\n      \"show interfaces counters\"\n    ],\n    \"version\": 1\n  },\n  \"id\": \"EapiExplorer-1\"\n}"
headers = {'content-type': 'application/json'}

response = requests.request("POST", url, data=payload, headers=headers)

print(response.text)
```

```
                "inDiscards": 0,
                "inOctets": 87936,
                "outDiscards": 0,
                "outOctets": 143498
            }
          }
        }
      ]
    }
```

DevOps Approach

What is DevOps?

A culture, movement or practice that emphasizes the collaboration and communication of both software developers and other information-technology (IT) professionals while **automating the process** of software delivery and **infrastructure changes**.

GitHub	**Revision control** Git, backups, auditing, peer review, and access-control, blame ☺
	Continuous Integrations (CI) automated testing and workflows
ANSIBLE CHEF puppet	**Configuration management tools** like Ansible / Chef / Puppet **Continuous, consistent, auditing** Eg: New network modules in Ansible 2.1: eos_command, eos_eapi, eos_config
servicenow	**Change Control Management** schedule, authorize, track changes, approval workflows

DIY

DevOps

Turnkey

DevOps Benefits

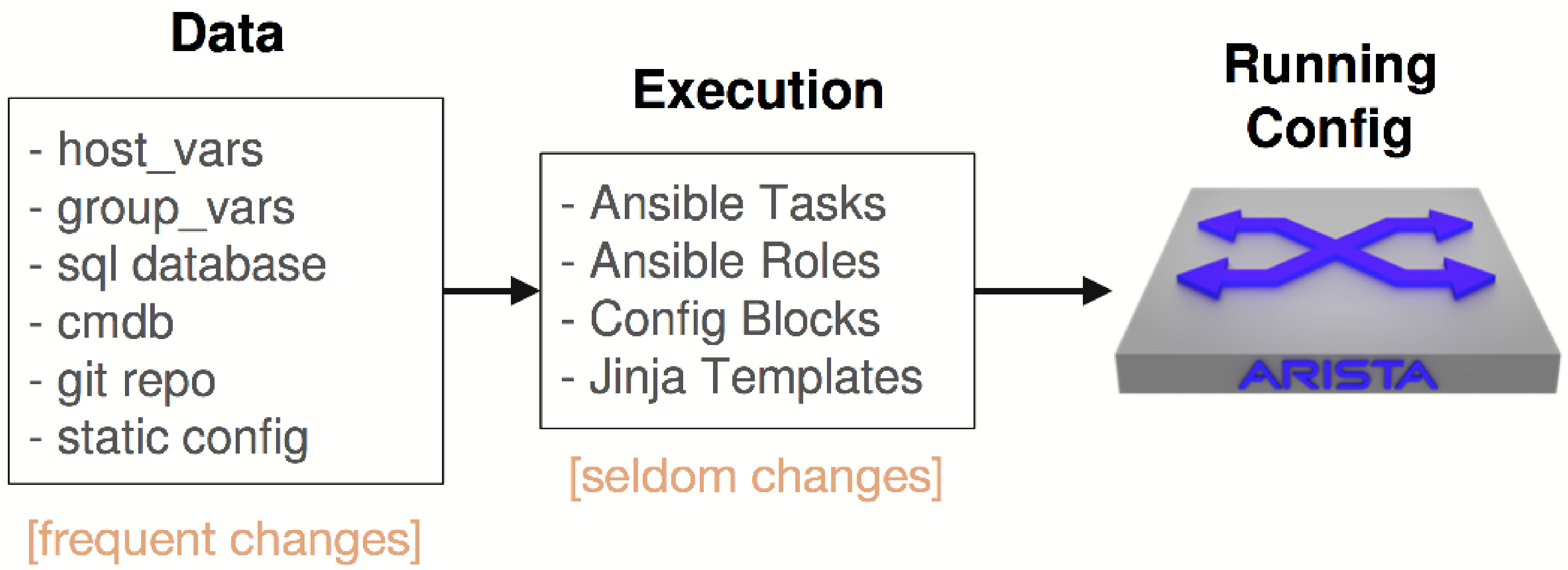
- Culture
- Change Management
- Automated Testing
- Accelerated deployment
- Infrastructure as code
- Security & Compliance Audits
- Monitoring
- Increased availability
- Fail fast, fail often, learn from your mistakes
- Get your life back - Spend more time doing architecture… and less adding VLANs!

ARISTA

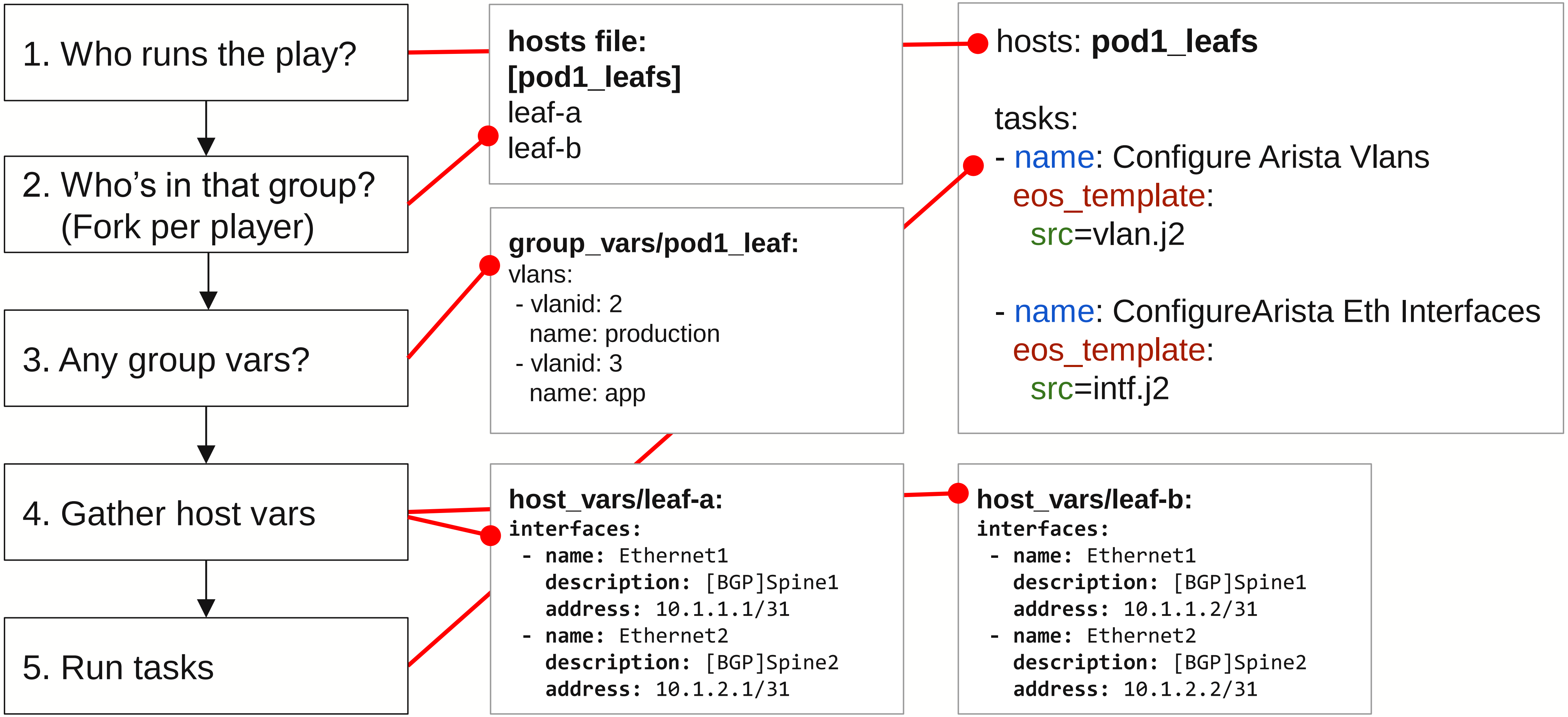
Configuration Management

	ANSIBLE	puppet	CHEF	SALTSTACK
EOS Integration	Built-in	Forge modules	Cookbook	Minions
EOS Agent	None	EOS SWIX	el6 RPM	Yes
Architecture	Push	Pull	Pull	Continuous
Transport	SSH/SSL	SSL	SSL	ZeroMQ
Language	Python	Ruby	Ruby	Python Napalm
Community	Huge	4000	3000	Growing
Price	Free/Paid Ansible Tower	Free/Paid Puppet Enterprise	Free/Paid Chef Automate	Free/Paid Saltstack Enterprise

DevOps Approach: Ansible Workflow



DevOps Approach: Running an Ansible Playbook



hosts file:
[pod1_leafs]
leaf-a
leaf-b

group_vars/pod1_leaf:

```
vlans:
 - vlanid: 2
   name: production
 - vlanid: 3
   name: app
```

host_vars/leaf-a:

```
interfaces:
  - name: Ethernet1
    description: [BGP]Spine1
    address: 10.1.1.1/31
  - name: Ethernet2
    description: [BGP]Spine2
    address: 10.1.2.1/31
```

```
hosts: pod1_leafs

tasks:
- name: Configure Arista Vlans
  eos_template:
    src=vlan.j2

- name: ConfigureArista Eth Interfaces
  eos_template:
    src=intf.j2
```

host_vars/leaf-b:

```
interfaces:
  - name: Ethernet1
    description: [BGP]Spine1
    address: 10.1.1.2/31
  - name: Ethernet2
    description: [BGP]Spine2
    address: 10.1.2.2/31
```

DIY

DevOps

Turnkey

Ansible Example: Adding a VLAN

`---`	
`- name: Add a VLAN`	Play name
`hosts: 172.16.198.130`	Host to run against
`gather_facts: no`	Fetch configuration prior to running?
`connection: local`	What device to connect from
`vars:`	Defines variable section
`provider:`	Creates a new variable set called “provider”
`host: "{{ ansible_host }}"`	Built-in variable for the host Ansible is being run from
`username: "admin"`	Super top secret username
`password: "arista"`	Super top secret password
`authorize: yes`	Enable before issuing commands?
`transport: cli`	Transport – CLI or eAPI
`tasks:`	Defines tasks section
`- eos_config:`	Specifies the “eos_config” module
`lines:`	Begins configuration section
`- name foo`	Actual configuration to be issued
`parents: vlan 500`	The parent configuration section
`provider: "{{ provider }}"`	Tells this task to use the previous slides info to connect

```
---
- name: Add a VLAN
  hosts: 172.16.198.130
  gather_facts: no
  connection: local
 vars:
   provider:
     host: "{{ ansible_host }}"
     username: "admin"
     password: "arista"
     authorize: yes
     transport: cli

 tasks:
     - eos_config:
         lines:
          - name foo
         parents: vlan 500
         provider: "{{ provider }}"
```

ARISTA

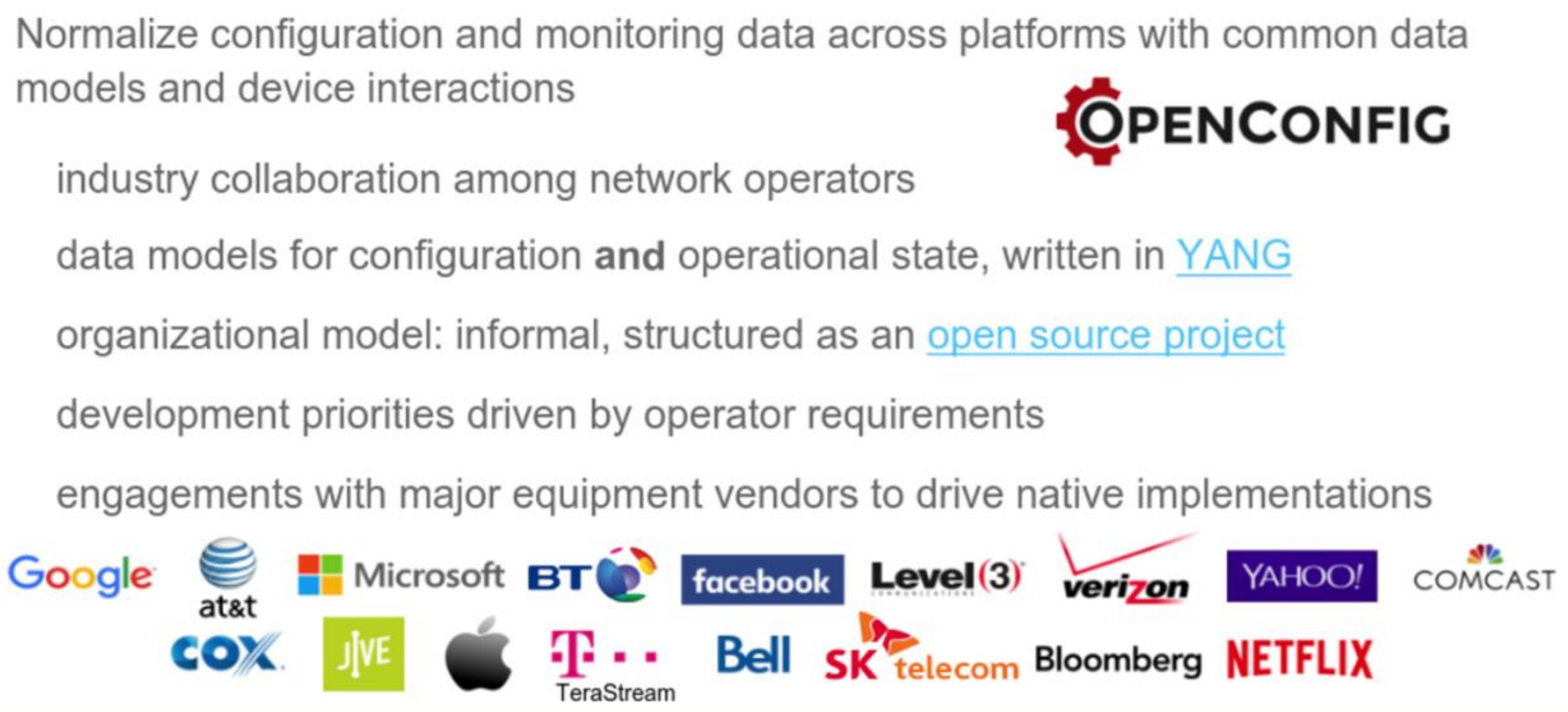
Starting a DevOps Culture

- Start with ad hoc commands or simple one-liners
- Show value to the organization by demonstrating quicker provisioning times with fewer errors
- Begin conversations about treating infrastructure as code
- Find your friendly developers/QA teams and pair up with them – we find that the most successful organizations will pair a developer with a network resource
- Remember that this is also a huge cultural change that requires buy in from everyone – top down

DevOps Automation Can Be Difficult

- Different vendors have different CLI’s
- Different vendors have different API’s
- Different vendors use different modules for Ansible, Saltstack, Chef, Puppet, etc.
- Different vendors return the same data in different formats (JSON, XML, etc)

OpenConfig: Open Data Models for Network Management


Normalize configuration and monitoring data across platforms with common data models and device interactions

- industry collaboration among network operators
- data models for configuration **and** operational state, written in YANG
- organizational model: informal, structured as an open source project
- development priorities driven by operator requirements
- engagements with major equipment vendors to drive native implementations

DIY

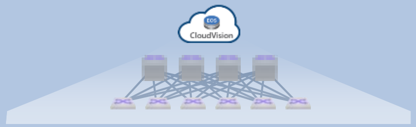
DevOps

Turnkey

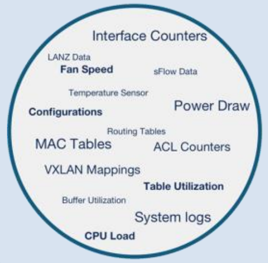
Arista CloudVision

Overlay Integration

API’s for simplified network integration to a best of breed ecosystem



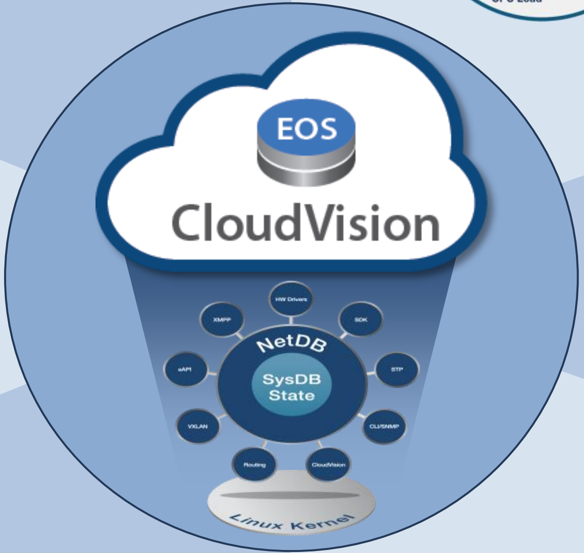
Telemetry & Analytics



Real-time state streaming and historical analytics

Automated Deployments

Initial and ongoing provisioning network-wide



Macro-Segmentation Services (MSS)

Service insertion for securing today’s cloud networks

Change Controls

Network-wide upgrades, rollback and snapshots. Compliance and Bug Visibility

DANZ TAP Aggregation

Purpose-built to capture traffic at cloud scale and speed

ARISTA

DIY

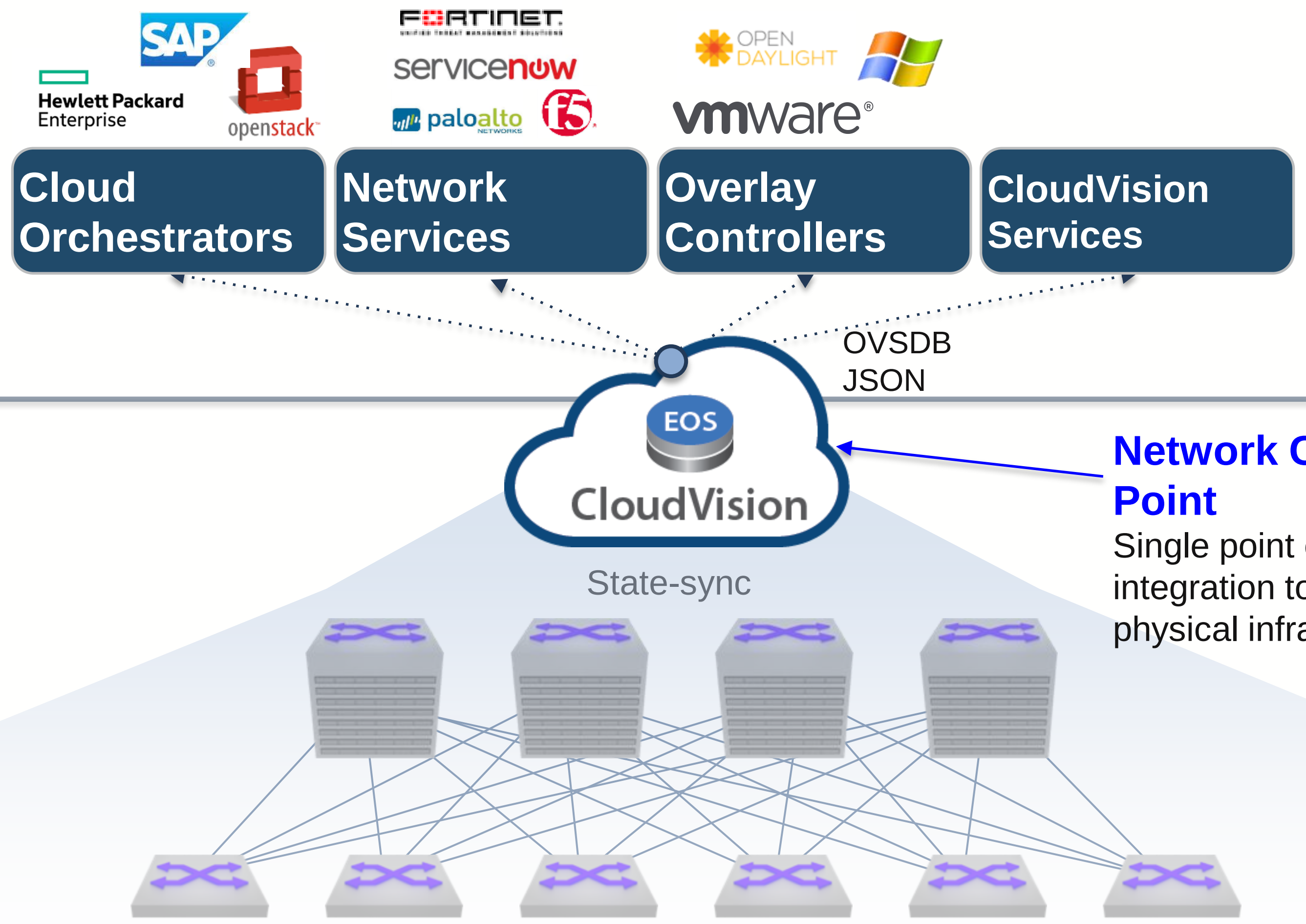
DevOps

Turnkey

Integration Point to the Underlay

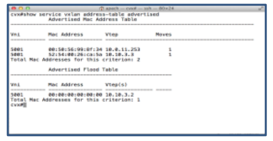
Points of Integration

Network Layer



Network Control Point

Single point of integration to the physical infrastructure



```
cvx#show service vxlan address-table advertised
              Advertised Mac Address Table
--------------------------------------------------------------------------------

Vni          Mac Address        Vtep               Moves
------------ ------------------ ------------------------------------------------

5001         00:50:56:99:0f:34 10.0.11.253          1
5001         52:54:00:26:ca:5a 10.10.3.3            1
Total Mac Addresses for this criterion: 2

              Advertised Flood Table
--------------------------------------------------------------

Vni          Mac Address        Vtep(s)
------------ ------------------ ----------------- ------
5001         00:00:00:00:00:00 10.10.3.2
Total Mac Addresses for this criterion: 1
cvx#
```

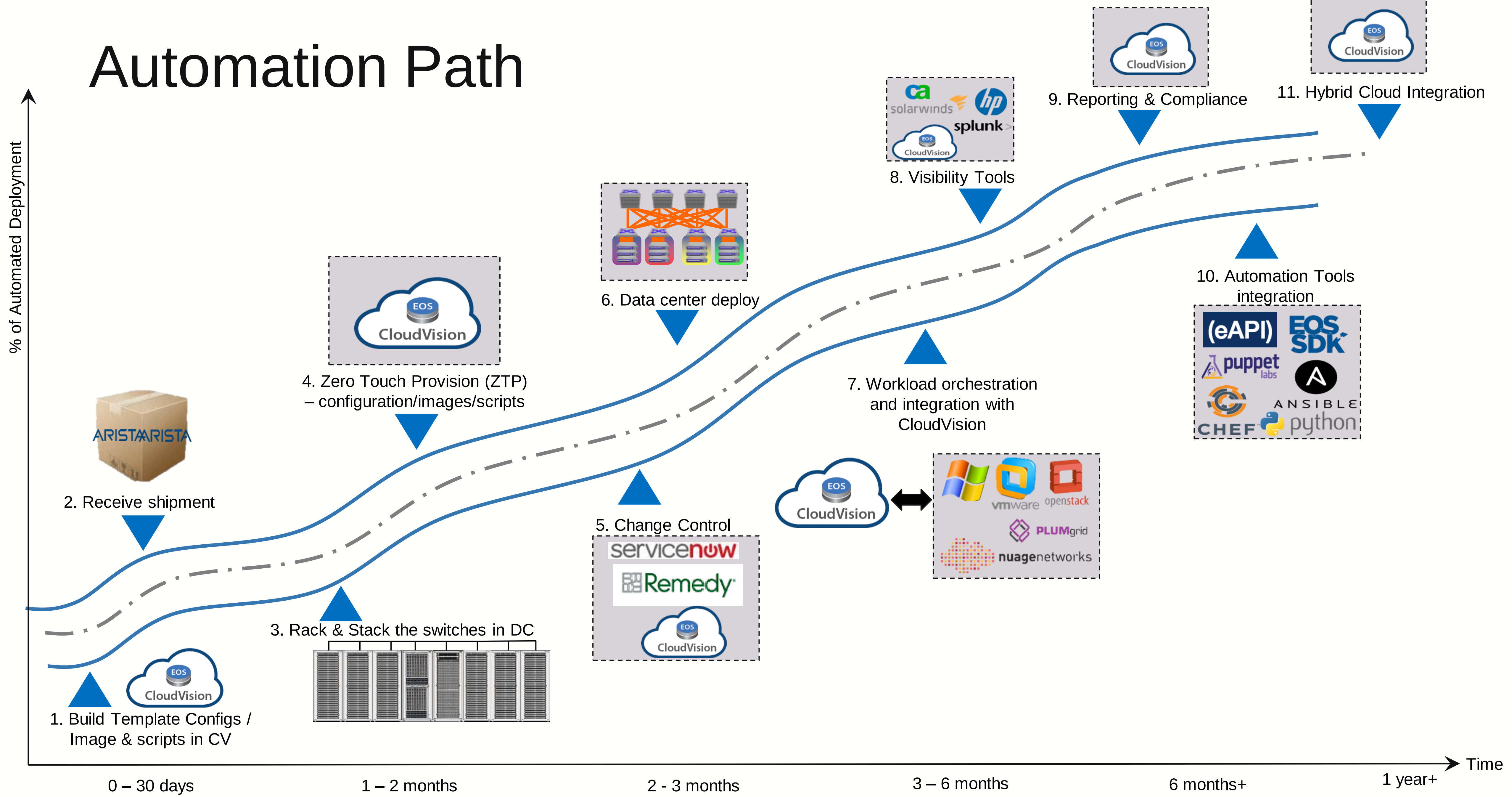
CLI

Web-based GUI

Platform for Automation and Visibility across the Network

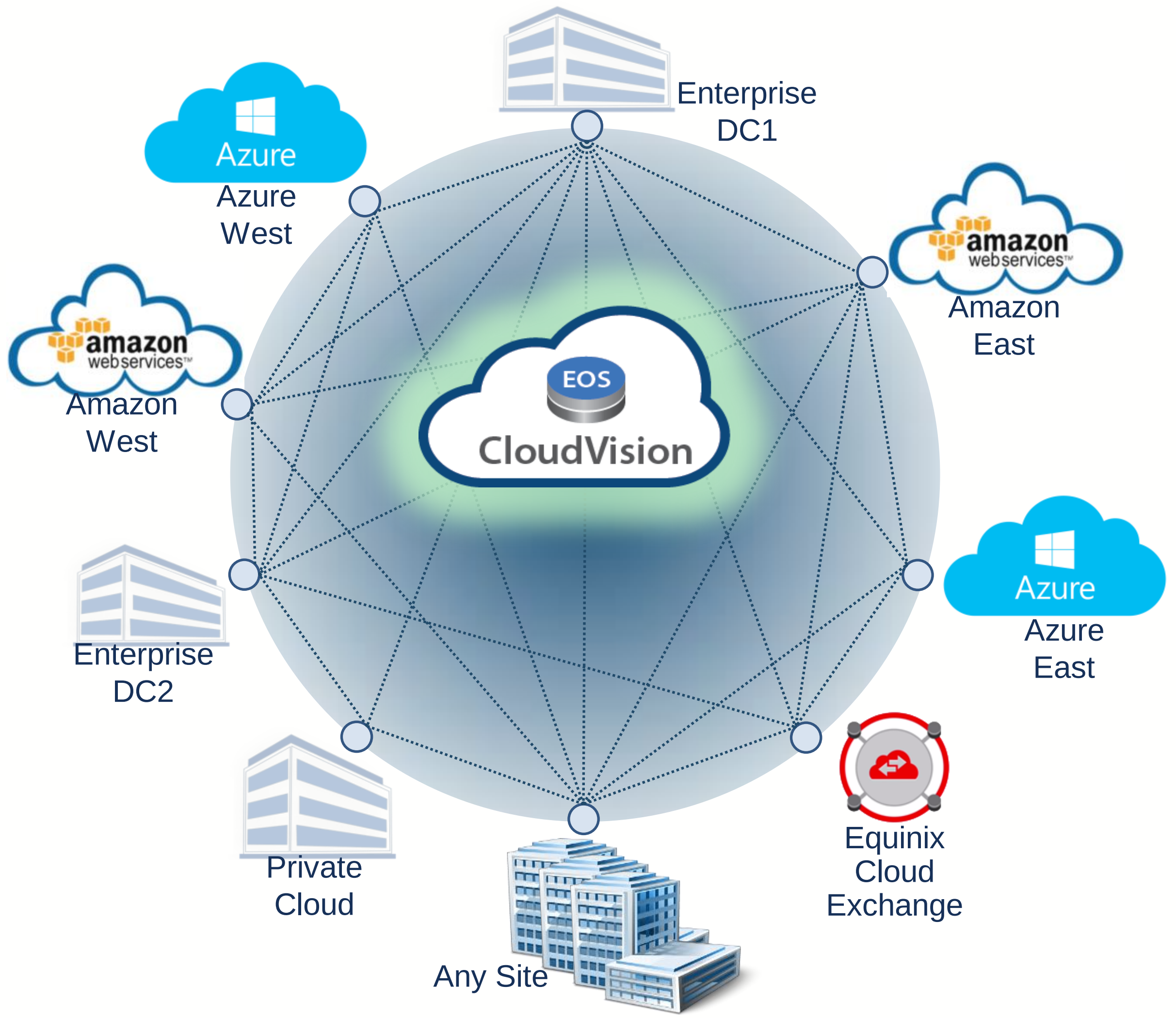
ARISTA

Automation Path



% of Automated Deployment
Time
1. Build Template Configs / Image & scripts in CV
2. Receive shipment
3. Rack & Stack the switches in DC
4. Zero Touch Provision (ZTP) – configuration/images/scripts
5. Change Control
6. Data center deploy
7. Workload orchestration and integration with CloudVision
8. Visibility Tools
9. Reporting & Compliance
10. Automation Tools integration
11. Hybrid Cloud Integration
EOS CloudVision
ARISTA
servicenow
Remedy
ca
solarwinds
hp
splunk
vmware
openstack
PLUMgrid
nuagenetworks
(eAPI)
EOS SDK
puppet labs
ANSIBLE
CHEF
python
0 – 30 days
1 – 2 months
2 - 3 months
3 – 6 months
6 months+
1 year+

CloudVision for the Hybrid Cloud



- **Zero Touch Provisioning**: Quickly spin up routing services
- **Automated Change Management:** Streamlined NetOps across clouds
- **State Streaming**: Real-time telemetry across any EOS use-case
- **Analytics Engine:** for historic event correlation and anomaly detection
- **Visualization Apps:** common dashboard for advanced telemetry

ARISTA

Thank You

www.arista.com

ARISTA