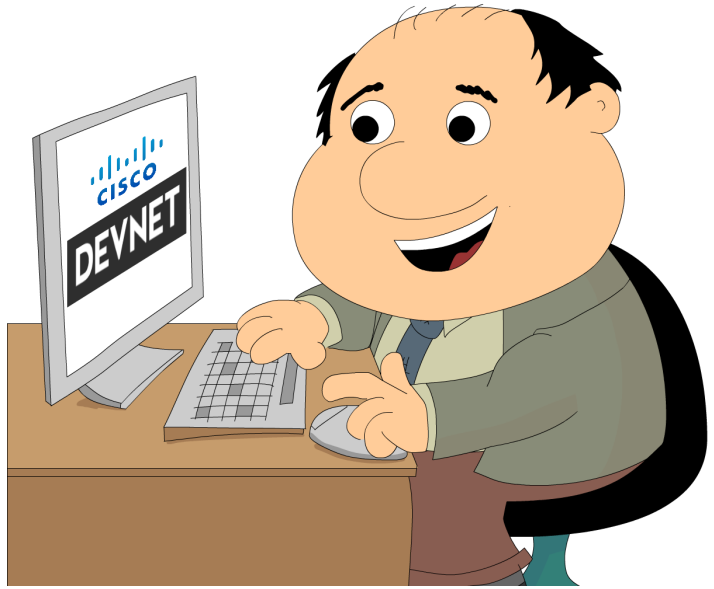
Data Formats: Understanding and using JSON, XML and YAML

A Network Programmability Basics Presentation

Hank Preston, ccie 38336
Developer Evangelist
@hfpreston

Network Programmability Basics Modules

- Introduction: How to be a Network Engineer in a Programmable Age
- **Programming Fundamentals**
- Network Device APIs
- Network Controllers
- Application Hosting and the Network
- NetDevOps



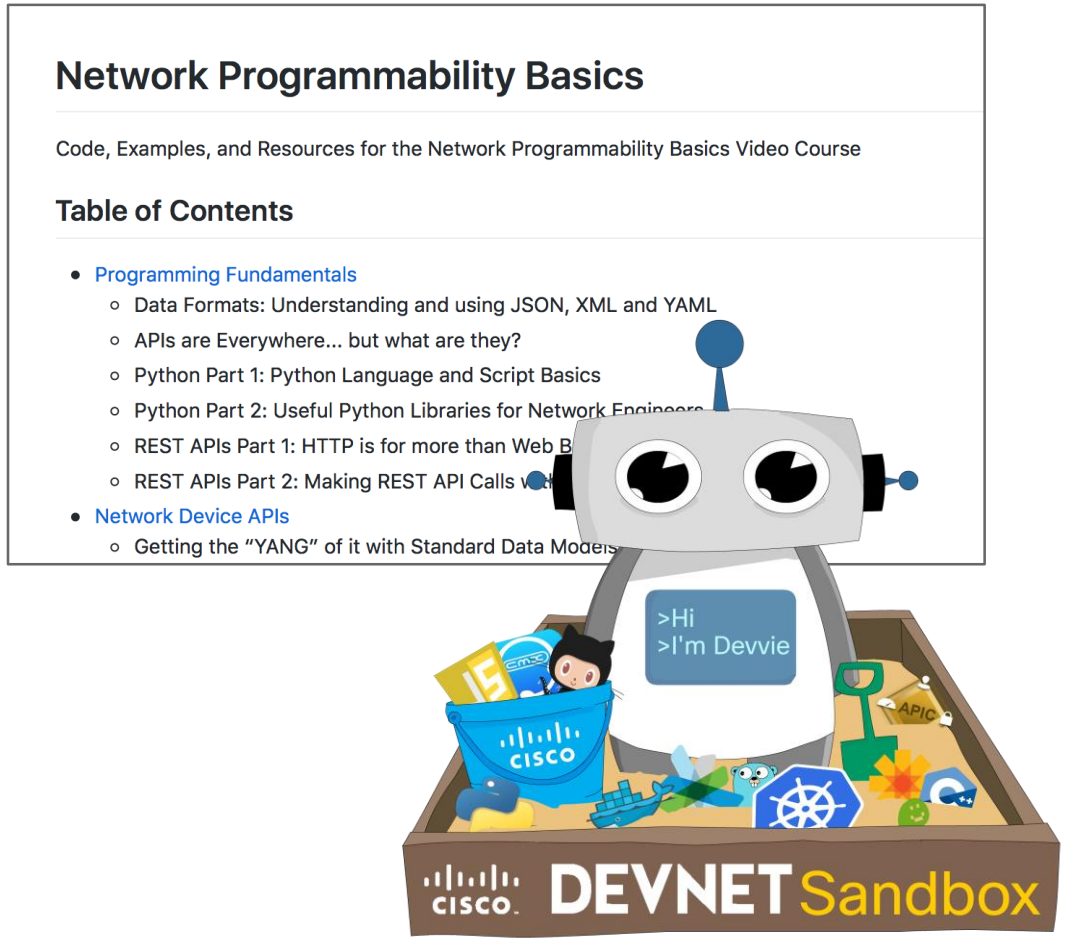
Network Programmability Basics: The Lessons

Module: Programming Fundamentals

- Data Formats: Understanding and using JSON, XML and YAML
- APIs are Everywhere… but what are they?
- REST APIs Part 1: HTTP is for more than Web Browsing
- REST APIs Part 2: Making REST API Calls with Postman
- Python Part 1: Python Language and Script Basics
- Python Part 2: Working with Libraries and Virtual Environments
- Python Part 3: Useful Python Libraries for Network Engineers

Code and Develop Along

- Get the Code!
  - github.com/CiscoDevNet/netprog_basics
- Setup Lab Prerequisites
  - Each lab includes a README with details
- Access to Infrastructure
  - DevNet Sandbox
  - Specifics in lab README



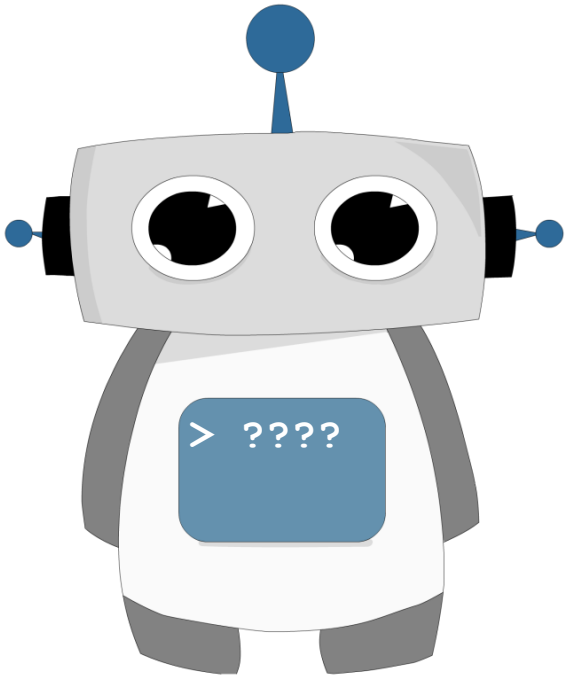
Topics to Cover

- Importance of a Data Format
- Common Data Formats in Programming
- Demystify XML
- Breakdown JSON
- Simplify YAML

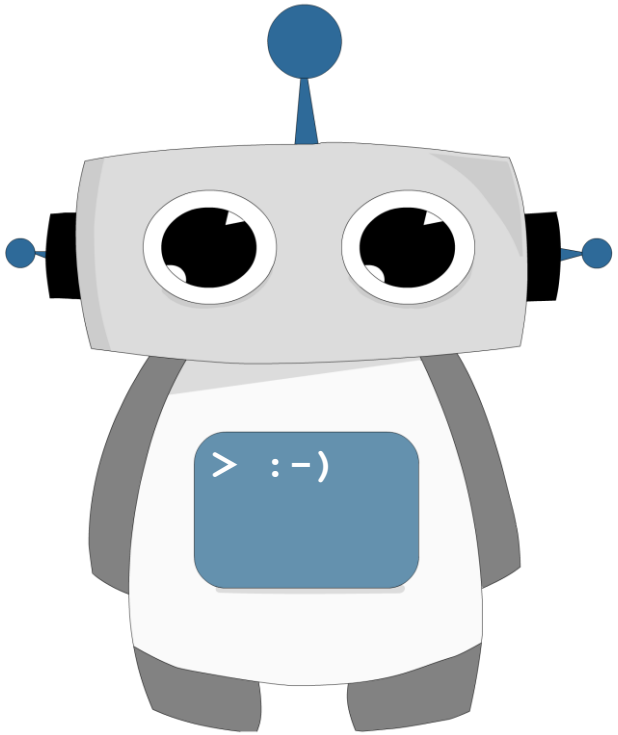
Importance of a Data Format

Know Your Audience

```
Interface              IP-Address      OK? Method Status                Protocol
GigabitEthernet1       10.0.2.15       YES DHCP   up                    up
GigabitEthernet2       172.16.0.2      YES manual up                    up
GigabitEthernet3       172.17.0.1      YES manual up                    up
```



Know Your Audience




```
{
    "ietf-interfaces:interfaces": {
        "interface": [
            {
                "name": "GigabitEthernet2",
                "description": "Wide Area Network",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "172.16.0.2",
                            "netmask": "255.255.255.0"
                        }
                    ]
                }
            },
            {
                "name": "GigabitEthernet3",
                "description": "Local Area Network",
                "type": "iana-if-type:ethernetCsmacd",
                "enabled": true,
                "ietf-ip:ipv4": {
                    "address": [
                        {
                            "ip": "172.17.0.1",
                            "netmask": "255.255.255.0"
                        }
                    ]
                }
            }
        ]
    }
}
```

Common Data Formats in Programming

Common Data Formats in Programming

JSON

```
{
    "ietf-interfaces:interface": {
        "name": "GigabitEthernet2",
        "description": "Wide Area Network”,
        "enabled": true,
        "ietf-ip:ipv4": {
            "address": [
                {
                    "ip": "172.16.0.2",
                    "netmask": "255.255.255.0"
                }
            ]
        }
    }
}
```

XML

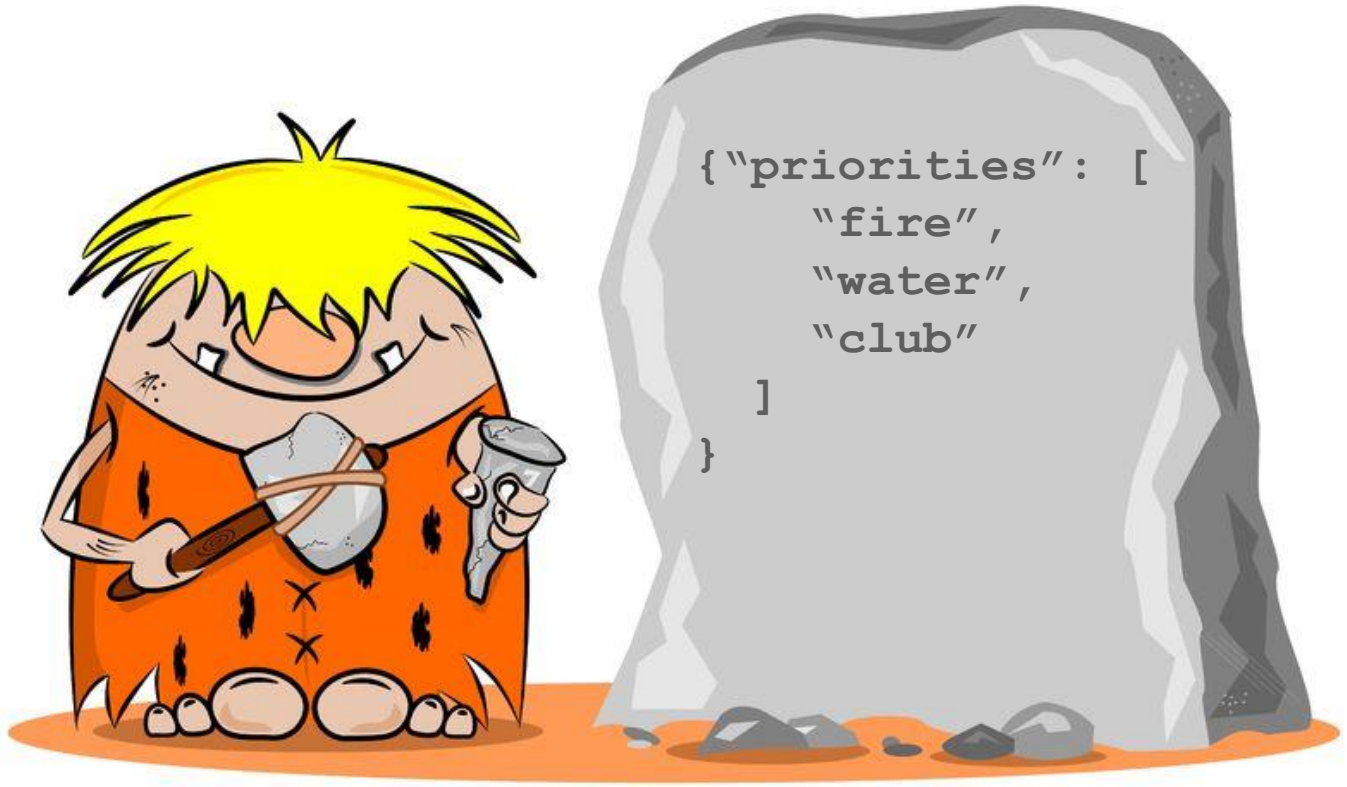
```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
  <name>GigabitEthernet2</name>
  <description>Wide Area Network</description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

YAML

```
---
ietf-interfaces:interface:
  name: GigabitEthernet2
  description: Wide Area Network
  enabled: true
  ietf-ip:ipv4:
    address:
    - ip: 172.16.0.2
      netmask: 255.255.255.0
```

Common Elements in a Data Format

- Format Syntax
- Objects Representation
- Key / Value Notation
  - Values can be objects, lists, strings, numbers, boolean
- Arrays or List Notation



"Key" : "Value"

- "Key" identifies/labels a set of data
- Left side of the colon
- Inside of "quotes"

- "Value" is the Data
- Right side of colon
- Can be:
  - String
  - Integer
  - Array/List
  - Bool
  - Object

```
{
  "name": "GigabitEthernet2",
  "description": "Wide Area Network”,
  "enabled": true
}
```

Demystify XML

XML- eXtensible Markup Language.

A human readable data structure that applications use to store, transfer, and read data.

```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
  <name>GigabitEthernet2</name>
  <description>
    Wide Area Network
  </description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

XML

- Designed for the Internet
- Schema or namespace defines data model
- **<tags></tags>** surround **elements** for structure and layout
- Key/Value representation
  - **<key>value</key>**
- Whitespace not significant

```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
  <name>GigabitEthernet2</name>
  <description>
    Wide Area Network
  </description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

XML Object

- A related set of data surrounded by **<tags></tags>**
- An object can contain other objects or data entries
- **<key>value</key>** contained within the object tags

```
<?xml version="1.0" encoding="UTF-8" ?>
<interface xmlns="ietf-interfaces">
  <name>GigabitEthernet2</name>
  <description>
    Wide Area Network
  </description>
  <enabled>true</enabled>
  <ipv4>
    <address>
      <ip>172.16.0.2</ip>
      <netmask>255.255.255.0</netmask>
    </address>
  </ipv4>
</interface>
```

XML List

- List of data
  - Can be composed of XML objects
- Repeated instances of **<tags></tags>** for each element

```
<?xml version="1.0" encoding="UTF-8" ?>
<addresses>
  <ip>172.16.0.2</ip>
  <netmask>255.255.255.0</netmask>
</addresses>
<addresses>
  <ip>172.16.0.3</ip>
  <netmask>255.255.255.0</netmask>
</addresses>
<addresses>
  <ip>172.16.0.4</ip>
  <netmask>255.255.255.0</netmask>
</addresses>
```

Breakdown JSON

JSON - JavaScript Object Notation

A human readable data structure that applications use to store, transfer, and read data.

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

JSON

- A data-interchange text format
- Notated with {} for objects, [] for arrays
- Key/Value representation
  - `"key": value`
- Whitespace not significant

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

JSON Object

- Data surrounded by { }
- An object can contain other objects or data entries
- Key/Value set separated by comma
  - No comma at the end!

```
{
  "ietf-interfaces:interface": {
    "name": "GigabitEthernet2",
    "description": "Wide Area Network",
    "enabled": true,
    "ietf-ip:ipv4": {
      "address": [
        {
          "ip": "172.16.0.2",
          "netmask": "255.255.255.0"
        }
      ]
    }
  }
}
```

JSON List

- List of data
  - Can be composed of JSON objects
- Notated with brackets
- Comma Separated

```
{
  "addresses": [
    {
      "ip": "172.16.0.2",
      "netmask": "255.255.255.0"
    },
    {
      "ip": "172.16.0.3",
      "netmask": "255.255.255.0"
    },
    {
      "ip": "172.16.0.4",
      "netmask": "255.255.255.0"
    }
  ]
}
```

Simplify YAML

YAML – "YAML Ain't Markup Language"

A human readable data structure that applications use to store, transfer, and read data.

```
---
ietf-interfaces:interface:
  name: GigabitEthernet2
  description: Wide Area Network
  enabled: true
  ietf-ip:ipv4:
    address:
    - ip: 172.16.0.2
      netmask: 255.255.255.0
```

YAML

- Minimalist format commonly used for configuration files
- Whitespace indentation defines structure
  - *No commas*
- Key/Value representation
  - **key: value**

```
---
ietf-interfaces:interface:
  name: GigabitEthernet2
  description: Wide Area Network
  enabled: true
  ietf-ip:ipv4:
    address:
    - ip: 172.16.0.2
      netmask: 255.255.255.0
```

YAML Object

- Related set of data at the common indentation level under name
- An object can contain other objects or data entries
- **key**: **value** pairs left aligned

```
---
ietf-interfaces:interface:
  name: GigabitEthernet2
  description: Wide Area Network
  enabled: true
  ietf-ip:ipv4:
    address:
    - ip: 172.16.0.2
      netmask: 255.255.255.0
```

YAML List

- List of data
  - Can be composed of YAML objects
- Uses “-” character to indicate a list element

```
---
addresses:
- ip: 172.16.0.2
  netmask: 255.255.255.0
- ip: 172.16.0.3
  netmask: 255.255.255.0
- ip: 172.16.0.4
  netmask: 255.255.255.0
```

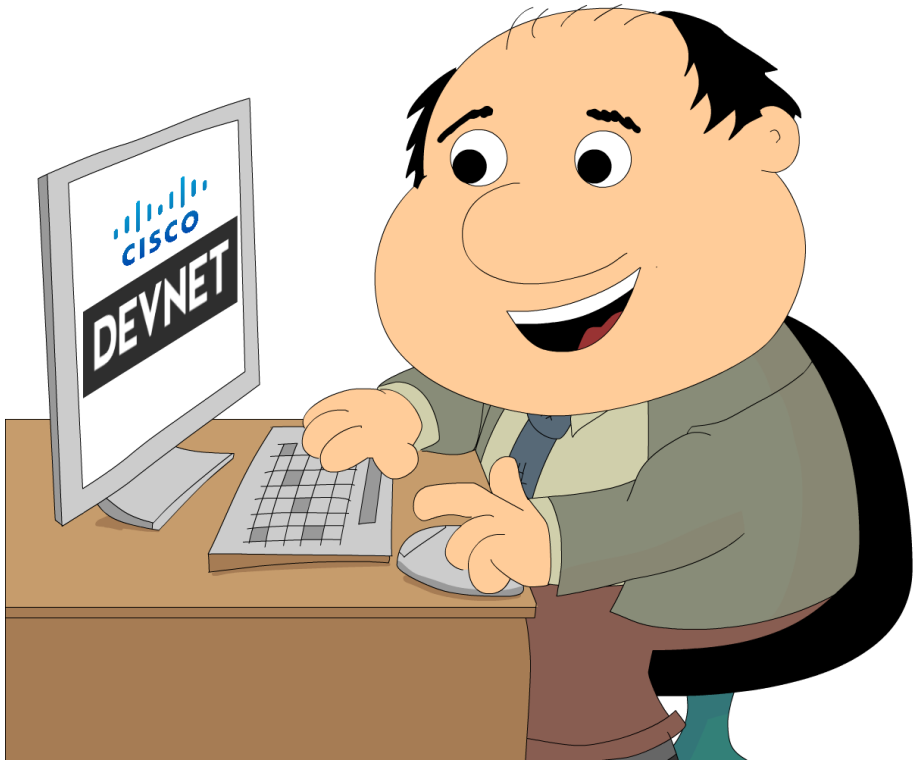
Summing up

Review

- Importance of the Audience
- Common data formats in programming
  - XML
  - JSON
  - YAML
- Data Formats are mostly interchangeable

Call to Action!

- Complete the full **Network Programmability Basics** Course
- Run the examples and exercises yourself!
  - Bonus Examples!
- Join DevNet for so much more!
  - Learning Labs
  - Development Sandboxes
  - Code Samples and API Guides



Got more questions? Come find me!

hapresto@cisco.com

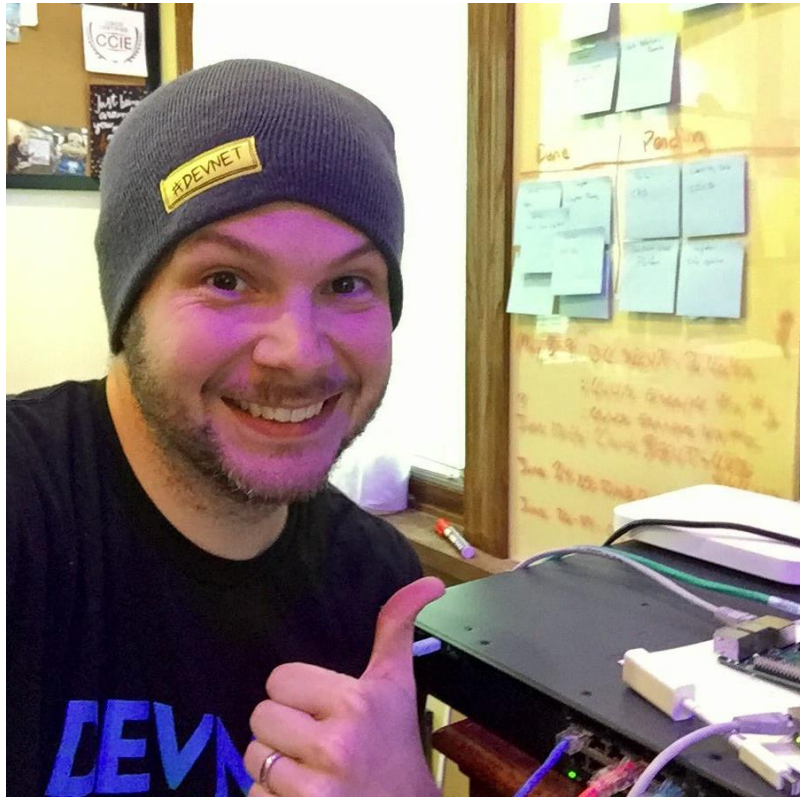
@hfpreston

http://github.com/hpreston

@CiscoDevNet

facebook.com/ciscodevnet/

http://github.com/CiscoDevNet




CISCO
DEVNET