



# DEVNET

# A Breakneck Journey into Network Service Based Automation in DevNet Sandbox

Hank Preston

Network Engineer writing code...

Twitter: @hfpreston

December 2019

- What is DevNet Sandbox?
- A Bit of History
- Our Data Center Project
- Planning Phase 1 Network Services
- Demos
- Some Lessons Learned
- What's Next in our NSO Journey







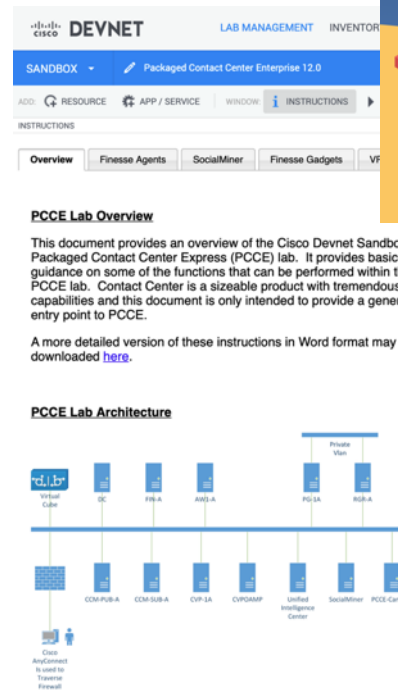
**What is DevNet Sandbox?**



# DevNet Sandbox

- **FREE** Development Lab as a Service from Cisco
- Access to labs with hardware and/or software from Cisco and partners
- Some Numbers...
  - 100,000+ users
  - 250+ years of reservations
  - Over 1 million API calls a month!

*We make innovation easy!*



The image shows the DevNet Sandbox landing page. At the top, it says 'Looking for a development lab?' and 'Dig in with DevNet Sandbox - technology packed Cisco labs - today! FREE with 24x7 access!'. There are two buttons: 'Learn more' and 'Get started with Sandbox'. Below this, there's a section titled 'EXPLORE TECHNOLOGIES' with icons for Networking, Data Center, Cloud, Security, IoT, Collaboration, Analytics & Automation, and Open Source. A 3D red box is placed over the Networking icon. Below the landing page, there's a network diagram showing a central 'VLAN 671' connected to various nodes. The nodes include 'Customer Voice Por...', 'CVP Operation Cons...', 'Progger 12.0\_b3d54...', 'Rogger 12.0\_c918c...', 'Cisco Unified Intelli...', 'AW1-A\_2c8e6b42', 'Social Miner 12.0\_1...', 'vCUBE\_b3617b34', and 'Finess 12.0\_62c8e4...'. Each node has a small icon and a status indicator (green dot). At the bottom, there's a terminal window showing the output of the lab setup process. The output includes: 'Sandbox setup finished successfully', '17:45:36 Just a moment - a few more things left to setup...', '17:45:54 Waiting 10 minutes for all systems to become fully ready', '17:55:55 Adding user doklinge on VPN, for POD 71, VLAN 671', '17:56:11 VPN CONNECTION INFORMATION', 'VPN Network: [redacted]', 'Username: [redacted]', 'Password: [redacted]', 'Sending email with VPN Credentials.', '17:56:15 Lab Setup Complete. Lab is ready for use!', and '17:56:18 Driver Complete.'



A photograph of a server room. In the foreground, there is a black chain-link fence that covers the entire view. Behind the fence, several server racks are visible, filled with electronic equipment and cables. The room is dimly lit, with some light coming from the top. The text "We build our own cloud..." is overlaid in the center in a bold, white, italicized font.

***We build our own cloud...***

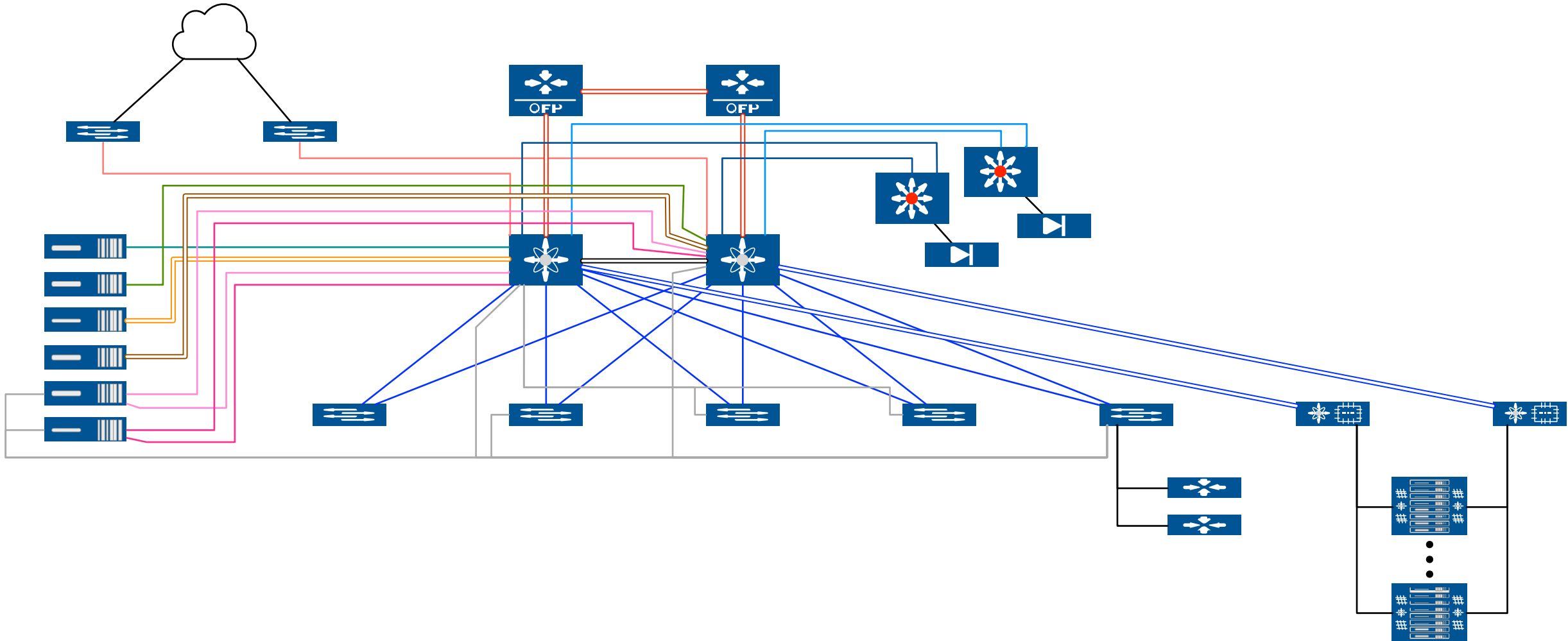




Where we came from?

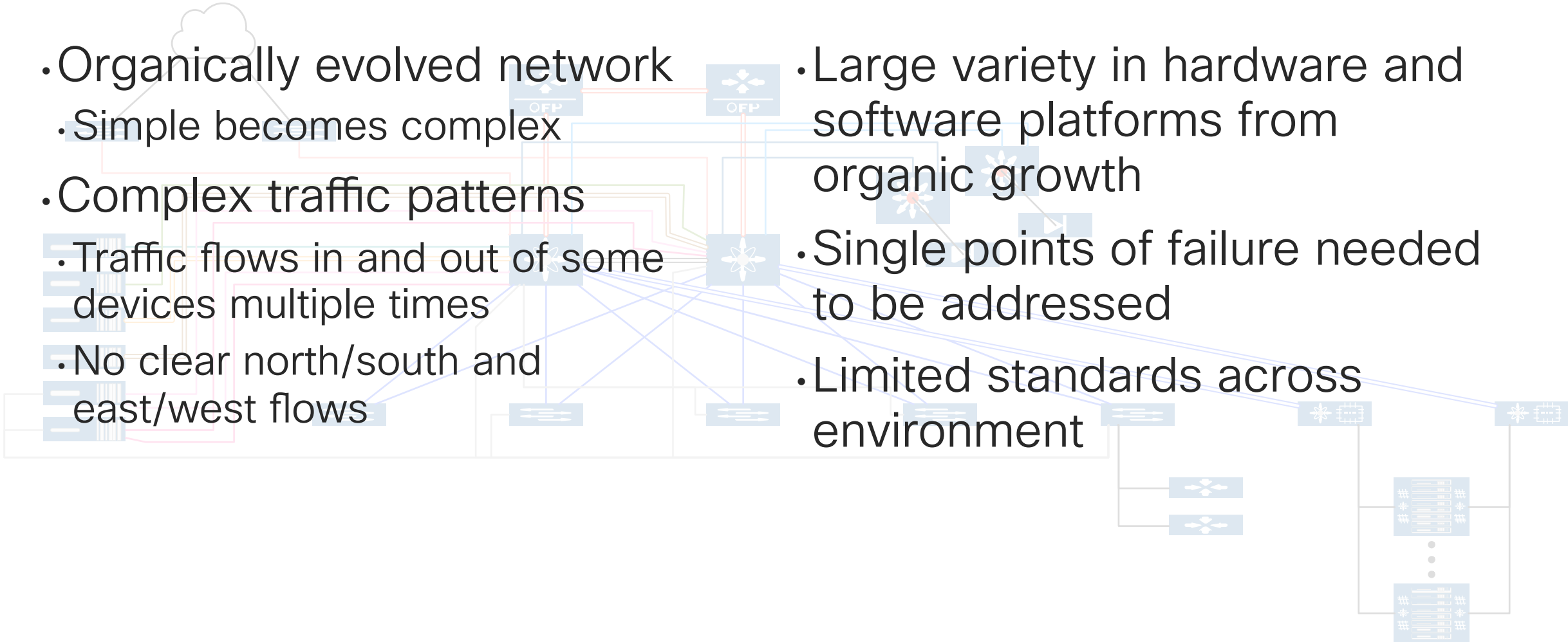


# DevNet Sandbox Physical Network Architecture



# DevNet Sandbox Physical Network Architecture

- Organically evolved network
  - Simple becomes complex
- Complex traffic patterns
  - Traffic flows in and out of some devices multiple times
  - No clear north/south and east/west flows
- Large variety in hardware and software platforms from organic growth
- Single points of failure needed to be addressed
- Limited standards across environment





# Process and Tooling

- Manual processes with minimal automation and often based on “experience”
- File based “source of truth”
  - Excel, text files, etc
- Cloud Storage (eg Box)
  - “Config v1.6.txt”
- Many many snowflakes



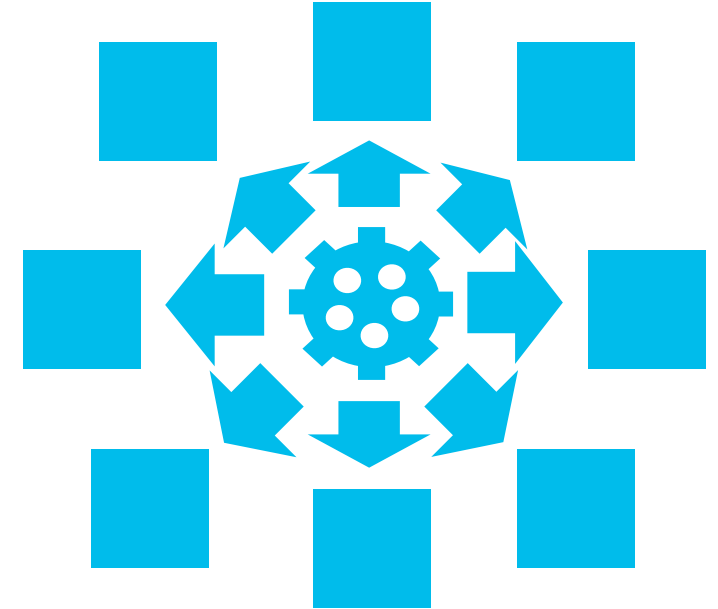
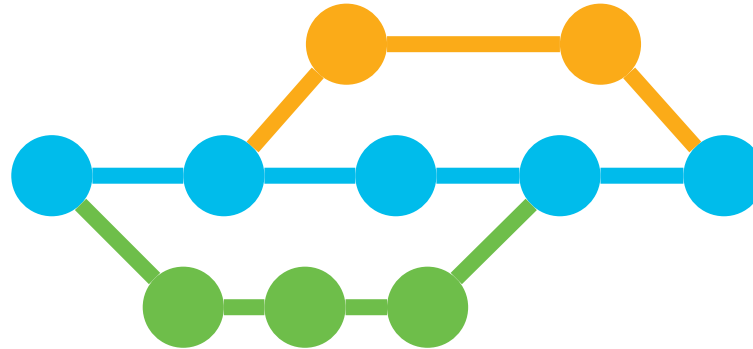
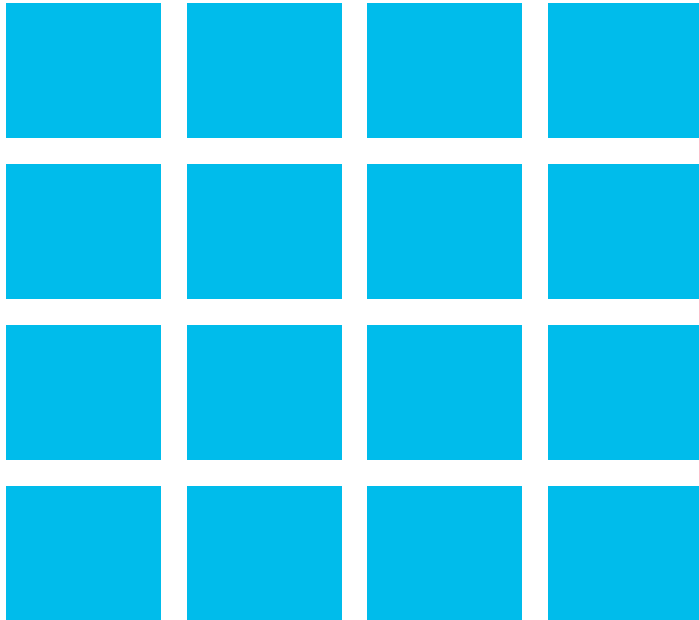




**We can do better!**



# NetDevOps In Sandbox Will Deliver



**Consistent Version Controlled Infrastructure deployed  
with Parallel & Automated Provisioning**

# Early Network Automation Wins (with NSO)

- Standardize firewall access lists across 500 devices
- NSO Device Templates
- Saved at least 32 error prone engineer man hours
- Basic Network “Service” Compliance (snmp, syslog, local admin)
- NSO Device Templates and Compliance Reports
- It just wasn’t right before...

```
devices template TACACS-ISE
config
  asa:aaa-server ISE_GROUP protocol tacacs+
  asa:aaa-server-host-conf aaa-server ISE_GROUP
    ({$MGMT_INTERFACE}) {$ISE_SERVER_1}
    key secret {$CLEAR_SECRET}

  asa:aaa authentication ssh console server_group ISEGROUP
  asa:aaa authentication ssh console LOCAL
  asa:aaa authorization command server_group ISE_GROUP
  asa:aaa accounting command server_group ISE_GROUP

  nx:feature tacacs+
  nx:tacacs-server host {$ISE_SERVER_1}
    secret key 7
    secret shared-secret {$TYPE7_SECRET}
    timeout 5

  nx:aaa group server tacacs+ ISE_GROUP
    server {$ISE_SERVER_1}
    source-interface {$MGMT_INTERFACE}
    use-vrf {$MGMT_VRF}
```





**Our New DC Project**



# Preparing for US West 1 Data Center

- Decision to move made July 2019
- Move to be completed December 31, 2019
- Possession October 7, 2019
- *3 months to design, test, and prove all changes*
- *2.5 months from empty racks to production service*

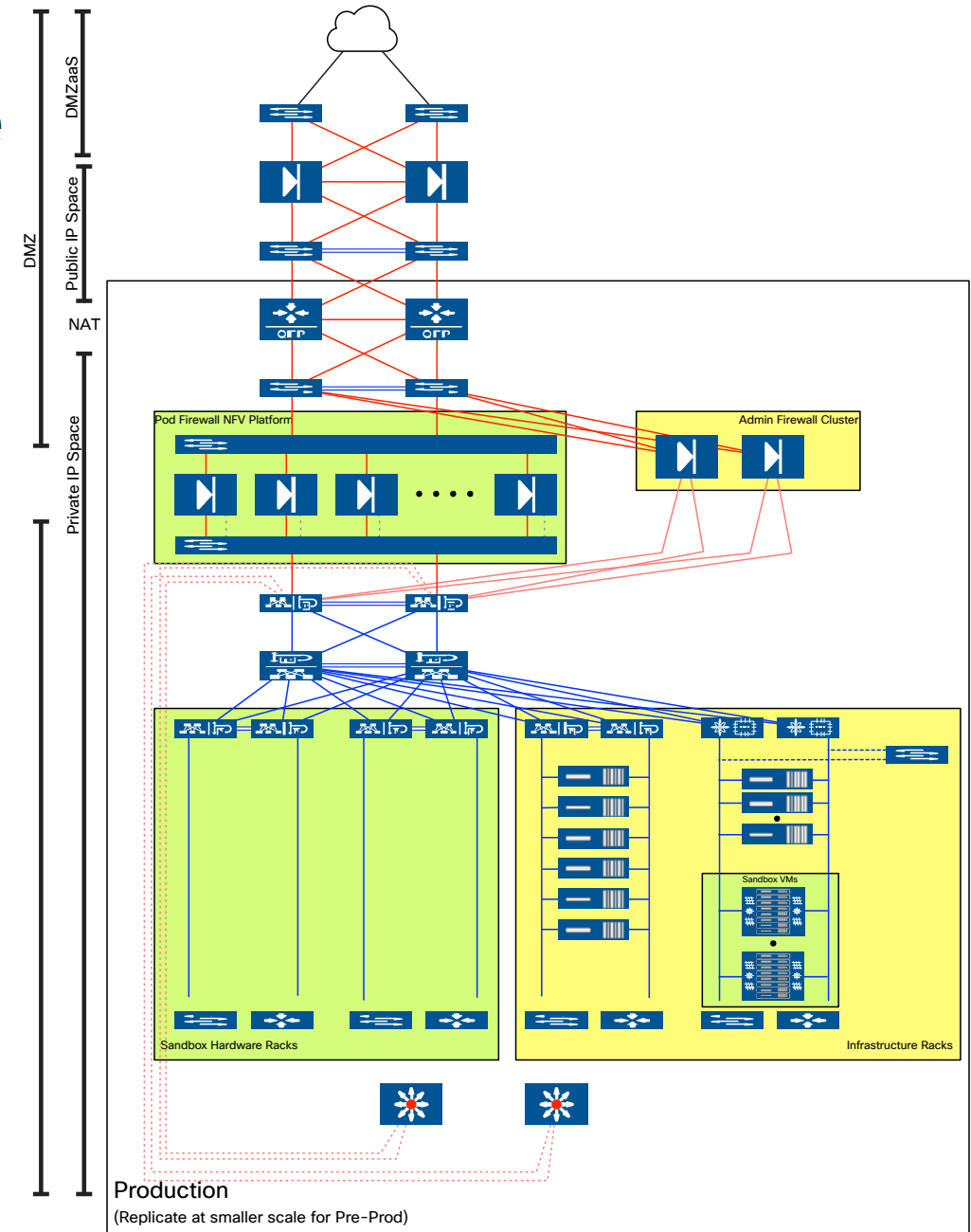
*"Innovation up to the point of panic"*

~ Sandbox Leadership



# Physical Network Architecture

- Fully redesigned target physical infrastructure
  - Physical and virtual components
- Clear North/South and East/West traffic flows
- Standardize on current hardware and software platforms
- No single points of failure
- Designed for horizontal scaling



# DevNet Sandbox – A Multi-Domain Network

- Deliver secure and reliable network services to a variety of public labs
- Both physical hardware and virtual machine environments
- Multi-Tenant network where customer facing and internal systems share physical resources

A vertical stack of network hardware components, including various Cisco switches and firewalls, with blue labels overlaid on the right side.

Cisco Firepower – NG Firewall

Cisco ASR 1K – Core Routing

Cisco ASA – Firewall

Cisco Nexus – Data Center Fabric

Cisco Catalyst – Mgmt Network

Cisco UCS – Compute

Cisco Container Platform – K8s

VMware – Virtual Switching





**Not everything can be  
done at once...**



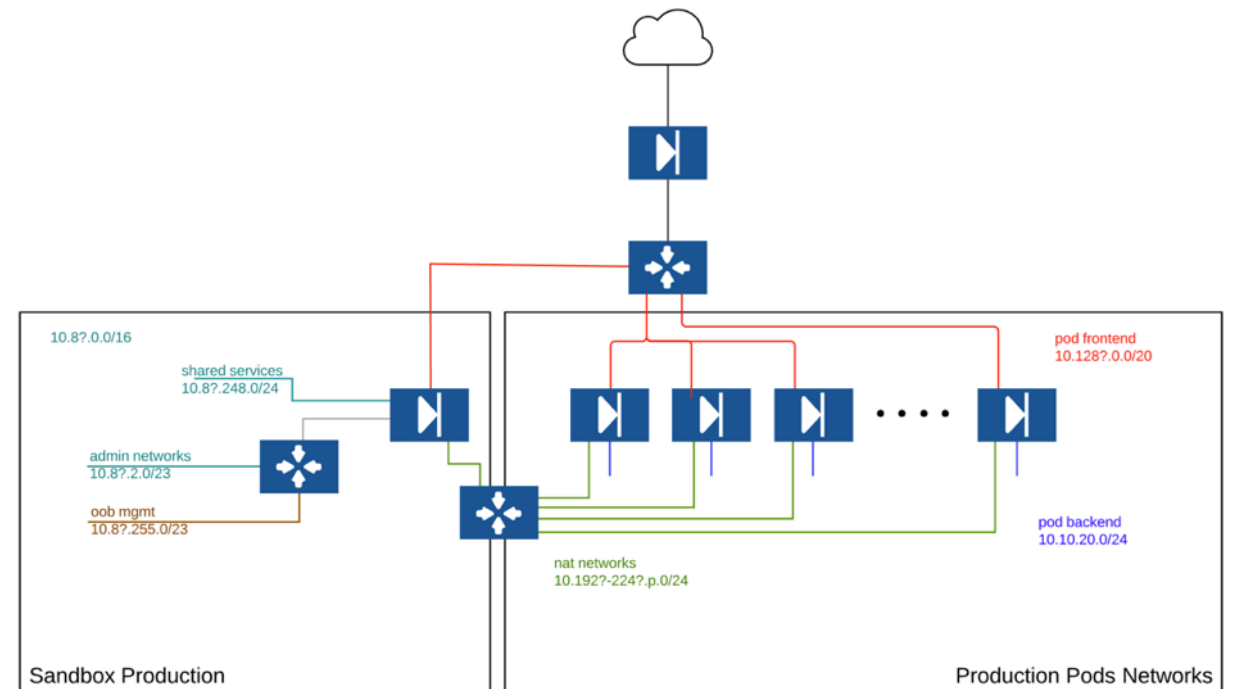


**Planning our  
first network services**



# Scoping the Services

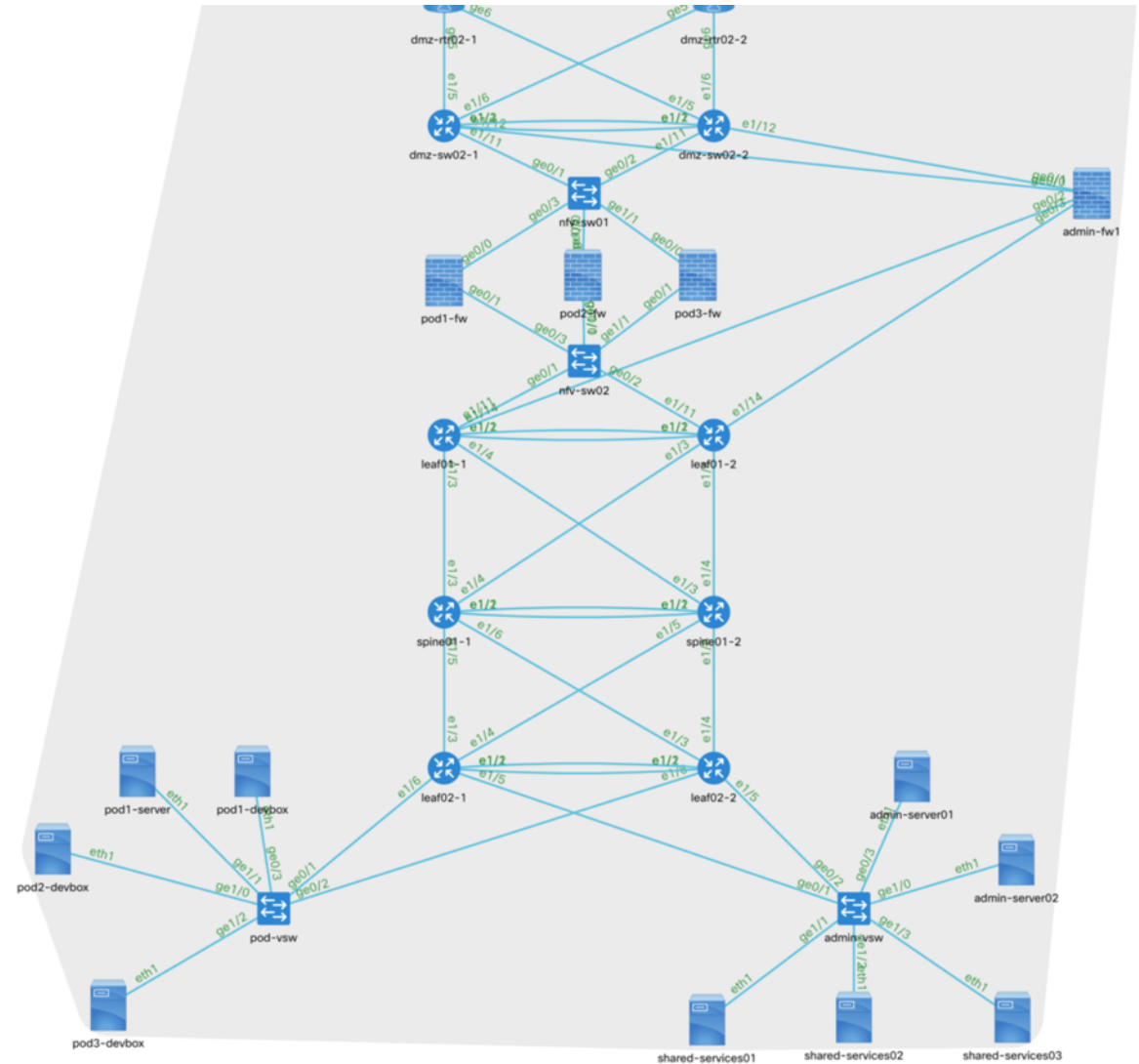
- Targeted highest impact parts of the network for service creation
- Underlay/overlay network configuration
- Firewall configuration



Logical Network Representation

# Scoping the Services

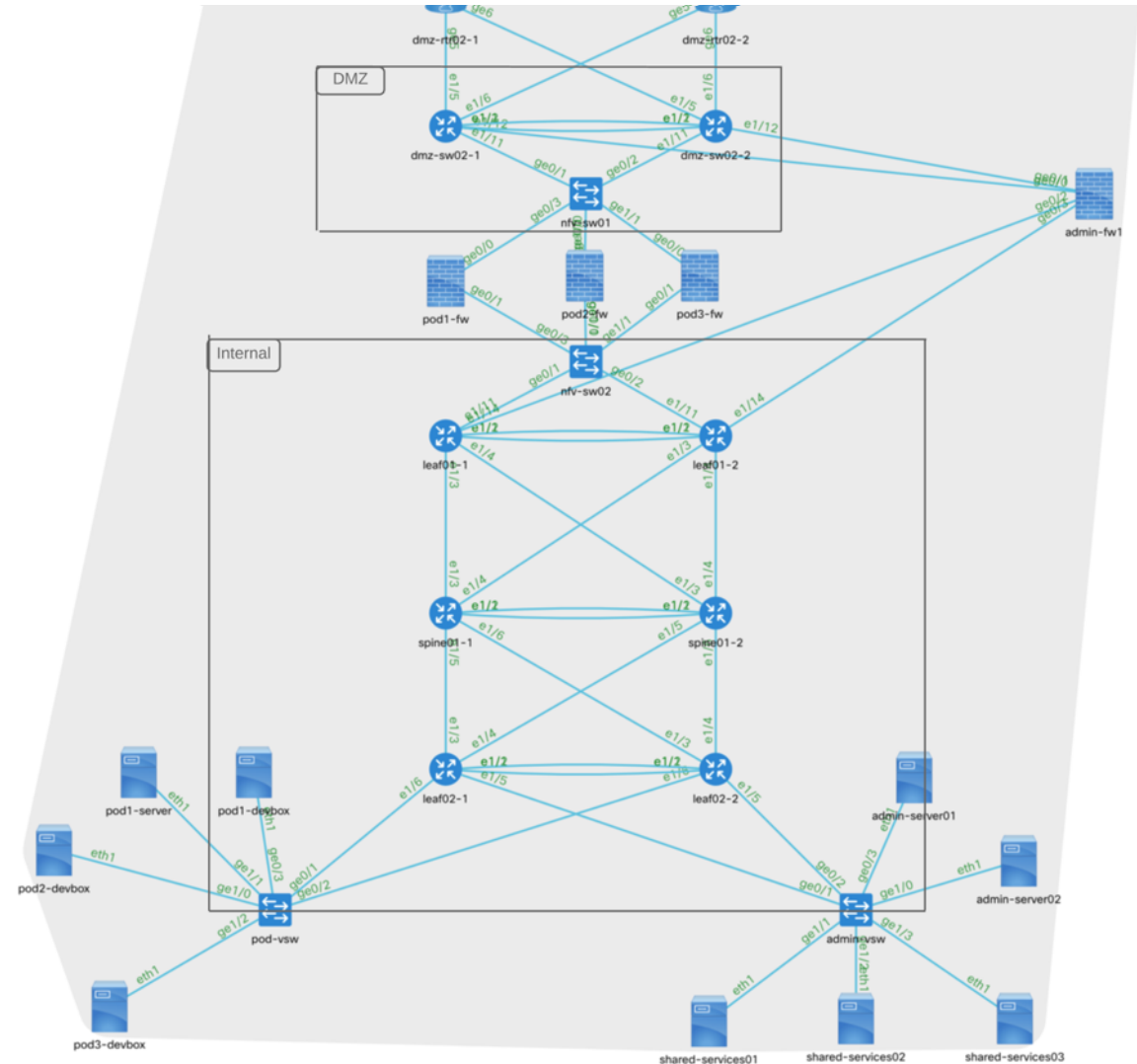
- Targeted highest impact parts of the network for service creation
- Underlay/overlay network configuration
- Firewall configuration





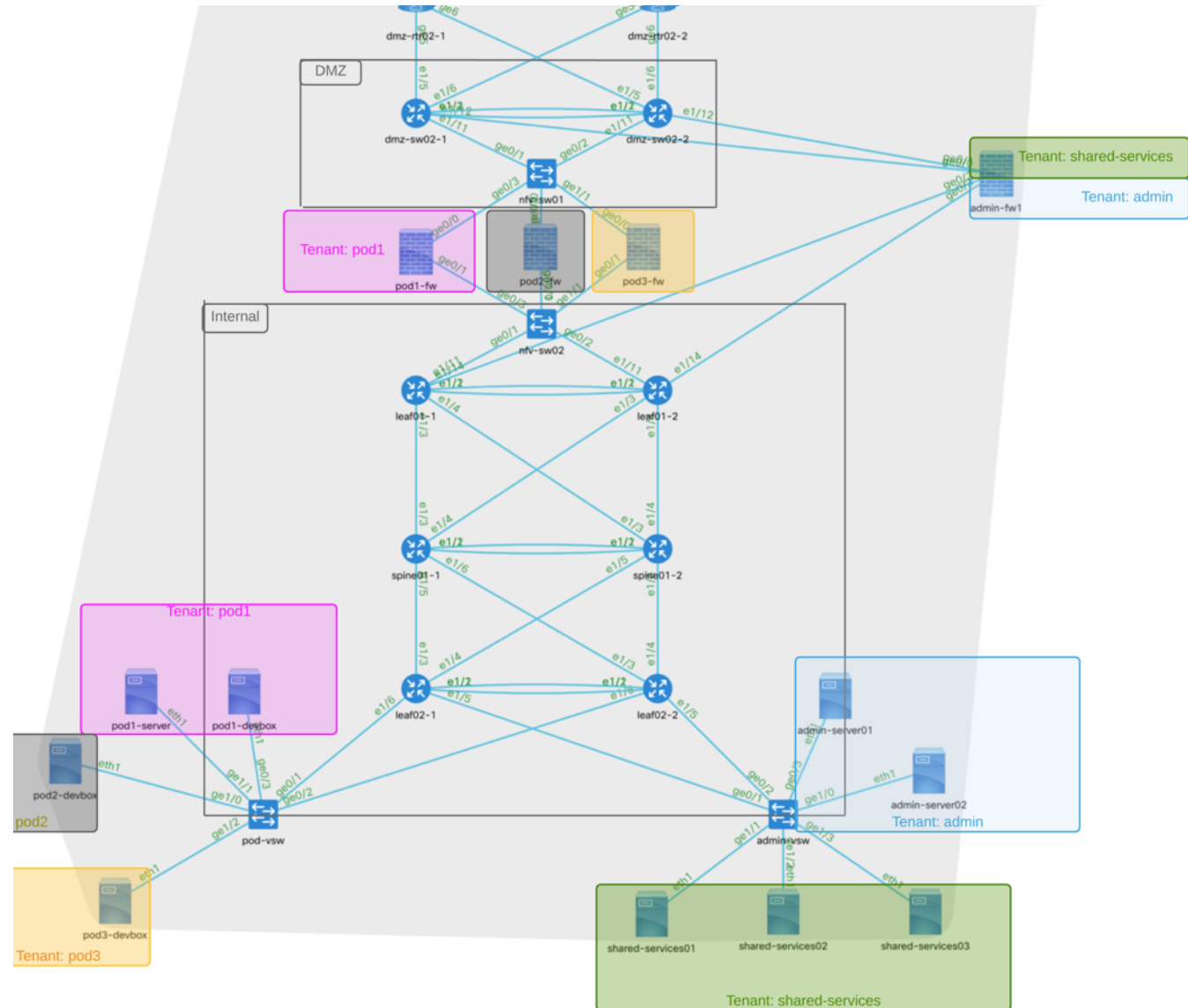
# Scoping the Services

- Targeted highest impact parts of the network for service creation
- Underlay/overlay network configuration
- Firewall configuration



# Scoping the Services

- Targeted highest impact parts of the network for service creation
- Underlay/overlay network configuration
- Firewall configuration



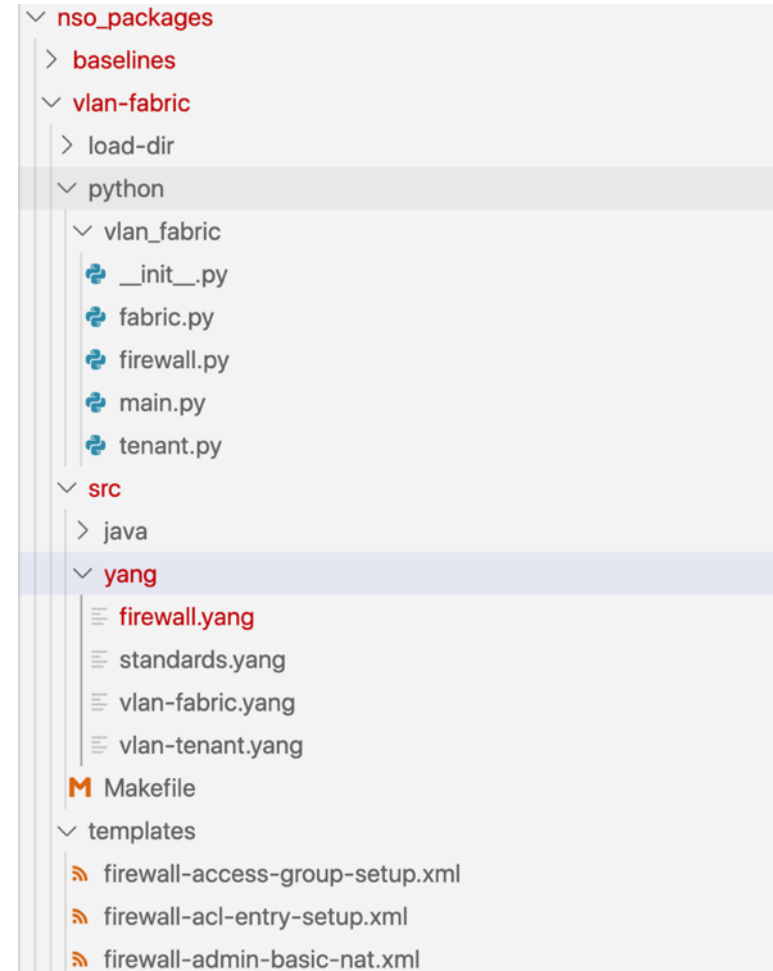


# Multi-Domain Network Automation

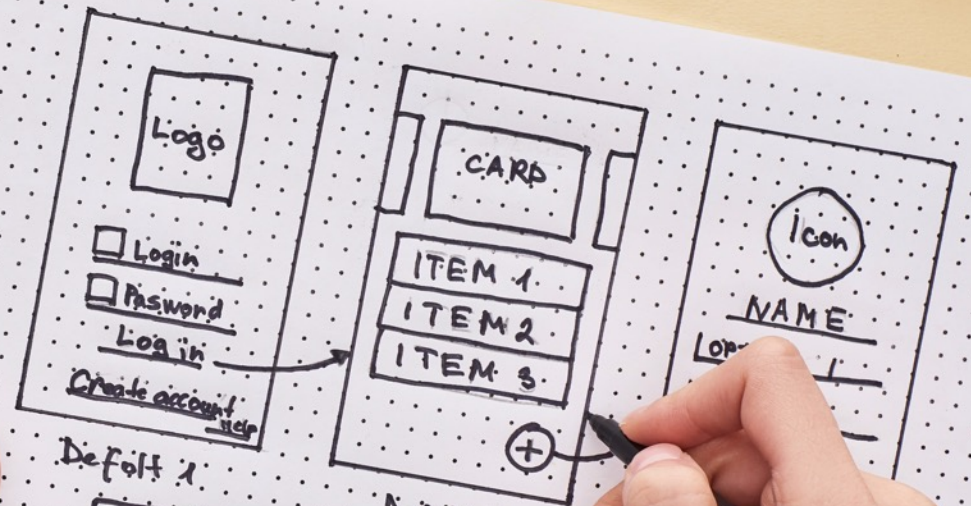
## Switch, Compute, and Virtual Network

### Initial Services Built

- **vlan-fabric**: Physical underlay
  - MLAG domains & interswitch trunks
- **vlan-tenant**: Overlay tenants
  - L2 and L3 domains
  - Physical network attachments
- **firewall**: Simplify and Consistency
  - Interfaces, Access Lists, Public Services, VPN management



# Prototyping



Default 1

Default 2

Start screen

Login

Help

Create account

personal info  
login  
password  
agreement

Create  
personal  
login and  
e-mail  
agreement



# Step 1 – Start with User Experience

## VLAN Fabric Data Model

- What should it be like to consume service?
- Psuedo-code drives YANG model

### **vlan-fabric**

- Describe underlay connectivity
- Cover “traditional” switches as well as “non-traditional” ones

```
vlan-fabric internal
  switch-pair leaf01
    layer3      true
    primary     leaf01-1
    secondary   leaf01-2
    vpc-peerlink id 1
    vpc-peerlink interface 1/53
    vpc-peerlink interface 1/54
    fabric-trunk 2
      interface 1/49
      interface 1/50

  fabric-interconnect fi01 ←
    vnic-template-trunks myorg1 vm-network-a
    vnic-template-trunks myorg2 esxi-vnic-a

  vmware-dvs vcenter1 mydatacenter mydvs ←
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

# Step 1 – Start with User Experience

## VLAN Tenant Data Model

- What should it be like to consume service?
- Psuedo-code drives YANG model

### **vlan-tenant**

- Describe the L2/L3 environment
- Focus on unique details per network

```
vlan-tenant admin
fabric internal ←
static-routes 0.0.0.0/0
gateway 172.23.250.4

network admin-containers
vlanid          25
network         172.23.4.0/23
layer3-on-fabric true
dhcp-relay-address 172.23.2.11

network admin-main
vlanid          11
network         172.23.2.0/23
layer3-on-fabric true
connections switch-pair leaf01
interface 1/33
description "Link to NUC ESXI"
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*



# Step 1 – Start with User Experience

## Firewall Data Model

- What should it be like to consume service?
- Psuedo-code drives YANG model

### **firewall**

- Connect firewall interfaces to tenants
- “simplify” the configuration

```
firewall myfirewall
  type    myfirewall
  device  fw01-1
  interface inside
    tenant admin ←
  network admin-fw-transit ←
  route 172.23.252.0/23
    gateway 172.23.250.1
  ospf    true
  outside-nat outside-ip 131.226.217.1
  outside-nat supernet 172.23.0.0/16

  acl-allow DNS_SERVERS
    description "Allow DNS Lookups"
    destination ip-list dns-servers
    destination protocol udp
    destination port 53
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

# Step 1 – Start with User Experience

## Firewall Data Model

- What should it be like to consume service?
- Psuedo-code drives YANG model

### **firewall**

- Connect firewall interfaces to tenants
- “simplify” the configuration

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

```
firewall myfirewall
  tunnel myfirewall
  device myfirewall
  interface myfirewall
    vpn username testuser
      password Hnkai7anaf

  vpn secure-network 172.23.0.0/21
  vpn secure-network 172.23.232.0/21
  vpn dns server [ 172.23.2.11 ]
  vpn dns secured-domain devnetsandbox.local
  vpn dns default-domain devnetsandbox.local
  vpn client anyconnect-linux64-4.7.pkg
  vpn client anyconnect-macos-4.7.pkg
  vpn client anyconnect-win-4.7.pkg
```



# Step 2 – Intended Network Configuration

## Native Device Configuration

- Understand the desired end state
- Configure the network “the old way”
- Capture final configurations
- Use to build XML templates

```
version 9.3(1) Bios:version 05.38
feature vpc

vpc domain 10
  peer-switch
  peer-keepalive destination 172.23.252.30
    source 172.23.252.29
  peer-gateway
  auto-recovery

interface port-channel1
  vpc peer-link
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

# Step 2 - Intended Network Configuration

## NSO Modeled Configuration

- Understand the desired end state
- Configure the network “the old way”
- Capture final configurations
- Use to build XML templates

```
show running-config devices device leaf01-1 \
  config nx:interface port-channel 1 vpc \
  | display xml
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

```
<config xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>usw1-leaf01-1</name>
      <config>
        <vpc xmlns="http://tail-f.com/ned/cisco-nx">
          <domain >
            <id >10</id>
            <peer-gateway />
            <peer-keepalive>
              <destination >10.17.252.30</destination>
              <source >10.17.252.29</source>
            </peer-keepalive>
            <peer-switch />
          </domain>
        </vpc>
        <interface xmlns="http://tail-f.com/ned/cisco-nx">
          <port-channel>
            <name>1</name>
            <vpc>
              <peer-link />
            </vpc>
          </port-channel>
        </interface>
      </config>
    </device>
  </devices>
</config>
```



# Step 2 – Intended Network Configuration

## NSO Templatized Configuration

- Understand the desired end state
- Configure the network “the old way”
- Capture final configurations
- Use to build XML templates

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <devices xmlns="http://tail-f.com/ns/ncs">
    <device>
      <name>{$DEVICE_NAME}</name>
      <config>
        <feature xmlns="http://tail-f.com/ned/cisco-nx">
          <vpc/>
          <lacp/>
        </feature>
        <vpc xmlns="http://tail-f.com/ned/cisco-nx">
          <domain>
            <id>{$VPC_DOMAIN_ID}</id>
            <peer-gateway/>
            <peer-keepalive>
              <destination>{$VPC_PEER_KEEPALIVE_DESTINATION}</destination>
              <source>{$VPC_PEER_KEEPALIVE_SOURCE}</source>
            </peer-keepalive>
            <peer-switch/>
          </domain>
        </vpc>
      </config>
    </device>
  </devices>
</config-template>
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

# Step 2 - Intended Network Configuration

## NSO Templatized Configuration

- Understand the desired end state
- Configure the network “the old way”
- Capture final configurations
- Use to build XML templates

*Note: Configurations, templates, code, etc have been simplified for this presentation.*

```
<config-template xmlns="http://tail-f.com/ns/config/1.0">
  <
    .
    .
    .
    <interface xmlns="http://tail-f.com/ned/cisco-nx">
      <port-channel>
        <name>{$VPC_PEERLINK_ID}</name>
        <enable>
          <switchport>true</switchport>
        </enable>
        <spanning-tree>
          <port>
            <type>network</type>
          </port>
        </spanning-tree>
        <switchport>
          <mode>trunk</mode>
        </switchport>
        <vpc>
          <peer-link/>
        </vpc>
      </port-channel>
    </interface>
    </config>
  </device>
</devices>
</config-template>
```

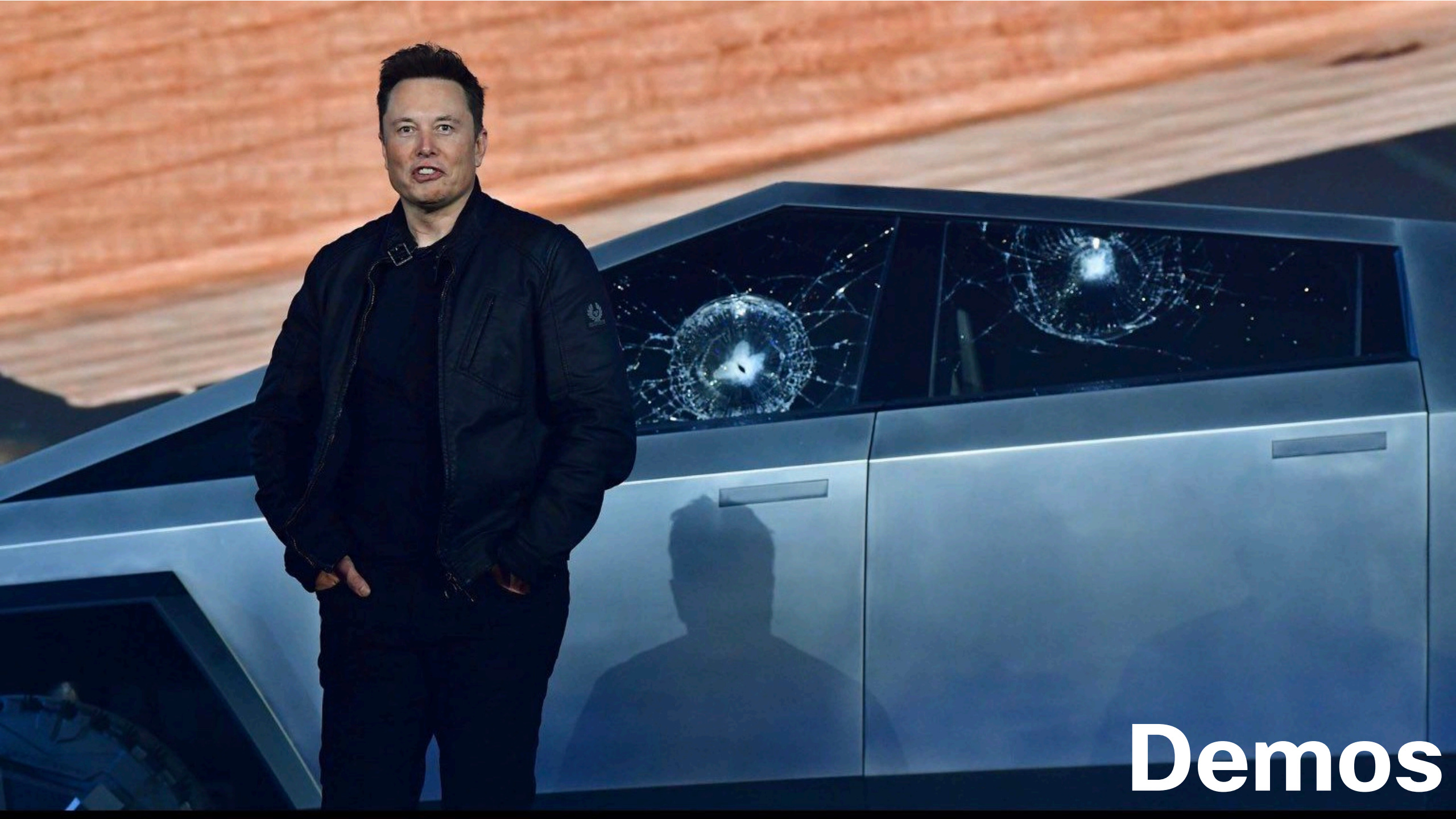


## Step 3 – Put it together

- Write code that uses the model and templates to replicate the intended configuration
- Start small with parts of the config and iterate until complete

```
class ServiceCallbacks(Service):
    @Service.create
    def cb_create(self, tctx, root, service, proplist):
        for pair in service.switch_pair:
            vpc_vars = ncs.template.Variables()
            vpc_vars.add("VPC_DOMAIN_ID", vpc_domain_id)
            vpc_vars.add("VPC_PEERLINK_ID", pair.vpc_peerlink.id)
            vpc_vars.add("LAYER3", pair.layer3)
            vpc_vars.add("DEVICE_NAME", pair.primary)
            vpc_vars.add("VPC_PEER_KEEPALIVE_SOURCE", primary_ip_address.split("/")[0])
            vpc_vars.add("VPC_PEER_KEEPALIVE_DESTINATION", secondary_ip_address.split("/")[0])
            template.apply("vpc-domain-base", vpc_vars)
```

*Note: Configurations, templates, code, etc have been simplified for this presentation.*



**Demos**



# Lesson's Learned





Be flexible...







Finding balance can be hard



# Performance tuning







**Perfect maybe the enemy of  
good... but some decisions  
you must live with**



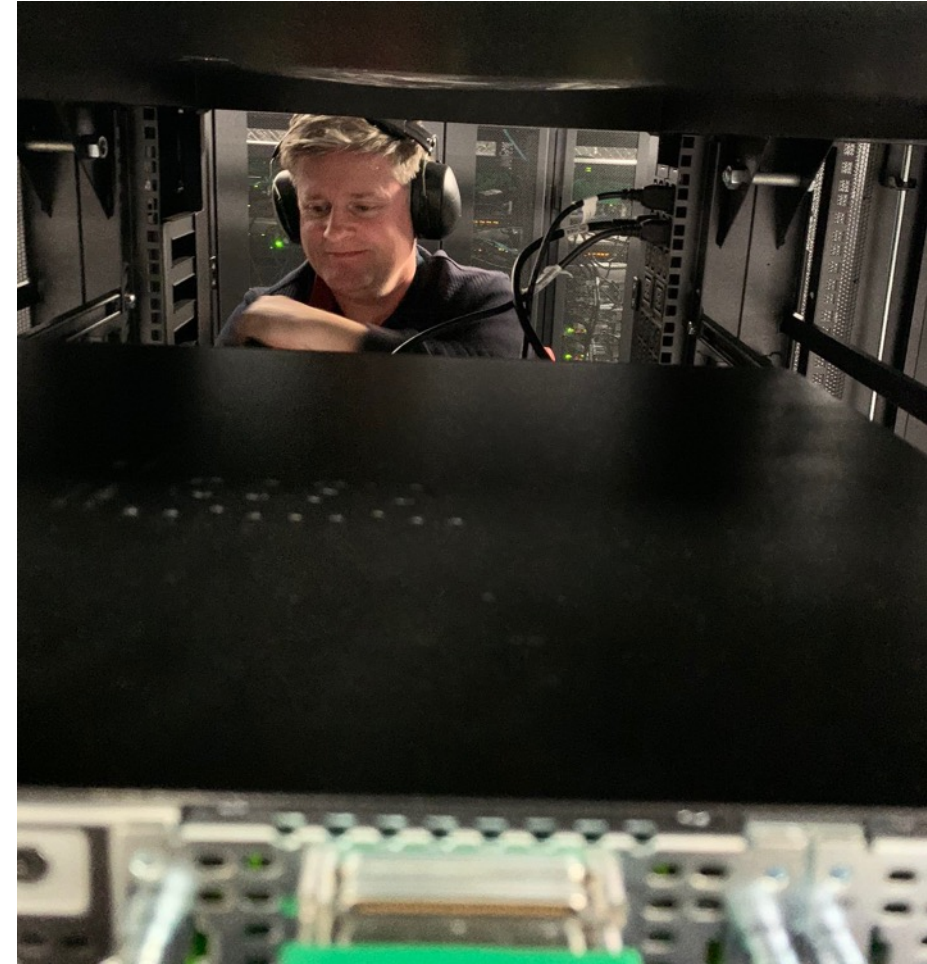


**What's next?**



# Somethings we're hoping to dive into next

- New services
- CI/CD for services
- Better “packaging”
- Train our engineers and build comfort



# Closing





# Got more questions? Stay in touch!



**Hank Preston**



**[developer.cisco.com](https://developer.cisco.com)**



[hapresto@cisco.com](mailto:hapresto@cisco.com)



[@hfpreston](https://twitter.com/hfpreston)



<http://github.com/hpreston>



[@CiscoDevNet](https://twitter.com/CiscoDevNet)



[facebook.com/ciscocodevnet/](https://facebook.com/ciscocodevnet/)



<http://github.com/CiscoDevNet>



**DEVNET**  
developer.cisco.com