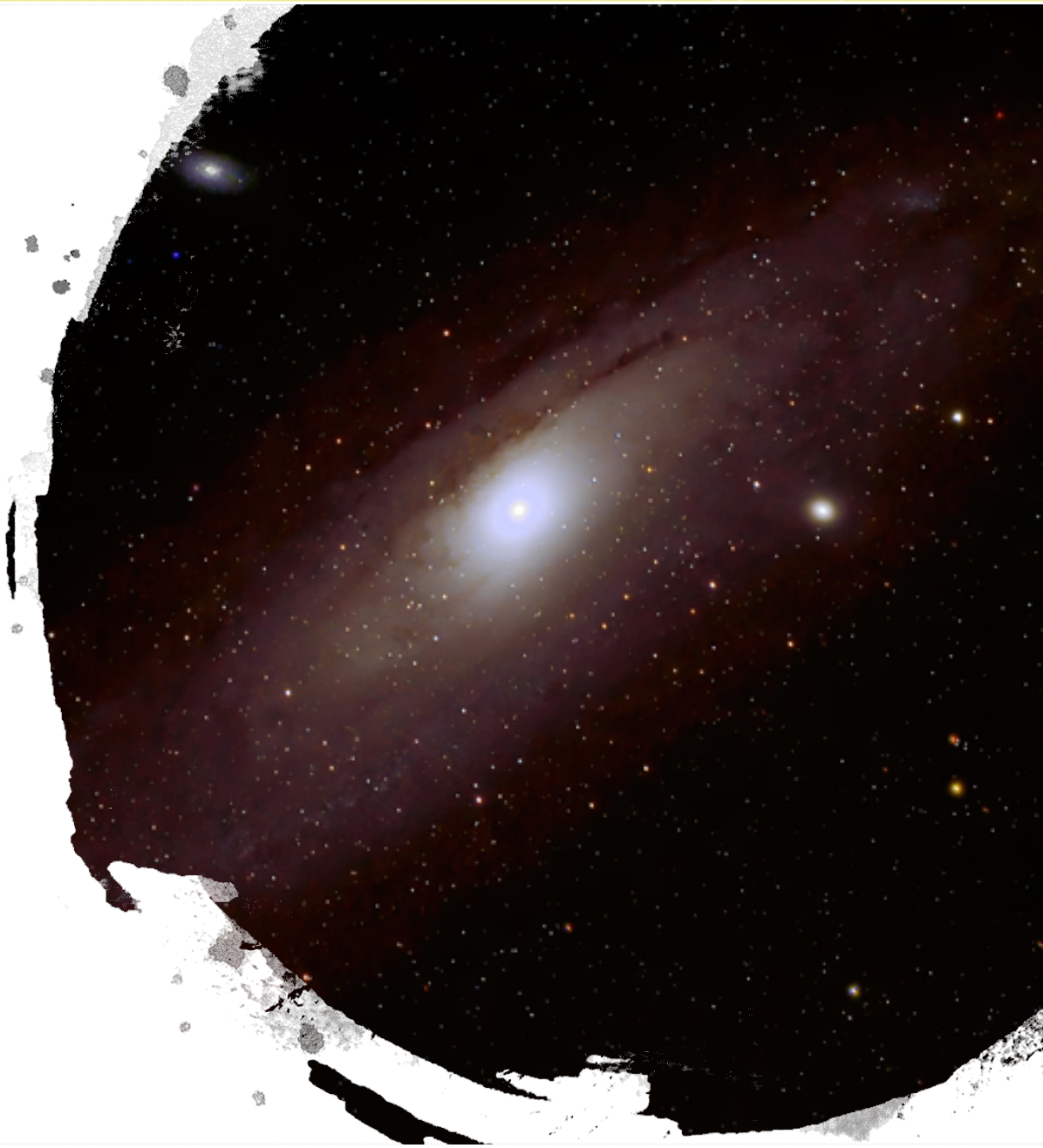
How We Built an Automated Data Center Network in Less than 6 Months

Hank Preston
Principal Engineer
Labs, Sandbox and Training Delivery with DevNet
Twitter: @hfpreston Email: hapresto@cisco.com

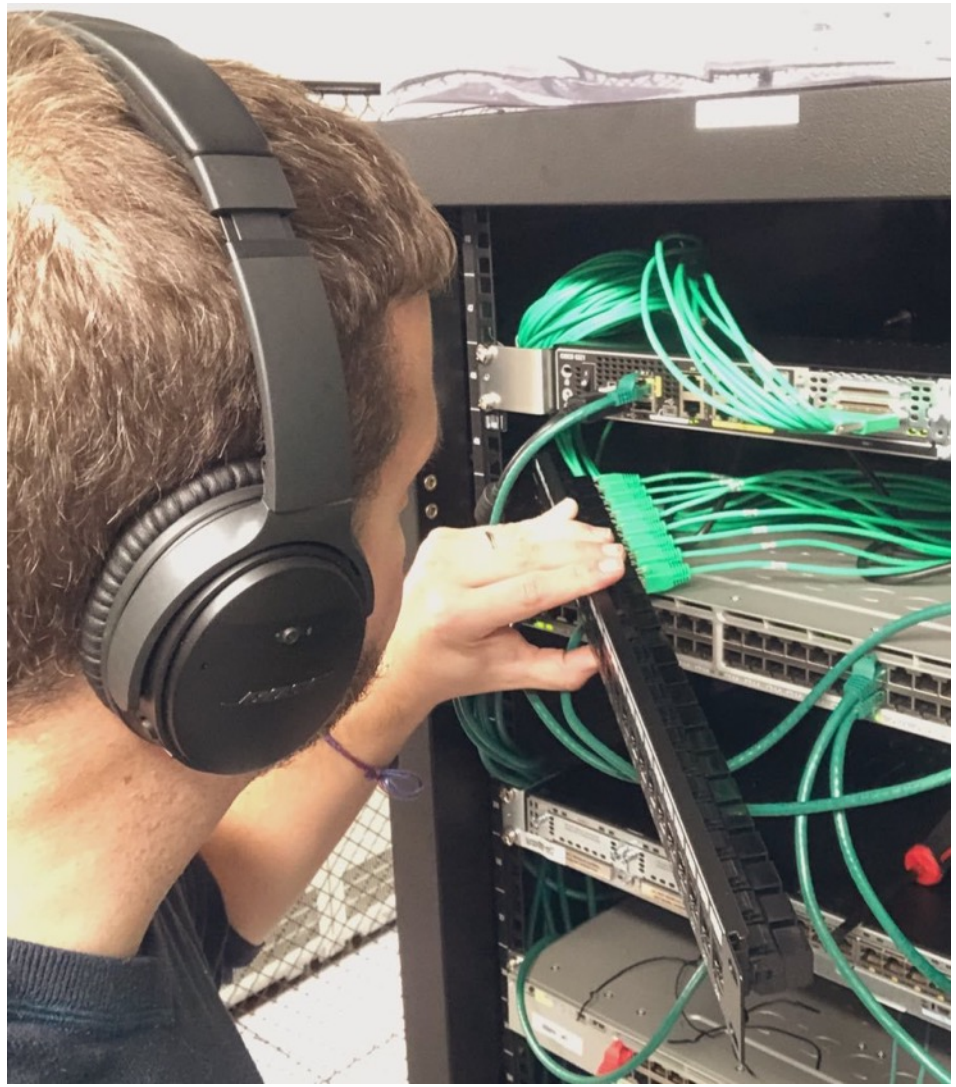
About me

- Returned to networking about 6 years ago to help engineers automate
- Definitely ***NOT*** the best coder in the room
- Day job: Architecting the data centers behind DevNet Sandbox and Cisco Learning Classes and Certifications
- New hobbies… Woodworking and Astrophotography
- Want to learn more about logging, telemetry, and testing for networking



Headline

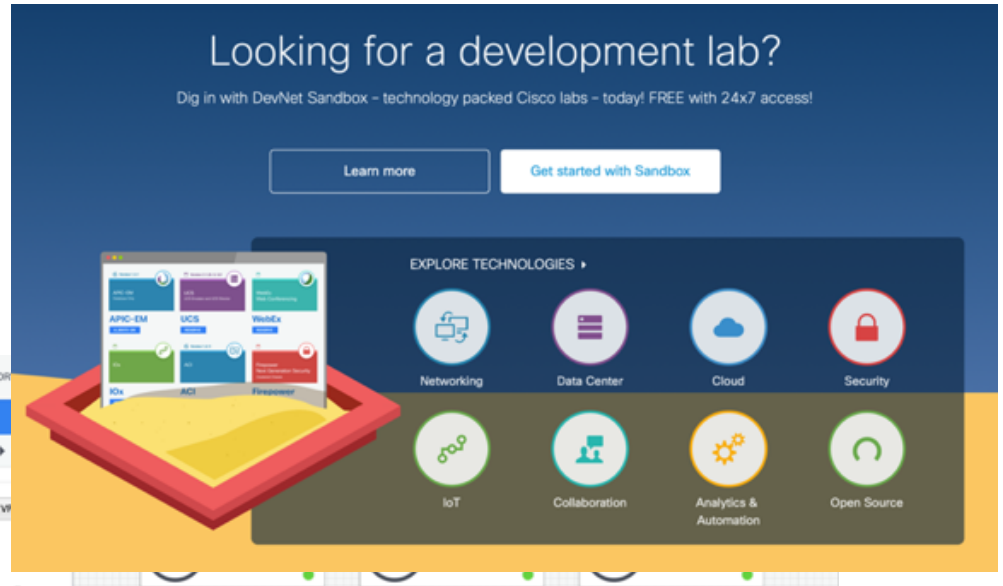
- What is DevNet Sandbox?
- A Bit of History
- Our Data Center Project
- Planning Phase 1 Network Automation
- Some Lessons Learned
- What's Next in our Journey

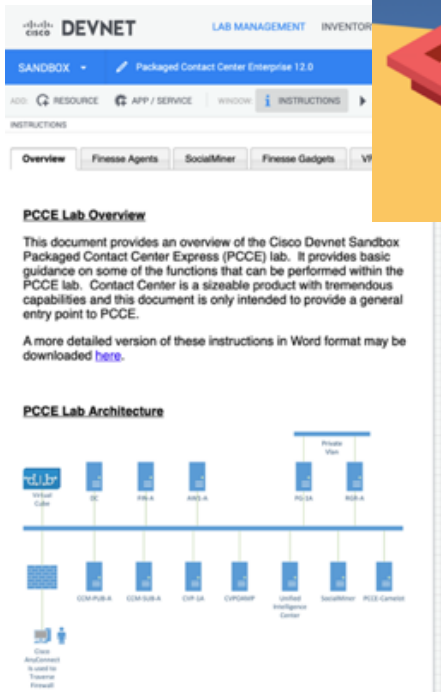


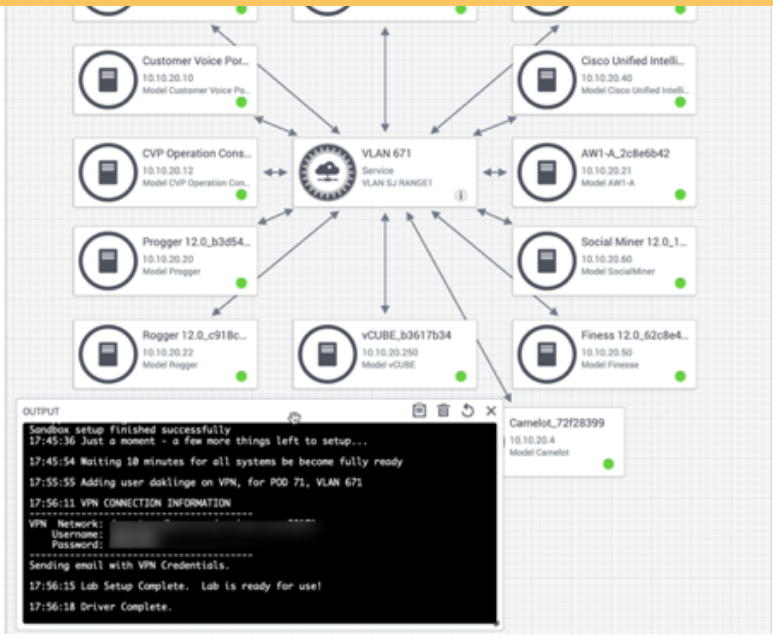
What is DevNet Sandbox?

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!





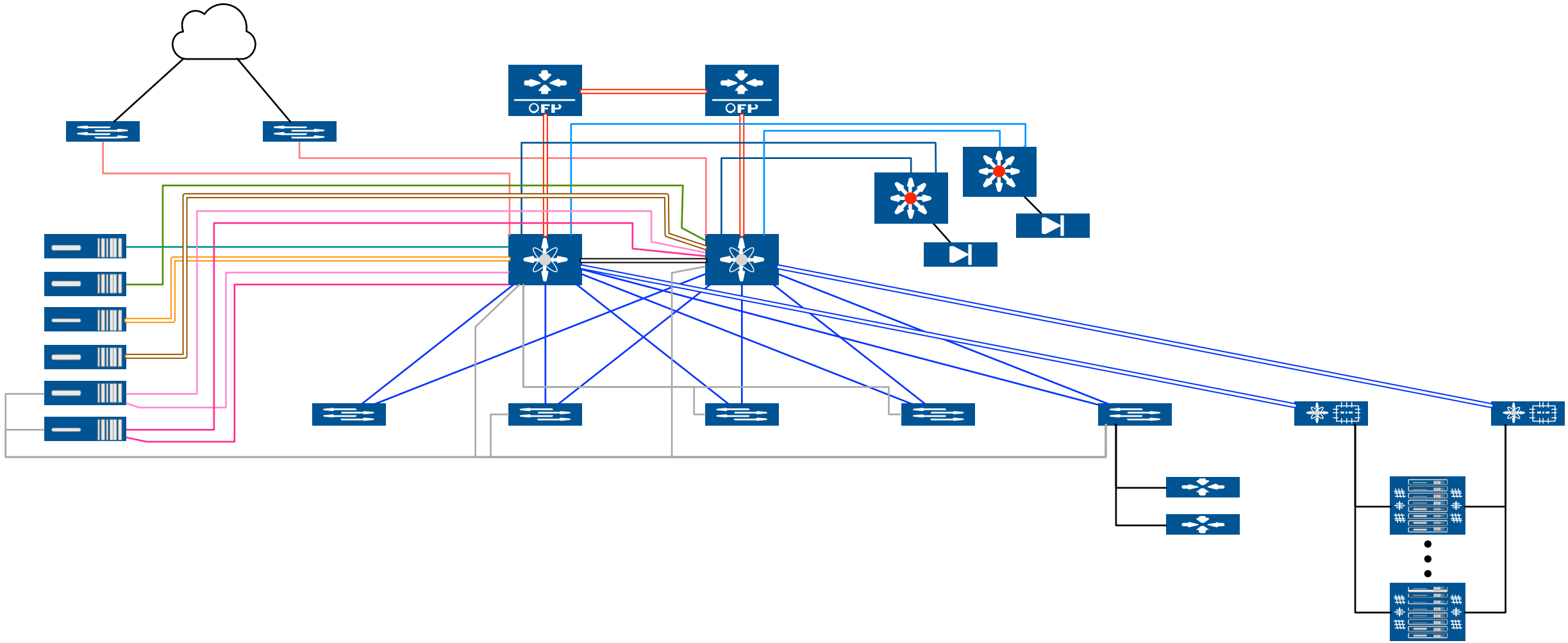


We make innovation easy!


We build our own cloud...

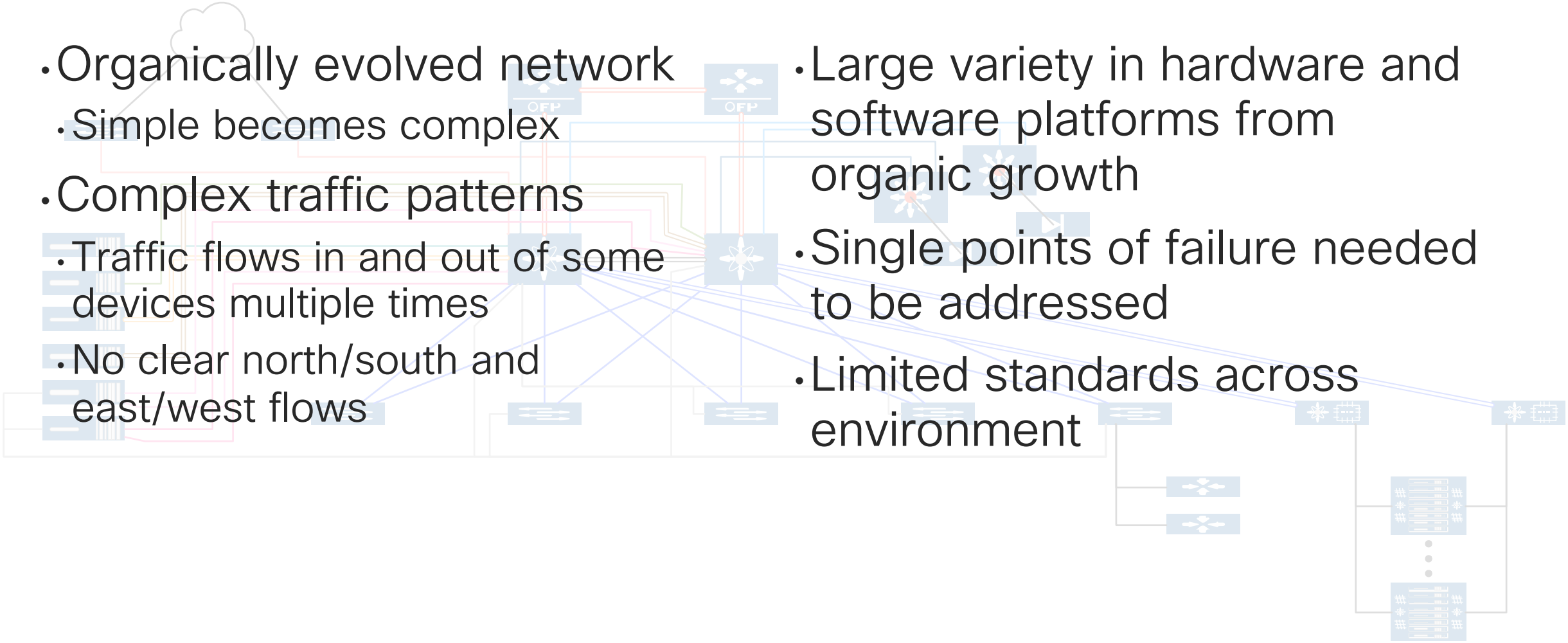

Where we came from?

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

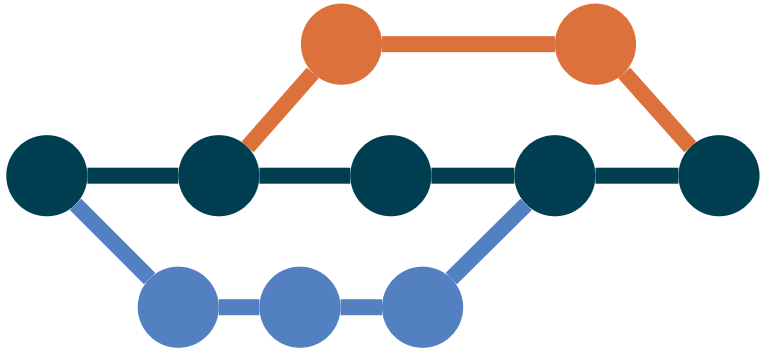


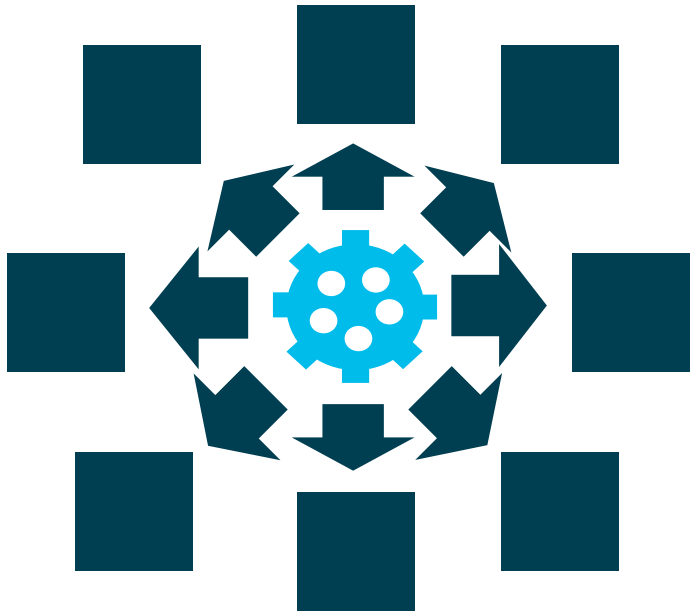
Process and Tooling

- 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

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”

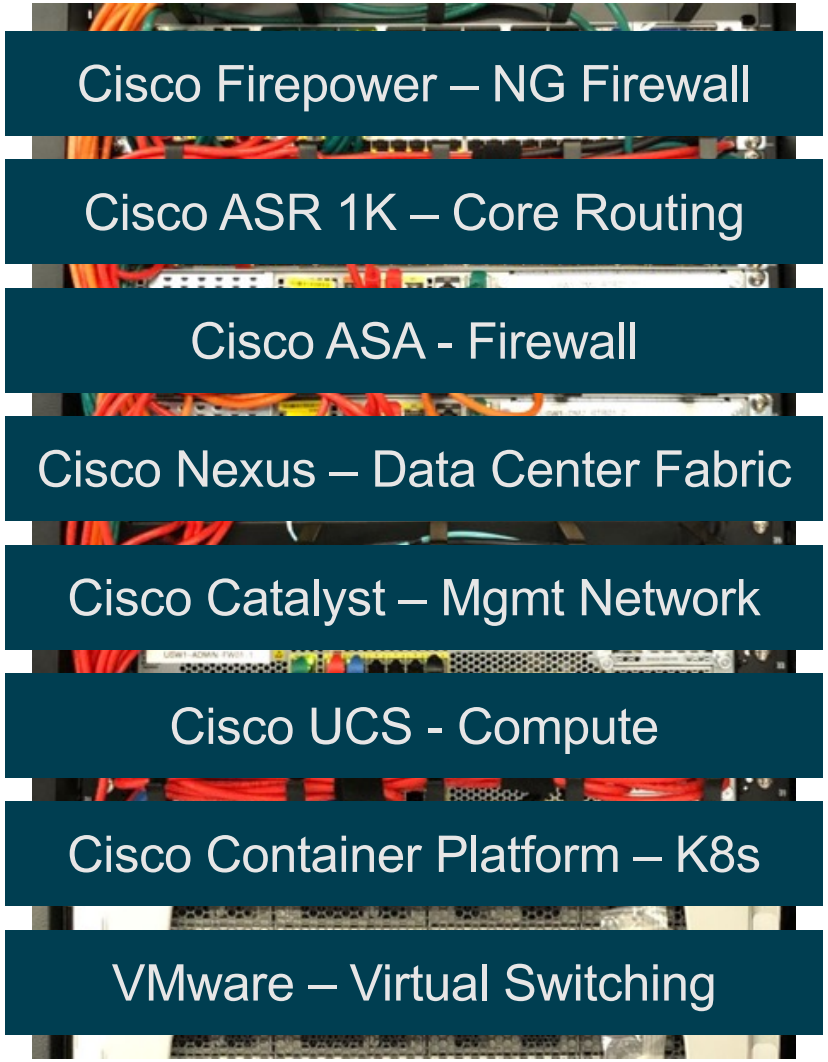
~ Sandbox Leadership

“We deploy the network for 200 pods in less than 5 minutes.”

~ Hank Preston – Sandbox Automation Architect

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



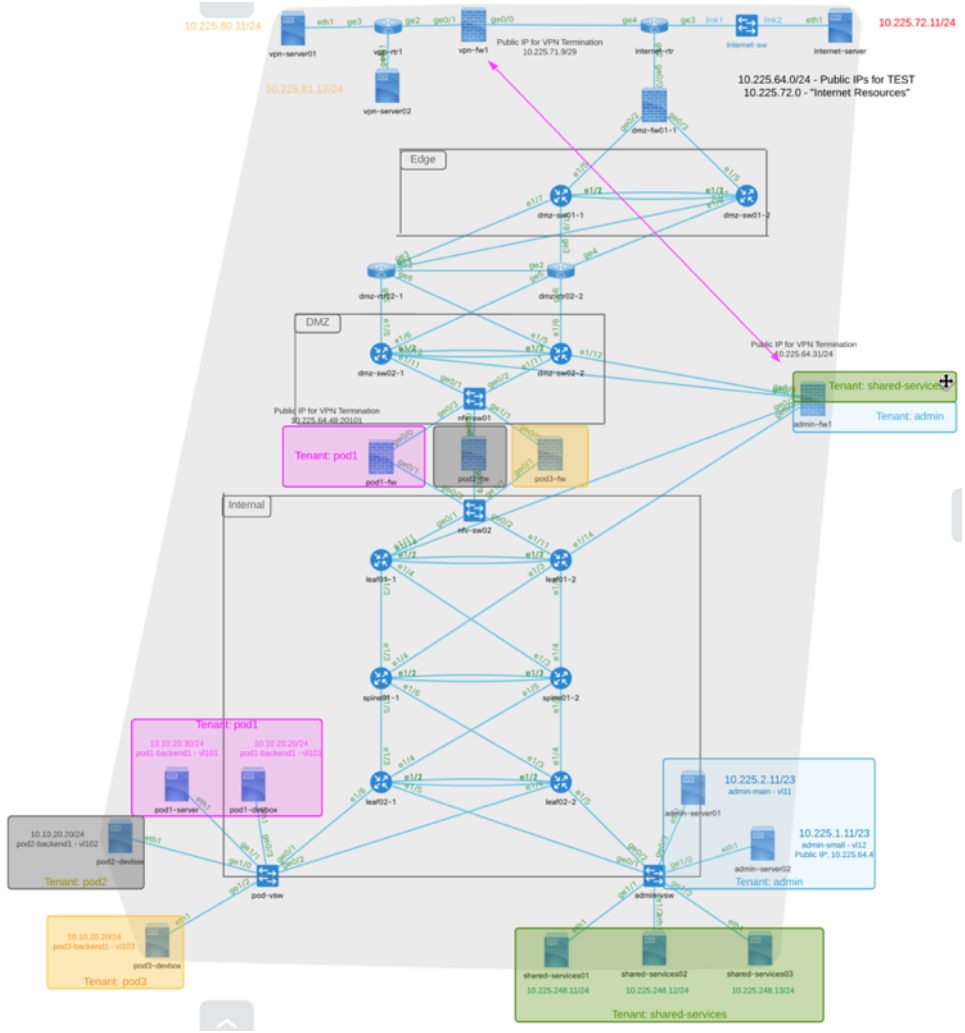
Not everything can be done at once...


Planning our
first network services

Cisco Modeling Labs

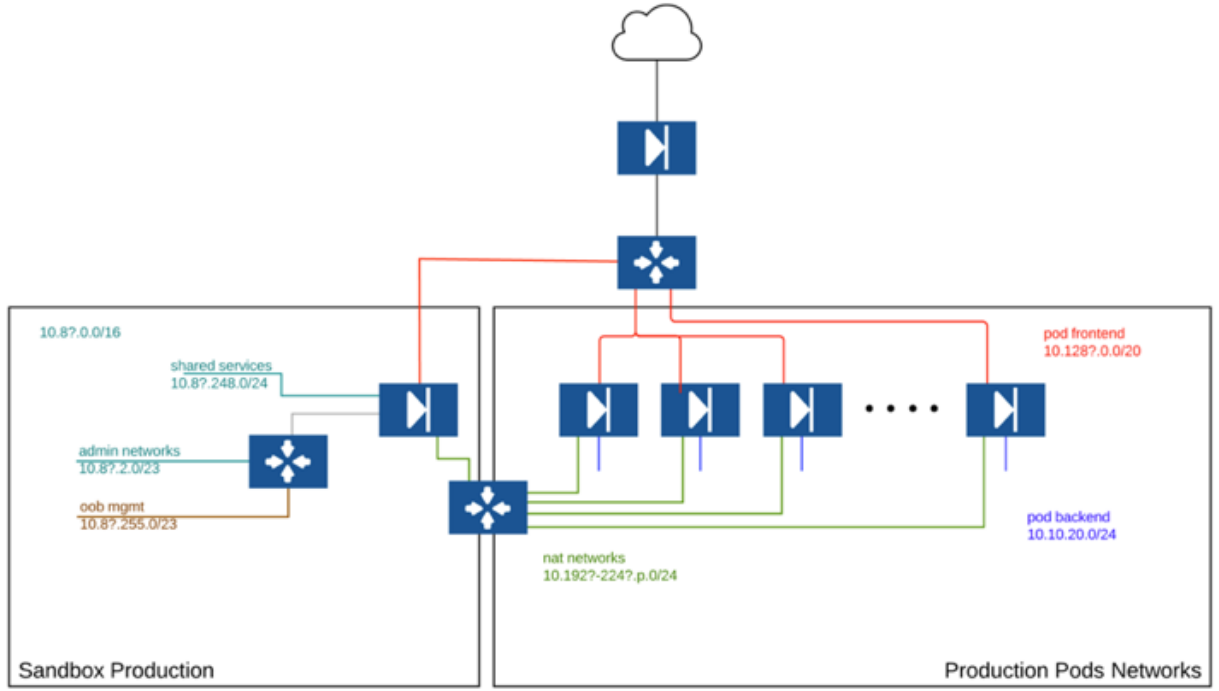
Building and Automating a New Network without the Network

- Production Grade Development and Test Networks
- Simulate all layers of network topology
- Include key application and client systems
- From Internet to hosted sandboxes
- Test network architecture, security policy, templates and automation tooling



Scoping the Services

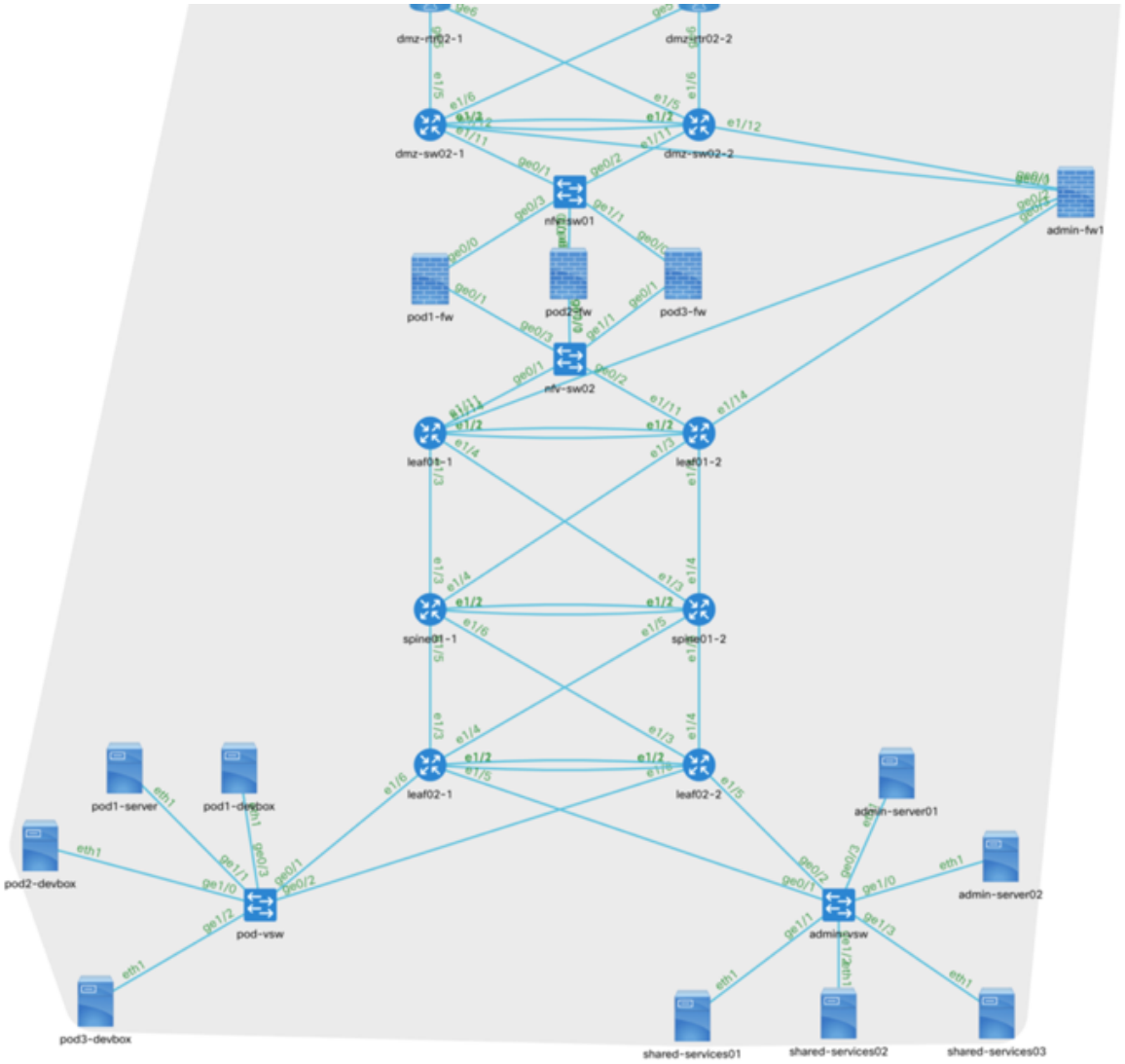
- Targeted highest impact parts of the network for service creation
- Underlay/overlay network configuration
- Firewall configuration



Logical Network Representation

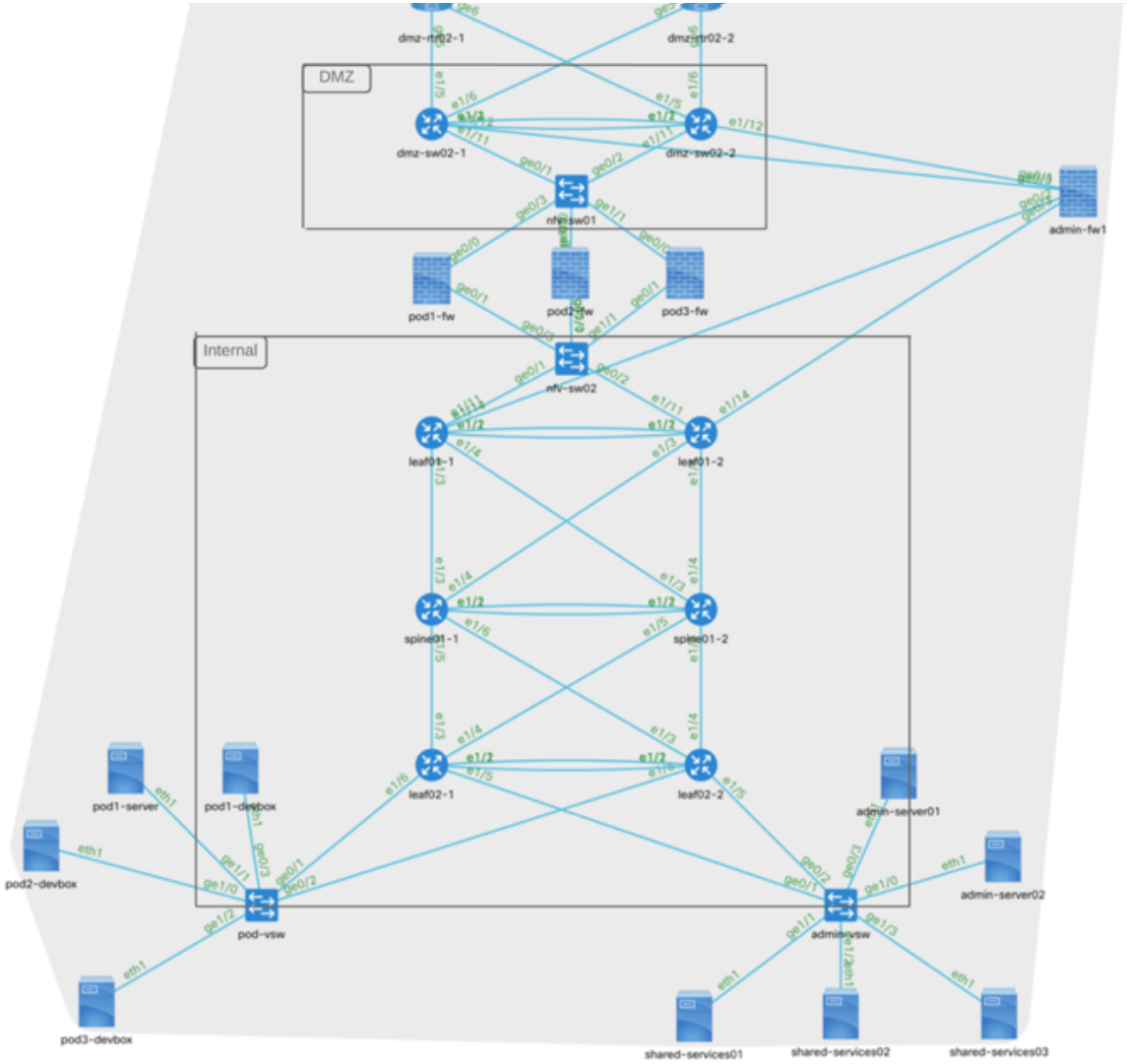
Scoping the Services

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



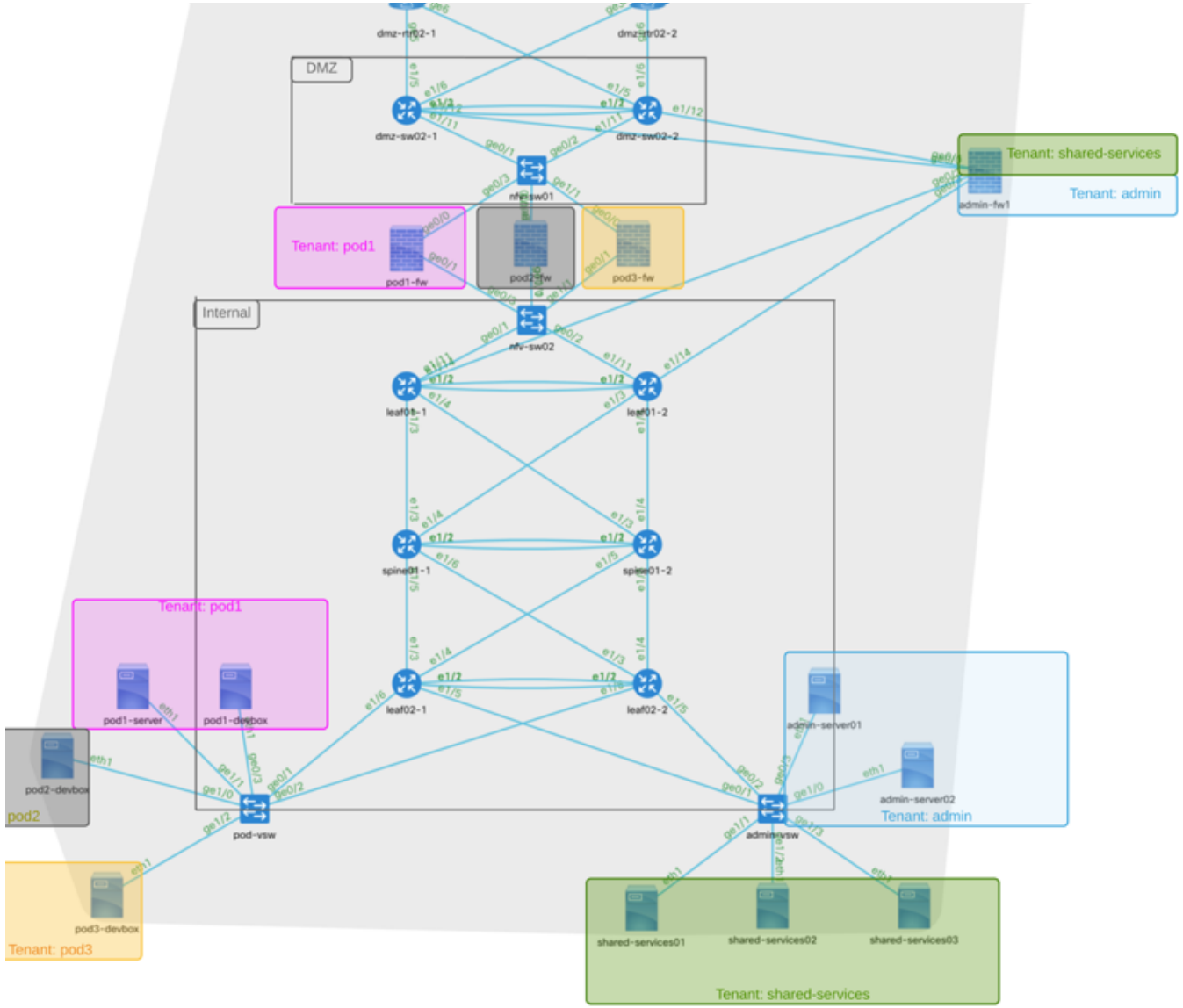
Scoping the Services

- 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

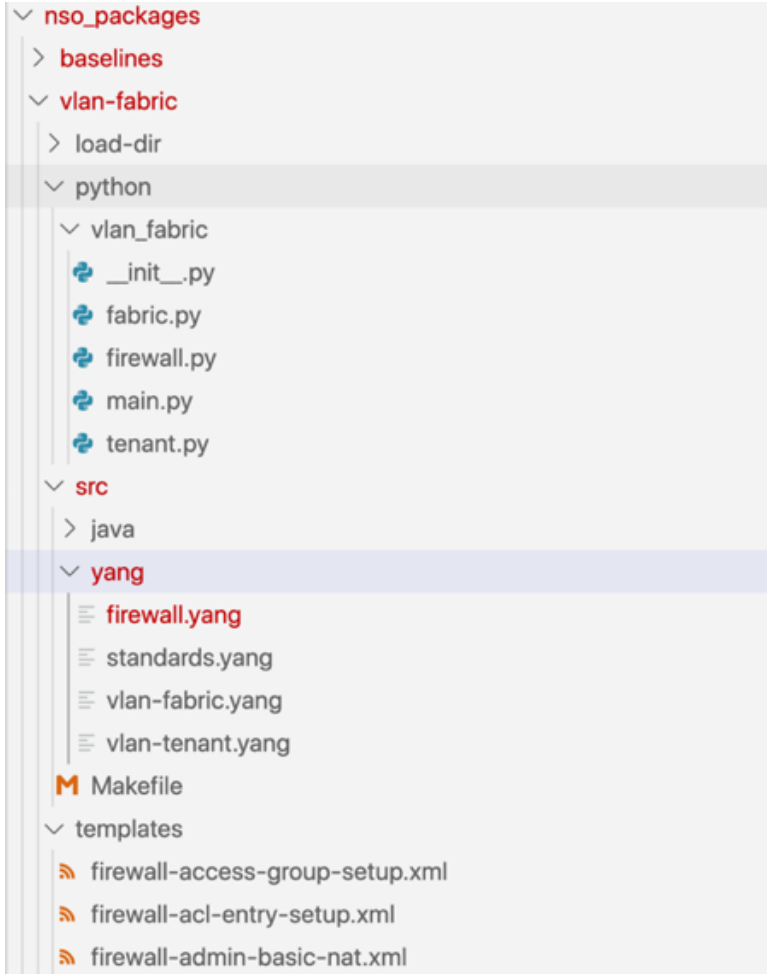


Multi-Domain Network Automation

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
Logo
Login
Password
Log in
Create account
Help
Default 1
CARD
ITEM 1
ITEM 2
ITEM 3
Default 2
Icon
NAME
Start screen
Log in
Help
Create account
personal info
login
password
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 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>
.
.
.
```

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

```
.
.
.
    <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>
```

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

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.

Being Serious about Network Automation Demands Dev/Test/Prod Environments

Dev → **Test** → **Pre-Prod** → **Prod**

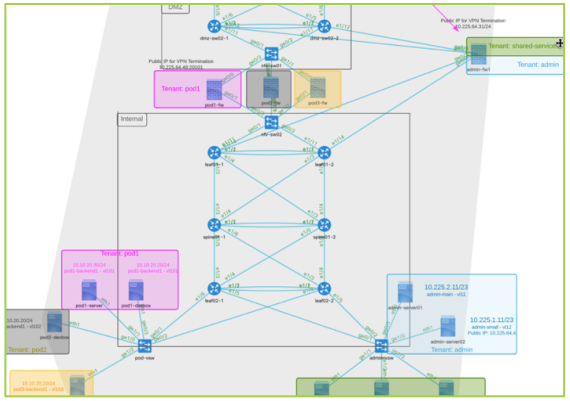
- Light weight
- Each engineer

- Virtualized
- Shared Resource

- Just like prod
- Only smaller

- End Customer
- Service Delivering

```
(std) ✔ ~/code/usw1-network-as-code/dev
08:21 $ make dev
Setting Up Netsim Instances...
DEVICE admin-fw1 CREATED
DEVICE admin-vsw CREATED
DEVICE dmz-fw01-1 CREATED
DEVICE dmz-rtr02-1 CREATED
DEVICE dmz-rtr02-2 CREATED
DEVICE dmz-sw01-1 CREATED
DEVICE dmz-sw01-2 CREATED
DEVICE dmz-sw02-1 CREATED
DEVICE dmz-sw02-2 CREATED
DEVICE leaf01-1 CREATED
DEVICE leaf01-2 CREATED
DEVICE leaf02-1 CREATED
DEVICE leaf02-2 CREATED
```



- Absolute requirement for successful achievement of goals
- Reduce risk, costs, and time to deliver
- Build confidence through iteration and pipeline

Lesson's Learned

Be flexible…


Finding balance can be hard

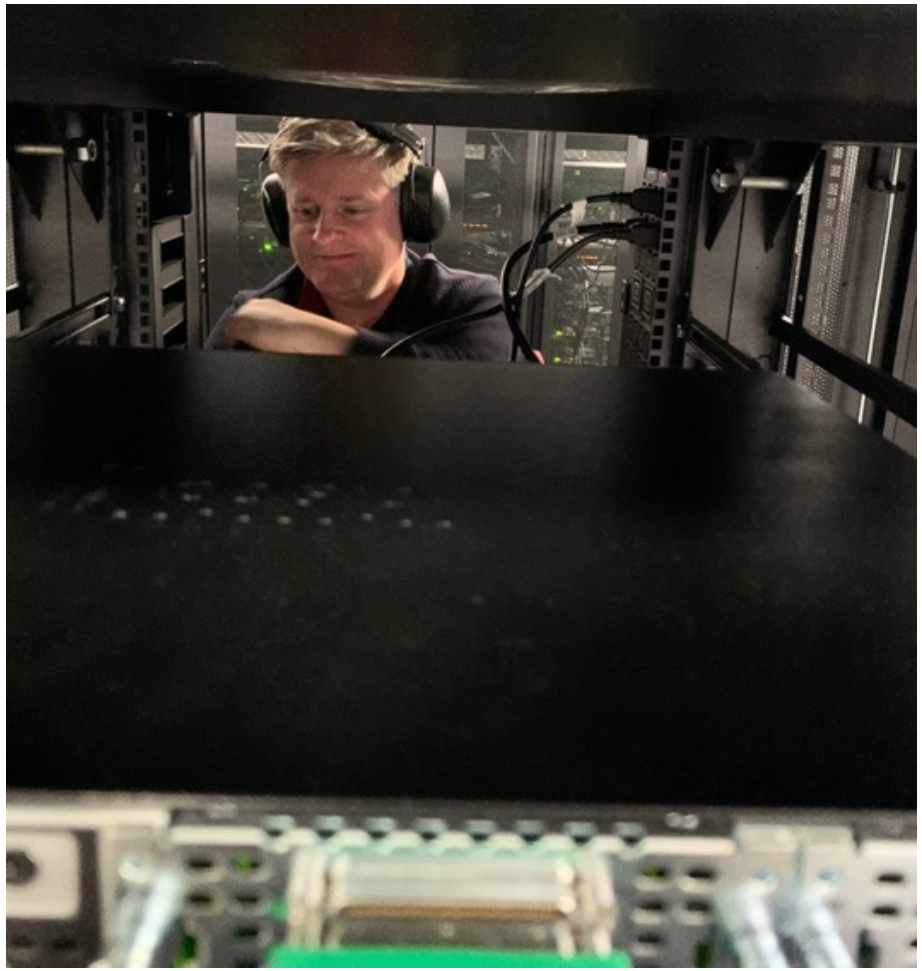
Performance tuning


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
- CICD for services
- Better "packaging"
- Train our engineers and build comfort



Got more questions? Stay in touch!

Hank Preston

hapresto@cisco.com

@hfpreston

hfpreston (Network to Code)

http://github.com/hpreston

Reference Links

- Code samples for the NSO services discussed
  - https://developer.cisco.com/codeexchange/github/repo/DevNetSandbox/sandbox-nso
- Blog posts discussing the use of networking services in Sandbox
  - https://blogs.cisco.com/developer/network-service-based-automation-1
  - https://blogs.cisco.com/developer/network-service-automation-2
- Free Learning Lab on NSO
  - https://developer.cisco.com/learning/modules/nso_fundamentals
- Documentation and free download for NSO
  - https://developer.cisco.com/nso