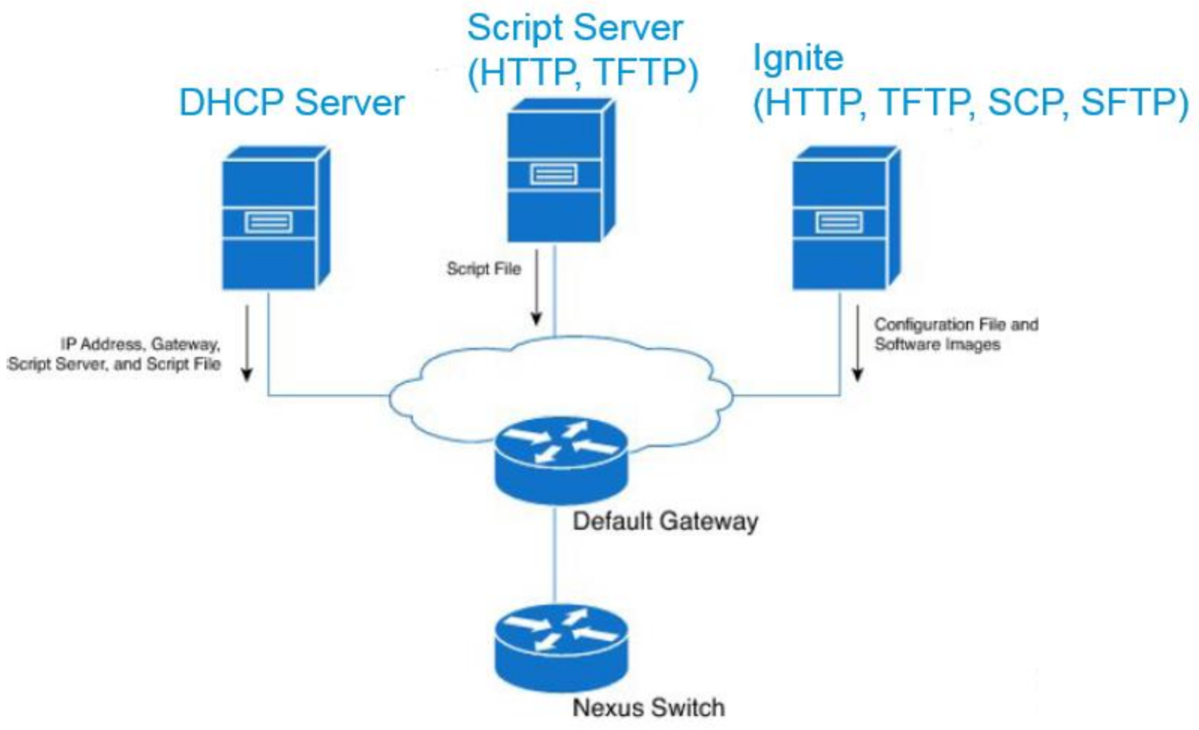
Ignite

Ignite automates the process of installing and upgrading software images and installing configuration files on Cisco Nexus 9000 and 3000 Series switches that are being deployed in the network.

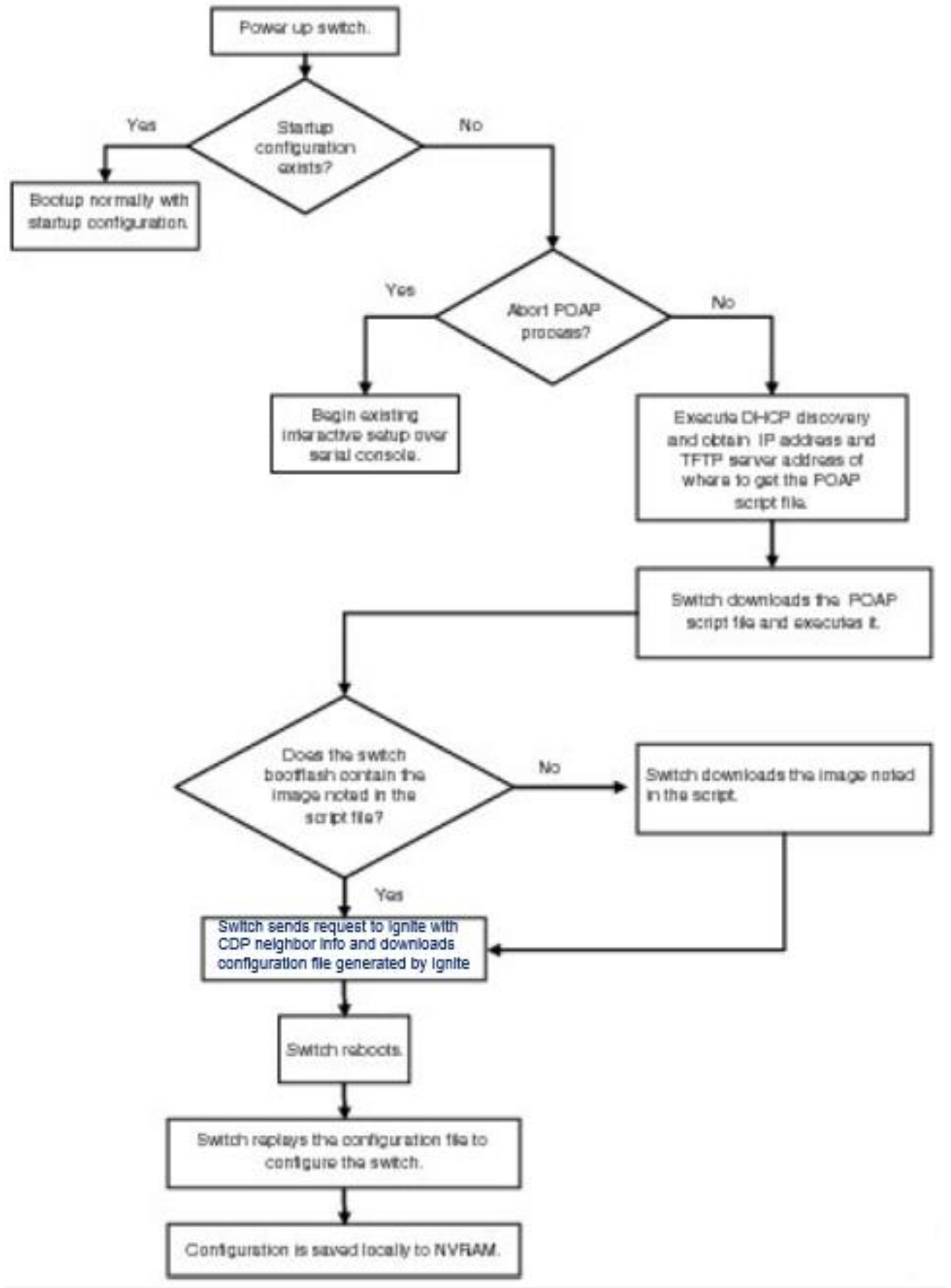
Overview

Cisco Nexus Power On Auto Provisioning using Ignite requires the following Network Infrastructure.

- A DHCP server to bootstrap the interface IP address, gateway address, and Domain Name System (DNS) server.
- A TFTP server that contains the configuration script used to automate the software image installation and configuration process. It is recommended that this script server also be installed in server running Ignite. Ignite_poap.py – startup script needed for interaction with Ignite is available in github repository.
- A server running Ignite that contains the desired software images and rules to dynamically build configuration files.



The Cisco Nexus POAP process has the following phases:


Power up switch.
Startup configuration exists?
Yes
No
Bootup normally with startup configuration.
Abort POAP process?
Yes
No
Begin existing interactive setup over serial console.
Execute DHCP discovery and obtain IP address and TFTP server address of where to get the POAP script file.
Switch downloads the POAP script file and executes it.
Does the switch bootflash contain the image noted in the script file?
No
Switch downloads the image noted in the script.
Yes
Switch sends request to Ignite with CDP neighbor info and downloads configuration file generated by Ignite
Switch reboots.
Switch replays the configuration file to configure the switch.
Configuration is saved locally to NVRAM.

1. Power up

   When you power up the device for the first time, it loads the software image that is installed at manufacturing and automatically enters a DHCP-based POAP discovery phase.

2. DHCP discovery

   The switch sends out DHCP discover messages on the front-panel interfaces or the MGMT interface that solicits DHCP offers from the DHCP server or servers. The DHCP client on the Cisco Nexus switch uses the switch serial number/MAC address in the client-identifier option to identify itself to the DHCP server. DHCP server assigns an IP address and responds with a DHCP offer, which includes other options to support the switch bootstrapping (example: script server, script filename to get the bootstrap script). Please refer to Cisco POAP documentation *http://www.cisco.com/c/en/us/td/docs/switches/datacenter/nexus9000/sw/6-x/fundamentals/configuration/guide/b_Cisco_Nexus_9000_Series_NX-OS_Fundamentals_Configuration_Guide/b_Cisco_Nexus_9000_Series_NX-OS_Fundamentals_Configuration_Guide_chapter_011.html* for additional details:

   Script server name or Script server address—DHCP server relays the Script server name or Script server address to DHCP client. DHCP client uses this information to contact the Script server to obtain script file. TFTP or HTTP may be used to download the script file.

   Script file name—DHCP server relays the script file name to the DHCP client (this file should be named "poap.py"). The boot file name includes complete path to script file on the Script server. DHCP client (switch) uses this information to download the script file. To take advantage of the dynamic configuration generation capabilities of Ignite server a special bootstrap script named "ignite_poap.py" is available. This script should be modified for the deployment environment and renamed to "poap.py" and stored in script server (see github README section on "Installing poap.py in the script server").

3. Script execution

   After device bootstraps itself using the information in the DHCP acknowledgement, script file is downloaded from the Script server.

This script (*poap.py derived from ignite_poap.py*) should be loaded in Script server for the switch to download it in this phase (*Note: best practice is to run the Script server on the Ignite VM*). When switch executes this script, it enables the switch interfaces and collects CDP neighbor adjacencies. The CDP neighbor information along with the system serial number, MAC address is then sent to Ignite in an HTTP GET request.

Ignite, acting as a server that handles these requests, attempts to identify this specific switch in its database, using the information provided in the request. If successful, the server builds configuration file for this switch and provides file details in its response. Details of how Ignite identifies the switch and builds the configuration file are explained in later sections.

During script execution, all events and errors are logged by the script both locally in the bootflash as well as in Ignite.

The script also downloads and installs the software image appropriate for this switch. Software image details are provided by Ignite server based on the image profile selected. See github README section, “Image Profiles” on how to set up image profiles.

The configuration file retrieved from Ignite saved as the “scheduled configuration” to be applied when the switch boots up with the correct image. On reload, the switch boots up with the new software image and the scheduled config is applied.

4. Post-installation reload

   On successful completion of reload, switch copies running configuration to startup configuration.

Ignite

Ignite provides a powerful and flexible framework through which a data center network administrator can define datacenter network in terms of topology, data center fabric, repository of switch discovery rules, configuration templates (configlets), configuration scripts and resource pools used to allocate IP Addresses/VLANs/identifiers used in provisioning network features. Ignite server automatically identifies the switch from its neighbors and links connecting to its neighbors, matching it with fabric topology information provided by the administrator. Ignite server then builds a configuration unique to this switch from rules provided by administrator using configlets, scripts and resource pools filling in with values provided in the fabric.

Topology

Topology is defined using:

- Switches used in topology (switch models, images)
- Links connecting the switches (number of links, type of links)
- vPC peers (leaf peers in the same VPC domain)
- switch name (name association to generate hostname of the switch)

Once the topology with its related components is defined, administrator can describe the fabrics deployed in the data center using this topology. Many fabrics will use the same topology.

Fabric

To define fabric(s), administrator associates the following attributes with it:

Name – base name which will be used in constructing host names for all the switches associated with this fabric(s)

Topology – topology previously defined by the administrator used by this fabric(s)

Replicas – used in constructing the host name of the switches. Replica number is appended to fabric name to which switch name from the topology is appended. For e.g. for fabric name = DeptA, replicas=2 and topology switch name = leaf3, following unique names are generated:

- fabric: DeptA replica 1 – **DeptA_1_leaf3**
- fabric: DeptA replica 2 – **DeptA_2_leaf3**

Locked – YES: topology changes are automatically reflected in this fabric instance, NO: fabric instance does not follow topology changes*1

Validate – YES: Ignite server validates the fabric connectivity and reports anomalies, NO: connectivity validation is not performed*1

***1 – future release**

Information available in fabric can be used, while building configurations for switches in this fabric. For example when two switches form a VPC domain, Ignite server knows the peer switches and peer links from the fabric topology. All the information required building the configuration for VPC peer, VPC peer interface, and VPC host links can be derived from the fabric topology and associated resource pools.

Following fabric topology information can be used in building configlets, scripts to dynamically generate configurations.

Parameter	Value
SWITCH_NAME	Switch Name as shown in Fabric details. Switch Name is constructed using Fabric name, replica number and name of the switch from fabric topology.
HOST_INTERFACE*	Access Interfaces connected to servers or other hosts. *Not available currently
VPC_PEER_LINK_IF_NAMES	Names of VPC peer link Interfaces
VPC_PEER_DST	IP Address of the VPC neighbor to which this switch is connected to
VPC_PEER_SRC TRUNK_PORTS	IP address assigned to this switch Interfaces used between Spine and Leaf. Spine switch will refer to interface names used to connect to Leaf switch and Leaf switch will refer to interface names used to connect to Spine Switch.

Configlets, Pool and Configurations

Configlets, pool and configurations are used to define rules to generate startup configurations for the switches.

Configurations are built using configlets, python scripts and configuration snippets received from external systems. The administrator defines:

- sequence in which these get assembled
- substitution rules to replace parameters with actual values

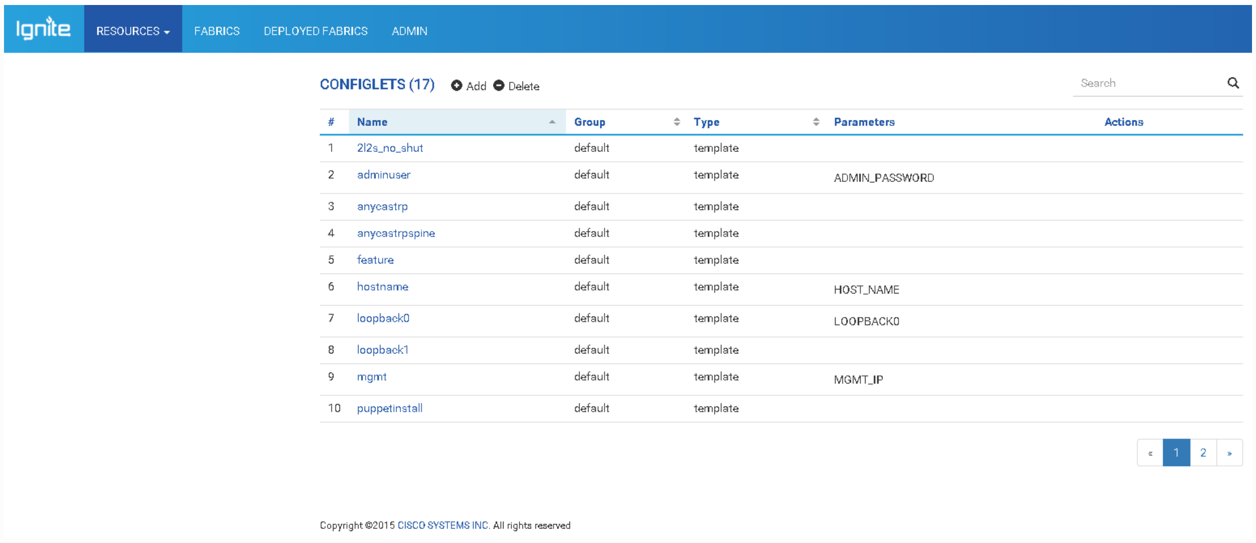
Parameter values can be derived from fabric topology, values allocated from Pool and user provided values.

Configlets

Configlets are configuration templates. These contain sequence of Nexus 9000 CLI commands with parameterized values. When using these configlets to build a configuration, administrator specifies how to substitute the values. The various options available are:

- FIXED – a constant value provided by administrator
- POOL – a value allocated by the server from a pool of values defined by administrator
- INSTANCE – a value generated from fabric topology (switch names, port numbers and IP addresses)
- VALUE – value referenced by the parameter name (typically used when the parameter value has already been associated from an earlier assignment)
- AUTO – value generated automatically starting from a starting value.

A configlet is created by importing a text file containing the parameterized configuration commands.



CONFIGLETS (17) Add Delete

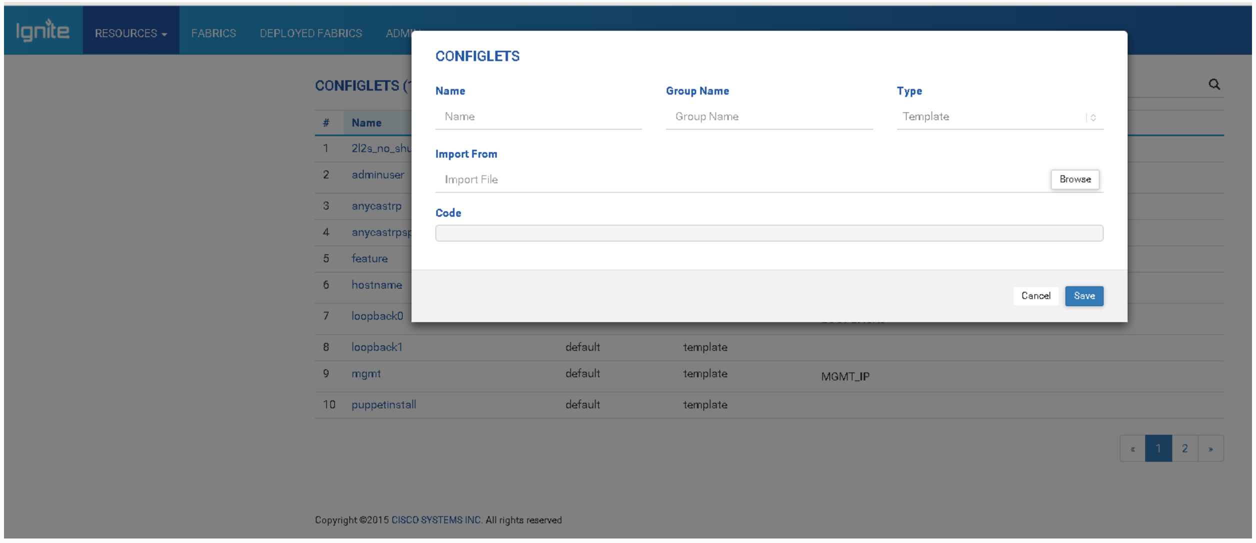
Search

#	Name	Group	Type	Parameters	Actions
1	2l2s_no_shut	default	template		
2	adminuser	default	template	ADMIN_PASSWORD	
3	anycastrp	default	template		
4	anycastrpspine	default	template		
5	feature	default	template		
6	hostname	default	template	HOST_NAME	
7	loopback0	default	template	LOOPBACK0	
8	loopback1	default	template		
9	mgmt	default	template	MGMT_IP	
10	puppetinstall	default	template		

« 1 2 »

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

In the Resources select “CONFIGLETS”. List of configlets present in the server are displayed. To add a new configlet, select ***add*** button and follow the prompts.



CONFIGLETS

Name	Group Name	Type
Name	Group Name	Template

Import From

Import File Browse

Code

Cancel Save

Assign a name to the configlet. Configlet can be a template or a script. Template is a text file containing CLI commands and script is a python script which the server runs to generate CLI commands.

An example template:

```
interface loopback0
  ip address $$LOOPBACK0$$
  ip router ospf 1 area 0.0.0.0
  ip pim sparse-mode
```

The template shown above generates CLIs required to define a loopback interface. IP Address for loopback interface is a parameter $$LOOPBACK0$$, and user can assign a value to this, when this configlet is used to build a Configuration.

An example script:

```
for TRUNK_PORT in $$TRUNK_PORTS$$:
    print "interface ", TRUNK_PORT
    print "no switchport"
    print "ip router ospf 1 area 0.0.0.0"
    print "ip pim sparse-mode"
    print "no shutdown"
```

Python script shown above generates a set of CLI commands to set up all trunk interfaces (spine <-> leaf interfaces) on this switch. Parameter $$TRUNK_PORT$$ is a list of interfaces connecting leaf to spine (spine to leaf). When this configlet is used in Configuration, user can assign a value INSTANCE.TRUNK_PORT which will automatically provide list of interfaces connecting this switch to switches in layer above or in layer below.

Pool

Pool provides a user defined collection of values used in allocating IP Addresses, VLANs and Identifiers.

Pool has a scope.

- Global - values from the pool are used globally. All switches defined in the Ignite server share this global pool. For example, management IP addresses will be in a global pool, so that every switch in the data center gets a unique IP address.
- Fabric – every fabric defined in the server has its own pool with the same values. Switches within a fabric share values from the fabric specific pool. For example,

backbone VLAN Ids need to be unique across the fabric and this pool can be defined with a “fabric” scope.

- Switch – Each switch defined in the Ignite Server has its own pool with the same values. For example, each switch uses port channels. Each port channel requires an identifier. This identifier is unique with in a switch. Identifier can be reused in another switch. In this case, administrator can define a pool called *“portchannelidentifier”* of type integer and assign *“switch”* scope.

***Note:** Fabric and Switch scope are planned to be supported in a future release.*

Pools with following value types are supported:

- IP – IPv4 address pool
- IPv6 – IPv6 address pool
- Integer – integer values
- MgmtIP – Management IP address pool
- VLAN – vlan ids pool

Configurations

Configurations are built using a sequence of configlets, scripts and external system provided inputs*2. Ignite server provides the following constructs to build a configuration.

*2 – future release

Append configlet – allows administrator to choose a configlet from the repository and append it to the current configuration. When administrator appends a configlet, server processes the configlet file and identifies all the parameters required by the configlet. Parameters are identified within delimiters $$ and $$.

Example: `user name admin password $$ADMIN_PASSWORD$$`. “ADMIN_PASSWORD” is a parameter in this configlet.

Administrator will be prompted to define how the value gets assigned to each parameter. Administrator can select different value generation methods explained in ***Configlets*** section.

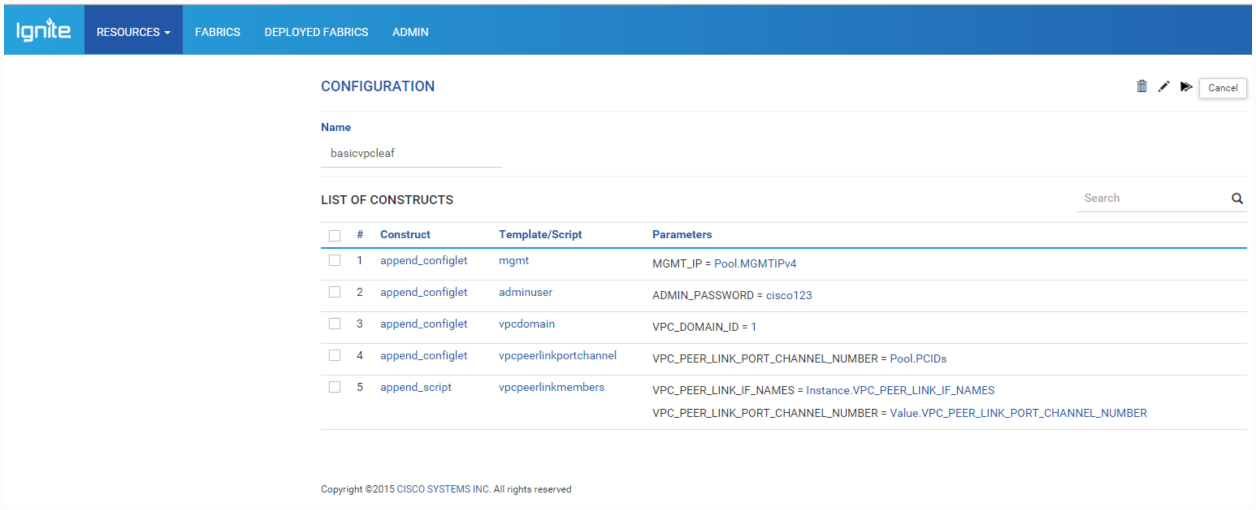
Append script – allows administrator to choose a python script from the repository and append it to the current configuration. When administrator appends a script, server processes the script file and identifies all the parameters required by the script. Administrator will then be prompted to define how the value gets assigned to each parameter. Administrator can select different value generation methods explained in ***Configlets*** section.

Example script:

```
for VPC_PEER_LINK_IF_NAME in $$VPC_PEER_LINK_IF_NAMES$$ :

    print "interface", VPC_PEER_LINK_IF_NAME

    print "switchport mode trunk"

    print "channel-group $$VPC_PEER_LINK_PORT_CHANNEL_NUMBER$$ mode active"


print "interface port-channel $$VPC_PEER_LINK_PORT_CHANNEL_NUMBER$$"

print "vpc peer-link"
```

In the above script, `VPC_PEER_LINK_IF_NAMES` and `VPC_PEER_LINK_PORT_CHANNEL_NUMBER` are the parameters, where `VPC_PEER_LINK_IF_NAMES` is a list which contains interface names of the ports connecting peer leaf switches. When this script is used in a configuration, administrator can assign value for `VPC_PEER_LINK_IF_NAMES` simply using `Instance.VPC_PEER_LINK_IF_NAMES`. Here the Ignite server will determine the peer links connected between switch requesting configuration and its peer from fabric topology.

An example configuration definition in Ignite is shown below:



CONFIGURATION

Name

basicvpcleaf

LIST OF CONSTRUCTS

#	Construct	Template/Script	Parameters
1	append_configlet	mgmt	MGMT_IP = Pool.MGMTIPv4
2	append_configlet	adminuser	ADMIN_PASSWORD = cisco123
3	append_configlet	vpcdomain	VPC_DOMAIN_ID = 1
4	append_configlet	vpcpeerlinkportchannel	VPC_PEER_LINK_PORT_CHANNEL_NUMBER = Pool.PCIDs

5	append_script	vpcpeerlinkmembers	VPC_PEER_LINK_IF_NAMES = Instance.VPC_PEER_LINK_IF_NAMES
			VPC_PEER_LINK_PORT_CHANNEL_NUMBER = Value.VPC_PEER_LINK_PORT_CHANNEL_NUMBER

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

The configuration named ***basicvpcleaf*** is built using 5 ***configlets, mgmt, adminuser, vpcdomain, vpcpeerlinkportchannel, and vpcpeerlinkmembers***. Parameters corresponding to each configlet and values assigned by the administrator are also shown in the configuration.

Profile Templates, Fabric Profiles

Profile Templates and Fabric Profiles are constructs which will enable Ignite server to interface with AutoNetkit. Autonetkit is a network management tool, which automatically generates network configurations from a topology. It extracts the topology information and builds configurations required to enable BGP, OSPF protocols between network elements. It also has the ability to allocate IP addresses required for such configurations. It is a highly flexible and configurable tool for building network configurations for mutli-device, multi-vendor networks.

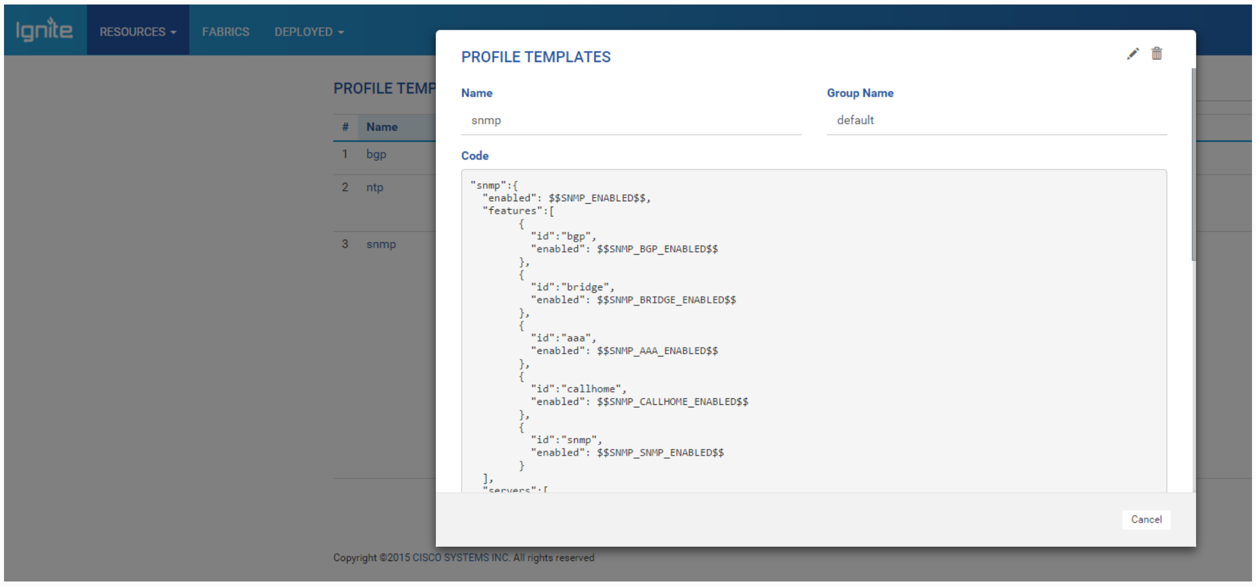
Since Ignite supports a topology based network discovery and configuration generation, it could easily leverage AutoNetkit capabilities to automatically generate complex configurations required for overlay and underlay networks in a data center. Ignite uses a modified version of AutonNetkit which allows passing the user defined topology and configuration requirements in a JSON format to build necessary configurations.

Profile Templates are AutoNetkit JSONs which specify a method to communicate certain parameters and/or inputs required by ANK to generate a configuration.

As an example JSON, to configure bgp in network elements an ASN number is associated with every switch involved in bgp routing. Based on the topology adjacencies and ASN number, AutoNetkit generates the appropriate configurations.

```
"bgp":
        {
           "asn":"$$asn_number$$"
        }
```

Once this profile template for bgp is created, it now needs to be associated with each switch in the topology which will run BGP. Associating or applying a Profile Template to a switch or a set of switches is accomplished by building a Fabric Profile which contains a number of Profile Templates and then associating Fabric Profile with the required switches in Fabrics view.



Name: snmp

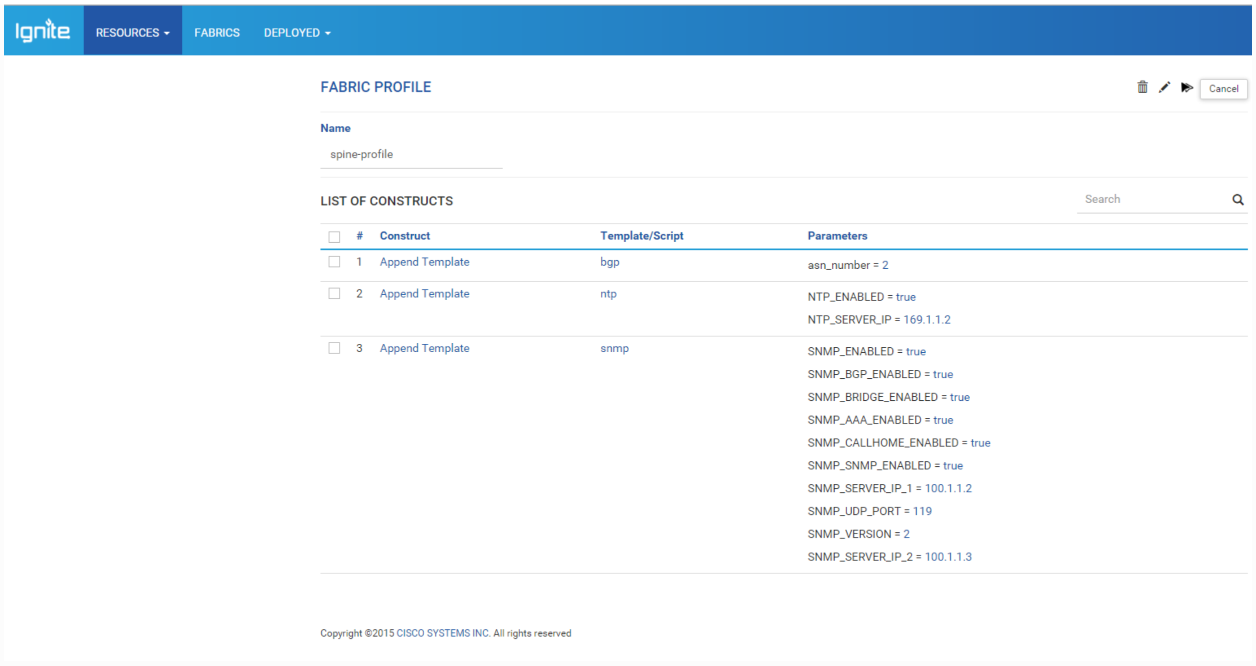
Group Name: default

Code

```
"snmp":{
  "enabled": $$SNMP_ENABLED$$,
  "features":[
        {
          "id":"bgp",
          "enabled": $$SNMP_BGP_ENABLED$$
        },
        {
          "id":"bridge",
          "enabled": $$SNMP_BRIDGE_ENABLED$$
        },
        {
          "id":"aaa",
          "enabled": $$SNMP_AAA_ENABLED$$
        },
        {
          "id":"callhome",
          "enabled": $$SNMP_CALLHOME_ENABLED$$
        },
        {
          "id":"snmp",
          "enabled": $$SNMP_SNMP_ENABLED$$
        }
  ],
  "servers":[
```

Fabric Profile is a collection of Profile Templates. For example, each switch in the network in addition to running BGP also requires NTP, SNMP, OSPF, VXLAN configurations. Fabric Profile allows to bundle together these profile templates and create a single profile, which then can be associated or applied to a group of similar switches.

Example shows a Fabric Profile named “Spine Profile”, which groups together BGP, SNMP and NTP profile templates.



FABRIC PROFILE

Name

spine-profile

LIST OF CONSTRUCTS

#	Construct	Template/Script	Parameters
1	Append Template	bgp	asn_number = 2

2	Append Template	ntp	NTP_ENABLED = true
			NTP_SERVER_IP = 169.1.1.2

3	Append Template	snmp	SNMP_ENABLED = true
			SNMP_BGP_ENABLED = true
			SNMP_BRIDGE_ENABLED = true
			SNMP_AAA_ENABLED = true
			SNMP_CALLHOME_ENABLED = true
			SNMP_SNMP_ENABLED = true
			SNMP_SERVER_IP_1 = 100.1.1.2
			SNMP_UDP_PORT = 119
			SNMP_VERSION = 2
			SNMP_SERVER_IP_2 = 100.1.1.3

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

Discovery Rule

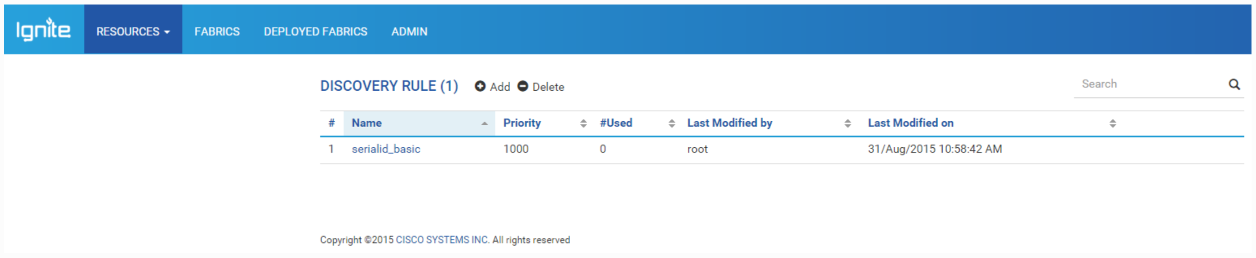
Discovery Rules are used by Ignite to identify a specific switch or group of switches. Ignite will search through the discovery rules till one rule matches or no rules match. If a rule matches, the configuration associated with that discovery rule is used to build the startup configuration. Administrator can specify how to build startup configuration for the identified switch(s), using configurations from resource repository.

A discovery rule is built using a sequence of match expressions. Match expressions will use inputs received from requesting switch to compare values provided by administrator. If multiple match expressions are used in a rule, administrator can decide when the rule is successful - “Any” match expression returns success or “All” match expressions return success.

Discovery Rule - Switch Serial Number

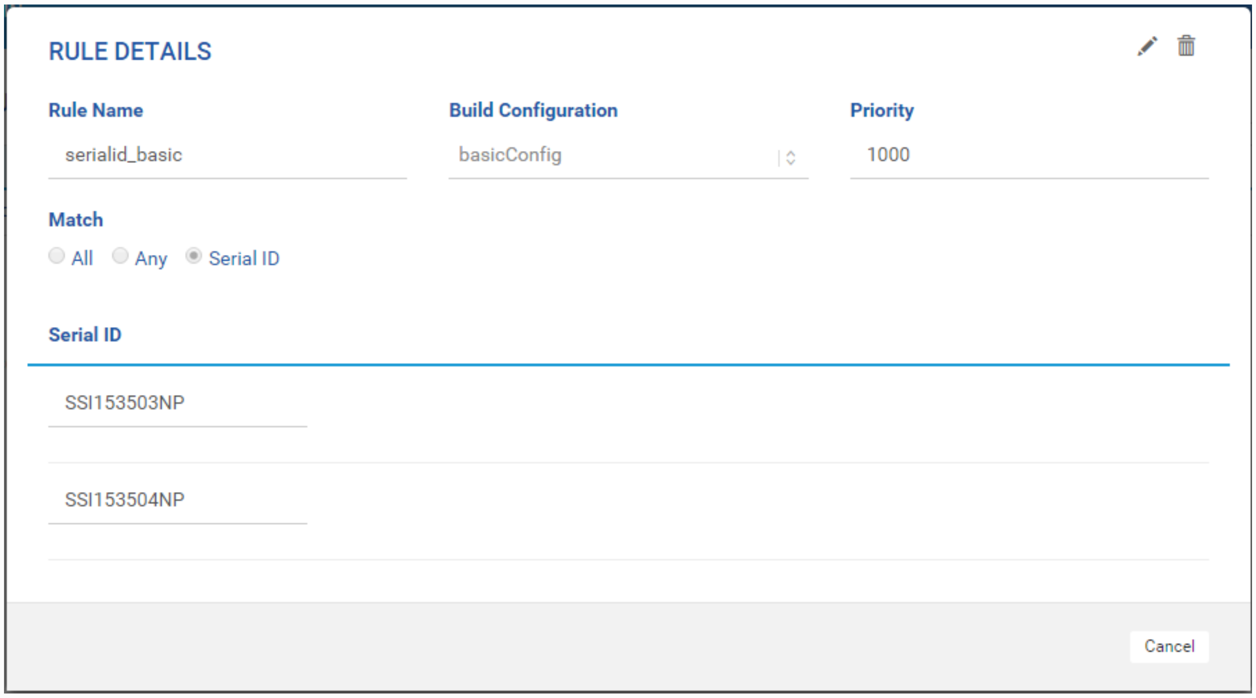
Discovery Rule can be used to match Serial Number of a switch to identify it, and assign a startup configuration.

In the example below a discovery rule named "`serialid_basic`" is created. This rule assigns `basicConfig` configuration to two switches with serial numbers `SSI153503NP`, `SSI153504NP`.



#	Name	Priority	#Used	Last Modified by	Last Modified on
1	serialid_basic	1000	0	root	31/Aug/2015 10:58:42 AM

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved



Rule Name: serialid_basic
Build Configuration: basicConfig
Priority: 1000
Match: All Any [✓ Serial ID]
Serial ID
SSI153503NP
SSI153504NP

Discovery Rule – CDP Neighbor

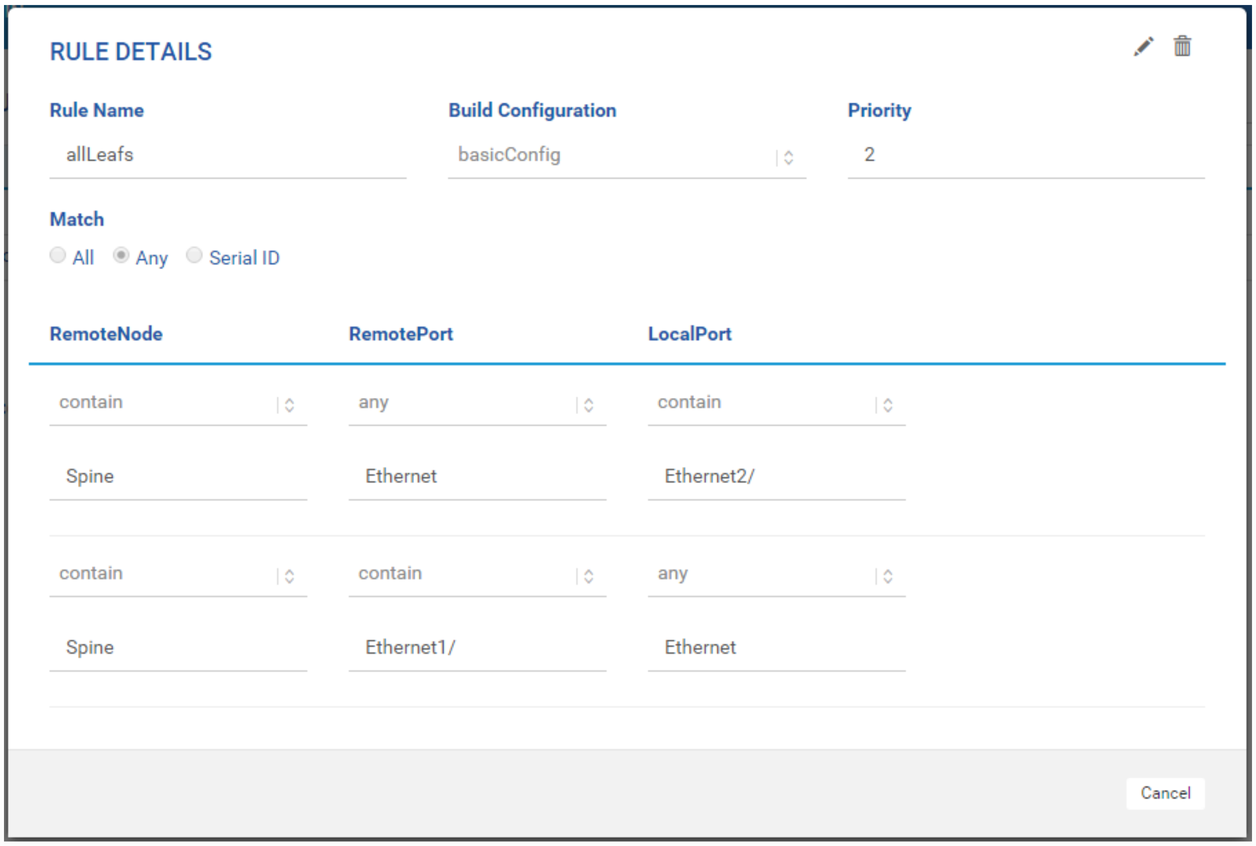
Discovery Rules can use the CDP neighbor data provided by the requesting Switch to make an identification of the switch and decide on a proper initial configuration. Discovery Rule allows configuring multiple match expressions, to compare the remote node, remote port and local port CDP data and then decide on the identity. The rule could say either "All" match expressions be successful or "Any" match expressions be successful to conclude the identification and build the configuration. Multiple such discovery rules can be defined each with a priority. Ignite will scan through the rules in the order of priority (rules with same priority are scanned in the order they are entered) till the match is success or no rules match.

Following match constructs are provided to build the discovery rules:

- contain – contains a string defined by user e.g. `contain “spine”`
- no_contain – does not contain a string defined by user e.g. `no_contain “leaf-5”`
- match – matches a string or a regular expression
- no_match – does not match a string or a regular expression
- any – any value *(Note: when match construct is any, parameter value should also be made any)*
- oneof – any one of the values defined in parenthesis e.g. `oneof(“Ethernet1/1”, “Ethernet2/1”, “Ethernet3/1”)`
- none – no value is present

On a successful rule match, Ignite will start building the configuration file using the definition provided by administrator for *Build Configuration,* named in the rule.

In the example below a discovery rule named “`allLeafs`” is defined. The rule has two match expressions and the success criteria for the rule is “`Any`”. First match expression compares the “`RemoteNode`” to “`contain`” a string “`Spine`”, “`RemotePort`” to match “`any`” string and “`LocalPort`” to contain “`Ethernet2/`”. Second expression, compares the “`RemoteNode`” to contain a string “`Spine`”, “`RemotePort`” to contain “`Ethernet1/`” and “`LocalPort`” to contain “`Ethernet1/`”. The rule is successful for a leaf switch whose uplink is “`Ethernet2/1`” or “`Ethernet2/2`” or “`Ethernet2/*`” and is connected to spine switch named “`PodA_Spine1`” on any port. The rule is also successful for a leaf switch who has a remoted node “`PodA_Spine1`” and any of its uplinks are connected to this spine switch on port “`Ethernet1/1`” or “`Ethernet1/2`” or “`Ethernet1/*`”. Here the remote node name “`PodA_Spine1`” contains the string “`Spine`” as defined in the discovery rule “`allLeafs`”. On successful match, Ignite will build the startup configuration using steps provided in “`basicConfig`”.


RULE DETAILS

Rule Name	Build Configuration	Priority
allLeafs	basicConfig	2

Match

All [✓ Any] Serial ID

RemoteNode	RemotePort	LocalPort
contain	any	contain
Spine	Ethernet	Ethernet2/
contain	contain	any
Spine	Ethernet1/	Ethernet

Fabric based discovery

Global discovery rules are required to identify a switch which is not defined in a Fabric. Ignite automatically builds rules to identify a switch in a fabric, as the topology is known to it. Following steps are used in Fabric based topology discovery:

1. From CDP neighbor information received from requesting switch, Ignite takes all remote node names and searches its database to see if the name contains a string which matches a fabric name + replica number + switch name defined in Fabric database.
2. If any of the remote node names are present in Ignite database, it will take the <remote port, local port> tuples from request and scan through fabric topology data of the identified fabric and see if this matches with any topology edges.
3. If a match is found, switch name associated with local port in fabric topology is used as the switch identity.
4. Switch identity derived from step 3, is used to determine the startup configuration. When a fabric is defined, administrator also assigns a "Build

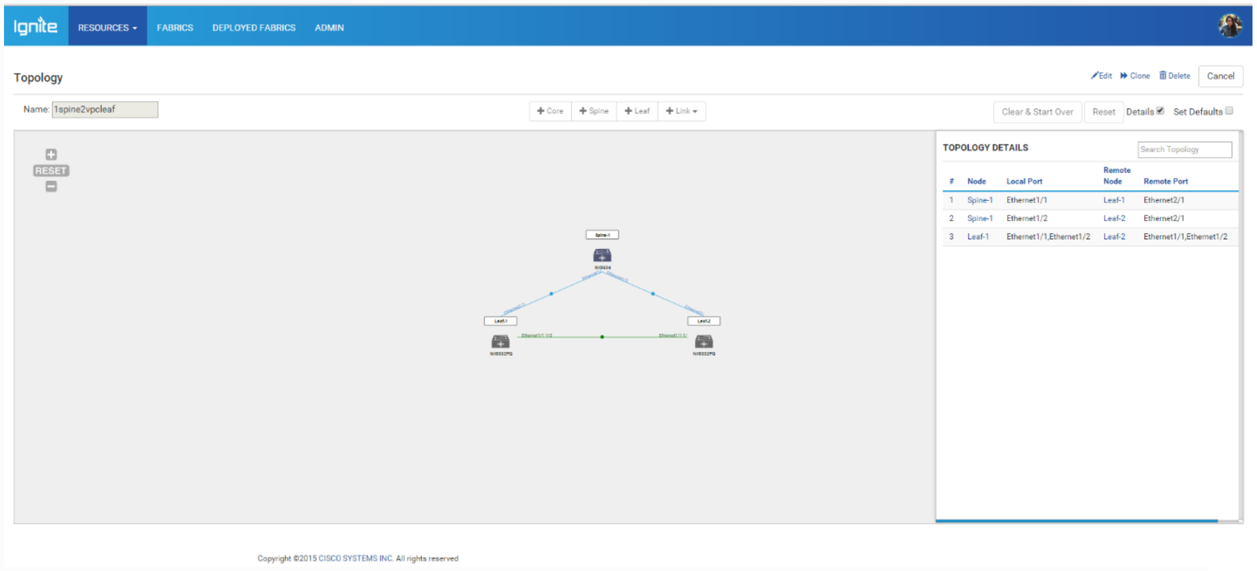
Configuration" to it. Ignite will start building configuration file based on the steps defined in the assigned "Build Configuration".

5. If a match is not found in step 3, serial ID provided in the request is searched to see if this identity is assigned to any switches in the fabric topology through "Switch Identity". Switch Identity maps a serial number to a specific switch (e.g. spine-n or leaf-n) in the fabric topology.
6. If a match is not found in step 3 or step 5, Ignite will use "Discovery Rules" to identify the switch and assign a configuration.

In the following illustration, fabric discovery is explained, step by step.

(a) Create a topology

Administrator creates a topology which has 1 spine and 2 leaf switches. Leaf switches form a VPC domain.



Name: 1spine2vpcleaf

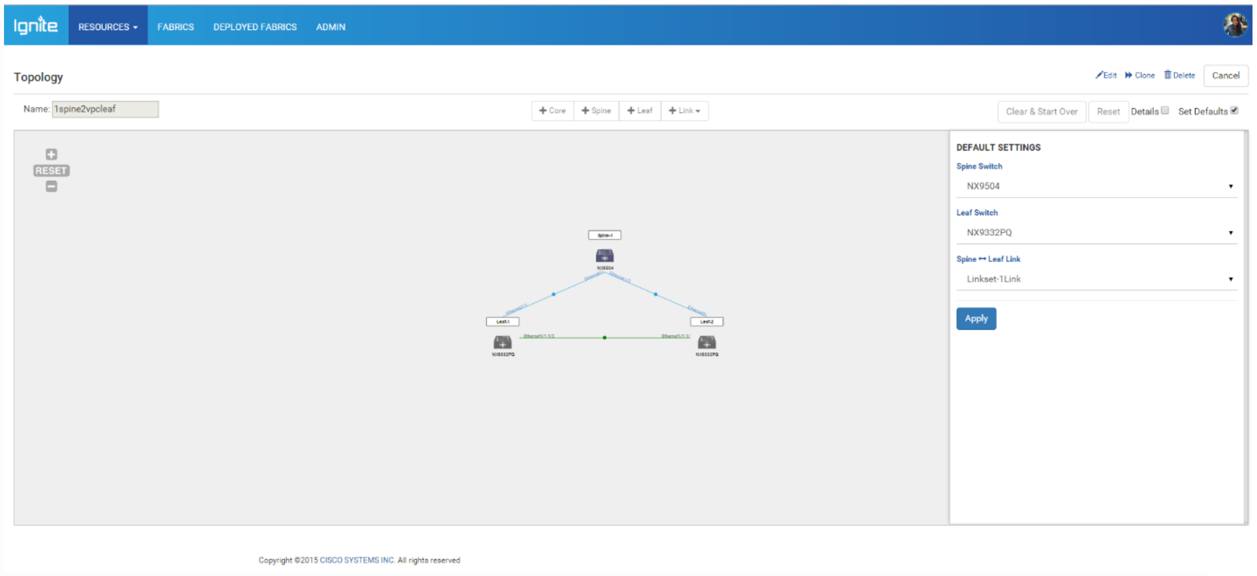
Details ✓ Set Defaults ☐

TOPOLOGY DETAILS

#	Node	Local Port	Remote Node	Remote Port
1	Spine-1	Ethernet1/1	Leaf-1	Ethernet2/1
2	Spine-1	Ethernet1/2	Leaf-2	Ethernet2/1
3	Leaf-1	Ethernet1/1,Ethernet1/2	Leaf-2	Ethernet1/1,Ethernet1/2

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

Administrator first adds a spine by clicking on `+ Spine` and clicking on `+ Leaf` twice to add 2 leaf switches. Ignite uses default settings[*3] to select Spine switch model, Leaf switch model and the number of uplinks between Spine and Leaf. In this example the defaults are set to Spine Switch NX9504, Leaf Switch NX9392PQ, Spine<->Leaf Link as Linkset-1Link (40 G x 1 link). The ports between leaf and spine are numbered automatically. For leaf switch, it uses the ports from Ethernet2/1 and for spine switch it uses the ports from Ethernet1/1.



Topology

Name: 1spine2vpcleaf

Details ☐ Set Defaults ☑

DEFAULT SETTINGS

Spine Switch: NX9504

Leaf Switch: NX9332PQ

Spine ↔ Leaf Link: Linkset-1Link

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

User can view the defaults by checking “Set Defaults” in the Topology design page. See the figure above.

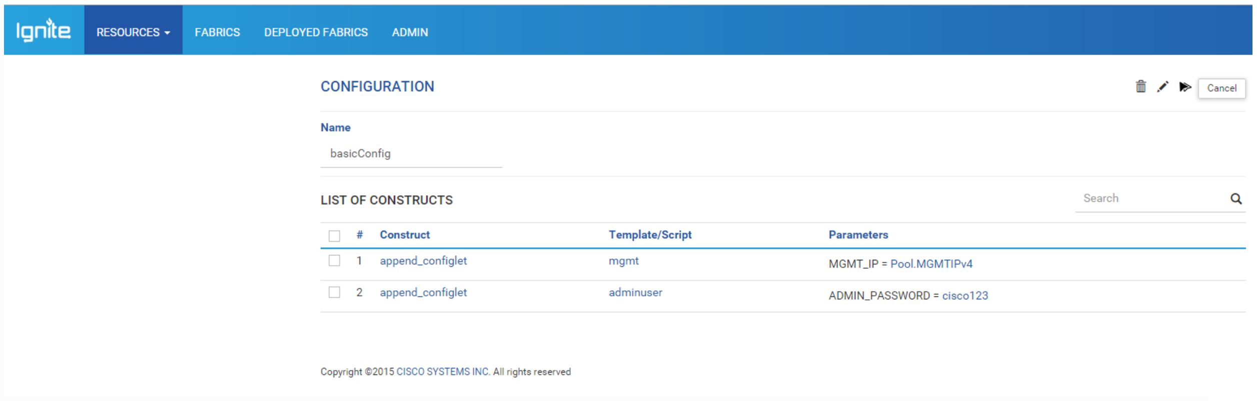
(b) create configlets and configuration

For the above topology, we need a minimum of 2 configurations (1) which gets applied to Spine and (2) which gets applied to VPC leafs. If the topology has leafs without VPC then another configuration for the same may be needed.

Administrator creates two configurations “basicConfig” and “basicvpcleaf”.

See the illustration below for the details:

basicConfig



CONFIGURATION

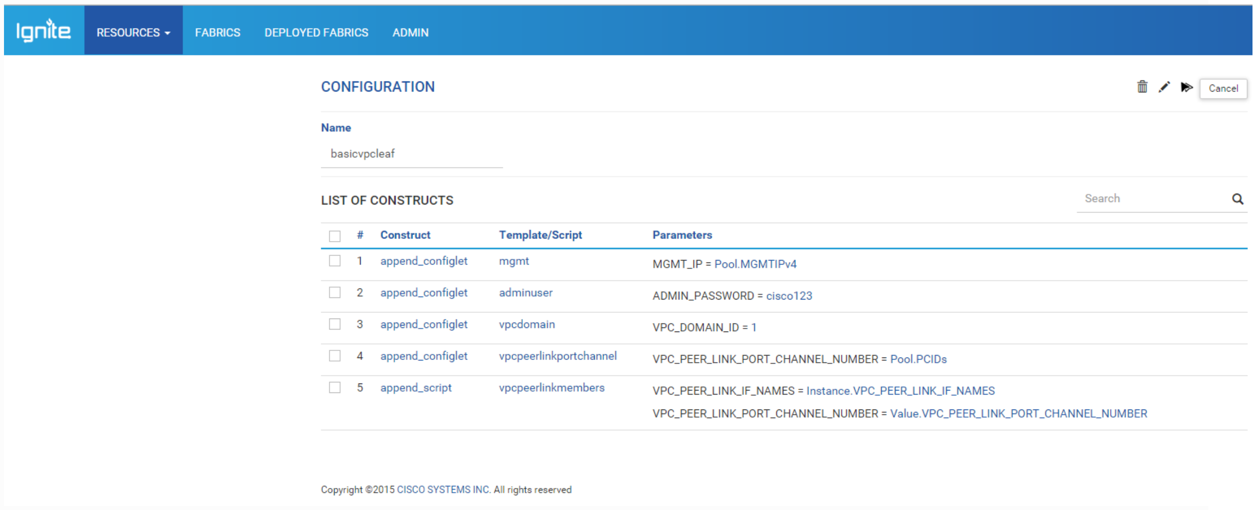
Name
basicConfig

LIST OF CONSTRUCTS

#	Construct	Template/Script	Parameters
1	append_configlet	mgmt	MGMT_IP = Pool.MGMTIPv4
2	append_configlet	adminuser	ADMIN_PASSWORD = cisco123

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

Basicvpcleaf config



CONFIGURATION

Name
basicvpcleaf

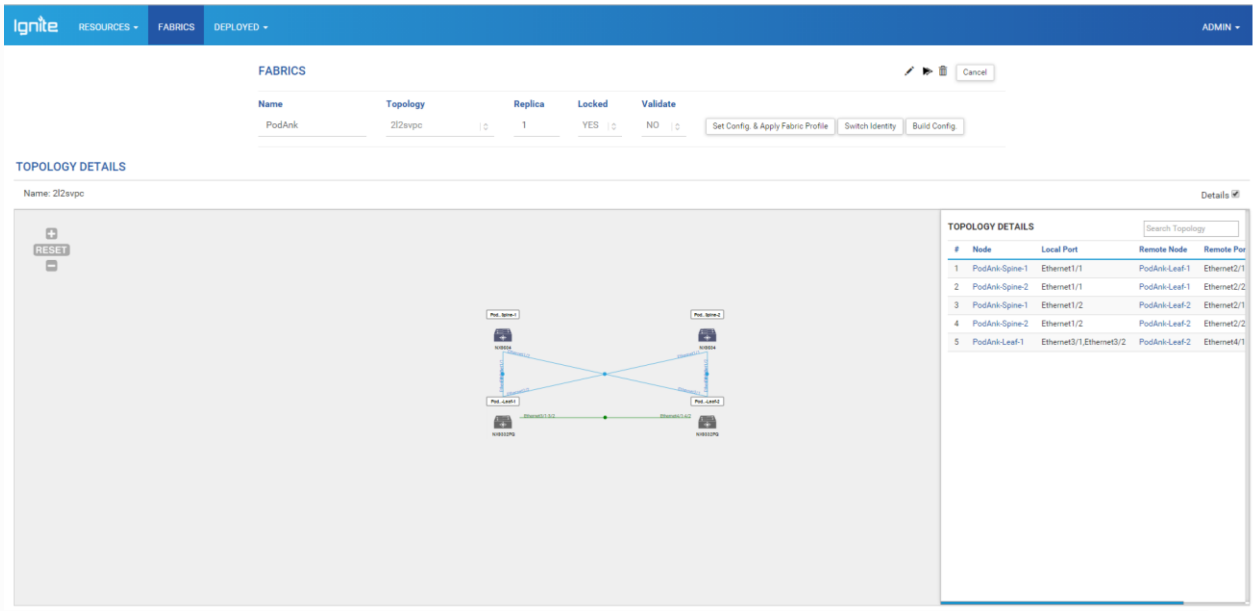
LIST OF CONSTRUCTS

#	Construct	Template/Script	Parameters
1	append_configlet	mgmt	MGMT_IP = Pool.MGMTIPv4
2	append_configlet	adminuser	ADMIN_PASSWORD = cisco123
3	append_configlet	vpcdomain	VPC_DOMAIN_ID = 1
4	append_configlet	vpcpeerlinkportchannel	VPC_PEER_LINK_PORT_CHANNEL_NUMBER = Pool.PCIDs

5	append_script	vpcpeerlinkmembers	VPC_PEER_LINK_IF_NAMES = Instance.VPC_PEER_LINK_IF_NAMES
			VPC_PEER_LINK_PORT_CHANNEL_NUMBER = Value.VPC_PEER_LINK_PORT_CHANNEL_NUMBER

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

(c) Administrator now defines a Fabric, which uses the topology “1spine2vpcleaf”. Once the Fabric is defined, it is possible to associate which physical switches are present in this fabric, through “Switch Identity” button. Associating Serial ID of a switch with a specific Fabric switch helps to bring up that switch with right configuration required for its role, when no other switches in the Fabric are booted. “Set Config and Apply Fabric Profile” allows to set user defined configurations created through configlets/scripts, AutoNetkit Fabric Profiles, and Image profiles. Once configuration and profiles are defined, use “Build Config” to generate the configuration file which will get downloaded to the switch when it sends the boot request. “Build Config”, will initiate AutoNetkit configuration generation process by sending JSONs associated with “Profile Templates”, which are part of “Fabric Profile”.



FABRICS

Name	Topology	Replica	Locked	Validate
PodAnk	2l2svpc	1	YES	NO

TOPOLOGY DETAILS

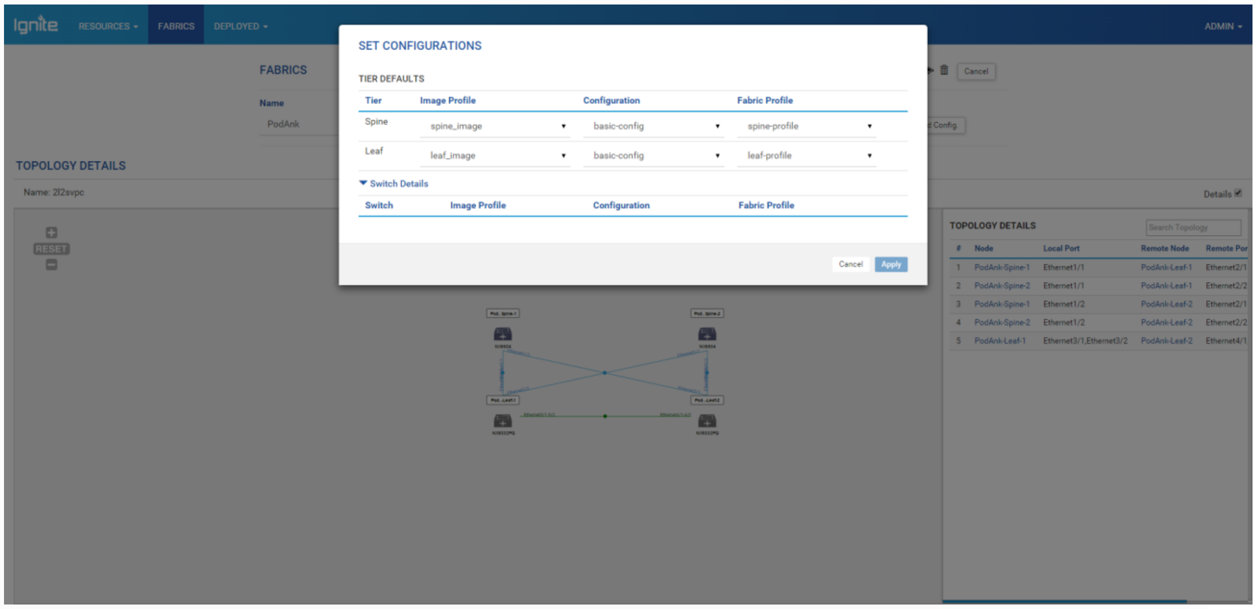
Name: 2l2svpc

Details ✓

TOPOLOGY DETAILS

#	Node	Local Port	Remote Node	Remote Por
1	PodAnk-Spine-1	Ethernet1/1	PodAnk-Leaf-1	Ethernet2/1
2	PodAnk-Spine-2	Ethernet1/1	PodAnk-Leaf-1	Ethernet2/2
3	PodAnk-Spine-1	Ethernet1/2	PodAnk-Leaf-2	Ethernet2/1
4	PodAnk-Spine-2	Ethernet1/2	PodAnk-Leaf-2	Ethernet2/2
5	PodAnk-Leaf-1	Ethernet3/1,Ethernet3/2	PodAnk-Leaf-2	Ethernet4/1

SET CONFIGURATIONS dialog box allows defining Image Profile, Configuration and Fabric Profile to all switches in Spine tier, all switches in leaf tier and/or individual leaf or spine switches. Tier level definitions are overridden by individual switch level definitions.



SET CONFIGURATIONS

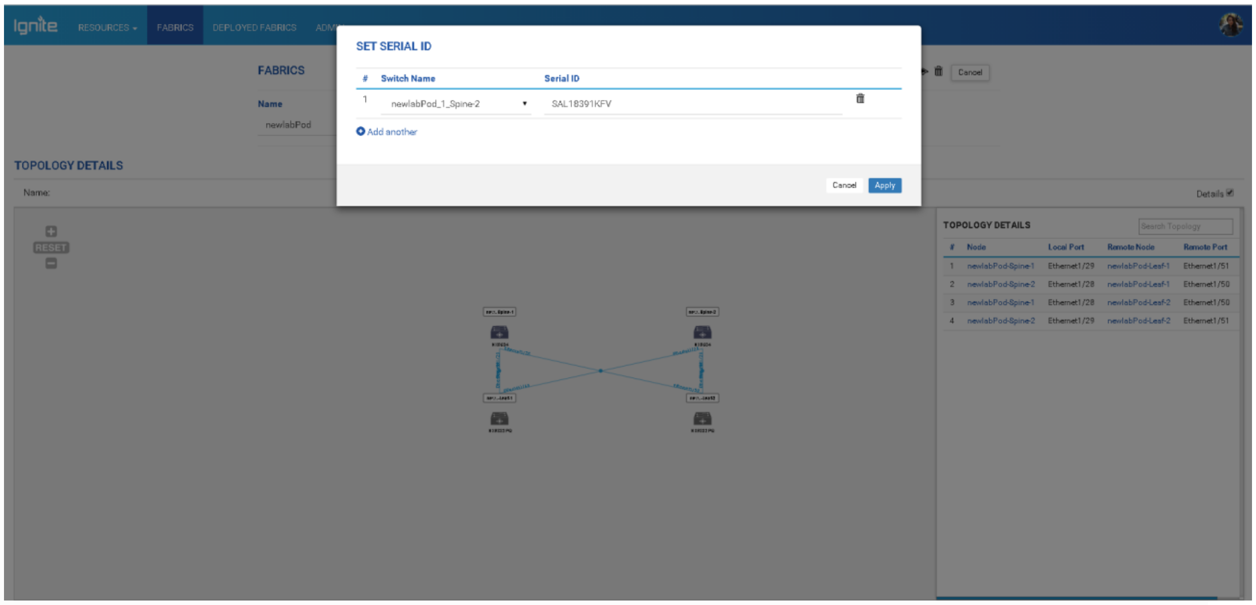
TIER DEFAULTS

Tier	Image Profile	Configuration	Fabric Profile
Spine	spine_image	basic-config	spine-profile
Leaf	leaf_image	basic-config	leaf-profile

▼ Switch Details

Switch	Image Profile	Configuration	Fabric Profile

(d) Use “Switch Identity” to map a serial ID to a specific switch in the fabric topology:



SET SERIAL ID

#	Switch Name	Serial ID
1	newlabPod_1_Spine-2	SAL18391KFV

Add another

Cancel Apply

TOPOLOGY DETAILS

#	Node	Local Port	Remote Node	Remote Port
1	newlabPod-Spine-1	Ethernet1/29	newlabPod-Leaf-1	Ethernet1/51
2	newlabPod-Spine-2	Ethernet1/28	newlabPod-Leaf-1	Ethernet1/50
3	newlabPod-Spine-1	Ethernet1/28	newlabPod-Leaf-2	Ethernet1/50
4	newlabPod-Spine-2	Ethernet1/29	newlabPod-Leaf-2	Ethernet1/51

After setting the desired configurations, submit the Fabric for deployment.

In the example above, serial ID “SAL18391KFV” is identified as the switch with host name “newlabPod_1_Spine-2”. Before bringing up any other switch in this pod, user will first boot spine-2. Once this switch gets its configuration from Ignite server, other leaf switches from this fabric can be booted using fabric based discovery. When a leaf switch boots through POAP, it will recognize Spine-2 as its neighbor and identify local and remote ports. This information is used to search Fabric database to locate this switch and apply build configuration.

Once one leaf switch and one spine switch are booted, other switches in the fabric can be brought up through Fabric based discovery.

Following boot up scenarios are now possible in the above Fabric.

(a) No switches up - Spine switch boots up
(b) No switches up – Leaf1 or Leaf2 boots up
(c) Spine is up – Leaf1 or Leaf2 boots up
(d) Spine and Leaf1 (Leaf2) are up – Leaf2 (Leaf1) boots up
(e) Leaf1 (Leaf2) is up – Spine boots up
(f) Leaf1 (Leaf2) is up – Leaf2 (Leaf1) boots up

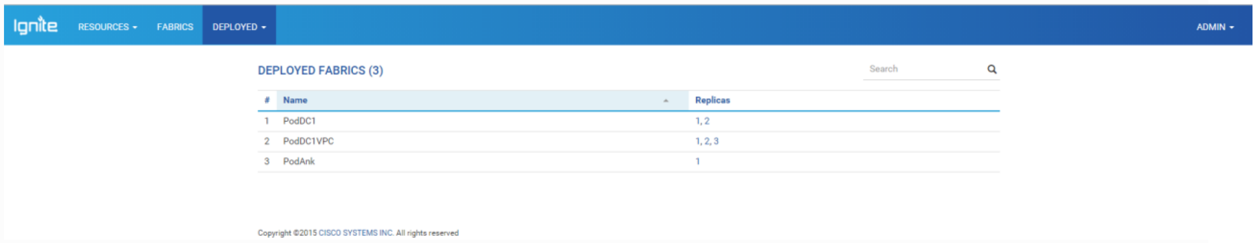
Fabric Status	Booting Switch	Ignite Actions	Remarks
No switches up	Spine	CDP neighbor information in the request will not contain any known neighbors. Fabric based discovery fails and Ignite will use Discovery Rule to identify the switch. Administrator should provide serial ID based rules to identify a configuration. **OR** If serial ID is defined in the Fabric through "Switch Identity", this switch will get mapped to a switch in the Fabric and the configuration set for this switch in the fabric will be provided as startup configuration.	See section ***Discovery Rule – Switch Serial Number*** ***Or*** ***Fabric based discovery (d) Switch Identity***
No switches up	Leaf1 or Leaf2	CDP neighbor information in the request will not contain any known neighbors. Fabric based discovery fails and Ignite will use Discovery Rule to identify the switch. Administrator should provide serial ID based rules to identify a configuration. **OR** If serial ID is defined in the	Best practice: Always, bring up a Spine switch in a fabric using, ***Fabric based discovery – Switch Identity*** All the leaf switches then can identify the Spine as their neighbor and boot up with right

		Fabric through “Switch Identity”, this switch will get mapped to a switch in the Fabric and the configuration set for this switch in the fabric will be provided as startup configuration.	configuration.
.Spine up	Leaf1 or Leaf2	Booting switch will discover Spine1 as the neighbor. Leaf1 will request configuration with the neighbor data {PodHA-Spine-1, Ethernet1/1, Ethernet2/1} and Leaf2 will request configuration with the neighbor data {PodHA-Spine-1, Ethernet1/2, Ethernet2/1}. Ignite will identify the switch as PodHA-Leaf-1 or PodHA-Leaf-2 by matching the data provided in the request with the fabric topology information.	Ignite builds the configuration dynamically from the definition “basicvpcleaf” and creates the config files with all IP addresses allocated, VPC PEER LINK PORTCHANNELS and KEEP ALIVE links configured.
Spine up and Leaf1 or Leaf2 up	Leaf2 or Leaf1	CDP neighbor information in the request will have two nodes PodHA-Spine-1 and PodHA-Leaf-1 or PodHA-Leaf-2. Ignite will identify the switch by matching fabric topology data with the request data and build using	Ignite builds the configuration dynamically from the definition “basicvpcleaf” and creates the config files with all IP addresses allocated, VPC PEER LINK

		“basicvpcleaf” configuration.	PORTCHANNELS and KEEP ALIVE links configured.

Deployed Fabric

Deployed Fabric shows status of switches which were booted through Fabric discovery.

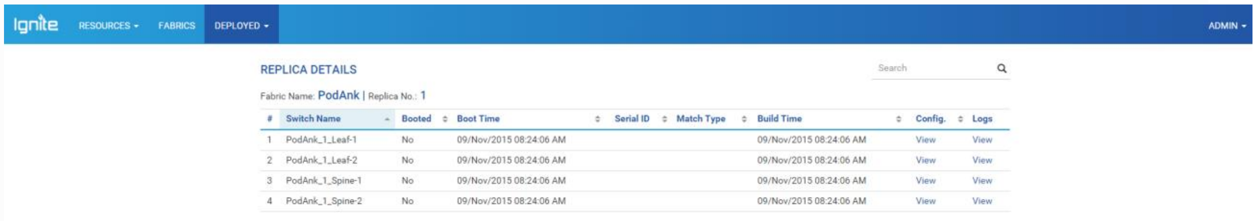


DEPLOYED FABRICS (3)

#	Name	Replicas
1	PodDC1	1, 2
2	PodDC1VPC	1, 2, 3
3	PodAnk	1

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

View showing list of fabrics where a switch was brought up through Ignite POAP process.



REPLICA DETAILS

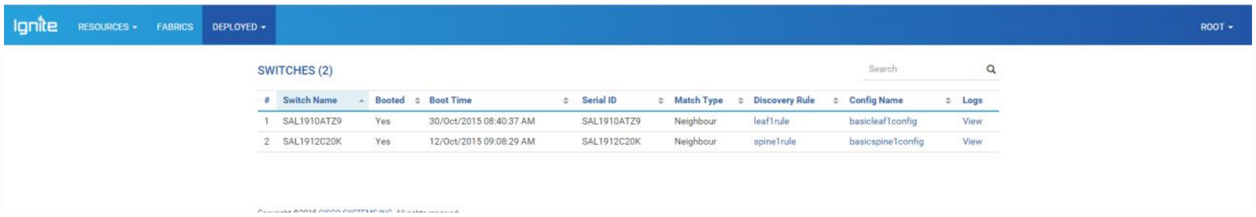
Fabric Name: PodAnk | Replica No.: 1

#	Switch Name	Booted	Boot Time	Serial ID	Match Type	Build Time	Config.	Logs
1	PodAnk_1_Leaf-1	No	09/Nov/2015 08:24:06 AM			09/Nov/2015 08:24:06 AM	View	View
2	PodAnk_1_Leaf-2	No	09/Nov/2015 08:24:06 AM			09/Nov/2015 08:24:06 AM	View	View
3	PodAnk_1_Spine-1	No	09/Nov/2015 08:24:06 AM			09/Nov/2015 08:24:06 AM	View	View
4	PodAnk_1_Spine-2	No	09/Nov/2015 08:24:06 AM			09/Nov/2015 08:24:06 AM	View	View

Once user selects a replica, list of switches in that fabric replica are displayed. Boot time for each switch, configuration which was downloaded etc. is displayed. If configurations were generated and switch was not booted status column “Booted” shows “No”. Configuration files can be viewed by clicking on Config View. Logs generated during POAP.py script execution, can be viewed by clicking Log View.

Deployed Switches

Deployed Switches display list of switches which were booted through global discovery which are not associated with any Fabric. As in Deployed Fabrics view, config file, and log file can be viewed.



SWITCHES (2)

#	Switch Name	Booted	Boot Time	Serial ID	Match Type	Discovery Rule	Config Name	Logs
1	SAL1910ATZ9	Yes	30/Oct/2015 08:40:37 AM	SAL1910ATZ9	Neighbour	leaf1rule	basicleaf1config	View
2	SAL1912C20K	Yes	12/Oct/2015 09:08:29 AM	SAL1912C20K	Neighbour	spine1rule	basicspine1config	View

Image Profiles

Image profiles specify details of the software images available for download to the switch during POAP process. Following atttributes define an image profile: {

```
"image_profile_name": "spine_image",
"image": "n9000-dk9.6.1.2.I3.2.bin",
"imageserver_ip": "172.31.216.138",
"username": "root",
"password": "cisco123",
"access_protocol": "scp"
```

}

image_profile_name - uniquely identifies the profile

image - filename where the image is stored

imageserver_ip : server where the image is stored

username - username to login to the image server

password - password to login to the image server

access_protocol - ftp/http/scp/tftp protocol used to transfer files from server

Image profiles are stored in ignite/fabric/image_profile.py. The profiles should be modified to match your image needs and environment. New profiles can be added similar to the pre-defined ones. spine_image, leaf_image and default_image are example profiles. An image profile is applied to spine tier or leaf tier of the fabric using "Set Defaults" in "Fabrics" view. Once applied, this image profile will be sent to the switch during the POAP boot process along with start up config details. If image profile is not applied, profile named "default_image" is sent to the switch during POAP boot process.