

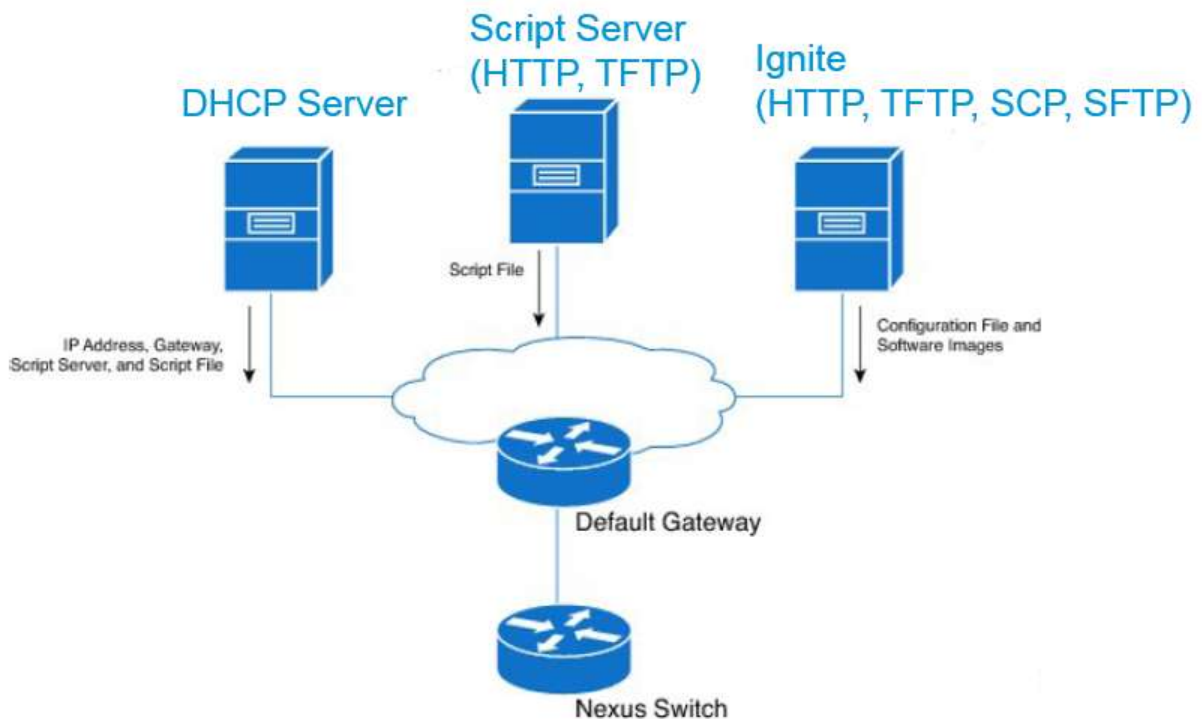
# Ignite

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.

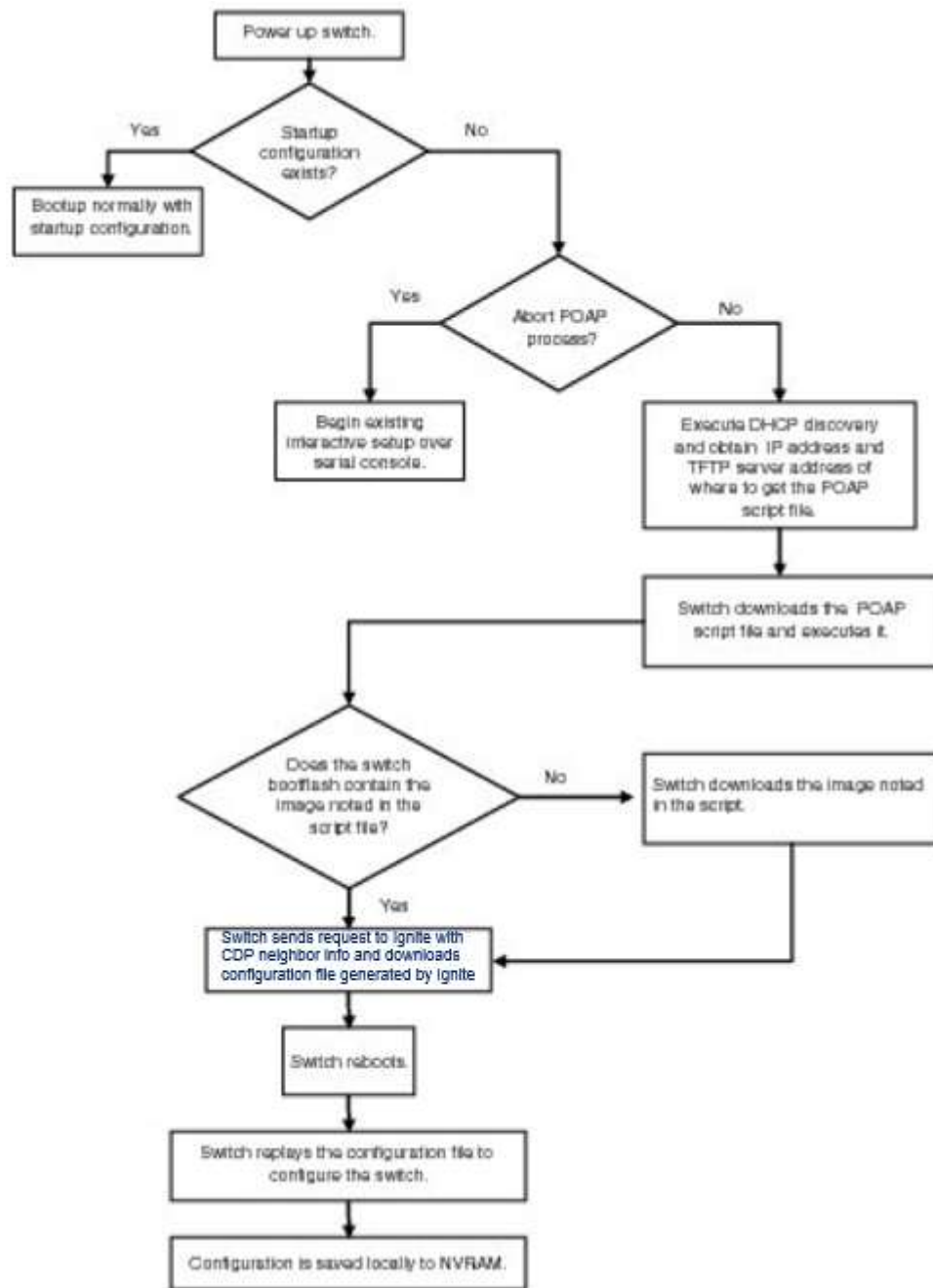
## Overview

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:



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

Script file name—The DHCP server relays the script file name to the DHCP client. The boot file name includes the complete path to the script file on the Script server. The 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.

## 3. Script execution

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

This script (*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 `ignite_poap.py` script also downloads and installs the software image appropriate for this switch.

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 application of the scheduled config as the running config, the switch copies the running configuration to the 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<sup>\*1</sup>

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

<sup>\*1</sup> – 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
HOST_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.
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	IP address assigned to this switch
TRUNK_PORTS	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.

Ignite RESOURCES FABRICS DEPLOYED FABRICS ADMIN

CONFIGLETS (17) Add Delete Search

#	Name	Group	Type	Parameters	Actions
1	212's_no_shut	default	template		
2	adminuser	default	template	ADMIN_PASSWORD	
3	anycastrip	default	template		
4	anycastrip.pine	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		

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.

Ignite RESOURCES FABRICS DEPLOYED FABRICS ADMIN

CONFIGLETS (17) Add Delete Search

#	Name	Group	Type	Parameters	Actions
1	212's_no_shut	default	template		
2	adminuser	default	template	ADMIN_PASSWORD	
3	anycastrip	default	template		
4	anycastrip.pine	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		

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

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 within 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.

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<sup>\*2</sup>. Ignite server provides the following constructs to build a configuration.

<sup>\*2</sup> – 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:

ignite

RESOURCES ▾

FABRICS

DEPLOYED FABRICS

ADMIN

CONFIGURATION

Name

basicvpcleaf

LIST OF CONSTRUCTS

Search Q

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

## 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`.

The screenshot shows the Ignite web interface. At the top, there is a navigation bar with 'Ignite' logo and tabs for 'RESOURCES', 'FABRICS', 'DEPLOYED FABRICS', and 'ADMIN'. Below the navigation bar, there is a section titled 'DISCOVERY RULE (1)' with 'Add' and 'Delete' buttons. A table lists the discovery rules:

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

Below the table, there is a 'Copyright ©2015 CISCO SYSTEMS INC. All rights reserved' notice.

The 'RULE DETAILS' section is shown below the table. It includes a 'Rule Name' field with the value 'serialid\_basic', a 'Build Configuration' field with the value 'basicConfig', and a 'Priority' field with the value '1000'. There is a 'Match' section with radio buttons for 'All', 'Any', and 'Serial ID', where 'Serial ID' is selected. Below the 'Match' section, there is a 'Serial ID' section with two input fields containing the values 'SSI153503NP' and 'SSI153504NP'. A 'Cancel' button is located at the bottom right of the 'RULE DETAILS' section.

### ***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

allLeafs

Build Configuration

basicConfig

Priority

2

Match

☐ All
☒ Any
☐ Serial ID

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

Cancel

## 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

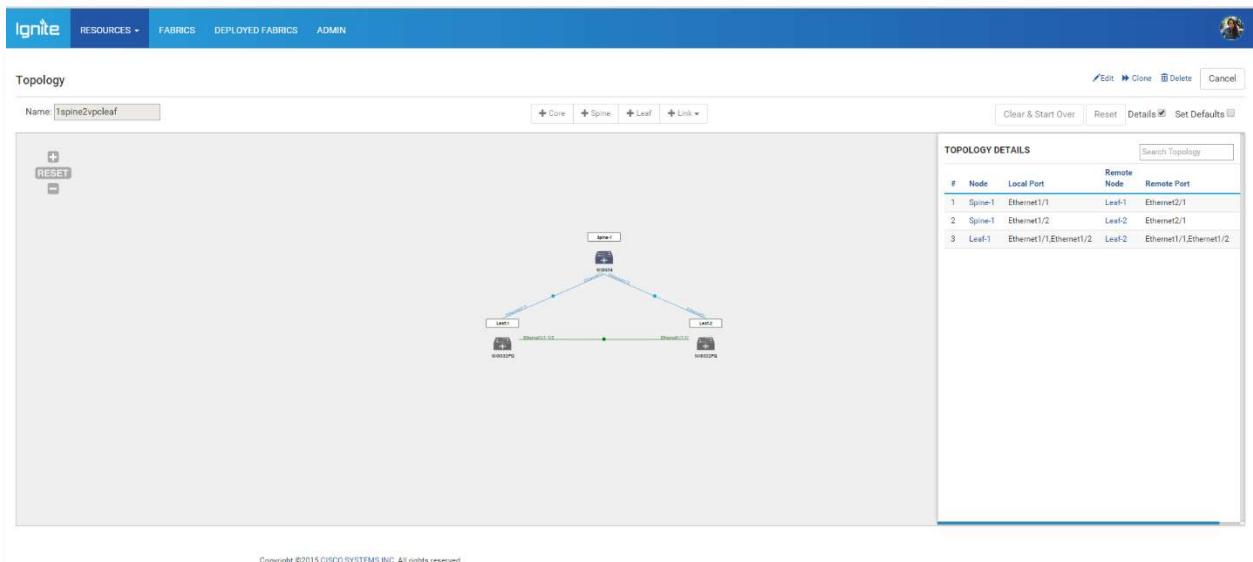
Configuration” to it. Ignite will start building configuration file based on the steps defined in the assigned “Build Configuration”.

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.

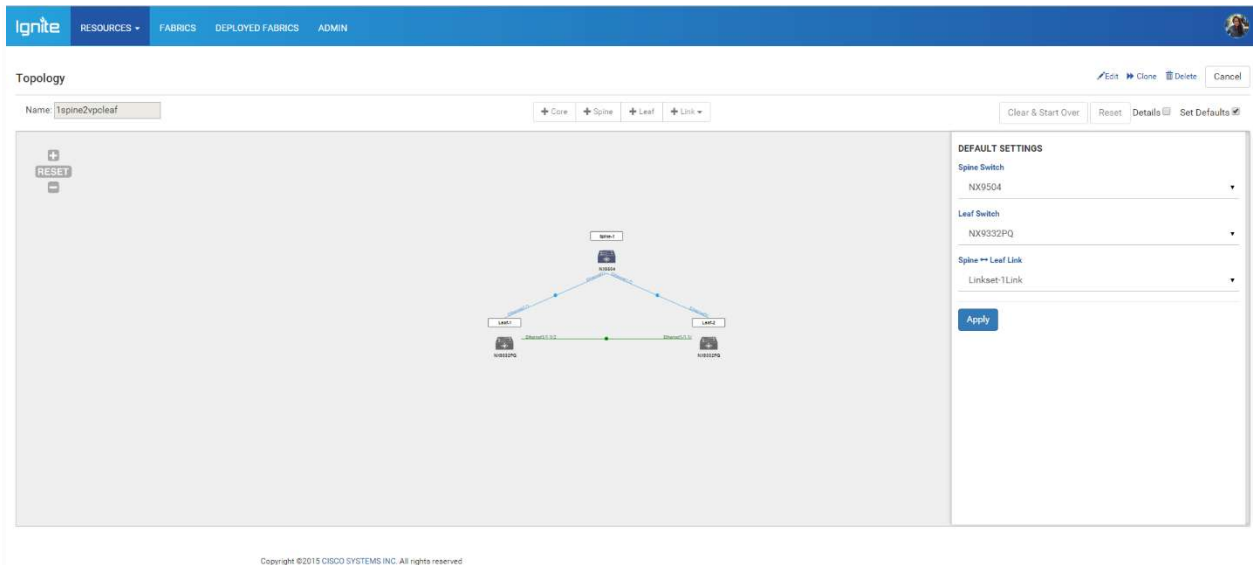
#### (a) Create a topology

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



Administrator first adds a spine by clicking on **+ Spine** and clicking on **+ Leaf** twice to add 2 leaf switches. Ignite uses default settings<sup>\*3</sup> 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.





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***

ignite

RESOURCES ▾

FABRICS

DEPLOYED FABRICS

ADMIN

CONFIGURATION

Cancel

Name

basicConfig

LIST OF CONSTRUCTS

Search

Q

<input type="checkbox"/>	#	Construct	Template/Script	Parameters
<input type="checkbox"/>	1	append_configlet	mgmt	MGMT_IP = Pool.MGMTIPv4
<input type="checkbox"/>	2	append_configlet	adminuser	ADMIN_PASSWORD = cisco123

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

## Basicvpcleaf config

ignite

RESOURCES ▾

FABRICS

DEPLOYED FABRICS

ADMIN

CONFIGURATION

Cancel

Name

basicvpcleaf

LIST OF CONSTRUCTS

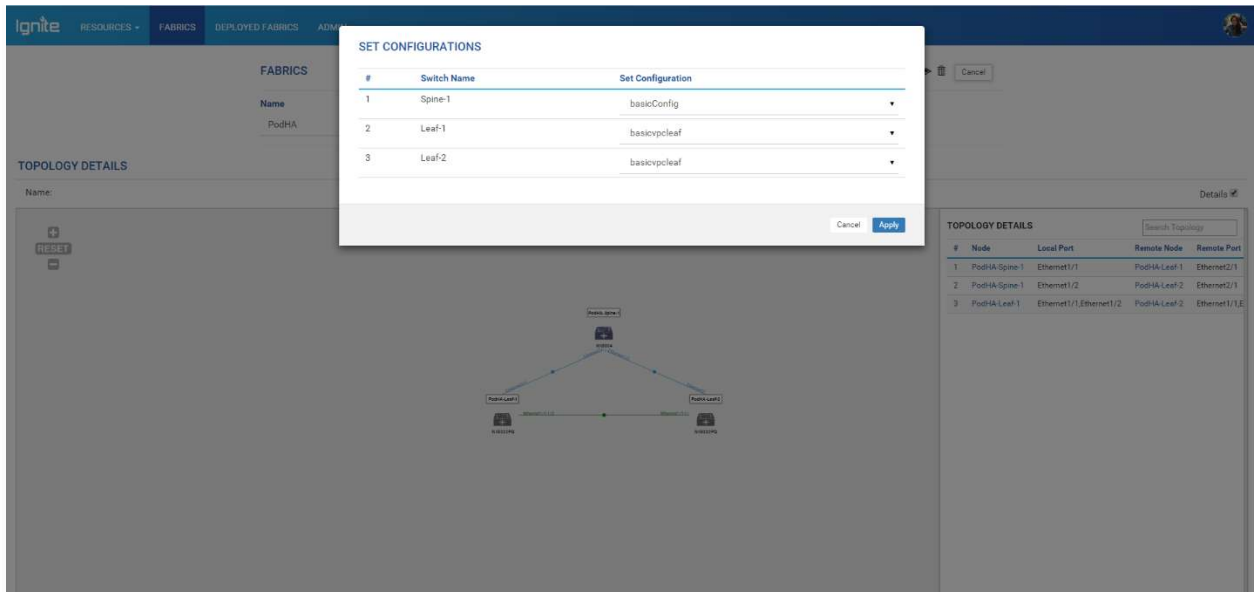
Search

Q

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

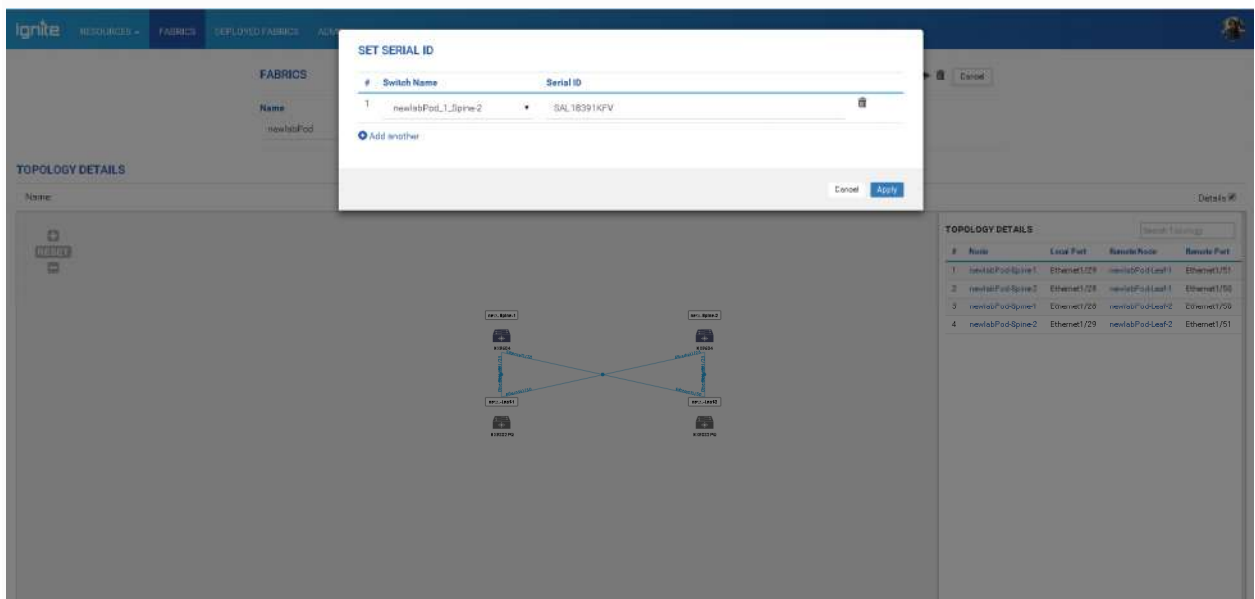
Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

(c) Administrator now defines a Fabric, which uses the topology “1spine2vpcleaf”.  
Using “Set Configuration”



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

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



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 a leaf and spine are booted with startup configurations, 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	<p>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.</p> <p><b>OR</b></p> <p>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</p>	<p>See section <b><i>Discovery Rule – Switch Serial Number</i></b></p> <p><b><i>Or</i></b></p> <p><b><i>Fabric based discovery (d) Switch Identity</i></b></p>

		startup configuration.	
No switches up	Leaf1 or Leaf2	<p>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.</p> <p><b>OR</b></p> <p>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.</p>	<p>Best practice:</p> <p>Always, bring up a Spine switch in a fabric using, <b><i>Fabric based discovery – Switch Identity</i></b></p> <p>All the leaf switches then can identify the Spine as their neighbor and boot up with right configuration.</p>
.Spine up	Leaf1 or Leaf2	<p>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</p>	<p>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</p>

		PodHA-Leaf-1 or PodHA-Leaf-2 by matching the data provided in the request with the fabric topology information.	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 "basicvpcleaf" configuration.	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.

## Deployed Fabric

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

ignite <span>RESOURCES ▾</span> <span>FABRICS</span> <span>DEPLOYED FABRICS</span> <span>ADMIN</span>			
DEPLOYED FABRICS (1)			Search 
#	Name		Replicas
1	newlabPod		1

Copyright ©2015 CISCO SYSTEMS INC. All rights reserved

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

ignite

RESOURCES +

FABRICS

DEPLOYED FABRICS

ADMIN

REPLICA DETAILS

Search

Q

Fabric Name: **newlabPod** | Replicas No.: 1

#	Switch Name	Booted	Boot Time	Serial ID	Match Type	Discovery Rule	Config Name	Logs
1	newlabPod_1_Leaf-1	Yes	01/Oct/2015 12:50:44 PM	SAL190B9WQZ	Neighbour	-	II18lab_Leaf1config	View
2	newlabPod_1_Leaf-2	Yes	30/Sep/2015 04:23:03 PM	SAL190B9WMV	Neighbour	-	II18lab_Leaf2config	View
3	newlabPod_1_Spine-1	Yes	30/Sep/2015 04:35:29 PM	SAL18391KF8	Neighbour	-	II18lab_spineconfig	View
4	newlabPod_1_Spine-2	Yes	30/Sep/2015 04:44:34 PM	SAL18391KFV	System_ID	-	II18lab_spineconfig	View

Copyright ©2015 DISCO SYSTEMS INC. All rights reserved

Once user selects a replica, list of switches brought up so far, boot time for each switch, configuration which was downloaded etc. is displayed. Clicking on the Config file will display the full start up configuration.

