Introduction to Using DevOps Tools for Network Switch Management

Introduces DevOps tools that help ease deployment, configuration and provisioning resources

Describes Ansible, Chef and Puppet and their benefits to network administrators and developers

Provides a foundation on ways to adopt DevOps for creating a model for running network operations

Describes the benefits of the open and programmable features of Lenovo CNOS, mapping to DevOps

Mohammed Yasser A K

Abstract

DevOps brings together tasks of system developers and operators/administrators under a common framework. The use of DevOps ensures service continuity, higher quality of delivery and faster service creation, all with reduced cost.

Lenovo® Cloud Network Operating System (CNOS) provides an open and agile environment that makes it easy to develop switch applications by using standards-based interfaces, protocols and API. This sets the foundation for building robust networks that could be managed and monitored from one place. Extending the use of DevOps Tools to automate the overall manageability and network resource provisioning enables seamless integration with the larger enterprise data center ecosystem supporting cloud and virtualization environments.

This paper describes how network developers and operators can use DevOps tools to create an automated and efficient operation model to manage Lenovo CNOS-based networking switches. This paper is intended for network installers, system/network administrators and service providers who are engaged in configuring and maintaining a network. The reader is expected to be familiar with networking concepts, networking automation and orchestration, and network management.

Contents

Introduction

DevOps brings together the tasks of system developers and network operators under a common framework across an enterprise that is easy to adapt to. This ensures service continuity, higher quality of delivery, and faster service creation with reduced cost. This paper is part of an effort of the Lenovo networking team to enable network developers (Dev) and network administrators/operators (Ops) to use DevOps tools to support and create an automated and efficient operations model in their enterprise networking infrastructure, using Lenovo CNOS-based networking devices.

This document presents a guide to help you map and deliver specific features and capabilities of Lenovo Cloud Network Operating System (CNOS). Administrators can use a variety of third-party DevOps tools, to provide easy setup and maintenance (software and hardware upgrades), as well as troubleshooting and monitoring of network switches.

For example, updating firmware on our switches using traditional methodologies requires significant effort to:

1. Follow instructions that require additional skills/abilities
2. Check for prerequisites (down-level OS, configurations, backup, etc.)
3. Troubleshoot (rollback or recovery)
4. Repeat the task on multiple switches

Automating the firmware upgrade and related configuration tasks with the right DevOps tools makes life easier in an enterprise. Traditional automation frameworks require a dedicated team of developers with advanced scripting and coding skills, working independently of the network operators/administrators team. One of the most difficult aspects is building and maintaining this framework.

DevOps tools address this gap by enabling operators/administrators to write user-readable code and playbooks that leverage our modules and APIs that are certified and included with the tool. As an added benefit, it also provides reuse, rollback, and recovery.

Cloud agility for scale and speed, high availability, programmability, and zero touch provisioning are de facto requirements of today's enterprise network infrastructure. Networks play the most important role in running business critical applications, and every business expects 100% uptime and seamless scaling to meet future requirements. To run newer advanced business tool and to scale and upgrade other data center hardware (servers and storage), there is a need to run a prerequisite check on network quality, compatibility, and compliance beforehand.

Applying DevOps tools with network switches running Lenovo CNOS addresses these challenges through its extensive use for automation and orchestration. Most of these tools also extend their support to servers, storage, management applications, operating systems, cloud orchestrators and security appliances. As a result, they form a universal framework for the entire enterprise data center, not just networking. When selecting a DevOps tool, factors to consider beyond automation and orchestration include security, user interface, programmatic interfaces and communication protocols for seamless integration into existing high-end applications and management ecosystems, such as Azure, vRealize Suite, OpenStack, AWS, etc.

In this era of virtualization, most hypervisor/cloud OS vendors provide a unified means to control and manage compute, storage, and network resources using a single orchestration tool. Cloud Service Providers (CSPs) or Managed Service Providers (MSP) use these tools to integrate their self-service portals that hold demand-based billing and service catalogues, which depend on the underlying network infrastructure for quality of service and seamless delivery. DevOps tools provide the ability to provision and manage networking resources without having to run explicit network operations on the switches directly.

DevOps with CNOS

Lenovo provides a set of certified modules/libraries for its Cloud Networking Operating System (CNOS), available for the most widely used DevOps tools, such as Ansible, Chef, and Puppet. The modules/libraries either are part of the DevOps Software Installer package or are available on a secure download link maintained by the DevOps tool vendor. Chef and Puppet require a client agent running on the switch or managed by a proxy VM. Ansible is agentless and uses communication protocols, APIs, or CLIs to communicate with the switch.

Lenovo CNOS provides an open and agile operating environment that offers programmability and supports industry-standard interfaces and protocols, connecting to CNOS building blocks that can be managed and monitored from one place.

One of the most common application of DevOps tools are in the management and provisioning of network resources following the network switch deployment and maintenance lifecycle as depicted in Figure 1 on page 4.

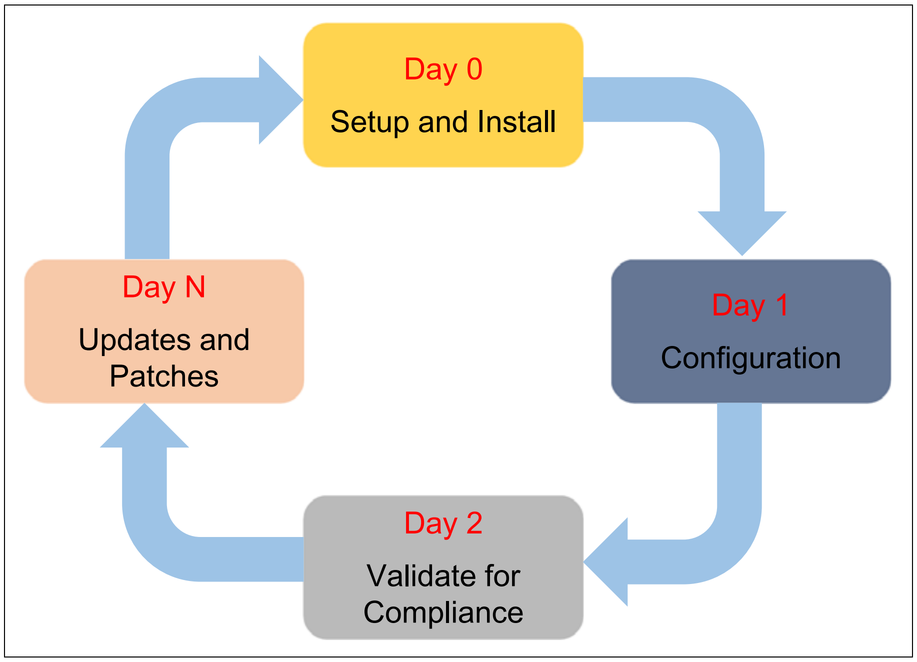

Figure 1   Deployment and maintenance lifecycle

Ansible

Ansible is an automation framework used for configuration management of your data center infrastructure. For CNOS, it provides a library of modules that accelerate switch deployment and provisioning of network resources, and offers configuration options for CNOS network requirements. Ansible for CNOS provides tools to view Ansible Facts, manage/save/rollback configuration, install/upgrade NOS, Interface/Protocol configuration, and reset/reboot/shut down managed network switches.

ENOS support: Lenovo also supports Ansible with ENOS (Enterprise Networking Operating System)-based switches

With Ansible, you can:

- Automate repetitive tasks to speed routine network changes and free up your time for more strategic work
- Leverage the same simple, powerful, and agentless automation tool for network tasks that operations and development use
- Separate the data model (in a playbook or role) from the execution layer (via Ansible modules) to manage heterogeneous network devices
- Benefit from community and vendor-generated sample playbooks and roles to help accelerate network automation projects
- Communicate securely with network hardware over SSH or HTTPS

Ansible uses a client-server architecture. The Ansible software is installed on a control node running a BSD UNIX or Linux distribution (such as Red Hat Enterprise Linux or Ubuntu), which manages one or more other nodes. Ansible uses an agentless architecture and thus does not require installing any Ansible-specific software on the managed nodes.

The components in the Ansible architecture are as follows:

- Control Node – Any machine with Ansible installed
  - Run commands invoking /usr/bin/ansible
  - Run playbooks invoking /usr/bin/ansible-playbook
- Managed Nodes – Network switches/devices (hosts) - agentless
- Inventory – List of Managed Nodes (hostfile)
- Modules – Units of code
  - Each module has a particular use (for example, managing VLANs)
  - Invoke a single module with a task
  - Invoke multiple modules in a playbook
- Tasks – Units of Action (for example, show VLAN)
- Playbooks – Ordered list of tasks
  - Includes variables and tasks
  - Written in YAML (Easy to read, share and understand)

Ansible modules work with ad-hoc commands, playbooks, and roles. Ansible supports multiple communication protocols for working with network switches.

- network_cli: CLI over SSH
- netconf: XML over SSH
- local: depends on provider (e.g., API over HTTPS)

Ansible is written in Python, but it uses simple YAML syntax to run automation jobs from a playbook or role.

Running Ansible with Lenovo CNOS

Executing an Ansible command is similar to executing other Linux commands. For example:

```
ansible all -i <cnos.example.com>, -c network_cli -u <admin> -k -m cnos_facts -e
ansible_network_os=cnos
```

Ansible commands have seven values:

`all`	host group(s)
`-i`	inventory (the device or devices-inventory-file to target)
`-c`	connection method
`-u`	user
`-k`	password prompt
`-m`	module
`-e`	extra variable (-e of NOS)

Meaning of the colors: The colors in the commands and examples have the following meaning:

- Red text means Lenovo CNOS attributes – DNS hostname, credential, module name and OS
- Blue text means Ansible attributes – Connection Method and Network OS Variable
- Green text means a Task Summary

Creating your first Ansible playbook (first_playbook.yml) follows the format shown in Figure 2.

```
- name: Network Getting Started First Playbook
  connection: network_cli
  hosts: all
  tasks:
    - name: Get config for CNOS devices
      cnos_facts:
        gather_subset: all
    - name: Display the config
      debug:
        msg: "The hostname is {{ ansible_net_hostname }} and the OS is {{ ansible_net_version }}"
```

Figure 2 Contents of first_playbook.yml

Running your playbook requires the following command syntax:

```
ansible-playbook -i <cnos.example.com>, -u <username> -k -e
ansible_network_os=cnos first_playbook.yml
```

Ansible-playbook command has four values:

`-i`	inventory (the device or devices-inventory-file to target)
`-u`	user (credential of Ansible control node)
`-k`	password prompt
`-e`	extra variable

The output of the command is shown in Figure 3.

```
PLAY [First Playbook]
**********************************************************************
TASK [Gathering Facts]
**********************************************************************ok: [cnos.example.com]
TASK [Get config for CNOS devices]
**********************************************************************ok: [cnos.example.com]
TASK [Display the config]
**********************************************************************ok: [cnos.example.com] => {
    "failed": false,
    "msg": "The hostname is cnos.example.com and the OS is CNOS"
}
```

Figure 3 Output of the ansible-playbook command

Lenovo CNOS Ansible roles

Roles are sets of Ansible defaults, files, tasks, templates, variables, and other Ansible components that work together. Moving from a command to a playbook makes it easy to run multiple tasks and repeat the same tasks in the same order. Moving from a playbook to a role makes it even easier to reuse and share your ordered tasks. Ansible Galaxy lets you share your roles and use others' roles, either directly or as inspiration.

Lenovo provides the following roles for CNOS on Ansible Galaxy:

- **cnos-clos-leaf** - leaf switches configuration for BGP in a CLOS fabric
- **cnos-clos-spine** - spine switches configuration for BGP in a CLOS fabric
- **cnos-vlag-1tier-leaf** - leaf switches configuration for VLAG in a Tier 1 fabric
- **cnos-vlag-2tier-leaf** - leaf switches configuration for VLAG in a Tier 2 fabric
- **cnos-vlag-1tier-spine** - spine switches configuration for VLAG in a Tier 1 fabric
- **cnos-vlag-2tier-spine** - spine switches configuration for VLAG in a Tier 2 fabric

Figure 4 shows how Lenovo Ansible CNOS modules are used to derive tasks/roles and playbooks to be applied on CNOS switches (hosts).

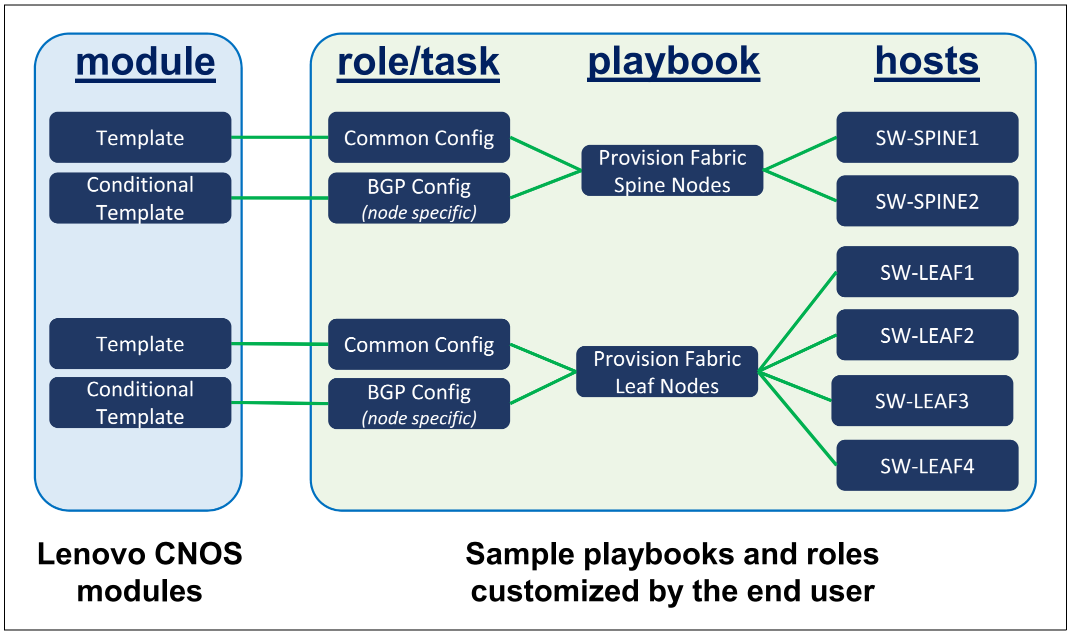

Figure 4 Creating custom playbooks for switches in a spine-leaf network

Puppet

Puppet is a configuration management tool that can be used to deploy, configure, manage, and maintain a network switch. You can use Puppet for the entire life of a network switch, from connecting the switch to getting facts and applying configurations. You can use Puppet to define distinct configurations for each switch, and to continuously check and confirm whether the required configuration is in place and unaltered.

The Lenovo Network Development Toolkit for Puppet is designed to be used for configuring port channels, IP management, and telemetry on switches running CNOS.

Puppet uses a master/slave architecture as depicted in Figure 5 on page 9.

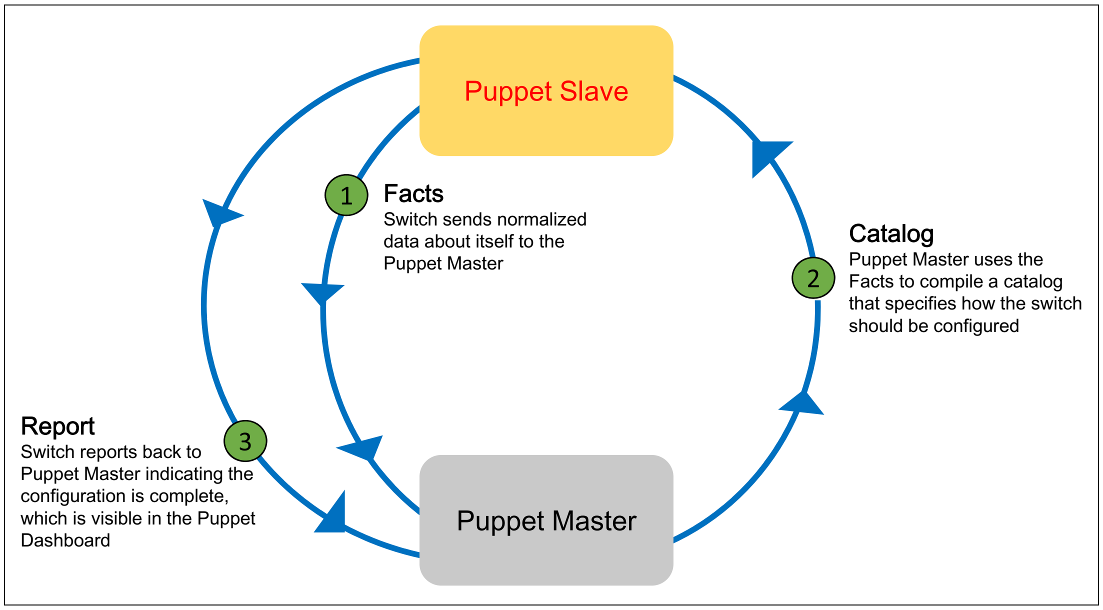
Puppet Slave
1
Facts
Switch sends normalized data about itself to the Puppet Master
2
Catalog
Puppet Master uses the Facts to compile a catalog that specifies how the switch should be configured
3
Report
Switch reports back to Puppet Master indicating the configuration is complete, which is visible in the Puppet Dashboard
Puppet Master

Figure 5 Puppet master/slave architecture

Puppet uses Declarative Domain Specific Language (DSL) and runs on open source software written in Ruby. Puppet comes in the open source and commercial versions (Puppet Enterprises). Puppet Modules for Lenovo CNOS are available at Puppet Forge. Puppet Master talks to managed nodes (switches) using its client software (Puppet Agent), running either on the switch (slave) or on a Proxy Linux instance. Lenovo CNOS-based switches use a Linux instance to act as a Proxy Slave for talking to the Puppet Master.

Puppet Agent running on a slave sends facts to the Puppet Master. Facts are key/value data pairs that represent some aspect of the switch state, such as its IP address, uptime, operating system, etc. Puppet Master uses the facts to compile a catalog that defines how the slave will be configured. A catalog is a document that describes the desired state for each resource that the Puppet Master manages on a slave. The slave reports back to the master to indicate that configuration is complete.

Puppet consists of the following components:

- **Manifests**
  - Puppet Master stores switch configuration in the native Puppet language
  - Using the .pp extension, you can write, for example, a Manifest on the Master that creates a VLAN on all Slaves (switches) connected to the Master
  - The default main manifest file is located on your Master at: /etc/puppetlabs/code/environments/production/manifests/site.pp
- **Modules**
  - A collection of libraries/APIs located in the Module Path of Puppet Master
  - Self-contained bundles of code and data
- **Resources** and **Resource Types**
  - A resource is the fundamental unit for modeling system configurations
  - Each resource describes some aspect of a system, like a specific service or package
- **Providers**
  - Also referred to as Resource Providers - Fulfillers of any particular resource

  - Not all resources have or need providers
- **Catalog**
  - The desired state of each managed resource on a Slave
  - A compilation of all resources, with relationships that the Master applies to a given Slave
  - Compiled by a Master from manifests and Slave-provided data (such as Facts, certificates, and an environment, if one is provided)
  - The Master then serves the compiled catalog to the Slave when requested
- **Templates**
  - Ruby expressions to define customized content and variable input
  - Defined in Manifests
- **PuppetDB** stores all the data generated by Puppet
- **Puppet Console** – WebUI (Available through Puppet Enterprise)

Lenovo CNOS model for Puppet

CNOS operates in *puppet device* mode using a Puppet Proxy Slave running the Agent, rather than running Puppet Agent on the switch directly. The CNOS model for Puppet is shown in Figure 6.

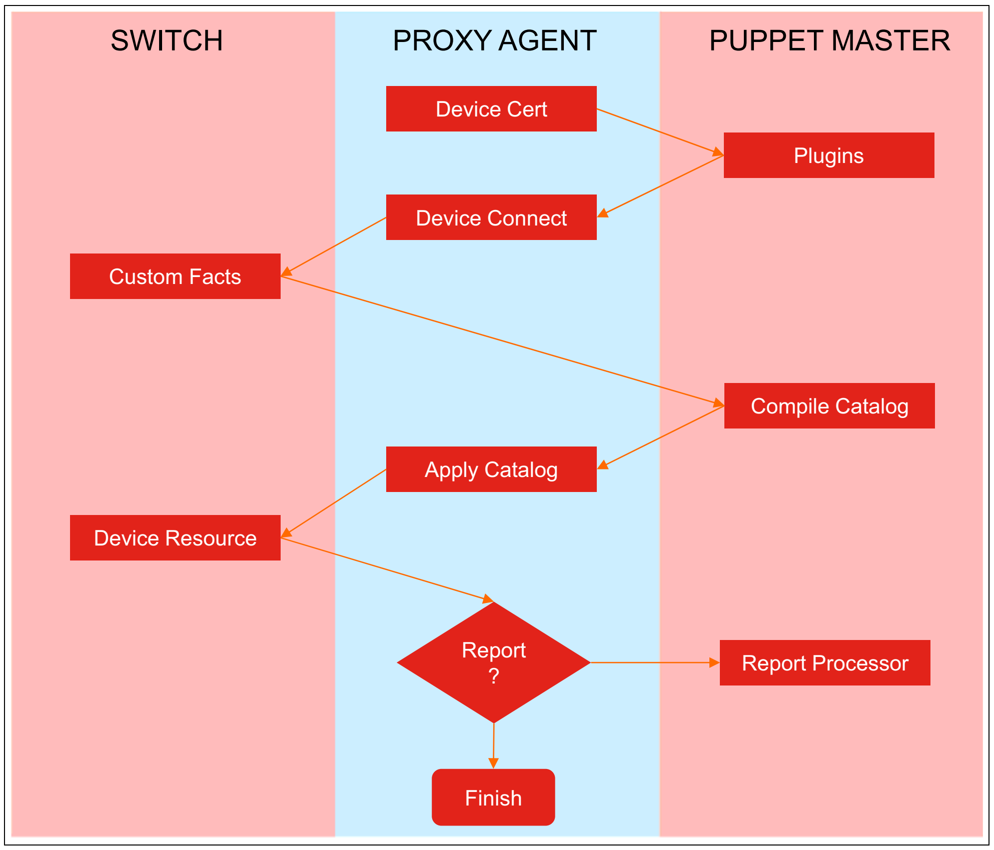

Figure 6 CNOS model for Puppet

Puppet Manifest for Lenovo CNOS offer the following features:

- Image upgrade
- Configure download/upload
- VLAN provisioning
- IP interface configuration
- Telemetry configuration
- VRRP configuration

Figure 7 shows the communication framework for Lenovo CNOS switches in a Puppet environment.

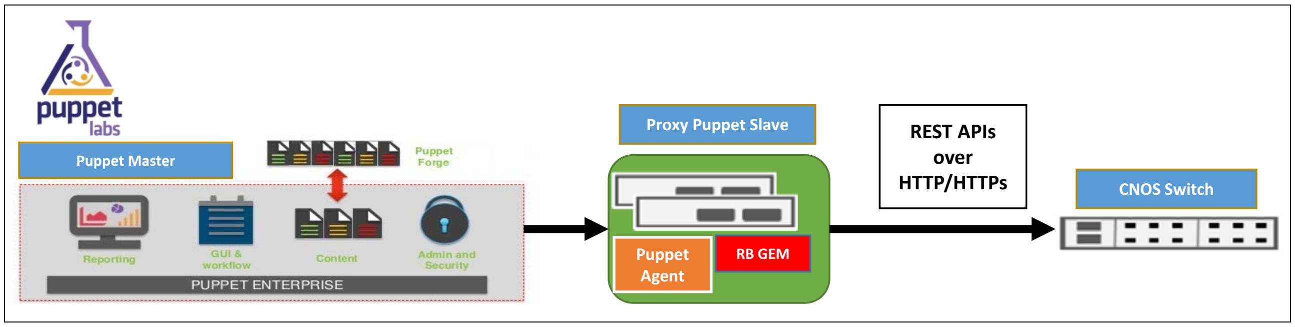

Figure 7 Lenovo CNOS switch connection to Puppet Enterprise

Chef

Chef is a Ruby-based configuration management tool designed to streamline the tasks of configuring and maintaining your network switches. Chef can integrate with cloud-based platforms to automatically provision and configure new switches.

The Lenovo Development Toolkit for Chef provides a native Ruby implementation for programming Lenovo CNOS network devices using Ruby. The Ruby client APIs can be used to build native applications in Ruby that can communicate with CNOS remotely using HTTP or HTTPs.

The Ruby API implementation also provides an API layer for building native Ruby objects to configure and manage Lenovo switches. The library is freely provided to the open source community for building applications using CNOS REST API infrastructure.

The Chef system configuration files are called *recipes* and they are stored in a *cookbook*. A Cookbook relates to a single task but can have multiple switches assigned. Recipes and cookbooks are written using the Ruby programming language.

Figure 8 on page 12 shows the layout of the Chef DevOps setup in an enterprise data center.

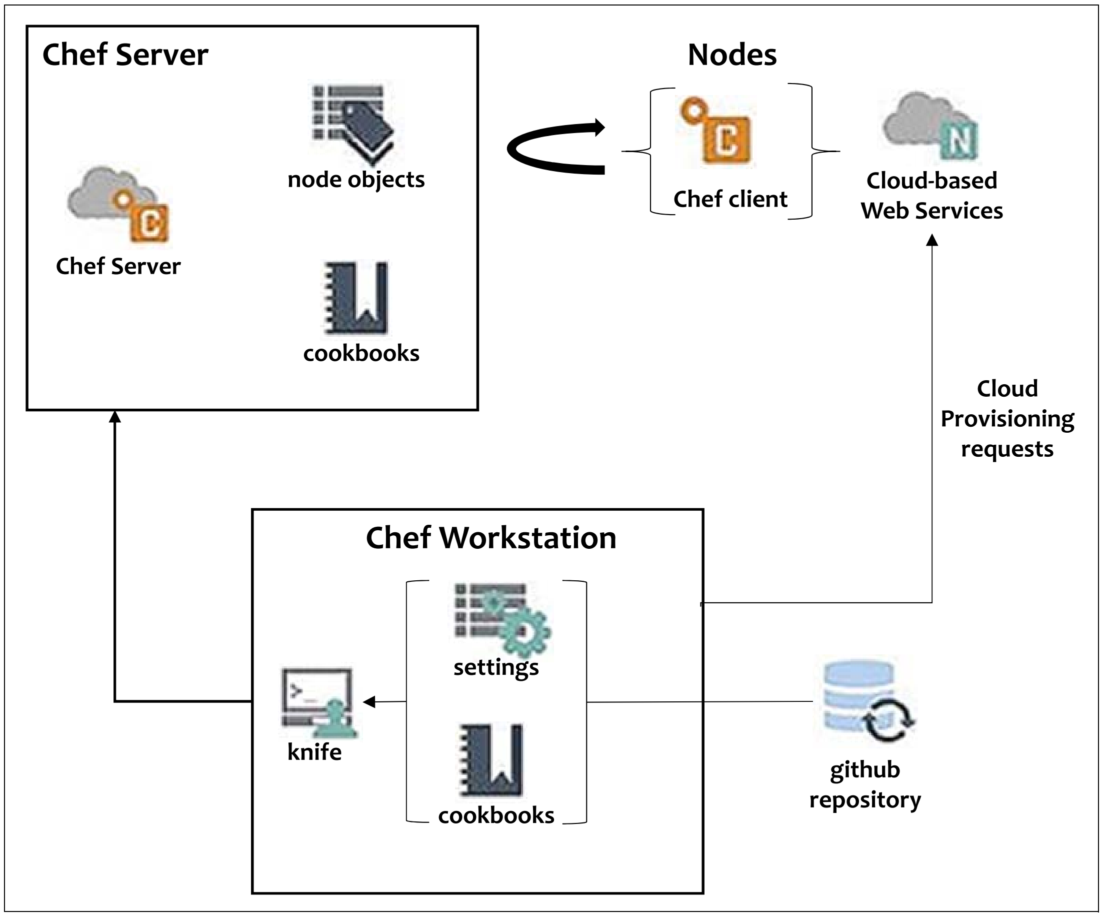

Figure 8 Chef DevOps in an enterprise data center

The Chef server stores cookbooks and when a new Chef client node checks in with the server, Recipes are sent to tell the node how to configure itself. The client then checks the server occasionally to see if any changes have been made. If so, the client node updates itself. By changing a recipe, you can roll out patches and updates over your entire infrastructure, rather than having to do so on each individual machine.

Lenovo offers certified cookbooks through the Chef Supermarket portal. Chef Server also holds configuration files, metadata and status of all the nodes.

Chef Workstation is where you create, test, and maintain cookbooks and policies that will be pushed to nodes. Chef Repository is where cookbooks are written and maintained. It is located in the directory `~/chef-repo`.

Chef communicates with the server using the knife command. Knife enables the Chef server to maintain sync with this repository.

Lenovo CNOS model for Chef

Similarly to Puppet, Chef enables communication to managed nodes (switches) using its client software (Chef client) either running on the switch (Chef node) or on a Proxy Linux instance (proxy Chef node). Lenovo CNOS switches use a Linux instance to act as a proxy Chef node while talking to Chef Workstation and Chef Server.

Chef resources and providers can also define new functionality for use in Chef recipes. A resource defines a set of actions and attributes; a provider tells the Chef client how to commit each action.

Chef cookbooks for Lenovo CNOS offer the below features:

- Image upgrade
- Configuration download/upload
- VLAN provisioning
- IP interface configuration

Figure 9 shows the communication framework for Lenovo CNOS switches in a Chef environment.

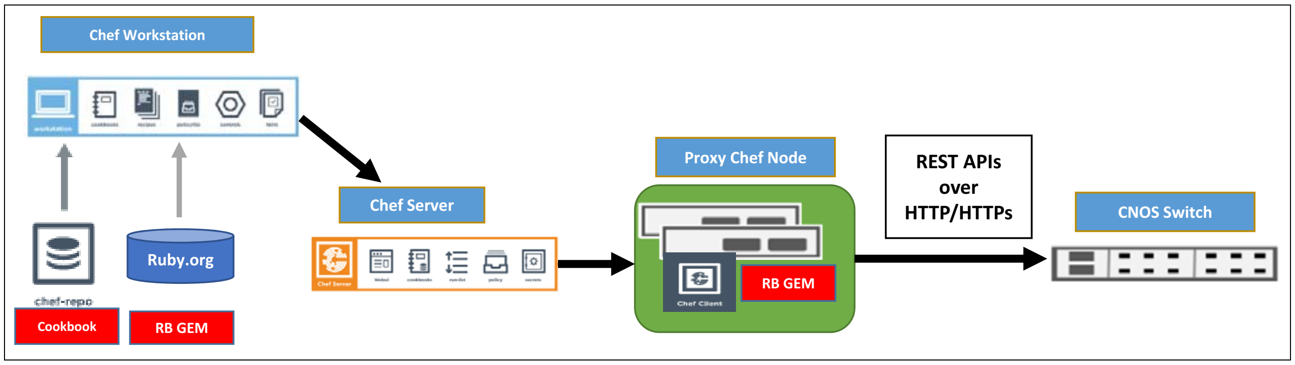

Figure 9 Lenovo CNOS switch connection to Chef Server and Workstation

Conclusion

There are many operational tasks involved in setting up a network, provisioning network resources, and maintaining everything. As a result, adapting DevOps in the networking world can seem dauntingly complex.

For this reason, Lenovo provides supported services and libraries that enable DevOps tools like Ansible, Puppet, and Chef to simplify the configuration management of switches and routers running Lenovo CNOS.

Links and resources

For more information on DevOps and Lenovo CNOS, see these resources:

- Lenovo DevOps Solutions page

  `https://www3.lenovo.com/us/en/data-center/solutions/DevOps/p/devops`

- Product web page for Lenovo Networking Development Toolkit for Chef

  `https://www3.lenovo.com/us/en/data-center/networking/networking-software/network-management/#tab-restr_00004829`

- Lenovo Information Center - Ansible for Lenovo Networking

  `http://systemx.lenovofiles.com/help/index.jsp?topic=%2Fcom.lenovo.switchmgt.ansible.doc%2FAnsible_for_Lenovo_Networking.html`

- Ansible and Lenovo

  https://www.ansible.com/integrations/networks/lenovo
- Developer's Guide for Lenovo Network Development Toolkit for Puppet

  http://systemx.lenovofiles.com/help/topic/com.lenovo.switchmgt.puppet.doc/Puppet_DG_1-0.pdf
- Puppet Forge - Module to manage and configure Lenovo CNOS devices

  https://forge.puppet.com/lenovo/cnos
- Developer's Guide for Lenovo Network Development Toolkit for Chef

  http://systemx.lenovofiles.com/help/topic/com.lenovo.switchmgt.chef.doc/Chef_DG_1-0.pdf
- Chef download page - recipes for managing network resources on Lenovo switches

  https://supermarket.chef.io/cookbooks/chef-cnos

Author

Mohammed Yasser A K is a Senior Development Manager - Quality in the Networking Development division of Lenovo Global Technologies, with over 15 years of experience in program management and SQA leadership in enterprise datacenter product engineering. He is a postgraduate in Information Technology and works from Bangalore, India.

Thanks to the following people for their contributions to this project:

- Chidambaram Bhagavathiperumal
- Anil Kumar Muraleedharan
- Scott Lorditch
- Jim Whitten
- David Watts
- Mark Chapman

Notices

This document was created or updated on May 30, 2018.

Send us your comments via the **Rate & Provide Feedback** form found at
http://lenovopress.com/lp0895

Trademarks