Hosting Applications Using Configuration Management Tools

Configuration management tools are used to automate manual tasks, such as setting up servers and network devices. As application delivery requirements keep changing, reconfiguring network equipment becomes a challenge. The manual reconfiguration process is prone to errors, which in turn can cause network outages. Configuration management tools help when configurations need to be updated constantly, and on multiple network devices.

The Cisco IOS XR Software works well with the following configuration management tools:

- Chef
- Puppet

This section explains how you can install, configure, and use the configuration management tools, Chef and Puppet for application hosting on IOS XR.

- Using Chef for Configuring Cisco IOS XR, on page 1
- Using Puppet for Configuring Cisco IOS XR, on page 5

Using Chef for Configuring Cisco IOS XR

Chef is an open-source IT automation tool that you can use to install, configure, and deploy various applications natively on Cisco IOS XR.

To use Chef, you need the following components:

- Chef Client RPM Version 12.5, or later for Cisco IOS XR 6.0
- Chef Server Version 12.4, or higher
- Applications that are compatible with the Wind River Linux 7 environment of IOS XR

You also need three Chef built-in resources to deploy your application natively on IOS XR. The three built-in Chef Resources are:

- Package Resource
- File Resource
- Service Resource

Access the links provided in the following table for additional details on Chef and Chef resources:

Table 1: Chef Resources

Topic	Link
Chef Software, Inc.	https://www.chef.io/
Chef Overview	https://docs.chef.io/chef_overview.html
Package Resource Reference	https://docs.chef.io/resource_package.html
File Resource Reference	https://docs.chef.io/resource_file.html
Service Resource Reference	https://docs.chef.io/resource_service.html
Chef Server Reference	https://docs.chef.io/install_server.html
Chef Client for Native XR Environment	Chef Client

Installing and Configuring the Chef Client

This section describes the procedure for installing the Chef Client on IOS XR.

Prerequisites

Ensure that the following requirements are met before you proceed with installation:

- Your workstation is set up with the Chef repository and the Chef Development Kit.
- Chef Server Version 12.4, or higher is installed and accessible from your Linux box.
- The Chef Server identification files are available.
- You have the right name server and domain name entries configured in the Linux environment (`/etc/resolv.conf`).
- The router is using a valid NTP server.

Configuration Procedure

To install and configure the Chef Client on IOS XR, follow these steps:

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

   You have entered the IOS XR prompt.

2. Enter the third-party network namespace or global VRF, depending on the version of Cisco IOS XR you are using in your network.

   You can verify whether you are in the namespace by viewing the interfaces, as shown here:

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
```

```
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2 or higher, run the following command */
RP/0/RP0/CPU0:ios# bash


[xr-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

3. (Optional) Configure a proxy server (`http_proxy`, `https_proxy`) as needed.

```
http_proxy=http://proxy.youtube.com:8080
https_proxy=https://proxy.youtube.com:8080
```

4. Install the Chef Client.

```
[xr-vm_node0_RP0_CPU0:~]$ yum install https://chef.io/chef/install.sh
```

The Chef **install.sh** script automatically determines the latest version of the Chef Client RPM for installation.

5. Copy the `validation.pem` file from the Chef server to `/etc/chef/validation.pem`
6. Edit the Chef Client configuration file at `/etc/chef/client.rb` with Chef Server identification and Client settings.

```
validation_client_name 'chef-validator'
chef_server_url 'https://my_chef_server.youtube.com/organizations/chef'
node_name 'n3k.youtube.com' # "This" client device.
cookbook_sync_threads 5 # necessary for small memory switches (4G or less)
interval 30 # client-run interval; remove for "never"
```

7. Run the Chef Client.

```
[xr-vm_node0_RP0_CPU0:~]$ chef-client
```

Note To run the Client once, use the **chef-client --once** command. For more information, see the Chef documentation at https://docs.chef.io/chef_client.html

The Chef Client is successfully installed on IOS XR.

Creating a Chef Cookbook with Recipes

A Chef cookbook, loaded with Chef recipes, can be created on your Linux workstation, and copied to the Chef server. After you install the Chef client on IOS XR, the cookbook with recipes can be downloaded from the Chef server, and used while running the client.

Prerequisites

Ensure the following requirements are met before you proceed:

- You have access to the application package compatible with the native IOS XR environment.
- Target application package is hosted on an accessible repository or downloaded to a boot flash.

Configuration Procedure

Use the following procedure to create a Chef recipe that starts the `bootlogd` service, and installs iPerf on IOS XR:

1. Create a cookbook on your Linux workstation by using the corresponding knife command.

```
knife cookbook create cisco-network-chef-cookbook
```

2. Create the Chef recipe file to install iPerf, and add it to the cookbook.

   The Chef recipe must be created in the `cisco-network-chef-cookbook/recipes/` directory. For it to be loaded automatically by the Chef Client, the Chef recipe must be named as `default.rb`.

```
#
# Recipe:: demo_default_providers
#
# Copyright (c) 2015 The Authors, All Rights Reserved.
```

```
package = 'iperf-2.0.5-r0.0.core2_64.rpm'
service = 'bootlogd'

remote_file "/#{package}" do
  source "http://10.105.247.73/wrl7_yum_repo/#{package}"
  action :create
end

yum_package "#{package}" do
  source "/#{package}"
  action :install
end

service "#{service}" do
  action :start
end
```

3. Access the Chef Server from your Linux workstation and upload the cookbook to the server.
4. Log into the IOS XR shell, and run the Chef Client to load and execute the cookbook.

```
[xr-vm_node0_RP0_CPU0:~]$chef-client
```

The iperf RPM is installed on IOS XR.

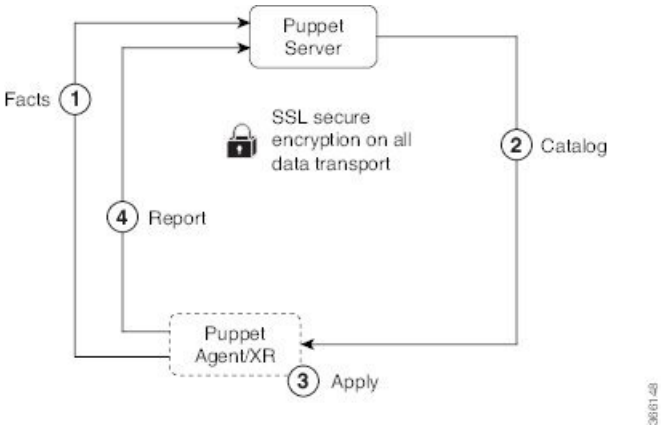
For additional details on the Chef Client, refer to https://docs.chef.io/chef_client.html

Using Puppet for Configuring Cisco IOS XR

Puppet is an open-source configuration management and automation tool that you can use to install and configure various applications on IOS XR. Puppet is provided by Puppet Labs, and runs well on Windows, Unix, and Linux systems. Puppet uses its own declarative langauage to describe system configuration.

Puppet follows a client-server model. The Puppet client is known as the Puppet Agent, and is installed on XR. The configuration file, called the Puppet manifest, is stored on the Puppet Server and contains configuration for multiple nodes. On receiving the information about XR from the Puppet Agent, the Puppet Server compiles the manifest into a catalog that can be used to configure the node that sent the information. This workflow is described in the following illustration.

Figure 1: Basic Puppet Workflow



1. The Puppet Agent sends information about IOS XR to the Puppet Server.
2. The Puppet Server compiles the information into a Catalog, and sends to the Puppet Agent.
3. The Puppet Agent applies the catalog to XR.
4. The Puppet Agent sends a configuration complete report to the Puppet Server.

To use Puppet, you need the following components:

- Puppet RPM built for IOS XR.
- Puppet Server Version 4.0, or higher.

Table 2: Puppet Resources

Topic	Link
Cisco Github Puppet Yang Module	https://github.com/cisco/cisco-yang-puppet-module
Puppet Labs	https://puppetlabs.com/
Package Type Reference	https://docs.puppetlabs.com/references/latest/type.html#package
File Type Reference	https://docs.puppetlabs.com/references/latest/type.html#file
Service Type Reference	https://docs.puppetlabs.com/references/latest/type.html#service
Puppet Server Reference	Puppet Server
Puppet Agent for IOS XR Environment	Puppet Agent

Installing and Configuring the Puppet Agent

This section describes how you can install and configure the Puppet Agent on IOS XR.

Prerequisites

Ensure that the following requirements are met before you proceed with installation.

- Puppet Server Version 4.0, or higher is installed and accessible from your workstation.
- You have the right name server and domain name entries configured in the Linux environment (`/etc/resolv.conf`).
- The router is using a valid NTP server.

Configuration Procedure

To install and configure the Puppet Agent on IOS XR, follow these steps:

1. From your Linux box, access the IOS XR console through SSH, and log in.

```
cisco@host:~$ ssh root@192.168.122.188
root@192.168.122.188's password:
RP/0/RP0/CPU0:ios#
```

You have entered the IOS XR prompt.

2. Enter the third-party network namespace or global VRF, depending on the version of Cisco IOS XR you are using in your network.

   You can verify whether you are in the namespace by viewing the interfaces, as shown:

```
/* If you are using Cisco IOS XR Version 6.0.0, run the following command */
RP/0/RP0/CPU0:ios# run ip netns exec tpnns bash

/* If you are using Cisco IOS XR Version 6.0.2 or higher, run the following command */
RP/0/RP0/CPU0:ios# bash

[xr-vm_node0_RP0_CPU0:~]$ ifconfig
Gi0_0_0_0 Link encap:Ethernet  HWaddr 52:46:04:87:19:3c
          inet addr:192.164.168.10  Mask:255.255.255.0
          inet6 addr: fe80::5046:4ff:fe87:193c/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

Mg0_RP0_CPU0_0 Link encap:Ethernet  HWaddr 52:46:12:7a:88:41
          inet addr:192.168.122.197  Mask:255.255.255.0
          inet6 addr: fe80::5046:12ff:fe7a:8841/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1514  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:3 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:210 (210.0 B)

fwd_ew    Link encap:Ethernet  HWaddr 00:00:00:00:00:0b
          inet6 addr: fe80::200:ff:fe00:b/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

fwdintf   Link encap:Ethernet  HWaddr 00:00:00:00:00:0a
          inet6 addr: fe80::200:ff:fe00:a/64 Scope:Link
          UP RUNNING NOARP MULTICAST  MTU:1482  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:2 errors:0 dropped:1 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:0 (0.0 B)  TX bytes:140 (140.0 B)

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
          RX packets:0 errors:0 dropped:0 overruns:0 frame:0
          TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:0 (0.0 B)  TX bytes:0 (0.0 B)

lo:0      Link encap:Local Loopback
          inet addr:1.1.1.1  Mask:255.255.255.255
          UP LOOPBACK RUNNING  MTU:1500  Metric:1
```

3. (Optional) Configure a proxy server (`http_proxy`, `https_proxy`), as needed.

```
http_proxy=http://proxy.youtube.com:8080
https_proxy=https://proxy.youtube.com:8080
```

4. Install the Puppet Agent.

```
[xr-vm_node0_RP0_CPU0:~]$ wget http://yum.puppetlabs.com/RPM-GPG-KEY-puppetlabs
[xr-vm_node0_RP0_CPU0:~]$ wget http://yum.puppetlabs.com/RPM-GPG-KEY-reductive
[xr-vm_node0_RP0_CPU0:~]$ rpm --import RPM-GPG-KEY-puppetlabs RPM-GPG-KEY-reductive
[xr-vm_node0_RP0_CPU0:~]$ yum install
http://yum.puppetlabs.com/puppetlabs-release-pc1-cisco-wrlinux-7.noarch.rpm
```

5. Edit the Puppet Agent configuration file at `/etc/puppetlabs/puppet/puppet.conf` with Puppet Server identification and Agent settings.

The Puppet Agent is successfully installed on IOS XR.

Creating a Puppet Manifest

This section explains how you can create a Puppet manifest on the Puppet Server for installing an application, such as iPerf (RPM file).

Prerequisites

Ensure the following requirements are met before you proceed:

- You have access to the application package compatible with the native IOS XR environment.
- Target application package is hosted on an accessible repository, or downloaded to a boot flash.

Configuration Procedure

To create a sample Puppet manifest to start the `bootlogd` service, and install iPerf on IOS XR, follow these steps:

1. Create a Puppet manifest on Puppet Server to install your application.

   The manifest must be created in the `/etc/puppetlabs/code/environments/production/manifests/` directory. For it to be launched automatically by Puppet Server, the manifest file must be named, `site.pp`.

```
# Manifest to demo builtin providers
#

class ciscopuppet::demo_builtin_providers {

  $package = 'iperf'
  $service = 'bootlogd'

  yumrepo { 'wrl7-repo':
    ensure => present,
    name => 'wrl7-repo',
    baseurl => 'http://10.105.247.73/wrl7_yum_repo/',
    gpgcheck => 0,
    enabled => 1,
    proxy => '_none_',
  }

  package { $package:
    ensure => present,
    require => Yumrepo['wrl7-repo'],
```

```
  }

  service { $service:
    ensure => running,
  }

}

node 'default' {
  include ciscopuppet::demo_builtin_providers
}
```

2. Access and trigger the Puppet Agent to converge the system based on the manifest defined on the Puppet Server.

The iPerf RPM is installed on IOS XR by the Puppet Agent.