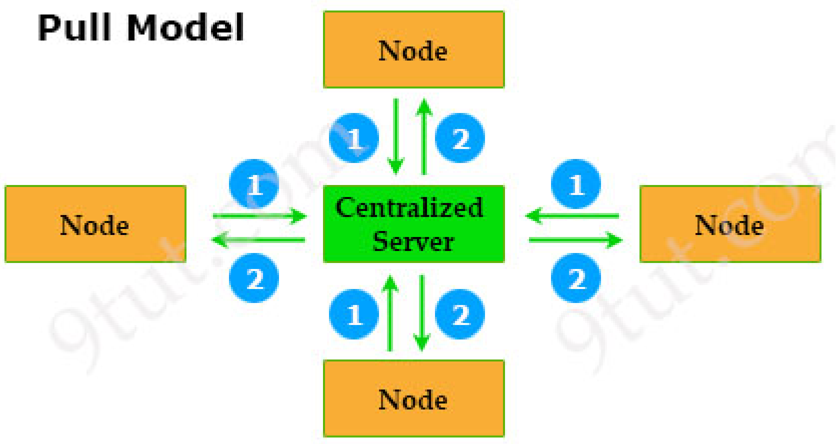
Puppet Tutorial

December 23rd, 2020 Go to comments

Besides Ansible and Chef, Puppet is another automation tool in CCNA certification so in this tutorial we will learn about it.

Puppet is built on server-client architecture which comprises a master (centralized server) and some/many nodes (clients). In each node, a Puppet Agent is installed to communicate with the Puppet Master. Puppet Master is the place where all Puppet codes are written and stored. These codes dictate the instructions for performing various tasks for the client. If the Clients need something, they simply request them.

Puppet is based on a Pull deployment model, where the nodes check in regularly after every 1800 seconds with the Master to see if anything needs to be updated in the agent. If anything needs to be updated the agent pulls the necessary Puppet codes from the Master and performs required actions.



Puppet Master Components

Manifests

Manifest is the most important component in a Puppet Master so we will mention about it first. Manifest is just the file where the all Puppet scripts for configuring Puppet clients are written (in Ruby code). Manifest filenames use ".pp" (means Puppet policy) extension.

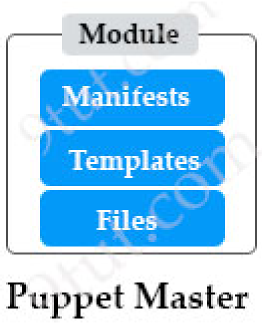
Based on the Facts received from Factor, Master compiles manifests into catalogs (which will be discussed later), then sends them to the client.

Module

Module also plays an important part in a Puppet Master. Module is a collection of manifests and other related data files organized in a predefined way to facilitate sharing and reusing. Modules tie manifests, templates, and files into a single unit.

Module = Manifests + Data (Templates, Files)

Modules have a specific directory path which is usually “/etc/puppet/manifests/”. They are useful for organizing our Puppet code, because they allow to split code into multiple manifests. It is considered best practice to use modules to organize all of our Puppet manifests.



Templates

Templates are typically used to set up configuration files, allowing for the use of variables and other features intended to make these files more versatile and reusable.

Catalogs

The entire configuration and manifest files that are written in Puppet are changed into a compiled format. This compiled format is known as a catalog, which can be applied to the target node. All the desired states of client resources are described in the catalog.

Catalog = Facts + Manifests

Other components of Puppet Master are: Resource (a basic unit of system configuration modeling), Class (like class in programming languages, to organize the code in a better way. Puppet class is a collection of various resources that are grouped into a single unit)

Puppet Client Components

Agent

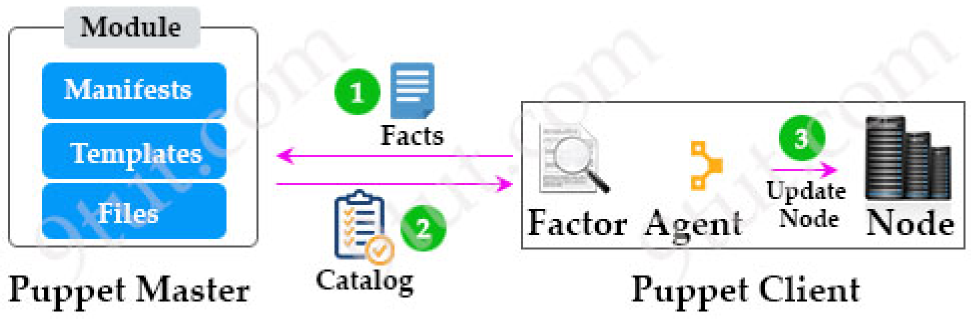
Agent is the program installed on each client to apply the configuration catalogs that it pulls from the Puppet Master to the client.

Factor (or Facter)

The factor collects *facts*, which are important information about the node and sends them to the Puppet Master. Facts are the key-value data pair which represents puppet client states such as IP address, operating system, network interface, uptime and whether the client machine is virtual or not…

Based on the facts received, Puppet Master compiles manifests into catalogs, then sends them to the client. On the client side, Agents execute any required changes and send reports back to the Master. If a system fails, the Master has a record of all system changes for a rollback to a previous working state.

All of the above components of Puppet Master and Client can be summarized in the figure below:



1. Puppet Agent sends data about its state to the Puppet Master (includes the hostname, kernel details, IP address, file name details…)
2. Puppet Master does analysis on the data, and if any change is needed (such as package installation, upgrades or removals, file system creation, user creation or deletion, server reboot, IP configuration changes…), it sends the required changes to the client via Catalog. For example, after analysis, Master decides to send a new software version to the Agent and asks it to install.
3. The Agent installs the required update for the Node and reply back to the Master that it has upgraded the software successfully.

Puppet code

An example of the manifest *vlan.pp* which is for creating and enabling VLAN 10 with its name configured to '9tut':

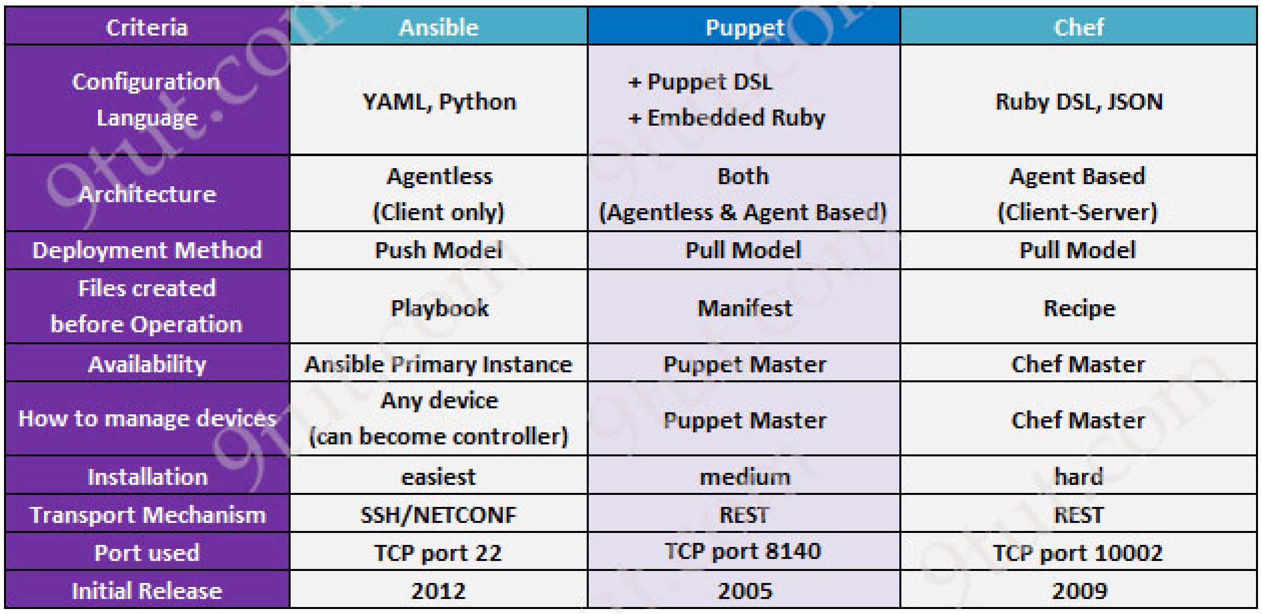
```
class cnos::vlan {
  cnos_vlan { '10':
    ensure      => 'present',
    vlan_id     => 10,
    admin_state => 'up',
    vlan_name   => '9tut',
  }
}
```

In order to apply this manifest on the Puppet Master, we can use "puppet apply " command:

```
$ puppet apply vlan.pp
Notice: Compiled catalog for puppetmaster.9tut in environment production in 0.03 seconds
Notice: Finished catalog run in 0.03 seconds
```

In summary, please remember the following important facts about Puppet:
+ Use "pull" model
+ Use TCP port 8140 to reach Puppet Master
+ Use Ruby for device configuration
+ Files needed for operation: Manifest, Templates…
+ Puppet Master only works on Linux/Unix and Puppet Agents also works on Windows.

We also made a comparison list of Ansible, Puppet and Chef automation tool here:



Criteria	Ansible	Puppet	Chef
Configuration Language	YAML, Python	+ Puppet DSL + Embedded Ruby	Ruby DSL, JSON
Architecture	Agentless (Client only)	Both (Agentless & Agent Based)	Agent Based (Client-Server)
Deployment Method	Push Model	Pull Model	Pull Model
Files created before Operation	Playbook	Manifest	Recipe
Availability	Ansible Primary Instance	Puppet Master	Chef Master
How to manage devices	Any device (can become controller)	Puppet Master	Chef Master
Installation	easiest	medium	hard
Transport Mechanism	SSH/NETCONF	REST	REST
Port used	TCP port 22	TCP port 8140	TCP port 10002
Initial Release	2012	2005	2009

Comments (3) Comments

1. Jey0195
   April 27th, 2021

   Geat Summary. Thanks!

2. Johnson
   May 5th, 2021

   Good Idea

3. ALBERT COMAN
   October 30th, 2021

   You have great teaching skills, thanks for running this site.

Add a Comment

Name

Submit Comment

Subscribe to comments feed

CCNAv7 (2020) – New Questions Part 4 Section 3 Chef Tutorial

Premium Member Zone

Welcome Gurjeet singh!

CCNA 200-301

CCNA Self-Study

- Practice CCNA GNS3 Labs
- CCNA Knowledge
- CCNA Lab Challenges
- Puppet Tutorial
- Chef Tutorial
- Ansible Tutorial
- JSON Tutorial
- Layer 2 Threats and Security Features
- AAA TACACS+ and RADIUS Tutorial
- STP Root Port Election Tutorial
- GRE Tunnel Tutorial
- Basic MPLS Tutorial
- TCP and UDP Tutorial
- Border Gateway Protocol BGP Tutorial
- Point to Point Protocol (PPP) Tutorial
- WAN Tutorial
- DHCP Tutorial
- Simple Network Management Protocol SNMP Tutorial
- Syslog Tutorial
- Gateway Load Balancing Protocol GLBP Tutorial
- EtherChannel Tutorial
- Hot Standby Router Protocol HSRP Tutorial
- InterVLAN Routing Tutorial
- Cisco Command Line Interface CLI
- Cisco Router Boot Sequence Tutorial
- OSI Model Tutorial
- Subnetting Tutorial – Subnetting Made Easy
- Frame Relay Tutorial
- Wireless Tutorial
- Virtual Local Area Network VLAN Tutorial
- VLAN Trunking Protocol VTP Tutorial
- IPv6 Tutorial
- Rapid Spanning Tree Protocol RSTP Tutorial
- Spanning Tree Protocol STP Tutorial
- Network Address Translation NAT Tutorial
- Access List Tutorial
- RIP Tutorial
- EIGRP Tutorial
- OSPF Tutorial

Network Resources

- Free Router Simulators
    - CCNA Website
    - ENCOR Website
    - ENSDWI Website
    - ENARSI Website
    - DevNet Website
    - CCIE R&S Website
    - Security Website
    - Wireless Website
    - Design Website
    - Data Center Website
    - Service Provider Website
    - Collaboration Website