Devops

Puppet Interview Questions

By Devops - April 07, 2017

http://thelinuxstuff.blogspot.com/2015/04/how-does-puppet-work.html

How Does Puppet Work ?

Puppet Master:

This machine contains all the configuration for different hosts.

Puppet master will run as a daemon on this master server.

Puppet Agent:

This is the daemon that will run on all the servers, which are to be managed using puppet.

Puppet agent will go and ask the configuration for itself from the puppet master server

at a specific time interval.

- Puppet agent is a daemon that runs on all the client servers.
- The connection between puppet agent and master is made in a secure encrypted channel with the help of SSL.
- 30 minutes is the default interval when puppet agent daemon will go and fetch config data from puppet master.

Steps involved whenever a puppet agent of any node connects to a puppet master server :

Step 1: Whenever a client node connects to the master, the master server analyses the

configuration to be applied to the node, and how to apply that configs on the node.

Step 2: Puppet master server Takes and collects all the resources and configurations

to be applied to the node, and compiles it and make it a catalog. This catalog is given

to the puppet agent of the node.

Step 3: Puppet agent will apply the configuration on the node, according to the catalog,

and then reply back, and submit the report of the configuration applied to the puppet master server.

http://devopschef.blogspot.com/2016/01/puppet-interview-questions-answers.html
Q: – What is Puppet ?
Puppet is a configuration Tool which is use to automate administration tasks.Puppet Agent(Client) sends request to Puppet Master (Server) and Puppet Master Push Configuration on Agent.
Q: – What is Manifests ?
Manifests, in Puppet, are the files in which the client configuration is specified.

Q: – What is Module and How it is different from Manifest ?
Whatever the manifests we defined in modules, can call or include into other manifests. Which makes easier management of Manifests.It helps you to push specific manifests on specific Node or Agent.
Q: – Command to check requests of Certificates ?
puppetca –list (2.6)
puppet ca list (3.0)
Q: – Command to sign Requested Certificates
puppetca –sign hostname-of-agent (2.6)
puppet ca sign hostname-of-agent (3.0)
Q: – Where Puppet Master Stores Certificates
/var/lib/puppet/ssl/ca/signed
Q: – What is Facter ?
Sometime you need to write manifests on conditional experession based on agent specific data which is available through Facter. Facter provides information like Kernel version,Dist release, IP Address, CPU info and etc.You can defined your facter also.
Q: – What is the use of etckeeper-commit-post and etckeeper-commit-pre on Puppet Agent ?
etckeeper-commit-post: In this configuration file you can define command and scripts which executes after pushing configuration on Agent
Etckeeper-commit-pre: In this configuration file you can define command and scripts which executes before pushing configuration on Agent
Q: – What is Puppet Kick ?
By default Puppet Agent request to Puppet Master after a periodic time which known as "runinterval". Puppet Kick is a utility which allows you to trigger Puppet Agent from Puppet Master.
Q: – What is MCollective ?
MCollective is a powerful orchestration framework. Run actions on thousands of servers simultaneously, using existing plugins or writing your own.

http://mrsripati.blogspot.com/2016/08/top-30-linux-system-admin-interview.html

Q:1 Why LVM is required ?
Ans: LVM stands for Logical Volume Manager , to resize filesystem's size online we required LVM partition in Linux. Size of LVM partition can be extended and reduced using the lvextend & lvreduce commands respectively.
Q:2 How To check Memory stats and CPU stats ?
Ans: Using 'free' & 'vmstat' command we can display the physical and virtual memory statistics respectively.With the help of 'sar' command we see the CPU utilization & other stats.
Q:3 What does Sar provides and at which location Sar logs are stored ?
Ans: Sar Collect, report, or save system activity information. The default version of the sar command (CPU utilization report) might be one of the first facilities the user runs to begin system activity investigation, because it monitors major system resources. If CPU utilization is near 100 percent (user + nice + system), the workload sampled is CPU-bound.
By default log files of Sar command is located at /var/log/sa/sadd file, where the dd parameter indicates the current day.
Q:4 How to increase the size of LVM partition ?
Ans: Below are the Logical Steps :
– Use the lvextend command (lvextend -L +100M /dev/<Name of the LVM Partition> , in this example we are extending the size by 100MB.
– resize2fs /dev/<Name of the LVM Partition>
– check the size of partition using 'df -h' command
Q:5 How to reduce or shrink the size of LVM partition ?
Ans: Below are the logical Steps to reduce size of LVM partition :
-Umount the filesystem using umount command,
-use resize2fs command , e.g resiz2fs /dev/mapper/myvg-mylv 10G
-Now use the lvreduce command , e.g lvreduce -L 10G /dev/mapper/myvg-mylv
Above Command will shrink the size & will make the filesystem size 10GB.
Q:6 How to create partition from the raw disk ?
Ans: Using fdisk utility we can create partitions from the raw disk.Below are the steps to create partition from the raw dsik :
– fdisk /dev/hd* (IDE) or /dev/sd* (SCSI)
– Type n to create a new partition
– After creating partition , type w command to write the changes to the partition table.
Q:7 Where the kernel modules are located ?
Ans: The '/lib/modules/kernel-version/' directory stores all kernel modules or compiled drivers in Linux operating system. Also with 'lsmod' command we can see all the installed kernel modules.
Q:8 What is umask ?
Ans: umask stands for 'User file creation mask', which determines the settings of a mask that controls which file permissions are set for files and directories when they are created.
Q:9 How to set the umask permanently for a user?
Ans: To set this value permanently for a user, it has to be put in the appropriate profile file which depends on the default shell of the user.
Q:10 How to change the default run level in linux ?
Ans: To change the run level we have to edit the file "/etc/inittab" and change initdefault entry (id:5:initdefault:). Using 'init' command we change the run level temporary like 'init 3' , this command will move the system in runlevl 3.
Q:11 How to share a directory using nfs ?

Ans: To share a directory using nfs , first edit the configuration file '/etc/exportfs' , add a entry like '/<directory-name> <ip or Network>(Options)' and then restart the nfs service.

Q:12 How to check and mount nfs share ?

Ans: Using 'showmount' command we can see what directories are shared via nfs e.g 'showmount -e <ip address of nfs server>'.Using mount command we can mount the nfs share on linux machine.

Q:13 What are the default ports used for SMTP,DNS,FTP,DHCP,SSH and squid ?

Ans: Service Port

SMTP 25

DNS 53

FTP 20 (data transfer) , 21 (Connection established)

DHCP 67/UDP(dhcp server) , 68/UDP(dhcp client)

SSH 22

Squid 3128

Q:14 What is Network Bonding ?

Ans: Network bonding is the aggregation of multiple Lan cards into a single bonded interface to provide fault tolerance and high performance. Network bonding is also known as NIC Teaming.

Q:15 What are the different modes of Network bonding in Linux ?

Ans: Below are list of modes used in Network Bonding :

balance-rr or 0 – round-robin mode for fault tolerance and load balancing.

active-backup or 1 – Sets active-backup mode for fault tolerance.

balance-xor or 2 – Sets an XOR (exclusive-or) mode for fault tolerance and load balancing.

broadcast or 3 – Sets a broadcast mode for fault tolerance. All transmissions are sent on all slave interfaces.

802.3ad or 4 – Sets an IEEE 802.3ad dynamic link aggregation mode. Creates aggregation groups that share the same speed & duplex settings.

balance-tlb or 5 – Sets a Transmit Load Balancing (TLB) mode for fault tolerance & load balancing.

balance-alb or 6 – Sets an Active Load Balancing (ALB) mode for fault tolerance & load balancing.

Q:16 How to check and verify the status the bond interface.

Ans: Using the command 'cat /proc/net/bonding/bond0' , we can check which mode is enabled and what lan cards are used in this bond. In this example we have one only one bond interface but we can have multiple bond interface like bond1,bond2 and so on.

Q:17 How to check default route and routing table ?

Ans: Using the Commands 'netstat -nr' and 'route -n' we can see the default route and routing tables.

Q:18 How to check which ports are listening in my Linux Server ?

Ans: Use the Command 'netstat –listen' and 'lsof -i'

Q:19 List the services that are enabled at a particular run level in linux server ?

Ans: With the help of command 'chkconfig –list | grep 5:on' we can list all the service that are enabled in run level5. For other run levels just replace 5 with the respective run level.

Q:20 How to enable a service at a particular run level ?

Ans: We can enable a service using the Command 'chkconfig <Service-Name> on –level 3'

Q:21 How to upgrade Kernel in Linux ?

Ans: We should never upgrade Linux Kernel , always install the new New kernel using rpm command because upgrading a kenel can make your linux box in a unbootable state.

Q:22 How To scan newly asssigned luns on linux box without rebooting ?

Ans: There are two ways to scan newly assigned luns :

Method:1 if sg3 rpm is installed , then run the command 'rescan-scsi-bus.sh'

Method:2 Run the Command , echo " – – – " > /sys/class/scsi_host/hostX/scan

Q:23 How to find WWN numbers of HBA cards in Linux Server ?

Ans: We can find the WWN numbers of HBA cards using the command 'systool -c fc_host -v | grep port_name'

Q:24 How to add & change the Kernel parameters ?

Ans: To Set the kernel parameters in linux , first edit the file '/etc/sysctl.conf' after making the changes save the file and run the command 'sysctl -p' , this command will make the changes permanently without rebooting the machine.

Q:25 What is Puppet Server ?

Ans: Puppet is an open-source & enterprise software for configuration management toll in UNIX like operating system. Puppet is a IT automation software used to push configuration to its clients (puppet agents) using code. Puppet code can do a variety of tasks from installing new software, to check file permissions, or updating user accounts & lots of other tasks.

Q:26 What are manifests in Puppet ?

Ans: Manifests in Puppet are the files in which the client configuration is specified.

Q:27 Which Command is used to sign requested certificates in Puppet Server ?

Ans: 'puppetca –sign hostname-of-agent' in (2.X) & 'puppet ca sign hostname-of-agent' in (3.X)

Q:28 At which location Puppet Master Stores Certificates ?

Ans: /var/lib/puppet/ssl/ca/signed

Q:29 How to find all the regular files in a directory ?

Ans: using the command 'find /<directory -type f'.

Q:30 What is load average in a linux ?

Ans: Load Average is defined as the average sum of the number of process waiting in the run queue and number of process currently executing over the period of 1,5 and 15 minutes. Using the 'top' and 'uptime' command we find the load average of a linux sever.

http://pnaplinux.blogspot.com/2015/04/linux-interview-questions.html

Linux interview questions

Question 1: What is puppet server?

Answer: Puppet is a configuration management system that allows you to define the state of your IT infrastructure, then automatically enforces the correct state. Whether you're managing just a few servers or thousands of physical and virtual machines, Puppet automates tasks that sysadmins often do manually, freeing up time and mental space so sysadmins can work on the projects that deliver greater business value.

Whether you're deploying vendor-supplied applications or working with a team of internal software developers, Puppet automates every step of the software delivery process: from provisioning of physical and virtual machines to orchestration and reporting; from early-stage code development through testing, production release and updates. Puppet ensures consistency, reliability and stability. It also facilitates closer collaboration between sysadmins and developers, enabling more efficient delivery of cleaner, better-designed code.

Question 2: What is .bashrc file?

Answer: .bashrc is a shell script that Bash runs whenever it is started interactively. You can put any command in that file that you could type at the command prompt. You put commands here to set up the shell for use in your particular environment, or to customize things to your preferences.

Question 3: What is the different between rpm and yum?

Answer: The Red Hat Package Manager or RPM is the default package manager for Linux distributions that use packages with the same name. Initially developed by Red Hat, it eventually found widespread acceptance in a lot of Linux distributions. YUM stands for Yellowdog Updater Modified and is a front end for Linux distributions that utilize the RPM package format. Both of these are only usable with RPM based distros and are not usable with those that use debian packages like Ubuntu.

Although RPM is a very robust tool that a lot of users are already familiar with, there are still some minor flaws that are an annoyance to users. The most prominent problem is a state commonly referred to by most people as 'dependency hell'. This problem occurs with packages that depend on a lot of other packages, some of those packages also depend on a lot of other packages. It is common knowledge that you must install all dependencies for the program to work correctly. RPM is unable to automatically do this for you. It can only check whether all the required packages are installed prior to installing the needed package. Manually tracking and installing each dependency is a major chore for most people who only want to install a single package initially.

YUM is capable of tracking the dependencies of a package and installing them prior to installing the package that the user wanted to install. This simplifies the whole process as you need only know the name of the package that you want to install and not worry whether the required packages have been installed or not. Packages that can't be found on the system are searched for in the repositories that are available to the system.

Although both RPM and YUM are what really installs the packages, you would probably not be using either of those unless you are proficient with command lines and the various parameters that need to be passed. To make it easier for ordinary people to quickly grasp total control of their system, there are various graphical user interfaces or GUIs that can be used on top of either YUM or RPM. These GUIs are what people commonly see and interact with and not YUM or RPM.

Summary:

1. RPM is a package manager while YUM is a frontend that can be used with RPM.
2. The RPM package manager is unable to track dependencies while YUM can.

Question 4: What is different between telnet and SSH?

Answer: Telnet vs SSH

Secure Shell, commonly known as SSH, and Telnet are two network protocols that have been used widely at one point in time or another. They are both used to connect to remote servers in order to facilitate some sort of communications. The primary difference, which also led to one superseding the other, is in security. SSH offers security mechanisms that protect the users against anyone with malicious intent while Telnet has no security measures whatsoever.

Telnet was designed to work within a private network and not across a public network where threats can appear. Because of this, all the data is transmitted in plain text, including passwords. This is a major security issue and the developers of SSH used encryptions to make it harder for other people to sniff the password and other relevant information. Telnet also omits another safety measure called authentication. This ensures that the source of the data is still the same device and not another computer. Without authentication, another person can intercept the communication and do what he wishes. This is also addressed in SSH as it uses a public key to authenticate the source of the data.

Due to the security measures that were necessary for SSH to be used in public networks, each packet contains less data to make room for the data of the security mechanisms. In order to transmit the same amount of data, you would need to take-up a lot more bandwidth. This is called overhead and was such a major issue back when internet speeds were very low because it translates to a performance hit.

The security issues of Telnet forced a lot of people to use SSH in order to protect themselves. It didn't take a long time before SSH replaced Telnet in a great majority of its uses. Telnet did not fade away though as it is still used in some areas, mostly in testing and debugging. Telnet extensions were developed to provide security but they are not used in most Telnet implementations.

Summary:

1. SSH and Telnet commonly serves the same purpose
2. SSH is more secure compared to Telnet
3. SSH encrypts the data while Telnet sends data in plain text
4. SSH uses a public key for authentication while Telnet does not use any authentication
5. SSH adds a bit more overhead to the bandwidth compared to Telnet
6. Telnet has been all but replaced by SSH in almost all uses

Question 5: How can restrict SSH access for a particular user?

Answer: Restricting which users can log in

Open the configuration file **vi /etc/ssh/sshd_config**

The syntax is:

DenyUsers user1 user2 user3

Use DenyUsers to block user login. You can use wild cards as well as user1@porulagam.com (user1 is not allowed to login from porulagam.com host) pattern.

DenyGroups group1 group2

A list of group names, if user is part of primary of supplementary group login access is denied. You can use wildcards. Please note that you cannot use a numeric group or username ID. If these directives are not used, default is to allow everyone.

PermitRootLogin no

Allowing selected users or group explicitly to log in

The syntax is:

AllowUsers user1 user2

This directive is opposite of DenyUsers directive i.e. user1 and user2 are only allowed to log in into the server.

AllowGroups group1 group2

This directive is opposite of DenyGroups directive i.e. members of group1 and group2 users are only allowed to log in into the server.

Question 6: How to check the running ports?

Answer: # netstat -tulpn

Question 7. How will you restrict ip in FTP server?

Answer: while it is possible to do what you want, only making the public server available out of office hours, this is not the ideal solution. It is reasonable to assume that those abusing your server are not always putting legal material there, which could lead to legal action against you or losing your Internet connection. Remember, this is your server and you are responsible for the content available from it. Providing anonymous upload and download capability is asking for trouble. If you must do this, keep the upload area separate from the downloads, so people cannot download material that has been anonymously uploaded, it has to be moved over by someone with a login account. A better solution is to disable anonymous uploads altogether and provide your clients with their own FTP accounts. If you really want to continue offering unrestricted anonymous access out of office hours, you can use the hosts.allow and hosts.deny files in /etc. You enable this by putting

tcp_wrappers=YES

in /etc/vsftpd.conf. Then ensure your local network has access - add this line to /etc/hosts.allow

vsftpd: 192.168.1.

Note the trailing "." on the address to match the whole subnet, change the address to match your network. Now put these two lines into /etc/ cron.d/vsftpd

0 18 * * 1-5 root sed -i '/^vsftpd/d'
/etc/hosts.deny
0 8 * * 1-5 root echo "vsftpd:
ALL" >>/etc/hosts.deny

and force cron to reload with

killall -HUP cron

This will deny modify hosts.deny to deny all addresses, except those specified in hosts.allow, between 0800 and 1800 Monday to Friday and clear the block at other times.

Question 8: What are daemons?

Answer: A daemon is a type of program on Unix-like operating systems that runs unobtrusively in the background, rather than under the direct control of a user, waiting to be activated by the occurrence of a specific event or condition.

Question 9: What is the default port number of HTTP?

Answer: By default, HTTP uses port 80 and HTTPS uses port 443 but a URL like http://www.example.com:8080/path/ specifies that the web resource be served by the HTTP server on port 8080.

Question 10: How will you change the default port no of HTTP?

Answer:

1. Assume that your new port number is 78
2. cp /etc/httpd/ports.conf /etc/httpd/ports.conf_backupgedit /etc/httpd/ports.conf
3. Find this line- Listen 80
4. Replace with the following line – Listen 78
5. Save the edited file
6. /etc/init.d/httpd restartApache HTTP server, Linux

Question 11: How can I run the particular script file when the user log in?

Answer: You should setup a .bashrc & .bash_profile for your user account

Question 12: Network is not working, What are the trouble shooting?

Answer: Refer the link, which explains the various commands on this question.

http://www.tecmint.com/linux-network-configuration-and-troubleshooting-commands/

Question 13: What is the different between soft links and hard links?

Answer: A symbolic link is a link to another name in the file system. Once a hard link has been made the link is to the inode. deleting renaming or moving the original file will not affect the hard link as it links to the underlying inode. Any changes to the data on the inode is reflected in all files that refer to that inode.

Question 14: What is cron in Linux?

Answer: Crontab is the short form of **Chronological Table**. Cron is a daemon that executes scheduled commands. Cron is started automatically from /etc/init.d on entering multi-user runlevels. Cron searches its spool area (/var/spool/cron/crontabs) for crontab files (which are named after accounts in /etc/passwd); crontabs found are loaded into memory.

Six fields of crontab in Linux

A number (or list of numbers, or range of numbers), m, representing the **minute** of the hour;
A number (or list of numbers, or range of numbers), h, representing the **hour** of the day;
A number (or list of numbers, or range of numbers), dom, representing the **day of the month;**
A number (or list, or range), or name (or list of names), mon, representing the **month** of the year;

A number (or list, or range), or name (or list of names), dow, representing the **day of the week**; and
command, which is the command to be run, exactly as it would appear on the **command** line.

Question 15: How would you reset the root password?

Answer:

This method of reseting/recovering of lost Linux root password should work on most of linux distributions. ...
Edit Grub boot menu options. First you need to get into grub menu options. ...
Remount / and /proc. ...
reset / recover forgotten linux root password. ...
Reboot.

Question 16: Web server is running in the server. What are the troubleshooting, if it is not accessible from the client?

Answer: Check the configuration file /etc/httpd/conf/httpd.conf
Check the default configuration settings
Reset the service
Check the enableness of the port number in the firewall.
Check the SE linux as well

Question 17: What is the different between locate and slocate commands?

Answer:

locate command is used for finding the output in all locations. But the updatedb command has to be given earlier.
slocate command enables system users to search entire file systems without displaying unauthorized files.

Question 18: I want to search a word Linux in all the text file in the current directory. How can I do it?

Answer:

grep "text string to search" directory-path

OR

grep [option] "text string to search" directory-path

OR

grep -r "text string to search" directory-path

OR

grep -r -H "text string to search" directory-path

OR

egrep -R "word-1|word-2" directory-path

OR

egrep -w -R "word-1|word-2" directory-path

Question 19: How to send a mail with out a mail client using command line?

Answer:

Mail without attachment

mail -s "Test Subject" user@example.com < /dev/null

Mail with attachment

mail -a /opt/backup.sql -s "Backup File" user@example.com < /dev/null

Question 20: How will you find all the files which are accessed last 30 days?

Answer: Use these options with find command
-mtime +60 means you are looking for a file modified 60 days ago.
-mtime -60 means less than 60 days.

-mtime 60 If you skip + or - it means exactly 60 days.

Question 21: How can I filter only first two columns?

Answer: awk '/^UUID/ {print $1;}' /etc/fstab

Question 22: How will take the backup of mysql database?

Answer: mysqldump -u root -p --all-databases > alldb_backup.sql

Question 23: Basic command to view the content of the table in mysql?

Answer:

create table tablename;
select * from table;
select * from table where (condition);

Question 24: How to run java program in Linux?

Answer: javac filename.java (for java program compilation)
java classname (for running java program)

Question 25: What is make command in Linux?
Answer: make is a utility for building and maintaining groups of programs (and other types of files) from source code.

---

http://linux.amitmaheshwari.in/2015/01/nagios-interview-questions.html

---

https://www.edureka.co/blog/interview-questions/puppet-interview-questions/

Q1. What is Puppet?

I will advise you to first give a small definition of Puppet. Puppet is a Configuration Management tool which is used to automate administration tasks.

Now, you should describe how Puppet Master and Agent communicates.

Puppet has a Master-Slave architecture in which the Slave has to first send a Certificate signing request to Master and Master has to sign that Certificate in order to establish a secure connection between Puppet Master and Puppet Slave as shown on the diagram below. Puppet Slave sends request to Puppet Master and Puppet Master then pushes configuration on Slave.

Refer the diagram below that explains the above description:

Puppet Master Slave Connection - Puppet Interview Questions - Edureka

Q2. How Puppet Works?

For this question just explain Puppet Architecture. Refer the diagram below:


Puppet Master Slave Architecture - Puppet Interview Questions - Edureka

The following functions are performed in the above image:

- The Puppet Agent sends the Facts to the Puppet Master. Facts are basically key/value data pair that represents some aspect of Slave state, such as its IP address, up-time, operating system, or whether it's a virtual machine. I will explain Facts in detail later in the blog.
- Puppet Master uses the facts to compile a Catalog that defines how the Slave should be configured. Catalogis a document that describes the desired state for each resource that Puppet Master manages on a Slave. I will explain catalogs and resources in detail later.
- Puppet Slave reports back to Master indicating that Configuration is complete, which is visible in the Puppet dashboard.

Now the interviewer might dig in deep, so the next set of Puppet interview questions will test your knowledge about various components of Puppet.

Q3. What are Puppet Manifests?

Every node (or Puppet Agent) has got its configuration details in Puppet Master, written in the native Puppet language. These details are written in the language which Puppet can understand and are termed as Manifests. Manifests are composed of Puppet code and their filenames use the .pp extension.

Now give an example, you can write a manifest in Puppet Master that creates a file and installs apache on all Puppet Agents (Slaves) connected to the Puppet Master.

Q4. What is Puppet Module and How it is different from Puppet Manifest?

A Puppet Module is a collection of Manifests and data (such as facts, files, and templates), and they have a specific directory structure. Modules are useful for organizing your Puppet code, because they allow you to split your code into multiple Manifests. It is considered best practice to use Modules to organize almost all of your Puppet Manifests.

Puppet programs are called Manifests. Manifests are composed of Puppet code and their file names use the .pp extension.

Q5. What is Facter in Puppet?

Facter is basically a library that discovers and reports the per-Agent facts to the Puppet Master such as hardware details, network settings, OS type and version, IP addresses, MAC addresses, SSH keys, and more. These facts are then made available in Puppet Master's Manifests as variables.

Q6. What is Puppet Catalog?

When configuring a node, Puppet Agent uses a document called a catalog, which it downloads from a Puppet Master. The catalog describes the desired state for each resource that should be managed, and may specify dependency information for resources that should be managed in a certain order.

If your interviewer wants to know more about it mention the below points:

Puppet compiles a catalog using three main sources of configuration info:

- Agent-provided data
- External data
- Puppet manifests

Q7. What size organizations should use Puppet?

There is no minimum or maximum organization size that can benefit from Puppet, but there are sizes that are more likely to benefit. Organizations with only a handful of servers are unlikely to consider maintaining those servers to be a real problem, Organizations with many servers are more likely to find, difficult to manage those servers manually so using Puppet is more beneficial for those organizations.

Q8. How should I upgrade Puppet and Facter?

The best way to install and upgrade Puppet and Facter is via your operating system's package management system, using either your vendor's repository or one of Puppet Labs' public repositories.

If you have installed Puppet from source, make sure you remove old versions entirely (including all application and library files) before upgrading. Configuration data (usually located in/etc/puppet or /var/lib/puppet, although the location can vary) can be left in place between installs.

The next set of Puppet Interview Questions will test your experience with Puppet.

Q9. What is the Command to check requests of Certificates from Puppet Agent (Slave) to Puppet Master?

To check the list of Certificate signing requests from Puppet Agent to Puppet Master execute **puppet cert list** command in Puppet Master.

I will advise you to also add:

If you want to sign a particular Certificate execute: **puppet cert sign <Hostname of agent>.** You can also sign all the Certificates at once by executing: **puppet cert sign all**.

Q10. What is the use of etckeeper-commit-post and etckeeper-commit-pre on Puppet Agent?

Answer to this question is pretty direct just tell the uses of the above commands:

- **etckeeper-commit-post:** In this configuration file you can define command and scripts which executes after pushing configuration on Agent.
- **etckeeper-commit-pre**: In this configuration file you can define command and scripts which executes before pushing configuration on Agent.

I hope you have enjoyed the above set of Puppet interview questions, the next set of questions will be more challenging, so be prepared.

Q11. What characters are permitted in a class name? In a module name? In other identifiers?

Class names can contain lowercase letters, numbers, and underscores, and should begin with a lowercase letter. "::" (Scope Resolution Operator) can be used as a namespace separator.

The same rules should be used when naming defined resource types, modules, and parameters, although modules and parameters cannot use the namespace separator.

Variable names can include alphanumeric characters and underscores, and are case-sensitive.

Q12. Does Puppet runs on windows?

Yes. As of Puppet 2.7.6 basic types and providers do run on Windows, and the test suite is being run on Windows to ensure future compatibility.

Q13. Which version of Ruby does Puppet support?

I will suggest you to mention the below points in your answer:

- Certain versions of Ruby are tested more thoroughly with Puppet than others, and some versions are not tested at all. Run **ruby –version** to check the version of Ruby on your system.

- Starting with Puppet 4, Puppet Agent packages do not rely on the OS's Ruby version, as it bundles its own Ruby environment. You can install puppet-agent alongside any version of Ruby or on systems without Ruby installed.
- Puppet Enterprise (PE) also does not rely on the OS's Ruby version, as it bundles its own Ruby environment. You can install PE alongside any version of Ruby or on systems without Ruby installed.
- The Windows installers provided by Puppet Labs don't rely on the OS's Ruby version, and can be installed alongside any version of Ruby or on systems without Ruby installed.

Q14. Which open source or community tools do you use to make Puppet more powerful?

Explain about some tools that you have used along with Puppet to do a specific task. You can refer the below example:

Changes and requests are ticketed through Jira and we manage requests through an internal process. Then, we use Git and Puppet's Code Manager app to manage Puppet code in accordance with best practices. Additionally, we run all of our Puppet changes through our continuous integration pipeline in Jenkins using the beaker testing framework.

Q15. Tell me about a time when you used collaboration and Puppet to help resolve a conflict within a team?

Explain them about your past experience of Puppet and how it was useful to resolve conflicts, you can refer the below mention example:

The development team wanted root access on test machines managed by Puppet in order to make specific configuration changes. We responded by meeting with them weekly to agree on a process for developers to communicate configuration changes and to empower them to make many of the changes they needed. Through our joint efforts, we came up with a way for the developers to change specific configuration values themselves via data abstracted through Hiera. In fact, we even taught one of the developers how to write Puppet code in collaboration with us.

Q16. Can I access environment variables with Facter in Puppet?

I will suggest you to start this answer by saying:

Not directly. However, Facter reads in custom facts from a special subset of environment variables. Any environment variable with a prefix of FACTER_ will be converted into a fact when Facter runs.

Now explain the interviewer with an example:

```
1    $ FACTER_FOO="bar"
2     $ export FACTER_FOO</span>
3     $ facter | grep  'foo'</span>
4       foo => bar
```

The value of the FACTER_FOO environment variable would now be available in your Puppet manifests as $foo, and would have a value of 'bar'. Using shell scripting to export an arbitrary subset of environment variables as facts is left as an exercise for the reader.

Q17. What is the use of Virtual Resources in Puppet

First you need to define Virtual Resource.

Virtual Resources specifies a desired state for a resource without necessarily enforcing that state. Although virtual resources can only be declared once, they can be realized any number of times.

I will suggest you to mention the uses of Virtual Resources as well:

- Resources whose management depends on at least one of multiple conditions being met.
- Overlapping sets of resources which might be needed by any number of classes.
    - Resources which should only be managed if multiple cross-class conditions are met.

http://tekslate.com/puppet-interview-questions-and-answers

1. What is Puppet ?

Puppet is a configuration Tool which is use to automate administration tasks. Puppet Agent(Client) sends request to Puppet Master (Server) and Puppet Master Push Configuration on Agent.

2. What is Manifests ?

Manifests, in Puppet, are the files in which the client configuration is specified.

3. What is Module and How it is different from Manifest ?

Whatever the manifests we defined in modules, can call or include into other manifests. Which makes easier management of Manifests.It helps you to push specific manifests on specific Node or Agent.

4. What are the Commands to check requests of Certificates ?

puppetca –list (2.6)

puppet ca list (3.0)

5. What are the Commands to sign Requested Certificates ?

puppetca –sign hostname-of-agent (2.6)

puppet ca sign hostname-of-agent (3.0)

6. Where Puppet Master Stores Certificates

/var/lib/puppet/ssl/ca/signed

7. What is Facter ?

Sometime you need to write manifests on conditional expression based on agent specific data which is available through Facter. Facter provides information like Kernel version,Dist release, IP Address, CPU info and etc.You can defined your facter also.

8. What is the use of etckeeper-commit-post and etckeeper-commit-pre on Puppet Agent ?

etckeeper-commit-post: In this configuration file you can define command and scripts which executes after pushing configuration on Agente

etckeeper-commit-pre: In this configuration file you can define command and scripts which executes before pushing configuration on Agent

9. What is Puppet Kick ?

By default *Puppet Agent* request to Puppet Master after a periodic time which known as "runinterval". Puppet Kick is a utility which allows you to trigger Puppet Agent from Puppet Master.

10. What is MCollective ?

MCollective is a powerful orchestration framework. Run actions on thousands of servers simultaneously, using existing plugins or writing your own.

Puppet Devops Interview Questions

11. What's special about Puppet's model-driven design?

Traditionally, managing the configurations of a large group of computers has meant a series of imperative steps; in its rawest state, SSH and a *for* loop. This general approach grew more sophisticated over time, but it retained the more profound limitations at its root.

Puppet takes a different approach, which is to model everything — the current state of the node, the desired configuration state, the actions taken during configuration enforcement — as data: each node receives a catalog of resources and relationships, compares it to the current system state, and makes changes as needed to bring the system into compliance.

The benefits go far beyond just healing the headaches of configuration drift and unknown system state: modeling systems as data lets Puppet simulate configuration changes, track the history of a system over its lifecycle, and prove that refactored manifest code still produces the same system state. It also drastically lowers the barrier to entry for hacking and extending Puppet: instead of analyzing code and reverse-engineering the effects of each step, a user can just parse data, and sysadmins have been able to add significant value to their Puppet deployments with an afternoon's worth of perl scripting.

12. Why does Puppet have its own language? Why not use XML or YAML as the configuration format? Why not use Ruby as the input language?

The language used for manifests is ultimately Puppet's human interface, and XML and YAML, being data formats developed around the processing capabilities of computers, are horrible human interfaces. While some people are comfortable reading and writing them, there's a reason why we use web browsers instead of just reading the HTML directly. Also, using XML or YAML would limit any assurance that the interface was declarative — one process might treat an XML configuration differently from another.

13. Can Puppet manage workstations?

Yes, Puppet can manage any type of machine, and is used to manage many organizations that have a mix of laptops and desktops.

14. Does Puppet run on Windows?

Yes. As of Puppet 2.7.6 basic types and providers do run on Windows, and the test suite is being run on Windows to ensure future compatibility. More information can be found on the Puppet on Windows page, and bug reports and patches are welcome.

15. What size organizations should use Puppet?

There is no minimum or maximum organization size that can *benefit from Puppet,* but there are sizes that are more likely to benefit. Organizations with only a handful of servers are unlikely to consider maintaining those servers to be a real problem, while those that have more

need to consider carefully how they eliminate manual management tasks.

16. My servers are all unique; can Puppet still help?

Yes.

All servers are at least somewhat unique, but very few servers are entirely unique; host names and IP addresses (e.g.) will always differ, but nearly every server runs a relatively standard operating system. Servers are also often very similar to other servers within a single organization — all Solaris servers might have similar security settings, or all web servers might have roughly equivalent configurations — even if they're very different from servers in other organizations. Finally, servers are often needlessly unique, in that they have been built and managed manually with no attempt at retaining appropriate consistency.

Puppet can help both on the side of consistency and uniqueness. Puppet can be used to express the consistency that should exist, even if that consistency spans arbitrary sets of servers based on any type of data like operating system, data centre, or physical location. Puppet can also be used to handle uniqueness, either by allowing special provision of what makes a given host unique or through specifying exceptions to otherwise standard classes.

17. Who is Puppet Labs?

Puppet Labs (formerly Reductive Labs) is a small, private company focused on re-framing the server automation problem.

18. How should I upgrade Puppet and Facter?

The best way to install and upgrade Puppet and Facter is via your operating system's package management system, using either your vendor's repository or one of Puppet Labs' public repositories.

If you have installed Puppet from source, make sure you remove old versions entirely (including all application and library files) before upgrading. Configuration data (usually located in/etc/puppet or /var/lib/puppet, although the location can vary) can be left in place between installs.

19. What characters are permitted in a class name? In a module name? In other identifiers?

Class names can contain lowercase letters, numbers, and underscores, and should begin with a lowercase letter. "::" can be used as a namespace separator.

The same rules should be used when naming defined resource types, modules, and parameters, although modules and parameters cannot use the namespace separator.

Variable names can include alphanumeric characters and underscores, and are case-sensitive.

Puppet Interview Questions And Answers For Experienced

20. How do I document my manifests?

The puppet language includes a simple documentation syntax, which is currently documented on the Puppet Manifest Documentation wiki page. The puppetdoc command uses this inline documentation to automatically generate RDoc or HTML documents for your manifests and modules.

21. How do I manage passwords on Red Hat Enterprise Linux, CentOS, and Fedora Core?

As described in the Type reference, you need the Shadow Password Library, which is provided by the ruby-shadow package. The ruby-shadow library is available natively for fc6 (and higher), and should build on the corresponding RHEL and CentOS variants.

22. How do all of these variables, like $operatingsystem, get set?

The variables are all set by Facter. You can get a full listing of the available variables and their values by running facter by itself in a shell.

facter

Check out this blog post to learn more Puppet Tutorials.

23. Can I access environment variables with Facter?

Not directly. However, Facter reads in custom facts from a special subset of environment variables. Any environment variable with a prefix of FACTER_ will be converted into a fact when Facter runs. For example:

The value of the FACTER_FOO environment variable would now be available in your Puppet manifests as $foo, and would have a value of 'bar'. Using shell scripting to export an arbitrary subset of environment variables as facts is left as an exercise for the reader.

24. Why shouldn't I use autosign for all my clients?

It is very tempting to enable autosign for all nodes, as it cuts down on the manual steps required to bootstrap a new node (or indeed to move it to a new puppet master).

Typically this would be done with a *.example.com or even * in the autosign.conf file.

This however can be very dangerous as it can enable a node to masquerade as another node, and get the configuration intended for that node. The reason for this is that the node chooses the certificate common name ('CN' – usually its fqdn, but this is fully configurable), and the puppet master then uses this CN to look up the node definition to serve. The certificate itself is stored, so two nodes could not connect with the same CN (eg alice.example.com), but this is not the problem.

The problem lies in the fact that the puppet master does not make a 1-1 mapping between a node and the first certificate it saw for it, and hence multiple certificates can map to the same node, for example:

– alice.example.com connects, gets node alice { } definition.

– bob.example.com connects with CN alice.bob.example.com, and also matches node alice { } definition.

Without autosigning, it would be apparent that bob was trying to get alice's configuration – as the puppet cert process lists the full fqdn/CN presented. With autosign turned on, bob silently retrieves alice's configuration.

Depending on your environment, this may not present a significant risk. It essentially boils down to the question 'Do I trust everything that can connect to my puppet master?'.

If you do still choose to have a permanent, or semi-permanent, permissive autosign.conf, please consider doing the following:

– Firewall your puppet master – restrict port tcp/8140 to only networks that you trust.

– Create puppet masters for each 'trust zone', and only include the trusted nodes in that Puppet masters manifest.

– Never use a full wildcard such as *.

Puppet Master Interview Questions

25. What happens if I am on Puppet 2.6x or earlier?

Nothing changes for you. Puppet 2.6.x remains licensed as GPLv2. The license change is not retroactive.

26. Does this change affect all the components of Puppet?

As part of this change, we're also changing the license of the Facter system inventory tool to Apache. This change will take effect with Facter version 1.6.0, and earlier versions of Facter will remain licensed under the GPLv2 license. This change will bring the licensing of Puppet's two key components into alignment.

Our other major product, MCollective, is already licensed under the Apache 2.0 license.

27. What does this mean if I or my company have or want to contribute code to Puppet?

As part of this license change, Puppet Labs has approached every existing contributor to the project and asked them to sign a Contributor License Agreement or CLA.

Signing this CLA for yourself or your company provides both you and Puppet Labs with additional legal protections, and confirms:

1. That you own and are entitled to the code you are contributing to Puppet

2. That you are willing to have it used in distributions

This gives assurance that the origins and ownership of the code cannot be disputed in the event of any legal challenge.

28. What if I haven't signed a CLA?

If you haven't signed a CLA, then we can't yet accept your code contribution into Puppet or Facter. Signing a CLA is very easy: simply log into your GitHub account and go to our CLA page to sign the agreement.

We've worked hard to try find to everyone who has contributed code to Puppet, but if you have questions or concerns about a previous contribution you've made to Puppet and you don't believed you've signed a CLA, please sign a CLA or contact us for further information.

29. Does signing a CLA change who owns Puppet?

The change in license and the requirement for a CLA doesn't change who owns the code. This is a pure license agreement and NOT a Copyright assignment. If you sign a CLA, you maintain full copyright to your own code and are merely providing a license to Puppet Labs to use your code.

All other code remains the copyright of Puppet Labs.

30. Which versions of Ruby does Puppet support?

Puppet requires an MRI Ruby interpreter. Certain versions of Ruby are tested more thoroughly with Puppet than others, and some versions are not tested at all. Run ruby –version to check the version of Ruby on your system.

Starting with Puppet 4, puppet-agent packages do not rely on the OS's Ruby version, as it bundles its own Ruby environment. You can install puppet-agent alongside any version of Ruby or on systems without Ruby installed. Likewise Puppet Enterprise does not rely on the OS's Ruby version, as it bundles its own Ruby environment. You can install PE alongside any version of Ruby or on systems without Ruby installed. The Windows installers provided by Puppet Labs don't rely on the OS's Ruby version, and can be installed alongside any version of Ruby or on systems without Ruby installed.

http://www.ezdevinfo.com/view/puppet/7228.html

How Can I Pre-Sign Puppet Certificates?

Puppet Requires Certificates Between The Client (Puppet) Being Managed And The Server (Puppetmaster). You Can Run Manually On The Client And Then Go Onto The Server To Sign The Certificate, But How Do You Automate This Process For Clusters / Cloud Machines?

How Can The Little Guys Effectively Learn And Use Puppet?

Six Months Ago, In Our Not-For-Profit Project We Decided To Start Migrating Our System Management To A Puppet-Controlled Environment Because We Are Expecting Our Number Of Servers To Grow Substantially Between Now And A Year From Now.

Since The Decision Has Been Made Our IT Guys Have Become A Bit Too Annoyed A Bit Too Often. Their Biggest Objections Are:

- "We're Not Programmers, We're Sysadmins";
- Modules Are Available Online But Many Differ From One Another; Wheels Are Being Reinvented Too Often, How Do You Decide Which One Fits The Bill;
- Code In Our Repo Is Not Transparent Enough, To Find How Something Works They Have To Recurse Through Manifests And Modules They Might Have Even Written Themselves A While Ago;
- One New Daemon Requires Writing A New Module, Conventions Have To Be Similar To Other Modules, A Difficult Process;
- "Let's Just Run It And See How It Works"
- Tons Of Hardly Known 'Extensions' In Community Modules: 'Trocla', 'Augeas', 'Hiera'... How Can Our Sysadmins Keep Track?

I Can See Why A Large Organisation Would Dispatch Their Sysadmins To Puppet Courses To Become Puppet Masters. But How Would Smaller Players Get To Learn Puppet To A Professional Level If They Do Not Go To Courses And Basically Learn It Via Their Browser And Editor?

Puppet Security And Network Topologies

Background:

I Am Finally Setting Aside Some Time To Join The 21st Century And Look At Puppet.

As It Stands Today We Version Control All Server Configurations In A Repository That Is Held Internally At The Office. When An Update Needs Making, The Changes Are Checked Back Into The Repos And Manually Pushed Out To The Machine In Question. This Usually Means SFTP'ing To The Remote Machine And Then Moving Files Into Place, With The Relevant Permissions, From A Shell.

So I Am Hopeful That Puppet Is Going To Be An Simple Yet Amazing Extension To What We Already Have.

Now I Consider The Process That We Currently Have To Be Reasonably Secure. On The Assumption That Our Internal Network Will Always Be Relatively More Secure Than The Public Networks In Our Datacentres.

- The Process Is Always One Way. Changes Traverse From A Secure Environment To Insecure And Never The Other Way Round.
- The Master Store Is In The Safest Possible Place. The Risk Of Compromise, Either By Stealing Configurations Or Sending Out Malicious Modifications, Is Greatly Reduced.

Question:

From What I Understand Of The Puppet Server/Client Model Is That The Clients Poll And Pull Updates Down Directly From The Server. The Traffic Is SSL Wrapped So Cannot Be Intercepted Or Spoofed. But It Differs From What We Currently Do Because The Puppet Server[S] Would Need To Be Hosted In A Public Location. Either Centrally, Or One For Each Datacentre Site That We Maintain.

So I Am Wondering:

- Am I Being Unnecessarily Paranoid About The Change From Push To Pull?
- Am I Being Unnecessarily Paranoid About Centrally Storing All Of That Information On A Public Network?
- How Are Others Maintaining Multiple Networks - Separate Server For Each Site?

Update 30/07/09:

I Guess That One Of My Other *Big* Concerns Is Placing So Must Trust In A Single Machine. The Puppetmaster(S) Would Be Firewalled, Secured And Such. But Even So Any Public Machine With Listening Services Has An Attack Surface Of A Certain Size.

Presumably If The Master Has Permission To Update Any File On Any One Of The Puppet Clients, Then It's Compromise Would Ultimately Result In The Compromise Of All It's Clients. The "Kings To The Kingdom" So To Speak.

- Is That Hypothesis Correct?
- Is There Any Way That It Can Be Mitigated?

Puppet: Node Name Seems Dependent On Reverse Dns?

I Seem To Be Running Into A Little Bit Of A Problem Understanding How To Get This To Work. I Have A New Server I'm Building Sitting Behind The Office NAT At Work, Its Reverse Dns Maps To Office.Mydomain.Com, But I Want The Machine To Be Ns2.Mydomain.Com For The Sake Of Puppet.

Nodes.Pp Snippet:

```
Node 'Ns2.Mydomain.Com' Inherits Basenode {

Info('Ns2.Mydomain.Com')

}


Node 'Office.Mydomain.Com' Inherits Basenode {

Info('Office.Mydomain.Com')

}
```

And My 'Puppet.Conf' On The Client:

```
[Main]

#Was Node_name=Ns2.Mydomain.Com

#Was Fqdn=Ns2.Mydomain.Com

Certname=Ns2.Mydomain.Com

Node_name=Cert
```

My Syslog On The Server Reports:

```
Sep 16 22:59:12 Support Puppetmasterd[2800]: Host Is Missing Hostname And/Or Domain: Office.Mydomain.Com

Sep 16 22:59:12 Support Puppetmasterd[2800]: (Scope(Node[Office.Mydomain.Com])) Office.Mydomain.Com

Sep 16 22:59:12 Support Puppetmasterd[2800]: Compiled Catalog For Office.Mydomain.Com In 0.03 Seconds

Sep 16 22:59:12 Support Puppetmasterd[2800]: Caching Catalog For Ns2.Mydomain.Com
```

How Can I Make It Grab The Config For Ns2.Mydomain.Com Without Doing Something Like This:

```
Node 'Ns2.Mydomain.Com' Inherits Basenode {

Info('Ns2.Mydomain.Com')

}


Node 'Office.Mydomain.Com' Inherits 'Ns2.Mydomain.Com' {

Info('Office.Mydomain.Com')

}
```

UPDATE: This Problem Seems To Be Causing Other Issues As Well. For Instance If I Info("$Fqdn") While The Machine Is Sitting Behind Office.Mydomain.Com The Fqdn Fact Is Empty, As Well As The $Operatingsystem. Its Almost Like The Facts Aren't Being Discovered Properly. Is There Perhaps A NAT Issue? Are There Any Suggestions For Tracking Down This Cause Of This Problem?

How To Update A Package Using Puppet And A .Deb File

I Am Trying To Figure Out The Proper Way To Update/Upgrade A Deb Package Using Puppet From A Local Source Deb File. My Current Config Looks Like This...

```
Class Adobe-Air-2-0-4 {


File { "/Opt/Air-Debs":

Ensure => Directory

}


File { "/Opt/Air-Debs/Adobeair-2.0.4.Deb":

Owner   => Root,

Group   => Root,

Mode    => 644,

Ensure  => Present,

Source  => "Puppet://Puppet/Adobe-Air-2-0-4/Adobeair-2.0.4.Deb"

}


Package { "Adobeair":

Provider => Dpkg,

Ensure => Installed,

Source => "/Opt/Air-Debs/Adobeair-2.0.4.Deb"

}


}
```

I First Copy The Deb File Down To The Client Machine And Then Use 'Package' With The Provider Set To 'Dpkg'. This Works And I Get The Correct Version Installed.

My Question Is What Is The Proper Way To Update This Package In The Future. Can I Simply Change Out The Source File And Puppet Will Know That It's A Different Version And Update This Package? How Does Puppet Determine What Version Of A Package It Has Installed Versus The Version Of The Source Deb File?

I Am Pretty New To Puppet, So If You Have An Suggestions For Improvements To My Existing Config They Are Very Much Appreciated.

Are Configuration Management Tools (Puppet, Chef) Capable Of Keeping Installed Packages Up To Date?

This Is Probably A Simple Question For Those Of You Already Running Configuration Management Tools. Are Configuration Management Tools Such As Puppet Or Chef The Right Approach For Keeping Installed Packages Up To Date?

Suppose I Run A Number Of Servers, Mostly Based On Debian And Ubuntu. Do Configuration Management Tools Make It Easier To Update Packages Installed From The Repositories When Security Updates Or Bug Fixes Come Along?

I Currently Run "Unattended Upgrades" To Let The Systems Automatically Install Security Updates, But I Still Have To Connect To The Servers And Run Aptitude Update && Aptitude Safe-UpgradeEvery So Often. Naturally This Gets Boring, Tedious And Error-Prone The More Servers There Are.

Are Tools Such As Puppet Or Chef The Right Approach To Keeping Installed Packages Up To Date? Do Any Of You Use These Tools To Avoid Manually Running Aptitude Or An Equivalent On 15 Servers? I Am Quite Certain The Answer To These Questions Is "Yes, Of Course!"

But Where Can I Find More Information About This Particular Use Case? I Have Not Yet Had The Time To Study Puppet Or Chef In-Depth, And The Example Cookbooks Or Classes Only Show More Or Less Trivial Examples Of Installing One Particular Package, Such As Ssh. Do You Have Any Resources To Recommend, Other Than The Official Documentation (I Am, Of Course, Going To Study The Docs Once I Know Which, If Any, Of The Tools Are Right For Me).

How Can A Linux Administrator Improve Their Shell Scripting And Automation Skills?

In My Organization, I Work With A Group Of NOC Staff, Budding Junior Engineers And A Handful Of Senior Engineers; All With A Focus On Linux. One Interesting Step In The Way The Company Grows Talent Is That There's A Path From The NOC To The Senior Engineering Ranks. Viewing The Talent Pool As A Relative Newcomer, I See That There's A Split In The Skill Sets That Tends To Grow Over Time...

- There Are Engineers Who Know One Or Several Particular Technologies Well And Are Constantly Immersed... E.G. MySQL, Firewalls, SAN Storage, Load Balancers...
- There Are Others Who Are Generalists And Can Navigate Multiple Technologies.
- All Learn Enough Linux (Commands, Processes) To Do What They Need And Use On A Daily Basis.

A Differentiating Factor Between Some Of The Staff Is How Well They Embrace Scripting, Automation And Configuration Management Methodologies. For Instance, We Have Two Engineers Who Do The Bulk Of Amazon AWS CloudFormation Work, And Another Who Handles Most Of The Puppet Infrastructure. Perhaps A Quarter Of The Engineers Are Adept At BASH Shell Scripting.

Looking At This In The Context Of The *Incredibly* High Demand For DevOps Skills In The Job Market, I'm Curious How Other Organizations Foster The Development Of These Skills And Grow Their Internal Talent. Scripting Doesn't Seem Like A Particularly-Teachable Concept.

- How Does A Sysadmin Improve Their Shell Scripting?
- Is There Still A Place For Engineers Who Do Not/Cannot Keep Up In The DevOps Paradigm?
- Are We Simply To Assume That Some People Will Be Left Behind As These Technologies Evolve? Is That Okay?

Adding A Yum Repo To Puppet Before Doing Anything Else

Is There A Way To Force Puppet To Do Certain Things First? For Instance, I Need It To Install An RPM On All Servers To Add A Yum Repository (IUS Community) Before I Install Any Of The Packages.

Do Chef And Puppet Cost Money?

I Intend To Use Chef Or Puppet To Do Administration (I'm Thinking More Of Chef As It's Younger And I Get A Better Feeling About It).

In Both Home Pages I Saw There Is An "Enterprise Edition" That Costs Money And I Don't Intend To Buy Anything. What Would I Miss In Chef / Puppet If I Don't Buy Them?

What Does Chef Offer That Costs Money Exactly?

What Does Puppet Offer That Costs Money Exactly?

It Was Not So Clear To Me From Their Web Site, As It's Kind Of Obscure.

What Advantages/Features Does Puppet Or Chef Offer Over Salt (Or Vice Versa)? [Closed]

I Am Looking At Rolling Out A New Configuration Management Tool To Replace Our Home-Grown Solution. The Defacto Standards Are Chef And Puppet, Both Of Which Are Ruby-Centric (Though Can Be Used To Deploy Non-Ruby Environment, Obviously). The Vast Majority Of Our Development Is Done In Python And Our In-House Deployment Tools Make Heavy Use Of Fabric. Therefore I Am Learning Towards Salt Since It Too Is Python, Even Though It Is Not As Mature As Chef Or Puppet. But Since I'm Not Familiar Enough With The Options, I'm Finding It Difficult To Compare Apples-To-Apples.

Other Than The Smaller Community, Would I Be Giving Up Anything Signifcant By Using Salt Rather Than Puppet/Chef?

Update

It's Been Six Months Since I Posted This Question. And Despite It Being Closed, It's Been Viewed Over 1,000 Times So I Thought I'd Comment On My Experiences.

I Eventually Decided On Puppet Since It Had A Bigger Community. However, It Was An Immensely Frustrating Experience, Mainly Due To The Convoluted Puppet Configuration Syntax. Since I Now Had A Frame Of Reference To Compare The Two, I Recently Took Another Look At Salt--I'm Not Going Back. It Is Very, Very Cool. The Things I Like Best:

- Seamless Integration Of Both Push And Pull Configuration Models. Puppet Uses A Pull Model (Node Periodically Polls Server For Updates) And Has A Sister Component Called Marionette For Pushing Changes. Both Are Important To Me And I Prefer How Salt Works. Salt Also Executes Much Faster When You Have A Lot Of Nodes.
- Configuration Syntax Uses YAML, Which Is Just A Simple Text Format That Uses Indentation And Bullet Points. You Can Also Choose To Use Other Configuration Formats Via Template. This Makes Salt About 10x Easier To Learn And Maintain, In My Experience.
- Python-Based. This Was The Biggest Reason I Started Looking At Salt In The First Place. It Ended Up Being One Of The More Minor Reasons I Stayed. But If You're A Python Shop Like Us, It Makes It Easier To Develop Salt Plugins.

Automate Dpkg-Reconfigure Tzdata

I'm Using Puppet To Admin A Cluster Of Debian Servers. I Need To Change The Timezone Of Each Machine On The Cluster. The Proper Debian Way To Do This Is To Use Dpkg-Reconfigure Tzdata. But I Can Only Seem To Change It If I Use The Dialog. Is There Some Way To Automate This From The Shell So I Can Just Write An Exec To Make This Easy?

If Not, I Think The Next Best Way Would Probably Be To Have Puppet Distribute /Etc/Timezone And /Etc/Localtime With The Correct Data Across The Cluster.

Any Input Appreciated!

Puppet Vs Chef, Pro And Contra From Users And Use Cases [Closed]

I Already Googled And Read The "To-Puppet-Or-To-Chef-That-Is-The-Question" Article.

I'm Interested In Use Cases, Real World Implementations In Which People Had Choosen One Or The Other On Real Problems Bases.

I'm Particularly Interested In ***Integration With Cobbler*** Issues (I Know Puppet Is Much A Standard Approach In This Direction); As Anybody Any Experience In ***Cobbler-Chef Integration*** ?

Thanks In Advance

What Should NOT Be Managed By Puppet?

I'm Learning My Way Through Configuration Management In General And Using Puppet To Implement It In Particular, And I'm Wondering What Aspects Of A System, If Any, Should *Not* Be Managed With Puppet?

As An Example We Usually Take For Granted That Hostnames Are Already Set Up Before Lending The System To Puppet's Management. Basic IP Connectivity, At Least On The Network Used To Reach The Puppetmaster, Has To Be Working. Using Puppet To Automatically Create Dns Zone Files Is Tempting, But DNS Reverse Pointers Ought To Be Already In Place Before Starting Up The Thing Or Certificates Are Going To Be Funny.

So Should I Leave Out IP Configuration From Puppet? Or Should I Set It Up Prior To Starting Puppet For The First Time But Manage Ip Addresses With Puppet Nonetheless? What About Systems With Multiple IPs (Eg. For WAN, LAN And SAN)?

What About IPMI? You Can Configure Most, If Not All, Of It With Ipmitool, Saving You From Getting Console Access (Physical, Serial-Over-Lan, Remote KVM, Whatever) So It Could Be Automated With Puppet. But Re-Checking Its State At Every Puppet Agent Run Doesn't Sound Cool To Me, And Basic Lights Out Access To The System Is Something I'd Like To Have Before Doing Anything Else.

Another Whole Story Is About Installing Updates. I'm Not Going In This Specific Point, There Are Already Many Questions On SF And Many Different Philosophies Between Different Sysadmins. Myself, I Decided To Not Let Puppet Update Things (Eg. Only Ensure => Installed) And Do Updates Manually As We Are Already Used To, Leaving The Automation Of This Task To A Later Day When We Are More Confident With Puppet (Eg. By Adding MCollective To The Mix).

Those Were Just A Couple Of Examples I Got Right Now On My Mind. Is There Any Aspect Of The System That Should Be Left Out Of Reach From Puppet? Or, Said Another Way, Where Is The Line Between What Should Be Set Up At Provisioning Time And "Statically" Configured In The System, And What Is Handled Through Centralized Configuration Management?

Why Is It So Difficult To Upgrade Between Major Versions Of Red Hat And CentOS?

"Can We Upgrade Our Existing Production EL5 Servers To EL6?"

A Simple-Sounding Request From Two Customers With *Completely* Different Environments Prompted My Usual Best-Practices Answer Of "Yes, But It Will Require A Coordinated Rebuild Of All Of Your Systems"...

Both Clients Feel That A Complete Rebuild Of Their Systems Is An Unacceptable Option For Downtime And Resource Reasons... When Asked Why It Was Necessary To Fully Reinstall The Systems, I Didn't Have A Good Answer Beyond, "That's The Way It Is..."

I'm Not Trying To Elicit Responses About Configuration Management ("Puppetize *Everything*" Doesn't Always Apply) Or How The Clients Should Have Planned Better. This Is A Real-World Example Of Environments That Have Grown And Thrived In A Production Capacity, But Don't See A Clean Path To Move To The Next Version Of Their OS.

Environment A:

Non-Profit Organization With **40 X Red Hat Enterprise Linux 5.4 And 5.5** Web, Database Servers And Mail Servers, Running A Java Web Application Stack, Software Load Balancers And Postgres Databases. All Systems Are Virtualized On Two VMWare VSphere Clusters In Different Locations, Each With HA, DRS, Etc.

Environment B:

High-Frequency Financial Trading Firm With **200 X CentOS 5.X** Systems In Multiple Co-Location Facilities Running Production Trading Operations, Supporting In-House Development And Back-Office Functions. The Trading Servers Are Running On Bare-Metal Commodity Server Hardware. They Have Numerous Sysctl.Conf, Rtctl, Interrupt Binding And Driver Tweaks In Place To Lower Messaging Latency. Some Have Custom And/Or Realtime Kernels. The Developer Workstations Are Also Running A Similar Version(S) Of CentOS.

In Both Cases, The Environments Are Running Well As-Is. The Desire To Upgrade Comes From A Need For A Newer Application Or Feature Available In EL6.

- For The Non-Profit Firm, It's Tied To Apache, The Kernel And Some Things That Will Make The Developers Happy.
- In The Trading Firm, It's About Some Enhancements In The Kernel, Networking Stack And GLIBC, Which Will Make The Developers Happy.

Both Are Things That Can't Be Easily Packaged Or Updated Without Drastically Altering The Operating System.

As A Systems Engineer, I Appreciate That Red Hat Recommends Full Rebuilds When Moving Between Major Version Releases. A Clean Start Forces You To Refactor And Pay Attention To Configs Along The Way.

Being Sensitive To Business Needs Of Clients, I Wonder Why This Needs To Be Such An Onerous Task. The RPM Packaging System Is More Than Capable Of Handling In-Place Upgrades, But It's The Little Details That Get You: /Boot Requiring More Space, New Default Filesystems, RPM Possibly Breaking Mid-Upgrade, Deprecated And Defunct Packages...

What's The Answer Here? Other Distributions (.Deb-Based, Arch And Gentoo) Seem To Have This Ability Or A Better Path. Let's Say We Find The Downtime To Accomplish This Task The *Right* Way:

- What Should These Clients Do To Avoid The Same Problem When EL7 Is Released And Stabilizes?
- Or Is This A Case Where People Need To Resign Themselves To Full Rebuilds Every Few Years?
- This Seems To Have Gotten Worse As Enterprise Linux Has Evolved... Or Am I Just Imagining That?
- Has This Dissuaded Anyone From Using Red Hat And Derivative Operating Systems?

I Suppose There's The Configuration Management Angle, But Most Puppet Installations I See Do Not Translate Well Into Environments With Highly-Customized Application Servers (**Environment B** Could Have A Single Server Whose Ifconfig Output Looks Like This). I'd Be Interesting In Hearing Suggestions On How Configuration Management Can Be Used To Help Organizations Get Across The RHEL Major Version Bump, Though.

Why Use Chef/Puppet Over Shell Scripts?

New To Puppet And Chef Tools. Seems Like The Job That They Are Doing Can Be Done With Shell Scripting. Maybe It Was Done In Shell Scripts Until These Came Along.

I Would Agree They Are More Readable. But, Are There Any Other Advantages Over Shell Scripts Besides Just Being Readable?

Options For Multisite High Availability With Puppet

I Maintain Two Datacenters, And As More Of Our Important Infrastructure Starts To Get Controlled Via Puppet, It Is Important The The Puppet Master Work At The Second Site Should Our Primary Site Fail.

Even Better Would Be To Have A Sort Of Active / Active Setup So The Servers At The Second Site Are Not Polling Over The WAN.

Are There Any Standard Methods Of Multi-Site Puppet High Availability?

NFS With Encrypted Ubuntu Home Directory

I Am Having Trouble Getting NFS Setup On With Vagrant:

On My Local Machine I Have Installed NFS:

Apt-Get Install Nfs-Common Nfs-Kernel-Server

And In My Vagrantfile Set It To Be Used:

Config.Vm.Share_folder("V-Root", "/Vagrant", ".", :Nfs => True)

On Vagrant Up I Get:

Exportfs: /Home/<User>/Path/To/Dir Does Not Support NFS Export

Mounting NFS Shared Folders Failed. This Is Most Often Caused By The NFS

Client Software Not Being Installed On The Guest Machine. Please Verify

That The NFS Client Software Is Properly Installed, And Consult Any Resources

Specific To The Linux Distro You're Using For More Information On How To

Do This.

Am I Missing A Step Or Two Here?

I'm Aware Of Some Issues With Ubuntus Encrypted Home Folders And NFS But I Understand This Is Only Meant To Be A Problem Before Boot.

[Update] My /Etc/Exports File Looks Like This:

VAGRANT-BEGIN: 5af3e5d6-B086-416d-8eab-987275445634

/Home/<User>/Path/To/Dir 192.168.33.11(Rw,No_subtree_check,All_squash,

Anonuid=1000,Anongid=1000,Fsid$

VAGRANT-END: 5af3e5d6-B086-416d-8eab-987275445634

Pros And Cons Of A Decentralized Puppet Architecture

We Have Around 300 RHEL Servers That Are Currently Connecting To A Puppetmaster Server. However, We Have Noticed Some Performance Bottlenecks And It Is The Point Of Failure In Our System. I Am Fairly New To Puppet In General And I Am Considering Creating A Decentralized Puppet Architecture Instead Of Having Puppet Clients Connect To The Puppetmaster Server. Aside From What I Would Suspect To Happen Such As Performance Gain And Lack Of Signing And Exchanging SSL Certs For New Machines, What Are Other Pros And Cons To Setting Up A Decentralized Architecture?

Fixing Services That Have Been Disabled In /Etc/Default/ With Puppet?

I'm Using Puppet To (Theoretically) Get Npcd To Start Upon Installation, However On Ubuntu, That Service Comes Installed With The Default Setting In /Etc/Default/Npcd Of RUN="No":

$ Cat /Etc/Default/Npcd

Default Settings For The NPCD Init Script.

Should NPCD Be Started? ("Yes" To Enable)

RUN="No"

Additional Options That Are Passed To The Daemon.

DAEMON_OPTS="-D -F /Etc/Pnp4nagios/Npcd.Cfg"

I Would Think That This Block Of Puppet Config Would Take Care Of Things:

Service { "Npcd":

Enable => True,

Ensure => "Running",

Require => Package["Pnp4nagios"],

}

But Alas, It Doesn't, And Short Of Actually Rewriting The File In /Etc/Default, I'm Not Sure What To Do. Is There A Straightforward Way To Enable The Service That I'm Not Seeing?

For The Record, I'm Using Ubuntu 12.04.2 And Puppet Version 3.1.0.

Adding Lines To /Etc/Profile With Puppet?

I Use Puppet To Install A Current JDK And Tomcat.

Package {

["Openjdk-6-Jdk", "Openjdk-6-Doc", "Openjdk-6-Jre",

"Tomcat6", "Tomcat6-Admin", "Tomcat6-Common", "Tomcat6-Docs",

"Tomcat6-User"]:

Ensure => Present,

}

Now I'd Like To Add

JAVA_HOME="/Usr/Lib/Java"

Export JAVA_HOME

To /Etc/Profile, Just To Get This Out Of The Way. I Haven't Found A Straightforward Answer In The Docs, Yet. Is There A Recommended Way To Do This?

In General, How Do I Tell Puppet To Place This File There Or Modify That File? I'm Using Puppet For A Single Node (In Standalone Mode) Just To Try It Out And To Keep A Log Of The Server Setup.

Managing An Application Across Multiple Servers, Or PXE Vs CfEngine/Chef/Puppet

We Have An Application That Is Running On A Few (5 Or So And Will Grow) Boxes. The Hardware Is Identical In All The Machines, And Ideally The Software Would Be As Well. I Have Been Managing Them By Hand Up Until Now, And Don't Want To Anymore (Static Ip Addresses, Disabling All Necessary Services, Installing Required Packages...) . Can Anyone Balance The Pros And Cons Of The Following Options, Or Suggest Something More Intelligent?

1: Individually Install Centos On All The Boxes And Manage The Configs With Chef/Cfengine/Puppet. This Would Be Good, As I Have Wanted An Excuse To Learn To Use One Of Applications, But I Don't Know If This Is Actually The Best Solution.

2: Make One Box Perfect And Image It. Serve The Image Over PXE And Whenever I Want To Make Modifications, I Can Just Reboot The Boxes From A New Image. How Do Cluster Guys Normally Handle Things Like Having Mac Addresses In The /Etc/Sysconfig/Network-Scripts/Ifcfg* Files? We Use Infiniband As Well, And It Also Refuses To Start If The Hwaddr Is Wrong. Can These Be Correctly Generated At Boot?

I'm Leaning Towards The PXE Solution, But I Think Monitoring With Munin Or Nagios Will Be A Little More Complicated With This. Anyone Have Experience With This Type Of Problem?

All The Servers Have SSDs In Them And Are Fast And Powerful.

Thanks, Matt.

Configuration Management: Push Versus Pull Based Topology

The More Established Configuration Management (CM) Systems Like Puppet And Chef Use A Pull-Based Approach: Clients Poll A Centralized Master Periodically For Updates. Some Of Them Offer A *Masterless* Approach As Well (So, Push-Based), But State That It Is 'Not For Production' (Saltstack) Or 'Less Scalable' (Puppet). The Only System That I Know Of That Is Push-Based From The Start Is Runner-Up Ansible.

What Is The Specific Scalability Advantage Of A Pull Based System? Why Is It Supposedly Easier To Add More Pull-Masters Than Push-Agents?

For Example, Agiletesting.Blogspot.Nl Writes:

In A 'Pull' System, Clients Contact The Server Independently Of Each Other, So The System As A Whole Is More Scalable Than A 'Push' System

On The Other Hand, Rackspace Demonstrates That They Can Handle 15K Systems With A Push-Based Model.

Infastructures.Org Writes:

We Swear By A Pull Methodology For Maintaining Infrastructures, Using A Tool Like SUP, CVSup, An Rsync Server, Or Cfengine. Rather Than Push Changes Out To Clients, Each Individual Client Machine Needs To Be Responsible For Polling The Gold Server At Boot, And Periodically Afterwards, To Maintain Its Own Rev Level. Before Adopting This Viewpoint, We Developed Extensive Push-Based Scripts Based On Ssh, Rsh, Rcp, And Rdist. The Problem We Found With The R-Commands (Or Ssh) Was This: When You Run An R-Command Based Script To Push A Change Out To Your Target Machines, Odds Are That If You Have More Than 30 Target Hosts One Of Them Will Be Down At Any Given Time. Maintaining The List Of Commissioned Machines Becomes A Nightmare. In The Course Of Writing Code To Correct For This, You Will End Up With Elaborate Wrapper Code To Deal With: Timeouts From Dead Hosts; Logging And Retrying Dead Hosts; Forking And Running Parallel Jobs To Try To Hit Many Hosts In A Reasonable Amount Of Time; And Finally Detecting And Preventing The Case Of Using Up All Available TCP Sockets On The Source Machine With All Of The Outbound Rsh Sessions. Then You Still Have The Problem Of Getting Whatever You Just Did Into The Install Images For All New Hosts To Be Installed In The Future, As Well As Repeating It For Any Hosts That Die And Have To Be Rebuilt Tomorrow. After The Trouble We Went Through To Implement R-Command Based Replication, We Found It's Just Not Worth It. We Don't Plan On Managing An Infrastructure With R-Commands Again, Or With Any Other Push Mechanism For That Matter. They Don't Scale As Well As Pull-Based Methods.

Isn't That An Implementation Problem Instead Of An Architectural One? Why Is It Harder To Write A Threaded Push Client Than A Threaded Pull Server?

Puppet And Launchd Services?

We Have A Production Environment Configured With Puppet, And Want To Be Able To Set Up A Similar Environment On Our Development Machines: A Mix Of Red Hats, Ubuntus And OSX. As Might Be Expected, OSX Is The Odd Man Out Here, And Sadly, I'm Having A Lot Of Trouble With Getting This To Work.

My First Attempt Was Using Macports, Using The Following Declaration:

Package { 'Rabbitmq-Server':

Ensure => Installed,

Provider => Macports,

}

But This, Sadly, Generates The Following Error:

Error: /Stage[Main]/Rabbitmq/Package[Rabbitmq-Server]: Could Not Evaluate: Execution Of '/Opt/Local/Bin/Port -Q Installed Rab

Cut -C List [File ...]

Cut -F List [-S] [-D Delim] [File ...]

While Executing

"Exec Dscl -Q . -Read /Users/$Env(SUDO_USER) NFSHomeDirectory | Cut -D ' ' -F 2"

(Procedure "Mportinit" Line 95)

Invoked From Within

"Mportinit Ui_options Global_options Global_variations"

Next Up, I Figured I'd Give Homebrew A Try. There Is No Package Provider Available By Default, But Puppet-Homebrew Seemed Promising. Here, I Got Much Farther, And Actually Managed To Get The Install To Work.

Package { 'Rabbitmq':

Ensure => Installed,

Provider => Brew,

}

File { "Plist":

Path => "/Library/LaunchDaemons/Homebrew.Mxcl.Rabbitmq.Plist",

Source => "/Usr/Local/Opt/Rabbitmq/Homebrew.Mxcl.Rabbitmq.Plist",

Ensure => Present,

Owner => Root,

Group => Wheel,

Mode => 0644,

}

Service { "Homebrew.Mxcl.Rabbitmq":

Enable => True,

Ensure => Running,

Provider => "Launchd",

Require => [File["/Library/LaunchDaemons/Homebrew.Mxcl.Rabbitmq.Plist"]],

}

Here, I Don't Get Any Error. But RabbitMQ Doesn't Start Either (As It Does If I Do A Manual Load With Launchctl)

[... Snip ...]

Debug: Executing '/Bin/Launchctl List'

Debug: Executing '/Usr/Bin/Plutil -Convert Xml1 -O /Dev/Stdout

/Library/LaunchDaemons/Homebrew.Mxcl.Rabbitmq.Plist'

Debug: Executing '/Usr/Bin/Plutil -Convert Xml1 -O /Dev/Stdout

/Var/Db/Launchd.Db/Com.Apple.Launchd/Overrides.Plist'

Debug: /Schedule[Weekly]: Skipping Device Resources Because Running On A Host

Debug: /Schedule[Puppet]: Skipping Device Resources Because Running On A Host

Debug: Finishing Transaction 2248294820

Debug: Storing State

Debug: Stored State In 0.01 Seconds

Finished Catalog Run In 25.90 Seconds

What Am I Doing Wrong?

Edit: For The Record, We're Now Doing This With Vagrant VMs Instead On Our OSX Machines, But The Native Solution Would Still Be Preferred.

Puppet: Ensure A File Is Empty

I Would Like To Be Sure That The Motd File Is Empty. I Would Love To Do Like This:

File { "/Etc/Motd":

Ensure => Empty

}

This Obviously Does Not Work.

Is There A Simple Way To Ensure A File Is Empty Instead Using The "Source" Declaration And Store An Empty File In The File Repository?

What Are The Right Questions To Ask When Deciding Whether To Use Chef Or Puppet?

I Am About To Start A New Project Which Will, In Part, Require Deploying Many Identical Nodes Of Approximately Three Different Classes:

- *Data Nodes*, Which Will Run Sharded Instances Of MongoDB.
- *Application Nodes*, Which Will Run Instances Of A Ruby On Rails Application And An Older ASP.NET MVC Application.
- *Processing Nodes*, Which Will Run Jobs Requested By The Application Nodes.

All The Nodes Will Run On Instances Of Ubuntu 10.04, Though They Will Have Different Packages Installed.

I Have Some Familiarity With Chef From Previous Projects, Though I Don't Consider Myself An Expert. In An Effort To Do Due Diligence, I Have Been Investigating Alternative Possibilities. We Have A Number Of Folks In-House Who Are Long-Time Puppet Users, And They Have Encouraged Me To Take A Look.

I Am Having Trouble Evaluating Both Choices, Though. Chef And Puppet Share Many Of The Same Domain Terminology - - *Packages*, *Resources*, *Attributes*, And So On, And They Have A Common History That Stems From Taking Different Approaches To The Same Problem. So In Some Sense They Are Very Similar. But Much Of The Comparison Information I've Found, Like **This Article**, Is A Little Outdated.

If You Were Starting This Project Today, What Questions Would You Ask Yourself To Decide Whether You Should Use Chef Or Puppet For Configuration Management? (Note: I *Don't* Want Answer To The Question "Should I Use Chef Or Puppet?")

What's The Strengths And Weaknesses Of Existing Configuration Management Systems? [Closed]

I Was Looking Up Here For Some Comparisons Between CFEngine, Puppet, Chef, Bcfg2, AutomateIt And Whatever Other Configuration Management Systems Might Be Out There, And Was Very Surprised I Could Find Very Little Here On Server Fault. For Instance, I Only Knew Of The First Three Links Above -- The Other Two I Found On A Related Google Search.

So, I'm Not Interested In What People Think Is The Best One, Or Which They Like. I'd Like To Know The Following:

1. Configuration Management System's Name.
2. Why It Was Created (As Opposed To Using An Existing Solution).
3. Relative Strengths.
4. Relative Weaknesses.
5. License.
6. Link To Project And Examples.

Could Not Find Class, And Yet It Is There

When Doing A Puppet Agent Call From A New Image, I'm Getting A Err: Could Not Find Class Custommod Error. The Module Itself Is In /Etc/Puppet/Modules/Custommod Same As All Of The Other Modules We're Calling, But This One Is Obstinante.

[Site.Pp]

Node /Clunod-Wk\D+\.Sub\.Example\.Local/ {

Include Base

Include Curl

Include Custommod

Class{ "Custommod::Apps": Frontend => "False}

[...]

}

When The Puppetmaster Is Run With Debug Output, It Clearly Finding The Information For Base And Curl:

Debug: Importing '/Etc/Puppet/Modules/Base/Manifests/Init.Pp' In Environment Production

Debug: Automatically Imported Base From Base Into Production

Debug: Importing '/Etc/Puppet/Modules/Curl/Manifests/Init.Pp' In Environment Production

Debug: Automatically Imported Curl From Curl Into Production

Err: Could Not Find Class Custommod For Clunod-Wk0130.Sub.Example.Local At /Etc/Puppet/Manifests/Site.Pp:84 On Node Clunc

Line 84 Is Include Custommod

An Abbreviated Directory And File Structure:

/Etc/Puppet

|- Manifests

| |- Site.Pp

|

|- Modules

|- Base

| |- Manifests

| |- Init.Pp

|

|- Curl

| |- Manifests

| |- Init.Pp

|

|- Custommod

|- Files

| |- Apps

| |- [...]

|

|- Manifests

|- Init.Pp

|- Apps.Pp

I Did Check Spelling :}

The Content Of Init.Pp In The Custommod Directory Is Completely Unremarkable:

Class Custommod {

}

The Intent Is To Create An Empty Class For The Apps.Pp File, Which Is Where The Meat Is.

Class Custommod::Apps {

[Lots Of Stuff]

}

Only, It's Never Getting To The Apps File. If I Comment Out The Include Custommod, The Above Error Is Generated On The Class{ "Custommod::Apps": Frontend => "False} Line Instead.

What Am I Missing In My Hunt To Find Out How This Error Is Being Generated? I Need To Note That This Repo Works Just Fine If It Is Run Locally Via Puppet Apply.

Puppet: Node Name Seems Dependent On Reverse Dns?

I Seem To Be Running Into A Little Bit Of A Problem Understanding How To Get This To Work. I Have A New Server I'm Building Sitting Behind The Office NAT At Work, Its Reverse Dns Maps To Office.Mydomain.Com, But I Want The Machine To Be Ns2.Mydomain.Com For The Sake Of Puppet.

Nodes.Pp Snippet:

Node 'Ns2.Mydomain.Com' Inherits Basenode {

Info('Ns2.Mydomain.Com')

}

Node 'Office.Mydomain.Com' Inherits Basenode {

Info('Office.Mydomain.Com')

}

And My 'Puppet.Conf' On The Client:

[Main]

#Was Node_name=Ns2.Mydomain.Com

#Was Fqdn=Ns2.Mydomain.Com

Certname=Ns2.Mydomain.Com

Node_name=Cert

My Syslog On The Server Reports:

Sep 16 22:59:12 Support Puppetmasterd[2800]: Host Is Missing Hostname And/Or Domain: Office.Mydomain.Com

Sep 16 22:59:12 Support Puppetmasterd[2800]: (Scope(Node[Office.Mydomain.Com])) Office.Mydomain.Com

Sep 16 22:59:12 Support Puppetmasterd[2800]: Compiled Catalog For Office.Mydomain.Com In 0.03 Seconds

Sep 16 22:59:12 Support Puppetmasterd[2800]: Caching Catalog For Ns2.Mydomain.Com

How Can I Make It Grab The Config For Ns2.Mydomain.Com Without Doing Something Like This:

```
Node 'Ns2.Mydomain.Com' Inherits Basenode {

Info('Ns2.Mydomain.Com')

}


Node 'Office.Mydomain.Com' Inherits 'Ns2.Mydomain.Com' {

Info('Office.Mydomain.Com')

}
```

UPDATE: This Problem Seems To Be Causing Other Issues As Well. For Instance If I Info("$Fqdn") While The Machine Is Sitting Behind Office.Mydomain.Com The Fqdn Fact Is Empty, As Well As The $Operatingsystem. Its Almost Like The Facts Aren't Being Discovered Properly. Is There Perhaps A NAT Issue? Are There Any Suggestions For Tracking Down This Cause Of This Problem?

How To Update A Package Using Puppet And A .Deb File

I Am Trying To Figure Out The Proper Way To Update/Upgrade A Deb Package Using Puppet From A Local Source Deb File. My Current Config Looks Like This...

```
Class Adobe-Air-2-0-4 {


File { "/Opt/Air-Debs":

Ensure => Directory

}


File { "/Opt/Air-Debs/Adobeair-2.0.4.Deb":

Owner   => Root,

Group   => Root,

Mode    => 644,

Ensure  => Present,

Source  => "Puppet://Puppet/Adobe-Air-2-0-4/Adobeair-2.0.4.Deb"

}


Package { "Adobeair":

Provider => Dpkg,

Ensure => Installed,

Source => "/Opt/Air-Debs/Adobeair-2.0.4.Deb"

}
```

}

I First Copy The Deb File Down To The Client Machine And Then Use 'Package' With The Provider Set To 'Dpkg'. This Works And I Get The Correct Version Installed.

My Question Is What Is The Proper Way To Update This Package In The Future. Can I Simply Change Out The Source File And Puppet Will Know That It's A Different Version And Update This Package? How Does Puppet Determine What Version Of A Package It Has Installed Versus The Version Of The Source Deb File?

I Am Pretty New To Puppet, So If You Have An Suggestions For Improvements To My Existing Config They Are Very Much Appreciated.

Puppet Security And Network Topologies

Background:

I Am Finally Setting Aside Some Time To Join The 21st Century And Look At Puppet.

As It Stands Today We Version Control All Server Configurations In A Repository That Is Held Internally At The Office. When An Update Needs Making, The Changes Are Checked Back Into The Repos And Manually Pushed Out To The Machine In Question. This Usually Means SFTP'ing To The Remote Machine And Then Moving Files Into Place, With The Relevant Permissions, From A Shell.

So I Am Hopeful That Puppet Is Going To Be An Simple Yet Amazing Extension To What We Already Have.

Now I Consider The Process That We Currently Have To Be Reasonably Secure. On The Assumption That Our Internal Network Will Always Be Relatively More Secure Than The Public Networks In Our Datacentres.

- The Process Is Always One Way. Changes Traverse From A Secure Environment To Insecure And Never The Other Way Round.
- The Master Store Is In The Safest Possible Place. The Risk Of Compromise, Either By Stealing Configurations Or Sending Out Malicious Modifications, Is Greatly Reduced.

Question:

From What I Understand Of The Puppet Server/Client Model Is That The Clients Poll And Pull Updates Down Directly From The Server. The Traffic Is SSL Wrapped So Cannot Be Intercepted Or Spoofed. But It Differs From What We Currently Do Because The Puppet Server[S] Would Need To Be Hosted In A Public Location. Either Centrally, Or One For Each Datacentre Site That We Maintain.

So I Am Wondering:

- Am I Being Unnecessarily Paranoid About The Change From Push To Pull?
- Am I Being Unnecessarily Paranoid About Centrally Storing All Of That Information On A Public Network?
- How Are Others Maintaining Multiple Networks - Separate Server For Each Site?

Update 30/07/09:

I Guess That One Of My Other *Big* Concerns Is Placing So Must Trust In A Single Machine. The Puppetmaster(S) Would Be Firewalled, Secured And Such. But Even So Any Public Machine With Listening Services Has An Attack Surface Of A Certain Size.

Presumably If The Master Has Permission To Update Any File On Any One Of The Puppet Clients, Then It's Compromise Would Ultimately Result In The Compromise Of All It's Clients. The "Kings To The Kingdom" So To Speak.

- Is That Hypothesis Correct?
- Is There Any Way That It Can Be Mitigated?

http://www.learnitguide.net/2016/09/what-is-puppet-how-puppet-works.html

1. **What is Puppet?**

Puppet is a configuration management tool available as an open-source and Enterprise versions. It runs on many Unix-like systems as well as on Microsoft Windows.

Puppet is produced by Puppet Labs, founded by Luke Kanies in 2005. It is written in Ruby and released as free software under the GNU General Public License (GPL) until version 2.7.0 and the Apache License 2.0 after that.

Puppet is designed to manage the configuration of Unix-like and Microsoft Windows systems declaratively. The user describes system resources and their state, either using Puppet's declarative language or a Ruby DSL (domain-specific language).

2. Why do we use Puppet?

We use Puppet, because puppet is a configuration management tool which is more powerful that helps system administrators to automate the provisioning, configuration, and management of a server infrastructure. Puppet enables system administrators and Dev Ops to work faster and smarter.

3. How Puppet works?

This information is stored in files called "Puppet manifests". Puppet discovers the system information via a utility called Facter, and compiles the Puppet manifests into a system-specific catalog containing resources and resource dependency, which are applied against the target systems. Any actions taken by Puppet are then reported.

4. Understanding the Puppet Architecture and Puppet components.

Puppet master

Puppet master is a service runs on the main server which used to manage the entire clients to deploy, configure and maintains the infrastructures.

Puppet agent

Puppet agent is a service which runs on the client which sends the request the catalog to the puppet master and applies it by checking each resource the catalog describes. If it finds any resources that are not in their desired state, it makes any changes necessary to correct them. After applying the catalog, the agent submits a report to the Puppet master.

Catalog

A catalog is a document that describes the desired system state for one specific server. It lists all of the resources that need to be managed, as well as any dependencies between those resources.

Manifests

Manifests are files with extension ".pp", where we declare all resources to be checked or to be changed. Resources may be files, packages, services and so on.

Resources types

Resource Types are,

a. A type (package, service, file, user, mount, exec)

b. A title (how the resources types are called and referred)

Sample syntax:

```
type { 'title':
 argument  => value,
 other_arg => value,
}
```

Simple samples of resources

Verify the OpenSSH package

```
package { 'openssh':
 ensure => present,
}
```

Create a /etc/motd file

```
file { 'motd':
 path => '/etc/motd',
}
```

Start a httpd service

```
service { 'httpd':
 ensure => running,
 enable => true,
}
```

If you are looking for Resource Types reference, use the below command.

```
puppet describe file
```

For the full list of available descriptions:

```
puppet describe --list
```

Classes

Classes are containers or groups of different resources.

Example of a class definition:

```
class mysql (
 root_password => 'default_value',
 port        => '3306',
) {
 package { 'mysql-server':
  ensure => present,
 }
 service { 'mysql':
  ensure   => running,
 }
 [...]
}
```

Note that when we define a class we just describe what it does and what parameters it has, we don't actually add it and its resources to the catalog.

5. How the puppet connections are getting established between puppet master and puppet agent nodes?

Puppet agent nodes and Puppet masters communicate via HTTPS with client-verification. The Puppet master provides an HTTP interface, with various endpoints available. When requesting or submitting anything to the master, the agent makes an HTTPS request to one of those endpoints.

In the previous articles, we have explained about the below topics, Refer that also which are very essential to understand the puppet from the beginning.,

How to install Puppet Master and Puppet Agent on Linux?

Understand the Basics of Puppet Manifests

Understand the Basics of Puppet Modules

---

http://serverfault.com/questions/504070/why-use-chef-puppet-over-shell-scripts

---

http://www.elsotanillo.net/2015/09/devops-job-interviews-with-old-fashioned-check-list-questions/

What is Puppet?

- What is puppet manifest?
- What is manifest ordering and its importance?
- What is puppet module?
- Why Puppet matters to Devops? (Advantage of Puppet over other devops tools)
- What is puppet catalog?
- Can you describe the puppet module layouts / structure?
- What are agent nodes?
- What is EPP templates?
- What is the use of Puppet DB?
- What is the use of filebucket in puppet?
- How do you perform dry run? (no-op / noop)
- What is virtual resource in puppet?
- How can you realize a virtual resource?
- Is puppet resource idempotent?
- What is the purpose of Hiera tools?
- Which node is called masterless node?
- How can you manage nodes using node manager?

Mr Tic *November 22, 2017 at 8:49 PM*

Wonderful article! We are linking to this great post on our site.Keep up the good writing. Visit: Please Read More: Download Ebook: Ultimate Guide To Job Interview Questions Answers:

REPLY

soumya Teja *March 21, 2018 at 4:59 AM*

The information on this blog is very useful and very interesting. If anyone needs to know about these just check at **Devops Online Course**

REPLY

Unknown *March 27, 2018 at 5:22 AM*

• Nice and good article. It is very useful for me to learn and understand easily. Thanks for sharing your valuable information and time. Please keep updating. **Power Bi Online Training Bangalore**

REPLY

Enter your comment...

Popular posts from this blog

Jenkins Interview Questions

By Devops - March 27, 2017

Image I wrote a blog post to help prepare people for their DevOps interview. These questions are all gathered from different blogs and sites. I have mentioned the site name, from where I have copied. Thanks to all those authors and blog writers. ...

READ MORE

Docker Interview Questions

By Devops - March 12, 2017

I wrote a blog post to help prepare people for their DevOps interview. These questions are all gathered from different blogs and sites. I have mentioned the site name, from where I have copied. Thanks to all those authors and blog writers. ...

READ MORE

Important

ArchiveList

Search This Blog

Search this blog

DEVOPS

VISIT PROFILE

Archive

Report Abuse