PUPPET

Puppet Tutorial

Puppet Tutorial is the second blog of Puppet blog series. I hope you have read my previous blog on "*What is Puppet*" that explains Configuration Management and why it is important with the help of use-cases.

In this Puppet Tutorial following topics will be covered:

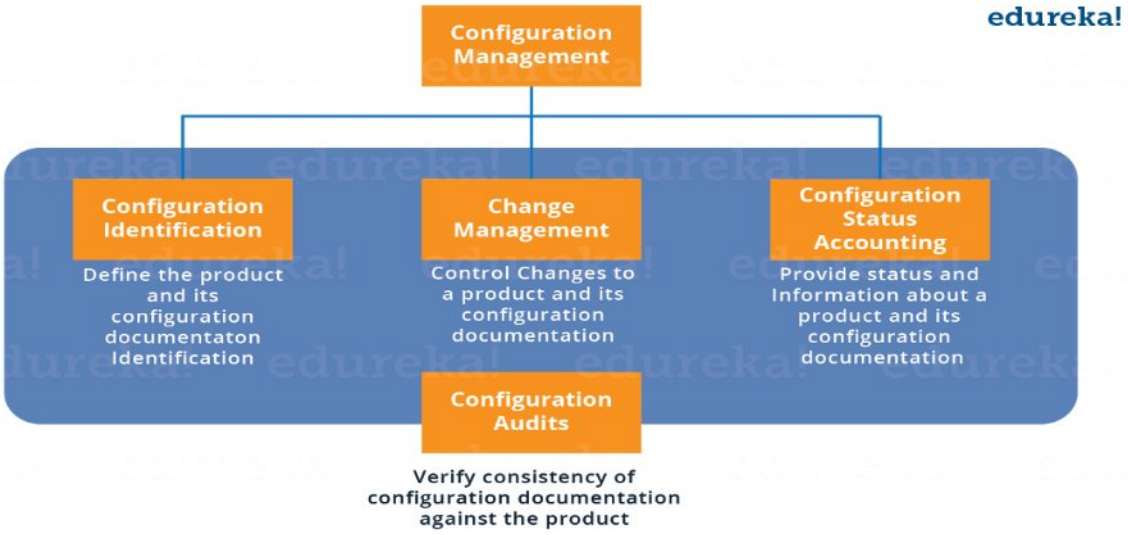
What is Configuration Management?

In my *previous blog*, I have given an introduction to Configuration Management and what challenges it helps us to overcome. In this Puppet Tutorial, I will explain you about different interdependent activities of Configuration Management. But before that, let us understand what is Configuration Item (CI). A Configuration Item is any service component, infrastructure element, or other item that needs to be managed in order to ensure the successful delivery of services. Examples of CI include individual requirements documents, software, models, and plans.

Configuration Management consists of the following elements:

- Configuration Identification
- Change Management
- Configuration Status Accounting
- Configuration Audits

The diagram below explains these components:



Configuration Identification: It is the process of:

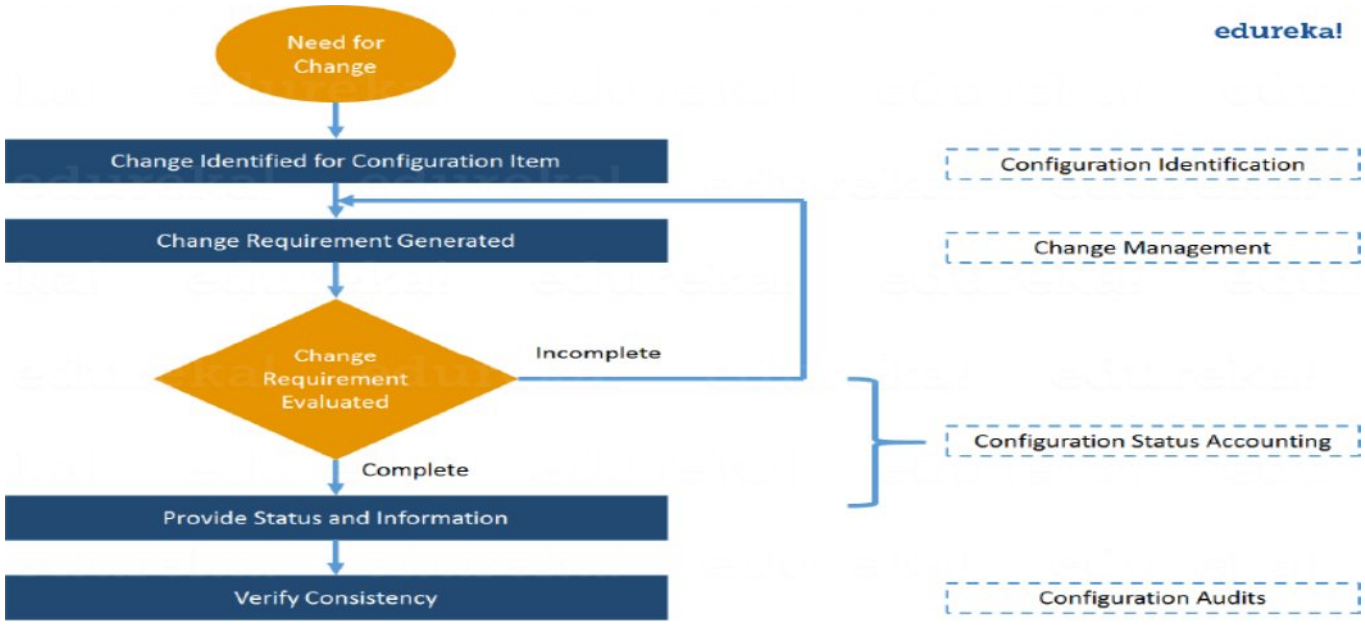
- Labeling software and hardware configuration items with unique identifiers
- Identifying the documentation that describes a configuration item
- Grouping related configuration items into baselines
- Labeling revisions to configuration items and baselines.

Change Management: It is a systematic approach to dealing with change both from the perspective of an organization and the individual.

Configuration Status Accounting: It includes the process of recording and reporting configuration item descriptions (e.g., hardware, software, firmware, etc.) and all departures from the baseline during design and production. In the event of suspected problems, the verification of baseline configuration and approved modifications can be quickly determined.

Configuration Audits: Configuration audits provide a mechanism for determining the degree to which the current state of the system is consistent with the latest baseline and documentation. Basically, it is a formal review to verify that the product being delivered will work as advertised, promoted or in any way promised to the customers. It uses the information available as an outcome of the quality audits and testing along with the configuration status accounting information, to provide assurance that what was required has been build.

Let us understand Configuration Management with a use-case. Suppose if you have to update a particular software or you want to replace it, In that case the below flowchart should be followed for successful Configuration Management:



Now is the correct time to understand Puppet Architecture.

Puppet Tutorial – Components of Puppet

Manifests: Every Slave 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. They are composed of Puppet code and their filenames use the *.pp* extension. These are basically Puppet programs. For example: You can write a Manifest in Puppet Master that creates a file and installs Apache server on all Puppet Slaves connected to the Puppet Master.

Module: 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. Modules are self-contained bundles of code and data.

Resource: Resources are the fundamental unit for modeling system configurations. Each Resource describes some aspect of a system, like a specific service or package.

Facter: Facter gathers basic information (facts) about Puppet Slave 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.

Mcollective: It is a framework that allows several jobs to be executed in parallel on multiple Slaves. It performs various functions like:

- Interact with clusters of Slaves, whether in small groups or very large deployments.
- Use a broadcast paradigm to distribute requests. All Slaves receive all requests at the same time, requests have filters attached, and only Slaves matching the filter will act on requests.
- Use simple command-line tools to call remote Slaves.

- Write custom reports about your infrastructure.

Catalogs: A Catalog describes the desired state of each managed resource on a Slave. It is a compilation of all the resources that the Puppet Master applies to a given Slave, as well as the relationships between those resources. Catalogs are compiled by a Puppet Master from Manifests and Slave-provided data (such as facts, certificates, and an environment if one is provided), as well as an optional external data (such as data from an external Slave classifier, exported resources, and functions). The Master then serves the compiled Catalog to the Slave when requested.

Check out this Puppet tutorial video for deep understanding of Puppet.

What is Puppet? Puppet DevOps Tutorial | DevOps Tools | Edureka

Now in this Puppet Tutorial my next section will focus on Hands-On.

Puppet Tutorial – Hands-On

I will show you how to deploy MySQL and PHP from Puppet Master to Puppet Slave. I am using only one Slave for demonstration purpose there can be hundreds of Slaves connected to one Master. To deploy PHP and MySQL I will use predefined modules available at forge.puppet.com. You can create your own modules as well.

Step 1: In Puppet Master install MySQL and PHP modules.

Execute this:

1) puppet module install puppetlabs-mysql –version 3.10.0

This MySQL module installs, configures, and manages the MySQL service. This module manages both the installation and configuration of MySQL, as well as extending Puppet to allow management of MySQL resources, such as databases, users, and grants.

```
[root@PuppetMaster ~]# puppet module install puppetlabs-mysql --version 3.10.0
Notice: Preparing to install into /etc/puppet/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/modules
└─┬ puppetlabs-mysql (v3.10.0)
  ├── puppet-staging (v2.0.1)
  └── puppetlabs-stdlib (v4.13.1)
```

2) puppet module install mayflower-php –version 4.0.0-beta1

This module is used for managing PHP, in particular php-fpm. PHP-FPM (FastCGI Process Manager) is an alternative PHP FastCGI implementation with some additional features useful for sites of any size, especially busier sites.

```
[root@PuppetMaster ~]# puppet module install mayflower-php --version 4.0.0-beta1
Notice: Preparing to install into /etc/puppet/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppet/modules
└─┬ mayflower-php (v4.0.0-beta1)
  ├── darin-zypprepo (v1.0.2)
  ├─┬ example42-yum (v2.1.27)
  │ └── example42-puppi (v2.2.1)
  ├─┬ puppetlabs-apt (v2.3.0)
  │ └── puppetlabs-stdlib (v4.13.1)
  └── puppetlabs-inifile (v1.6.0)
```

Step 2: In Puppet Manifests include MySQL server and PHP.

Execute this: vi /etc/puppet/manifests/site.pp

You can use any other editor as well like vim, gedit etc. In this site.pp file add the following:

```
1  include '::mysql::server'

2  include '::php'
```

Save and quit.

Step 3: Puppet Slaves pulls its configuration from the Master periodically (after every 30 minutes). It will evaluate the main manifest and apply the module that specifies MySQL and PHP setup. If you want to try it out immediately, you need to run the following command on every Slave node:

Execute this: puppet agent -t

```
[root@PuppetSlave /]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
```

So MySQL and PHP is installed successfully on the Slave node.

Step 4: To check the version of MySQL and PHP installed:

Execute this:

1) mysql -v

```
[root@PuppetSlave /]# mysql -v
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.1.73 Source distribution

Copyright (c) 2000, 2013, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Reading history-file /root/.mysql_history
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>
```

2) php -version

```
[root@PuppetSlave /]# php -version
PHP 5.3.3 (cli) (built: Aug 11 2016 20:23:18)
Copyright (c) 1997-2010 The PHP Group
Zend Engine v2.3.0, Copyright (c) 1998-2010 Zend Technologies
[root@PuppetSlave /]#
```

Congratulations! MySQl and PHP is up and running in your Puppet Slave. Here I have shown you only one Slave but imagine if there are hundreds of Slaves. In that scenario your work becomes so easy, Just specify the configurations in Puppet Master and Puppet Slaves will automatically evaluate the main manifest and apply the module that specifies MySQL and PHP setup.

INTERVIEW QUS

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.

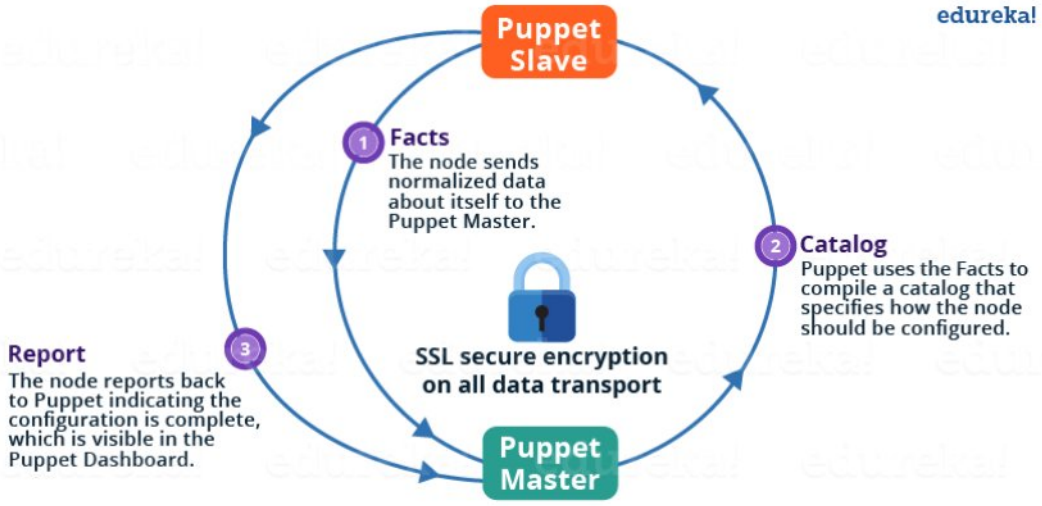
Now, you should describe how Puppet Master and Agent communicates.

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:

Q2. How Puppet Works?

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



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?

It is a very important question and just make sure you go in a correct flow according to me you should first define 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?

For this answer I will prefer the below mentioned explanation:

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?

You are expected to answer what exactly Facter does in Puppet so, according to me you should start by explaining:

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?

I will suggest you to first, tell the uses of 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?

According to me you should mention the command first.

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?

I will advise you to answer this by mentioning the characters:

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:

```
$ FACTER_FOO=”bar”
 $ export FACTER_FOO</span>
 $ facter | grep  ‘foo’</span>
   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.