What Is Puppet?

Puppet is a Configuration Management tool that is used for deploying, configuring and managing servers. It performs the following functions:

- Defining distinct configurations for each and every host, and continuously checking and confirming whether the required configuration is in place and is not altered (if altered Puppet will revert back to the required configuration) on the host.
- Dynamic scaling-up and scaling-down of machines.
- Providing control over all your configured machines, so a centralized (master-server or repo-based) change gets propagated to all, automatically.

What Is Puppet – Key Metrics

Below are few facts about Puppet:

- **Large installed base:** Puppet is used by more than 30,000 companies worldwide including Google, Red Hat, Siemens, etc. along with several universities like Stanford and Harvard law school. An average of 22 new organizations per day use Puppet for the first time.
- **Large developer base:** Puppet is so widely used that lots of people develop for it. Puppet has many contributors to its core source code.
- **Long commercial track record:** Puppet has been in commercial use since 2005, and has been continually refined and improved. It has been deployed in very large infrastructures (5,000+ machines) and the performance and scalability lessons learned from these projects have contributed in Puppet's development.
- **Documentation:** Puppet has a large user-maintained wiki with hundreds of pages of documentation and comprehensive references for both the language and its resource types. In addition, it's actively discussed on several mailing lists and has a very popular IRC channel, so whatever your Puppet problem, it's easy to find the answer.
- **Platform support:** Puppet Server can run on any platform that supports ruby for ex: CentOS, Microsoft Windows Server, Oracle Enterprise Linux etc. It not only supports the new operating systems but it can also run on relatively old and out-of-date OS and Ruby versions as well.

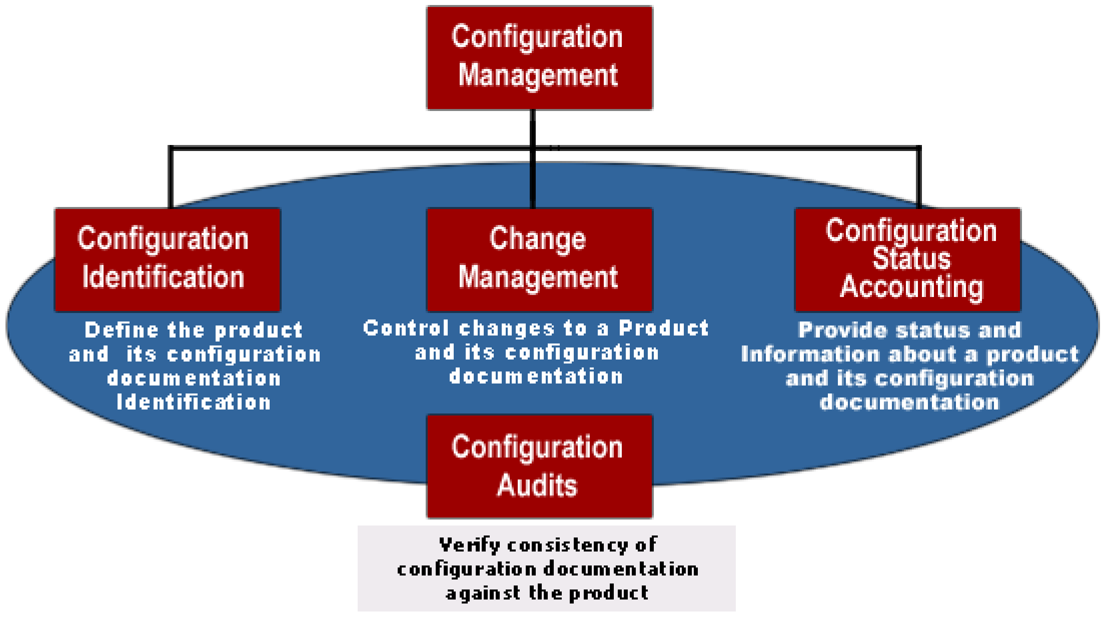
It is now evident that Puppet has huge demand globally.

Configuration Management

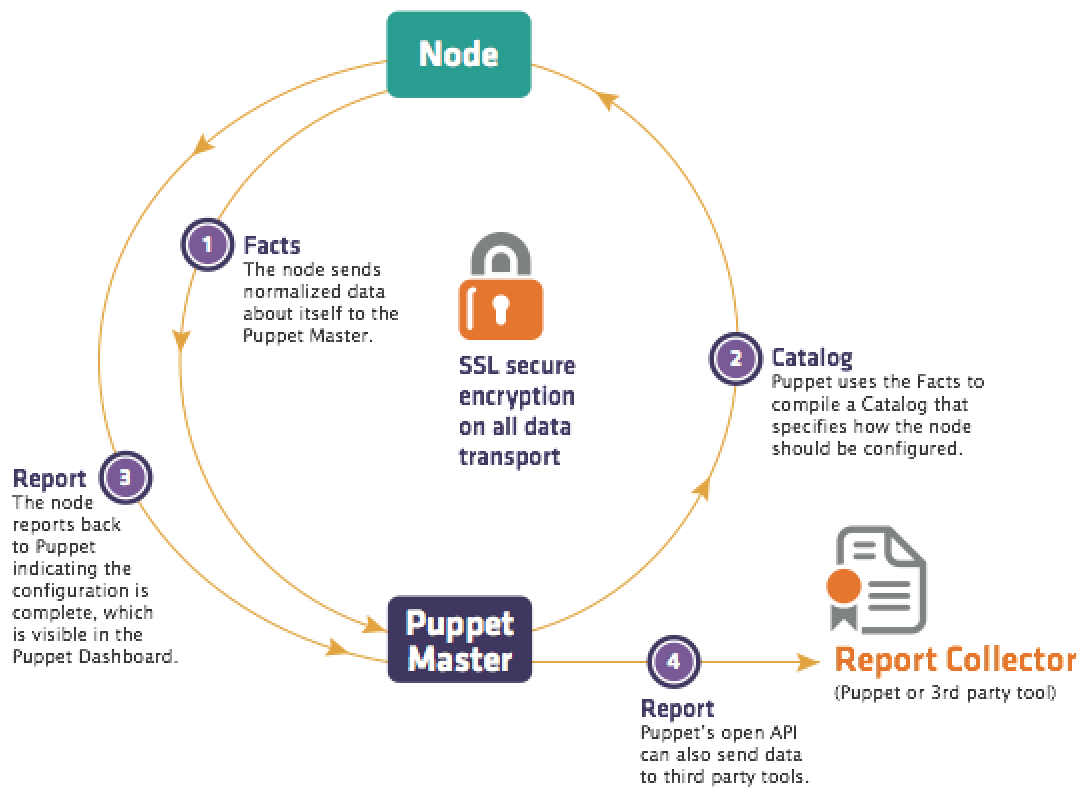
System Administrators usually perform repetitive tasks such as installing servers, configuring those servers, etc. They can automate this task, by writing scripts, but it is a very hectic job when you are working on a large infrastructure.

To solve this problem, *Configuration Management* was introduced. Configuration Management is the practice of handling changes systematically so that a system maintains its integrity over time. Configuration Management (CM) ensures that the current design and build state of the system is known, good & trusted; and doesn't rely on the tacit knowledge of the development team. It allows access to an accurate historical record of system state for project management and audit purposes. Configuration Management overcame the following challenges:

- Figuring out which components to change when requirements change.
- Redoing an implementation because the requirements have changed since the last implementation.
- Reverting to a previous version of the component if you have replaced with a new but flawed version.
- Replacing the wrong component because you couldn't accurately determine which component needed replacing.


Configuration Management
Configuration Identification
Define the product and its configuration documentation Identification
Change Management
Control changes to a Product and its configuration documentation
Configuration Status Accounting
Provide status and Information about a product and its configuration documentation
Configuration Audits
Verify consistency of configuration documentation against the product

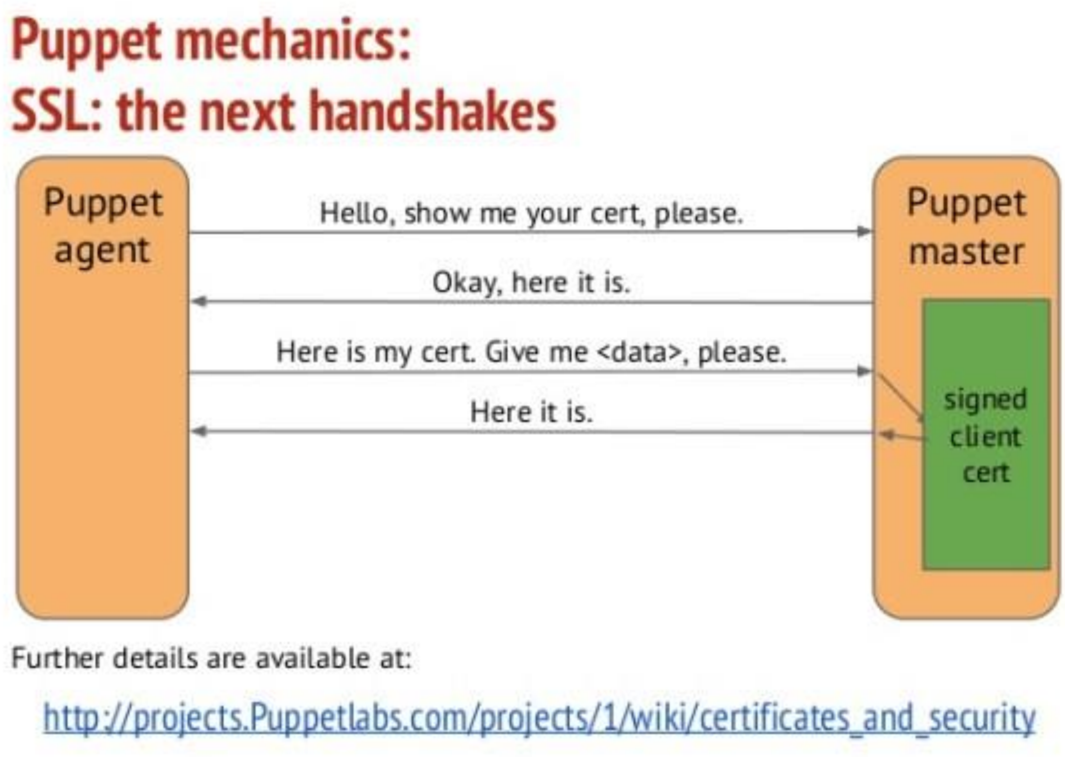
Puppet Architecture:


Node
1 Facts
The node sends normalized data about itself to the Puppet Master.
SSL secure encryption on all data transport
2 Catalog
Puppet uses the Facts to compile a Catalog that specifies how the node should be configured.
Report 3
The node reports back to Puppet indicating the configuration is complete, which is visible in the Puppet Dashboard.
Puppet Master
4
Report
Puppet's open API can also send data to third party tools.
Report Collector
(Puppet or 3rd party tool)

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
- Puppet Master uses the facts to compile a Catalog that defines how the Slave should be configured. Catalog is 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.

Puppet and master slave communication:


Puppet mechanics:
SSL: the next handshakes
Puppet agent
Puppet master
Hello, show me your cert, please.
Okay, here it is.
Here is my cert. Give me <data>, please.
Here it is.
signed client cert
Further details are available at:
http://projects.Puppetlabs.com/projects/1/wiki/certificates_and_security

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.