

EXPERIMENT NO: 10

Date of Performance	
Date of Submission	

AIM

To install and Configure Pull based Software Configuration Management and provisioning tools using Puppet

PROBLEM DEFINITION

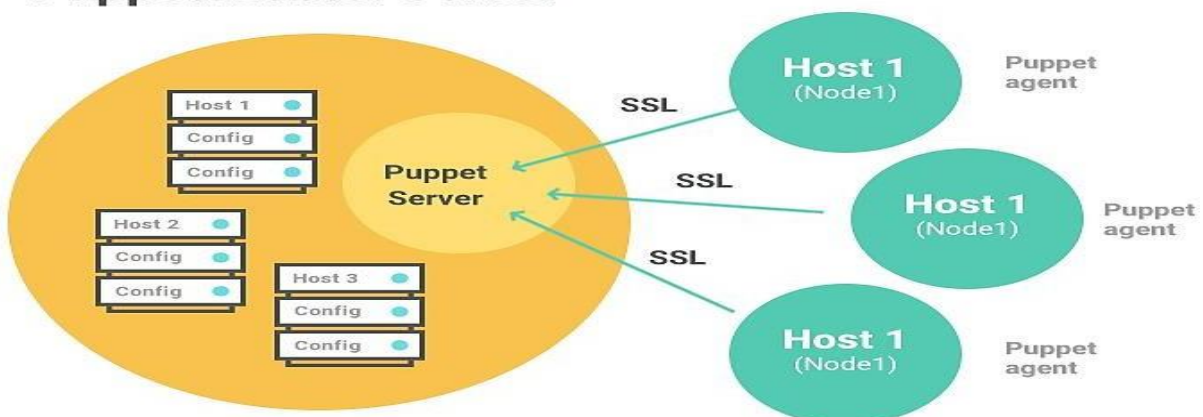
Set up and configure pull-based software configuration management and provisioning tools with Puppet.

THEORY

Configuration Management: Configuration management is the process of maintaining software and computer systems (e.g., servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes. System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate these tasks by writing scripts. However, it is challenging when working on a massive infrastructure. Configuration management tools like Puppet were introduced to resolve such issues.

Puppet: Puppet is a system management tool for centralizing and automating the configuration management process. Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across an organization's infrastructure. Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby and uses its unique Domain-Specific Language (DSL) to describe system configuration.

Puppet Master Server



Puppet performs the following functions:

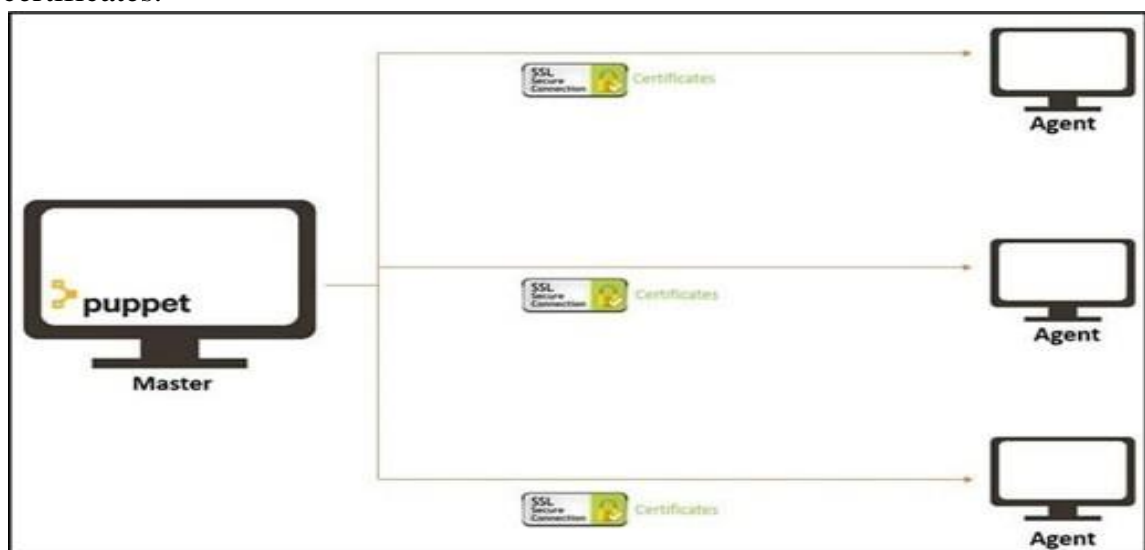
- Puppet allows you to define distinct configurations for every host.
- The tool continuously monitors servers to confirm whether the required configuration exists or not and if it is not altered. If the configuration is changed, the Puppet tool will revert to the pre-defined configuration on the host.
- It also provides control over all the configured systems, so a centralized change gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements infrastructure as code because policies and configurations are written as code.

Pull-based deployment model: In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software, and then configure themselves accordingly—for example, Puppet and Chef.

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

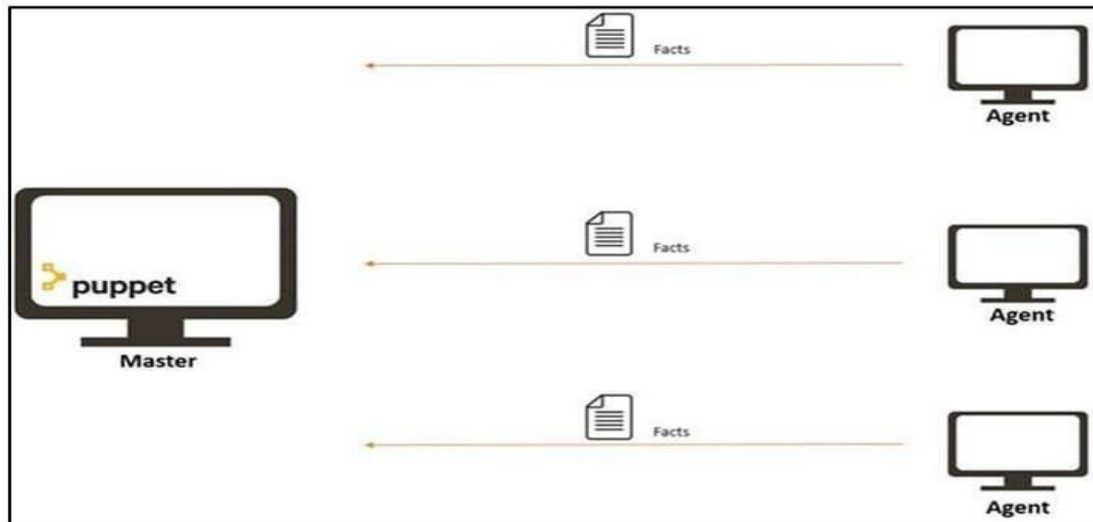
Example: Master – Agent Setup:

- **The Master:** A Linux-based machine with Puppet master software installed on it. It is responsible for maintaining configurations in the form of Puppet codes. The master node can only be Linux.
- **The Agents:** The target machines managed by Puppet with the Puppet agent software installed on them. The agent can be configured on any supported operating system such as Linux, Windows, Solaris, or Mac OS. The communication between master and agent is established through secure certificates.

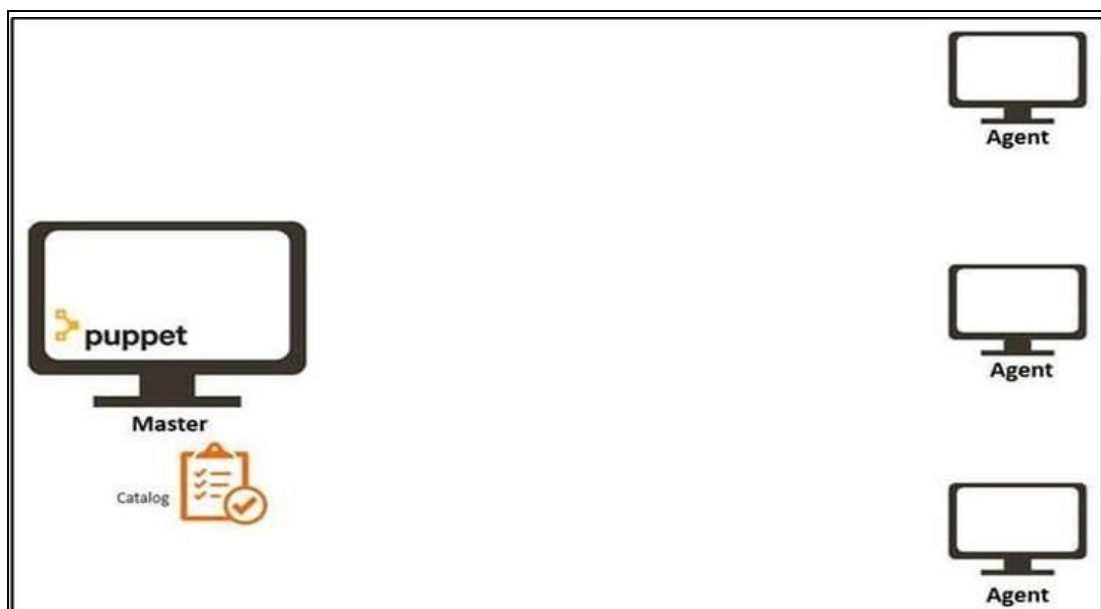


Communication between the Master and the Agent:

1. Once the connectivity is established between the agent and the master, the Puppet agent sends the data about its state to the Puppet master server. These are called Facts: This information includes the hostname, kernel details, IP address, file name details, etc.

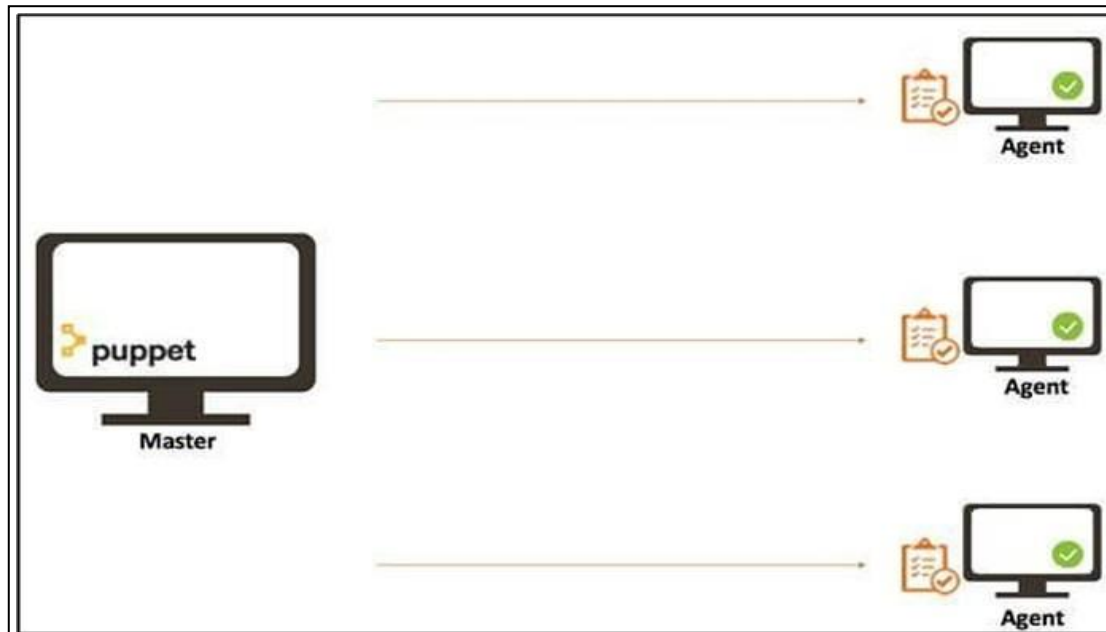


2. Puppet Master uses this data and compiles a list with the configuration to be applied to the agent. This list of configurations to be performed on an agent is known as a catalog. This could involve changes such as package installation, upgrades or removals, File System creation, user creation or deletion, server reboot, IP configuration changes, etc.



3. The agent uses this list of configurations to apply any required configuration changes on the node. If there are no drifts in the configuration, the agent does

not perform any configuration changes and leaves the node to run with the same configuration.

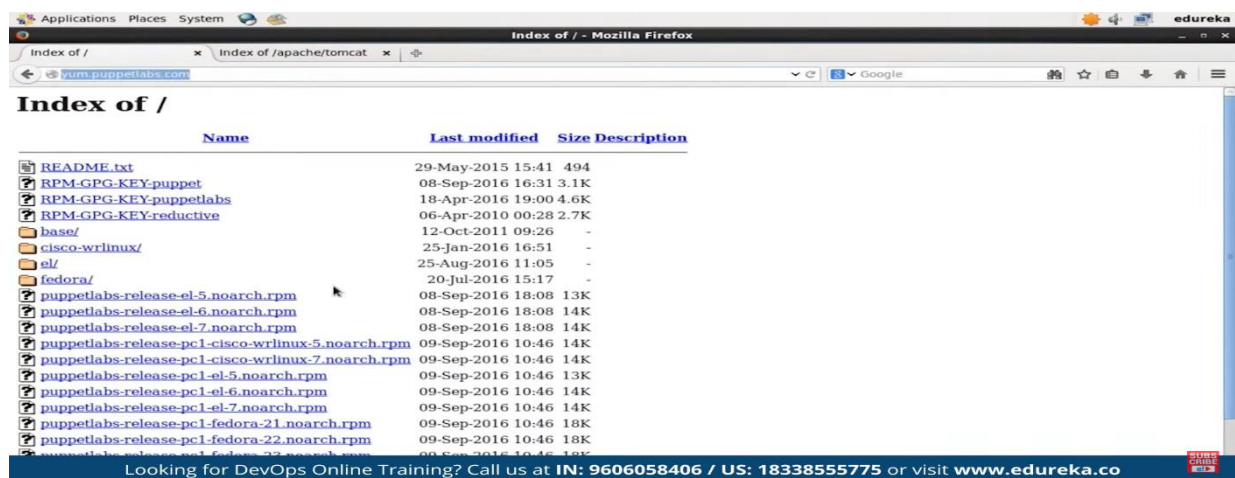


4. Once it is done, the node reports back to Puppet master indicating that the configuration has been applied and completed.

Installation of Puppet on master and agent and initial configuration:

- **Initial configuration on Puppet Master:** Install Puppet software with `apt install puppet master -y` command on the master.
- **On Agent:** Install with `apt install puppet -y` command.
- **Exchanging certificates:** Signing certificate from master.

OUTPUT




```
Applications Places System edureka
edureka@localhost:~
File Edit View Search Terminal Help
ruby 1686 1.8.7.374-4.el6_6 base products 1.6 M
ruby-augeas 1386 0.4.1-3.el6 puppetlabs-de 538 k
ruby-irb 1686 1.8.7.374-4.el6_6 base ps 21 k
ruby-libs 1686 1.8.7.374-4.el6_6 base 317 k
ruby-rdoc 1686 1.8.7.374-4.el6_6 base 1.6 M
ruby-shadow 1386 1:2.0-2.el6 puppetlabs-de 381 k
rubygem-json 1386 1.5.5-3.el6 puppetlabs-deps 12 k
rubygems noarch 1.3.7-5.el6 base ps 763 k
virt-what 1686 1.11-1.2.el6 base 207 k
Updating for dependencies:
libselinux 1686 2.0.94-7.el6 base 24 k
libselinux-python 1686 2.0.94-7.el6 base 109 k
libselinux-utils 1686 2.0.94-7.el6 base 200 k
pciutils-libs 1686 3.1.10-4.el6 base 82 k
34 k

Transaction Summary
=====
Install 17 Package(s)
Upgrade 4 Package(s)

Total download size: 6.6 M
Is this ok [y/N]: y
Downloading Packages:
(1/21): augeas-libs-1.0.0-10.el6.i686.rpm | 311 kB 00:00
(2/21): compat-readline5-5.2-17.1.el6.i686.rpm | 128 kB 00:00
(3/21): facter-2.4.6-1.el6.i386.rpm | 99 kB 00:00
(4/21): hiera-1.3.4-1.el6.noarch.rpm

Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co
SUBSCRIBE

Applications Places System edureka
edureka@localhost:~
File Edit View Search Terminal Help
(17/21): ruby-rdoc-1.8.7.374-4.el6_6.i686.rpm | 381 kB 00:01
(18/21): ruby-shadow-2.2.0-2.el6.i386.rpm | 12 kB 00:00
(19/21): rubygem-json-1.5.5-3.el6.i386.rpm | 763 kB 00:01
(20/21): rubygems-1.3.7-5.el6.noarch.rpm | 207 kB 00:00
(21/21): virt-what-1.11-1.2.el6.i686.rpm | 24 kB 00:00
-----
Total 33 kB/s | 6.6 MB 03:22
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA512 Signature, key ID 4bd6ec30: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Importing GPG key 0x4BD6EC30:
  Userid : Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
  Package: puppetlabs-release-22.0-2.noarch (installed)
  From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Is this ok [y/N]: y
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppet
Importing GPG key 0xEF8D349F:
  Userid : Puppet, Inc. Release Key (Puppet, Inc. Release Key) <release@puppet.com>
  Package: puppetlabs-release-22.0-2.noarch (installed)
  From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppet
Is this ok [y/N]: y
Running rpm_check_debug
Running Transaction Test
Transaction Test Succeeded
Running Transaction
Warning: RPMDB altered outside of yum.
  Updating : libselinux-2.0.94-7.el6.i686 1/25
  Installing : libselinux-ruby-2.0.94-7.el6.i686 2/25

Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co
SUBSCRIBE

Applications Places System edureka
edureka@localhost:~
File Edit View Search Terminal Help
(2/20): compat-readline5-5.2-17.1.el6.i686.rpm | 128 kB 00:03
(3/20): facter-2.4.6-1.el6.i386.rpm | 99 kB 00:01
(4/20): hiera-1.3.4-1.el6.noarch.rpm | 23 kB 00:00
(5/20): libselinux-2.0.94-7.el6.i686.rpm | 109 kB 00:04
(6/20): libselinux-python-2.0.94-7.el6.i686.rpm | 200 kB 00:04
(7/20): libselinux-ruby-2.0.94-7.el6.i686.rpm | 98 kB 00:01
(8/20): libselinux-utils-2.0.94-7.el6.i686.rpm | 82 kB 00:01
(9/20): pciutils-3.1.10-4.el6.i686.rpm | 85 kB 00:01
(10/20): pciutils-libs-3.1.10-4.el6.i686.rpm | 34 kB 00:00
(11/20): puppet-3.8.7-1.el6.noarch.rpm | 1.6 MB 00:11
(12/20): ruby-1.8.7.374-4.el6_6.i686.rpm | 538 kB 00:14
(13/20): ruby-augeas-0.4.1-3.el6.i386.rpm | 21 kB 00:00
(14/20): ruby-irb-1.8.7.374-4.el6_6.i686.rpm | 317 kB 00:06
(15/20): ruby-libs-1.8.7.374-4.el6_6.i686.rpm | 1.6 MB 00:51
(16/20): ruby-rdoc-1.8.7.374-4.el6_6.i686.rpm | 381 kB 00:05
(17/20): ruby-shadow-2.2.0-2.el6.i386.rpm | 12 kB 00:00
(18/20): rubygem-json-1.5.5-3.el6.i386.rpm | 763 kB 00:06
(19/20): rubygems-1.3.7-5.el6.noarch.rpm | 207 kB 00:00
(20/20): virt-what-1.11-1.2.el6.i686.rpm | 24 kB 00:00
-----
Total 49 kB/s | 6.6 MB 02:17
warning: rpmts_HdrFromFdno: Header V4 RSA/SHA512 Signature, key ID 4bd6ec30: NOKEY
Retrieving key from file:///etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Importing GPG key 0x4BD6EC30:
  Userid : Puppet Labs Release Key (Puppet Labs Release Key) <info@puppetlabs.com>
  Package: puppetlabs-release-22.0-2.noarch (installed)
  From : /etc/pki/rpm-gpg/RPM-GPG-KEY-puppetlabs
Is this ok [y/N]:
```

```
Applications Places System edureka@localhost:~
File Edit View Search Terminal Help
[root@PuppetMaster ~]# puppet resource service puppetmaster ensure=running
Notice: /Service[puppetmaster]/ensure: ensure changed 'stopped' to 'running'
service { 'puppetmaster':
  ensure => 'running',
}
[root@PuppetMaster ~]# service puppetmaster status
puppet (pid 14406) is running...
[root@PuppetMaster ~]# c
```

Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

```
Applications Places System edureka@PuppetAgent:~
File Edit View Search Terminal Help
[root@PuppetAgent ~]# puppet agent -t
Info: Creating a new SSL key for puppetagent
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for puppetagent
Info: Certificate Request fingerprint (SHA256): C5:F9:DE:75:94:2C:A2:4A:9A:F8:6C:3C:68:14:10:CF:BB:E6:DC:A4:C8:1C:D7:46:3B:B9:51:D2:1A:CF:B7:AB
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
[root@PuppetAgent ~]#
```

Looking for DevOps Online Training? Call us at IN: 9606058406 / US: 18338555775 or visit www.edureka.co

```
File Edit View Search Terminal Help
[root@PuppetMaster ~]# puppet cert list
"puppetagent" (SHA256) C5:F9:DE:75:94:2C:A2:4A:9A:F8:6C:3C:68:14:10:CF:BB:E6:DC:A4:C8:1C:D7:46:3B:B9:51:D2:1A:CF:B7:AB
[root@PuppetMaster ~]# puppet cert sign puppetagent
Notice: Signed certificate request for puppetagent
Notice: Removing file Puppet::SSL::CertificateRequest puppetagent at '/var/lib/puppet/ssl/ca/requests/puppetagent.pem'
[root@PuppetMaster ~]#
```

```
root@puppet-agent-ubuntu:~# puppet --version
5.4.0
root@puppet-agent-ubuntu:~# nano /etc/puppet/puppet.conf
root@puppet-agent-ubuntu:~# nano /etc/puppet/puppet.conf
root@puppet-agent-ubuntu:~# puppet agent --no-daemonize --onetime --verbose
Info: Creating a new SSL key for puppet-agent-ubuntu.us-east-2.compute.internal
Info: Caching certificate for ca
Info: csr_attributes file loading from /etc/puppet/csr_attributes.yaml
Info: Creating a new SSL certificate request for puppet-agent-ubuntu.us-east-2.compute.internal
Info: Certificate Request fingerprint (SHA256): 0E:25:63:EE:83:1D:08:CB:FC:A8:FA:23:5C:8D:A1:89:DB:66:B8:80:1B:45:ED:8A:59:DC:E6:77:63:85:8D:85
Info: Caching certificate for ca
Exiting; no certificate found and waitforcert is disabled
root@puppet-agent-ubuntu:~#
```

```
ubuntu@puppet:~$ sudo -l
root@puppet:~# puppet cert list -all
"puppet-agent-ubuntu.us-east-2.compute.internal" (SHA256) 0E:25:63:EE:83:1D:08:CB:FC:A8:FA:23:5C:8D:A1:89:DB:66:B8:80:1B:45:ED:8A:59:DC:E6:77:63:85:8D:85
+ "puppet.us-east-2.compute.internal" (SHA256) B2:04:21:F0:3A:FF:26:AB:A6:95:C3:5F:1A:13:84:9D:69:01:88:AB:6E:29:6D:10:98:01:59:37:A1:87:ED:46 (alt names: "DNS:puppet", "DNS:puppet.us-east-2.compute.internal")
root@puppet:~#
```

Signing certificate from master

```
root@puppet:~# puppet cert sign puppet-agent-ubuntu.us-east-2.compute.internal
Signing Certificate Request for:
  "puppet-agent-ubuntu.us-east-2.compute.internal" (SHA256) 0E:25:63:EE:83:1D:08:CB:FC:A8:FA:23:5C:8D:A1:89:DB:66:B8:80:1B
:45:ED:8A:59:DC:E6:77:63:85:8D:85
Notice: Signed certificate request for puppet-agent-ubuntu.us-east-2.compute.internal
Notice: Removing file Puppet::SSL::CertificateRequest puppet-agent-ubuntu.us-east-2.compute.internal at '/var/lib/puppet/s
sl/ca/requests/puppet-agent-ubuntu.us-east-2.compute.internal.pem'
root@puppet:~#
```

CONCLUSION

Hence, installing and configuring pull-based software configuration management and provisioning tools using Puppet facilitated efficient configuration management and automation.

EXPERIMENT NO: 11

Date of Performance	
Date of Submission	

AIM

To learn Software Configuration Management and provisioning using Puppet Blocks (Manifest, Modules, Classes, Function).

PROBLEM DEFINITION

Acquire knowledge in software configuration management and provisioning using Puppet components such as manifests, modules, classes, and functions.

THEORY

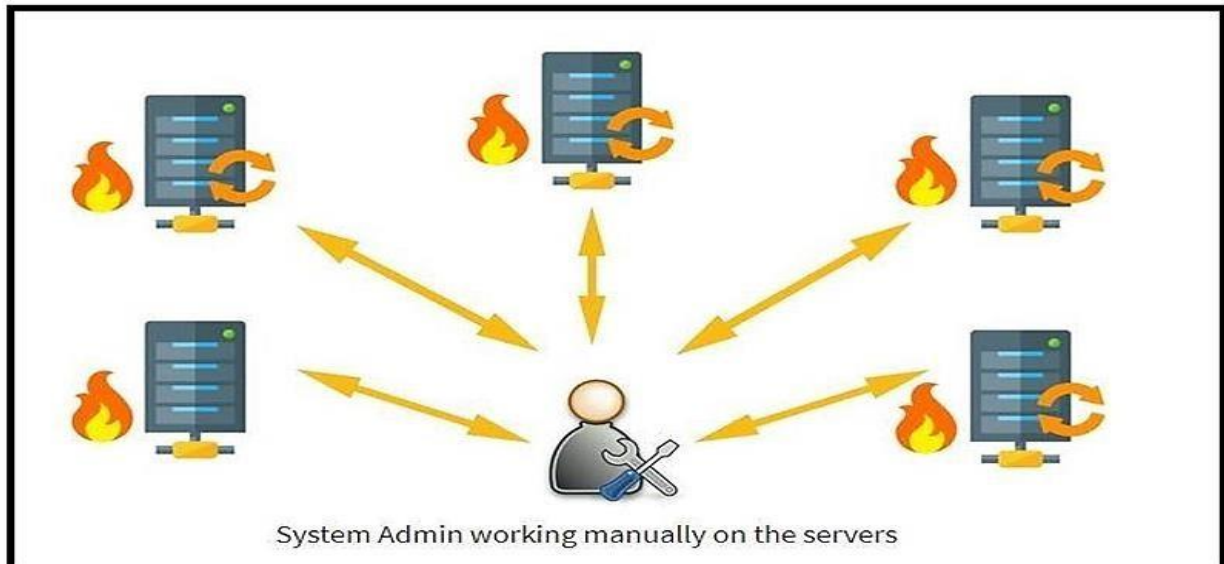
What is Configuration Management? Configuration management is the process of maintaining software and computer systems (e.g., servers, storage, networks) in a known, desired, and consistent state. It also allows access to an accurate historical record of system state for project management and audit purposes. System Administrators mostly perform repetitive tasks like installing servers, configuring those servers, etc. These professionals can automate these tasks by writing scripts. However, it is challenging when working on a massive infrastructure. Configuration management tools like Puppet were introduced to resolve such issues.

What is Puppet? Puppet is a system management tool for centralizing and automating the configuration management process. Puppet is also used as a software deployment tool. It is an open-source configuration management software widely used for server configuration, management, deployment, and orchestration of various applications and services across an organization's infrastructure. Puppet is specially designed to manage the configuration of Linux and Windows systems. It is written in Ruby and uses its unique Domain-Specific Language (DSL) to describe system configuration.

What are the Puppet versions? Puppet comes in two versions:

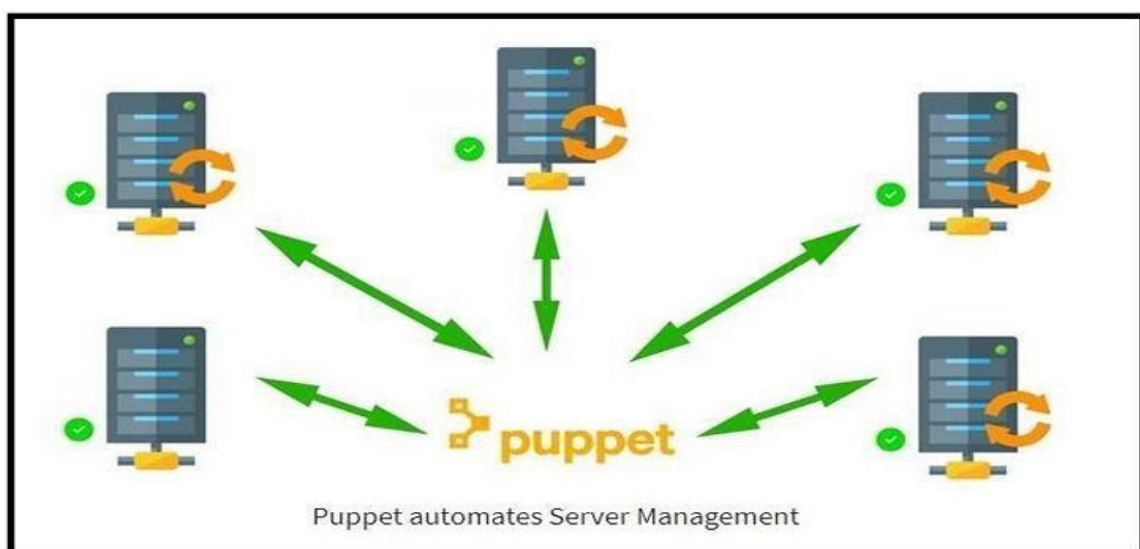
- **Open Source Puppet:** It is a basic version of the Puppet configuration management tool, also known as Open Source Puppet. It is available directly from Puppet's website and is licensed under the Apache 2.0 system.
- **Puppet Enterprise:** A commercial version that offers features like compliance reporting, orchestration, role-based access control, GUI, API, and command-line tools for effective management of nodes.

What Puppet can do: For example, if you have an infrastructure with about 100 servers, Puppet allows you to write simple code that can be deployed automatically on these servers. This reduces human effort and makes the development process faster and more effective.



Puppet performs the following functions:

- Puppet allows you to define distinct configurations for every host.
- The tool continuously monitors servers to confirm whether the required configuration exists or not and if it is not altered. If the configuration is changed, the Puppet tool will revert to the pre-defined configuration on the host.
- It also provides control over all the configured systems, so a centralized change gets automatically effected.
- It is also used as a deployment tool as it automatically deploys software to the system. It implements infrastructure as code because policies and configurations are written as code.



Puppet DSL and Programming Paradigms: Before learning Puppet DSL, let's understand programming paradigms. A programming paradigm is a style used in computer programming. Four types of paradigms are:

1. Imperative
2. Declarative
3. Functional (a subset of the declarative paradigm)
4. Object-oriented

We will focus on Imperative and Declarative.

Imperative Paradigms: This programming paradigm expresses the logic of a computation (what to do) and describes its control flow (how to do it). Example: If you are going to your office, you book a cab and give step-by-step directions to the driver until you reach the office. Specifying what to do and how to do it is an imperative style.

Declarative Paradigms: This programming paradigm expresses the logic of a computation (what to do) without describing its control flow (how to do it). Example: If you are going to your office, you book an Uber cab and specify the final destination (office). Specifying what to do, not how to do it, is a declarative style. Puppet uses a declarative programming approach.

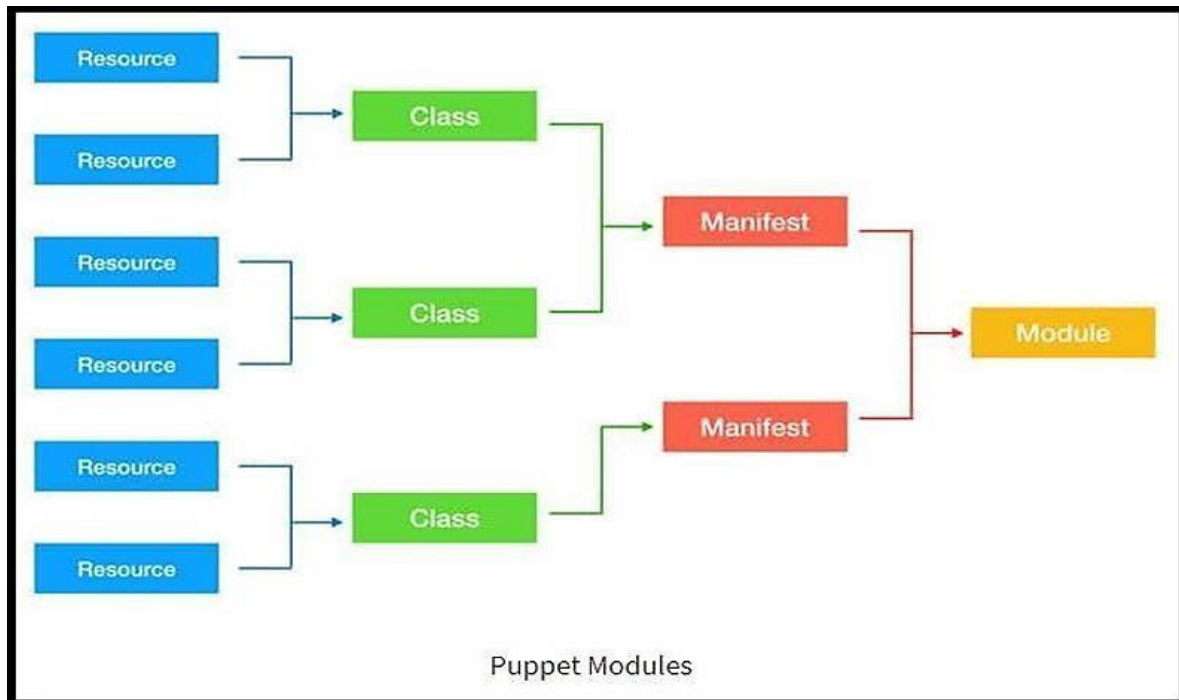
Example: Create a user on the system. It can be done using the imperative programming pattern by a shell script: Here, we specify how to create the user and what commands to use on the operating system. However, it can be done using the declarative programming pattern with only a few lines of Puppet code, Puppet DSL, and still achieve the same result.

Deployment models of configuration management tools: There are two deployment models for configuration management tools:

- **Push-based deployment model:** Initiated by a master node.
- **Pull-based deployment model:** Initiated by agents.

Push-based deployment model: In this deployment model, the master server pushes the configurations and software to the individual agents. After verifying a secure connection, the master runs commands remotely on the agents. For example, Ansible and Salt Stack.

Pull-based deployment model: In this deployment model, individual servers contact a master server, verify and establish a secure connection, download their configurations and software, and then configure themselves accordingly—for example, Puppet and Chef.



Types of Puppet resources

In general, a system consists of files, users, services, processes, packages, etc. In Puppet, these are called resources. Resources are the fundamental building blocks in Puppet. All the operations on puppet agents are performed with the help of puppet resources. Puppet resources are the readymade tools that are used to perform various tasks and operations on any supported platform. We can use a single puppet resource to perform a specific task, or we can use multiple puppet resources together to perform some complex application configurations deployments.

Resources can have different types. Puppet uses resources and resource types in order to describe a system's configuration. There are three kinds of resource types:

1. Puppet core or built-in resource types.
2. Puppet defined resource types.
3. Puppet custom resource types.

Puppet core or built-in resource types:

Core or built-in resource types are the pre-built puppet resource types shipped with puppet software. All of the core or built-in Puppet resource types are written and maintained by Puppet team.

Puppet defined resource types: Defined resource types are lightweight resource types written in Puppet declarative language using a combination of existing resource types.

Puppet custom resource types:

Custom resource types are completely customized resource types written in Ruby. command to display a list of Puppet relevant subcommands:

In our case, we are interested in the subcommand “resource” which we will use to find the information about inbuilt puppet resource types. In the terminal, type any of the following commands to display a list of actions associated with the puppet subcommand “resource“:

In this case, we have the resource as subcommand and –types as action. Puppet has 49 inbuilt core resource types. In the terminal, type the following command to display a list of available inbuilt puppet resource types:

Each type supports a list of attributes. These attributes provide a detailed description that Puppet uses to manage the resource. To find out all the attributes associated with the puppet resource type, use the following command:

```
puppet describe <resource type name>
```

Parameters will list all the available attributes for that resource type. puppet describe package. It’s hard for a new person to understand and relate many unmanaged puppet code files. This is where we need some grouping to tie together operations. The aim is to solve a single problem, such as all operations required to configure ssh on a server or ntp service or a complete web server or database server from scratch.

What Are Puppet Classes?
Puppet classes are the collection of Puppet resources bundled together as a single unit. Puppet introduced classes to make the structure reusable and organized. First, we need to define a class using class definition syntax; classes must be unique and can be declared only once with the same name:

```
class <class-name> {  
<Resource declarations>  
}
```

Example:

```
class ntpconfig {  
  
  file {  
    "/etc/ntp.conf":  
      ensure => "present",  
      content => "server 0.centos.pool.ntp.org iburst\n",  
  }  
}
```

So far, we have only defined the class, but we have not used it anywhere. Meaning this code that we have written will never get executed unless we declare this class elsewhere.

Class

Declaration

To use a defined class in code, use the include keyword.

```
class ntpconfig {  
  file {  
    "/etc/ntp.conf":  
      ensure => "present",  
      content => "server 0.centos.pool.ntp.org iburst\n",  
  }  
}
```

```
include ntpconfig
```

Demo Install NTP First, make sure the NTP package is not already present on the server; the following command will return nothing if the telnet is not present on the server:

As we can see, the NTP package is already present on the server. Let's remove the existing NTP package:

After removing the package, ensure that the ntp.conf file is not existing:

Verify the NTP service does not exist by running the following command:

Create a new .pp file to save the code. From the command line:

Change to insert mode by pressing i from the keyboard. Code to create a new file:

After done with Editing: Press Esc. To save the file, press :wq!.

Next step is to check whether the code has any syntax errors. Execute the following command:

```
puppet parser validate demontp.pp
```

Make sure that you switch to root to be able to complete the test without any error, by executing the command:
su root

Testing is the next step in the code creation process. Execute the following command to perform a smoke test:
puppet apply demontp.pp --noop

Last step is to run Puppet in real mode and verify the output. puppet apply demontp.pp

Puppet didn't perform anything because the demo class was just defined but not declared.

So, until you declare the Puppet class, the code will not get applied. Let's declare the demo class inside the same code using include class name at the end of the code:

Again check whether the code has any syntax errors. Execute the following command: puppet parser validate demontp.pp

Make sure that you switch to root to be able to complete the test without any error, by executing the command: su root

Testing is the next step in the code creation process. Execute the following command to perform a smoke test: puppet apply demontp.pp --noop

Last step is to run Puppet in real mode and verify the output. puppet apply demontp.pp

This time the code gets applied because the class was defined and then declared.

Ensure that ntp.conf is now existing: ls -lrt /etc/ntp.conf

Verify the NTP service has been started by running the following command: systemctl status ntpd

OUTPUT

```
File Edit View Search Terminal Help
[osboxes@puppetselfcontained ~]$ puppet --help

Usage: puppet <subcommand> [options] <action> [options]

Available subcommands:

Common:
agent      The puppet agent daemon
apply      Apply Puppet manifests locally
config     Interact with Puppet's settings.
help       Display Puppet help.
lookup     Interactive Hiera lookup
module     Creates, installs and searches for modules on the Puppet Forge.
resource   The resource abstraction layer shell
```

```
File Edit View Search Terminal Help
[osboxes@puppetselfcontained ~]$ puppet help resource

puppet-resource(8) -- The resource abstraction layer shell
=====

SYNOPSIS
-----
Uses the Puppet RAL to directly interact with the system.

USAGE
-----
puppet resource [-h|--help] [-d|--debug] [-v|--verbose] [-e|--edit]
[-p|--param <parameter>] [-t|--types] [-y|--to_yaml] <type>
[<name>] [<attribute>=<value> ...]
```

```
File Edit View Search Terminal Help
[osboxes@puppetselfcontained ~]$ puppet resource --types
augeas
cron
exec
file
filebucket
group
host
mount
notify
package
resources
schedule
scheduled_task
selboolean
selmodule
service
ssh_authorized_key
sshkey
stage
tidy
user
whit
yumrepo
zfs
```

```
File Edit View Search Terminal Help
[osboxes@puppetselfcontained ~]$ puppet describe package

package
=====
Manage packages. There is a basic dichotomy in package
support right now: Some package types (such as yum and apt) can
retrieve their own package files, while others (such as rpm and sun)
cannot. For those package formats that cannot retrieve their own files,
you can use the 'source' parameter to point to the correct file.
Puppet will automatically guess the packaging format that you are
using based on the platform you are on, but you can override it
using the 'provider' parameter; each provider defines what it
requires in order to function, and you must meet those requirements
to use a given provider.
You can declare multiple package resources with the same 'name', as long
as they specify different providers and have unique titles.
Note that you must use the 'title' to make a reference to a package
resource; 'Package[<NAME>]' is not a synonym for 'Package[<TITLE>]' like
it is for many other resource types.
**Autorequires:** If Puppet is managing the files specified as a
package's 'adminfile', 'responsefile', or 'source', the package
resource will autorequire those files.

Parameters
.....
```

```
File Edit View Search Terminal Help

Parameters
.....

**adminfile**
A file containing package defaults for installing packages.
This attribute is only used on Solaris. Its value should be a path to a
local file stored on the target system. Solaris's package tools expect
either an absolute file path or a relative path to a file in
'/var/sadm/install/admin'.
The value of 'adminfile' will be passed directly to the 'pkgadd' or
'pkgrm' command with the '-a <ADMINFILE>' option.

**allow virtual**
Specifies if virtual package names are allowed for install and
uninstall.
Valid values are 'true', 'false', 'yes', 'no'.
Requires features virtual_packages.

**allowcdrom**
Tells apt to allow cdrom sources in the sources.list file.
Normally apt will bail if you try this.
Valid values are 'true', 'false'.

**category**
```

```
File Edit View Search Terminal Help
[root@puppetselfcontained etc]# rpm -qa | grep -i ntp
ntp-4.2.6p5-28.el7.centos.x86_64
python-ntplib-0.3.2-1.el7.noarch
fontpackages-filesystem-1.44-8.el7.noarch
ntpd-4.2.6p5-28.el7.centos.x86_64
[root@puppetselfcontained etc]#
```

```
File Edit View Search Terminal Help
[root@puppetselfcontained etc]# ls -lrt /etc/ntp.conf
ls: cannot access /etc/ntp.conf: No such file or directory
[root@puppetselfcontained etc]#
```

```
File Edit View Search Terminal Help
[root@puppetselfcontained demo]# systemctl status ntpd
Unit ntpd.service could not be found.
[root@puppetselfcontained demo]#
```

```
File Edit View Search Terminal Help
[root@puppetselfcontained demo]# puppet apply demontp.pp
Notice: Compiled catalog for puppetselfcontained.example.com in environment production
in 0.69 seconds
Notice: /Stage[main]/Ntpconfig/Package[ntp]/ensure: created
Notice: /Stage[main]/Ntpconfig/File[/etc/ntp.conf]/content: content changed '{md5}dc9e5
754ad2bb6f6c32b954c04431d0a' to '{md5}83fd3914bd6bb10f2b717c277d995b75'
Notice: /Stage[main]/Ntpconfig/Service[ntpd]/ensure: ensure changed 'stopped' to 'runni
ng'
Notice: Applied catalog in 1.88 seconds
[root@puppetselfcontained demo]#
```

Testing

```
File Edit View Search Terminal Help
[root@puppetselfcontained demo]# systemctl status ntpd
● ntpd.service - Network Time Service
   Loaded: loaded (/usr/lib/systemd/system/ntpd.service; disabled; vendor preset: disab
led)
   Active: active (running) since Sat 2019-03-23 04:14:35 EDT; 15s ago
     Process: 21487 ExecStart=/usr/sbin/ntpd -u ntp:ntp $OPTIONS (code=exited, status=0/SU
CCESS)
```

CONCLUSION

Thus, learning software configuration management and provisioning with Puppet Blocks (Manifest, Modules, Classes, Function) provided in-depth knowledge for effective configuration and automation.