Puppet

Class Notes

edureka!

Version 1.0

Class Notes – Module wise

Module #1

##Enable Repos

##YUM Systems

#puppetlabs-release-<COLLECTION>-<OS ABBREVIATION>-<OS VERSION>.noarch.rpm

#Therefore for RHEL,

#sudo rpm - Uvh https://yum.puppetlabs.com/puppetlabs-release-pc1-el-7.noarch.rpm

for specific version For centos7/

#sudo rpm -ivh https://yum.puppetlabs.com/el/7/products/x86_64/puppetlabs-release-7-11.noarch.rpm

#for Ubuntu 14.04

```
wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb
dpkg -i puppetlabs-release-trusty.deb
apt-get update
apt-get install puppetmaster
```

#Edir /etc/puppet/puppet.conf edit below 2 lines

[main]

Cert=puppet

dns_alt_names = demo1,demo1.sridemo.com

#Also, remove the line templatedir=$confdir/templates, which has been deprecated.

#Restart Server

```
service puppetmaster start
```

TIP: Before doing any installation and restart, ensure first ensure

DNS has been set up correctly (See Class notes for DNS setup)

Edit /etc/hosts and /etc/hostname and ensure correct hostname has been set (also use hostname command to set this correctly.

For client, the same setup is required. Here are the full steps:

For latest versions, you still need to get the DEB/YUM package and install it..

***** the documentation.

For ubuntu 14.0.4, the agent is already installed...

but if you need to do it,

```
wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb
sudo dpkg -i puppetlabs-release-trusty.deb
sudo apt-get update -y
sudo apt-get install puppet -y
```

TIP: To remove an existing puppet client installation, do the following:

```
sudo apt-get purge puppet puppet-common -y
sudo apt-get autoremove -y
rm -rf /etc/puppet # if you so desire, take a backup
rm -rf /var/lib/puppet
rm -rf /var/run/puppet
```

Change /etc/puppet/puppet.conf

Client install for Ubuntu 14.0.4

```
apt-get install puppet
```

for YUM

#sudo yum install puppet-agent

Depending on whic version, auth.conf is not automatically created..

if not already created, "touch" (create) it...

#touch /etc/puppet/auth.conf

#Edir /etc/puppet/puppet.conf edit below 2 lines

```
[agent]
server = demo1.sridemo.com
```

#Restart client

#service puppet start

to make this auto...

enable in /etc/default/puppet (client) or /etc/default/puppetmaster (server) and set START to yes.

Client install on RPM (specific to Centos)On

rpm -ivh https://yum.puppetlabs.com/el/7/products/x86_64/puppetlabs-release-7-11.noarch.rpm

```
yum install puppet
```

#Edir /etc/puppet/puppet.conf edit below 2 lines

```
[agent]
server = demo1.sridemo.com
```

#Restart client

```
systemctl start puppet
```

TIP: when you install the puppet client, by default, it will create a section called [master]. Rename this to [agent] instead manually.

Now on client, verify you can reach your master and type the following command:

```
puppet agent -t
```

on server, run the following command:

```
puppet cert list –all # you should see your new client listed up without a + sign
pupper cert sign <your client certificate name>
puppet cert list # you should see nothing
puppet cert list –all # you should see your new client listed up with a + sign
```

Now on client, verify you can reach your master and type the following command:

```
puppet agent -t
```

Interesting commands to be seen:

```
puppet agent --fingerprint # on client, used on server to verify client identity
puppet agent --configprint runinterval
puppet config print manifest --section master --environment <ENVIRONMENT>
```

Module #2

Sample code and commands are described below:

#Example #1: Creation of nginx.conf with content retrieved from different sources. The first found resource is applied.

```
file { 'nginx.conf':
  ensure  => present,
  path    => '/etc/nginx/nginx.conf',
  source  => [
     "puppet:///modules/site/nginx.conf--${::fqdn}",
     "puppet:///modules/site/nginx.conf" ],
}
```

#Example #2: Get information on types and resources

```
puppet describe <type>
puppet describe --list
puppet status <server>
puppet resource –types
```

 # Example #3: Apply file apache_php.pp on client locally

```
puppet apply apache_php.pp --verbose
```

apache_php.pp

Download link for: apache_php.pp
https://edureka.wistia.com/medias/u5rybl5r0f/download?media_file_id=135865582

 # Example #4: Examples of a define definition and declaration

```
define apache::virtualhost (
  $ensure   = present,
  $template = 'apache/virtualhost.conf.erb' ,
  [...] ) {
  file { "ApacheVirtualHost_${name}":
    ensure  => $ensure,
    content => template("${template}"),
  }
}
apache::virtualhost { 'www.demo.com':
  template => 'site/apache/www.demo.com.erb'
}
```

#Example #5: Example of scope (define in scopetest.pp and apply locally)

```
# scope-example.pp
# Run with puppet apply --certname www1.example.com scope-example.pp
$myvar = "Top scope value"
node 'www1.example.com' {
  $myvar = "Node scope value"
  notice( "from www1: $myvar" )
  include myclass
}
```

```
node 'db1.example.com' {
  notice( "from db1: $myvar" )
  include myclass
}
class myclass {
  $myvar = "Local scope value"
  notice( "from myclass: $myvar" )
}
# Example #6 of metaparameter usage:
package { 'openssh-server':
  ensure => present,
  before => File['/etc/ssh/sshd_config'],
}

file { '/etc/ssh/sshd_config':
  ensure  => file,
  mode    => '0600',
  source  => 'puppet:///modules/sshd/sshd_config',
  require => Package['openssh-server'],
}
# Example #7: Same as above with more dependencies;

service { 'sshd':
  ensure  => running,
  require => [
    Package['openssh-server'],
    File['/etc/ssh/sshd_config'],
  ],
}
package { 'openssh-server':
  ensure => present,
  before => Service['sshd'],
}
file { '/etc/ssh/sshd_config':
  ensure => file,
  mode   => '0600',
  source => 'puppet:///modules/sshd/sshd_config',
  before => Service['sshd'],
}

#Example #8: Module installation
puppet module list
puppet module upgrade puppetlabs-stdlib
puppet module install puppetlabs-apache
puppet module install puppetlabs-tomcat
## See link on all of this.. https://forge.puppet.com/modules
## https://github.com/puppetlabs/puppetlabs-tomcat
```

Module #3

Extlookup data setup

```
## Data file for ExtLookup
## Created under /etc/puppet/data - make this directory first..
## Update the data to point to your name server and domain
## Tested on Ubuntu 14.04 - adapt accordingly
dnsserver,192.168.33.80
searchdomain,sridemo.com

## Sample PP to verify this..
$extlookup_datadir = "/etc/puppet/data"
$extlookup_precedence = ['common']
#Class definition --> Pay attention to spaces and spelling :-)
class dnsdns {
$dnsserver= extlookup('dnsserver')
$searchdomain = extlookup('searchdomain')
#again check for spacess...
file {'/etc/resolv.conf':
content => "search ${searchdomain}\nnameserver ${dnsserver}\n",
notify => Exec['dns-clean'],
}
# execute 'restart DNS Cache"
exec { 'dns-clean':                 # exec resource named 'dns-clean"
  command => '/etc/init.d/dns-clean restart',  # command this resource will run
  require => File['/etc/resolv.conf']
}
}
# actuall run the class
include dnsdns
## if all ran fine
```

Hiera setup

```
### Hiera is automatically installed with Puppet 3.8.7
## from a command line
hiera ntp_server
hiera ntp_server --yaml web01.example.com.yaml
## The config file is found here /etc/puppet/code/hiera.yaml (or /etc/puppetlabs/code/hiera.yaml. For
Hiera 1,


   /etc/puppet/hiera.yaml in *nix open source Puppet
   /etc/puppetlabs/puppet/hiera.yaml in *nix Puppet Enterprise
   COMMON_APPDATA\PuppetLabs\puppet\etc\hiera.yaml on Windows
)
##use -m or --mcollective IDENTITY to collect data via Mcollective
## Use --config to specify differnt config file..

NOTE: to test this from command line, the file hiera.yaml has to be present in /etc. instead of copying
the file, link it as below:
```

ln -s /etc/puppet/hiera.yaml /etc/hiera.yaml

Without this step, command line testing for hiera will not work.
Sample hiera.yaml file.

:backends:
 - yaml
 - json
:yaml:
 :datadir: /etc/puppet/hieradata
:json:
 :datadir: /etc/puppet/hieradata
:hierarchy:
 - node/%{::fqdn}
 - "%{::environment}"
 - common

Other sample hiera files.


common.yaml


staging.yaml


devweb01.sridemo.com.yaml

Download links:
Common.yaml - https://edureka.wistia.com/medias/oag2wfcae4/download?media_file_id=135866149
Stagin.yaml - https://edureka.wistia.com/medias/ybzepruw8i/download?media_file_id=135866161
Devweb01.sridemo.com.yaml - https://edureka.wistia.com/medias/cvh4du2yqi/download?media_file_id=135866148

These files are used from hiera.yaml file.

Module #4

#Dynamically creating resources…

```
$type = "user"
$resources = {
 'dev1' => { uid    => '1330',
          groups => ['developers', 'operations', 'release'], },
 'dev2'  => { uid    => '1308',
          groups => ['developers', 'prosvc', 'release'], },
}
$defaults = { gid => 'allstaff',
          managehome => true,
          shell      => 'bash',
         }
# create resource like this….
$resources.each |String $resource, Hash $attributes| {
 Resource[$type] {
  $resource: * => $attributes;
  default:   * => $defaults;
 }
```

```
}

# Dependency between foo and bar without anchor pattern

class wrapper {
  contain foo
  contain bar
  contain end

  Class['foo'] ->
  Class['bar'] ->
  Class['end']
}
Include wrapper

# dependency between foo and bar with anchor pattern
class wrapper {.
  anchor { 'wrapper::begin': } ->
  class { 'foo': }          ->
  class { 'bar': }          ->
  class { 'end': }          ->
  anchor { 'wrapper::end': }
}
Include wrapper
# Roles and profile are already covered in project solution and left as an exercise.
```

Module #5

#Enable custom facts

```
# first enable FACTERLIB
#Create a folder /etc/puppet/facts and set it as follow:
mkdir -p /etc/puppet/facts
FACTERLIB=$FACTERLIB:/etc/puppet/facts; export FACTERLIB
#Sample usercount.rb fact
Facter.add('usercount') do
  setcode do
    Facter::Core::Execution.exec('/usr/bin/who | /usr/bin/wc -l')
  end
end
# from command line, run
facter usercount
```

TIP: For puppet master to automatically set this, use the LOAD_PATH variable instead. In /etc/init.d/puppetmaster, set the LOAD_PATH (to see where it is use ruby -e "puts $:"), create a facter subdirectory and place facts there. <MODULE>/lib/facter/ is the directory where you will put in custom facts otherwise.

```
## Sample Code for fact precedence
# Check to see if this server has been marked as a postgres server
Facter.add(:role) do
  has_weight 100
```

```
  setcode do
    if File.exist? '/etc/postgres_server'
      'postgres_server'
    end
  end
end
# Guess if this is a server by the presence of the pg_create binary
Facter.add(:role) do
  has_weight 50
  setcode do
    if File.exist? '/usr/sbin/pg_create'
     'postgres_server'
    end
 end
end
# If this server doesn't look like a server, it must be a desktop
Facter.add(:role) do
 setcode do
   'desktop'
  end
end

#installing  ActiveMQ
## to install ActiveMQ, first install Java
# Next
wget https://archive.apache.org/dist/activemq/5.11.2/apache-activemq-5.11.2-bin.tar.gz

cd [activemq_install_dir]
tar zxvf apache-activemq-5.11.2-bin.tar.gz

#From a command shell, change to the installation directory and run ActiveMQ as a foregroud process:
cd [activemq_install_dir]/bin
./activemq console

#From a command shell, change to the installation directory and run ActiveMQ as a daemon process:
cd [activemq_install_dir]/bin
./activemq start

# Verify this... (user name admin/password admin)
 http://127.0.0.1:8161/admin/

 netstat -an|find "61616" #(also 61614 and 8161)

#next install mcollective on nodes
#Ensure you are installing from official repositories (see Module #1 notes)
sudo apt-get install mcollective

# to install the mcollective agent on an admin workstation
sudo apt-get install mcollective-client

#If all is installed correctly you should see something like below:
```

```
root@puppetmaster: /etc/puppet/facts
root@puppetmaster:/etc/puppet/facts# mco find
devweb01.sridemo.com
puppetmaster.sridemo.com
devweb02.sridemo.com
root@puppetmaster:/etc/puppet/facts# mco ping
devweb01.sridemo.com                     time=40.94 ms
devweb02.sridemo.com                     time=79.23 ms
puppetmaster.sridemo.com                 time=80.07 ms


---- ping statistics ----
3 replies max: 80.07 min: 40.94 avg: 66.74
root@puppetmaster:/etc/puppet/facts#
```

TIP: It is recommended to use puppet to install mcollective on the nodes to be managed. Sample code is below:

```
class mcollective {
 # Install
 package {'mcollective':
   ensure => latest,
 }
 # Run
 service {'mcollective':
   ensure  => running,
   enable  => true,
   require => Package['mcollective'],
 }

 # Restart the service when any settings change
 Package['mcollective'] -> Mcollective::Setting <| |> ~> Service['mcollective']
}
```

The above assumes that only servers are being managed and clients are managed separately.

TIP: If libdir specified in server.cfg or client.cfg does not exist, create one as
mkdir -p /usr/share/mcollective/plugins

Interesting commands:

```
mco find
mco ping
mco inventory <hostname>
mco rpc rpcutil agent_inventory -I some.node
mco rpc service start service=apache2
mco plugin doc service
mco rpc service status service=apache2
mco rpc service status service=httpd -S "environment=development or customer=acme"
mco ping -A service
mco ping -C apache2
mco ping -C /service/
mco ping -F "country=/uk|us/"
mco ping -I dev1 -I dev2
mco ping -I dev1 -I dev2
mco ping -I /sridemo.com$/
```

mco ping -S "((customer=acme and environment=staging) or environment=development) and /apache/"
mco ping -S "environment=development and !customer=acme"
mco ping -S "environment=development and not customer=acme"
mco ping -S "fstat('/etc/hosts').md5=/baa3772104/ and environment=development"
mco rpc nrpe runcommand command=check_load -I dev1 -v #see data verbose
mco rpc nrpe runcommand command=check_load -I dev1 -j # see in JSON format
mco package status kernel

See this link: http://workblog.intothenevernever.com/?p=127 for exposing facts between mcollective and puppet.

mco facts mcollective -v #Output is as below

```
root@puppetmaster:~# mco facts mcollective -v
Discovering hosts using the mc method for 2 second(s) .... 3
Report for fact: mcollective

        1                                               found 3 times

            devweb01.sridemo.com
            devweb02.sridemo.com
            puppetmaster.sridemo.com


---- rpc stats ----
           Nodes: 3 / 3
     Pass / Fail: 3 / 0
      Start Time: 2016-07-04 05:35:25 +0000
  Discovery Time: 2008.72ms
      Agent Time: 88.96ms
      Total Time: 2097.68ms
```

mco facts consumernode -v #Output is as below

```
root@puppetmaster:~# mco facts consumernode -v
Discovering hosts using the mc method for 2 second(s) .... 3
Report for fact: consumernode

        edureka                                         found 1 times

            puppetmaster.sridemo.com


---- rpc stats ----
           Nodes: 3 / 3
     Pass / Fail: 3 / 0
      Start Time: 2016-07-04 05:37:40 +0000
  Discovery Time: 2011.04ms
      Agent Time: 63.95ms
      Total Time: 2074.99ms
```

Module #6

Only Theory and Q/A. No actual demo unless we want to show the following to update from webrick to apache.

By default, webrick is installed and listens on port 8140 (on https). E.g. https://puppetmaster.sridemo.com:8140.

Install Apache 2

- sudo apt-get install apache2
- #sudo apt-get install ruby1.8-dev rubygems
- sudo a2enmod ssl
- sudo a2enmod headers
- apt-get install libcurl4-openssl-dev zlib1g-dev libapr1-dev libaprutil1-dev
- service apache2 restart
- sudo gem install rack passenger
- sudo passenger-install-apache2-module

If you're running Puppet 3.x, make sure the always_cache_features setting is set to true in the [master] (not [main]) section of puppet.conf. This improves performance.

In post-4.0 versions of Puppet, the example config.ru file hardcodes this setting to true.

Make sure Apache's KeepAliveTimeout setting is set to at least 5. (5 is the default value, but your global Apache config may have set a different value, in which case you'll need to change it.)

Although this setting is valid at virtual host scope, the way Apache reads its value means it's safer to set it globally.

Puppet includes a config.ru file, which tells Rack how to spawn Puppet master processes. To install this Rack application in a form Passenger can use, you'll need to:

» Create three directories for the application (a parent directory, a "public" directory, and a "tmp" directory)
» Copy the ext/rack/config.ru file from the Puppet source code into the parent directory
» Set the ownership of the config.ru file

Also, make sure the Apache user (which may vary by platform) can both read and traverse all three directories, can traverse all of its parent directories, and can write to the "tmp" directory.

These steps will look something like this:

```
sudo mkdir -p /usr/share/puppet/rack/puppetmasterd

sudo mkdir /usr/share/puppet/rack/puppetmasterd/public /usr/share/puppet/rack/puppetmasterd/tmp

sudo cp /usr/share/puppet/ext/rack/config.ru /usr/share/puppet/rack/puppetmasterd/

sudo chown puppet:puppet /usr/share/puppet/rack/puppetmasterd/config.ru
```

Sample vhost configuration is below:

Download link:
https://edureka.wistia.com/medias/l482wx3ps9/download?media_file_id=135866837

Adjust paths and permissions as required

On Debian

sudo cp puppetmaster /etc/apache2/sites-available/

sudo a2ensite puppetmaster

RHEL/CentOS:

$ sudo cp puppetmaster.conf /etc/httpd/conf.d/

Start or Restart the Apache service

Ensure that any WEBrick Puppet master process is stopped before starting the Apache service; only one can be bound to TCP port 8140. Basically stop the puppetmaster service

Debian/Ubuntu:

sudo /etc/init.d/apache2 restart

RHEL/CentOS:

sudo /etc/init.d/httpd restart

If all works well, you'll want to make sure the WEBrick service no longer starts on boot:

Debian/Ubuntu:

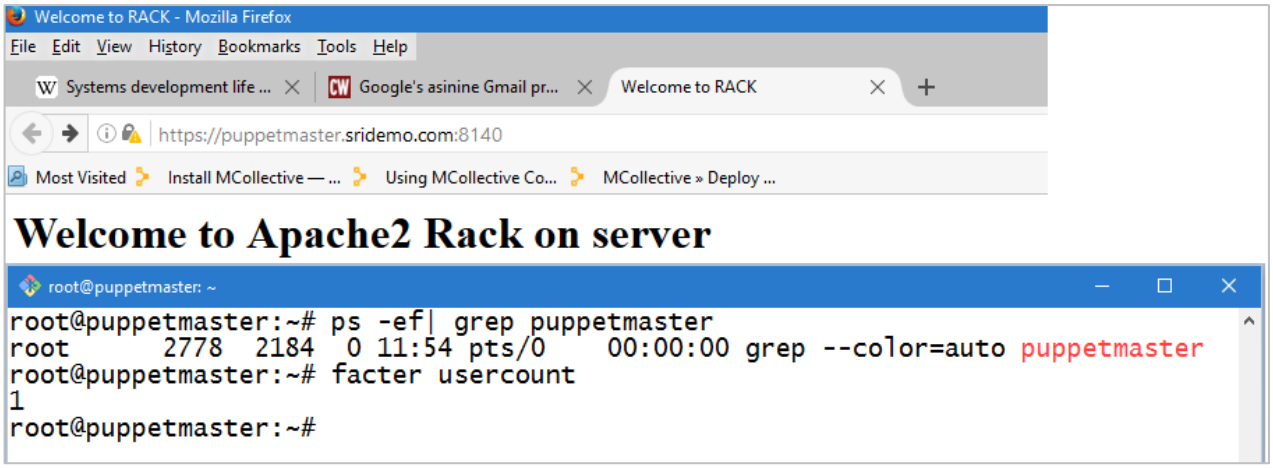
sudo update-rc.d -f puppetmaster remove

RHEL/CentOS:

sudo chkconfig puppetmaster off

sudo chkconfig httpd on

If all goes ok, apache2 should be run (we added our own index.html) and puppetmaster should run as an Apache2 application and custom facts should still run ☺



And puppet file server (after integration should still work ☺)

A much easier way is to install Passenger separately along with Apache. Next install puppet gems and that's it. In both cases, pay close attention to PassengerUser (it should be set to puppet (if not already set)) and the config.ru file should be accessible by this user