

Puppet

Puppet is an open-source IT automation tool. The Puppet Domain Specific Language (DSL) is a Ruby-based coding language that provides a precise and adaptable way to describe a desired state for each machine in your infrastructure. Once you've described a desired state, Puppet does the work to bring your systems in line and keep them there.

Puppet Enterprise (PE) is a complete configuration management platform, with an optimized set of components proven to work well together. It combines a version of open source Puppet (including a preconfigured production-grade Puppet master stack), with MCollective, PuppetDB, Hiera, and more than 40 other open source projects that Puppet Labs has integrated, certified, performance-tuned, and security-hardened to make a complete solution for automating mission-critical enterprise infrastructure.

Contents

Configure Graphite class	2
Assign puppet resources	4
Puppet manifests	6
Cowsay class	6
Fortune class	7
Main class	8
Puppet module	10
Creating a module	10
Using existing module – NTP	13
Using puppet to install and configure a server - MySQL	16
Puppet variables and parameters	19
Variables	19
Class parameter	19
Managing accounts in nodes with conditional statements	22
If statement	22
Unless keyword	23
Case keyword	23
Selector keyword	24
Resource Ordering	25
Chaining arrows and Autorequires	28

Configure Graphite class

Puppet version

```
[root@learning ~]# puppet -V
3.8.1 (Puppet Enterprise 3.8.1)
[root@learning ~]#
```

Using Puppet Enterprise to set up Graphite

Graphite is built from several components, including the Graphite Django webapp frontend, a storage application called Carbon, and Whisper, a lightweight database system. Each of these components has its own set of dependencies and requires its own installation, and configuration.

Search for modules (Collection of config files) of graphite

puppet module search graphite

```
jbusdieker-graphite_web      Puppet Graphite m...  @jbusdieker      graphite
opentable-graphite_powershell Module to send sy...  @opentable       graphite
puppet-grafana               This module provi...  @puppet          graphite
puppet-collectd              Puppet module for...  @puppet          graphite
bfraser-grafana              This module provi...  @bfraser         graphite
olivierHa-influxdb           Puppet module to ...  @olivierHa       graphite
dwerder-grafana              Grafana Dashboard...  @dwerder         graphite
camlow325-grafanadash        Installs graphite...  @camlow325       graphite
bfraser-gdash                This module provi...  @bfraser         graphite
factorit-grafana             This module provi...  @factorit        graphite
garethr-tasseo               Module to manage ...  @garethr         graphite
stevenmerrill-graphiteapi     UNKNOWN               @stevenmerrill   graphite
gadga-stated                 Install stated on ...  @gadga           graphite
```

Install the graphite module **dwerder-graphite**

```
[root@learning ~]# puppet module install dwerder-graphite
Notice: Preparing to install into /etc/puppetlabs/puppet/environments/production/modules ...
Notice: Downloading from https://forgeapi.puppetlabs.com ...
Notice: Installing -- do not interrupt ...
/etc/puppetlabs/puppet/environments/production/modules
├─ dwerder-graphite (v7.1.0)
└─ puppetlabs-stdlib (v4.17.0)
[root@learning ~]#
```

If the machine is offline, install local copy with the following command.

```
for m in /usr/src/forge/*; do puppet module install $m --ignore-
dependencies; done
```

Puppet modules path are found at:

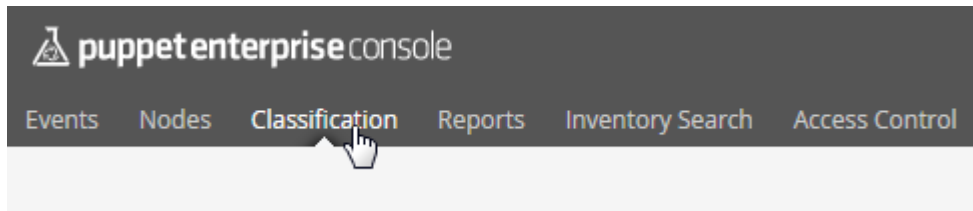
```
/etc/puppetlabs/puppet/environments/production/modules
```

A puppet module contains a main class that generally has the same name as the module name. Graphite class has the instruction puppet needs to set up graphite such as where and how to apply the class across your infrastructure (node - individual computers in a network). This is known as **classification**.

Get ipaddress of learning VM to access Puppet Enterprise console to set up nodes with facter. Facter contains facts about the system, e.g. ipaddress, OS type.

```
[root@learning ~]# facter ipaddress
192.168.1.124
```

Go to <https://ipaddress> to access the console and click on classification



Add new node

Node group name	Parent name	Environment
<input type="text"/>	default	production

Add rules for the node

Nodes in this node group match All of the following rules:

Fact	Operator	Value	Node matches
<input type="text"/>	is	<input type="text"/>	-
name	is	learning.puppetlabs.vm	0

Add graphite class; disable default apache web server since graphite already has its own

Rules	Matching nodes	Classes (3 changes)	Variables	Activity
-------	----------------	---------------------	-----------	----------

Add or modify classes that apply to this group and its child groups.

Add new class

<input type="text"/>

Class: graphite

Parameter		Value
<input type="text"/>	=	<input type="text"/>
gr_disable_webapp_cache	=	true
gr_web_server	=	"none"

Now that you have classified the `learning.puppetlabs.vm` node with the `graphite` class, Puppet knows how the system should be configured, but it won't make any changes until a Puppet run occurs. (Puppet agent does this automatically every 30mins)

To carry out puppet run manually

```
puppet agent --test
```

```
192.168.1.124 - PuTTY
Notice: /Stage[main]/Graphite::Config/File[/opt/graphite/conf/carbon.conf]/ensure: defined content
as '{md5}621a284ff9b16a507301bd3fd21d1955'
Info: /Stage[main]/Graphite::Config/File[/opt/graphite/conf/carbon.conf]: Scheduling refresh of Se
rvice[carbon-cache]
Notice: /Stage[main]/Graphite::Config/File[/opt/graphite/conf/storage-aggregation.conf]/ensure: de
fined content as '{md5}9a9a9319750430659ba6b0c938c9655b'
Notice: /Stage[main]/Graphite::Config/File[/opt/graphite/conf/whitelist.conf]/ensure: defined cont
ent as '{md5}be63d267d82661b9391dbbe59b55b41c'
Notice: /Stage[main]/Graphite::Config/File[/opt/graphite/conf/blacklist.conf]/ensure: defined cont
ent as '{md5}12d3c50b2398a6ba571ed26dc99d169f'
Notice: /Stage[main]/Graphite::Config/File[/opt/graphite/bin/carbon-logrotate.sh]/ensure: defined
content as '{md5}4a210d7c1078fd43468801d653013eaf'
Notice: /Stage[main]/Graphite::Config/Cron[Rotate carbon logs]/ensure: created
Notice: /Stage[main]/Graphite::Config/File[/etc/init.d/carbon-cache]/ensure: defined content as '{
md5}5a5f12808ff845457f19c5a657837736'
Notice: /Stage[main]/Graphite::Config/Service[carbon-cache]/ensure: ensure changed 'stopped' to 'r
unning'
Info: /Stage[main]/Graphite::Config/Service[carbon-cache]: Unscheduling refresh on Service[carbon-
cache]
Notice: Finished catalog run in 93.00 seconds
[root@learning ~]#
[0] 0:*
Quest: Power_of_puppet - Progress: 4/4 Tasks.
```

Assign puppet resources

Resources contain **users**, **files**, **services**, **packages** etc

`puppet resource` = inspect

`puppet describe` = learn about

`puppet apply` = make changes to the resource

Check a user's resource

```
[root@learning ~]# puppet resource user root
user { 'root':
  ensure      => 'present',
  comment     => 'root',
  gid         => '0',
  home        => '/root',
  password    => '$1$jrm5tnjw$h8JJ9mCZLmJvIxxvDLjw1M/',
  password_max_age => '99999',
  password_min_age => '0',
  shell       => '/bin/bash',
  uid         => '0',
}
```

Resource declaration format

```
type {'title':
  attribute => 'value',
}
```

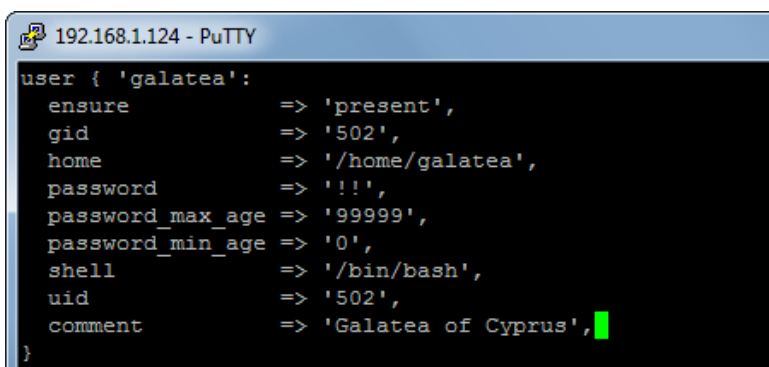
Creating a new user

apply -e is for one time use; i.e. if you want to reference a user again, you have to type the whole resource content

```
[root@learning ~]# puppet apply -e "user { 'galatea': ensure=> 'present', }"
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.63 seconds
Notice: /Stage[main]/Main/User[galatea]/ensure: created
Notice: Finished catalog run in 1.65 seconds
[root@learning ~]# puppet resource user galatea
user { 'galatea':
  ensure      => 'present',
  gid         => '502',
  home        => '/home/galatea',
  password    => '!!!',
  password_max_age => '99999',
  password_min_age => '0',
  shell       => '/bin/bash',
  uid         => '502',
}
[root@learning ~]#
```

To edit the resource using vim, use the following command

```
puppet resource -e user galatea
```



```
user { 'galatea':
  ensure      => 'present',
  gid         => '502',
  home        => '/home/galatea',
  password    => '!!!',
  password_max_age => '99999',
  password_min_age => '0',
  shell       => '/bin/bash',
  uid         => '502',
  comment     => 'Galatea of Cyprus',
}
```

Confirm changes

```
[root@learning ~]# puppet resource user galatea
user { 'galatea':
  ensure      => 'present',
  comment     => 'Galatea of Cyprus',
  gid         => '502',
  home        => '/home/galatea',
  password    => '!!!',
  password_max_age => '99999',
  password_min_age => '0',
  shell       => '/bin/bash',
  uid         => '502',
}
[root@learning ~]#
```

Puppet manifests

Manifest is nothing more than some puppet code saved to a file with the .pp extension

In Puppet's DSL a class is a named block of Puppet code. The class is the next level of abstraction above a resource. A class declares a set of resources related to a single system component.

Cowsay class

To use puppet to manage cowsay package

```
cd /etc/puppetlabs/puppet/environments/production/modules
```

To use cowsay command from command line, we can install cowsay using manifest

```
vim cowsayings/manifests/cowsay.pp
```

Enter the class definition on the pp file.

```
class cowsayings::cowsay {  
  package { ['cowsay']:  
    ensure => 'present',  
  }  
}
```

Use parser to validate the syntax of the pp file. Nothing returned means no error.

```
puppet parser validate cowsayings/manifests/cowsay.pp
```

```
root@learning /etc/puppetlabs/puppet/environments/production/modules# puppet parser validate cow  
sayings/manifests/cowsay.pp  
root@learning /etc/puppetlabs/puppet/environments/production/modules#
```

Declare the class using **include** keyword on test folder

```
vim cowsayings/tests/cowsay.pp
```

```
include cowsayings::cowsay
```

To carry out a dry run of the system, use **noop** keyword

```
puppet apply --noop cowsayings/tests/cowsay.pp
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop cowsayings/tests/cowsay.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.27 seconds
Notice: /Stage[main]/Cowsayings::Cowsay/Package[cowsay]/ensure: current_value absent, should be present (noop)
Notice: Class[Cowsayings::Cowsay]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 4.12 seconds
```

Run it for real without - -noop flag

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply cowsayings/tests/cowsay.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.26 seconds
Notice: /Stage[main]/Cowsayings::Cowsay/Package[cowsay]/ensure: created
Notice: Finished catalog run in 23.32 seconds
```

Confirm it's working.

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# cowsay Hello world!
< Hello world! >
-----
      \   ^__^
       (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
```

Fortune class

Create Manifest

```
vim cowsayings/manifests/fortune.pp
```

Add class definition and validate the file

```
class cowsayings::fortune {
  package { ['fortune-mod']:
    ensure => 'present',
  }
}
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# vim cowsayings/manifests/fortune.pp
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet parser validate cowsayings/manifests/fortune.pp
[root@learning /etc/puppetlabs/puppet/environments/production/modules]#
```

Declare the class file on test folder

```
192.168.1.124 - PuTTY
include cowsayings::fortune
~
```

Carry out dry-run with --noop flag, and then run without if everything is fine

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop cowsayings/tests/fortune.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.27 seconds
Notice: /Stage[main]/Cowsayings::Fortune/Package[fortune-mod]/ensure: current_value absent, should be present (noop)
Notice: Class[Cowsayings::Fortune]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 4.83 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply cowsayings/tests/fortune.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.26 seconds
Notice: /Stage[main]/Cowsayings::Fortune/Package[fortune-mod]/ensure: created
Notice: Finished catalog run in 20.88 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]#
```

Confirm it's working. Fortune calls a random text and that is piped to cowsay

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# fortune | cowsay

/ The day advanced as if to light some \
| work of mine; it was morning, and lo! |
| now it is evening, and nothing      |
| memorable is accomplished.          |
|                                     |
\ -- H.D. Thoreau                      /

-----
      \      ^      ^
      (oo)\_____)
      (__) \       )\/\
           ||----w |
           ||     ||
```

Main class

Often a module will gather several classes that work together into a single class to let you declare everything at once.

*The main class shares the name of the module itself, but instead of following the pattern of naming the manifest for the class it contains, Puppet recognizes the special file name **init.pp** for the manifest that will contain a module's main class.*

Create init.pp file that contains the cowsayings class

```
vim cowsayings/manifests/init.pp
```

Declare the classes as within the tests directory

```
class cowsayings {
  include cowsayings::cowsay
  include cowsayings::fortune
}
```

Validate the file


```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet parser validate cowsayings/manifests/init.pp
[root@learning /etc/puppetlabs/puppet/environments/production/modules]#
```

To carry out testing, delete the previously installed cowsay and fortune class using resource

```
puppet resource package fortune-mod ensure=absent

puppet resource package cowsay ensure=absent
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource package fortune-mod ensure=absent
Notice: /Package[fortune-mod]/ensure: removed
package { 'fortune-mod':
  ensure => 'absent',
}

[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource package cowsay ensure=absent
Notice: /Package[cowsay]/ensure: removed
package { 'cowsay':
  ensure => 'absent',
}
```

Create a test and declare the class

```
vim cowsayings/tests/init.pp
```

Add the following

```
include cowsayings
```

Carry out dry-run with **--noop** flag and run it after

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop cowsayings/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.27 seconds
Notice: /Stage[main]/Cowsayings::Cowsay/Package[cowsay]/ensure: current_value absent, should be present (noop)
Notice: Class[Cowsayings::Cowsay]: Would have triggered 'refresh' from 1 events
Notice: /Stage[main]/Cowsayings::Fortune/Package[fortune-mod]/ensure: current_value absent, should be present (noop)
Notice: Class[Cowsayings::Fortune]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 2 events
Notice: Finished catalog run in 5.04 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply cowsayings/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.28 seconds
Notice: /Stage[main]/Cowsayings::Cowsay/Package[cowsay]/ensure: created
Notice: /Stage[main]/Cowsayings::Fortune/Package[fortune-mod]/ensure: created
Notice: Finished catalog run in 47.98 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]#
```

A class is a collection of related resources and other classes which, once defined, can be declared as a single unit. Puppet classes are also singleton, which means that unlike classes in object oriented programming, a Puppet class can only be declared a single time on a given node.

A manifest is a file containing Puppet code, and appended with the .pp extension. In this quest, we used manifests in the ./manifests directory each to define a single class, and used a corresponding test manifest in the ./tests directory to declare each of those classes.

Puppet module

Modules allow you to organize your Puppet code into units that are testable, reusable, and portable, in short, **modular**. This means that instead of writing Puppet code from scratch for every configuration you need, you can mix and match solutions from a few well-written modules. And because these modules are separate and self-contained, they're much easier to test, maintain, and share than a collection of one-off solutions.

At their root, modules are little more than a structure of directories and files that follow Puppet's naming conventions. The module file structure gives Puppet a consistent way to locate whatever classes, files, templates, plugins, and binaries are required to fulfill the function of the module.

Puppet Labs hosts a free service called the [Forge](#) where you can find a wide array of modules developed and maintained by others.

To find the **modulepath** where all modules are stored

```
puppet master --configprint modulepath
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet master --configprint modulepath
/etc/puppetlabs/puppet/environments/production/modules:/etc/puppetlabs/puppet/modules:/opt/puppet/share/puppet/modules
```

Puppet will look in the 3 directories above to find available modules

To list the modules

```
puppet module list
```

Use **tree** command, **-d** limit the output to directories, and **-L 2** limit the depth to two directories

```
tree -L 2 -d /etc/puppetlabs/puppet/environments/production/modules/
```

Creating a module

The module will manage settings for the VIM editor.

Navigate to the directory puppet master will search for available modules

```
cd /etc/puppetlabs/puppet/environments/production/modules
```

Create a directory for the module

```
mkdir vimrc
```

Make 3 directories, for manifest, tests, file

```
mkdir vimrc/{manifests,tests,files}
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# mkdir vimrc
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# mkdir vimrc/{manifests,tests,files}
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# tree vimrc
vimrc
├── files
├── manifests
└── tests
3 directories, 0 files
```

Copy existing VIM setting files on the Learning VM node group.

```
cp ~/.vimrc vimrc/files/vimrc
```

Edit the file to include **set number**

```
vim vimrc/files/vimrc
```

```
" turn off auto adding comments on next line
" so you can cut and paste reliably
" http://vimdoc.sourceforge.net/html/doc/change.html#fo-table
set fo=tcq
set nocompatible
set modeline
set bg=dark
set number
syntax on
```

The source file is ready. Now write a manifest to tell puppet what to do with it.

- Create a main **init.pp** manifest for vimrc module
- Make vimrc class and tell Puppet server to take the **/vimrc/files/vimrc** file to required node/location
- The vimrc setting file path within puppet is in **/root/.vimrc**. We want to manage this and use a different one.
- **ensure => 'present'**, ensures that the file is present in the system
- **source => ...**, tells what the actual file should contain

```
class vimrc {
  file { [ '/root/.vimrc':
    ensure => 'present',
    source => 'puppet:///modules/vimrc/vimrc',
  ]
}
```

Validate the file

```
root@learning /etc/puppetlabs/puppet/environments/production/modules# vim vimrc/manifests/init.pp
root@learning /etc/puppetlabs/puppet/environments/production/modules# puppet parser validate vimrc/manifests/init.pp
```

Declare and test the class

```
vim vimrc/tests/init.pp
```

```
include vimrc
```

Run with `--noop` flag and without

```
root@learning /etc/puppetlabs/puppet/environments/production/modules# puppet apply --noop vimrc/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.44 seconds
Notice: /Stage[main]/Vimrc/File[/root/.vimrc]/content: current_value {md5}db168521cef060747509a316af8db546, should be {md5}30240d420467b08bba36838c9391cae4 (noop)
Notice: Class[Vimrc]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 0.47 seconds
root@learning /etc/puppetlabs/puppet/environments/production/modules# puppet apply vimrc/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.43 seconds
Notice: /Stage[main]/Vimrc/File[/root/.vimrc]/content: content changed '{md5}db168521cef060747509a316af8db546' to '{md5}30240d420467b08bba36838c9391cae4'
Notice: Finished catalog run in 0.51 seconds
```

It checks old file and new file hash to check if it has been modified.

Vim is now using numbering system on our Learning VM which is also the agent/node.

```
1 include vimrc
2
```

Using existing module – NTP

Check the package

```
puppet resource package ntp
```

Check configuration file

```
puppet resource file /etc/ntp.conf
```

Check if it is running

```
puppet resource service ntpd
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource package ntp
package { 'ntp':
  ensure => '4.2.6p5-3.el6.centos',
}
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource file /etc/ntp.conf
file { '/etc/ntp.conf':
  ensure  => 'file',
  content => '{md5}7fda24f62b1c7ae951db0f746dc6e0cc',
  ctime   => '2015-07-21 09:15:39 +0000',
  group   => '0',
  mode    => '644',
  mtime   => '2015-03-09 14:21:08 +0000',
  owner   => '0',
  type    => 'file',
}
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource service ntpd
service { 'ntpd':
  ensure => 'stopped',
  enable => 'false',
}
[root@learning /etc/puppetlabs/puppet/environments/production/modules]#
```

Install ntp by puppetlabs

```
puppet module install puppetlabs-ntp
```

Module is installed in

```
/etc/puppetlabs/puppet/environments/production/modules
```

`site.pp` is the first manifest the Puppet agent checks when it connects to the master. It defines global settings and resource defaults that will apply to all nodes in your infrastructure. It is also where you will put your **node definitions** (sometimes called `node statements`). A node definition is a block of Puppet code that specifies a set of nodes and declares the classes that Puppet will enforce on those nodes. You can think of it as a code-defined version of the node group you set up in the Power of Puppet quest.

Open the site.pp

```
vim /etc/puppetlabs/puppet/environments/production/manifests/site.pp
```

Include the ntp protocol to be used at bottom of the site.pp file

```
node default {  
  include ntp  
}
```

Parse and run it manually

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# puppet parser validate /etc/  
/puppetlabs/puppet/environments/production/manifests/site.pp  
(root@learning /etc/puppetlabs/puppet/environments/production/modules)#
```

```
puppet agent -t
```

```
Info: Computing checksum on file /etc/ntp.conf  
Info: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]: Filebucketed /etc/ntp.conf to main with sum 7fd  
a24f62b1c7ae951db0f746dc6e0cc  
Notice: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]/content: content changed '{md5}7fda24f62b1c7ae  
951db0f746dc6e0cc' to '{md5}c9d83653966c1e9b8dfbca77b97ff356'  
Info: Class[Ntp::Config]: Scheduling refresh of Class[Ntp::Service]  
Info: Class[Ntp::Service]: Scheduling refresh of Service[ntp]  
Notice: /Stage[main]/Ntp::Service/Service[ntp]/ensure: ensure changed 'stopped' to 'running'  
Info: /Stage[main]/Ntp::Service/Service[ntp]: Unscheduling refresh on Service[ntp]  
Notice: Finished catalog run in 73.74 seconds
```

Check which default time server ntp is using

```
grep server /etc/ntp.conf
```

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# grep server /etc/ntp.conf  
server 0.centos.pool.ntp.org  
server 1.centos.pool.ntp.org  
server 2.centos.pool.ntp.org
```

Replace the `include ntp` on `node default {}` of site.pp with

```
class { 'ntp':  
  servers =>  
    ['nist-time-server.eoni.com', 'nist1-lv.ustiming.org', 'ntp-  
nist.ldsbc.edu']  
}
```

Parse and run

```
Info: Computing checksum on file /etc/ntp.conf
Info: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]: Filebucketed /etc/ntp.conf to main with sum c9d8
3653966c1e9b8dfbca77b97ff356
Notice: /Stage[main]/Ntp::Config/File[/etc/ntp.conf]/content: content changed '{md5}c9d83653966c1e9b
8dfbca77b97ff356' to '{md5}3c026c141e3b177a2be3fd476fe99255'
Info: Class[Ntp::Config]: Scheduling refresh of Class[Ntp::Service]
Info: Class[Ntp::Service]: Scheduling refresh of Service[ntp]
Notice: /Stage[main]/Ntp::Service/Service[ntp]: Triggered 'refresh' from 1 events
Notice: Finished catalog run in 93.99 seconds
```

Using puppet to install and configure a server - MySQL

Install the mysql module

```
puppet module install puppetlabs-mysql
```

To classify the Learning VM with MySQL server class, edit the

```
/etc/puppetlabs/puppet/environments/production/manifests/site.pp
```

Create password and set max connection allowed to 1024

```
class { '::mysql::server':  
  root_password    => 'strongpassword',  
  override_options => { 'mysqld' => { 'max_connections' => '1024' } },  
}
```

Parse and validate then run it with **puppet agent -t**

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# puppet parser validate /etc/  
puppetlabs/puppet/environments/production/manifests/site.pp  
Error: Could not parse for environment production: Syntax error at end of file; expected '}' at /etc/  
puppetlabs/puppet/environments/production/manifests/site.pp:51  
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# vim /etc/puppetlabs/puppet/e  
nvironments/production/manifests/site.pp  
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# puppet parser validate /etc/  
puppetlabs/puppet/environments/production/manifests/site.pp  
(root@learning /etc/puppetlabs/puppet/environments/production/modules)#
```

Check the result on MySQL **my.conf** file

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# cat /etc/my.cnf | grep -B 9  
max_connections  
[mysqld]  
basedir = /usr  
bind-address = 127.0.0.1  
datadir = /var/lib/mysql  
expire_logs_days = 10  
key_buffer_size = 16M  
log-error = /var/log/mysqld.log  
max_allowed_packet = 16M  
max_binlog_size = 100M  
max_connections = 1024  
(root@learning /etc/puppetlabs/puppet/environments/production/modules)#
```

Check MySQL manifests


```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# tree -P *.pp /etc/puppetlabs
/puppet/environments/production/modules/mysql/manifests/
/etc/puppetlabs/puppet/environments/production/modules/mysql/manifests/
├── backup.pp
├── bindings
│   ├── java.pp
│   ├── perl.pp
│   ├── php.pp
│   ├── python.pp
│   └── ruby.pp
├── bindings.pp
├── client
│   └── install.pp
├── client.pp
├── db.pp
├── init.pp
├── params.pp
├── server
│   ├── account_security.pp
│   ├── backup.pp
│   ├── config.pp
│   ├── install.pp
│   ├── monitor.pp
│   ├── mysqltuner.pp
│   ├── providers.pp
│   ├── root_password.pp
│   ├── service.pp
│   └── server.pp
└── server.pp

3 directories, 22 files
```

Trigger the `server::account_security.pp` class to remove default accounts

Edit the `site.pp` file and include that class in `node default {..}`

```
mysql::server::account_security
```

```
Notice: /Stage[main]/Mysql::Server::Account_security/Mysql_user[root@learning.puppetlabs.vm]/ensure: removed
Notice: /Stage[main]/Mysql::Server::Account_security/Mysql_user[root@127.0.0.1]/ensure: removed
Notice: /Stage[main]/Mysql::Server::Account_security/Mysql_user[@learning.puppetlabs.vm]/ensure: removed
Notice: /Stage[main]/Mysql::Server::Account_security/Mysql_user[@localhost]/ensure: removed
Notice: /Stage[main]/Mysql::Server::Account_security/Mysql_database[test]/ensure: removed
Notice: Finished catalog run in 103.80 seconds
```

Create a database with puppet by including the following on `site.pp`

```
mysql_database { 'lvm':
  ensure => 'present',
  charset => 'utf8',
}
```

Add user

```
mysql_user { 'lvm_user@localhost':
  ensure => 'present',
}
```

Grant privilege

```
mysql_grant { 'lvm_user@localhost/lvm.*':
  ensure       => 'present',
  options      => ['GRANT'],
  privileges   => ['ALL'],
}
```

```
table      => 'lvm.*',
user       => 'lvm_user@localhost',
}
```

```
50 include mysql::server::account_security
51 mysql_database { 'lvm':
52     ensure => 'present',
53     charset => 'utf8',
54 }
55 mysql_user { 'lvm_user@localhost':
56     ensure => 'present',
57 }
58 mysql_grant { 'lvm_user@localhost/lvm.*':
59     ensure      => 'present',
60     options     => ['GRANT'],
61     privileges  => ['ALL'],
62     table       => 'lvm.*',
63     user        => 'lvm_user@localhost',
64 }
65 }
```

Parse and run

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet parser validate /etc/
puppetlabs/puppet/environments/production/manifests/site.pp
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for learning.puppetlabs.vm
Info: Applying configuration version '1496682890'
Notice: /Stage[main]/Main/Node[default]/Mysql_database[lvm]/ensure: created
Notice: /Stage[main]/Main/Node[default]/Mysql_user[lvm_user@localhost]/ensure: created
Notice: /Stage[main]/Main/Node[default]/Mysql_grant[lvm_user@localhost/lvm.*]/ensure: created
Notice: Finished catalog run in 92.92 seconds
```

Puppet variables and parameters

Variables

On puppet module path, create a directory with manifests and tests folder

```
root@learning /etc/puppetlabs/puppet/environments/production/modules# mkdir web
root@learning /etc/puppetlabs/puppet/environments/production/modules# mkdir web/{manifests,tests}
root@learning /etc/puppetlabs/puppet/environments/production/modules#
```

Make init.pp file with the following

```
vim web/manifests/init.pp
```

```
class web {

    $doc_root = '/var/www/html/questguide/'
    $english  = 'Hello world!'
    $french   = 'Bonjour le monde!'

    file { ["${doc_root}hello.html":
        ensure => 'present',
        content => "<em>${english}</em>",
    ]
    file { ["${doc_root}bonjour.html":
        ensure => 'present',
        content => "<em>${french}</em>",
    ]
}
```

Parse the manifest file; then create class declaration on tests folder and include that class

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# puppet parser validate web/manifests/init.pp
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# vim web/tests/init.pp
```

```
1 include web
```

Carry out dry run

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules)# puppet apply --noop web/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.47 seconds
Notice: /Stage[main]/Web/File[/var/www/html/questguide/hello.html]/ensure: current_value absent, should be present (noop)
Notice: /Stage[main]/Web/File[/var/www/html/questguide/bonjour.html]/ensure: current_value absent, should be present (noop)
Notice: Class[Web]: Would have triggered 'refresh' from 2 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 0.36 seconds
```

Class parameter

A class can be defined with variable within manifests

```
class classname ( $parameter = 'default' ) {
  ...
}
```

Then it can be **declared** within tests folder like resources

```
class { 'classname':
  parameter => 'value',
}
```

Change the web class created earlier to add parameters and use them in file

```
class web ( $page_name, $message ) {
  file { "${doc_root}${page_name}.html":
    ensure => 'present',
    content => "<em>${message}</em>",
  }
}
```

```
1 class web ( $page_name, $message) {
2
3   $doc_root = '/var/www/html/questguide/'
4   $english = 'Hello world!'
5   $french = 'Bonjour le monde!'
6
7   file { "${doc_root}hello.html":
8     ensure => 'present',
9     content => "<em>${english}</em>",
10  }
11  file { "${doc_root}bonjour.html":
12    ensure => 'present',
13    content => "<em>${french}</em>",
14  }
15  file { "${doc_root}${page_name}.html":
16    ensure => 'present',
17    content => "<em>${message}</em>",
18  }
19 }
```

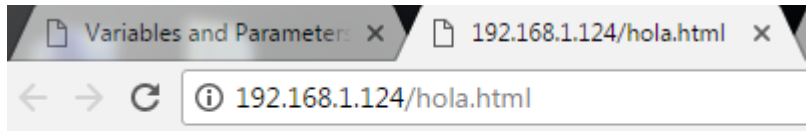
Declare the class with required variables in its **web/tests/init.pp** file

```
class { 'web':
  page_name => 'hola',
  message => 'Hola mundo!',
}
```

Carry out dry run before actual run

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop web/test
s/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.50 seconds
Notice: /Stage[main]/Web/File[/var/www/html/questguide/hola.html]/ensure: current_value absent, shou
ld be present (noop)
Notice: Class[Web]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 0.37 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply web/tests/init.
pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.45 seconds
Notice: /Stage[main]/Web/File[/var/www/html/questguide/hola.html]/ensure: created
Notice: Finished catalog run in 0.35 seconds
```

The file can be seen; page is **hola.html** and message is **'Hola mundo!'**.



Hola mundo!

Managing accounts in nodes with conditional statements

If statement

Use puppet facts to make your modules portable.

Full list of facts available

```
factor -p | less
```

(Press q to exit)

Create account directory

```
mkdir accounts  
  
mkdir accounts/{manifests,tests}
```

On accounts/manifests/init.pp include the following

```
1 class accounts ($name) {  
2   if $::operatingsystem == 'centos' {  
3     $groups = 'wheel'  
4   }  
5   elsif $::operatingsystem == 'debian' {  
6     $groups = 'admin'  
7   }  
8   else {  
9     fail( "This module doesn't support ${::operatingsystem}." )  
10  }  
11  notice ( "Groups for user ${name} set to ${groups}" )  
12  user { $name:  
13    ensure => present,  
14    home   => "/home/${name}",  
15    groups => $groups,  
16  }  
17 }
```

On accounts/tests/init.pp declare the class with a user

```
1 class ['accounts':  
2   name => 'dana',  
3 ]
```

Check how the manifest will run on debian using factor

```
FACTER_operatingsystem=Debian puppet apply --noop accounts/tests/init.pp
```

Dana's group is correctly set to admin

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# FACTER_operatingsystem=Debian puppet apply --noop accounts/tests/init.pp
Notice: Scope(Class[Accounts]): Groups for user dana set to admin
```

```
FACTOR_operatingsystem=Darwin puppet apply --noop
accounts/tests/init.pp
```

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# FACTER_operatingsystem=Darwin puppet apply --noop accounts/tests/init.pp
Error: This module doesn't support Darwin. at /etc/puppetlabs/puppet/environments/production/modules/accounts/manifests/init.pp:9 on node learning.puppetlabs.vm
```

Run it within the Learning VM which is CentOS (Master and agent is same in this case)

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop accounts/tests/init.pp
Notice: Scope(Class[Accounts]): Groups for user dana set to wheel
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.47 seconds
Notice: /Stage[main]/Accounts/User[dana]/ensure: current value absent, should be present (noop)
Notice: Class[Accounts]: Would have triggered 'refresh' from 1 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 0.37 seconds
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply accounts/tests/init.pp
Notice: Scope(Class[Accounts]): Groups for user dana set to wheel
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.46 seconds
Notice: /Stage[main]/Accounts/User[dana]/ensure: created
Notice: Finished catalog run in 1.10 seconds
```

Check user dana and group is correctly set to wheel

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet resource user dana
user { 'dana':
  ensure      => 'present',
  gid         => '503',
  groups      => ['wheel'],
  home        => '/home/dana',
  password    => '!!!',
  password_max_age => '99999',
  password_min_age => '0',
  shell       => '/bin/bash',
  uid         => '503',
}
```

Unless keyword

Opposite of “if statement”. Only execute if the conditional statement is false. If true, do nothing.

Case keyword

```
case $::operatingsystem {
  'CentOS': { $apache_pkg = 'httpd' }
  'Redhat': { $apache_pkg = 'httpd' }
  'Debian': { $apache_pkg = 'apache2' }
  'Ubuntu': { $apache_pkg = 'apache2' }
  default: { fail("Unrecognized operating system for webserver.") }
}

package { $apache_pkg :
  ensure => present,
```

```
}
```

Selector keyword

```
$rootgroup = $::osfamily ? {  
  'Solaris'   => 'wheel',  
  'Darwin'    => 'wheel',  
  'FreeBSD'   => 'wheel',  
  'default'   => 'root',  
}
```


Resource Ordering

Create sshd directory with manifests and tests sub directory.

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# mkdir sshd
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# mkdir sshd/{manifests,tests}
```

Fill in sshd class with the openssh-server package resource and sshd service resource.

If you're writing a module to manage SSH, for instance, you will need to ensure that the openssh-server package is installed before you try to manage the sshd service

Create **init.pp** file on **sshd/manifests/**

```
1 class sshd{
2   package { 'openssh-server':
3     ensure => present,
4     before => Service['sshd'],
5   }
6   service { 'sshd':
7     ensure => running,
8     enable => true,
9   }
10 }
```

Declare the class using **init.pp** file under **sshd/tests/**

```
include sshd
```

When Puppet compiles a catalog, it generates a graph that represents the network of resource relationships in that catalog.

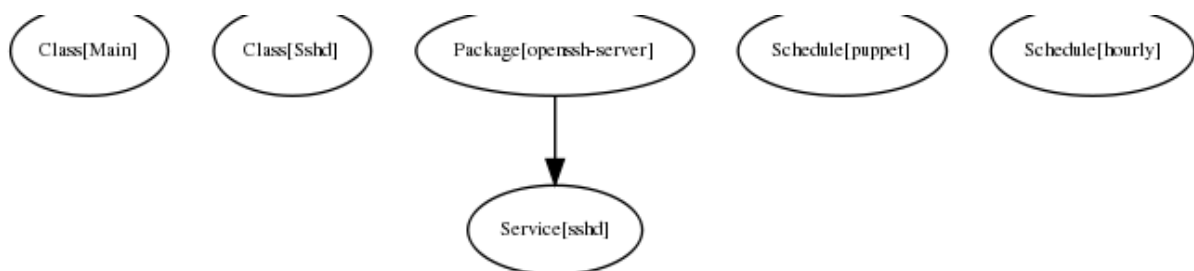
Dry run

```
puppet apply sshd/tests/init.pp --noop --graph
```

Real run

```
puppet config print graphdir
```

```
dot -Tpng /var/opt/lib/pe-puppet/state/graphs/relationships.dot -o
/var/www/html/questguide/relationships.png
```



Relationships

To configure the sshd copy its config file

```
cp /etc/ssh/sshd_config sshd/files/sshd_config
```

Make some change to the file **sshd/files/sshd_config** such as change **GSSAPIAuthentication** line from **yes** to **no**

```
79 # GSSAPI options
80 #GSSAPIAuthentication no
81 GSSAPIAuthentication no
82 #GSSAPICleanupCredentials yes
83 GSSAPICleanupCredentials yes
84 #GSSAPIStrictAcceptorCheck yes
85 #GSSAPIKeyExchange no
```

Go to manifest init file to add file resource

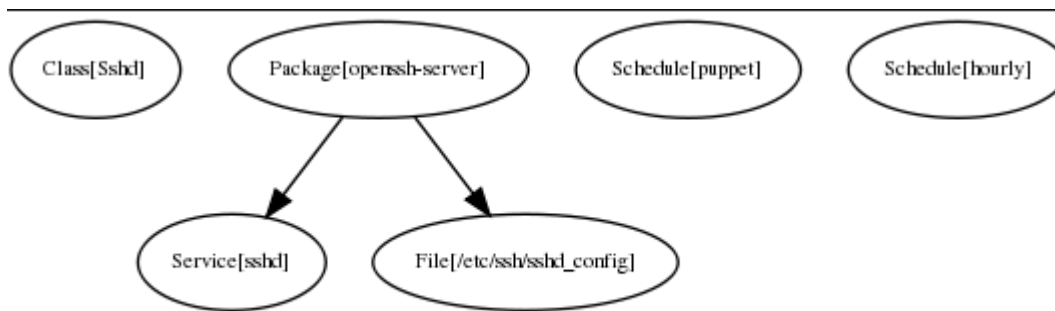
```
1 class sshd{
2   package { 'openssh-server':
3     ensure => present,
4     before => Service['sshd'],
5   }
6   service { 'sshd':
7     ensure => running,
8     enable => true,
9   }
10  file {'/etc/ssh/sshd_config':
11    ensure => present,
12    source => 'puppet:///modules/sshd/sshd_config',
13    require => Package['openssh-server'],[]
14  }
15 }
```

Dry run

```
puppet apply sshd/tests/init.pp --noop --graph
```

Regenerate graph

```
dot -Tpng /var/opt/lib/pe-puppet/state/graphs/relationships.dot -o
/var/www/html/questguide/relationships.png
```



Relationships

Puppet uses another pair of metaparameters to manage this special relationship between a service and its configuration file: `notify` and `subscribe`. The

`notify` and `subscribe` metaparameters establish the same dependency relationships as `before` and `require`, respectively, and also trigger a refresh whenever Puppet makes a change to the dependency.

Make sshd service subscribe to the config file so it uses it before running the service.

```
1 class sshd{
2   package { 'openssh-server':
3     ensure => present,
4     before => Service['sshd'],
5   }
6   service { 'sshd':
7     ensure    => running,
8     enable    => true,
9     subscribe => File['/etc/ssh/sshd_config'],
10  }
11  file { ['/etc/ssh/sshd_config':
12    ensure => present,
13    source => 'puppet:///modules/sshd/sshd_config',
14    require => Package['openssh-server'],
15  }
16 }
```

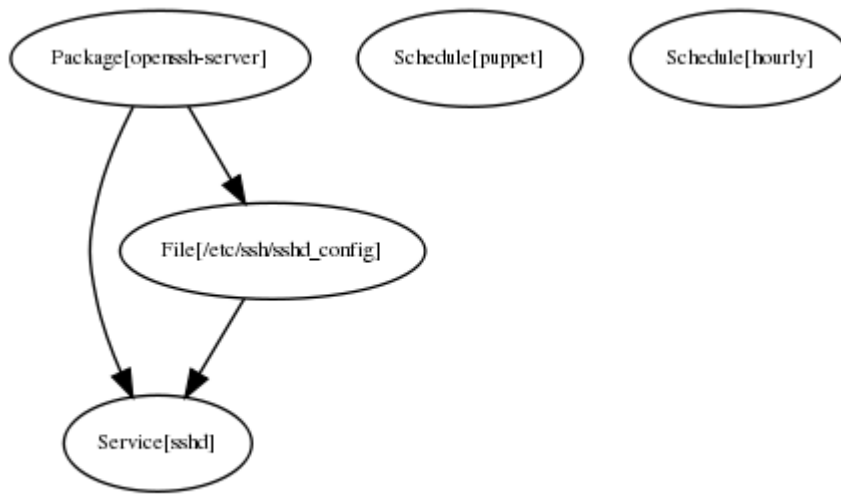
Parse and dry run

```
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet parser validate sshd/manifests/init.pp
[root@learning /etc/puppetlabs/puppet/environments/production/modules]# puppet apply --noop --graph sshd/tests/init.pp
Notice: Compiled catalog for learning.puppetlabs.vm in environment production in 0.46 seconds
Notice: /Stage[main]/Sshd/File[/etc/ssh/sshd_config]/content: current_value {md5}95f289e1ad7e3e8df4600beb48355a01, should be {md5}e9e4aa96ef2a893fc29f8a999dcb352a (noop)
Notice: /Stage[main]/Sshd/Service[sshd]: Would have triggered 'refresh' from 1 events
Notice: Class[Sshd]: Would have triggered 'refresh' from 2 events
Notice: Stage[main]: Would have triggered 'refresh' from 1 events
Notice: Finished catalog run in 4.63 seconds
```

Make graph again

```
dot -Tpng /var/opt/lib/pe-puppet/state/graphs/relationships.dot -o /var/www/html/questguide/relationships.png
```

```
(root@learning /etc/puppetlabs/puppet/environments/production/modules) # dot -Tpng /var/opt/lib/pe-pu
puppet/state/graphs/relationships.dot -o /var/www/html/questguide/relationships.png
(root@learning /etc/puppetlabs/puppet/environments/production/modules) # []
```



Relationships

Chaining arrows and Autorequires

Chaining arrows provide another means for creating relationships between resources or groups of resources. The appropriate occasions for using chaining arrows involve concepts beyond the scope of this quest, but for the sake of completeness, we'll give a brief overview.

The `->` (ordering arrow) operator causes the resource to the left to be applied before the resource to the right.

The `~>` (notification arrow) operator causes the resource on the left to be applied before the resource on the right, and sends a refresh event to the resource on the right if the left resource changes.

Autorequires are relationships between resources that Puppet can figure out for itself. For instance, Puppet knows that a file resource should always come after a parent directory that contains it, and that a user resource should always be managed after the primary group it belongs to has been created.

```

**Autorequires:** If Puppet is managing the user's primary group (as
provided in the 'gid' attribute), the user resource will autorequire
that group. If Puppet is managing any role accounts corresponding to the
user's roles, the user resource will autorequire those role accounts.

```