2014

Linux Academy.com

Puppet
Study Sheet

Puppet Ports
- 3000 - Web-based installer of puppet master
- 8140 puppet requests
- 61613 - orchestration requests
- 443 for console managemet
- 5432 for split installs if PuppetDB is running on a different system

puppet-enterprise-installer
- -a read answers form a file and quit with error if an answer is missing
- -A read answers from a file and prompt for input if an answer is missing
- -D display debugging information
- -h display help
- -l log commands and results to file
- -n run in noop mode (show commands that would have been run during installing without running them)
- -q run in quite mode; the installation process is not displayed. Requires answer file.
- -s save answers to file and quit without installing
- -V display very verbose debugging information
- Answer file location after initial creation: **/opt/puppet/share/installer/answers**

Use puppet enterprise uninstall

Installation Directories
- Puppet enterprise **/etc/puppetlabs/puppet**
- Puppet opensource /etc/puppet

Code and data directories
- modules - main directory for puppet modules (applies to master only)
- manifests - contains the main starting point for catalog compilation (applies to master only)
- environments - environments contains alternate versions of the modules/manifests directors, to allow code changes to be tested on smaller sets of nodes before entering production (master only)
- ssl - contains each nodes certificate infrastructure (all nodes)

Puppet Enterprise Master Components
- Puppet
- PuppetDB: PuppetDB contains
  - The most recent facts from every node
  - The most recent catalog for every node
  - fourteen days (configurable) of event reports for every node (an optional, configurable setting)
- Facter
  - -p
- Mcollective

- Hiera
- Puppet Dashboard
- PostgresSQL
- Ruby
- Passenger
- Java
- OpenSSL
- Augeas

Facts

- **external fact:** External facts provide a way to use arbitrary executables or scripts as facts, in other words you can write custom facts with a coding language, an external fact essentially executes an executable such as c/bash/perl/python script to determine fact value
- A ruby fact executes a command on the system.

license file - /etc/puppetlabs/license.key

Log Files:

- Puppet master is logged in the syslog
- Apache/Passenger
  - /var/log/pe-httpd/
- ActiveMQ Logs
  - /var/log/pe-activemq
  - /var/opt/puppet
- Orchestration
  - /var/log/pe-mcollective/mcollective.log
- Console logs
  - /var/log/pe-console-auth
  - /var/log/pe-puppet-dashboard
- Installer logs
  - /var/log/pe-installer
  - /var/log/pe-postgresql

Services

- pe-activemq - ActiveMQ messages server used to pass messages to collective servers on agent nodes. Runs on puppet master only
- pe-httpd - Apache2 serves puppet master and console (uses passenger to run)
- pe-mcollective - Orchectration engine listesns for ochestartion messages and invokes actions (runs on every agent)
- pe-memcached - runs only when PE console is installed
- pe-puppet -Runs on every node including puppet master
- pe-puppet-dashboard-workers - manages consoles background processes runs on servers with console (important for troubleshooting)

Download and install puppet agent (from the puppet enterprise)

curl -k https://<master hostname>:8140/packages/current/install.bash | sudo bash

Puppet master handles

- authenticating agent connection (through signed certs)
- puppet cert —help ***note: puppet cert must be run as the root user or with sudo or will not show any nodes (troubleshooting)***
  - command line
    - puppet cert list (view outstanding requests)
    - puppet cert sign <cert name>
    - puppet cert sign —all
  - Removing certs
    - puppet cert clean: Revoke a hosts' certificate and remove all files related to that host from puppet cert's storage
    - pupet cert revoke: Certificate is revoked by adding it to the Certificate Revocation list. Puppet master needs to be restarted after revoking certificates
  - Viewing all certs signed
    - puppet cert list -a
- processing posted reports
- serving files
- server a compiled catalog to the agent

Puppet agent - Run without any flags it will go into the background, attempt to get a signed certificate, and retrieve and apply it's configuration every 30 minutes
https://docs.puppetlabs.com/guides/install_puppet/post_install.html#configure-a-puppet-agent-node
https://docs.puppetlabs.com/references/3.6.2/man/agent.html

- Flags:
  - **—test**
    - **—no-daemonize**
    - **—verbose**
    - **—onetime**
    - **—onetime**
    - **—ignorecahce**
    - **—no-usecacheonfailure**
    - **—detailed-exitcodes**
    - **—show_diff**
    - **—no-display**
  - **—noop**
  - **—debug**
  - **—tags**
  - **—environment**
  - **—configprint**
  - **—genconfig**
  - **—server (specify puppet master server to connect too on command line)**

- Concepts
  - Default node name for a new agent (hostname?)
- Puppet agent install (add rpm node)
- Puppet agent post install
- puppet agent —waitforcert [sec]
- What is an agents node name by default? (host name) when the puppet agent first requests signing.
- What is included in a puppet agent run report? https://docs.puppetlabs.com/guides/reporting.html

Puppet Conf
[main]
[master]
[agent]
Section headers in a puppet conf file, each corresponding to a **run mode.**
Options are resolved in the following order command line > run mode > main > puppet defaults, meaning agent run mode will override the main renamed.

/etc/puppetlabs/puppet.conf - enterprise config file
/etc/puppet/puppet.conf - opensource config file

Agent Settings basic/runbehavior/service
Basic Settings
- **vardir**: location where puppet stores growing information
- **rundir**: location where puppet PID files are stored
- **ssldir**: location where SSL certs are stored
- **ca_server**: the server to use for certificate authority requests
- **certname**: the certificate name to use when communicating with the master
- **server**: the hostname of the puppet master
- **Environment:** Defaults to production, is the environment to request but can be overridden by masters ENC (External Node Classifier)

Run Behavior settings
- **noop -** The agent won't do any work it will only simulate changes that should be made and report to the master about what should have been done
- **priority -** Allows you to set the nice of the puppet agent in order not to starve other important processes while applying the catalog
- **report -** defaults to true and determines if it should send reports
- **tags -** Limit the puppet run to include only resources with certain tags (cool), specific data centers, etc
- **usecacheonfailure -** determines if it should fall back to the last known good catalog if the master fails to send a good catalog. (defaults to true)
- **ignoreschedules -** schedules allow you to only execute a resource if it's during a specific time period this setting can disable that feature might be used when you are doing an initial setup on a node and everything needs to be executed or enforced the first time around
- **prerun_command -** command to run before puppet command executes

- **postrun_command -** command to run after puppet command execute

Service behavior
- **runinterval -** how often the puppet agent daemon runs
- **waitforcert -** keep trying to run puppet agent if the certificate is not initially available (gives time for the master to sign)

Idempotence: is the property of certain operations in mathematics and computer science, that can be applied multiple times with changing the result beyond the initial application.
- **Catalog can be applied multiple times without causing issue**

Puppet master common settings

Basic Settings
- **always_cache_features**
- **dns_alt_names -** A list of hostnames the server is allowed to use when acting as the puppet master. The hostname that an agent uses must be included this list or the agent will fail connecting to master. The hostname can also live in the certname setting
- **environmentpath - (enable directory environments)**
- **basemodulepath - Directories continain puppet modules that can be used within all environments**
- **manifest - If environments are not enabled then defaults to site.pp, used as main entry point for compiling catalogs**
- **reports - Which report handlers to use**
  - **http** - send reports via http/https in via post to the reporturl address in the puppet.conf file. The body of the report is a YAML dump
  - **log** - Send all received logs to the local log destinations, usually the log is the syslog
  - **rrdgraph** - Graph all data using RRD library
  - **store** - Store yaml report on disk each node sends report as a YAMl dump and just stores the file in the reportdir (setting) directory
  - **tagmail** - send specific reports to specific emails based on tags in the log messages (requires tagmail.conf)

Commands
- Puppet describe will provide information about resource types within puppet
  - puppet describe -l (list all resource types available)
  - puppet describe -s <type> (short information about resource type)
  - puppet describe <type> (long listing information about resource)
- Puppet resource will describe information about resources already installed on a running node
  - puppet resource <type>
  - puppet resource <type> <name>
- puppet agent
  - Puppet agent will send a report to the puppet master with all facts and information about the node, this is the node object
  - Puppet agent will ensure that the agent's private key file is present
  - A puppet agent run is started from the **node being managed**

    - puppet agent will enforced a retrieved catalog from the puppet master

Common Resource Types:
- **exec -** execute a command on the linux system, is not idempotent by default
    - Add idempotent with attributes like
        - creates - Exec will only execute if the file it creates does not currently exist
        - onlyif - Exec command will execute onlyif the command associated with the onlyif attribute is zero
        - unless - Exec command will run unless the command associated with the unless attribute has an exit code of zero
        - refreshonly - Only allows the resource to receive refresh events

Running Multiple Puppet Masters
- **In order to run multiple Puppet masters in an environment you will need to specify all the potential hostnames**
- dns_alt_names must contain all the potential dns names for the puppet master in order for the certificates to be generated correctly for puppet agents.

Live Management
- Live management can be disabled by setting the disable_live_management setting in /etc/puppetlabs/puppet-dashboard/settings.yml
- /etc/init.d/pe-httpd restart to apply changes
- nodes are listd by the same puppet certificate names used int he rest of the console interface
- Willd cards for node usage ? matches one character * matches many or zero characters
- Can search by value of any fact on a node
- Live management helps provide viisbility to the state of reosources on all nodes and quickly filter and browser to find information on based on nodes. Vacation reports can be generated with live management.
- Live management provides vacation reports that can be generated on the fly
- Live management provides instant visibility into the state of the resource on all the nodes
- Live management provides the ability to quickly filter and browse to find the information you need

Module and Class Basics
puppet module generate - will generate common files/directories for a module
Each module will contain the following directories:
- files - Files to be copied to the node will be stored here
- manifests - Classes created with the puppet DSL stored here
- templates - Templates used to build files stored here
- tests - Tests to instantiate classes stored here

Manifests - All files end with .pp
- Each module must have an init.pp and contains a class or the high level definition of the module

Autoloading:
Class modules should be placed in the same directory that modelpath specifies in the puppet.conf
If modules are placed within a directory that is searchable by puppet.con then it is "auto discovered" if it's files end in .pp

include

- **If you include a class do you have access to it's variables? Or is that only for inheritance?**
  - **No if you include a class it does not give you access to that classes local variables. Inheritance would be required**

class

- **when/how/why you have to use this specific one**

Class names map directly to where Puppet expects to find them
localusers::groups::manage - would be found in
/etc/puppetlabs/puppet/modules/localusers/manifests/groups/manage.pp
firstsegment::secondsegment::third segment
Modules default class is located in manifests/init.pp file and has to be the same name as the module itself.

Puppet parser validate

- puppet parser will not error if two resource types have the same title

Types of errors puppet parser will catch

-

Class/variable/module naming conventions

- Reserved class/module names you cannot use
  - settings
  - main
- Class names, defined type names, and custom types can include the following:
  - **must begin with lower case letter**
  - lowercase letters
  - Numbers
  - Underscores (cannot begin a class name, or function name with an underscore)
-

Variable naming convention

- Variable names can include the following
  - uppercase and lowercase letters
  - Variable names should not start with uppercase letters in puppet 3.7
  - Numbers
  - Do not use dashes
  - Underscores (cannot begin a class name, or function name with an underscore)

Module names

- **must being with a lowercase letter**
- **can include**

◦ **Lowercase letters**
◦ **Numbers**
◦ **underscores** (cannot begin a class name, or function name with an underscore)

P.99

Matching

A given node will only get the contents of one node definition, even if two node statements could match a node's name. Puppet will do the following checks in order when deciding which definition to use:

1. If there is a node definition with the node's exact name, Puppet will use it.

2. If there is at least one regular expression node statement that matches the node's whole name, Puppet will use the first one it finds.

3. If the node's name looks like a fully qualified domain name (i.e. multiple period-separated groups of letters, numbers, underscores and dashes), Puppet will chop off the final group and start again at step 1. (That is, if a definition forwww01.example.comisn't found, Puppet will look

for a definition matchingwww01.example.) 4. Puppet will use thedefaultnode.

Thus, for the nodewww01.example.com, Puppet would try the following, in order:

www01.example.com

The first regex matchingwww01.example.comPuppet 3.6 Reference Manual • Language: Node Definitions

••

Merging With ENC Data

Node definitions andexternal node classifierscan co-exist. Puppet merges their data as follows: Variables from an ENC are set attop scopeand can thus be overridden by variables in a node

definition.

Classes from an ENC are declared atnode scope, which means they will be affected by any variables set in the node definition.

Although ENCs and node definitions can work together, we recommend that most users pick one or the other.

Exported resources

Classes declared in the site.pp standard node definition will be merged with classes defined in an enc
https://docs.puppetlabs.com/guides/external_nodes.html

Exported Resources
- Exported resources require catalog storage and searching enabled on a node master. This requires PuppetDB
- https://docs.puppetlabs.com/puppet/latest/reference/lang_exported.html

Auto loading
When the class is defined or declared puppet ill use it's full name to find the class or defined type in our modules directory.

Publishing modules to Puppet forge
Module Requirements:
- **metadata.json**
- **Uploadable tarbal of the module**
- Modulefile

https://docs.puppetlabs.com/puppet/latest/reference/modules_publishing.html

Puppet Resource Dependencies/requirements (metaparameters)
- require: Causes a resource to be applied after the target resource
- before: causes a resource to be applied before the target resource (implies before)
- subscribe: Causes a resource to be applied after the target resource. The subscribing resource will refresh if the target resource changes (implies require)
- notify: Causes a resource to be applied before the target resource. The target resource will refresh if the notifying resource changes

arrays (and arrays in titles)
- arrays can differ in title
- they can be referred to individually
- they are treated as individual resources

Additional Resource metaparameters
- schedule
- alias

Manage Modules With Puppet Module
- **puppet module**
  - puppet subcommand similar to a distribution package manager. It can be used for creating model skeletons, installing modules from the puppet forge, and searching the puppet forge.
  - **list -** list installed modules
  - **uninstall -** uninstall a puppet module
  - **install**
    - —version specifies which version of the module you would like to install
    - —modulepath specify which module path to install to
    - —environment option to install in different environment
    - —force forcibly install a module
    - —debug see additional information about what is happening
    - — ignore—dependencies skip installing any modules required by this module
  - **upgrade -** upgrade a puppet module
  - **search -** search the puppet forge for a module
  - **build -** build a module release package
  - **generate -** generate a boilerplate for a new module
  - **changes** - show modified files of an installed module

Classes
- Classes are singletons and will only ever be defined once.
- A compiled catalog will only contain one single instance of a class (singleton)
- **Smoke test**
  - Apply the class one time locally to validate the code, view temporary changes, and is managed by the puppet apply command.

What happens if different resource types have the same name? - No compilation error
What happens if the same resource type has the same name? - Compilation error due to duplicate resource types

Node classification
- A node should only be assigned a single “role"
- The node classification should expire NO implementation details

- Roles should be named business role of the node (web server, dbserver, etc)
- Profiles should be named after the technology the stack they configure. Database, web server, etc)
- Classification takes place in the /etc/puppetlabs/puppet/manifests/site.pp file
  - Location of the site.pp file can be changed by the **manifest** setting in puppet.conf
  - By default an agents node name is it's cert name. the crert name is usually the fqdn when the puppet agent is run run to create the cert name. When it comes to managing in site.pp the node name is the cert name.
- A role is multiple classes declared together in a node definition. It is best practice to actually use profiles/roles to create abstraction between the node classifications and the classes being declared
- **Console classification**
  - Classes available to be assigned to nodes in the console must be specifically added to the console. The console allows for easy management of classes by JR level employees.
  -

PuppetDB - https://docs.puppetlabs.com/puppetdb/latest/
- PuppetDB stores
  - The most recent facts presented from every node
  - The most recent compiled catalog for every node
  - Fourteen days (configurable) of event reports for every node (this is an optional setting)
- Benefits of PuppetDB?
    - Third party programs or even you can query information about your environment. Can query nodes with specific facts and or configurations (resource types assigned) for each node. This is available because the most recent compiled catalog and facts are stored from the last Puppet agent run from the node and is stored in the PuppetDB.
    - Event log

Hiera
- **https://docs.puppetlabs.com/hiera/1/**
- Is not configured on the agent but the Puppet master
- It is an external data lookup tool based on key:value data storage
- Allows the setting of values on a per node basis or on all nodes (node specific data)
- The concept is you retrieve configurations form hiera rather than hardcoding it. It creates a central location for all configurable data rather than having to find classes and configure the classes.
- **Hiera Lookup Functions: best practice is to use lookup functions only in manifests and assign the value to a local variable**
  - **hiera** - standard priority lookup, gets the most specific value for a given key, can retrieve values of any type (array/string/hashes)
  - **hiera_array** - Gets all the array or string of given values in the hierarchy for a given

get, then flattens them into a single array of unique values.
- ◦ **hiera_hash -** expects every value in the hierarchy for a given key to be a hash and merges the top-level keys in each hash into a single hash.
- Heira_include() function
- **Automatic parameter lookup**
- **test hiera on command line tool**
- **test heira)_array lookup and what it does in examples**

Event Inspector
- **https://docs.puppetlabs.com/pe/latest/console_event-inspector.html**
- **P**rovides data for investigating the *current* state of the infrastructure
- Provides insight as to *how Puppet* is managing configurations and *what* is happening *where* when an event occurs
- Lets you **monitor** and **analyze** the details about Puppet agent runs throughout your environment. This is a snapshot overview that will provide over view events
- What is an event?
  - ◦ An event is Puppet enterprises attempt at modifying a resource or specific property of a given resource. If the current state of the node does not match the desired state (during an agent run) then it will attempt some action to put the node into configuration. Any event or action that occurs is then sent back to the Puppet master at the end of the convergence (agent run).
- Types of events
  - ◦ Change: property was out of sync and Puppet made a change to enforce the desired configuration and put the property in its desired state
  - ◦ Failure: a property was out of sync, when Puppet attempt to make changes, the changes failed.
  - ◦ No-op: property was out of sync but Puppet was instructed not to make changes to the property
  - ◦ Skip: a requirement for this resource was not met. Thus Puppet does not compare its current state to the desired state. This will occur when the schedule metaparemter is set and the time interval does not meet the requirements or it is a general failure on the resources dependencies)
- Types of event inspector views, when looking at views we can potentially find root causes of issues faster. The view you look at depends on what happened and where it happened. Knowing how to search to find errors quickly is important and is why the inspector views are provided. **(perspectives)**
  - ◦ Classes - If a service fails to start this might be the first place you look for reported events
  - ◦ Nodes - Geographic outage, might start looking at the nodes events first
  - ◦ Resources
- Event inspector does not refresh the page automatically. It will fetch the data once during loaded and uses the same data until reloaded
- Extra considerations
  - ◦ If a Puppet agent runs and during the Puppet agent run it restarts PuppetDb then the

event log from that run will not be available because PuppetDB is not available to store the events during the run.
    - Time out of sync can cause inspector issues (as well as general issues in your environment)

Troubleshooting - https://docs.puppetlabs.com/pe/latest/trouble_install.html | https://docs.puppetlabs.com/guides/troubleshooting.html

- Puppet agent run errors
    - *certificates were not trusted* ***error:*** issue is the previous cert with a node name of the same name probably has not been clear
        - puppet cert —clean {node cert name}
        - rm -r (/etc/puppet/ssl; rm -r /var/lib/puppet/ssl)
    - *hostname was not match with the server certificate:* issue is a node is attempting to contact a master with a different name than the masters node name when the certificate was originally generated
        - To solve we need to first figure out what the puppet masters certfied hostname is
            - puppet master —configprint certname (perform on the puppet master server)
        - Then modify the agent nodes settings to point to one of the masters certified hostnames
        -
- Installer issues
    - Check DNS (hostname DNS and FQDN are important)
    - Check permissions/ports (make sure the proper ports are open)
    - Have you attempted to install any other Puppet software BEFORE the puppet master install?
    - To recover from a failed install first make sure you have all the configuration settings correct and then run the puppet-enterprise-uninstaller in the puppet directory (same location as the puppet-enterprise-installer)
- Agent Issues
    - Is the DNS configured on the agents to the puppet master? Is the server setting in puppet.conf file reachable?
    - Have you signed the agents certificate on the puppet master? (Puppet cert list to view waiting certs and puppet cert list —all to view alls signed certs)

Common Maintenance Issues - https://docs.puppetlabs.com/pe/latest/maintain_console-db.html#pruning-the-console-database-with-a-cron-job

- What happens in the Puppet enterprise console becomes slow and is hogging a significant amount of disk space?
    - Restart console background tasks
        - service **pe-puppet-dashboard-workers restart** (depending on distribution but the service name is pe-puppet-dashboard-workers)
    - Clean up old agent node reports that have built up over time
        - **reports:prune** rake command (will display additional params if none are

given)
    - sudo /opt/puppet/bin/rake \-f /opt/puppet/share/puppet-dashboard/Rakefile \RAILS_ENV=production \reports:prune upto=1 unit=mon
  - Optimize the database
    - ensure autovacuum=on is set, if the DB is growing to large then this setting might be off for some reason

Puppet Agent Run Report

- How do you access a run report and what is included in it?
- **Logs the following**
  - **By default the logs are stored by the Puppet master in the reportdir (puppet.conf setting)**
  - Every log message generated during the transaction (sends all client logs to the Puppet master)
  - Keeps track of how long each resource type took to configure
  - Logs
    - total - Number of recourse being managed
    - skipped - number of resources skipped due to tagging or schedule metaparameter
    - scheduled - how many resources were within the scheduling metaparameter
    - applied - how many resources were attempted to be fixed
    - out of sync - how many resources were out of sync
    - restarted - how many resources were restarted due to dependency changes (updated config file for example)
    - failed restarts - how many resources could not be restarted
    - total number of changes in the transaction

Custom Facts

- custom facts to have them synchronized using plugin would go into lib/facter

Puppet dashboard Maintenance

- Pruning console database with a cronjob
  - By default in 3.3 and later the cronjob is managed by the **puppetlabs-pe_console_prune** module. This module will prevent bloating of the console database by deleting old data that consists mainly of old puppet agent run reports after x number of days. The amount of days to prune can be tweaked in the parameters class of the module on the **prune_upto** parameter, and is set to 30 days by default.
- Fixing “increased number of tasks” (cause, background task processes may have died and left invalid PID files
  - Fix by restarting the **pe-puppet-dashboard-workers** service.
- Optimizing the console database
  - autovauum=on is on by default, if the database is growing to large it is possible that this setting has for some reason been changed to off
- Cleaning old reports
  - Use the **reports:prune** rake task if you which to delete oldest reports for performance, storage, or policy reasons this is run as the cron in the pe_console_prune class but

can be run manually

    - /opt/puppet/bin/rake \ - f /opt/puppet/share/puppet-dashboard/Rakefile \ RAILS_ENV=production \ reports:prune upto=1 unit=mon

- Backing up the console databsae
  - pe-postgres -s /bin/bash
    - pg_dump pe-puppetdb -f /tmp/pe-puppetdb.backup --create
- Forgot the console username/password?
  - stop pe-httpd
  - alter user console password 'new password'; (inside of PostgreSQL administration tool)
  - Start pe-httpd

Troubleshooting

https://docs.puppetlabs.com/guides/troubleshooting.html

- Recover from a failed puppet install
  - Use the uninstaller script **pe-enterprise-uninstaller**
    - -p: Remove (purge) all configuration files, modules, manifests, certificates, andhome directories of any users created by the PE installer Will also remove GPG key
    - -d: Remove any database created during installation
    - -h: Help
    - -n: Run in noop mode
    - -y: Default yes to all questions to confirm uninstallation
    - -s: Save an answer file and quit without uninstalled
    - -a: read answers from a fail and fail if answer is missing
    - -A: read answers from file and prompt if an answer is missing
- If installing the console and puppet master on separate servers ensure that you install the Puppet master before the console or the installation process might fail
- The Puppet master can log additional debug-level messages about the time it takes for each step of the catalog compilation.
  - In order to do this set the profile setting to true in the [agent] section of the puppet.conf (on the agent). Optionally you can also run —profile if running the puppet agent manually

Deactivating Puppet Enterprise Agent Nodes

1. Stop the agent service on the node you want to stop
2. On the master deactivate the node by running puppet node deactivate <node name>
3. revoke the certificate puppet cert clean <node name>
4. Complete the certificate revocation by restarting puppet service pe-puppetserver restart
5. Delete the node on the console
6. Stop the mcollective service on the node
7. remove certificates from the node rm -rf /etc/puppetlabs/mcollective/ssl/clients

Certificates and managing certificates

- Removing the Puppet certificate authority certificate (puppet master)
  - **rm -rf $(puppet master —configprint ssldir)**
- Removing a nodes certificate from the Puppet master for regeneration
  - **puppet cert clean <certname>**
- Removing the certificates on the Puppet Agent for regeneration
- Error message err: Cloud not retrieve catalog from remote server: SSL_connect, could be due to the node attempting to connect to a puppet master with a hostname not in the certificate
  - To see what hostnames a node/agent is configured to communicate with perform the command:
  - **puppet agent —configprint server**
- Having a previous node with the same cert name
  - Clean the certificate on the Puppet master **puppet cert clean <certname>**
  - Remove the ssl configuration on the node attempting to connect: **rm -r $(puppet agent —configprint slider)**

Puppet environments

Steps for configuration environments (directory)

1. Configure environments
2. Create the directory structure for environments
3. Assign nodes to the environments

Configure Environments

- Nodes can be assigned to environments using the agent's config file or an ENI (external node classifier)
- $environment variable can be used in manifests to get the current environment
- basemodulepath = setting in the [main] section of puppet.conf that states the base modules for all environments
  - basemodulepath must always include the /opt/puppet/share/puppet/modules directory
- Directory environments will replace config environments and they are mutually exclusive (both of them cannot be enabled)
- puppet.conf configuration for directory environments
  - environmentpath
  - default_manifest
  - basemodulepath
- Restart the puppet master after puppet.conf files have been changed

Creating Directory Environments

- Directory name is the environment name
- Must be located on the Puppet master in the environment path setting directory i.e $confdir/environments

- Should contain a {**modules,manifests**} directory
- Puppet enterprise has two additional requirements than Puppet opensource
  - Filebucket resource must exist in the Main manifest
    - filebucket {'main': server => '<server>' path => false,}
    - File {backup => 'main' }
  - A default manifest can be set so that directory environments do not require one.
    default_manifest = $confdirt/manifests in puppet.conf
- An environment can contain an environment.conf file in the environment directory and settings in the environment.conf file will override settings in the puppet.conf

Puppet only uses global hiera.yaml is only global and cannot be configured on a per environment basis - need more details

Troubleshooting environments

- if a node is assigned to a non-existent environment the catalog compilation for the node will fail
- Environments cannot contain the following names {main,master,agent,user}

Running Multiple Puppet Masters

- Can install multiple Puppet masters but the ca and certificates need to have each available DNS name configured for each possible Puppet master.
- Configure in the [main] section of puppet.conf on the puppet master and agent node
- If the agent or master has already run and created a certificate then you must remove the certificates and regenerate the certificate to include the new hostnames
  - **rm -rf $(puppet master —configprint ssldir)**

Conditional statements

- Selectors are used for returning plain-text values
- Selectors can be used to set attributes or titles
- Selectors REQUIRE a default match, without one the compile will fail
- Case statements are used for evaluating a block of code
- Case statements do not require a default match but it is best practice
- Selectors cannot be used in the case section of a case statement

Functions

- **Run only once during compilation**
- **Cannot be used to make conditional decisions during catalog enforcement on the agent**
- **Functions execute on the Puppet master, they do not execute on the agent**
- **Statements**
  - statements are functions that take action without returning a value
- **rvalues**
  - rvalue functions will return a value

if$hostname**=~**/^www(\d+)\./{notice("Welcome to web server number $1")}

This example would capture any digits from a hostname like www01 and www02 and store them in the $1variable.

Resource Collectors (Spaceship operators):

- Collect a group of resources from within the catalog (catalog only it, cannot read the node but it can read resources that are defined anywhere within the catalog) by searching by the attributes of resources in the catalog.
- User <| title == "root" |> // Collect a resource that has the title of root, only one resource should be returned.
- User <| groups == "sudo"|> // *Will collect any user resource whose list of supplemental groups includes 'admin*'t
- Resource collectors can be used to add or modify existing attributes within a catalog.
- Examples:
  - Create a new resource dependency +> add a dependency (array basically)
  - Add an attribute like group name to a user
  - Modify attributes much like you would with inheritance best practice.

Automatic Parameter lookup

- If it was a resource-like declaration/assignment, Puppet will use any parameters that were explicitly set. These always win if they exist.
- **Puppet will automatically look up parameters in Hiera,** using <CLASS NAME>::<PARAMETER NAME> as the lookup key. (E.g. myclass::parameter_one in the example above.)
- If 1 and 2 didn't result in a value, Puppet will use the default value from the class's definition. (E.g."default text" in the example above.)
- If 1 through 3 didn't result in a value, fail compilation with an error.

Hiera: Sample Config File

Source: https://docs.puppetlabs.com/hiera/1/hierarchy.html

```
---
:backends:
    - yaml
    - json
:yaml:
    :datadir: /etc/puppet/hieradata
:json:
    :datadir: /etc/puppet/hieradata
:hierarchy:
    - "%{::clientcert}"
    - "%{::custom_location}"
    - common
```

Backends

- If multiple backends are specified (yaml/json) then hirea will first look through the yam backends for matching data sources then the json (it is in order from top down)

Hiera: Hierarchy

- Must be a **string** or an **array of strings**
- Each string is the name of a static or dynamic data source
    - data source is in the format of %{datasource} an interpolation variable the data source makes up the hierarchy and is what hiara looks at in order to populate the data
    - Hiera hierarchy data is configured in /etc/puppetlabs/hiera.yaml
    - The Hierarchy is an array (a single string is an array with a single item)
    - Each element under the hierarchy is treated as a data source
    - If a data source is specified by a puppet variable then it is **best practice** to use the fully qualified variable (use the scope operator)
- **Static data source** A hierarchy element that is not an interpolated variable. An exact string match such as “webserver” vs a variable to match the dynamic data on a node
- **Dynamic data source** $::clientcert then the data will be matched based off of client cert and a data source matching the client cert will need to be fond in the data source location
- **What if a node matches multiple hierarchy data sources?**
    - Each listing under the :hierarchy: is a data source
        - IF a data source in the hierarchy doesn’t exist then hiera will move on to the next data source
        - if a data source does exist but does not have the piece of data hiera is looking for then it will move onto the next data source.
        - If a value is found in the data source specified
            - In a normal heira() function lookup, Hiera will use the first data source found
            - An hiera_array() lookup Hiera will continue onto each data source and return all of the discovered values as a flattened array.
                - Values from higher data sources in the hierarchy will be the first elements in the array
            - hiera_hash() lookup will continue expecting every value to be a hash and throwing an error if a non-hash value is found. It will search the entire hierarchy and merge all the discovered hashes and return the results, it will then allow values from higher in the hierarchy to replace values from the lower.
- Hier will use the “common/default” data source that provides data if it goes through the hierarchy and cannot find any data. If no data is found it will fail with an error
- **Best practices**
    - Do not use non (local puppet variables) fact variables in a yaml file or as a data source.
    - Use scope operator to reference the facts
- **Automatic Parameter Lookup**

- **If a class has parameters then automatic parameter lookup will occur in the following order**
  - First any parameters passed using the class definition
  - Then if there are no parameters passed it will automatically look at hiara data, this means we can still use the **include** declaration and have parameters
  - Then if both 1 and 2 fail it will resort to defaults $parameter = $default)
  - If there is not a default then compiling will fail
- Automatic parameter lookup only works with **priority** (hiera()) lookup type and not hiera_array() or hiera_merge()

Hiera Lookups

- hiera_array() - (Array merge) Will return all possible matches from data sources. It will return the returned data from sources higher in the hierarchy first. If a hash is found the hiera_array() lookup will fail.
- Hiera() - primary lookup and it is a priority lookup. A priority lookup gets a value from the most specific matching level of the hierarchy and only one hierarchy is matched once a match is found it returns that data and stops searching.
- hiera_hash() - (hash merge) Grabs a matching value from every level in the hierarchy and retrieves alls of the hash values for a given key and then merges the hashes into a single hash. It will fail if any versions of data found return anything but a hash.
- What is a hash? A hash is a key/value pair surrounded by curly braces.
  - { key => 'value', ke1 => 'value' }

External Node Classifiers And Classifying nodes with the console

ENC is an executable that is called by the Puppet master that takes the name of the node and yaml document describing the node.
- Can be any program that has that above characteristic
- More commonly used is the console

A standard node definition is considered the site.pp file. An ENC and the site.pp standard node definition **can** co-exist.
What happens if a site.pp and ENC both classify a node?
- The classes are declared in each source are merged

node_terminus - Puppet.conf setting that tells Puppet to use an external node classifier. Can be set to exec or console,if this is set than the **external_nodes** needs to be set which has a path to the proper ENC executable (again mostly the console)

Compiling A Catalog With Site.pp (standard node definition) and ENCs Puppet will include

- Any classes specified in the node object it received from the node terminus
- Any classes or resources which are in the site manifest but outside any node definitions
- Any classes or resources in the most specific node definition in site.pp that matches the current node (if site.pp contains any node definitions)
  - if site.pp contains a node definition then there **has** to be a a node definition for every node this could be the default as well declared or else compiling will fail.
  - If the node name resembles a dot-separated fully qualified domain name, Pupept will make multiple attempts to match a node definition, removing the right-most part of the name each time. Puppet will try node1.mylabserver.com, then node1.mylabserver, then node1. However, this only applies to site.pp and not an ENC.
  - Puppet will try default {} if no matching node name is found
- Console can be used to set TOP-Scope variables on a node
- You can assign parameters to a class much like using the class {} function to declare parameters. However, if you declare inside the ENC and site.pp then an error will occur.
- *Note: if a node matches multiple node definitions due to regular expressions, puppet will use ONE of them with no guarantee as to which one it will use.*

Tip: you can tell puppet not to manage a given metaparameter by setting it do undef

Puppet Data Library

The puppet data library stores large amounts of data that puppet automatically collects about your infrastructure. Once the information is stored the data library has APIs that allow SysAdmins the ability to write scripts to interact with that data. Advantages include writing scripts for customized reporting.

The Puppet data library is made up of:

- **PuppetDB**
  - Stores up to date copies of every nodes facts, resource catalogs, and run reports from each puppet run.
- **Puppet Run Report Service**
  - Provides push access to reports that every node submits as part of each Puppet run. The node submits these reports after the agent run. Puppet run report service allows you to write a custom report processor and divert these reports to a custom service, which can use them to determine whether a puppet run was successful.
  - Puppet run report service also works with out-of-band report processors that consume the YAMl files written to disk by the puppet masters default report handler.
- **Puppet Resource Dependency Graph**

- Provides a complete mathematical graph of all the dependencies between resources under management by puppet.

Deactivating A Puppet Enterprise Agent Node

1. Stop the Puppet Agent service
2. On the puppet master run **puppet node deactivate <nodename>** (this command will deactivate the agent node in puppet db
3. Run puppet cert clean <node name> on the puppet master
4. After the puppet cert has been cleaned run a pe-puppetserver restart this prevents the node from checking in again
5. Delete the node from the console
6. On the node, in order to stop mcollective you must uninstall the puppet agent or destroy the node
7. On the node if not destroyed then remove the certificates from /etc/puppetlabs/mcollective/ssl/clients for mcollective to not be accessible anymore
    1. service pe-mcollective stop

Common Console Tasks

- **Inventory Search**
    - Based off of reports from run reports only
    - Uses PuppetDB to search the data
    - Includes run reports and fact information about a node
- Live Management
- Reporting
- Node Classification
- Event Inspector
- Delete a node from the console

Things to add to the study guide

- The order of resoruces in a Puppet manifest does not matter because Puppet assumes that most resources are not related to each other and will manage the resources in whatever order is most efficient.
- if a resource receives multiple refresh events, they will be combined and the resource will only refresh once
- Puppet resource names are case sensitive
- puppet resource file /etc/motd/ ensure=absent

Resource ordering: https://docs.puppetlabs.com/learning/ordering.html
https://docs.puppetlabs.com/references/3.4.3/man/ca.html

Note: Some of the notes were taken directly form puppetlabs.com online documentation.