Contents

Directory structure

The directory structure of a COI install will look like this:

```
/etc/puppet
/etc/puppet/files
/etc/puppet/manifests
/etc/puppet/modules
/etc/puppet/templates
```

```
/etc/puppet
```

Is created when puppet is installed. It should essentially contain this on the appropriate branch (usually multi-node). This repo contains the manifests and templates directories where the one part of COI will reside.

`/etc/puppet/manifests`

Contains the `site.pp`, which is used to customise the install in common ways, such as configuring the network settings and defining which nodes to manage. Site.pp is internally documented and will be different for each site. This directory also has `core.pp` and `cobbler_node.pp`: `core.pp` is used to provide a clean interface between the Openstack puppet modules and the user-facing `site.pp`; `cobbler_node.pp` is specifically targeted at managing the cobbler module. `site.pp` imports core and `cobbler_node` The scripts in the directory are helpers that perform the following functions:

- `clean_node.sh` Is used to wipe the puppet cert of a node and set it to install on next boot. Usage: `clean_node.sh $target_node`
- `puppet-modules.sh` Is used to install all the modules in modules.list via apt. Usage `puppet-modules.sh`
- `reset_build_node.sh` Will clean up the build node so that a subsequent puppet apply will test with a (roughly) clean slate. Purges installed packages, removes var directories and some config files. Usage: `reset_build_node.sh`
- `reset_nodes.sh` Removes all nodes from cobbler db, then runs puppet to insert them again and then runs `clean_node.sh` on all nodes . Usage: `reset_nodes.sh`

`/etc/puppet/templates`

Contains a template for `/etc/network/interfaces`. This can be modified by sites that have complex networks to meet their requirements. It is applied via a late command and not directly managed by puppet. The second file in this directory can be used to move IP addresses between physical nics and Openvswitch ports and is not needed in the majority of cases. It is controlled by `numbered_vs_port` in site.pp.

`/etc/puppet/modules` or `/usr/share/puppet/modules`

Either of these locations can be used to house the puppet modules. The COI apt packages will install them to `/usr/share/puppet/modules`.

```
apache          cobbler   concat    drbd      horizon   memcached  naginator  openstack         pupp
apt             coe       corosync  glance    inifile   monit      nova       openstack_admin   quar
apt-cacher-ng   collectd  dnsmasq   graphite  keystone  mysql      ntp        pip               rabb
```

Understanding the install sequence

Because COI handles both provisioning of the base OS and deployment of applications the install sequence for a node is quite long. The install takes the following steps:

1. COI puppet modules and manifests are installed on the build node from either git or apt
2. `puppet apply` is run on the build node, which does the following:
    1. Cobbler is installed on the build node
    2. The Ubuntu 12.04 install image is loaded into Cobbler using `cobbler-import-ubuntu-x86_64`
    3. The out-of-band information for the target node is inserted into the Cobbler database
    4. A preseed file is created to automate the install, which has the following in the late command:
        1. Sets puppet to run after the node has booted, depending on whether `autostart_puppet` has been set

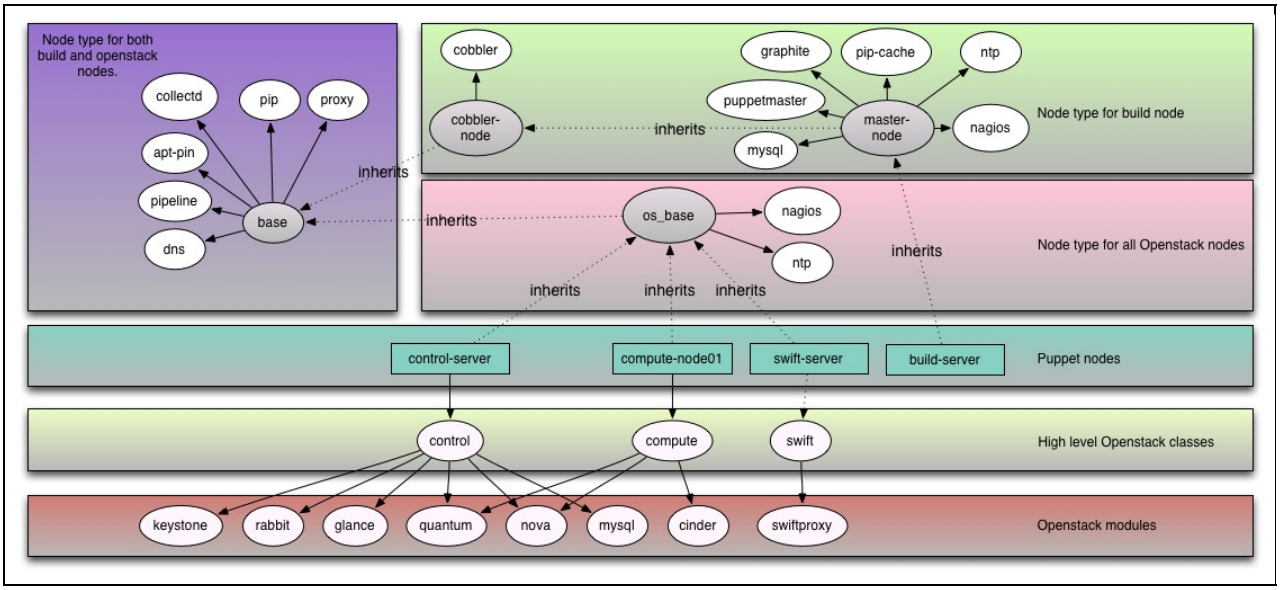
2. Sets the puppetmaster address and sync interval in `puppet.conf`
3. Syncs to the build node ntp server
4. Optionally disables IPv6 router advertisement
5. Optionally installs the ethernet bonding module
6. Sets `/etc/network/interfaces` based on `/etc/puppet/templates/interfaces.erb` in the late command
7. HTTP posts to cobbler on the build node to say that the install completed successfully
8. HTTP posts to cobbler on the build node to request no more install boots (so the machine will receive a PXE command to boot from local disk instead)

3. The target node is rebooted using the `clean_node.sh` script on the build node, or by hand, which will install ubuntu and run everything in step 2.4
4. The target node will finish the install, reboot, then boot into the newly installed OS
5. If `autostart_puppet` has been set, the node will run puppet agent, and install everything needed for either a control or a compute node.

Most of the complexity here is tied up in the late command. If you need to add a system module and want it to be available as soon as the system reboots then this is the place to put it. The easiest way to do this is by modifying `cobbler_node.pp` and adding lines to the late command there. This is an example of a general guideline: try to avoid modifying the puppet modules, and instead change things from the manifests folder where possible.

Modules

The following diagram shows the structure of COI. Node types are in grey, while the nodes themselves are in green. Classes/Modules are shown in white.



apache

[DEPRECATED] Manages the apache http daemon. Apache is used by Horizon, puppetmaster and Graphite among other things, but is handled by requiring the apache package and creating site-enabled entries instead of using the module.

apt

The base node as defined in `core.pp` defines an `apt::source` which contains the PGP key for Cisco's Openstack and puppet packages. This means every node has access to the Cisco apt repo.

apt-cacher-ng

This manages the apt-cacher-ng daemon which greatly accelerates the install process by eliminating the need for all nodes to install from the internet. The build node runs the apt cacher, which is defined in `core.pp` under `master-node`

cobbler

The cobbler module is used to install and maintain the core functionality of the build node: deploying servers. The cobbler module is configured via `cobbler-node.pp` in the manifests folder. The module itself is not very mature, and it is conceivable that an advanced developer may need to customise this module in order to change some part of the node install process. A good example would be installing 32 bit ubuntu instead of 64, which would require a different arch to be passed into the ubuntu class in the cobbler module, so that cobbler-ubuntu-import will bring in the correct install image. Cobbler also manages dhcp, dnsmasq, PXE, tftp and some http services. The module itself is quite barebones and should be easy to extend if needed.

coe

This is a very small module that adds a web page on the build server with links to other services such as Horizon, Nagios and Graphite.

collectd

Collectd is a metrics collection system. This module will install the collectd client, and point the client at the graphite server (on the build node).

concat

This is a puppet module for constructing files out of fragments. It is used by the glance and keystone modules.

corosync

[DEPRECATED] Corosync is used by the openstack_admin class to provide HA services to the controller.

dnsmasq

[DEPRECATED] Although dnsmasq is still used by cobbler and openstack, the dnsmasq module is not used.

drbd

[DEPRECATED] Used by openstack_admin to provide HA services on the control node.

glance

The openstack image registry. For more info on Glance, go here. Glance is one of the simplest pieces of an Openstack cloud. There is no support in this puppet module for managing what images are available, or for inserting images into the registry. The backend can be changed from the default file to swift for production deployments.

graphite

Graphite is a scalable real-time graphing system. It is included in the build node via `'master-node'` in `core.pp`. All collectd agents need to be aware of the graphite host location, so if you want to move graphite off the build node, update the collectd definition in the base node in `core.pp`.

horizon

Horizon is the django based web interface for an Openstack cloud. It runs on the control node and is included via `openstack::controller`. There is no mention of horizon in `core.pp` since it generally doesn't require any configuration as a very simple web app.

inifile

Used by Glance, Keystone and Quantum to easily create ini files.

keystone

Keystone is the openstack identity service. The keystone module contains providers/types for the contents of the keystone DB: users, roles, services, tenants and endpoints. The admin and service elements that are required for openstack to function are created in the `openstack::controller` class.

memcached

Memcached is instantiated by openstack::controller to act as a cache for Horizon.

monit

[DEPRECATED] Ignore.

mysql

Used by the puppet module to create a mysql server on the build node to enable the use of storeconfigs, and used by the openstack::controller class to create the my sql server required for Openstack.

naginator

The top level class naginator will install the nagios server, this is included in the node type `'master-node'`. There are then classes for the other types of node that will monitor the apprpriate things: `naginator::compute_target`, `naginator::control_target`, `naginator::swift_target`. There is a `naginator::base_target` that is included in the `base` node type that all nodes inherit from.

nova

Nova is the part of openstack responsible for VM management. There are two obvious pitfalls when working with this module: there are hundreds of potential flags to be passed into nova.conf, and nova is deeply tied into quantum via openvswitch, so care must be taken when modifying either. `nova.conf` configuration takes the following form:

```
nova_config { 'flag_name' : value 'flag_value' }
```

All of these are aggregated at runtime to create the `nova.conf` file.

ntp

Configures ntp such that the build server will sync with the list of servers given in `site.pp`, and the other nodes will sync with the build node. The `master-node<code> node type contains the former and the <code>os_base` node type contains the latter.

openstack

This is a high level module that contains classes that aggregate openstack services into useful classes such as `openstack::control` (containing nova-api/mysql/rabbitmq etc.) and `openstack::compute` (containing nova-compute/quantum-openvswitch-agent etc.)

COI creates classes called control and compute in core.pp, which wrap the `openstack::control` and `openstack::compute` classes in this module along with other classes. A user can then instantiate the wrapper classes from `site.pp` as shown in `site.pp.example`.

openstack_admin

[DEPRECATED] This module was previously used to create HA control classes.

pip

Pip, the python package manager. The class pip is instantiated in the `base` node so that all nodes in the cluster can install using pip. This class adds the pip provider so that a package can be set to install using pip instead of apt or yum.

The pip caching is not managed here, and is only set up if there is no default gateway configured. This is done in `core.pp` under `master-node`.

puppet

Installs the puppetmaster with mysql. This is done in `core.pp` under `master-node`. This module does not configure clients to contact the build node as their puppetmaster: this is done during the PXE install process.

quantum

Quantum is the Openstack networking service. As noted in the nova section, this module is coupled to the nova module, and care should be taken when modifying it. Currently, the module is very OpenVSwitch centric, although other plugins should be supported in the future. Quantum takes responsibility for creating its own keystone user, role, service and endpoint. Quantum services are instantiated using the

`quantum::agent::[dhcp|l3|ovs]` classes, as well as `quantum::server` which creates an api server. All of this is handled by the `openstack::control` and `openstack::compute` classes.

rabbitmq

RabbitMQ is an AMQP server that is used to pass messages between openstack services. Confusingly, this is actually created by nova, despite its use by other modules and services within the Openstack system. `rabbitmq::server` is instantiated by `modules/nova/manifests/rabbitmq.pp` in the `nova::rabbitmq class`, which will install rabbitmq with appropriate config for openstack. The `nova::rabbitmq` class is in turn instantiated by `openstack::controller`.

rsync

[DEPRECATED] This is no longer used, but is required for Swift support.

ssh

Used to set user SSH keys. This isn't used directly by COI but may be useful for creating admin accounts on machines with passwordless access.

stdlib

This is a standard library of resources for puppet. The documentation in the module is pretty extensive so I won't cover it here.

sysctl

This is used to set sysctl properties on nodes. Nova uses it to set `net.ipv4.ip_forward` on network nodes, though this won't affect COI installs since nova-network is not used.

vswitch

This is a module for managing vswitches. Each type of switch should implement a provider, though at the moment only openvswitch is supported. The quantum ovs plugin and agent both use this module to initialise openvswitch.

xinetd

[DEPRECATED] This is no longer used, but is required for Swift support.