Deploying your first app to Cisco Metacloud™ using Puppet

This tutorial will explain the process for using Puppet Enterprise to deploy an application (Apache web server) to a separate Centos instance (agent) running in Cisco Metacloud. This process is fairly straightforward and customers can use Puppet Enterprise to insert a layer of configuration management for applications running as instances in Metacloud. The assumption is that the reader should have a baseline understanding of application delivery and configuration management.

In order to provide a baseline understanding of the process to deploy applications in Cisco Metacloud using Puppet, we must also understand the basics of Puppet installation and integration with Metacloud. This tutorial will discuss the following topics:

- Introduction to Cisco Metacloud
- Introduction to Puppet Enterprise
- Installation of Puppet in Metacloud
- Puppet Agent Installation
- Basics of Puppet Manifest
- Deploying applications with Puppet

Introduction to Cisco Metacloud

Cisco Metacloud delivers a true public cloud experience for customers on their premises and behind their own firewalls. It offers full administrative control and multi-tenancy. It's a production-ready, OpenStack-based solution that Cisco engineers, deploys, upgrades, and remotely operates on the customer's behalf, 24 hours a day, 365 days a year. This managed OpenStack distribution allows our customers to deliver IT as a service to their own lines of business as if the IT department were their own public cloud provider. Metacloud is truly the OpenStack easy button!

The remainder of this tutorial will document the steps needed to implement and use Puppet to deploy an application in a running Metacloud virtual machine instance.

Introduction to Puppet Enterprise (PE)

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.

In addition to these integrated open source projects, PE has many of its own features, including a graphical web interface for analyzing reports and controlling your infrastructure, orchestration features to keep your applications running smoothly as you coordinate updates and maintenance, event inspection, role-based access control, certification management, and cloud provisioning tools.

The appeal of Puppet is that it allows you to describe all the details of a configuration in a way that abstracts away from operating system specifics, then manage those configurations on as many machines as you like. Puppet allows the customer to control their entire IT infrastructure (think hundreds or thousands of nodes) in a way that is simpler to maintain, understand, and audit than a collection of complicated scripts.

Puppet uses client server based architecture with the following main components:

- Puppet Master: This machine contains the configuration for different hosts. Puppet Master will run as a daemon on the master server.
- Puppet Agent: This is the daemon that will run on all the servers, which are to be managed using Puppet. Puppet Agent will go and ask the configuration for itself from the Puppet Master server at a specific time interval.

Here's what happens when the Puppet Agent is sent to fetch data from the Puppet Master server:

- When a client node connects to the master, the master server analyzes the configuration to be applied to the node, and how to apply the configs on the node.
- The Puppet Master server collects all the resources and configurations to be applied to the node, and compiles it to make a catalog. This catalog is given to the Puppet Agent of the node.

The Puppet Agent will apply the configuration on the node, according to the catalog, and then reply back, and submit the report of the configuration applied to the Puppet Master server.

The aforementioned steps will occur at a configurable time interval (default 30 minutes) or every time a client node reboots. Note, you can also manually ask Puppet Agent to go and fetch the configuration from the Puppet Master server whenever required.

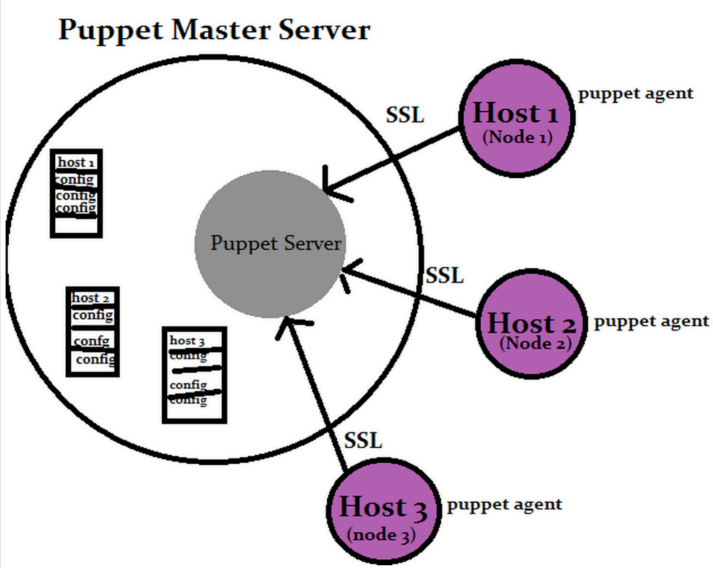

Figure 1. Puppet Client Server Architecture

As mentioned previously, Puppet doesn't use scripts. Instead Puppet uses a declarative language called DSL. This means that instead of defining a process or set of commands, Puppet code describes (or declares) only the desired end state, and relies on built-in providers to deal with implementation.

This DSL is implemented using Manifests and Classes. Manifests are files containing Puppet code. They are standard text files saved with the .pp extension. There is a global site based (site.pp) manifest that is applied to all nodes but most manifests should be arranged into modules for granularity.

The core of the Puppet language is declaring resources. A resource declaration looks like this:

```
# A resource declaration:
file { '/etc/passwd':
  ensure => file,
  owner  => 'root',
  group  => 'root',
  mode   => '0600',
}
```

Figure 2. Resource Declaration Example

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.

```
# A class with parameters
class apache (String $version = 'latest') {
  package {'httpd':
    ensure => $version, # Using the class parameter from above
    before => File['/etc/httpd.conf'],
  }
  file {'/etc/httpd.conf':
    ensure  => file,
    owner   => 'httpd',
    content => template('apache/httpd.conf.erb'), # Template from a module
  }
  service {'httpd':
    ensure    => running,
    enable    => true,
    subscribe => File['/etc/httpd.conf'],
  }
}
```

Figure 3. Puppet Class Example

In this tutorial we will leverage Cisco Metacloud's enhanced dashboard to orchestrate the instances of virtual machines for both the Puppet Master and Agent components of Puppet Enterprise. These two instances need to consume the underlying Infrastructure as a Service (IaaS) such as (compute, network, and storage), while using Puppet Enterprise to deploy applications and manage the quest operation systems running inside the Metacloud instances.

Installation of Puppet in Metacloud

Currently there are two options to evaluate Puppet Enterprise:

1. Option 1: Download and install PE here: https://puppetlabs.com/download-puppet-enterprise

   Includes license to manage (10) agents. PE can be installed on most Linux guest operating systems that meet the pre-install prerequisites.

2. Option 2: Download Learning VM and import image into Cisco Metacloud. Also includes license to manage (10) agents. Download the Learning VM here: https://puppetlabs.com/download-learning-vm

Option (2) was selected for this tutorial because the Learning VM includes a fully functional Puppet Master as well as an interactive tutorial called Quest. The quest tool will provide structure and feedback as you progress through a Puppet tutorial. If new to Puppet, it would be highly recommended to first complete "Quest" as a useful primer for understanding this tutorial before deploying the first application.

```
<quest --start welcome>
```

The remainder of this section will focus on getting the Learning VM imported and powered-on in Metacloud.

Import Learning VM:

Step 1. Download .zip file and extract to local folder

Step 2. Create a project in Metacloud

Step 3. Upload image (.vmdk) file to Metacloud

Step 4. Create security group to allow Puppet ports

Step 5. Launch Puppet Learning VM Instance in Metacloud

Step 6. Configure Instance and power on Learning VM

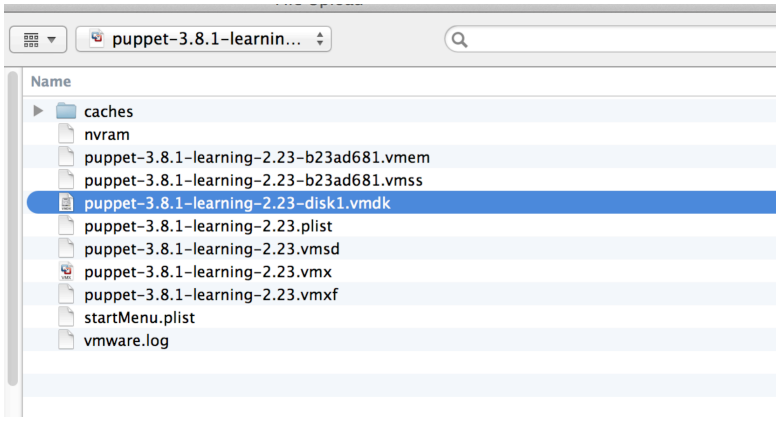

Figure 4. Download .zip file and extract to local folder

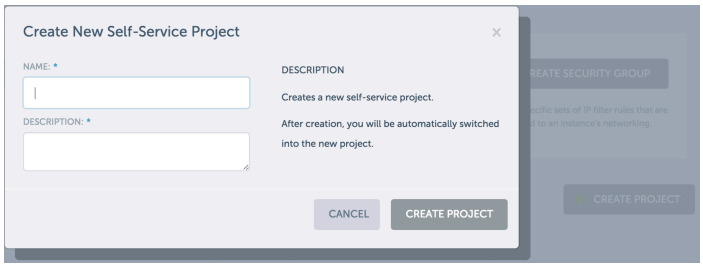

Figure 5. Create a project in Metacloud

Follow easy to use “Create Project” wizard located on the main Metacloud dashboard page that is available to existing Metacloud subscribers.

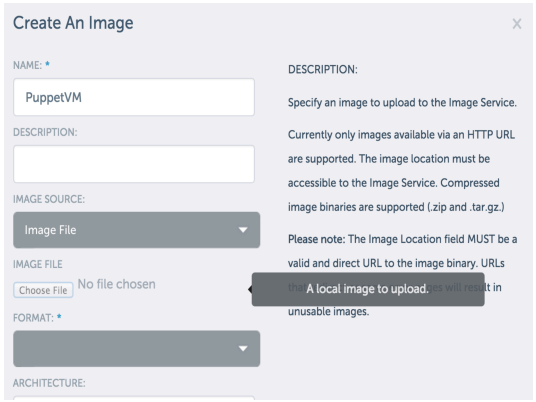

Figure 6. Upload image (.vmdk) file to Metacloud

For the image source, select the Image File option and then select Choose File to navigate to the Puppet learning vmdk file stored on your local laptop. The FORMAT dropdown should include “vmdk”. Please note the status bar for the actual upload after selecting “create image” at the bottom of the open window.

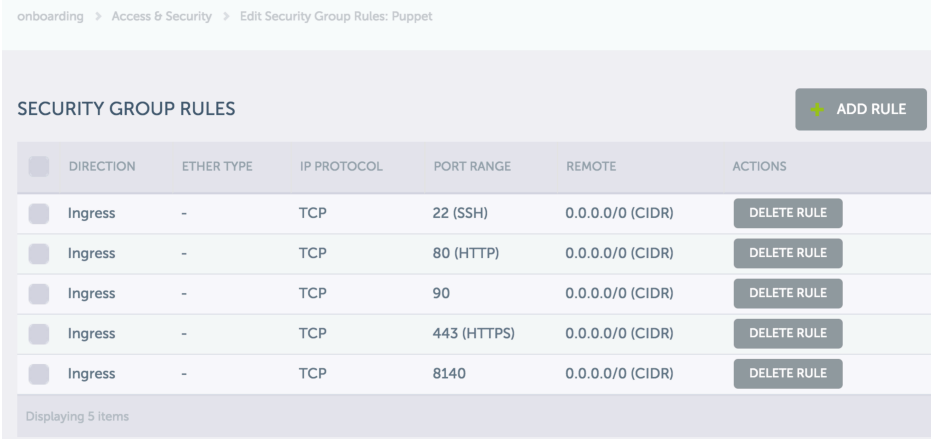

Figure 7. Create security group to allow Puppet ports

In order to create a new security group you can either navigate to “Access and security” from the left frame or select the security group wizard from the main screen. In this example the security group is named Puppet and the required ports were opened to allow management and client to server traffic to work in the Metacloud cloud. Please note, additional port access could be required depending on the use case and application to be launched by Puppet.

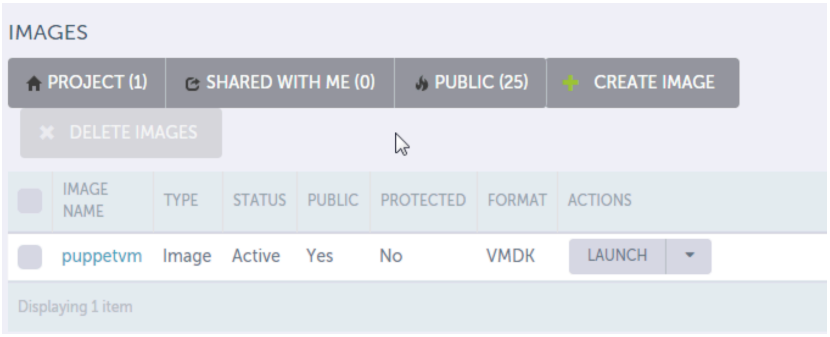

Figure 8. Launch Puppet Learning VM Instance in Metacloud

From the project panel on the left frame, select images and the puppetvm image prepared earlier. You can now launch the image to provision a new virtual machine instance of a Learning VM in the Metacloud project.

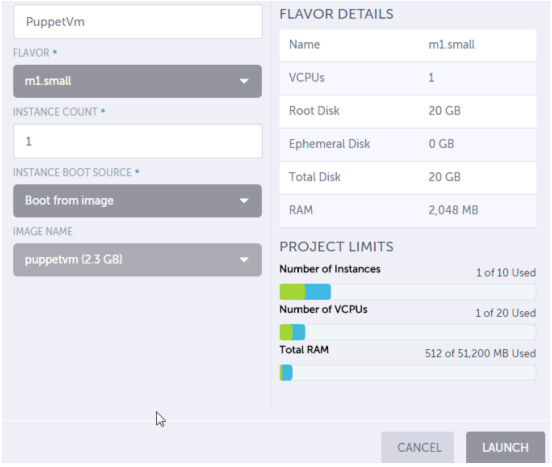

Figure 9. Configure Instance and power on Learning VM

In this example we named the instance PuppetVm. Considering the Learning VM is only a demo environment, we decided to use a small flavor of m1small instance type, which only allocates 1 vCPU with 2gig of ram and 20 gig of storage. After selecting a security group from the “Access & security” tab (not shown), the virtual machine and launch, the instance will automatically power on.

Puppet Agent Installation

Client Nodes can have the agent software preinstalled or download the Puppet Agent from the Puppet master. In either case the Puppet agent certificate has to be accepted on the Puppet master to prevent masquerading.

For the purposes of this tutorial we used a CentOS 6.4 guest operating system launched from an existing image in Metacloud. This image did not currently contain the Puppet agent so we installed it form the Puppet master.

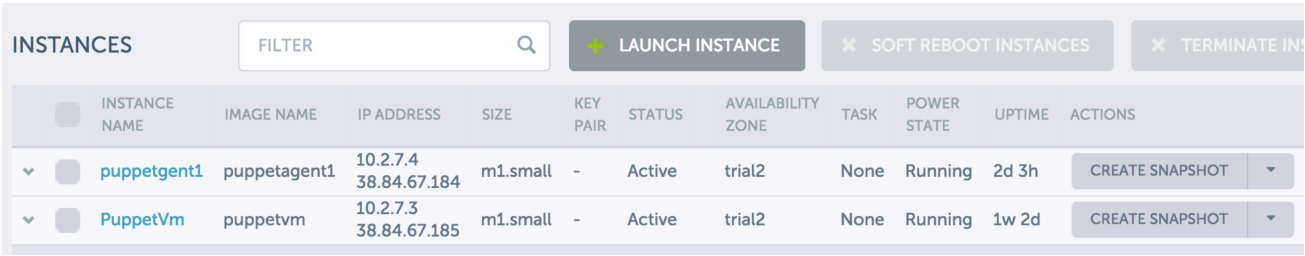

Figure 10. Puppet Master and Agent instances

If we connect to the console of the Learning VM by clicking on “PuppetVM” we can see the default login credentials to configure the Puppet Master to provide the proper repo and files for the Puppet Agent to pull down for installation. In this example we enabled a floating IP address for Internet access.

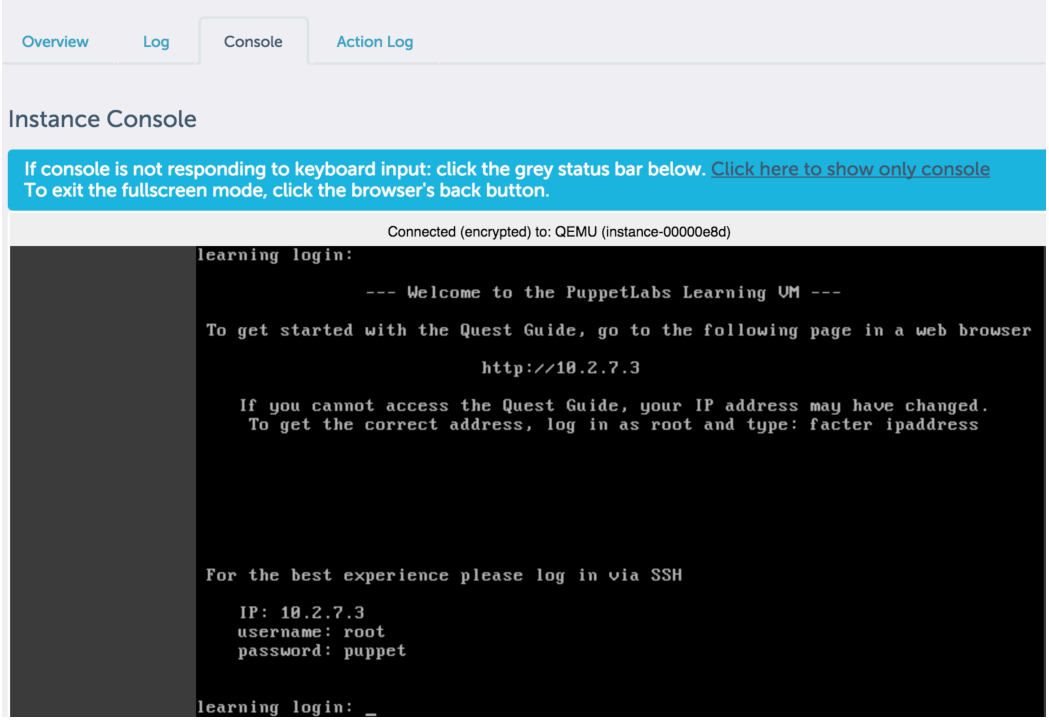

Figure 11. The Learning VM "Puppet Master" console

Once you have the credentials in hand, SSH is the recommended method to prepare the Puppet Master.

The following command will enable the repos on the Puppet Master:

```
# rpm -ivh http://yum.puppetlabs.com/puppetlabs-release-el-6.noarch.rpm
```

Now you can SSH or console into the client server and install the agent with the following command:

```
root@puppetgent1 etc]# curl -k https://10.2.7.3:8140/packages/current/install.bash | sudo bash
```

The previous command will also import the SSL certificate from the agent into the Puppet Master.

Next login to the Puppet Master server and list all certificates awaiting a signature:

```
# puppet cert list
```

"agent1" (SHA256)
73:D4:EF:1A:F6:B9:D8:2F:AB:6F:4F:95:CA:73:CE:3F:8C:8B:5C:23:BB:B2:17:47:98:08:C7:01:96:C1:17:E2

From the above we can see that certificate from a single host, agent1 is waiting for its certificate to be signed. Your output may be different and contain multiple certificates awaiting for a signature. From here we have two options on how to sign the above certificate. First, we can sign each certificate individually:

```
# puppet cert sign agent1
```

Notice: Signed certificate request for agent1

Notice: Removing file Puppet::SSL::CertificateRequest agent1

at '/etc/puppetlabs/puppet/ssl/ca/requests/agent1.pem'

Or we can sign all awaiting certificates at once:

```
# puppet cert sign —all
```

Basics of Puppet Manifest

Previously we mentioned that Puppet has a site.pp manifest that is used with all agents or modules can be created for a specific agent or subset of clients. In order to test our agent installation in the previous section we will configure a very basic class in the site.pp file and initiate the agent to pull from the catalog for any needed installation or changes.

The site.pp is found below on the Puppet master “Learning VM”. We can use VIM to edit the file.

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

In order to edit the file use “i” for insert and to make additions and changes. Also, when finished hit esc and then save with :wq

```
}
file {'/tmp/example-ip':
  ensure  => present,
  mode    => 0644,
  content => "Here is my Public IP Address: ${ipaddress_eth1}.\n",
}
```

Figure 12. File resource to check and print the interface IP address

The above manifest will make sure that all agent nodes will have a file at `/tmp/example-ip`, with `-rw-r--r--` permissions, and text that contains the node's public IP address.

You can either wait until the agent checks in with the master automatically, or you can run the `Puppet Agent--test` command (from one of your agent nodes). Then run the following command to print the file:

```
cat /tmp/example-ip
```

```
[root@agent ~]# cd /tmp
[root@agent tmp]# ls
example-ip      tmp.7ffz0Se9yB  tmp.D1URJHi6Xu  tmp.dC1qXj6xwi  tmp.dj8tqSPCBX  tmp.M111XTEB2A  yum.log
tmp.4m95IMNh3P  tmp.CXCboa38M8  tmp.dan17XUxVN  tmp.DH9oFtFBwP  tmpiu4knp       tmp.rNI3b8HLCA
[root@agent tmp]# cat example-ip
Here is my Public IP Address: 10.2.7.4.
[root@agent tmp]#
```

Figure 13. example-ip file

At this point we were successful with using a basic manifest to manipulate a file on the agent machine. The next step is to deploy our first application to Cisco Metacloud using Puppet.

Deploying an Application with Puppet

Apache web server was selected as the application to deploy to a CentOS instance running the Puppet Agent. In order to deploy Apache web server, we first need to locate the Apache files on Puppet Forge and install the Puppet

Apache class on the Puppet Master (Learning VM). To do that, create a new class for Apache in manifest site.pp. Afterwards the Puppet Master needs to be restarted. Finally, wait or manually force the agent to pull down the catalog by puppet agent –t.

Step One: puppet module search apache

Step Two: puppet module install puppetlabs-apache

Step Three: vi /etc/puppetlabs/puppet/environments/production/manifests/site.pp

Step Four: puppet master SIGHUP

Step Five: puppet agent –t

```
class { 'apache': }              # use apache module
  apache::vhost { 'example.com':  # define vhost resource
      port        => '8080',
          docroot => '/var/www/html'
            }
```

Figure 14. Edit site.pp manifest to add Apache server class

```
[root@agent www1]# puppet agent -t
Info: Retrieving pluginfacts
Info: Retrieving plugin
Info: Loading facts
Info: Caching catalog for puppetgent1.novalocal
Info: Applying configuration version '1437368491'
Info: Computing checksum on file /var/opt/lib/pe-puppet/concat/_etc_httpd_conf_ports.conf/fragments/10_Lis
ten 8080
Info: /Stage[main]/Apache/Concat[/etc/httpd/conf/ports.conf]/File[/var/opt/lib/pe-puppet/concat/_etc_httpd
_conf_ports.conf/fragments/10_Listen 8080]: Filebucketed /var/opt/lib/pe-puppet/concat/_etc_httpd_conf_por
ts.conf/fragments/10_Listen 8080 to puppet with sum dc90878e1fcc0d0241f57adcbef5bd44
Notice: /Stage[main]/Apache/Concat[/etc/httpd/conf/ports.conf]/File[/var/opt/lib/pe-puppet/concat/_etc_htt
pd_conf_ports.conf/fragments/10_Listen 8080]/ensure: removed
Info: Computing checksum on file /var/opt/lib/pe-puppet/concat/_etc_httpd_conf_ports.conf/fragments/10_Nam
eVirtualHost *_8080
Info: /Stage[main]/Apache/Concat[/etc/httpd/conf/ports.conf]/File[/var/opt/lib/pe-puppet/concat/_etc_httpd
_conf_ports.conf/fragments/10_NameVirtualHost *_8080]: Filebucketed /var/opt/lib/pe-puppet/concat/_etc_htt
pd_conf_ports.conf/fragments/10_NameVirtualHost *_8080 to puppet with sum 7c667d621f98996802aad6161f75fb56
Notice: /Stage[main]/Apache/Concat[/etc/httpd/conf/ports.conf]/File[/var/opt/lib/pe-puppet/concat/_etc_htt
pd_conf_ports.conf/fragments/10_NameVirtualHost *_8080]/ensure: removed
Info: /var/opt/lib/pe-puppet/concat/_etc_httpd_conf_ports.conf/fragments: Scheduling refresh of Exec[conca
t_/etc/httpd/conf/ports.conf]
```

Figure 15. Wait or force agent to pull down catalog with installation files for Apache web server

Figure 16. Check the client node to see if the expected Apache web page is running on port 8080. In my example the index.html was modified from the default.

Summary

Metacloud offers support for a robust selection of vendor and open source Application Lifecycle Management toolsets. Puppet complements Cisco Metacloud as an extensible configuration management solution with the ability to rapidly deploy and update applications, guest operating systems, and infrastructure devices running Puppet agents. Puppet and Metacloud together enable faster application delivery and simpler cloud and application lifecycle management.

For More Information

Visit our website to read more about Cisco Metacloud features and benefits.

To access technical tutorials about this product, visit our Community page.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV Amsterdam,
The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices.**

Printed in USA

CXX-XXXXXX-XX 10/11