puppet-scaleio Documentation

Release 0.1.0

Clint Kitson

November 05, 2014

Contents

The primary purpose of this project is create automation necessary for an Operations team to install and configure ScaleIO nodes using Puppet. As an additional benefit, we provided as a valuable example of Vagrantfiles which allow rapid deployment with pre-defined Puppet environments to configure ScaleIO deployments on a laptop or through other Vagrant providers. The project allows for a flexible ScaleIO environment serving for production, demos, or development.

CHAPTER 1

User Documentation

1.1 Overview

1.1.1 Summary

This documentation provides both simple and advanced views into running the ScaleIO Puppet module. There should be enough documentation here to solve for most viable use cases.

1.1.2 EMC CODE State

- Experimental

1.2 Quick Start

The easiest way to start using the ScaleIO Puppet module is to delpoy it locally to a laptop using Vagrant and Virtual Box. This method allows for a full deplyoment of Puppet, the ScaleIO Puppet module, and ScaleIO in five commands.

The default Vagrantfile includes one Puppet Master, and three ScaleIO nodes with mixed responsibilities. The Vagrantfile can be easily customized to deploy more nodes, and the module through the node classifier file (site.pp) is built to handle any configuration of nodes thereafter.

The module is currently intended to be used to configure a ScaleIO cluster from scratch for first-time MDM configurations. For running deployments, it can be used to define SDS items such as physical devices and volumes, and SDC items such as Volumes.

If you do not have Vagrant and Virtual Box installed, it is suggested that you either download them directly or use a package manager like http://brew.sh. In addition you will need git installed or download the zip from https://github.com/clintonskitson/vagrant-puppet-scaleio/archive/master.zip

Brew only works on Mac OS X.

```
brew cask install virtualbox
brew install vagrant
brew install git
```

1.2.1 Get ScaleIO RPMs

In order to perform the install of the components on the VMs, you must have the ScaleIO RPMs. These can be downloaded directly from the support.emc.com website.

1.2.2 Update ScaleIO Version

There is one parameter that must be modified depending on your ScaleIO RPM version numbers. There is a global parameter in the puppet/manifests/site.pp file that needs to be updated.

Check your RPM files and find the proper version number. The following RPM file has a version of 1.30-426.0. This is the SDC RPM, but is only being used as the example of identifying the version number.

```
EMC-ScaleIO-sdc-1.30-426.0.el6.x86_64.rpm
```

Plug that version into the puppet/manifests/site.pp file. Important Note: If you are using the example site.pp files, make sure you update this version once you copy those files from puppet/manifests/examples to puppet/manifests/site.pp.

```
$version = '1.30-426.0'
```

1.2.3 Vagrant Deploy

Issue the following commands to retrieve the proper repository with necessary files and issue an 'vagrant up' command.

```
git clone https://github.com/clintonskitson/vagrant-puppet-scaleio
cd vagrant-puppet-scaleio
cp RHEL6/EMC-ScaleIO-*.rpm puppet/modules/emcscaleio/files
cp ScaleIO_v1.30_GA/ScaleIO-GW\ \(IM+Rest-GW\)*.rpm puppet/modules/emcscaleio/files
vagrant up
vagrant status
```

Following this you should see the following Vagrant status report.

```
Current machine states:

puppetmaster              running (virtualbox)
tb                        running (virtualbox)
mdm1                      running (virtualbox)
mdm2                      running (virtualbox)
```

There are three network adapters that are auto-configured by default. Under the covers the 192.168.50.0/24 subnet is used for isolated ScaleIO communication.

Hostname	IP
tb	192.168.50.11
mdm1	192.168.50.12
mdm2	192.168.50.13

1.2.4 Vagrant Customization

The Vagrantfile is responsible for creating the proper VMs, configuring them, and then running the respective Puppet commands to start the server or agents. The Vagrantfile and the ScaleIO Puppet module are completely separate and can be used independently.

Configure the Puppet Master details in the puppetmaster_nodes hash.

```
puppetmaster_nodes = {
  'puppetmaster' => {
    :ip => '192.168.50.9', :hostname => 'puppetmaster', :domain => 'scaleio.local', :memory => 1024,
  }
}
```

The scaleio_nodes hash holds the configuration of the ScaleIO nodes. Three nodes are needed at a minimum for ScaleIO to be configured. Here you can see that we have a Tie-Breaker and two MDMs. These nodes are all multi-role where they are also serving as servers and clients for data services. The ScaleIO Puppet module determines these component roles.

```
scaleio_nodes = {
  'tb' => { :ip => '192.168.50.11', :hostname => 'tb', :domain => 'scaleio.local', :memory => 1024,
  'mdm1' => { :ip => '192.168.50.12', :hostname => 'mdm1', :domain => 'scaleio.local', :memory => 102
  'mdm2' => { :ip => '192.168.50.13', :hostname => 'mdm2', :domain => 'scaleio.local', :memory => 102
}
```

1.2.5 Vagrant Runtime

Vagrant status can be seen if you run 'vagrant status' from the directory with the Vagrantfile. Before modifying the Vagrantfile make sure that you destroy the existing ScaleIO VMs 'vagrant destroy'.

At any point you can add nodes to the Vagrant file scaleio_nodes hash followed by a 'vagrant up' command. See Advanced Usage for more details.

1.2.6 Verify Node Deploy

You can verify at any time that the nodes have been configured using Puppet and ScaleIO commands.

From the node you can run a Puppet Agent command. The following command will run the agent again and verify all configuration is correct.

```
vagrant ssh node
sudo puppet agent -t
```

From an MDM node you can run ScaleIO commands. The following command will return general information.

```
vagrant ssh mdm2
sudo /bin/scli --login --username admin --password 'Scaleio123' --mdm_ip 192.168.50.12
sudo /bin/scli --query_all --mdm_ip 192.168.50.12
```

Use this command for a verbose listing of commands.

```
sudo /bin/scli --all --help
```

1.2.7 Puppet Node Redeploy

If you have configured a node prior and want to redeploy, make sure that you remove the certificate for the old node from the Puppet Master. Since ScaleIO requires that you install two MDM's and a Tie Breaker as a base install, do not try and redeploy these nodes. This configuration must be done manually to get these nodes back online. Redeploys of other node types do not have the same ramifications.

```
vagrant ssh puppetmaster
sudo puppet cert clean node.scaleio.local
```

If the node still exists you also must remove the certificate from the node.

```
vagrant ssh node
sudo find / -name node.scaleio.local.cer -delete
```

1.3 Graphical UI

See the ScaleIO documentation for more details on loading the GUI.

1.3.1 Untar the RPM

The GUI requires that you download Java JRE to the system that is going to run the GUI. If you have brew installed you can leverage this to install via CLI.

```
brew update && brew cask install java
```

Verify installation with the following command.

```
which java
```

For Linux or OS X platforms, locate the GUI RPM named EMC-ScaleIO-gui-1.30-426.0.noarch.rpm. This RPM has a shell script and a JAR file. You can get the contents of the RPM without running an RPM command by using tar.

```
tar -zxvf EMC-ScaleIO-gui-1.30-426.0.noarch.rpm
./opt/emc/scaleio/gui/run.sh
```

1.3.2 Load the GUI

If Java is in the path, then the GUI should load. Enter admin for the username, and the password you configured in the site.pp file. The IP address is the primary MDM ip.

```
admin
Scaleio123
192.168.50.12
```

1.4 Advanced

In addition to using example vagrant file (Vagrantfile-default) that defines VMs and the Puppet node configuration file (site.pp-default), you can also customize these files to meet install requirements for most environments.

The Puppet Master server shares the puppet directory with the Vagrant host (OS X in some cases). Editing files under the puppet directory on the Vagrant host or /etc/puppet on the Puppet Master server have the same effect. But always remember to reset the Puppet Master service if editing these files.

1.4.1 Add SDS Nodes

Vagrantfile Updates

For example, if you are running the default configuration (3 mixed use nodes), you may want to add new dedicated data service nodes (SDS) to add more storage. Notice the following scaleio_nodes hash where we add two extra hash

keys named sds1 and sds2. Notice also that we add the SDS nodes before the mdm nodes! This is important since the mdm2 node is responsible for configuring the cluster and data services and expects that the nodes have already been deployed and configured by the Puppet Agents.

```
scaleio_nodes = {
  'tb' => { :ip => '192.168.50.11', :hostname => 'tb', :domain => 'scaleio.local', :memory => 1024,
  'sds1' => { :ip => '192.168.50.14', :hostname => 'sds1', :domain => 'scaleio.local', :memory => 102
  'sds2' => { :ip => '192.168.50.15', :hostname => 'sds2', :domain => 'scaleio.local', :memory => 102
  'mdm1' => { :ip => '192.168.50.12', :hostname => 'mdm1', :domain => 'scaleio.local', :memory => 102
  'mdm2' => { :ip => '192.168.50.13', :hostname => 'mdm2', :domain => 'scaleio.local', :memory => 102
}
```

Site.pp Updates

In addition to this we will need to update the sio_sds_device hash. This should include a couple of lines with the FQDN of the hosts as keys. See sds1.scaleio.local and sds2.scaleio.local where we also define identical capacity amounts. Ordering is not critical here.

There is a pre-configured file at path puppet/manifest/examples/site.pp-2sds that can be copied over puppet/manifest/examples/site.pp.

```
$sio_sds_device = {
  'tb.scaleio.local' => {
    'ip' => '192.168.50.11',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
  'mdm1.scaleio.local' => {
    'ip' => '192.168.50.12',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
  'mdm2.scaleio.local' => {
    'ip' => '192.168.50.13',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
'sds1.scaleio.local' => {
    'ip' => '192.168.50.14',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
```

```
  'sds2.scaleio.local' => {
    'ip' => '192.168.50.15',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
}
```

There is no need to update the sio_sdc_volume hash since this only relates to Volumes that are consumed and advertised to specific clients.

The last thing to set is the node statement. Here we specify a regular expression, but can be hardcoded to a specific FQDN of a node. Following this we specify the minimum amount of parameters that must be passed in order to configure the node for its component type (sds). Notice how we specify /sds/ which matches any node with SDS in the FQDN while applying global variables.

The site.pp-default file already has this section. Notice also the sds_ssd_env_flag setting. This sould be set to true if you want to optimize the SDS operating system for solid state drives before deploying the SDS services.

```
node /sds/ {
  include firewall
  class {'emcscaleio::params':
        password => $password,
        version => $version,
        mdm_ip => $mdm_ip,
        sds_network => $sds_network,
        sio_sds_device => $sio_sds_device,
        sds_ssd_env_flag => true,
        components => ['sds'],
  }
  include emcscaleio
}
```

Restart Puppet Master

The next step is to reset the Puppet Master service. This is only necessary if you are going to add nodes to a running environment. If you are going to start from scratch with the new configuration, then you do not need to restart the Puppet Master service.

```
vagrant ssh puppetmaster
sudo /etc/init.d/puppetmaster restart
```

Vagrant Deploy

With Puppet configured, you are now ready to deploy the new Vagrant node. If you want to start from scratch with your new configuration then you can do the following.

```
vagrant destroy -f
vagrant up
```

Otherwise, a simple 'vagrant up' command will find the missing nodes and deploy sds1 and sds2.

Force MDM2 to Check-In

Changes to SDCs, SDSs, Volumes, or other data service configurations require that the MDM2 Puppet Agent runs after the Puppet Agents on the respective nodes run.

```
vagrant ssh mdm2
sudo puppet agent -t
```

This will complete the configuration of the new SDS nodes.

1.4.2 Add New Devices to SDS

You can leverage this module to add new devices local to the SDS nodes to be consumed by the SDS service. Consult ScaleIO documentation for the requirements for new storage to SDS devices. Here we will show how to add same sized devices to all configured SDSs.

The following snippet shows the devices key with a single device (default) and two optional configuration parameters, size and storage_pool.

```
'devices' => {
'/home/vagrant/sio_device1' => {
        'size' => '100GB',
        'storage_pool' => 'capacity'
},
```

Below is the full sio_sds_device hash with mutliple devices per SDS.

Site.pp Updates

There is a pre-configured file at path puppet/manifest/examples/site.pp-2devices that can be copied over puppet/manifest/examples/site.pp.

```
$sio_sds_device = {
  'tb.scaleio.local' => {
    'ip' => '192.168.50.11',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
      '/home/vagrant/sio_device2' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
  'mdm1.scaleio.local' => {
    'ip' => '192.168.50.12',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
      '/home/vagrant/sio_device2' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
```

```
  'mdm2.scaleio.local' => {
    'ip' => '192.168.50.13',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
      '/home/vagrant/sio_device2' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
```

Restart Puppet Master

The next step is to reset the Puppet Master service.

```
vagrant ssh puppetmaster
sudo /etc/init.d/puppetmaster restart
```

Force TB, MDM1, and MDM2 to Check-In

This is possibly an optional statement. Addition of new devices that must be created inside of the guest (truncate file), requires the Puppet Agent run on that node. If it is an existing device inside of /dev/, then the Agent does not need to run and you can simply force the check in for mdm2.

Repeat this for tb, mdm1, and mdm2. The mdm2 node must be done last in since it will configure the newly established devices.

```
vagrant ssh node
sudo puppet agent -t
```

This will complete the configuration of the Volume.

You should see the following at the end of the Puppet Agent run.

```
Notice: /Stage[main]/Emcscaleio::Login/Exec[Normal Login Class]/returns: executed successfully
Notice: mdm2.scaleio.local
Notice: /Stage[main]/Emcscaleio::Sds_first/Notify[mdm2.scaleio.local ]/message: defined 'message' as
Notice: mdm1.scaleio.local
Notice: /Stage[main]/Emcscaleio::Sds_first/Notify[mdm1.scaleio.local ]/message: defined 'message' as
Notice: tb.scaleio.local
Notice: /Stage[main]/Emcscaleio::Sds_first/Notify[tb.scaleio.local ]/message: defined 'message' as 't
Notice: /Stage[main]/Emcscaleio::Sds/Exec[Add SDS mdm1.scaleio.local device /home/vagrant/sio_device2
Notice: /Stage[main]/Emcscaleio::Sds/Exec[Add SDS mdm2.scaleio.local device /home/vagrant/sio_device2
Notice: /Stage[main]/Emcscaleio::Sds/Exec[Add SDS tb.scaleio.local device /home/vagrant/sio_device2],
Notice: /Stage[main]/Emcscaleio::Sds_sleep/Exec[Add SDS mdm1.scaleio.local Sleep 30]/returns: execute
Notice: Finished catalog run in 36.98 seconds
```

1.4.3 Add Volume to SDC

You can leverage this module to manage the creation and mapping of Volumes to SDCs. For this you there is a sio_sdc_volume hash in the site.pp file. Notice how we added a key for volume2 specifying the size_gb, protection_domain, storage_pool, and which client SDC's to advertise the volume to.

Site.pp Updates

There is a pre-configured file at path puppet/manifest/examples/site.pp-volume2 that can be copied over puppet/manifest/examples/site.pp.

```
$sio_sdc_volume = {
  'volume1' => {
    'size_gb' => 8,
    'protection_domain' => 'protection_domain1',
    'storage_pool' => 'capacity',
    'sdc_ip' => [
      '192.168.50.11',
      '192.168.50.12',
      '192.168.50.13',
    ]
  },
  'volume2' => {
    'size_gb' => 16,
    'protection_domain' => 'protection_domain1',
    'storage_pool' => 'capacity',
    'sdc_ip' => [
      '192.168.50.11',
      '192.168.50.12',
      '192.168.50.13',
    ]
  }
}
```

Restart Puppet Master

The next step is to reset the Puppet Master service.

```
vagrant ssh puppetmaster
sudo /etc/init.d/puppetmaster restart
```

Force MDM2 to Check-In

Changes to Volumes are performed from the mdm2 node.

```
vagrant ssh mdm2
sudo puppet agent -t
```

This will complete the configuration of the Volume.

You should see the following at the end of the Puppet Agent run.

```
Notice: /Stage[main]/Emcscaleio::Login/Exec[Normal Login Class]/returns: executed successfully
Notice: /Stage[main]/Emcscaleio::Volume/Exec[Add Volume volume2]/returns: executed successfully
Notice: /Stage[main]/Emcscaleio::Map_volume/Exec[Add Volume volume2 to SDC 192.168.50.12]/returns: ex
Notice: /Stage[main]/Emcscaleio::Map_volume/Exec[Add Volume volume2 to SDC 192.168.50.11]/returns: ex
Notice: /Stage[main]/Emcscaleio::Map_volume/Exec[Add Volume volume2 to SDC 192.168.50.13]/returns: ex
Notice: Finished catalog run in 4.74 seconds
```

1.5 Parameters

There are a limited set of parameters that can be configured within the Puppet node classifier file. This file (site.pp) can be used as is, or can be switched out for any one of Puppet's External Node Classifiers (ENC) for more dynamic functionality.

The node classifier file (site.pp) is the file that a Puppet server uses to determine what a node configuration should be when it checks in. This file matches nodes for their FQDN.

Below we have broken up the (site.pp) file which can serve as a working example.

1.5.1 Global Parameters

A global parameter gets set in the following section. These parameters can also be set individually per node if desired. These variable names are used later on in the node configuration section.

Parameter	Description
version (install required)	ScaleIO RPM version numbers that should be installed or upgraded to
sds_network (optional)	Network address that SDS should communicate on
mdm_ip (config required)	Array of MDM IP addresses as ['Primary','Secondary']
tb_ip	Tie-Breaker IP that is required if the ScaleIO cluster is not installed yet
cluster_name	Cluster name that should be configured
enable_cluster_mode	Boolean that determines whether Cluster Mode is on or off
password	New password to be set on the MDM during initial install
gw_password	Password for the Gateway REST interface

```
$version = '1.30-426.0'
$sds_network = '192.168.50.0/24'
$mdm_ip = ['192.168.50.12','192.168.50.13']
$tb_ip = '192.168.50.11'
$cluster_name = "cluster1"
$enable_cluster_mode = true
$password = 'Scaleio123'
$gw_password= 'Scaleio123'
```

1.5.2 SDS Devices

The SDS Devices represent the block storage devices on an SDS node. As part of the parameters you also specify which protection domain an SDS node takes part in. The devices themselves can be block devices (partitions) that have a pre-determined size or files on file systems that will be truncated to create the sparse space.

This is an example of a hash that represents a node.

```
'FQDN' => {
  'ip' => 'IP of NODE',
  'protection_domain' => 'Protection Domain',
  'devices' => {
    'Device Path1' => {  'size' => '100GB',
                           'storage_pool' => 'Name of Storage Pool'
                        },
    'Device Path2' => {  'size' => '100GB',
                           'storage_pool' => 'Name of Storage Pool'
                        },
  }
```

This are working hashes that represents nodes.

```
$sio_sds_device = {
  'tb.scaleio.local' => {
    'ip' => '192.168.50.11',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
  'mdm1.scaleio.local' => {
    'ip' => '192.168.50.12',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
  'mdm2.scaleio.local' => {
    'ip' => '192.168.50.13',
    'protection_domain' => 'protection_domain1',
    'devices' => {
      '/home/vagrant/sio_device1' => {  'size' => '100GB',
                                        'storage_pool' => 'capacity'
                                     },
    }
  },
}
```

1.5.3 SDC Volumes

The volumes section represents the actual consumable device that a client has available to carve space from. Here you specify the Protection Domain and Storage Pool. You can also specify any abritrary size in GB. The sdc_ip represents an array of SDC IPs that will have access to the volume.

```
$sio_sdc_volume = {
  'volume1' => { 'size_gb' => 8,
  'protection_domain' => 'protection_domain1',
  'storage_pool' => 'capacity',
  'sdc_ip' => [
      '192.168.50.11',
      '192.168.50.12',
      '192.168.50.13',
    ]
  }
}
```

1.5.4 Call-Home

The Call-Home section is used to configure the Call-Home service.

```
$callhome_cfg = {
  'email_to' => "emailto@address.com",
  'email_from' => "emailfrom@address.com",
```

```
  'username' => "monitor_username",
  'password' => "monitor_password",
  'customer' => "customer_name",
  'smtp_host' => "smtp_host",
  'smtp_port' => "smtp_port",
  'smtp_user' => "smtp_user",
  'smtp_password' => "smtp_password",
  'severity' => "error",
}
```

1.5.5 Node Configuration

This is the node configuration. With each node statement there is a regular expression that determines what configuration is applied to nodes with specific FQDNs. The required configuration parameters differ depending on the node type. Since the MDM and TB nodes specified here have mixed components there are more variables specified that needed for those node types.

See the Global Parameters section for details of other parameters listed below. The components parameter is the only one that specifies something unique per node.

Parameter	Description
components	An array of the components that can be installed (tb,sds,sdc,gw,lia,callhome)

Notice that the nodes are using a Regular Expression (/tb/) which matches any node that checks in with TB in the name. The nodes can be duplicated and completely customizable based on site naming preferences. Notice the components line where we specify (tb,sds,sdc,gw) meaning the node has is multi-role as a Tie-Breaker, SDS, SDC, and Gateway.

```
node /tb/ {
  include firewall
  class {'emcscaleio::params':
    password => $password,
    version => $version,
    mdm_ip => $mdm_ip,
    tb_ip => $tb_ip,
    sds_network => $sds_network,
    callhome_cfg => $callhome_cfg,
    sio_sds_device => $sio_sds_device,
    sds_ssd_env_flag => true,
    components => ['tb','sds','sdc','gw'],
  }
  include emcscaleio
}

node /mdm/ {
  include firewall
  class {'emcscaleio::params':
    password => $password,
    version => $version,
    mdm_ip => $mdm_ip,
    tb_ip => $tb_ip,
    cluster_name => $cluster_name,
    sds_network => $sds_network,
    sio_sds_device => $sio_sds_device,
    sio_sdc_volume => $sio_sdc_volume,
    callhome_cfg => $callhome_cfg,
    components => ['mdm','sds','sdc','callhome'],
  }
```

```
  include emcscaleio
}

node /sds/ {
  include firewall
  class {'emcscaleio::params':
    password => $password,
    version => $version,
    mdm_ip => $mdm_ip,
    sds_network => $sds_network,
    sio_sds_device => $sio_sds_device,
    sds_ssd_env_flag => true,
    components => ['sds'],
  }
  include emcscaleio
}

node /sdc/ {
  include firewall
  class {'emcscaleio::params':
    password => $password,
    version => $version,
    mdm_ip => $mdm_ip,
    components => ['sdc'],
  }
  include emcscaleio
}

node /gw/ {
  include firewall
  class {'emcscaleio::params':
    gw_password => $gw_password,
    version => $version,
    mdm_ip => $mdm_ip,
    components => ['gw'],
  }
  include emcscaleio
}
```

1.6 Notes

1.6.1 Requires

- Tested with Puppet 3.7.2
- Internet access to download RPMs
- ScaleIO 1.30-426+

1.6.2 To Be Done

- Test updates
- Masterless Puppet Agents
- Fault Sets

- Licensing

1.6.3 Special Settings

- Puppet 3.7+
- puppet.conf [main] - parser = “future”