OpenShift Enterprise 2
Puppet Deployment Guide

Installing and Configuring OpenShift Enterprise Using Puppet

Red Hat OpenShift Documentation Team

OpenShift Enterprise 2 Puppet Deployment Guide

Installing and Configuring OpenShift Enterprise Using Puppet

Red Hat OpenShift Documentation Team

Legal Notice

Abstract

The OpenShift Enterprise Puppet Deployment Guide provides a walkthrough on configuring Puppet manifests to install and configure an OpenShift Enterprise 2.2 deployment. This guide is intended for experienced system administrators.

Table of Contents

Preface

1. Document Conventions

This manual uses several conventions to highlight certain words and phrases and draw attention to specific pieces of information.

1.1. Typographic Conventions

Four typographic conventions are used to call attention to specific words and phrases. These conventions, and the circumstances they apply to, are as follows.

`Mono-spaced Bold`

Used to highlight system input, including shell commands, file names and paths. Also used to highlight keys and key combinations. For example:

> To see the contents of the file **`my_next_bestselling_novel`** in your current working directory, enter the **`cat my_next_bestselling_novel`** command at the shell prompt and press **`Enter`** to execute the command.

The above includes a file name, a shell command and a key, all presented in mono-spaced bold and all distinguishable thanks to context.

Key combinations can be distinguished from an individual key by the plus sign that connects each part of a key combination. For example:

> Press **`Enter`** to execute the command.
>
> Press **`Ctrl`**+**`Alt`**+**`F2`** to switch to a virtual terminal.

The first example highlights a particular key to press. The second example highlights a key combination: a set of three keys pressed simultaneously.

If source code is discussed, class names, methods, functions, variable names and returned values mentioned within a paragraph will be presented as above, in **`mono-spaced bold`**. For example:

> File-related classes include **`filesystem`** for file systems, **`file`** for files, and **`dir`** for directories. Each class has its own associated set of permissions.

Proportional Bold

This denotes words or phrases encountered on a system, including application names; dialog-box text; labeled buttons; check-box and radio-button labels; menu titles and submenu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, select the **Left-handed mouse** check box and click **Close** to switch the primary mouse button from the left to the right (making the mouse suitable for use in the left hand).
>
> To insert a special character into a **gedit** file, choose **Applications** → **Accessories** → **Character Map** from the main menu bar. Next, choose **Search** → **Find…** from the **Character Map** menu bar, type the name of the character in the **Search** field and click **Next**. The character you sought will be highlighted in the

> **`Character Table`**. Double-click this highlighted character to place it in the **`Text to copy`** field and then click the **`Copy`** button. Now switch back to your document and choose **Edit** → **Paste** from the **`gedit`** menu bar.

The above text includes application names; system-wide menu names and items; application-specific menu names; and buttons and text found within a GUI interface, all presented in proportional bold and all distinguishable by context.

`Mono-spaced Bold Italic` or ***Proportional Bold Italic***

Whether mono-spaced bold or proportional bold, the addition of italics indicates replaceable or variable text. Italics denotes text you do not input literally or displayed text that changes depending on circumstance. For example:

> To connect to a remote machine using ssh, type **`ssh`** ***`username@domain.name`*** at a shell prompt. If the remote machine is **`example.com`** and your username on that machine is john, type **`ssh john@example.com`**.
>
> The **`mount -o remount`** ***`file-system`*** command remounts the named file system. For example, to remount the **`/home`** file system, the command is **`mount -o remount /home`**.
>
> To see the version of a currently installed package, use the **`rpm -q`** ***`package`*** command. It will return a result as follows: ***`package-version-release`***.

Note the words in bold italics above: username, domain.name, file-system, package, version and release. Each word is a placeholder, either for text you enter when issuing a command or for text displayed by the system.

Aside from standard usage for presenting the title of a work, italics denotes the first use of a new and important term. For example:

> Publican is a *DocBook* publishing system.

1.2. Pull-quote Conventions

Terminal output and source code listings are set off visually from the surrounding text.

Output sent to a terminal is set in **`mono-spaced roman`** and presented thus:

```
books        Desktop   documentation  drafts  mss    photos   stuff  svn
books_tests  Desktop1  downloads      images  notes  scripts  svgs
```

Source-code listings are also set in **`mono-spaced roman`** but add syntax highlighting as follows:

```
static int kvm_vm_ioctl_deassign_device(struct kvm *kvm,
                struct kvm_assigned_pci_dev *assigned_dev)
{
        int r = 0;
        struct kvm_assigned_dev_kernel *match;

        mutex_lock(&kvm->lock);

        match = kvm_find_assigned_dev(&kvm->arch.assigned_dev_head,
                                      assigned_dev->assigned_dev_id);
        if (!match) {
                printk(KERN_INFO "%s: device hasn't been assigned
before, "
                         "so cannot be deassigned\n", __func__);
                      r = -EINVAL;
                      goto out;
            }

            kvm_deassign_device(kvm, match);

            kvm_free_assigned_device(kvm, match);

out:
            mutex_unlock(&kvm->lock);
            return r;
}
```

1.3. Notes and Warnings

Finally, we use three visual styles to draw attention to information that might otherwise be overlooked.

Note

Notes are tips, shortcuts or alternative approaches to the task at hand. Ignoring a note should have no negative consequences, but you might miss out on a trick that makes your life easier.

Important

Important boxes detail things that are easily missed: configuration changes that only apply to the current session, or services that need restarting before an update will apply. Ignoring a box labeled “Important” will not cause data loss but may cause irritation and frustration.

Warning

Warnings should not be ignored. Ignoring warnings will most likely cause data loss.

2. Getting Help and Giving Feedback

2.1. Do You Need Help?

If you experience difficulty with a procedure described in this documentation, visit the Red Hat Customer Portal at http://access.redhat.com. Through the customer portal, you can:

- search or browse through a knowledgebase of technical support articles about Red Hat products.
- submit a support case to Red Hat Global Support Services (GSS).
- access other product documentation.

Red Hat also hosts a large number of electronic mailing lists for discussion of Red Hat software and technology. You can find a list of publicly available mailing lists at https://www.redhat.com/mailman/listinfo. Click on the name of any mailing list to subscribe to that list or to access the list archives.

2.2. We Need Feedback!

If you find a typographical error in this manual, or if you have thought of a way to make this manual better, we would love to hear from you! Please submit a report in Bugzilla: http://bugzilla.redhat.com/ against the product **OpenShift Enterprise**.

When submitting a bug report, be sure to mention the manual's identifier: *Docs Puppet Deployment Guide*

If you have a suggestion for improving the documentation, try to be as specific as possible when describing it. If you have found an error, please include the section number and some of the surrounding text so we can find it easily.

Chapter 1. Introduction to OpenShift Enterprise

OpenShift Enterprise by Red Hat is a Platform as a Service (PaaS) that provides developers and IT organizations with an auto-scaling, cloud application platform for deploying new applications on secure, scalable resources with minimal configuration and management overhead. OpenShift Enterprise supports a wide selection of programming languages and frameworks, such as Java, Ruby, and PHP. Integrated developer tools, such as Eclipse integration, JBoss Developer Studio, and Jenkins, support the application life cycle.

Built on Red Hat Enterprise Linux, OpenShift Enterprise provides a secure and scalable multi-tenant operating system for today's enterprise-class applications while providing integrated application runtimes and libraries.

OpenShift Enterprise brings the OpenShift PaaS platform to customer data centers, enabling organizations to implement a private PaaS that meets security, privacy, compliance, and governance requirements.

Report a bug

Chapter 2. Introduction to Puppet

Puppet is an open source configuration management utility produced by Puppet Labs®. You can define system configurations using Puppet's declarative language and store them in files called Puppet *manifests*, which use the **.pp** file extension.

Puppet can be run standalone on a system or used in an agent/master configuration. With standalone Puppet, systems run Puppet locally to apply their own configurations. With agent/master Puppet, a central Puppet master server or servers manages multiple systems running the Puppet agent. For either implementation, Puppet manifests are used to bring system configurations to the desired state. A commercial offering, Puppet Enterprise®, is also available from Puppet Labs®.

Note

Installation and configuration of Puppet is left to your discretion. While any of the Puppet implementations or products described above can be used to deploy OpenShift Enterprise 2.2, this guide covers standalone Puppet usage.

See the Puppet Labs® documentation [1] for more information on Puppet, including instructions for installing agent/master or standalone Puppet [2] or installing Puppet Enterprise® [3].

Report a bug

[1] https://docs.puppetlabs.com/

[2] https://docs.puppetlabs.com/guides/install_puppet/pre_install.html

[3] https://docs.puppetlabs.com/pe/latest/index.html

Chapter 3. Introduction to OpenShift Enterprise Puppet Deployments

Starting with OpenShift Enterprise 2.2, you can deploy a complete, production-capable OpenShift Enterprise environment using a supported Puppet module. While it can be used with or without a Puppet master, this guide covers the use of the module without a Puppet master by applying standalone Puppet manifests locally.

To deploy OpenShift Enterprise using Puppet, you must use the OpenShift Origin Puppet module. The OpenShift Origin Puppet module version 4.1.0 and later is capable of managing both OpenShift Origin and OpenShift Enterprise 2.2 or later. Versions of the Puppet module prior to 4.1.0 have not been tested for use with OpenShift Enterprise.

OpenShift Enterprise support with the Puppet module is enabled by setting the **ose_version** parameter in a host's Puppet manifest to the OpenShift Enterprise release version. For example, the line in the manifest would be:

```
ose_version                      => '2.2',
```

You can use the Puppet module to choose from the following *roles* to be configured on a host:

broker

Installs the broker and Management Console applications.

node

Installs the node component and cartridges.

msgserver

Installs an ActiveMQ message broker.

datastore

Installs MongoDB (not sharded/replicated).

nameserver

Installs a BIND DNS server configured with a TSIG key for dynamic updates.

Note

For more comprehensive information about the architecture of OpenShift Enterprise, including details about each of the above components, see the *OpenShift Enterprise Deployment Guide* [4].

The following features provided by the OpenShift Origin Puppet module are **not** supported when using OpenShift Enterprise:

- Collocation of **broker** and **node** roles.
- Using **keepalived** and **HAProxy** for broker high availability (HA) load balancing.
- **Avahi** and **Route53** DNS plug-ins.

- **`10gen-mms-agent`**, **`jbossas`**, and **`phpmyadmin`** cartridges (not distributed with OpenShift Enterprise).

Report a bug

[4] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html

Chapter 4. System Prerequisites

The following sections detail the system prerequisites for deploying OpenShift Enterprise 2.2 using Puppet 3.0 or later. This guide covers standalone Puppet usage. For information on other Puppet implementations, see Chapter 2, *Introduction to Puppet*.

Report a bug

4.1. Installing Puppet

Before you can successfully run Puppet to deploy OpenShift Enterprise, you must install Puppet 3.0 or later, provided by the *puppet* RPM package, on your target system. See the Puppet Labs® documentation [5] for more information on installing Puppet from the Puppet Labs® repository.

Report a bug

4.2. Configuring Repositories

The Puppet module does not currently support configuring **yum** repositories for OpenShift Enterprise. Before running Puppet, you must configure the appropriate subscriptions using either the Red Hat Subscription Manager (RHSM) or RHN Classic subscription method, or manually ensure that the appropriate **yum** repositories for each host role are available.

Repositories for Broker and Supporting Service Roles

If you are using RHSM or RHN Classic and configuring a host to have the **broker**, **msgserver**, or **datastore** role, ensure that the **Red Hat OpenShift Enterprise 2.2 Infrastructure** channel is enabled using your chosen subscription method. See the "Configuring Broker Host Entitlements" [6] section in the *OpenShift Enterprise Deployment Guide* for details.

Also ensure that the **yum** priorities and exclude directives are set appropriately by following the **oo-admin-yum-validator** tool instructions in the "Configuring Yum on Broker Hosts" [7] section that follows in the same guide.

If you are configuring a host to only have the **nameserver** role, only the Red Hat Enterprise Linux 6 Server base channel is required.

Repositories for the Node Role

If you are using RHSM or RHN Classic and configuring a host to have the **node** role, ensure that the **Red Hat OpenShift Enterprise 2.2 Application Node** channel is enabled using your chosen subscription method. If you intend to install any premium cartridges, ensure the host has access to any relevant add-on subscriptions as well. See the "Configuring Node Host Entitlements" [8] section in the *OpenShift Enterprise Deployment Guide* for details.

Also ensure that the **yum** priorities and exclude directives are set appropriately by following the **oo-admin-yum-validator** tool instructions in the "Configuring Yum on Node Hosts" [9] section that follows in the same guide.

Report a bug

4.3. Installing the OpenShift Origin Puppet Module

Once Puppet has been installed on the target system, you can install the OpenShift Origin Puppet module by running the following command:

```
# puppet module install openshift/openshift_origin
```

To instead work from the Puppet module source, you can clone the **puppet-openshift_origin** repository onto the target system with the following:

```
# git clone https://github.com/openshift/puppet-openshift_origin.git
/etc/puppet/modules/openshift_origin
```

Report a bug

4.4. Generating a BIND TSIG Key

If you want OpenShift Enterprise to act as the name server and manage DNS for applications hosted on OpenShift Enterprise, you must generate a TSIG key for the OpenShift Enterprise BIND instance. This key is used to update DNS records in the BIND server that will be installed, both for managing application DNS and (by default) for creating host DNS records.

Procedure 4.1. To Generate a BIND TSIG Key:

1. The **dnssec-keygen** command, provided by the *bind* package, can be used to generate a TSIG key. Install the *bind* package on a host, if required:

   ```
   # yum install bind
   ```

   Note

   The *bind* package is available in the Red Hat Enterprise Linux 6 Server base channel.

2. Configure the **$domain** environment variable to simplify the process in the following step, replacing **Cloud_Domain** with the domain name to suit your environment:

   ```
   # domain=Cloud_Domain
   ```

3. Generate a TSIG key for your chosen cloud domain:

   ```
   # dnssec-keygen -a HMAC-MD5 -b 512 -n USER -r /dev/urandom -K
   /var/named $domain
   # cat /var/named/K$domain.*.key | awk '{print $8}'
   ```

   The format for the TSIG key returned by the last command should resemble **CNk+wjszKi9da9nL/1gkMY7H+GuUng==**. This key is set in the **bind_key** Puppet parameter in later sections.

4. If you want your OpenShift Enterprise hosts to be in a separate domain than the zone used for applications hosted on OpenShift Enterprise, you can create a second TSIG key at this time as well:

```
# infra_domain=Infrastructure_Domain
# dnssec-keygen -a HMAC-MD5 -b 512 -n USER -r /dev/urandom -K
/var/named $infra_domain
# cat /var/named/K$infra_domain.*.key | awk '{print $8}'
```

This key can be set in the **dns_infrastructure_key** Puppet parameter in later sections, if the **dns_infrastructure_zone** parameter is set.

Report a bug

4.5. Updating the Host Name

If required, change the host name of the target system before you deploy.

Procedure 4.2. Updating the Host Name

1. Update the host name in the **/etc/sysconfig/network** file:

   ```
   NETWORKING=yes
   HOSTNAME=New_Hostname
   ```

2. Also update the host name using the **hostname** command:

   ```
   # hostname New_Hostname
   ```

The above instructions set the host name locally, but not in DNS. For node hosts, this is used as the server identity. Generally it is best to use a name that matches how the host will resolve in DNS.

Report a bug

[5] http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html#for-red-hat-enterprise-linux-and-derivatives

[6] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#sect-Configuring_Broker_Host_Entitlements

[7] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#Configuring_Yum_on_Broker_Hosts

[8] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#sect-Configuring_Node_Host_Entitlements

[9] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#Configuring_Yum_on_Node_Hosts

Chapter 5. Puppet Configuration and Deployment

After Puppet is installed, you can create a configuration file, or manifest, for Puppet's installation parameters for the host. This file defines one class (**`openshift_origin`**) that tells Puppet which OpenShift Enterprise components to install and configure on the host. If you are new to Puppet, you can learn more about how this works in the Puppet Labs® documentation [10].

Chapter 6, *Example Puppet Configurations* provides example Puppet configurations covering various deployment scenarios. For any of the examples, the indicated configuration can be written to a manifest file called **`configure_ose.pp`** on each given host. For a comprehensive list of the installation parameters for OpenShift Enterprise you can specify with Puppet manifests, see Chapter 8, *Puppet Parameters*.

Important

As mentioned in Chapter 3, *Introduction to OpenShift Enterprise Puppet Deployments*, ensure the **`ose_version`** parameter is set to **`2.2`** in each host's Puppet manifest to enable OpenShift Enterprise support with the module.

Deploying OpenShift Enterprise Using Puppet

After creating a Puppet manifest to your specifications for a given host, you can begin the deployment process by running the Puppet utility on the host and specifying the manifest file:

```
# puppet apply --verbose configure_ose.pp
```

This process may take up to an hour. After it is completed, see Chapter 7, *Manual Post-Deployment Tasks* for important information on how to finish the OpenShift Enterprise setup.

Report a bug

[10] http://docs.puppetlabs.com/guides/parameterized_classes.html

Chapter 6. Example Puppet Configurations

The following sections provide example Puppet configurations covering various OpenShift Enterprise deployment scenarios.

Report a bug

6.1. Configuring One Broker Host and One Node Host

The following configuration puts the OpenShift Enterprise broker component and most of the supporting services on one host (the *broker host*), but configures a second host as a dedicated node (the *node host*). This is a good template for a basic, production-capable OpenShift Enterprise deployment as you can add node hosts as needed to increase capacity.

Example 6.1. Broker Host Configuration

```
class { 'openshift_origin' :
  roles => ['msgserver','datastore','nameserver','broker'],
  ose_version          => '2.2',

  # Hostname values
  broker_hostname      => 'broker.openshift.local',
  datastore_hostname   => 'broker.openshift.local',
  nameserver_hostname => 'broker.openshift.local',
  msgserver_hostname  => 'broker.openshift.local',
  node_hostname        => 'node.openshift.local',

  # IP address values
  broker_ip_addr     => '10.10.10.24',
  nameserver_ip_addr => '10.10.10.24',
  node_ip_addr       => '10.10.10.27',
  conf_node_external_eth_dev => 'eth0',

  # You must pre-configure your RPM sources
  install_method    => 'none',

  # OpenShift Config
  domain                          => 'example.com',
  conf_valid_gear_sizes          => 'small,medium,large',
  conf_default_gear_capabilities => 'small,medium',
  conf_default_gear_size         => 'small',
  openshift_user1                => 'demo',
  openshift_password1            => 'IZPmHrdxOgqjqB0TMNDGQ',

  # Datastore Config
  mongodb_port              => 27017,
  mongodb_replicasets      => false,
  mongodb_broker_user      => 'openshift',
  mongodb_broker_password => 'brFZGRCiOlmAqrMbj0OYgg',
  mongodb_admin_user       => 'admin',
  mongodb_admin_password  => 'BbMsrtPxsmSi5SY1zerN5A',

  # MsgServer config
  msgserver_cluster    => false,
  mcollective_user     => 'mcollective',
  mcollective_password => 'eLMRLsAcytKAJmuYOPE6Q',

  # DNS Config
  dns_infrastructure_zone => 'openshift.local',
  dns_infrastructure_names =>
  [
   { hostname => 'broker.openshift.local',
     ipaddr   => '10.10.10.24'
   },
   { hostname => 'node.openshift.local',
     ipaddr   => '10.10.10.27'
   }
  ],
  bind_key               =>
'yV9qIn/KuCqvnu7SNtRKU3oZQMMxF1ET/GjkXt5pf5JBcHSKY8tqRagiocCbUX56GOM/iu
P//D0TteLc3f1N2g==',
  dns_infrastructure_key =>
'UjCNCJgnqJPx6dFaQcWVwDjpEAGQY4Sc2H/llwJ6Rt+0iN8CP0Bm5j5pZsvvhZq7mxx7/M
dTBBMWJIA9/yLQYg==',
}
```

Example 6.2. Node Host Configuration

```
class { 'openshift_origin' :
  roles => ['node'],
  ose_version         => '2.2',

  # Hostname values
  broker_hostname     => 'broker.openshift.local',
  datastore_hostname  => 'broker.openshift.local',
  msgserver_hostname  => 'broker.openshift.local',
  nameserver_hostname => 'broker.openshift.local',
  node_hostname       => 'node.openshift.local',

  # IP Address values
  broker_ip_addr     => '10.10.10.24',
  nameserver_ip_addr => '10.10.10.24',
  node_ip_addr       => '10.10.10.27',
  conf_node_external_eth_dev => 'eth0',

  # You must pre-configure your RPM sources
  install_method    => 'none',

  # OpenShift Config
  domain                          => 'example.com',
  openshift_user1                 => 'demo',
  openshift_password1             => 'IZPmHrdxOgqjqB0TMNDGQ',
  conf_valid_gear_sizes           => 'small,medium,large',
  conf_default_gear_capabilities => 'small,medium',
  conf_default_gear_size          => 'small',

  # Datastore config
  mongodb_port => 27017,
  mongodb_replicasets => false,
  mongodb_broker_user => 'openshift',
  mongodb_broker_password => 'brFZGRCiOlmAqrMbj0OYgg',
  mongodb_admin_user => 'admin',
  mongodb_admin_password => 'BbMsrtPxsmSi5SY1zerN5A',

  # MsgServer Config
  mcollective_user     => 'mcollective',
  mcollective_password => 'eLMRLsAcytKAJmuYOPE6Q',

  # DNS Config
  dns_infrastructure_zone => 'openshift.local',
  bind_key =>
'yV9qIn/KuCqvnu7SNtRKU3oZQMMxF1ET/GjkXt5pf5JBcHSKY8tqRagiocCbUX56GOM/iu
P//D0TteLc3f1N2g==',
  dns_infrastructure_key =>
'UjCNCJgnqJPx6dFaQcWVwDjpEAGQY4Sc2H/llwJ6Rt+0iN8CP0Bm5j5pZsvvhZq7mxx7/M
dTBBMWJIA9/yLQYg==',
}
```

Report a bug

6.2. Configuring High Availability Deployments

The **broker**, **msgserver**, and **datastore** roles can be deployed in high availability (HA) configurations.

Report a bug

6.2.1. Configuring a High Availability Broker

Broker clustering is accomplished using an external load balancer in front of an arbitrary number of broker instances. This allows the broker and Management Console applications to achieve high availability. These applications do not require session affinity; round-robin load balancing is sufficient.

See the *OpenShift Enterprise Deployment Guide* [11] for additional information on broker redundancy.

The following are the variations that you must make to your basic configurations.

Example 6.3. Non-broker Host Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Use the *virtual* broker info for these values on hosts that are
not Brokers
  broker_hostname => 'virtbroker.openshift.local',
  broker_ip_addr  => '10.10.20.250',

  ...
}
```

Example 6.4. Broker Host Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Use the actual target host info for these values on hosts that are
Brokers
  broker_hostname => <target_hostname>,
  broker_ip_addr  => <target_ip_addr>,

  # Provide the cluster info
  broker_virtual_hostname     => 'virtbroker.openshift.local',
  broker_virtual_ip_address   => '10.10.20.250',

  ...
}
```

Example 6.5. Name Server Configuration

If OpenShift Enterprise is also handling DNS, add the following information to the host where you are deploying the **nameserver** role:

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Use the *virtual* broker info for these values
  broker_virtual_hostname => 'virtbroker.openshift.local',
  broker_virtual_ip_addr  => '10.10.20.250',

  # Additionally if you are using this nameserver to serve the domain
for
  # OpenShift host systems, include the virtual host info in the
infrastructure
  # list:
  dns_infrastructure_names =>
  [
   ...
   { hostname => 'virtbroker.openshift.local',
     ipaddr   => '10.10.10.250'
   },
  ],

  ...
}
```

Report a bug

6.2.2. Configuring a High Availability Datastore

The OpenShift Origin Puppet module configures multiple **datastore** instances into a MongoDB replica set. If you choose to use an HA datastore, you must provide at least three datastore hosts and the total number of hosts must be odd.

Hosts that include the **broker** role must have the following additional information:

Example 6.6. Broker Host Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Include the MongoDB replica set information for Brokers
  mongodb_replicasets          => true,
  mongodb_replicasets_members =>
['10.10.20.30:27071','10.10.20.31:27071','10.10.20.32:27071'],
}
```

Hosts that include the **datastore** role must have the following information:

Example 6.7. Datastore Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Set the datastore_hostname value to the current datastore host's
name
  datastore_hostname => <this_datastore_hostname>,

  # Include the MongoDB replica set information
  mongodb_replicasets          => true,
  mongodb_replicasets_members =>
['10.10.20.30:27071','10.10.20.31:27071','10.10.20.32:27071'],

  # One and only of the datastore hosts will be the primary
  mongodb_replica_primary => true|false,

  # All datastore hosts will know the primary's IP address and a
  # common replica key value
  mongodb_replica_primary_ip_addr = <primary_datastore_ip_addr>,
  mongodb_key                     = <replica_key_value>,
}
```

Report a bug

6.2.3. Configuring a High Availability Message Server

For message server redundancy, OpenShift Enterprise uses ActiveMQ's native clustering capability.

Example 6.8. Message Server Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Set the msgserver_hostname to the current msgserver host
  msgserver_hostname => <this_msgserver_hostname>,

  # Set the shared password that the cluster members will use
  msgserver_password => <shared_cluster_password>,

  # Specify the hostnames of all of the cluster members.
  msgserver_cluster         => true,
  msgserver_cluster_members =>
['msgserver1.openshift.local','msgserver2.openshift.local','msgserver3.o
penshift.local'],
}
```

Hosts that have the **broker** or **node** role must have the following information as well, even if they are not message server hosts:

Example 6.9. Broker and Node Configuration

```
class { 'openshift_origin' :
  # Other settings as appropriate per above examples
  ...

  # Specify the hostnames of the msgserver cluster members.
  msgserver_cluster         => true,
  msgserver_cluster_members =>
['msgserver1.openshift.local','msgserver2.openshift.local','msgserver3.o
penshift.local'],
}
```

Report a bug

6.3. Configuring a Complete Environment

The following is an example configuration for a complete OpenShift Enterprise deployment consisting of:

- One host with the **nameserver** role.
- Two hosts with the **broker** role.
- Two hosts with the **msgserver** role.
- Three hosts with the **datastore** role.
- Multiple hosts with the **node** role.

This example uses a common set of variables to ensure that parameters between different roles match up properly. This example is also appropriate for deploying on Infrastructure-as-a-Service (IaaS) environments where public and private routing is in use; applications are made available via their public IP addresses by making use of the **`ec2_public_ipv4`** fact.

Recommended Order of Deployment:

1. Name server host.
2. Message server hosts.
3. Datastore hosts, ensuring the primary host is fully deployed before adding additional nodes to your replica set.
4. Broker hosts.
5. Node hosts.

Common Variables for This Example

```
$bind_key =
'oZmVeXEiAi3foJ5GPG/11aaliaw1Wm7hccODfqBDfKRlu08bUfHK08mFMxpBnSW2bNJb+567M
c2sOwWyg7a1AA=='
$broker_virtual_hostname = 'ose-broker.example.com'
$install_method = 'none'
$mongodb_replicasets = true
$mongodb_replicasets_members = ["ose-mongo01.example.com:27017","ose-
mongo02.example.com:27017","ose-mongo03.example.com:27017"]
$msgserver_cluster = true
$msgserver_cluster_members = ["ose-amq01.example.com","ose-
amq02.example.com"]
$nameserver_ip_addr = '10.3.8.18'
$nameserver_hostname = 'ose-ns01.example.com'
$ose_version = '2.2'
$register_host_with_nameserver = true
```

Example 6.10. Name Server Configuration

```
node /^ose-ns/ {
  class { 'openshift_origin':
    roles                         => ['nameserver'],
    bind_key                      => $bind_key,
    install_method                => $install_method,
    nameserver_ip_addr            => $nameserver_ip_addr,
    nameserver_hostname           => $nameserver_hostname,
    ose_version                   => $ose_version,
    register_host_with_nameserver => $register_host_with_nameserver,
  }
}
```

Example 6.11. Message Server Configuration (ActiveMQ)

```
node /^ose-amq/ {
  class { 'openshift_origin':
    roles                         => ['msgserver'],
    msgserver_cluster            => $msgserver_cluster,
    msgserver_cluster_members    => $msgserver_cluster_members,
    msgserver_hostname           => $::fqdn,
    msgserver_routing            => $msgserver_routing,
    msgserver_routing_password   => $msgserver_routing_password,
    # all node classes get these params
    bind_key                      => $bind_key,
    install_method                => $install_method,
    nameserver_ip_addr            => $nameserver_ip_addr,
    nameserver_hostname           => $nameserver_hostname,
    ose_version                   => $ose_version,
    register_host_with_nameserver => $register_host_with_nameserver,
  }
}
```

Example 6.12. Datastore Configuration

This assumes that you are naming your primary **`datastore`** node **`ose-mongo01.example.com`**.

```
node /^ose-mongo/ {
  class { 'openshift_origin':
    roles                           => ['datastore'],
    datastore1_ip_addr             => '10.3.9.214',
    datastore2_ip_addr             => '10.3.9.219',
    datastore3_ip_addr             => '10.3.9.221',
    mongodb_replica_primary_ip_addr => '10.3.9.214',
    mongodb_replicasets            => $mongodb_replicasets,
    mongodb_replicasets_members    => $mongodb_replicasets_members,
    mongodb_replica_primary         => $::fqdn ? {
      'ose-mongo01.example.com'    => true,
      default       => false,
    },
    datastore_hostname             => $::fqdn,
    # all node classes get these params
    bind_key                      => $bind_key,
    install_method                => $install_method,
    nameserver_ip_addr            => $nameserver_ip_addr,
    nameserver_hostname           => $nameserver_hostname,
    ose_version                   => $ose_version,
    register_host_with_nameserver => $register_host_with_nameserver,
  }
}
```

Example 6.13. Node Configuration

You must have one set of configuration options per district. This registers the **`node_ip_address`** using the public IP address so that all applications on these gears are publicly accessible.

```
node /^ose-small-node/ {
  class { 'openshift_origin':
    roles                           => ['node'],
    broker_hostname                 => $::fqdn,
    node_hostname                   => $::fqdn,
    node_ip_addr                    => $::ec2_public_ipv4,
    node_frontend_plugins           => ["apache-mod-rewrite","nodejs-
websocket"],
    install_cartridges              =>
["cron","diy","haproxy","mongodb","nodejs","perl","php","postgresql","p
ython","ruby","jenkins","jenkins-client","mysql"],
    msgserver_cluster               => $msgserver_cluster,
    msgserver_cluster_members       => $msgserver_cluster_members,
    # all node classes get these params
    bind_key                        => $bind_key,
    install_method                  => $install_method,
    nameserver_ip_addr              => $nameserver_ip_addr,
    nameserver_hostname             => $nameserver_hostname,
    ose_version                     => $ose_version,
    register_host_with_nameserver => $register_host_with_nameserver,
  }
}
```

Example 6.14. Broker Configuration

```
node /^ose-broker/ {
  class { 'openshift_origin':
    roles                           => ['broker'],
    broker_cluster_members          => ["ose-broker01.example.com", "ose-
broker02.example.com"],
    broker_hostname                 => $::fqdn,
    broker_virtual_hostname         => $broker_virtual_hostname,
    msgserver_cluster               => $msgserver_cluster,
    msgserver_cluster_members       => $msgserver_cluster_members,
    conf_broker_session_secret     => 'secret',
    conf_console_session_secret    => 'secret',
    mongodb_replicasets             => $mongodb_replicasets,
    mongodb_replicasets_members     => $mongodb_replicasets_members,
    # all node classes get these params
    bind_key                        => $bind_key,
    install_method                  => $install_method,
    nameserver_ip_addr              => $nameserver_ip_addr,
    nameserver_hostname             => $nameserver_hostname,
    ose_version                     => $ose_version,
    register_host_with_nameserver => $register_host_with_nameserver,
  }
}
```

Report a bug

[11] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Deployment_Guide/index.html#Broker_Web_Application

Chapter 7. Manual Post-Deployment Tasks

The Puppet module attempts to automate as many tasks as it reasonably can. However, it is constrained to setting up only a single host at a time. In an assumed multi-host setup, you must perform the following manual post-deployment tasks after the module has completed.

1. **Set up DNS entries for hosts.** If you installed BIND using the Puppet module, then any other components installed with the module on the same host received DNS entries. Other hosts must all be defined manually, including at least your node hosts. The **oo-register-dns** command on a broker host may prove useful for this.

2. **Copy the rsync public key to enable moving gears.** The broker **rsync** public key must go on nodes, but it is difficult to script the task generically. Nodes should not have password-less access to brokers to copy the **.pub** key, so this must be performed manually on each node host:

   ```
   # scp root@broker:/etc/openshift/rsync_id_rsa.pub /root/.ssh/
   (The above step will ask for the root password of the broker
   machine.)
   # cat /root/.ssh/rsync_id_rsa.pub >> /root/.ssh/authorized_keys
   # rm /root/.ssh/rsync_id_rsa.pub
   ```

   If you skip this step, each gear move requires typing **root** passwords for each of the node hosts involved.

3. **Copy SSH host keys between the node hosts.** All node hosts must identify with the same host keys, so that when gears are moved between hosts, **ssh**, and **git** do not give developers spurious warnings about the host keys changing. Copy **/etc/ssh/ssh_*** from one node host to all of the rest. Alternatively, if using the same image for all hosts, keep the keys from the image.

4. **Create districts.** Nodes must belong to a district in order to work properly. Adding a node to a district after the node already has hosted applications running on it is very difficult, so it is important to do during the initial deployment. For a discussion of what districts are, see the *OpenShift Enterprise Administration Guide* [12].

   On a broker host, run the following command to define a new district:

   ```
   # oo-admin-ctl-district -c create -n District_Name -p Gear_Profile
   ```

   To perform a blank assignment of all nodes to a district, run:

   ```
   # oo-admin-ctl-district -c add-node -n District_Name -a
   ```

   Otherwise add nodes one at a time with:

   ```
   # oo-admin-ctl-district -c add-node -n District_Name -i
   Node_Hostname
   ```

5. **Import cartridge manfiests.** Run the following command on a broker host to import the cartridge manifests for all cartridges installed on nodes:

   ```
   # oo-admin-ctl-cartridge -c import-profile --activate --obsolete
   ```

This registers the cartridges with the broker and makes them available to developers for new hosted applications.

Note

For further information on administering an OpenShift Enterprise deployment, see the *OpenShift Enterprise Administration Guide* [13].

Report a bug

---

[12] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Administration_Guide/index.html#sect-Capacity_Planning_and_Districts

[13] https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html-single/Administration_Guide/index.html

Chapter 8. Puppet Parameters

The following is a comprehensive list of the OpenShift Enterprise installation parameters you can specify with Puppet manifests.

Note

Passwords are used to secure various services. You are advised to specify only alphanumeric values with this module as others may cause syntax errors depending on context. If non-alphanumeric values are required, update them separately after installation.

roles

Choose from the following roles to be configured on the host.

`broker`
: Installs the broker and Management Console applications.

`node`
: Installs the node component and cartridges.

`msgserver`
: Installs an ActiveMQ message broker.

`datastore`
: Installs MongoDB (not sharded/replicated).

`nameserver`
: Installs a BIND DNS server configured with a TSIG key for dynamic updates.

Default: **`['broker','node','msgserver','datastore','nameserver']`**

install_method

This sets the method for providing packages to the installation process. Currently, the only supported option for OpenShift Enterprise is **`none`**, meaning installation sources must already set up when the module executes (for example, using RHSM or RHN Classic).

domain

The network domain, or cloud domain, under which applications and hosts will be placed.

Default: **`'example.com'`**

broker_hostname, node_hostname, nameserver_hostname, msgserver_hostname, datastore_hostname

These parameters supply the FQDN of the hosts containing the respective components. Used for configuring the host’s name at installation and for configuring the broker application to reach the required services.

Default: The root plus the defined **`domain`** (for example, **`broker.example.com`**), except **`nameserver=ns1.example.com`**.

Note

If installing a name server, the module creates DNS entries for the host names of the other components being installed on this host as well. If you are using a name server that is set up separately, you are responsible for all necessary DNS entries.

datastore1_ip_addr|datastore2_ip_addr|datastore3_ip_addr

IP addresses of the first three MongoDB servers in a replica set. Add **`datastore`*`X`*`_ip_addr`** parameters for larger clusters.

Default: **`undef`**

nameserver_ip_addr

IP of a name server instance or current IP if installing on this node. This is used by every host to configure its primary name server.

Default: The current IP at installation time.

bind_key

When the name server is remote, use this to specify the key for updates. This is the **`Key:`** field from the **`.private`** key file generated by the **`dnssec-keygen`** command. This field is required on all node hosts.

bind_key_algorithm

When using a BIND key, use this algorithm for the BIND key.

Default: **`'HMAC-MD5'`**

bind_krb_keytab

When the name server is remote, this Kerberos keytab together with a Kerberos principal can be used instead of the **`dnssec`** key for updates.

bind_krb_principal

When the name server is remote, this Kerberos principal together with a Kerberos keytab can be used instead of the **`dnssec`** key for updates.

aws_access_key_id

This and the **`aws_secret_key`** parameter are Amazon AWS security credentials. The **`aws_access_key_id`** is a string which identifies an access credential.

For more information, see http://docs.aws.amazon.com/AWSSecurityCredentials/1.0/AboutAWSCredentials.html#AccessCredentials.

aws_secret_key

This is the secret portion of Amazon AWS security credentials indicated by the **`aws_access_key_id`** parameter.

aws_zone_id

This is the ID string for an AWS Hosted zone which will contain the OpenShift Enterprise application records.

For more information, see http://docs.aws.amazon.com/Route53/latest/DeveloperGuide/CreatingHostedZone.html

conf_nameserver_upstream_dns

List of upstream DNS servers to use when installing a nameserver on this node.

Default: **`['8.8.8.8']`**

broker_ip_addr

This is used for node hosts to record its broker. It is also the default for the name server IP if none is given.

Default: The current IP at installation time.

broker_virtual_ip_address

The virtual IP address that will front-end the broker cluster.

Default: **`undef`**

broker_virtual_hostname

The host name that represents the broker API cluster. This name is associated to the **`broker_virtual_ip_address`** parameter and added to BIND for DNS resolution.

Default: **`'changeme'`**

node_ip_addr

This is used for node hosts to give a public IP, if different from the one on its NIC.

Default: The current IP at installation time.

Node Resource Limits

The following resource limits must be the same within a given district.

node_profile

This is the specific node's gear profile. Default: **`'small'`**

node_quota_files

The max number of files allowed in each gear. Default: **`'80000'`**

node_quota_blocks

The max storage capacity allowed in each gear (1 block = 1024 bytes). Default: **`'1048576'`**

node_max_active_gears

This is used for limiting or guiding gear placement. For no overcommit, this must be:

```
(Total System Memory - 1G) / memory_limit_in_bytes
```

Default: **'100'**

node_no_overcommit_active

This enforces the **node_max_active_gears** parameter in a more stringent manner than normal. However, it also adds overhead to gear creation, so it should only be set to **true** when required. For example, in the case of enforcing single tenancy on a node.

Default: **false**

node_limits_nproc

The max number of processes. Default: **'250'**

node_tc_max_bandwidth

mbit/sec, total bandwidth allowed for all gears. Default: **'800'**

node_tc_user_share

mbit/sec, one user is allotted. Default: **'2'**

node_cpu_shares

The CPU share percentage for each gear. Default: **'128'**

node_cpu_cfs_quota_us

Default: **'100000'**

node_memory_limit_in_bytes

Gear memory limit in bytes. Default: **'536870912'** (512MB)

node_memsw_limit_in_bytes

Gear max memory limit including swap (512M + 100M swap). Default: **'641728512'**

node_memory_oom_control

Kill processes when hitting out of memory. Default: **'1'**

node_throttle_cpu_shares

The CPU share percentage each gear receives at throttle. Default: **'128'**

node_throttle_cpu_cfs_quota_us

Default: **'30000'**

node_throttle_apply_period

Default: **120**

node_throttle_apply_percent

Default: **'30'**

Default: **'30'**

node_throttle_restore_percent

Default: **'70'**

node_boosted_cpu_cfs_quota_us

Default: **'200000'**

node_boosted_cpu_shares

The CPU share percentage each gear receives while boosted. Default: **'30000'**

configure_ntp

Enabling this configures NTP. It is important that the time be synchronized across hosts because MCollective messages have a TTL of 60 seconds and may be dropped if the clocks are too far out of sync. However, NTP is not necessary if the clock will be kept in sync by some other means.

Default: **true**

ntp_servers

If the **configure_ntp** parameter is set to **true** (default), this parameter allows users to specify an array of NTP servers used for clock synchronization.

Default: **['time.apple.com iburst', 'pool.ntp.org iburst', 'clock.redhat.com iburst']**

Note

Use **iburst** after every NTP server definition to speed up the initial synchronization.

msgserver_cluster

Set to **true** to cluster ActiveMQ for high availability and scalability of OpenShift Enterprise message queues.

Default: **false**

msgserver_cluster_members

An array of ActiveMQ server host names. Required when the **msgserver_cluster** parameter is set to **true**.

Default: **undef**

mcollective_cluster_members

An array of ActiveMQ server host names. Required when the **msgserver_cluster** is set to **true**.

Default: **$msgserver_cluster_members**

msgserver_password

Password used by ActiveMQ's **amquser**. The **amquser** is used to authenticate ActiveMQ inter-cluster communication. Only used when the **msgserver_cluster** is set to **true**.

Default: **'changeme'**

msgserver_admin_password

This is the password for the **admin** user for the ActiveMQ Admin Console, which is not needed by OpenShift Enterprise, but might be useful in troubleshooting.

Default: Generates a random password.

mcollective_user, mcollective_password

This is the user and password shared between broker and node for communicating over the MCollective topic channels in ActiveMQ. Must be the same on all broker and node hosts.

Default: **'mcollective'** / **'marionette'**

mongodb_admin_user, mongodb_admin_password

This is the user name and password of the administrative user that will be created in the MongoDB datastore. These credentials are not used by in this module or by OpenShift Enterprise, but an administrative user must be added to MongoDB in order for it to enforce authentication.

Default: **'admin'** / **'mongopass'**

Note

The administrative user will not be created if **CONF_NO_DATASTORE_AUTH_FOR_LOCALHOST** is enabled.

mongodb_broker_user, mongodb_broker_password

This is the user name and password of the normal user that will be created for the broker to connect to the MongoDB datastore. The broker application's MongoDB plug-in is also configured with these values.

Default: **'openshift'** / **'mongopass'**

mongodb_name

This is the name of the database in MongoDB in which the broker will store data.

Default: **'openshift_broker'**

mongodb_port

The TCP port used for MongoDB to listen on.

Default: **'27017'**

mongodb_replicasets

Enables or disables MongoDB replica sets for database high availability.

Default: **`false`**

mongodb_replica_name

The MongoDB replica set name when the **`mongodb_replicasets`** parameter is set to **`true`**.

Default: **`'openshift'`**

mongodb_replica_primary

Set the host as the primary with **`true`** or secondary with **`false`**. Must be set on one and only one host within the **`mongodb_replicasets_members`** array.

Default: **`undef`**

mongodb_replica_primary_ip_addr

The IP address of the primary host within the MongoDB replica set.

Default: **`undef`**

mongodb_replicasets_members

An array of **`[host:port]`** of replica set hosts.

Example: [10.10.10.10:27017, 10.10.10.11:27017, 10.10.10.12:27017]

Default: **`undef`**

mongodb_keyfile

The file containing the **`mongodb_key`** used to authenticate MongoDB replica set members.

Default: **`'/etc/mongodb.keyfile'`**

mongodb_key

The key used by members of a MongoDB replica set to authenticate one another.

Default: **`'changeme'`**

openshift_user1, openshift_password1

This user and password are entered in the **`/etc/openshift/htpasswd`** file as a test user. Red Hat recommends removing the user after installation or using a different authentication method.

Default: **`'demo'`** / **`'changeme'`**

conf_broker_auth_salt, conf_broker_auth_private_key

Salt and private keys used when generating secure authentication tokens for application-to-broker communication. Requests like scale up or down and Jenkins builds use these authentication tokens. This value must be the same on all broker nodes.

Default: Self-signed keys are generated. Does not work with a multi-broker setup.

conf_console_product_logo

Relative path to the product logo URL.

If the **`ose_version`** parameter is undefined, the default is **`/assets/logo-origin.svg`**. If the **`ose_version`** parameter is defined, the deafult is **`/assets/logo-enterprise-horizontal.svg`**.

conf_console_product_title

OpenShift instance name.

If the **`ose_version`** parameter is undefined, the default is **`OpenShift Origin`**. If the **`ose_version`** parameter is defined, the deafult is **`OpenShift Enterprise`**.

conf_broker_multi_haproxy_per_node

This setting is applied on a per-scalable-application basis. When set to **`true`**, OpenShift Enterprise allows multiple instances of the HAProxy gear for a given scalable application to be established on the same node. Otherwise, on a per-scalable-application basis, a maximum of one HAProxy gear can be created for every node in the deployment. The latter is the default behavior, which protects scalable applications from single points of failure at the node level.

Default: **`false`**

conf_broker_session_secret, conf_console_session_secret

Session secrets used to encode cookies used by the broker and Management Console applications. These values must be the same on all broker nodes.

Default: **`undef`**

conf_valid_gear_sizes

List of all gear sizes that will be used in this OpenShift Enterprise installation.

Default: **`['small']`**

conf_default_gear_size

Default gear size if one is not specified.

Default: **`'small'`**

conf_default_gear_capabilities

List of all gear sizes that newly created users will be able to create.

Default: **`['small']`**

conf_default_max_domains

Default max number of domains a user is allowed to use.

Default: **`'10'`**

conf_default_max_gears

Default max number of gears a user is allowed to use.

Default: **`'100'`**

broker_dns_plugin

DNS plug-in used by the broker to register application DNS entries. Only one option is supported with OpenShift Enterprise:

`nsupdate`

> An nsupdate-based plug-in. Supports TSIG and GSS-TSIG based authentication. Uses the **`bind_key`** parameter for TSIG and the **`bind_krb_keytab`** and **`bind_krb_principal`** parameters for GSS-TSIG.

Default: **`'nsupdate'`**

broker_auth_plugin

Authentication setup for users of the OpenShift service. Options:

`mongo`

> Stores user names and passwords in MongoDB.

`kerberos`

> Kerberos-based authentication. Uses the **`broker_krb_service_name`**, **`broker_krb_auth_realms`**, **`broker_krb_keytab`** parameters.

`htpasswd`

> Stores user names and passwords in the **`/etc/openshift/htpasswd`** file.

`ldap`

> LDAP-based authentication. Uses the **`broker_ldap_uri`** parameter.

Default: **`'htpasswd'`**

broker_krb_service_name

The **`KrbServiceName`** value for a **`mod_auth_kerb`** configuration.

broker_krb_auth_realms

The **`KrbAuthRealms`** value for a **`mod_auth_kerb`** configuration.

broker_krb_keytab

The **`Krb5KeyTab`** value of **`mod_auth_kerb`** is not configurable. The keytab is expected to be at **`/var/www/openshift/broker/httpd/conf.d/http.keytab`**.

broker_ldap_uri

The URI to the LDAP server, for example:

`ldap://ldap.example.com:389/ou=People,dc=my-domain,dc=com?uid?sub?(objectClass=*)`

broker_ldap_bind_dn

LDAP DN (Distinguished name) of the user to bind to the directory with. For example:

`cn=administrator,cn=Users,dc=domain,dc=com`

Default: Anonymous bind.

broker_ldap_bind_password

Password of the bind user set in the **broker_ldap_bind_dn** parameter.

Default: Anonymous bind with a blank password.

node_shmmax

The **kernel.shmmax sysctl** setting for the **/etc/sysctl.conf** file.

The default setting should work for most deployments, but if this is desired to be tuned higher, the general recommendations are as follows:

```
shmmax = shmall * PAGE_SIZE
- PAGE_SIZE = getconf PAGE_SIZE
- shmall = cat /proc/sys/kernel/shmall
```

It is not recommended to set the **shmmax** to a value higher than 80% of total available RAM on the system (expressed in BYTES).

Default: **kernel.shmmax = 68719476736**

node_shmall

The **kernel.shmall sysctl** setting for the **/etc/sysctl.conf** file. Defaults to 2097152 BYTES

This parameter sets the total amount of shared memory pages that can be used system wide. Therefore, SHMALL should always be at least:

```
ceil(shmmax/PAGE_SIZE)
```

Default: **kernel.shmall = 4294967296**

node_container_plugin

Specify the container type to use on the node. Currently, the **selinux** plug-in is the default and only supported option for OpenShift Enterprise.

Default: **'selinux'**

node_frontend_plugins

Specify one or more plug-ins to use to register HTTP and WebSocket connections for applications. Options:

apache-vhost

A Virtual Host-based plug-in for HTTP and HTTPS. Suited for installations with less application create and delete activity. Easier to customize. If **apache-mod-rewrite** is also selected, **apache-vhost** is be ignored.

nodejs-websocket

A WebSocket proxy listening on ports 8000 and 8443.

haproxy-sni-proxy

A TLS proxy using SNI routing on ports 2303 through 2308.

apache-mod-rewrite

Deprecated in OpenShift Enterprise 2.2. A **mod_rewrite**-based plug-in for HTTP and HTTPS requests. Suited for installations with many create, delete, and scale actions. Cannot be used at the same time as the **apache-vhost** plug-in.

Default: **['apache-vhost','nodejs-websocket']**

node_unmanaged_users

List of user names who have UIDs in the range of OpenShift Enterprise gears but must be excluded from gear setups.

Default: **[]**

conf_node_external_eth_dev

External facing network device. Used for routing and traffic control setup.

Default: **'eth0'**

conf_node_public_key, conf_node_private_key

Public and private keys used for gears on the default domain. Both values must be defined or default self-signed keys will be generated.

Default: Self-signed keys are generated.

conf_node_supplementary_posix_groups

Name of supplementary UNIX group to add a gear to.

conf_node_watchman_service

Enable or disable the OpenShift Enterprise node Watchman service.

Default: **true**

conf_node_watchman_gearretries

Number of restarts to attempt before waiting **RETRY_PERIOD**.

Default: **'3'**

conf_node_watchman_retrydelay

Number of seconds to wait before accepting another gear restart.

Default: **'300'**

conf_node_watchman_retryperiod

Number of seconds to wait before resetting retries.

Default: **'28800'**

conf_node_watchman_statechangedelay

Number of seconds a gear must remain inconsistent with its state before Watchman attempts to reset state.

Default: **`'900'`**

conf_node_watchman_statecheckperiod

Wait at least this number of seconds since last check before checking gear state on the node. Use this to reduce the impact of Watchman's **`GearStatePlugin`** on the system.

Default: **`'0'`**

conf_node_custom_motd

Define a custom MOTD to be displayed to users who connect to their gears directly. If undefined, uses the default MOTD included with the node package.

Default: **`undef`**

development_mode

Set development mode and extra logging.

Default: **`false`**

install_login_shell

Install a Getty shell which displays DNS, IP, and login information. Used for the all-in-one VM installation.

register_host_with_nameserver

Set up DNS entries for this host in a locally-installed BIND DNS instance.

Default: **`false`**

dns_infrastructure_zone

The name of a zone to create which will contain OpenShift Enterprise infrastructure hosts. If this is unset, then no infrastructure zone or other artifacts will be created.

Default: **`''`**

dns_infrastructure_key

A **`dnssec`** symmetric key which grants update access to the infrastructure zone resource records. This is ignored unless the **`dns_infrastructure_zone`** parameter is set.

Default: **`''`**

dns_infrastructure_key_algorithm

When using a BIND key, use this algorithm for the infrastructure BIND key. This is ignored unless the **`dns_infrastructure_zone`** parameter is set.

Default: **`'HMAC-MD5'`**

dns_infrastructure_names

An array of hashes containing host name and IP address pairs to populate the infrastructure zone. This is ignored unless the **`dns_infrastructure_zone`** parameter is set.

Host names can be simple names or fully qualified domain name (FQDN).

Simple names are placed in the **`dns_infrastructure_zone`** parameter. Matching FQDNs are placed in the **`dns_infrastructure_zone`**. Host names anchored with a dot (**`.`**) are added verbatim.

For example:

```
$dns_infrastructure_names = [
  {hostname => "10.0.0.1", ipaddr => "broker1"},
  {hostname => "10.0.0.2", ipaddr => "data1"},
  {hostname => "10.0.0.3", ipaddr => "message1"},
  {hostname => "10.0.0.11", ipaddr => "node1"},
  {hostname => "10.0.0.12", ipaddr => "node2"},
  {hostname => "10.0.0.13", ipaddr => "node3"},
]
```

Default: **`[]`**

manage_firewall

Indicate whether or not this module configures the firewall for you.

install_cartridges

List of cartridges to be installed on the node. Options:

- cron
- diy
- haproxy
- mongodb
- nodejs
- perl
- php
- postgresql
- python
- ruby
- jenkins
- jenkins-client
- mysql
- jbossews
- jbosseap (requires add-on subscription)

Default: **['cron','diy','haproxy','mongodb','nodejs','perl','php','postgresql','python','ruby','jenkins','jenkins-client','mysql']**

update_network_conf_files

Indicate whether or not this module will configure the **resolv.conf** file and network for you.

Default: **true**

ose_version

Set this to the *X.Y* release version (for example, **2.2**) of OpenShift Enterprise to ensure an OpenShift Enterprise supported configuration is used.

See the **README_OSE.asciidoc** distributed with the **openshift_origin** Puppet module for more details.

Default: **undef**

ose_unsupported

Set this to **true** to allow OpenShift Enterprise unsupported configurations. Only appropriate for proof of concept environments.

This parameter is only used when the **ose_version** parameter is set.

Default: **false**

Report a bug

Revision History

Revision 1.0-1	**Tue Nov 11 2014**	**Alex Dellapenta**

Updated Chapter 8, *Puppet Parameters* to clarify parameter default values.

Revision 1.0-0	**Mon Nov 10 2014**	**Alex Dellapenta**

Initial publication for OpenShift Enterprise 2.2 release.