



OpenShift Enterprise 2 User Guide

Managing Applications in the Cloud with OpenShift Enterprise

Red Hat OpenShift Documentation Team

OpenShift Enterprise 2 User Guide

Managing Applications in the Cloud with OpenShift Enterprise

Red Hat OpenShift Documentation Team

Legal Notice

Copyright © 2015 Red Hat.

This document is licensed by Red Hat under the [Creative Commons Attribution-ShareAlike 3.0 Unported License](#). If you distribute this document, or a modified version of it, you must provide attribution to Red Hat, Inc. and provide a link to the original. If the document is modified, all Red Hat trademarks must be removed.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, JBoss, MetaMatrix, Fedora, the Infinity Logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux® is the registered trademark of Linus Torvalds in the United States and other countries.

Java® is a registered trademark of Oracle and/or its affiliates.

XFS® is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL® is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js® is an official trademark of Joyent. Red Hat Software Collections is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack® Word Mark and OpenStack Logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

Abstract

The OpenShift Enterprise User Guide helps developers set up and configure a workstation to develop and deploy applications in an OpenShift Enterprise cloud environment with a command-line interface (CLI), more commonly known as the client tools. This guide provides detailed instructions and examples to help developers: Create and manage domains and SSL certificates Create, build, and deploy applications Manage applications and cartridges Monitor and manage application storage and resources

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>. From the Customer Portal, you can:

- ✧ Search or browse through a knowledge base 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 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 User 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](#)

1.1. Basic Architecture

OpenShift Enterprise provides disk space, CPU resources, memory, network connectivity, and an Apache or JBoss server to create, deploy, and manage applications in the cloud. For most types of applications, OpenShift Enterprise creates a file system layout that you can use as a template for building an application. It also generates a limited Domain Name System (DNS) so your application is accessible online.

The following table describes the basic system components of OpenShift Enterprise.

Table 1.1. Basic Components

System Component	Description
<i>Broker</i>	Responsible for managing user logins, DNS, and application state. It manages the nodes and coordinates provisioning and application management. User interaction with the broker is performed using the management console, CLI, or a REST-based API.
<i>Nodes</i>	Hold the gears that run the applications, and enable resource sharing so that multiple gears can run on a single physical or virtual machine. This machine is referred to as a node host.
<i>Message Bus</i>	Link between the broker and the nodes.
<i>Gears</i>	Resource-constrained containers for application code where cartridges run. Gears determine the amount of RAM and disk space available to a cartridge.
<i>Cartridges</i>	Cartridges provide the functionality to run applications. Numerous cartridges are currently available to support languages such as Perl, PHP, and Ruby, as well as many database cartridges, such as PostgreSQL and MySQL.

Consult your system administrator for details about the gears sizes that are available. A typical default gear size is a small gear, which provides 512MB of RAM, 100MB of swap space, and 1GB of disk space.

[Report a bug](#)

1.2. User Interfaces

There are two mechanisms available for interacting with OpenShift Enterprise: the Management Console graphical user interface and the command line interface (CLI), referred to as the client tools.

[Report a bug](#)

1.2.1. Management Console

The OpenShift Enterprise Management Console is a graphical interface accessed with a web browser. Consult your system administrator for details on accessing the Console. This guide assumes that the Console is installed at **broker.example.com**.

The Management Console is best suited for:

- ✧ Setting up, administering and managing accounts
- ✧ Launching new applications
- ✧ Managing and monitoring applications

The following screenshot shows the home page of the Management Console when you first log into your account. Each tab across the top navigation bar provides further functionality to help you manage your account, applications, and more.

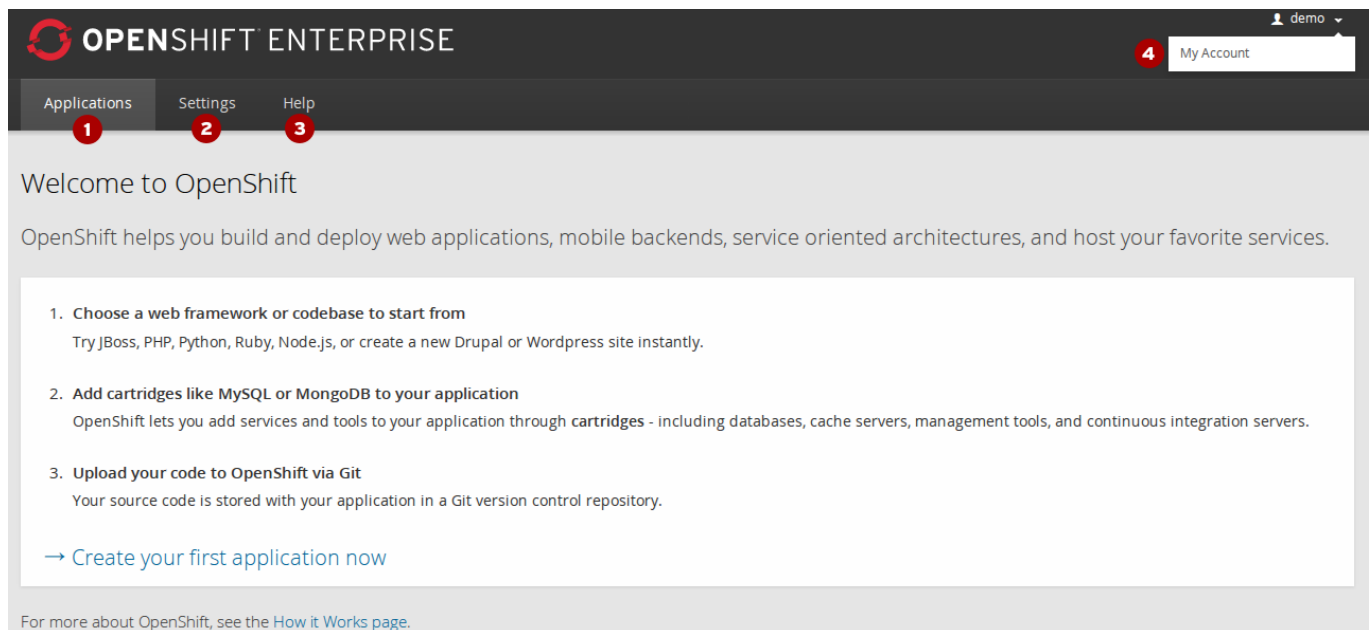




Figure 1.1. Management Console

The following table provides a brief description of the different pages and settings available in the Management Console.

	Page	Description
1	Applications	View and manage applications and cartridges. If there are no applications, you can create new applications from this page.
2	Settings	View and manage SSH keys, domains, and account authorizations.

	Page	Description
	Help	Access to KBase articles, community forums, tutorials, and other community resources. A wide variety of resources are available for diagnosing and resolving issues with your account or your applications.
	My Account	View and manage account information. This page shows account details and account usage.

Note that the Management Console currently provides limited functionality. Therefore, most of the instructions in this guide are for the client tools. However, tasks that can be performed in the Management Console are highlighted accordingly in their respective sections.

[Report a bug](#)

1.2.2. Client Tools

The client tools are used to manage an OpenShift Enterprise environment using a command line interface, and provide features that are not currently available in the Management Console.

The client tools are best suited for:

- » Coding
- » Debugging
- » Advanced application management

For example, although you can create an application using the Management Console, the application must be cloned to your workstation to make any code changes, and then redeployed to the remote server using the client tools.



Note

The ***rhc*** package found in the OpenShift Enterprise 2.0 Client Tools channel is based on the Red Hat Enterprise Linux 6 RPM version of the client tools, and not the Ruby gem version, which is updated more frequently. Therefore, some updated features may be temporarily only available for the Ruby gem version. See [Client Tools Installation Guide](#) to install the latest Ruby gem version of the client tools and get all available features.

[Report a bug](#)

1.3. What's New in OpenShift Enterprise 2.0

For a complete list of all the new features available in OpenShift Enterprise 2.0, see the current edition of *OpenShift Enterprise Release Notes* at <https://access.redhat.com/site/documentation>.

[Report a bug](#)

Chapter 2. Getting Started

2.1. OpenShift User Account

Before you can develop OpenShift Enterprise applications, you must first obtain an OpenShift Enterprise user account. Contact your system administrator for more information on how to set up an account, or see the *OpenShift Enterprise Deployment Guide* at <https://access.redhat.com/site/documentation> for more information. This guide assumes you already have an active OpenShift Enterprise user account.

[Report a bug](#)

2.2. Client Tools

The OpenShift Online client tools provide access to advanced management features currently not available in the Management Console. Most of the instructions provided in this guide assume that the client tools are already installed and configured on your workstation.

See the [Client Tools Installation Guide](#) for more information on how to install the client tools.

[Report a bug](#)

2.3. Basic Administration

2.3.1. Viewing Account Information

View basic information for an account with the following command:

```
$ rhc account
```

Example 2.1. Viewing Account Information

```
$ rhc account
Login User@example.com
-----
ID:                    52424geb2587c836b106001b
Gears Used:            10
Gears Allowed:         16
Domains Allowed:       3
Allowed Gear Sizes:    small
SSL Certificates:      yes
```

[Report a bug](#)

2.3.2. Ending Current Session

End the current session with the remote server and remove all local session files with the following command:

\$ rhc logout

[Report a bug](#)

Chapter 3. Authentication

3.1. Authorization Tokens

3.1.1. Introduction to Authorization Tokens

An *authorization token* is a secret value that is used to automatically log in to an OpenShift Enterprise account without entering login information each time. A token is also used to grant another user full or partial access to an account, determined by the *scope* of the token. The following table describes the different types of scopes available with authorization tokens.

Table 3.1. Authorization Token Scopes

Scope	Description	Validity
session	Access to all API functions against an account.	1 day
read	Read-only access to account resources, but cannot view authorization tokens.	1 month
userinfo	Access to login name, unique id, and user capabilities.	1 month

When the client tools are installed and the **rhc setup** command is initially run to configure the client tools, the setup wizard prompts you to create an authorization token. If you answer **YES**, the wizard creates a session token in the `~/ .openshift` directory. With this token, all client tool commands can be run without entering your login credentials each time. When the token expires you are automatically prompted to reenter login information to renew the existing token. See the [OpenShift Enterprise Client Tools Installation Guide](#) for more information on installing and configuring the client tools.

If an authorization token was not created when the client tools were installed, run the setup wizard again with the **rhc setup** command to create one.

If an existing authorization token is no longer required and you do not wish to be prompted for token renewal, run the **rhc logout** command to delete the token.

[Report a bug](#)

3.1.2. Creating Authorization Tokens

Create a new authorization token with the following command:

```
$ rhc authorization add --scopes Scope --note Name
```

Specify the scope for the token with the **--scopes** option, and a name for the token with the **--note** option.

Example 3.1. Creating an Authorization Token

```
rhc authorization add --scopes session --note My_Token
Adding authorization ... done

My_token
-----
Token:
```

```
787a57211d42f251204136b05d490038830d9b7057f54f816c2a9fcd0c8333b8
Scopes:      session
Created:     4:40 PM
Expires In:  about 1 day
```

After creating a new authorization token, use the `--token token_string` global option to run **rhc** commands as the user associated with the authorization token that was provided.

[Report a bug](#)

3.1.3. Viewing Authorization Tokens

View the tokens associated with your account with the following command:

```
$ rhc authorization list
```

Example 3.2. Viewing Authorization Tokens

```
$ rhc authorization list
My_token
-----
Token:
787a57211d42f251204136b05d490038830d9b7057f54f816c2a9fcd0c8333b8
Scopes:      session
Created:     4:40 PM
Expires In:  about 23 hours

RHC/1.8.0 (from laptop.example.com on x86_64-linux)
-----
Token:
28f6e375dc7ea57b0dcabb3850d08ee9bc023f7df5dbfa4958afe7ad71d33e37
Scopes:      session
Created:     12:58 PM
Expires In:  about 19 hours
```

[Report a bug](#)

3.1.4. Deleting Authorization Tokens

Delete authorization tokens when they are no longer required, or to end access to your account by other users:

Delete Specific Authorization Tokens

Delete one or more tokens with the following command, separating multiple tokens with commas:

```
$ rhc authorization delete token_1, token_2
```

Delete All Authorization Tokens

Delete all tokens associated with your account with the following command:


```
$ rhc authorization delete-all
```

[Report a bug](#)

3.2. SSH Keys

3.2.1. Introduction to SSH Keys

OpenShift Enterprise uses the Secure Shell (SSH) network protocol to authenticate account credentials to the OpenShift Enterprise servers for secure communication, and supports both RSA and DSA keys for SSH authentication. This section describes how authentication with OpenShift Enterprise works, and provides information on how to manage SSH keys for user accounts.

Successful authentication occurs when the private SSH key on your workstation matches the public key that has been uploaded to the OpenShift Enterprise server. When the client tools are initially configured, the interactive setup wizard generates a new pair of SSH keys in the default `.ssh` folder of your home directory. The SSH key pair consists of the public key, `id_rsa.pub`, and the private key, `id_rsa`. As part of the initial configuration, you have the option of automatically uploading the public key, `id_rsa.pub`, to the OpenShift Enterprise server. See the [Client Tools Installation Guide](#) for more information on how to configure the client tools.

The following table shows the types of SSH keys supported with OpenShift Enterprise.

Table 3.2. Supported SSH Keys

ssh-rsa
ssh-dss
ecdsa-sha2-nistp256-cert-v01@openssh.com
ecdsa-sha2-nistp384-cert-v01@openssh.com
ecdsa-sha2-nistp521-cert-v01@openssh.com
ssh-rsa-cert-v01@openssh.com
ssh-dss-cert-v01@openssh.com
ssh-rsa-cert-v00@openssh.com
ssh-dss-cert-v00@openssh.com
ecdsa-sha2-nistp256
ecdsa-sha2-nistp384
ecdsa-sha2-nistp521
krb5-principal

[Report a bug](#)

3.2.2. Generating Keys Manually

The following instructions describe how to generate a new pair of RSA or DSA keys.

1. Run the following command to generate a pair of keys, replacing *KeyType* with the type of key to generate:

```
$ ssh-keygen -t KeyType
```

2. Press **Enter** when prompted to save the key file in the default location:

```
...
Generating public/private rsa key pair.
Enter file in which to save the key (/home/username/.ssh/id_rsa):
/home/username/.ssh/id_rsa
```



Note

Red Hat recommends to save all SSH keys in the default location. If an ***id_rsa*** file already exists, rename the new SSH key file to avoid overwriting the existing one.

3. Enter a passphrase or leave blank when prompted, then press **Enter**:

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/username/.ssh/id_rsa
Your public key has been saved in /home/username/.ssh/id_rsa.pub.
...
```

[Report a bug](#)

3.2.3. Adding a Key

Once an SSH key has been generated, add the key by uploading it to the remote server with the following command, replacing *KeyName* and *KeyPath* with the name and path of the key to upload:

```
$ rhc sshkey add KeyName KeyPath
```

[Report a bug](#)

3.2.3.1. Adding a Specific SSH Key Type

After an SSH key is generated, rather than uploading the key file, add the contents of the key file directly to the remote server with the following command:

```
$ rhc sshkey add KeyName --type KeyType --content KeyContent
```

An SSH key is a long string of alphanumeric characters.

Adding a Kerberos Principal SSH Key

Add a Kerberos principal SSH key with the following command:

```
$ rhc sshkey add KeyName --type krb5-principal --content Principal
```

In contrast to a standard SSH key, a kerberos key is the kerberos principal: *username@domain.com*

[Report a bug](#)

3.2.4. Viewing All Public Keys

View a list of all public keys associated with an account with the following command:

```
$ rhc sshkey list
```

Example 3.3. Viewing All Public Keys

```
$ rhc sshkey list
libra (type: ssh-rsa)
-----
Fingerprint: 43:f5:29:ad:9f:b8:b3:a6:e7:88:c9:7f:4c:a9:0c:ad

winKey (type: ssh-rsa)
-----
Fingerprint: 0c:16:81:e3:51:eb:12:90:f6:03:80:g2:a2:10:78:14

default (type: ssh-rsa)
-----
Fingerprint: 43:f8:93:re:9f:a3:a8:f4:f3:34:g8:3d:1g:d8:3c:as
Available:   true

You have 3 SSH keys associated with your account.
```

[Report a bug](#)

3.2.5. Viewing a Specific Public Key

View details of a specific key with the following command, specifying the name of the key:

```
$ rhc sshkey show KeyName
```

[Report a bug](#)

3.2.6. Deleting a Key

Delete an existing public key from the remote server with the following command:

```
$ rhc sshkey remove KeyName
```

[Report a bug](#)

3.2.7. Resolving Authentication Issues

Occasionally, a local private key might not match the public key stored on the OpenShift Enterprise remote server, or the matching key might not be found on the local file system. This can cause connection issues, or the SSH key authentication process can fail, in which case a new pair of SSH keys must be generated. If you are having problems authenticating, generate a new pair of SSH keys in one of two ways:

- ✱ Use the interactive setup wizard (recommended)
- ✱ Manually generate and add SSH keys

[Report a bug](#)

3.2.7.1. Resolving Issues with Interactive Setup Wizard

Red Hat recommends resolving authentication issues with the interactive setup wizard to generate a new pair of SSH keys. The interactive setup wizard also provides the option to automatically upload a new public key to the OpenShift Enterprise server. Launch the interactive setup wizard with the **rhc setup** command and follow the onscreen instructions.

See the [Client Tools Installation Guide](#) for more information about the client tools and the interactive setup wizard.

[Report a bug](#)

3.3. Mutual SSL

Mutual SSL authentication, commonly referred to as x509 or two-way authentication, allows for an application developer, which is the SSL client, to authenticate to an application, which is the SSL server, and vice versa. Each side has a verification certificate, which is shared upon connection. Using mutual authentication ensures an additional level of security in your deployment, because without the approved authentication certificate a user is unable to connect to the SSL server.

If your system administrator has configured an OpenShift Enterprise deployment to require mutual SSL authentication, you must configure the client tools on your workstation to be able to make connections with that deployment. Contact your system administrator for details about the deployment's authentication requirements, or see the [OpenShift Enterprise Deployment Guide](#) for more information.

Procedure 3.1. To Configure the Client Tools for Mutual SSL Authentication:

1. Mutual SSL authentication requires the current version of the OpenShift client tools. Ensure your client tools are up to date using the method you used to install them. See the [OpenShift Enterprise Client Tools Installation Guide](#) for details.
2. If you are only connecting to one OpenShift Enterprise server, you can add the following to the `~/.openshift/express.conf` file on the workstation where the client tools are installed:

```
ssl_ca_file=path/to/SSLCA/certificate/file
ssl_client_cert_file=path/to/SSL/certificate/file
ssl_client_key_file=path/to/certificate/keyfile
insecure=false
use_authorization_tokens=true
libra_server=broker.example.com
```

Alternatively, you can perform the same configuration from the command line using the **rhc setup** command:

```
# rhc setup --ssl-client-cert-file path/to/SSL/certificate/file --
ssl-client-key-file path/to/certificate/keyfile --ssl-ca-file
path/to/SSLCA/certificate/file
```

If you are using your client tools to manage accounts for multiple OpenShift servers, you can use the **rhc server** command instead:

```
# rhc server add broker.example.com --nickname Server_Nickname --  
ssl-client-cert-file path/to/SSL/certificate/file --ssl-client-  
key-file path/to/certificate/keyfile --ssl-ca-file  
path/to/SSLCA/certificate/file
```

See [Chapter 4, Multiple OpenShift Servers](#) for general information on managing and switching between accounts for multiple OpenShift servers.

[Report a bug](#)

Chapter 4. Multiple OpenShift Servers

As outlined in the [OpenShift Enterprise Client Tools Installation Guide](#), previous versions of the client tools were configured for use with a single OpenShift server at a time. The configuration settings in the `~/ .openshift/express.conf` file would be overwritten each time the client tools were configured for use with a new server. Starting with OpenShift Enterprise 2.1.5, you can configure the client tools shipped with OpenShift Enterprise for use with multiple servers.

Configuring the client tools for multiple servers is useful for when you are using a single workstation to manage different applications hosted across multiple servers. For example, if your organization is using different OpenShift Enterprise servers for development, production, and testing, and you have an OpenShift Online account for personal use, you can manage applications on any of these servers from the same workstation without overwriting your client tools configuration each time.

[Report a bug](#)

4.1. Server Management

4.1.1. Adding a Server

Add an OpenShift server with the following command:

```
$ rhc server add server.name.example.com
```

This can be an OpenShift Enterprise, OpenShift Online, or OpenShift Origin server.

When adding a server, you can assign it a nickname to make it easier to reference in any future commands:

```
$ rhc server add server.name.example.com --nickname Server_Nickname
```

Note that when adding a server with the server name of **openshift.redhat.com**, the nickname defaults to **online**.

When adding a server, specifying a user name associates that user with the server, and you can have different user names with each server:

```
$ rhc server add server.name.example.com --nickname Server_Nickname --  
rhlogin user@company.com
```

The client tools automatically manage users when you switch between servers.

[Report a bug](#)

4.1.2. Switching Between Servers

When the client tools have been configured for multiple servers, switch to the desired server with the following command:

```
$ rhc server use server.name.example.com
```

If you assigned a nickname to a server, you can replace this with the server host name:

```
$ rhc server use Server_Nickname
```

[Report a bug](#)

4.1.3. Configuring Server Settings

When you run the **rhc setup** command for the first time, the `~/.openshift/express.conf` file is created, containing the settings for your initial server configuration. When you add another server, or run the **rhc setup** command again with the `--server` option, the `~/.openshift/servers.yml` file is created, containing the settings for each server. You can edit this file to make any changes to the server configuration, and this takes precedence over the initial `~/.openshift/express.conf` file.

Example 4.1. Contents of the `~/.openshift/servers.yml` File

```
-server:
  hostname: dev.openshift.company.com
  nickname: development
  login: user@company.com
  use_authorization_tokens: true
  insecure: true

-server
  hostname: openshift.redhat.com
  nickname: online
  login: user@personal.com
  use_authorization_tokens: true
  insecure: false
```

Configure any server settings with the following command, replacing *Server_Setting* with the desired server setting:

```
$ rhc server configure server.name.example.com Server_Setting
```

If you assigned a nickname to a server, you can replace this with the server host name:

```
$ rhc server configure Server_Nickname Server_Setting
```

You can use this command to configure any settings found in the `~/.openshift/servers.yml` file, such as the server nickname, the user name associated with the server, or any authorization tokens associated with the account.

Example 4.2. Changing the Nickname of a Server

```
$ rhc server configure dev.openshift.company.com --nickname dev
```

Example 4.3. Changing the Default User

```
$ rhc server configure dev --rhlogin username@example.com
```

[Report a bug](#)

4.1.4. Listing Servers

List any servers you have added with the following command:

```
$ rhc server list
```

Alternatively, you can list servers with the **rhc servers** command.

[Report a bug](#)

4.1.5. Viewing a Server

To see information about a specific server, use the following command:

```
$ rhc server show server.name.example.com
```

If you assigned a nickname to a server, you can replace this with the server host name:

```
$ rhc server show Server_Nickname
```

[Report a bug](#)

4.1.6. Removing a Server

Remove a server with the following command:

```
$ rhc server remove server.name.example.com
```

If you assigned a nickname to a server, you can replace this with the server host name:

```
$ rhc server remove Server_Nickname
```



Note

You cannot remove the server you are currently using.

[Report a bug](#)

Chapter 5. Domains

5.1. Introduction to Domains

An OpenShift Enterprise *domain* forms part of an application's URL and is unique to an account. The syntax for an application URL is *application-domain.example.com*. Each user name supports a single domain, but you can create multiple applications within the domain. Note that a domain must be created before you can create an application.

An OpenShift Enterprise blacklist restricts the domain names that are available. A warning message informs you if a blacklisted domain name has been selected when you attempt to create a domain.

Domain names consist of a maximum of 16 alphanumeric characters and cannot contain spaces or symbols.

[Report a bug](#)

5.2. Domain Management

5.2.1. Creating a Domain

A domain is required to create applications on OpenShift Enterprise. Create a new domain with the following command, specifying the name of the domain:

```
$ rhc domain create Domain_Name
```

The following example creates a domain named **automobile**.

Example 5.1. Creating a Domain

```
$ rhc domain create automobile
```

```
Creating domain 'automobile'
```

```
You may now create an application using the 'rhc app create' command
```



Note

The **Domains Allowed** setting determines the number of domains you can create. Use the **rhc account** command to view account settings. Contact your system administrator to request any changes to your account, or see the *OpenShift Enterprise Administration Guide* at <https://access.redhat.com/site/documentation> for more information.

See Also:

- ✱ [Section 11.10, “Custom Domains and SSL Certificates”](#)
- ✱ [Section 11.2, “Creating an Application”](#)

[Report a bug](#)

5.2.2. Listing Available Domains

List all available domains for an account with the following command:

```
$ rhc domain list
```

Example 5.2. Listing Available Domains

```
$ rhc domain list
Domain automobile
-----
Created:           Oct 01  7:28 PM
Allowed Gear Sizes: small, medium

Domain automobile2
-----
Created:           Oct 01  7:46 PM
Allowed Gear Sizes: small, medium
```

Alternatively, run the **rhc domains** command to list all available domains.

[Report a bug](#)

5.2.3. Viewing a Domain

View information about the default domain with the following command:

```
$ rhc domain show
```

Example 5.3. Viewing a Domain

```
$ rhc domain show
Domain automobile
-----
Created:           Oct 01  7:28 PM
Allowed Gear Sizes: small, medium

racer @ http://racer-automobile.example.com/ (uuid:
926056f8845b4e388b37f6735c89d0eb)
-----
Domain:  automobile
Created: Oct 01  7:28 PM
Gears:   2 (defaults to small)
Git URL: ssh://926056f8845b4e388b37f6735c89d0eb@racer-
automobile.example.com/~/.git/racer.git/
SSH:     926056f8845b4e388b37f6735c89d0eb@racer-
automobile.example.com
```

```
php-5.4 (PHP 5.4)
-----
Scaling: x2 (minimum: 2, maximum: 2) on small gears

You have 1 application in your domain.
```

If multiple domains exist, specify the name of the domain with the **-n** option:

```
$ rhc domain show -n Domain_Name
```

[Report a bug](#)

5.2.4. Renaming a Domain

When a domain is renamed, the old domain is deleted and a new one is created. Therefore, in order to prevent data loss, a domain cannot be renamed if it contains any applications.

The following instructions describe how to rename a domain.

1. Ensure the domain does not contain any applications with the following command:

```
$ rhc apps
```

Delete any applications that exist in that domain with the following command:

```
$ rhc app delete App_Name
```



Warning

Deleting an application deletes all remote data associated with that application, which cannot be recovered.

2. Rename a domain with the following command, specifying the current domain name and the new domain name:

```
$ rhc domain rename Old_Domain_Name New_Domain_Name
```

Example 5.4. Renaming a Domain

```
$ rhc domain rename olddomain newdomain
Renaming domain 'olddomain' to 'newdomain'... done
Applications in this domain will use the new name in their URL.
```

[Report a bug](#)

5.2.5. Deleting a Domain

The following instructions describe how to delete a domain if it is no longer required. However, note that a domain cannot be deleted if it contains any applications.

Procedure 5.1. To Delete a Domain:

1. Ensure the domain does not contain any applications with the following command:

```
$ rhc domain show Domain_Name
```

Delete any applications that exist in that domain with the following command:

```
$ rhc app delete App_Name
```



Warning

Deleting an application deletes all remote data associated with that application, which cannot be recovered.

2. Delete the domain with the following command:

```
$ rhc domain delete Domain_Name
```



Note

You must have at least one domain to create an application.

See Also:

» [Section 5.2.1, “Creating a Domain”](#)

[Report a bug](#)

5.2.6. Configuring Domain Gear Size

When configuring gear sizes for a domain, there are multiple options available. For example, you could limit the amount of resources available to a domain by restricting it to only allow small gears.

Table 5.1. Options When Configuring Domains

Option	Description
--no-allowed-gear-sizes	Does not allow any gear sizes in this domain.
--allowed-gear-sizes [SIZES]	Gear sizes to be allowed in this domain. To specify multiple sizes, use a comma-delimited list. To see available sizes, run the rhc account command.
-n, --namespace [NAME]	Name of a domain.

Configure gear sizes for a domain with the following command, specifying the desired gear sizes and the name of the domain:

```
$ rhc domain-configure domain --allowed-gear-sizes gearsizes
```

The following example shows the domain gear size configured to be small.highcpu.

Example 5.5. Configuring the Gear Size of a Domain

```
$ rhc domain-configure mydomain --allowed-gear-sizes small.highcpu
```

[Report a bug](#)

Chapter 6. Teams

6.1. Introduction to Teams

With the release of OpenShift Enterprise 2.1, multiple developers can be grouped together in a *team*. Domain access granted to a team applies to all of its members. In other words, a team counts as one member of a domain with the same permissions that a standard domain member would have, and you control and manage a team.

There are also global teams that are controlled and managed by a system administrator. For the ability to view and search global teams, contact your system administrator, or see the [OpenShift Enterprise Administration Guide](#) for more information.

Domain members with an administrator role can change the role of a team that is a member of that domain.

Teams and Roles

You can have explicit roles within a domain, and belong to a team which has a role within the domain. The following team roles are available: **view**, **edit**, and **admin**. If you have a specific role, and you are on a team that has a different role, the effective role is the higher of the two roles. Therefore, the following guidelines apply:

- If you have the **view** role in a domain, and you are not on a team, you can view the domain.
- If you are on a team that has the **view** role in a domain, you can view applications within that domain.
- If you have the **edit** role within a domain, and you are on a team that has the **view** role, you can edit applications within the domain.
- If you have the **view** role in a domain, and you are on a team that has the **edit** role, you can edit applications within the domain.
- If you do not have an explicit role in a domain, and you are on a team that has the **edit** role, you are not listed in the domain membership, except within the team.

[Report a bug](#)

6.2. Team Management

6.2.1. Creating a Team

Create a team with the following command:

```
$ rhc team create Team_Name
```

A team name must be a unique name between 2 and 250 characters, and cannot be modified once created.

**Note**

Consult your system administrator to determine the number of teams that you can create.

[Report a bug](#)

6.2.2. Adding Members to a Team

Add a member to a team by using the following command:

```
$ rhc member add user@myemail.com -t Team_Name
```

**Note**

You cannot add more than 100 members to a team.

[Report a bug](#)

6.2.3. Listing Your Teams

List the teams you are a member of with the following command:

```
$ rhc teams
```

You can exclusively list the teams you own by adding the **--mine** command:

```
$ rhc teams --mine
```

[Report a bug](#)

6.2.4. Viewing Team Information

View information about a team with the following command:

```
$ rhc team show Team_Name
```

You can also view information about a team by specifying the team ID:

```
$ rhc team show --team-id Team_ID
```

[Report a bug](#)

6.2.5. Leaving a Team

Remove yourself from a team with the following command:

```
$ rhc team leave Team_Name
```

You can also leave a team by specifying the team ID:

```
$ rhc team leave --team-id Team_ID
```



Note

You cannot leave a global team or any team that belongs to you.

[Report a bug](#)

6.2.6. Deleting a Team

Delete a team you own with the following command:

```
$ rhc team delete Team_Name
```

You can also delete a team by specifying the team ID:

```
$ rhc team delete --team-id Team_ID
```



Note

You can only delete a team if you are the owner.

[Report a bug](#)

Chapter 7. Domain Membership

7.1. Introduction to Domain Membership

Developers can collaborate on application development with domain membership. The following table describes the three roles that are available in domain membership.

Table 7.1. Domain Membership Roles

Role	Description
View	Member has read-only access to view information about the domain and its applications and cannot make any changes.
Edit	Member can create, update, and delete all applications in the domain, and has Git and SSH access.
Administer	Member has access to all features, but cannot change allowed gear sizes or edit the domain name.

The default role for each member is the **edit** role, but it can be changed.

[Report a bug](#)

7.2. Managing Domain Membership

7.2.1. Adding a Member

Add a user to a domain with the following command, specifying the user login and domain name. The user login must be a registered OpenShift Enterprise user.

```
$ rhc member add user@myemail.com -n Domain_Name
```

When a member is added to a domain, they receive the default role of **edit**. Use the **--role** option when adding a member to specify a different role:

```
$ rhc member add user@myemail.com -n Domain_Name --role Member_Role
```

Adding a Team to a Domain

When adding a team to a domain, use the **--type** option with **team** specified:

```
$ rhc member add Team_Name -n Domain_Name --type team
```

A global team can be added with the **--global** option:

```
$ rhc member add Global_Team_Name -n Domain_Name --type team --global
```

You can also add a team by specifying the team ID:

```
$ rhc member add Team_ID -n Domain_Name --type team --ids
```

**Note**

As with adding a member, use the **--role** option.

[Report a bug](#)

7.2.2. Changing Member Role

Change an existing member's role with the following command. *Member_Role* can be specified as **view**, **edit**, or **admin**:

```
$ rhc member update user@myadmin.com -n Domain_Name --role Member_Role
```

Changing Team Role

Change an existing team's role by using the **--type** option and specifying **team**:

```
$ rhc member update Team_Name -n Domain_Name --role Member_Role --type  
team
```

Or perform the same function using team IDs:

```
$ rhc member update Team_ID -n Domain_Name --role Member_Role --type  
team --ids
```

[Report a bug](#)

7.2.3. Listing Members of a Domain

View the existing members of a domain with the following command, specifying the name of the domain:

```
$ rhc member list Domain_Name
```

Example 7.1. Listing Domain Members

```
$ rhc member list automobile
Login                Login      Role  Type
-----
member@example.com member@example.com admin (owner) user
myteam              edit      team
member2@example.com member2@example.com view   user
member3@example.co member3@example.com edit   user
member4@example.com member4@example.com admin  user
```

Use the **--all** option to display all members, including team-members:

```
$ rhc member list Domain_Name --all
```

Example 7.2. Listing Domain Members

```
$ rhc member list automobile --all
Login                               Login      Role      Type
-----
member@example.com member@example.com admin (owner) user
myteam edit team
member2@example.com member2@example.com view user
member3@example.co member3@example.com edit user
member4@example.com member4@example.com admin user
team_member1@example.com team_member1@example.com edit (via myteam)
user
team_member2@example.com team_member2@example.com edit (via myteam)
user
team_member3@example.com team_member3@example.com edit (via myteam)
user
```

[Report a bug](#)**7.2.4. Listing Members of an Application**

View the existing members of an application with the following command, specifying the application name with the **-a** option:

```
$ rhc member list -a App_Name
```

Example 7.3. Listing Application Members

```
$ rhc member list -a myapp
Login                               Role
-----
user1@myemail.com admin (owner)
user2@myemail.com view
```

[Report a bug](#)**7.2.5. Removing a Member**

Remove an existing member from a domain with the following command, specifying the domain name with the **-n** option and the user name to be removed:

```
$ rhc member remove -n Domain_Name user@myemail.com
```

Alternatively, remove all existing members from a domain by including the **--all** option:

```
$ rhc member remove -n Domain_Name --all
```

Removing a Team

Remove a team from a domain by specifying the team name and adding the **--type** option:

```
$ rhc member remove Team_Name -n Domain_Name --type team
```

[Report a bug](#)

Chapter 8. Regions and Zones

8.1. Introduction to Regions and Zones

With the release of OpenShift Enterprise 2.1, you can group nodes into regions and zones. Regions and zones provide a way for brokers to manage several distinct geographies by controlling application deployments across a selected group of nodes. You can group nodes into zones and group zones into regions. These groups can represent physical geographies, such as different countries or data centers, or can be used to provide network level separation between node environments. Using regions, you can maximize your application performance with less latency by deploying applications closer to your expected users.

Only the system administrator can configure regions and zones, but application developers can create applications in specific regions, view region information when accessing a gear, and can list the regions available. Contact your system administrator, or see the [OpenShift Enterprise Administration Guide](#) for more information on regions and zones.

[Report a bug](#)

8.2. Region and Zone Management

8.2.1. Listing Available Regions

As an application developer, you can add applications into a specific region upon creation. You can list the existing regions with the following command:

```
$ rhc region list
```

Alternatively, you can list the available regions and zones with the **rhc regions** command.

See Also:

» [Section 11.2, “Creating an Application”](#)

[Report a bug](#)

Chapter 9. Cartridges

9.1. Introduction to Cartridges

Cartridges are the components of an OpenShift Enterprise application and contain the application code to provide the actual functionality required to run applications. Cartridges are available to support various programming languages, databases, monitoring services, and management.

Adding a cartridge to an application provides the desired capability without having to administer or update the included feature. When added to an application, a cartridge is deployed to one or more gears based on its requirements. Cartridges that listen to incoming traffic are placed on one or more gears, while other cartridges can be placed across multiple gears of an application.

[Report a bug](#)

9.1.1. Web Framework Cartridges

Web cartridges are available for a variety of programming languages and frameworks, and an application requires at least one web cartridge to listen to HTTP requests. The type of web framework cartridge must be specified when an application is created. Cartridges that listen to incoming traffic are placed on one or more gears, while other cartridges can be placed across multiple gears of an application.

The following web framework cartridges are currently available with OpenShift Enterprise 2:

Table 9.1. Available Web Framework Cartridges

Cartridge	Scalable	Version
Do-It-Yourself (DIY)	No	-
JBoss A-MQ	No	6.1 [a]
JBoss EAP	Yes	6.2
JBoss Fuse	No	6.1 [a]
JBoss Fuse Builder	Yes	6.1 [a]
Jenkins Server	No	1.5
Node.js	Yes	0.10
Perl	Yes	5.10
PHP	Yes	5.3, 5.4 [b]
Python	Yes	2.6, 2.7, 3.3 [c]
Ruby	Yes	1.8, 1.9, 2.0 [d]
Tomcat (JBoss EWS)	Yes	6, 7
[a] JBoss A-MQ 6.1, JBoss Fuse 6.1, and JBoss Fuse Builder 6.1 are available starting in OpenShift Enterprise 2.1.7		
[b] PHP 5.4 is available starting in OpenShift Enterprise 2.1.		
[c] Python 3.3 is available starting in OpenShift Enterprise 2.1.1.		
[d] Ruby 2.0 is available starting in OpenShift Enterprise 2.2.		

[Report a bug](#)

9.1.2. Add-on Cartridges

After an application is created with the required web framework cartridge, a number of add-on cartridges can provide extra functionality and capabilities to applications, such as databases, scheduled jobs, or continuous integration. The following table describes the functionality of the different types of add-on cartridges available with OpenShift Enterprise.

Table 9.2. Add-on Cartridge Functions

Function	Description
Database	Provide the application with one of several database back ends. Examples include MySQL and PostgreSQL.
Monitoring and Management	Provide a range of options for managing and monitoring the application. Examples include the Cron task scheduler, and the Jenkins Client.

The following add-on cartridges are currently available for OpenShift Enterprise.

Database Cartridges

The following table describes all available database cartridges.

Table 9.3. Database Cartridges

Cartridge	Scalable	Version	Description
MySQL	No	5.1, 5.5 [a]	Multi-user, multi-threaded SQL database server.
MongoDB	No	2.4 [b]	High-performance, open source NoSQL database.
PostgreSQL	No	8.4, 9.2	Advanced object-relational database management system

[a] MySQL 5.5 is available starting in OpenShift Enterprise 2.1.
[b] MongoDB 2.4 is available starting in OpenShift Enterprise 2.1.1.



Important

While at this time of writing database cartridges are not scalable, they can be added to scalable applications.

Monitoring and Management Cartridges

The following table describes all available management cartridges, and shows whether they are scalable or not.

Table 9.4. Monitoring and Management Cartridges

Cartridge	Scalable	Version	Description
HAProxy	Yes	1.4	High performance TCP/HTTP load balancer.
Cron	Yes	1.4	A daemon that runs specified programs at scheduled times.
Jenkins Client	No	1.5	A client for managing Jenkins-enabled applications.

[Report a bug](#)

9.1.3. Downloadable Cartridges

Downloadable cartridges are available for new and existing applications along with the supported standard OpenShift Enterprise cartridges. These are custom cartridges created by users, or available from the OpenShift community. These cartridges are downloaded and installed using the URL to the manifest of the hosted downloadable cartridge.

Visit <https://www.openshift.com/developers/download-cartridges> for more community tips and information on downloadable cartridges.

See Also:

- » [Section 11.2, “Creating an Application”](#)

[Report a bug](#)

Chapter 10. Applications

10.1. Introduction to Applications

When a new application is created, a URL with name of the application and the name of the domain is registered in DNS. A copy of the application code is checked out locally into a folder with the same name as the application. Note that different types of applications may require different folder structures. Application components are run on gears.

With each new application that is created with the client tools, a remote Git repository is populated with the selected cartridge, which is then cloned to the current directory on the local machine. The host name and IP address of the application are also added to the list of known hosts in the `~/ .ssh/known_hosts` directory.

The following table describes each component that makes up an OpenShift Enterprise application.

Table 10.1. Application Components

Component	Description
Domain	The domain provides a unique group identifier for all the applications of a specific user. The domain is not directly related to DNS; instead, it is appended to the application name to form a final application URL of the form <code>http://App_Name-domain.example.com</code>
Application Name	The name of the application is selected by a user. The final URL to access the application is of the form <code>http://App_Name-domain.example.com</code>
Alias	DNS names can be provided for the application by registering an <i>alias</i> with OpenShift Enterprise and pointing the DNS entry to the OpenShift Enterprise servers.
Git repository	A Git repository is used to modify application code locally. After the code is applied, the git push command is required to deploy the revised code.

OpenShift Enterprise provides dedicated `/var/tmp` and `/tmp` directories for each user application. The `/var/tmp` directory is a symbolic link to `/tmp`. Each `/tmp` directory is completely isolated from the `/tmp` directories of all other applications. Files that are untouched for any period of ten days are automatically deleted from these directories.

[Report a bug](#)

10.1.1. Application Life Cycle

The following table describes the general life cycle of most OpenShift Enterprise applications.

Table 10.2. Application Life Cycle

Process	Description
Code	Develop the application code with the desired language and tools. Continuously push the application code to the applications remote Git source code repository.
Build	OpenShift Enterprise supports various build mechanisms, whether it is a simple script, a personal Jenkins continuous integration server, or an external build system.

Process	Description
Deploy	Every application is composed of cartridges that simplify server maintenance and configuration. OpenShift Enterprise supports various technologies to provision the required services automatically.
Manage	OpenShift Enterprise allows real-time monitoring, debugging, and tuning of applications. Applications are scaled automatically depending on web traffic.

[Report a bug](#)

10.1.2. Scalable Applications

Applications can be created as either scalable or not scalable. Scalable applications feature the load-balancing proxy (HAProxy) gear, which is contained on the same gear as the web framework cartridge. The load-balancing proxy provides horizontal scaling by cloning the gears containing the framework cartridge and application data onto multiple gears.

Scalable applications are set to scale automatically by default. However, you can scale an application manually. See [Section 11.6, “Scaling an Application Manually”](#) for more information.

How Auto-scaling Works

Each application created on OpenShift Enterprise must have one web framework cartridge defined upon creation. For example, a PHP cartridge. When an application is defined as scalable, a second cartridge, the HAProxy cartridge, is added to the application. The HAProxy cartridge listens to all incoming web-page requests to an application and passes them to the web cartridge, following the defined guidelines for load monitoring.

When the number of web-page requests to an application increases, the HAProxy informs OpenShift Enterprise when an overload of requests is detected. A copy of the existing web cartridge and application data is then cloned to a separate gear. In such a case, the web cartridge has now been scaled up two times. This process is repeated as more web-page requests are detected by the HAProxy cartridge, and each time a copy of the web cartridge is created on a separate gear, the application scale factor increases by one.

When the application (by default) reaches three copies of the web framework cartridge, the HAProxy load-balancer disables routing to the framework cartridge located on the same gear. This gives the HAProxy cartridge full gear resource usage, which continues to route requests to the framework cartridges located on separate gears. Routing to the framework cartridge located with the HAProxy cartridge is enabled again once the application is scaled down.



Figure 10.1. Cartridges on Gears in a Scaling Application

[Report a bug](#)

10.1.3. Highly-Available Applications

Scalable applications duplicate application code onto multiple gears within your application. Any requests to your application go through the node router to the head (HAproxy) gear, which distributes requests to available framework gears.

Alternatively, when a scalable application is configured to be highly available, at least one load-balancer gear is ensured to survive in the event of an outage and the application remains accessible. High availability provides at least two load-balancing (HAproxy) gears on separate node hosts and any requests instead go through an external routing layer, bypass the node router, and are sent directly to HAproxy gears to be distributed to framework gears. During an outage, the routing layer ensures incoming traffic reaches an available HAproxy gear, and the HAproxy gears only route traffic to the framework gears that are available.

While you, as an application developer, can enable your application to be highly available, the OpenShift Enterprise infrastructure and configuration must be set to support highly-available applications. Contact your system administrator, or see the [OpenShift Enterprise Administration Guide](#) for more information on highly-available applications and the infrastructure needed to support them.

Enabling High Availability in Applications

At this time of writing, the only way to enable high availability in applications is through the REST API. See [Section 11.7, “Making Applications Highly Available”](#) for the curl command to make your applications highly available, or the relevant section in the *OpenShift Enterprise REST API Guide* for more information:

https://access.redhat.com/documentation/en-US/OpenShift_Enterprise/2/html/REST_API_Guide/Make_an_Application_Highly_Available_HA.html

[Report a bug](#)

Chapter 11. Application Management

11.1. General Information

A reliable network connection is required because only a single attempt is made to create an application. OpenShift Enterprise makes seven attempts to see if the DNS entry for the new application exists. If it is not found an error message is returned.

The `--timeout` option on the command line is used to override the default values when there are constant timeout issues. OpenShift Enterprise uses two timeout parameters: a *connection* timeout, which determines how long the client tries to connect to the server before timing out; and a *read* timeout, which determines how long the client waits for a response from the server. The default connection timeout value is 20 seconds. The default read timeout value is 120 seconds.

The `--timeout` option affects both timeout parameters, but it can only be used to increase the default values. The timeout value cannot be set to be less than the default. For example, if `--timeout 50` is used, it sets the connection timeout value to 50 seconds, but does not affect the read timeout value. Similarly, if `--timeout 150` is used, it sets both the connection and read timeout values to 150 seconds.

[Report a bug](#)

11.2. Creating an Application

Prerequisites:

✱ [Section 5.2.1, “Creating a Domain”](#)

There are some factors that must be considered before you create an application. There are certain aspects of the application that cannot be changed after it is created. For example, whether an application is scalable or not must be specified when it is created. An application that is not scalable cannot be changed to scalable after it is created, and vice versa. The web framework of a cartridge also cannot be changed after an application is created. An application can be created with either the Management Console or the client tools.

New applications are created with the `rhc app create` command and using the command options to supply the required information, such as the type of web framework to be used with the new application. Note that if multiple versions are available for the specified web framework cartridge, you are prompted to specify the version number to use for the new application.

The following table describes some of the common options available when creating a new application with the client tools.

Table 11.1. Options When Creating New Applications

Option	Description
<code>-n, --namespace [NAME]</code>	Domain where you wish to create the application.
<code>-g, --gear-size [SIZE]</code>	Gear size determines how much memory and CPU a cartridge consumes.
<code>-s, --scaling</code>	Creates a scalable application.
<code>--from-code [URL]</code>	URL to a Git repository that becomes the initial contents of the application.
<code>-a, --app [NAME]</code>	Name for the application to be created.

Option	Description
<code>--enable-jenkins [NAME]</code>	Enables Jenkins continuous integration, and creates a Jenkins application if one does not already exist. The default name is 'jenkins' if a name is not specified.
<code>--region [NAME]</code>	Specifies the region in which the application will be created.

Creating a Non-Scalable Application

Create a non-scalable application in the default domain with the following command:

```
$ rhc app create App_Name Cart_Name
```

Example 11.1. Creating a Non-Scalable Application

```
$ rhc app create racer php-5.4
Application Options
-----
Domain:      mydomain
Cartridges:  php-5.4
Gear Size:   default
Scaling:     no

Creating application 'racer' ... done

Waiting for your DNS name to be available ... done

Cloning into 'racer'...
The authenticity of host 'racer-mydomain.rhcloud.com (50.19.129.28)'
can't be established.
RSA key fingerprint is cf:ee:77:cb:0e:fc:02:d7:72:7e:ae:80:c0:90:88:a7.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'racer-mydomain.rhcloud.com,50.19.129.28'
(RSA) to the list of known hosts.

Your application 'racer' is now available.

URL:          http://racer-mydomain.rhcloud.com/
SSH to:       52ae91b8dbd93c8c43000001@racer-mydomain.rhcloud.com
Git remote:   ssh://52ae91b8dbd93c8c43000001@racer-
mydomain.rhcloud.com/~/.git/racer.git/
Cloned to:    /home/blank/racer

Run 'rhc show-app racer' for more details about your app.
```

Creating a Scalable Application

Create a scalable application by adding the `-s` parameter to the command:

```
$ rhc app create App_Name Cart_Name -s
```

Example 11.2. Creating a Scalable Application

```
$ rhc app create racer php-5.4 -s
```

With a scalable application the automatic scaling feature is enabled by default. However, an application can be scaled manually to control the number of gears that are used.

**Note**

At the time of this writing, if a scalable application is created, the scaling function of that application cannot be disabled. However, it is possible to clone a non-scaling application and all its associated data and create a new scaling application using the application clone command. See [Section 11.3, “Cloning an Existing Application”](#) for more information.

Creating an Application from a Downloadable Cartridge

Replace the web framework type with the URL of the manifest for the hosted cartridge to create an application from a downloadable cartridge:

```
$ rhc app create App_Name https://www.example.com/manifest.yml
```

Creating an Application in a Specific Domain

As described in [Section 5.2.1, “Creating a Domain”](#), each domain supports multiple applications. Therefore, if there are multiple domains associated with an account, you must specify in which domain to create the new application with the **-n** option:

```
$ rhc app create App_Name Cart_Name -n Domain_Name
```

When multiple applications are created in a domain, the application URLs are as follows:

- http://app1-domain.example.com
- http://app2-domain.example.com

Creating an Application With Jenkins Continuous Integration

Create an application and enable Jenkins continuous integration:

```
$ rhc app create App_Name Cart_Name --enable-jenkins Jenkins_App_Name
```

This command creates a Jenkins application, and then adds the Jenkins client cartridge to the specified application.

Example 11.3. Creating an Application With Jenkins Continuous Integration

```
$ rhc app create mynewapp php-5.4 --enable-jenkins myjenkinsapp
Application Options
-----
Domain:      mydomain
Cartridges:  php-5.4
```

```
Gear Size: default
Scaling:    no
```

```
Creating application 'mynewapp' ... done
```

```
Setting up a Jenkins application ... done
```

```
Jenkins created successfully. Please make note of these credentials:
```

```
User: admin
Password: wEXesNXyEe1M
```

```
Note: You can change your password at: https://myjenkinsapp-
mydomain.rhcloud.com/me/configure
```

```
Setting up Jenkins build ... done
```

```
Associated with job 'mynewapp-build' in Jenkins server.
```

```
Waiting for your DNS name to be available ... done
```

```
Cloning into 'mynewapp'...
```

```
The authenticity of host 'mynewapp-mydomain.rhcloud.com
(54.234.56.174)' can't be established.
```

```
RSA key fingerprint is cf:ee:77:cb:0e:fc:02:d7:72:7e:ae:80:c0:90:88:a7.
```

```
Are you sure you want to continue connecting (yes/no)? yes
```

```
Warning: Permanently added 'mynewapp-
mydomain.rhcloud.com,54.234.56.174' (RSA) to the list of known hosts.
```

```
Your application 'mynewapp' is now available.
```

```
URL:          http://mynewapp-mydomain.rhcloud.com/
SSH to:       52b10d7d2587c8415000012c@mynewapp-mydomain.rhcloud.com
Git remote:   ssh://52b10d7d2587c8415000012c@mynewapp-
mydomain.rhcloud.com/~/.git/mynewapp.git/
Cloned to:    /home/blank/mynewapp
```

```
Run 'rhc show-app mynewapp' for more details about your app.
```



Important

Take note of the login credentials for the newly created Jenkins application. These credentials are required to log in to the Jenkins home page.

Creating an Application in a Specific Region

OpenShift Enterprise supports grouping nodes into regions and zones. With the release of OpenShift Enterprise 2.2, you can create an application and assign it to a specific region using the **--region** option:

```
$ rhc app create App_Name Cart_Name --gear-size Gear_Size --region Region_Name
```

Example 11.4. Creating a Scalable Application into a Specific Region

```
$ rhc app create racer php-5.4 --gear-size medium --region eu-west-1a --scaling
```

If a region is not specified, one is picked randomly based on the regions available.

Creating an Application with Custom Code

To create an application using code from a Git repository, specify the URL:

```
$ rhc app create --from-code URL
```

The code from the specified URL becomes the initial contents of the application.

Creating an Empty Application

For build or other testing purposes, create an application of no specific type with the DIY cartridge:

```
$ rhc app create App_Name diy
```

The DIY cartridge creates an application that is not publicly available nor does it have anything running. Start the application with **git push** and a **.openshift/action_hooks/**.



Note

When an application is created, automatic deployment is configured by default. This means that each time you execute the **git push** command the application is automatically deployed and visible to customers.

See Also:

- [Section 13.1, “Introduction to Deployment”](#)
- [Section 13.3.1.1, “Configuring Automatic Deployment”](#)
- [Section 13.3.2.1, “Configuring Manual Deployment”](#)

[Report a bug](#)

11.3. Cloning an Existing Application

With the release of OpenShift Enterprise 2.1, you can now clone an existing application using the client tools. Create an application from existing application data using the following command:

```
$ rhc app create New_Name --from-app App_Name
```


This creates a new application using the same cartridges, gear sizes, and scaling and deployment configurations as an already existing application. Note that aliases are not copied, because they are unique to an application.

Cloning an Application from Another Domain

Use **domain/** to clone an application from another domain you have access to:

```
$ rhc app create New_Name --from-app domain/App_Name
```

Cloning an Application to a Specific Region

OpenShift Enterprise supports hosting applications in multiple regions and zones. With the release of OpenShift Enterprise 2.2, you can clone an application to a region other than us-east-1 using the **-region** option:

```
$ rhc app create New_App --from-app Existing_App --region Region_Name
```

The following table outlines different options you can use to configure the new application and give it different attributes than the original application:

Table 11.2. Application Clone Command Options

Option	Description
<code>--gear-size <i>Gear_Size</i></code>	Use this option to change the gear size of the new application. For example, if the original application uses the small gear size, use this option with <i>medium</i> for the new application to use medium gears.
<code>--[no-]scaling</code>	Use this option to configure the new application to be either scaling or non-scaling. For example, if the original application has scaling enabled, use the <code>--no-scaling</code> option to disable scaling for the new application and vice-versa.
<code>--env <i>En_var</i></code>	A cloned application will have the same environment variables as the original application. To add new environment variables to the new application, use this option with any desired environment variables. Additionally, you can override any environment variables that were set in the original application with this option and the environment variables to replace them.
<code>--no-git</code>	Use this to disable Git for the new application.
<code>--region <i>Region_Name</i></code>	Specifies the region to which the application will be cloned.

[Report a bug](#)

11.4. Cloning the Remote Application Repository

The remote application repository is not cloned to your local machine when an application is created with the Management Console. Therefore, it must be manually cloned so that the application code can be modified as required.

Clone the remote repository of an application into a local working directory with the following command:

```
$ rhc git-clone App_Name
```

Example 11.5. Cloning the Remote Application Repository

```
$ rhc git-clone racer
Cloning into 'racer'...
Your application Git repository has been cloned to '/home/apps/racer'
```

This command copies the template application files from the remote repository into the working directory so that the application code and files can be modified to suit your requirements.

[Report a bug](#)

11.5. Viewing Applications for a User

If you have created applications, view a list of all your applications with the following command:

```
$ rhc apps
```

Example 11.6. Viewing Applications for User

```
$ rhc apps
racer @ http://racer-automobile.example.com/ (uuid:
926056f8845b4e388b37f6735c89d0eb)
-----
Domain:      automobile
Created:     Dec 19 10:20 PM
Gears:       1 (defaults to small)
Git URL:     ssh://926056f8845b4e388b37f6735c89d0eb@racer-
automobile.example.com/~/.git/racer.git/
SSH:         926056f8845b4e388b37f6735c89d0eb@racer-
automobile.example.com
Deployment:   auto (on git push)

php-5.4 (PHP 5.4)
-----
Gears: 1 small
```

Use the `--mine` option to only list the applications that are within domains that you have created:

```
$ rhc apps --mine
```

Applications that you have access to, but did not create, will not be listed.

[Report a bug](#)

11.6. Scaling an Application Manually

Scalable applications can be manually scaled for various reasons that include:

- ✧ If a certain load is anticipated on an application and it must be scaled accordingly.
- ✧ There are a fixed set of resources for an application.
- ✧ The cost must be controlled manually.

Procedure 11.1. To Scale an Application Manually:

1. View the cartridges associated with a scalable application with the following command:

```
$ rhc app show App_Name
```

Example 11.7. Showing a Cartridge's Information

```
$ rhc app show hybrid
hybrid @ http://hybrid-automobile.example.com/ (uuid:
fjoe04cabdc4efa8f2513a21e2ed27d)
-----
Domain:    automobile
Created:   11:48 AM
Gears:     1 (defaults to small)
Git URL:   ssh://fjoe04cabdc4efa8f2513a21e2ed27d@hybrid-
automobile.example.com/~/.git/hybrid.git/
SSH:       fjoe04cabdc4efa8f2513a21e2ed27d@hybrid-
automobile.example.com
Deployment: auto (on git push)

php-5.4 (PHP 5.4)
-----
Scaling: x1 (minimum: 1, maximum: available) on small gears

haproxy-1.4 (OpenShift Web Balancer)
-----
Gears: Located with php-5.4
```

Locate the scaling cartridges as required. The example shows that the php-5.4 cartridge is scaling.

2. Set the minimum and maximum amount of gears the cartridge can use for scaling with the following command, specifying the application name and minimum and maximum number of gears:

```
$ rhc cartridge scale Cart_Name -a App_Name --min Min_Gears --max
Max_Gears
```

Example 11.8. Setting the Maximum and Minimum Amount of Gears for a

```
$ rhc cartridge scale php -a hybrid --min 1 --max 10
Setting scale range for php ... done

php-5.4 (PHP 5.4)
-----
Scaling: x1 (minimum: 1, maximum: 10) on small gears
```

Set the minimum and maximum gears back to **1** to stop a cartridge from scaling.

[Report a bug](#)

11.7. Making Applications Highly Available

When a scalable application is configured to be highly available, it provides multiple HAProxy gears to which an external routing layer can route traffic. This prevents any downtime.

To use this feature, you first must have the capability to make your application highly available. Contact your system administrator for more information.

After your system administrator has enabled this capability on your account, you can then enable scalable applications to be highly available using the client tools:

```
# rhc app enable-ha App_Name
```

Alternatively, you can make your application highly available using the following REST API call:

```
$ curl -X POST
https://Hostname/broker/rest/domains/Domain_Name/applications/App_Name/e
vents --user Username:Password --data-urlencode event=make-ha
```

[Report a bug](#)

11.8. Application Management Commands

Applications are managed with the client tools using the **rhc app** command and the available options. The following example shows the command syntax.

```
$ rhc app action App_Name [Options]
```

The following table describes the available application management command actions:

Table 11.3. Application Management Command Argument Options

Action	Details
start	Start an application.
stop	Stop an application.
force-stop	Stop all application processes.
restart	Restart an application.
reload	Reload an application.

Action	Details
show	Show information about an application.
tidy	Clean out the application's log files and tmp directories, and tidy up the Git repository on the server.
create	Create an application and add it to a domain.
delete	Remove an application.
configure	Configure an application's properties.
deploy	Deploy a Git reference or binary file of an application.

The following table describes the available options with application management commands:

Table 11.4. Application Management Command Options

Option	Description
-n, --namespace [NAME]	Name of a domain.
-a, --app [NAME]	Name of an application.
-l, --rhlogin [LOGIN]	OpenShift Enterprise user account.
-p, --password [PASSWORD]	OpenShift Enterprise account password.

Example 11.9. Starting an Application

```
$ rhc app start myapp -n mydomain
RESULT:
myapp started
```

[Report a bug](#)

11.9. Managing Applications in a Secure Shell Environment

11.9.1. Introduction to Secure Shell Environment

Managing applications in a secure shell environment provides specialized tools for advanced operations and general debugging. Access to applications with the shell environment is protected and restricted with Security-Enhanced Linux (SELinux) policies.



Important

Although accessing applications with the shell environment provides advanced operations, accidental damage to an application can occur. Therefore, Red Hat recommends to use shell access only when necessary.

The following table describes the available options when accessing applications in a secure shell environment:

Table 11.5. Options for Accessing Applications in Secure Shell Environment

Option	Description
-n, --namespace [NAME]	Domain where you wish to create the application.

Option	Description
--ssh [PATH]	Path to SSH executable or additional options.
--gears	Execute the command on all application gears; requires a command.
--limit [INTEGER]	Limit the number of simultaneous SSH connections that can be opened with the --gears option; default is 5.
--command [COMMAND]	Command to run in the application's secure shell environment.
-a, --app [NAME]	Name for the application to be created.
-l, --rhlogin [LOGIN]	OpenShift Enterprise user account.
-p, --password [PASSWORD]	OpenShift Enterprise account password.

[Report a bug](#)

11.9.2. Accessing an Application

When an application is accessed in a secure shell environment, the connection is made to the primary gear of the application by default. The primary gear is the gear where the Git repository and the web cartridge are located.

Access an application in a secure shell environment with the following command:

```
$ rhc ssh App_Name [Options]
```

Example 11.10. Accessing an Application in Secure Shell Environment

```
$ rhc ssh racer
Connecting to 517623ecd93cdffa000001@racer-automobile.example.com ...

*****

You are accessing a service that is for use only by authorized
users.
If you do not have authorization, discontinue use at once.
Any use of the services is subject to the applicable terms of the
agreement which can be found at:
https://www.openshift.com/legal

*****

Welcome to OpenShift shell

This shell will assist you in managing OpenShift applications.

!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
Shell access is quite powerful and it is possible for you to
accidentally damage your application. Proceed with care!
If worse comes to worst, destroy your application with 'rhc app
delete'
and recreate it
!!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!
```

Type "help" for more info.

```
[racer-automobile.example.com 517623ecd93cdffa000001]\>
```

From the shell environment, run the **help** command to see the available shell commands. General Linux commands are available for routine operations in the shell environment.

Specific SSH commands can be run by passing one or more arguments. A different SSH executable can be used, or further options can be passed to SSH with the **--ssh** option.

[Report a bug](#)

11.9.3. Accessing a Specific Gear

As mentioned earlier, a secure shell environment connection is made to the application's primary gear by default. However, a specific gear can be accessed for debugging gear problems in a scalable application. The following instructions describe how to access a gear with the gear's ID and SSH URL.

Procedure 11.2. To Access a Specific Gear:

1. Determine the gear's ID and SSH URL with the following command:

```
$ rhc app show App_Name --gears
```

Example 11.11. Viewing Application Gears

```
$ rhc app show automobile --gears
ID                               State   Cartridges                               Size  SSH
URL
-----
51774b712587c83ddb00009d started php-5.4 haproxy-1.4 small
51774b712587c83ddb00009d@hybrid-automobile.example.com
519b0fd02587c84b860002d8 started php-5.4 haproxy-1.4 small
519b0fd02587c84b860002d8@519b0fd02587c84b860002d8-
automobile.example.com
519b1018dbd93c85180001fc started php-5.4 haproxy-1.4 small
519b1018dbd93c85180001fc@519b1018dbd93c85180001fc-
automobile.example.com
519b06ebdbd93cd439000027 started postgresql-9.2      small
519b06ebdbd93cd439000027@519b06ebdbd93cd439000027-
automobile.example.com
```

In the example output the ID of the first scaling gear is **519b0fd02587c84b860002d8** and its SSH URL is **519b0fd02587c84b860002d8@519b0fd02587c84b860002d8-automobile.example.com**.

2. Open a secure shell environment to the desired gear with the gear's SSH URL:

```
$ ssh 519b0fd02587c84b860002d8@519b0fd02587c84b860002d8-
automobile.example.com
```

[Report a bug](#)

11.9.4. Accessing a Database Cartridge

The integrity of a database is verified by connecting to an application using SSH and running the shell for the database cartridge. A successful connection to the database shell indicates that the database has been created correctly.

The shell for each database also offers a selection of management commands. See the appropriate database documentation for more information on the available database shell commands.



Important

Although accessing applications with the shell environment provides advanced operations, accidental damage to your application can occur. Therefore, Red Hat recommends to use shell access only when necessary.

Procedure 11.3. To Manage a Database in a Shell Environment:

1. Access the desired application in a shell environment with the following command:

```
$ rhc ssh App_Name
```

2. From the application's shell environment prompt, run the appropriate command for the database to access the interactive database shell:

- ✎ Run the **mysql** command to access the MySQL shell.
- ✎ Run the **psql** command to access the PostgreSQL shell.
- ✎ Run the **mongo** command to access the MongoDB shell.

Example 11.12. Accessing a MySQL Shell

```
[racer-automobile.example.com 515e21acdbd93c051d000022]\> mysql
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 1
Server version: 5.1.71 Source distribution
```

```
Copyright (c) 2000, 2013, Oracle and/or its affiliates. All
rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.
```


Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql>



Note

Since OpenShift Enterprise does not allow editing MySQL server configuration, it may be necessary to specify certain settings in the client connection string. For example, the default character set is Latin-1. If you would like to use UTF-8 instead, set the character set and collation as parameters on the connection string. For example, in a Java EE application, the URL would be specified in `persistence.xml` like:

```
<connection-url>jdbc:mysql://xxx.x.xxx.xxx:xxxx/databaseName?
useUnicode=yes&characterEncoding=UTF-8</connection-url>
```

For more information, see https://bugzilla.redhat.com/show_bug.cgi?id=1023944.

See Also:

- ✦ [Section 11.9.1, “Introduction to Secure Shell Environment”](#)
- ✦ [Section 11.9.2, “Accessing an Application”](#)

[Report a bug](#)

11.10. Custom Domains and SSL Certificates

Custom domain aliases are designated so that applications can use custom DNS entries rather than the domain generated by the system. Note that a CNAME record with your DNS provider is required for custom aliases to work correctly.

Custom SSL certificates with domain aliases are available for added security, but must be enabled by a system administrator. See the *OpenShift Enterprise Administration Guide* at <https://access.redhat.com/site/documentation> for more information.

Management Console

Click on an application name in the **My Applications** tab in the Management Console to view custom domain name and SSL certificate management options for the selected application.

[Report a bug](#)

11.10.1. Managing Custom Domain Names

Adding a Custom Domain Name

Add a custom domain name to an application with the following command, specifying the application name and custom domain name:

```
$ rhc alias add App_Name Custom_Domain_Name
```

Example 11.13. Adding a Custom Domain Name

```
$ rhc alias add racer fast.cars.com
RESULT:
Alias 'fast.cars.com' has been added.
```

Viewing Custom Domain Names

View domain name aliases and SSL certificate status with the following command, specifying the application name:

```
$ rhc alias list App_Name
```

Example 11.14. Viewing Custom Domain Names

```
$ rhc alias list racer

Alias          Has Certificate? Certificate Added
-----
fast.cars.com  yes                2013-08-05
quick.cars.com no
```

Removing a Custom Domain Name

Remove a domain name alias from an application with the following command, specifying the application name and the custom domain name to be removed:

```
$ rhc alias remove App_Name Custom_Domain_Name
```

[Report a bug](#)

11.10.2. Managing Custom SSL Certificates

Adding a Custom SSL Certificate

Add a custom SSL certificate to an alias with the following command.

```
$ rhc alias update-cert App_Name Domain_Name --certificate Cert_File --
private-key Key_File
```

If the private key is encrypted, specify the passphrase with the **--passphrase** option.

Viewing Custom SSL Certificate Status

View domain name aliases and SSL Certificate status with the following command, specifying the application name:

```
$ rhc alias list App_Name
```

Example 11.15. Checking SSL Certificate Status

```
$ rhc alias list racer
```

Alias	Has Certificate?	Certificate Added
fast.cars.com	yes	2013-08-05
quick.cars.com	no	-

Removing a Custom SSL Certificate

Remove a custom SSL certificate from an alias with the following command, specifying the application name and alias name:

```
$ rhc alias delete-cert App_Name Alias
```

[Report a bug](#)

11.11. The Watchman Tool

The Watchman tool is a daemon that is used to protect your OpenShift Enterprise instance against common issues found by Red Hat. The Watchman tool solves these common issues autonomously, and includes the following built-in features:

- Watchman searches cgroup event flow through syslog to determine when a gear is destroyed. If the pattern does not match a clean gear removal, the gear will be restarted.
- Watchman monitors the application server logs for messages hinting at out of memory, then restarts the gear if needed.
- Watchman compares the user-defined status of a gear, then the actual status of the gear, and fixes any dependencies.
- Watchman searches processes to ensure they belong to the correct cgroup. It kills abandoned processes associated with a stopped gear, or restarts a gear that has zero running processes.
- Watchman monitors the usage rate of CPU cycles and restricts a gear's CPU consumption if the rate of change is too aggressive.

Watchman capabilities can be expanded with plug-ins. Consult your system administrator for configuring Watchman or see the *OpenShift Enterprise Administrators Guide* at <https://access.redhat.com/site/documentation> for more information.

[Report a bug](#)

11.12. Scheduling Cron Jobs

Cron jobs for applications are created with the OpenShift Enterprise cron scheduler. This is done by adding the cron scheduler cartridge to an application, adding the required cron jobs to the appropriate directories, and then updating the remote Git repository.

The following instructions describe how to enable cron support for an application. It is assumed the application has already been created.

Procedure 11.4. To Enable Cron Support for an Application:

1. Add the cron scheduler cartridge to an application:

```
$ rhc cartridge add cron -a App_Name
```

2. Add the cron jobs to the application's `.openshift/cron/{minutely, hourly, weekly, daily, monthly}/` directories.

Example 11.16. Sample Cron File

```
$ mkdir -p .openshift/cron/minutely
$ echo 'date >> $OPENSIFT_REPO_DIR/php/date.txt' >
  .openshift/cron/minutely/date.sh
```

The example cron job appends a new line of date information to the `$OPENSIFT_REPO_DIR/php/date.txt` file every minute.

3. Commit the changes and push them to the remote repository:

```
$ git add .openshift/cron/
$ git commit -m "configuring cron jobs"
$ git push
```

Verify that the cron job script you create works correctly.

Example 11.17. Verifying Cron Job Script Works Correctly

For the script used in the example, it can be verified with the following command:

```
$ curl http://holy-roller.example.com/date.txt
Thu Feb  2 01:02:01 EST 2012
Thu Feb  2 01:03:01 EST 2012
Thu Feb  2 01:04:01 EST 2012
```

The scripts placed in the `/cron` subdirectories are executed at the respective frequencies. For example, scripts in each subdirectory are executed in alphabetical order; scripts in the `/cron/hourly` directory are executed on the first minute of every hour.

Disabling Cron Job Scripts

Disable all cron job scripts with the following command:

```
$ rhc cartridge stop cron -a App_Name
```

Enabling Cron Job Scripts

Enable all cron job scripts with the following command:

```
$ rhc cartridge start cron -a App_Name
```



Note

The cron commands affect all cron jobs. You cannot disable or enable individual cron jobs.

[Report a bug](#)

11.13. Binding Applications to Ports

All ports less than 1024 are reserved for OpenShift Enterprise operations, and developers cannot bind to these ports. However, ports greater than 1024 are available for binding, and the following table shows the commonly used ports.



Important

While the following ports are suggestions for available applications or gears, ports 2303 - 2308 are reserved for OpenShift SNI Implementation, and port 10050 is reserved for the OpenShift Enterprise Zabbix agent. You cannot bind to these ports.

Table 11.6. Common Ports and Their Usage

Application Name	Port Number	Description
Kerberos	4444	Kerberos is used for user authentication identifying with a node.
Git	9418	Git is used for version control.
MySQLd	3306, 63132-63164	My Structured Query Language (MySQL) acts as a server providing access to databases.
Mongod	27017	MongoDB acts as a server providing access to databases.
PostgreSQL	5432	PostgreSQL acts as a server providing access to databases.
MS SQL	1433-1434	MS SQL acts as a server providing access to databases.
Oracle	1521, 2483, 2484	Oracle acts as a server providing access to databases.
HTTP/HTTPS	8008, 8009, 8443	Hypertext Transfer Protocol Secure (HTTPS) is used for secure communication over a server.
HTTP cache	8080, 8118, 8123, 10001-10010	HTTP cache is used for the temporary storage of documents.
memcache	11211	memcache is a memory caching system.

Application Name	Port Number	Description
jacORB	3528, 3529	JacORB is a Java request broker.
JBoss Debug	8787	A debug program for JBoss applications.
JBoss Management	4712, 4447, 7600, 9123, 9990, 9999, 18001	A management program for JBoss applications.
AMQP	5671-5672	Advanced Message Queuing Protocol (AMQP) is used to transfer messages between applications.
PulseAudio	4713	PulseAudio is a server that manages the use of audio devices.
Flash	1935	Flash is a multimedia platform.
Munin	4949	Munin is a network monitoring application.
Virt Migration	49152-49216	Virt migration is the copying of one machine's data and moving it to another.
OCSP	9080	Online Certificate Service Protocol (OCSP) is a protocol used for obtaining the status of a certificate.

Ports 15000 - 35530 are available for binding internal IP, but these ports are not externally addressable. You can also bind to \$ **OPENSIFT_Cart_Name_PORT** (8080) for HTTP connectivity, which reroutes externally through port 80.

[Report a bug](#)

11.13.1. Configuring WebSocket Ports

Because the main routing layer is currently based on Apache, WebSocket can be used by connecting to specific ports on an application. WebSocket connections are supported with browser based applications on OpenShift Enterprise, allowing bidirectional communication without requiring multiple, open HTTP connections. The TCP based protocol uses HTTP as an initiation handshake which is handled as an upgrade request. If successful, the connection remains open and switches to the WebSocket protocol.

For plain WebSocket connections (**ws** : //), requests are directed to port 8000, while WebSocket Secure connections (**wss** : //) use port 8443, as shown in the following example:

* http://example.example.com:8000

* https://example.example.com:8443

[Report a bug](#)

11.13.2. Configuring Email Ports

OpenShift Enterprise provides support for externally hosted email services, such as POP, IMAP, and SMTP. An application can be connected to your own email server, or to one of the popular public email services, such as Gmail or YahooMail. With support for popular blogging or wiki software, such as Drupal, Joomla, MediaWiki, or WordPress, email settings of the software can be modified to

point to the appropriate email service.

The following ports are the suggested options for email support:

- SMTP/submission: 25, 465, 587
- IMAP: 143, 220, 993
- POP: 109, 110, 995

Communication occurs at a limited rate. Port 587 (submission) is restricted to a maximum rate of 256 Kbps. Ports 25 (SMTP) and 465 (SMTPS) are restricted to a maximum rate of 24 Kbps. Both consume an extremely small share of the available bandwidth if there is congestion.



Important

Note that access to email servers from cloud providers may be blocked by *Realtime Blackhole Lists (RBLs)*, affecting connections to some email servers. If you are unable to connect to one of these services, ensure the email provider allows authenticated connections from your OpenShift Enterprise host.

[Report a bug](#)

11.14. Port Forwarding

11.14.1. Introduction to Port Forwarding

Port forwarding permits connections to remote services from a local workstation, without having to configure complicated firewall rules or SSH tunnels. The command used to forward ports to a local machine includes a wrapper that configures SSH port forwarding. After the ports are forwarded, the list of remote services and associated IP addresses that are being forwarded becomes available.

[Report a bug](#)

11.14.2. Application Port Forwarding

Configure port forwarding for an application with the following command, ensuring the application is running before doing so:

```
$ rhc port-forward App_Name
```

Example 11.18. Configuring Port Forwarding for an Application

```
$ rhc port-forward myapp
Checking available ports ... done
Forwarding ports ...
Address already in use - bind(2) while forwarding port 8080. Trying
local port 8081
Address already in use - bind(2) while forwarding port 8080. Trying
local port 8081
Address already in use - bind(2) while forwarding port 8081. Trying
local port 8082
```

To connect to a service running on OpenShift, use the Local address

Service	Local		OpenShift
-----	-----	-----	-----
haproxy	127.0.0.1:8080	=>	127.9.159.130:8080
haproxy	127.0.0.1:8081	=>	127.9.159.131:8080
httpd	127.0.0.1:8082	=>	127.9.159.129:8080
mysql	127.0.0.1:50226	=>	52347a1d2587c86695111697-
mydomain.rhcloud.com	:50226		

Press CTRL-C to terminate port forwarding

With port forwarding configured, access the remote application with a browser using the local ports.

The current implementation of the **rhc port-forward** command forwards all open ports on a running application to your local workstation. If an application contains multiple cartridges, the command output shows which remote services are being bound to local ports.

Specific ports are forwarded with the following command. Specify the local port and remote port as required, as well as the gear ID, and application and domain name of the remote port:

```
$ ssh -L local_port:host:remote_port gear_ID@app-domain.example.com
```

Example 11.19. Forwarding Specific Ports

```
$ ssh -L 8080:localhost:8080 70277280b8534c8a9fc76d2734393dfa@racer-
auto.example.com
```

This example allocates a socket to listen to the local port host *8080*. When a connection to this port is made, a secure channel forwards the connection to the remote host port *8080*.

[Report a bug](#)

11.14.3. Gear Port Forwarding

Diagnosing problems with scalable applications is easier with port forwarding specific gears.

After you have determined the gear's ID, port forward that gear with the following command:

```
$ rhc port-forward App_Name -g gear_ID
```

Example 11.20. Port Forwarding a Specific Gear

```
$ rhc port-forward racer -g 522d59745973caccab0000c1
Checking available ports ... done
Forwarding ports ...
```

To connect to a service running on OpenShift, use the Local address


```

Service Local                OpenShift
-----
httpd    127.0.0.1:8080  =>  127.12.166.129:8080

Press CTRL-C to terminate port forwarding

```

See Also:

- » [Section 11.9.3, “Accessing a Specific Gear”](#)

[Report a bug](#)

11.14.4. Port Forwarding on Mac OS X

Currently, out of the box, Mac OS X only provides the following interfaces for loopback addresses:

- » localhost
- » 127.0.0.1

Therefore, port forwarding on Mac OS X may not work correctly. The following example shows error messages that can occur when attempting to configure port forwarding using the IP address of an application.

Example 11.21. Error Messages When Port Forwarding on Mac OS X

```

$ rhc port-forward myapp
Checking available ports...
Error trying to forward ports. You can try to forward manually by
running:
ssh -N 70277280b8534c8a9fc76d2734393dfa@myapp-domain.example.com

```

The current workaround to enable port forwarding on Mac OS X is to manually configure an alias for each IP address used by the application:

```
$ sudo ifconfig lo0 alias application_IP_address
```

Example 11.22. Manually Configured IP Address

```
$ sudo ifconfig lo0 alias 127.10.51.129
```

**Note**

Root or administrative privileges are required to run the **ifconfig** command on Mac OS X.

If the application uses multiple IP addresses, you must configure an alias for each IP address. For example, suppose a PHP application has both MySQL and phpMyAdmin cartridges added, and it uses the IP addresses `127.11.25.1` and `127.11.25.2`. For port forwarding to work correctly, configure an alias for each IP address as shown in the following example:

Example 11.23. Configuring Aliases for Multiple IP Addresses

```
$ sudo ifconfig lo0 alias 127.11.25.1
$ sudo ifconfig lo0 alias 127.11.25.2
```

Finally, enable port forwarding with the **rhc port-forward** command.



Important

The IP address alias configured for an application is not persistent through system reboots. If the system is rebooted, you must repeat these steps to correctly enable port forwarding on Mac OS X.

[Report a bug](#)

11.15. Deleting an Application



Warning

Deleting an application deletes all remote data associated with that application, which cannot be recovered.

Run the following command to delete an application and all associated remote data, answering **yes** when prompted:

```
$ rhc app delete App_Name
```

Note that this process only deletes remote application data. Data stored on your local machine must be manually deleted.



Warning

The following process deletes the selected directory and all its files, which cannot be recovered. Ensure the correct directory is specified for deletion.

Deleting Local Data

Delete the local application data with the following command:

```
$ rm -rf ~/path/to/app_directory/
```

[Report a bug](#)

Chapter 12. Cartridge Management

12.1. Viewing Available Cartridges

View a list of all available cartridges with the following command:

```
$ rhc cartridge list
```

This command displays all available web framework and add-on cartridges.

[Report a bug](#)

12.2. Adding a Cartridge to an Application

When adding a cartridge to an application with the client tools, there are several options available that are used to specify the required information.

Table 12.1. Options When Adding Cartridges

Option	Description
-a, --app [NAME]	Name of an application.
-n, --namespace [NAME]	Name of a domain.
-e, --env [VARIABLE=VALUE]	Environment variable(s) to be set on this cartridge. It can also be a path to a file that contains the environment variables.
-g, --gear-size [SIZE]	Gear size determines how much memory and CPU a cartridge consumes.
-c, --cartridge [CART_TYPE]	Type of cartridge to add to an application.

Add a cartridge to an application with the following command, specifying the desired cartridge and the name of the application:

```
$ rhc cartridge add Cart_Name -a App_Name
```

Specifying Cartridge Gear Size

When adding a cartridge to an application, specify the cartridge gear size with the **-g**, or **--gear-size** option along with the gear size. Note that this option is not available with non-scalable applications, because the web cartridge and any add-on cartridges are placed on the same gear.

```
$ rhc cartridge add Cart_Name -a App_Name -g gear_size
```

[Report a bug](#)

12.3. Viewing Cartridges for an Application

View all cartridges associated with an application with the following command:

```
$ rhc app show App_Name
```

Example 12.1. List of Cartridges for an Application

```

$ rhc app show mynewapp
mynewapp @ http://mynewapp-mydomain.rhcloud.com/
(uuid: 5213190e2587c8817a000121)
-----
Domain:      mydomain
Created:     Aug 20, 2013  3:21 AM
Gears:       2 (defaults to medium)
Git URL:     ssh://5213190e2587c8817a000121@mynewapp-
mydomain.rhcloud.com/~/.git/mynewapp.git/
SSH:         5213190e2587c8817a000121@mynewapp-mydomain.rhcloud.com
Deployment:  auto (on git push)

php-5.4 (PHP 5.4)
-----
    Scaling: x1 (minimum: 1, maximum: available) on medium gears

haproxy-1.4 (Web Load Balancer)
-----
    Gears: Located with php-5.4

mysql-5.5 (MySQL 5.5)
-----
    Gears:          1 medium
    Connection URL:
mysql://$OPENSIFT_MYSQL_DB_HOST:$OPENSIFT_MYSQL_DB_PORT/
    Database Name:  mynewapp
    Password:       password
    Username:       username

```

[Report a bug](#)

12.4. Cartridge Management Commands

Manage cartridges with the client tools using the **rhc cartridge** command, with the following syntax:

```
$ rhc cartridge Action Cart_Type -a App_Name
```

The following table describes the available cartridge management actions:

Table 12.2. Cartridge Management Actions

Action	Details
list	List supported cartridges.
add	Add a cartridge.
remove	Remove a cartridge.
stop	Stop a cartridge.
start	Start a cartridge.
restart	Restart a cartridge.

Action	Details
status	Return the current status of a cartridge.
reload	Reload the configuration of a cartridge.
show	Show information about a cartridge.
storage	View and manipulate storage on a cartridge.
scale	Set the scaling range of a cartridge.

The following table describes the available options with cartridge management commands:

Table 12.3. Cartridge Management Command Options

Option	Description
-n, --namespace [NAME]	Name of a domain.
-a, --app [NAME]	Name of an application.
-c, --cartridge [CART_TYPE]	Name of cartridge.
-l, --rhlogin [LOGIN]	OpenShift Enterprise user account.
-p, --password [PASSWORD]	OpenShift Enterprise account password.

Example 12.2. Stopping a Cartridge

```
$ rhc cartridge stop php -a mynewapp
Using php-5.4 (PHP 5.4) for 'php'
Stopping php-5.4 ... done
```

[Report a bug](#)

Chapter 13. Build and Deployment

13.1. Introduction to Deployment

The application deployment process involves making any required changes to the application code, committing those changes to the local repository, and then updating the remote repository. Application files are stored in the local Git repository that was cloned when the application was created.

The deployment process uses the application's storage space as part of the build and test process. This means that the running application must be shut down so that its memory can be utilized. Therefore, the application is not available for the duration of the build.

The following table outlines and describes the associated tasks of the deployment process.

Table 13.1. The Deployment Process

Deployment Step	Description
Pre-build	This occurs when the git push command is run, but before the push is fully committed.
Build	This builds an application, downloads required dependencies, executes the .openshift/action_hooks/build script and prepares everything for deployment.
Deploy	This performs any required tasks necessary to prepare the application for starting, including running the .openshift/action_hooks/deploy script. This step occurs immediately before the application is issued a start command.
Post-deploy	This step enables interaction with the running application, including running the .openshift/action_hooks/post_deploy script. This step occurs immediately after the application is restarted.

[Report a bug](#)

13.2. Preparing an Application for Deployment

When preparing an application for deployment, all files must be committed to the appropriate directories in the local Git repository so that local application files are synchronized with the remote repository. The local Git repository is then pushed to the remote repository. For example, the local files for a PHP application are stored in the **App_Name/php/** directory. The following instructions describe how to prepare an application for deployment.

Procedure 13.1. To Prepare an Application for Deployment:

1. Add each new file and directory to the Git index:

```
$ git add path/to/newfile
```

2. Commit an application to the local repository:

```
$ git commit -m "commit message"
```

[Report a bug](#)

13.3. Deployment Mechanisms

13.3.1. Automatic Deployment

When an application is created as described in [Section 11.2, “Creating an Application”](#), it is configured for automatic deployment by default. If the application code is changed locally, run the following commands to commit and deploy the application automatically:

```
$ git commit
$ git push
```

The **git push** command sends the application data to the remote repository and automatically deploys the application. The application automatically stops, builds, and restarts when the code changes are pushed to the remote server.

[Report a bug](#)

13.3.1.1. Configuring Automatic Deployment

Automatic deployment is configured by default when a new application is created. However, if the deployment mechanism was changed and you wish to revert back to automatic deployment, do so with the following command:

```
$ rhc configure-app -a App_Name --auto-deploy
```

[Report a bug](#)

13.3.2. Manual Deployment

In contrast to automatic deployment, manual deployment of applications provides greater control of the application deployment process. Manual deployments of an application can be from a Git reference, such as commit ID, tag, or branch, or from a binary artifact. Because automatic deployment is configured by default when a new application is created, it must be disabled to configure manual deployment for that application.

[Report a bug](#)

13.3.2.1. Configuring Manual Deployment

Configure manual deployment for an application by disabling automatic deployment with the following command, specifying the application name:

```
$ rhc configure-app -a App_Name --no-auto-deploy
```

This command enables manual deployment of an application so that when the **git push** command is run, the application data is only pushed to the remote repository; the application is not deployed.

See Also:

- » [Section 13.3.1.1, “Configuring Automatic Deployment”](#)

[Report a bug](#)

13.3.2.2. Preserving Deployments

Preserving a number of deployments permits rollbacks to previous deployments of an application. Configure an application to preserve deployments with the following command:

```
$ rhc configure-app -a App_Name --keep-deployments No_of_Deps
```

where *No_of_Deps* is the number of deployments to keep in the application's history. Older deployments are deleted when this number is exceeded.

[Report a bug](#)

13.3.2.3. Deploying from a Git Branch

When manual application deployment is configured, deploy an application from any Git branch with the following command, specifying the Git branch to deploy from:

```
$ rhc configure-app -a App_Name deployment-branch Git_Branch
```

The Git references supported with this command are SHA, branch, and tag.

[Report a bug](#)

13.3.2.4. Deploying from a Snapshot

The following instructions describe how to deploy an application from a snapshot. Note that an application can be deployed from a binary artifact.

Procedure 13.2. To Deploy From a Snapshot:

1. Save an application snapshot to build a deployable .tar.gz artifact:

```
$ rhc save-snapshot App_Name --deployment
```

2. Configure the application for binary artifact deployments:

```
$ rhc configure-app App_Name --deployment-type binary
```

Note that this command changes the application's deployment process and disables the **git push** command.

3. Deploy the application using the binary artifact that was created:

```
$ rhc deploy ./app.tar.gz -a App_Name
```

Alternatively, use the following command to deploy from a URL:

```
$ rhc deploy http://foo.com/path/to/file.tar.gz -a App_Name
```

See Also:

✱ [Section 15.1, “Introduction to Snapshots”](#)

[Report a bug](#)

13.3.2.5. Viewing Previous Deployments

Previous deployments are viewed with the following command:

```
$ rhc deployments App_Name
```

This command displays the individual ID of each deployment, which is used to activate that deployment.

[Report a bug](#)

13.3.2.6. Activating a Previous Deployment

When manual deployment is configured, each application deployment contains a deployment ID, as described in [Section 13.3.2.5, “Viewing Previous Deployments”](#). Activate a previous deployment for an application with the following command, specifying the deployment ID to activate:

```
$ rhc activate-deployment -a App_Name Dep_ID
```

[Report a bug](#)

13.4. Action Hooks

13.4.1. Introduction to Action Hooks

Various entry points, referred to as action hooks, are available to modify certain processes during an application's life cycle, and are specifically used to interact with cartridges. These action hooks are located in the ***App_Name/.openshift/action_hooks*** directory.

During a process that supports an action hook, the application action hook directory is checked for an executable file matching the specified name. If it is found, the file is executed before control is returned to the normal process. There are no specific implementation requirements on action hooks other than that they be executable files. The action hook scripts are directly executed by OpenShift Enterprise.

[Report a bug](#)

13.4.2. Cartridge Action Hooks

Cartridge action hooks are used by creating a file in the ***App_Name/.openshift/action_hooks*** directory with the same name as the desired event.

Use the following list for a reference to all possible action hooks associated with a cartridge control action.

Table 13.2. Cartridge Action Hooks

Action	Description	Event-specific examples
Start	Start the software the cartridge controls.	pre_start_ <i>Cart_Name</i> , post_start_ <i>Cart_Name</i>

Action	Description	Event-specific examples
Stop	Stop the software the cartridge controls.	pre_stop_ <i>Cart_Name</i> , post_stop_ <i>Cart_Name</i>
Reload	The cartridge and the package software will re-read the configuration information.	pre_reload_ <i>Cart_Name</i> , post_reload_ <i>Cart_Name</i>
Restart	Current cartridge process is stopped and started again.	pre_restart_ <i>Cart_Name</i> , post_restart_ <i>Cart_Name</i>
Tidy	All unused resources are released.	pre_tidy_ <i>Cart_Name</i> , post_tidy_ <i>Cart_Name</i>

Cart_Name is a replaceable term used to represent the cartridge short-name. For example, for a JBossAS cartridge to be implemented during the pre-start process, create the file ***App_Name*/.openshift/action_hooks/pre_start_jbossas**, edit it and add the desired information.

[Report a bug](#)

13.4.3. Build and Deployment Action Hooks

The list of action hooks for build and deployment are:

- pre-build
- build
- deploy
- post-deploy

Create a new file in the ***App_Name*/.openshift/action_hooks** directory to use the build and deployment action hooks. For example, to use an action hook during the application build phase, create the file ***App_Name*/.openshift/action_hooks/build**, edit it and add the following to the file's contents:

Example 13.1. Adding an Action Hook to the Build Process

```
echo Downloading my.zip...
curl -o $OPENSHIFT_DATA_DIR/my.zip
http://myserver/my.zip
```

The file is downloaded during the **git push** process.

[Report a bug](#)

13.4.4. Scaling Action Hooks

Automatic scaling is controlled by the **haproxy_ctld** daemon. The **haproxy_ctld.rb** script, which changes the thresholds and algorithms used to control scale up and down behavior, can be customized for use as an action hook in scalable applications.

Procedure 13.3. To Customize Automatic Scaling for an Application:

1. Use SSH to connect to a scalable application and consult the generic **~/haproxy/usr/bin/haproxy_ctld.rb** script for detailed usage information.

2. Copy the file to the Git repository of the application in the **`App_Name/.openshift/action_hooks/`** directory.
3. Ensure the file is executable:

```
# chmod +x App_Name/.openshift/action_hooks/haproxy_ctld.rb
```

4. Edit the file to the desired specifications.
5. Deploy the changes. To ensure that the changes take effect immediately, the **HAProxy** cartridge restarts automatically during deployment if the **`haproxy_ctld.rb`** action hook is detected.

Example 13.2. Scaling Up Based on Memory Usage

To enable auto-scaling based on memory usage, edit the `~/haproxy/usr/bin/haproxy_ctld.rb` script by specifying the following parameters.

```
...
# Scale up when any gear is using 400M or more memory.
mem_scale_up = 419430400

# Scale down when any gear is using 300M or less memory
mem_scale_down = 314572800

# min_gears - Once this number of gears is met, don't try to scale down
any lower
min_gears = 2

gear_list['web'].each do |uuid, array|
  mem_usage = `ssh -i ~/.openshift_ssh/id_rsa #{uuid}@#{array['dns']}
'oo-cgroup-read memory.memsw.usage_in_bytes'`.to_i
  if mem_usage >= mem_scale_up
    @log.error("memory usage (#{mem_usage}) on #{array['dns']} is above
threshold(#{mem_scale_up}), adding new gear")
    self.add_gear
  end
end
```

[Report a bug](#)

13.4.5. Metrics Action Hooks

With the release of OpenShift Enterprise 2.1, you can now use metrics action hook to collect metrics at a gear level. Create a new file named **`metrics`** in the **`App_Name/.openshift/action_hooks`** directory to use the metrics action hook.

You can provide an application with a metrics action hook if the Watchman metrics plug-in is enabled by the system administrator. The application metrics action hook is invoked at an interval configured for the Watchman plug-in. Ensure the action hook is an executable file and that the action hook writes to standard out (STDOUT).

Message Format

A metrics message must include the following fields and be written to standard out (STDOUT):

```
type=metric <metric name>=<metric value>
```

Example 13.3. Metrics Message Example

```
type=metric thread.count=5
```

[Report a bug](#)

13.5. Environment Variables

13.5.1. Introduction to Environment Variables

Environment variables are placeholders for values that are provided to a software program at runtime. They are particularly useful when the values are likely to be different from one host system to the next, or from one run to the next. Including these placeholders in applications makes the application code more portable and flexible. This flexibility is critical for writing applications that are easily deployed and scaled on OpenShift Enterprise.

A number of standardized environment variables are available for applications hosted on OpenShift Enterprise. These variables serve as placeholders for application names, commonly accessed directory names, user names, passwords, host names, IP addresses, and more. The specific environment variables that are available to a given application is determined by the cartridges that have been added to that application. For example, an application with PHP and MySQL has access to environment variables that expose the PHP path information, including the host, port, user name, and password necessary for connecting to the MySQL database.

There are two ways to view the environment variables for an application:

1. Add an **export** statement to the **App_Name/.openshift/action_hooks/build** file, then run **git push**. The variables are listed in the Git output and start with **remote: declare -x**.
2. Access the application with SSH and run the **env** command at the shell prompt.

[Report a bug](#)

13.5.2. Informational Environment Variables

Informational environment variables provide information about an application. These variables are always available to the application, regardless of which cartridges the application is using.

Table 13.3. Informational Environment Variables

Environment Variable Name	Purpose
OPENSHIFT_APP_DNS	The fully-qualified domain namespace of the application.
OPENSHIFT_APP_NAME	The name of the application.
OPENSHIFT_APP_UUID	The UUID of the application (32 hexadecimal characters).
OPENSHIFT_Cart_Name_IP	The IP address the application listens on.
OPENSHIFT_Cart_Name_PORT	The port the application receives requests from.

Environment Variable Name	Purpose
OPENSIFT_SECRET_TOKEN	A 128-character string unique to an application that can be used for authentication, and can be overridden with the rhc env set command.

[Report a bug](#)

13.5.3. Directory Environment Variables

Directory environment variables return the directories where an application resides. These variables are always available to the application, regardless of which cartridges the application is using.

Table 13.4. Directory Environment Variables

Environment Variable Name	Purpose
OPENSIFT_HOMEDIR	The home directory of the application.
OPENSIFT_DATA_DIR	A persistent data directory.
OPENSIFT_REPO_DIR	Repository containing the currently deployed version of the application.
OPENSIFT_TMP_DIR	A temporary directory you can use; SELinux protects data in this directory from other users.
OPENSIFT_LOG_DIR	Where all cartridge logs are stored.



Note

Many of these directories are emptied and rebuilt whenever new code is pushed to an application. The only persistent directory is **OPENSIFT_DATA_DIR**. Therefore, Red Hat recommends that you store persistent files in the **OPENSIFT_DATA_DIR** directory.

[Report a bug](#)

13.5.4. Logging Environment Variables

With the release of OpenShift Enterprise 2.1, logging environment variables are available to configure the behavior of logs generated by an application. When logs are written to the **OPENSIFT_LOG_DIR** directory of an application, log files are rolled if their file size exceeds a configurable threshold. A configurable number of rolled files are retained before the oldest file is removed prior to the next roll.

Table 13.5. Logging Environment Variables

Environment Variable Name	Purpose
LOGSHIFTER_OUTPUT_TYPE	If permitted by your system administrator, overrides the default logging behavior. Set file to have logs written to the OPENSIFT_LOG_DIR directory of the application, or set syslog to enable Syslog logging, if available. Starting in OpenShift Enterprise 2.1.7, set multi to enable both file and syslog at the same time.

Environment Variable Name	Purpose
LOGSHIFTER_<i>Cart_Name</i>_MAX_FILESIZE	A case-insensitive string representing the maximum log file size that triggers a roll event. The default value is 10M . If a zero size is specified regardless of the unit, log rolling is effectively disabled.
LOGSHIFTER_<i>Cart_Name</i>_MAX_FILES	An integer representing the maximum number of log files to retain. The default is 10 .

Cart_Name is a replaceable term used to represent the cartridge short-name. The **LOGSHIFTER_*Cart_Name*_MAX_FILESIZE** variable accepts strings in kilobytes, megabytes, gigabytes, and terabytes. For example, for an application with a PHP cartridge, any of the following values would be valid:

- LOGSHIFTER_PHP_MAX_FILESIZE=500K
- LOGSHIFTER_PHP_MAX_FILESIZE=10M
- LOGSHIFTER_PHP_MAX_FILESIZE=2G
- LOGSHIFTER_PHP_MAX_FILESIZE=1T

[Report a bug](#)

13.5.5. Database Environment Variables

Database environment variables pertain to a database, if one exists, and are used to connect an application to a database. The exact variable names depend on the type of database; the value of `<database>` is MySQL or POSTGRES as appropriate. Note that these connections are only available to an application internally; you cannot connect from an external source.

OpenShift Enterprise does not currently support user changes to environment variables. This includes changing the default MySQL admin password (even outside of phpMyAdmin). If the password is changed, ensure the change takes effect correctly. Note that this restriction only applies to the default administrative user. You can add more users as required, and specify a custom password for these users.

Table 13.6. Database Environment Variables

Environment Variable Name	Purpose
OPENSIFT_<i>Cart_Name</i>_DB_HOST	The host name or IP address used to connect to the database.
OPENSIFT_<i>Cart_Name</i>_DB_PORT	The port the database server is listening on.
OPENSIFT_<i>Cart_Name</i>_DB_USERNAME	The database administrative user name.
OPENSIFT_<i>Cart_Name</i>_DB_PASSWORD	The database administrative user's password.
OPENSIFT_<i>Cart_Name</i>_DB_SOCKET	An AF socket for connecting to the database (for non-scaled apps only).
OPENSIFT_<i>Cart_Name</i>_DB_URL	Database connection URL.

[Report a bug](#)

13.5.6. Library Environment Variables

Library environment variables are used for customizing the location of bundled files.

Table 13.7. Library Environment Variables

Environment Variable Name	Purpose
OPENSIFT_Cart_Name_LD_LIBRARY_PATH_ELEMENT	Configures the location of each cartridge's library file.



Note

The global directory for a cartridge is set with **LD_LIBRARY_PATH**. However, cartridges may be competing for a place in the set directory. Configure the destination of each cartridge's files with **OPENSIFT_Cart_Name_LD_LIBRARY_PATH_ELEMENT** to merge each cartridge's library into the global directory. Note that the order that the files are entered into the global directory is add-on cartridges first, then web framework cartridges. Red Hat recommends not changing the location of the **LD_LIBRARY_PATH** environment variable.

[Report a bug](#)

13.5.7. Jenkins Environment Variables

Jenkins environment variables are available if an application has Jenkins enabled.

Table 13.8. Jenkins Environment Variables

Environment Variable Name	Purpose
JENKINS_USERNAME	System builder account on the Jenkins server.
JENKINS_PASSWORD	Password for the system builder account on the Jenkins server.
JENKINS_URL	DNS name for the associated Jenkins server where builds occur.

See the [Client Tools Installation Guide](#) for more information on environment variables.

[Report a bug](#)

13.5.8. Gear Environment Variables

Gear environment variables are available for scalable applications.

Table 13.9. Gear Environment Variables

Environment Variable Name	Purpose
OPENSIFT_GEAR_DNS	The fully-qualified domain name of the gear.
OPENSIFT_GEAR_NAME	The name of the gear.
OPENSIFT_GEAR_UUID	The UUID of the gear.

[Report a bug](#)

13.5.9. JBoss Environment Variables

JBoss environment variables are available for supported JBoss applications.

Table 13.10. JBoss Environment Variables

Environment Variable Name	Purpose
JAVA_OPTS	Controlled by the cartridge and used to specify additional arguments to the JVM where JBoss application will run.
JAVA_OPTS_EXT	Appended to the JAVA_OPTS environment variable before the JVM is invoked, and used to provide additional options to the JVM without rewriting the JAVA_OPTS environment variable. This allows developers to better support their application users.
DISABLE_OPENSIFT_MANAGED_SERVER_CONFIG	Set to true and the standalone.xml file from the repository is ignored, as is the copy that was retained.

JBoss environment variables are stored in the **/App_Name/.openshift/config/standalone.xml** file that is part of jbossas-7. The following example code shows the environment variables for a MySQL datasource connection URL in the form `jdbc:mysql://SERVER_NAME:PORT/DATABASE_NAME`:

```
<connection-
url>jdbc:mysql://${env.OPENSIFT_MYSQL_DB_HOST}:${env.OPENSIFT_MYSQL_DB
_PORT}/${env.OPENSIFT_APP_NAME}</connection-url>
```

The environment variables can be saved on the server so that sensitive information is not repeatedly passed to the command line. The following instructions describe how to set environment variables on the server.

Procedure 13.4. To Set Environment Variables on the Server:

1. Open the **App_Name/.openshift/config/standalone.xml** file.
2. Specify the required values for any of your environment variables, then save and close the file.
3. Commit and push the changes to the server:

```
$ git commit -a -m "COMMIT MESSAGE"
$ git push
```



Important

Sensitive information stored in environment variables is visible if you use the **rhc snapshot** commands.

**Note**

If you use the **jboss-cli.sh** tool or the JBoss Management Console to edit the **standalone.xml** file, it only edits the local gear's **standalone.xml** file and not the repository one provided by your OpenShift cartridge template:

App_Name/.openshift/config/standalone.xml. Manual changes will be lost upon an application restart, and the last version of the repository **standalone.xml** file will be used even if you remove the repository **standalone.xml** file.

To make your **jboss-cli.sh** tool or JBoss Management Console changes persistent or to stop the application from using the repository **standalone.xml** file, set the **DISABLE_OPENSIFT_MANAGED_SERVER_CONFIG** environment variable to **true** by running:

```
# rhc env set DISABLE_OPENSIFT_MANAGED_SERVER_CONFIG=true -a
App_Name
```

[Report a bug](#)

13.5.10. Ruby Environment Variables

Ruby environment variables are available for supported Ruby applications.

Table 13.11. Ruby Environment Variables

Environment Variable Name	Purpose
OPENSIFT_RUBY_LOG_DIR	Where cartridge-specific logs are stored.
BUNDLE_WITHOUT	Prevents Bundler from installing certain groups specified in the Gemfile.

[Report a bug](#)

13.5.11. Python Environment Variables

Python environment variables are available for supported Python applications.

Table 13.12. Python Environment Variables

Environment Variable name	Purpose
OPENSIFT_PYTHON_WSGI_APPLICATION	Sets a custom path for the WSGI entry point.
OPENSIFT_PYTHON_REQUIREMENTS_PATH	Sets a custom path for the pip requirements file. When git push is run, any dependencies listed in the requirements.txt file will be installed by the Python cartridge.

[Report a bug](#)

13.5.12. Custom Environment Variables

Custom environment variables are user defined to use with applications.

Setting Custom Environment Variables

Set one or more environment variables for an application with the following command:

```
$ rhc env set Variable=Value Variable2=Value2 -a App_Name
```

Add additional *Variable=Value* arguments separated by spaces to set multiple variables.

Viewing Custom Environment Variables

View the custom environment variables set for an application with the following command:

```
$ rhc env list -a App_Name
```

Viewing the Value of a Custom Environment Variable

Display the value of one or more custom environment variables with the following command:

```
$ rhc env show Variable Variable2 -a App_Name
```

Removing Custom Environment Variables

Remove a custom environment variable with the following command:

```
$ rhc env unset Variable -a App_Name
```

[Report a bug](#)

13.6. Hot Deployment

13.6.1. Introduction to Hot Deployment

When the **git push** command is run to upload code modifications, OpenShift Enterprise stops, builds, deploys, and restarts an application. This entire process takes time to complete and is unnecessary for many types of code changes. With hot deployment the changes to application code are applied without restarting the application cartridge, resulting in increased deployment speed and minimized application downtime.

OpenShift Enterprise provides support for hot deployment through a **hot_deploy** marker file. If the marker is present, supported application cartridges automatically hot deploy when the **git push** command is executed.

Table 13.13. Application Types That Can or Cannot Be Hot Deployed

Type of Application	Hot Deploy
JBoss Application Server	Yes
JBoss Enterprise Application Platform	Yes
Tomcat 6 (JBoss Enterprise Web Server 1.0)	Yes
Tomcat 7 (JBoss Enterprise Web Server 2.0)	Yes
PHP	Yes
Perl	Yes
Ruby	Yes
Python	Yes

Type of Application	Hot Deploy
Node.js	Yes
Zend Server	Yes
Jenkins	No
HAProxy	No
DIY	No

[Report a bug](#)

13.6.2. Hot Deployment Build Details

JBoss AS, JBoss EAP, Tomcat 6, and Tomcat 7

When JBoss AS, JBoss EAP, Tomcat 6, and Tomcat 7 applications are hot deployed, the Maven build is executed (either with Jenkins or without), but the server does not restart. Following the build, the JBoss HDScanner notices any modifications and redeploys them. If previously deployed artifacts are removed as part of the update, they are undeployed automatically.

PHP, Perl, Python, and Node.js

When PHP, Perl, Python, and Node.js applications are hot deployed, the application code is built (dependencies are processed and user build action_hooks are run) and deployed to the application server. The server does not restart. This is true regardless of whether an application has Jenkins enabled or not. For applications that have Jenkins enabled, the build is performed on a Jenkins slave instance and then synced to the gear(s) where the application server is running.

Ruby

When a Ruby application is hot deployed, the Passenger **restart.txt** file is touched, and the application server serves the new code without requiring a full server restart. See the [Passenger Documentation](#) for more information.

[Report a bug](#)

13.6.3. Enabling and Disabling Hot Deployment

Follow the instructions applicable to your operating system to enable or disable hot deployment.

Windows

Enable hot deployment by creating the **hot_deploy** marker file in the application's root directory with the following command:

```
C:\app_directory> copy NUL > .openshift\markers\hot_deploy
```

Disable hot deployment by deleting the **hot_deploy** marker file from the application's root directory:

```
C:\app_directory> del .openshift\markers\hot_deploy
```

Mac OS X and Linux

Enable hot deployment by creating the **hot_deploy** marker file in the application's root directory:

```
[user@user app_directory]$ touch .openshift/markers/hot_deploy
```

Disable hot deployment by deleting the **hot_deploy** marker file from the application's root directory:

```
[user@user app_directory]$ rm .openshift/markers/hot_deploy
```

[Report a bug](#)

13.7. Jenkins Continuous Integration

13.7.1. Introduction to Jenkins

The Jenkins cartridge integrates with OpenShift Enterprise applications to provide continuous integration by monitoring execution of repeated jobs. Visit <http://jenkins-ci.org/> for more information about Jenkins.

The Jenkins client cartridge must be added to a new or existing application for it to build with Jenkins. After the Jenkins client cartridge is added to an application, the **git push** command initiates a build process inside Jenkins. For custom applications, or applications that have no upstream repositories, the build process is initiated directly from the Jenkins web interface rather than with the **git push** command.

There are a number of benefits that come with using Jenkins to build applications:

- » Archived build information
- » No application downtime during the build process
- » Failed builds are not deployed; instead, a previous working version is left in place
- » Additional memory and storage resources are available
- » A large community of Jenkins plug-ins

Jenkins can be used to build any number of applications, and is only limited by the number of available gears. For example, if a **PHP** application is created and **MySQL** database is on the first gear, then Jenkins is added to a separate gear. A third gear is used for the Jenkins builder. In other words, whenever the Jenkins builder is active, it occupies one of the available gears.

[Report a bug](#)

13.7.2. Configuring Jenkins

13.7.2.1. Configuring Jenkins with New Applications

Jenkins is configured with a new application by using the **--jenkins-enable** option:

```
$ rhc app create App_Name App_Type --enable-jenkins Jenkins_App_Name
```

Add the **-s** to create a scalable application.

See Also:

» [Section 11.2, “Creating an Application”](#)

[Report a bug](#)

13.7.2.2. Configuring Jenkins with Existing Applications

The following instructions describe how to configure Jenkins with an existing application. Note that Jenkins can be configured with both scalable and non-scalable applications.

Procedure 13.5. To Configure Jenkins with an Existing Application:

1. Create the Jenkins application:

```
$ rhc app create App_Name jenkins-1
```

Example 13.4. Creating Jenkins Applications

```
$ rhc app create myjenkins jenkins-1
Application Options
-----
Domain:      mydomain
Cartridges:  jenkins-1
Gear Size:   default
Scaling:     no

Creating application 'myjenkins' ... done

Jenkins created successfully. Please make note of these
credentials:

User: admin
Password: Zek_Mdtr86uq

Note: You can change your password at: https://myjenkins-
mydomain.rhcloud.com/me/configure
.....
Run 'rhc show-app myjenkins' for more details about your app.
```

2. Add the Jenkins client cartridge to the specified application:

```
$ rhc cartridge add jenkins-client-1 -a App_Name
```

Example 13.5. Adding Jenkins Client to Application

```
$ rhc cartridge add jenkins-client-1 -a myapp
Adding jenkins-client-1 to application 'myapp' ... done

jenkins-client-1 (Jenkins Client)
-----
Gears:    Located with php-5.4
Job URL:  https://myjenkins-mydomain.rhcloud.com/job/myapp-
```

```
build/
```

Associated with job 'myapp-build' in Jenkins server.

[Report a bug](#)

13.7.3. Building Applications with Jenkins

Building applications with Jenkins uses dedicated application space, which can be larger than the application runtime space. The Jenkins online build system monitors applications that have an embedded Jenkins Client, and automatically rebuilds and deploys those applications whenever changes to the Git repository are pushed to the remote server without any further interaction. The existing application is not affected until a new, successful build has been created. If the build fails, the existing application continues to run. However, note that a failure in the deployment process (deploy - start - post_deploy) could leave the application partially deployed or inaccessible.

The actual build and deployment process that Jenkins executes involves the following steps:

1. The **git push** command is executed, and Jenkins is notified that a new push is ready.
2. A dedicated Jenkins slave (a *builder*) is created. The **rhc apps** command shows slave information. The application name is the same as that of the originating application, but with a **.bldr** suffix.



Important

The first 28 characters of the application name must be unique to avoid build issues that are caused when builders are shared across applications.

3. Jenkins runs the build.
4. Content from the originating application is downloaded to the builder application using **git** (for source code) and **rsync** (for existing libraries).
5. **ci_build.sh** is called from the Jenkins shell. This sets up the builder application for the Jenkins environment and performs some built-in bundling steps (PHP pear processing, Python virtual environment, etc).
6. **.openshift/action_hooks/build** is executed on the Jenkins builder.
7. Any additional desired steps are executed from the Jenkins shell (Maven build, Gem install, test cases, etc).
8. Jenkins stops the currently running application, and runs **rsync** to synchronize all new content over to the originating application.
9. **.openshift/action_hooks/deploy** is executed on the originating application.
10. Jenkins starts the originating application, and **.openshift/action_hooks/post_deploy** is executed on this application.
11. Jenkins archives all build artifacts for later reference.

12. After 15 minutes of idle time, the "build app" will be destroyed and will no longer appear in the output of the **rhc apps** command. The build artifacts, however, will still exist in Jenkins and can be viewed there.

The build job can be monitored using the Jenkins interface. The interface provides an extensive range of information about the current build, build history, artifacts, as well as plug-ins to graph, track, run tests, and perform other operations.

Log all errors related to Jenkins, such as DNS timeout and builder configuration, with the following command, specifying the name of the Jenkins application if it was changed:

```
$ rhc tail jenkins
```


Error logs for applications, such as compilation or test failures, are available from the Jenkins web interface under the corresponding build history. Deployment related errors are logged in the application's log files, and can be viewed with the following command:

```
$ rhc tail App_Name
```

[Report a bug](#)

13.7.3.1. Building Custom Applications

Build custom applications, or applications that have no upstream repositories, directly from the Jenkins web interface instead of using the **git push** command.

Click on the  icon of the application from the Jenkins web interface, located on the right side, to build it.

View the status of the build process in the web interface under the **Build Executor Status** section.

[Report a bug](#)

Chapter 14. Gear Storage and Disk Space Management

14.1. Introduction to Gear Storage and Disk Space

As an application is developed and the changes are pushed to the Git repository, the amount of available disk space to run an application slowly decreases. This is because Git stores all repository information, whether it is still required or not. Other aspects of developing and running applications also result in wasted disk space, such as old log files and unused application libraries. In such cases, either additional storage is required, or the existing disk space must be optimized to achieve the best possible application performance.

Gear storage and disk space can be managed with the client tools to optimize application performance.



Note

The feature to add additional gear storage is disabled by default. Contact your system administrator to enable this feature for your account. See the *OpenShift Enterprise Administration Guide* at <https://access.redhat.com/site/documentation> for more information on tracked and untracked storage administration.

[Report a bug](#)

14.2. Viewing Gear Storage

View the current gear storage allocation for each cartridge that exists in an application with the following command:

```
$ rhc cartridge storage --show -a App_Name
```

Example 14.1. Viewing Gear Storage

```
$ rhc cartridge storage --show -a myapp
```

RESULT:

MySQL Database 5.5

```
Base Gear Storage:      1GB
Additional Gear Storage: 3GB
```

OpenShift Web Balancer

```
Base Gear Storage:      1GB
Additional Gear Storage: None
```

PHP 5.4

```
Base Gear Storage:      1GB
Additional Gear Storage: None
```

View gear storage for a specific cartridge by using the **-c** option to specify the cartridge:

```
$ rhc cartridge storage --show -a App_Name -c Cart_Name
```

Example 14.2. Viewing Gear Storage for a Specific Cartridge

```
$ rhc cartridge storage --show -a myapp -c php-5
RESULT:
PHP 5.4
-----
Base Gear Storage:      1GB
Additional Gear Storage: None
```

[Report a bug](#)

14.3. Adding Gear Storage

Add a specified amount of gear storage to an application with the following command, specifying the application name and the amount of storage(GB) to add:

```
$ rhc cartridge storage Cart_Name -a App_Name --add Storage_Amount(GB)
```

Example 14.3. Adding Gear Storage

```
$ rhc cartridge storage php-5.4 -a myapp --add 3gb
Set storage on cartridge ... set to 3GB

Storage Info
-----
Base Gear Storage:      1GB
Additional Gear Storage: 3GB
```

If the same command is used to add another 1GB of storage, there will be a total of 4GB of additional gear storage.

[Report a bug](#)

14.4. Setting Gear Storage

Set a specific amount of gear storage for an application with the following command, using the **--set** option.

```
$ rhc cartridge storage php-5 -a App_Name --set Storage_Amount(GB)
```

Example 14.4. Setting Gear Storage

```
$ rhc cartridge storage php-5 -a racer --set 5gb
```

```
Set storage on cartridge ... set to 5GB
```

```
Storage Info
```

```
-----
```

```
Base Gear Storage:      1GB
```

```
Additional Gear Storage: 5GB
```

Note that this is different from the **--add** option because the exact amount of gear storage is specified, rather than adding more storage to the existing amount.

[Report a bug](#)

14.5. Removing Gear Storage

Remove a specified amount of gear storage with the following command, specifying the application name and the amount of storage to remove:

```
$ rhc cartridge storage Cart_Name -a App_Name --remove
Storage_Amount(GB)
```

Example 14.5. Removing Gear Storage

```
$ rhc cartridge storage php-5 -a myapp --remove 3gb
```

```
Set storage on cartridge ... 2GB
```

```
Storage Info
```

```
-----
```

```
Base Gear Storage:      1GB
```

```
Additional Gear Storage: 2GB
```

[Report a bug](#)

14.6. Tidying an Application

Tidying an application helps manage application disk space, and performs the following functions:

- » Run the **git gc** command on the application's Git repository on the server.
- » Clear the application's **/tmp** and log file directories that are specified by the application's **OPENSIFT_LOG_DIR** and **OPENSIFT_TMP_DIR** environment variables.
- » Clear unused application libraries and remove any library files previously installed by a **git push** command.



Important

Log files are not automatically backed up or rotated. Tidying an application runs the **rm -rf** command to clear the contents of these directories. Before performing this step, save the log files by creating a snapshot of the system with the **rhc snapshot save** command.

Tidy an application with the following command:

```
$ rhc app tidy App_Name
```

[Report a bug](#)

Chapter 15. Application Backup and Restoration with Snapshots

15.1. Introduction to Snapshots

Application snapshots are used to back up and restore applications. Snapshots are stored in **tar.gz** files, which contain the application and all local files, including log files.



Important

Application backups and user data are not stored on OpenShift Enterprise servers. These files are only stored on the local system.

Binary Deployment File Structure

A binary deployment file is a snapshot of an application used for deployment without using Git. Each snapshot has the same top-level structure:

```
build-dependencies/
dependencies/
repo/
```

The contents of the **repo** directory are the files that make up the application source code. The contents of the **build-dependencies** and **dependencies** directories are cartridge-specific, and are determined by the contents of the **managed_files.yml** file.

The following information on the **managed_files.yml** file dictates the construction of the snapshot:

- ✧ If an entry under **dependency_dirs** starts with **~/**, then a new folder will be created in the **dependencies** directory.
- ✧ If an entry under **dependency_dirs** does not start with **~/**, then a directory will be created in the **dependencies/Cartridge_Name** directory.
- ✧ If an entry under **dependency_dirs** has the format **key: Value**, then **Value** will be the directory name for the previous rules.

Example 15.1. A Sample Node.js **managed_files.yml** File

The following is an excerpt from the **managed_files.yml** file on a Node.js cartridge:

```
dependency_dirs:
- ~/.npm
- node_modules
- ~/.node_modules: node_modules
```

Using the above rules, this means that the **.npm** file will be in the **dependencies** directory, and the **node_modules** file will be in the cartridge dependencies directory **dependencies/nodejs** directory:

```
.npm
nodejs/node_modules
```

These same rules apply to the information under **build_dependencies** in the **managed_files.yml** file, though the **build_dependencies** file is mainly used in JBoss cartridges.

This information can be used to prepare your own **managed_files.yml** file for deployment. See [Section 13.1, “Introduction to Deployment”](#) for information on deployment, and [Section 13.3.2.4, “Deploying from a Snapshot”](#) for information on deploying from a binary snapshot file.

[Report a bug](#)

15.2. Creating an Application Snapshot

Create an application snapshot with the following command:

```
$ rhc snapshot save App_Name
```

Example 15.2. Creating an Application Snapshot

```
$ rhc snapshot save myapp
Pulling down a snapshot to myapp.tar.gz...
Creating and sending tar.gz
```

```
RESULT:
Success
```

The command prompts for any required information. The default filename for the snapshot is **\$App_Name.tar.gz** and is created in your current directory. Choose a different filename or file path by using the **--filepath** option to override the defaults.

[Report a bug](#)

15.3. Restoring from an Application Snapshot

Restoring from an application snapshot restores the Git repository, the application data directories, and the log files found in the specified archive. When the restoration is complete, the deployment script is run on the restored repository as though **git push** was run.



Warning

The **rhc snapshot restore** command overwrites the remote Git repository. Therefore, any changes made since taking the snapshot are lost. Importing snapshot data into a local environment can delete local content, for example a user table in a database. If you are unsure of the effect a snapshot import could have on local data, use SSH to access an application and create the backup directly.

Restore an application from an application snapshot with the following command, specifying the name of the application:

```
$ rhc snapshot restore App_Name
```

Example 15.3. Restoring from an Application Snapshot

```
$ rhc snapshot restore myapp
Restoring from snapshot myapp.tar.gz...
Removing old git repo: ~/git/myapp.git/
Removing old data dir: ~/app-root/data/*
Restoring ~/git/App_Name.git and ~/app-root/data
Activation status: success
```

```
RESULT:
Success
```

If the override process was used to save an application under a different filename, as described in [Section 15.2, “Creating an Application Snapshot”](#), you can restore this snapshot version of an application with the following command:

```
$ rhc snapshot restore App_Name --filepath Renamed_App
```

where *App_Name* is the name of the application, and *Renamed_App* is the file path where it was saved.

[Report a bug](#)

15.4. Migrating an Application to Another Gear

There may be cases when an application must be migrated to another gear.

Procedure 15.1. To Migrate an Application to Another Gear:

1. Create a snapshot of an existing application:

```
$ rhc snapshot save App_Name
```

2. Verify that the ***App_Name.tar.gz*** file has been created in the working directory. After confirming the application snapshot is saved, delete the existing application:

```
$ rhc app delete App_Name
```

3. Create a new application using the same cartridges, but with the correct gear size:

```
$ rhc app create App_Name Cart_Name -g gear_size
```

4. Finally, restore the previously saved application snapshot to the newly created application. Be sure to specify the correct path to the saved application snapshot:

```
$ rhc snapshot restore App_Name -f App_Name.tar.gz
```

[Report a bug](#)

Appendix A. Revision History

Revision 2.2-3	Tue Jun 09 2015	Vikram Goyal
BZ 1117560: Updated introduction for Section 6.1, "Introduction to Teams" .		
Revision 2.2-2	Wed Dec 03 2014	Bilhar Aulakh
OpenShift Enterprise 2.2.2 release. BZ 1163906: Fixed Ruby environment variable name in Section 13.5.10, "Ruby Environment Variables" .		
Revision 2.2-0	Tue Nov 4 2014	Timothy Poitras
OpenShift Enterprise 2.2 release. Added Section 3.3, "Mutual SSL" . Updated Section 9.1.1, "Web Framework Cartridges" for Ruby 2.0. Updated Section 11.2, "Creating an Application" and Section 11.3, "Cloning an Existing Application" with details about specifying a region for applications. Updated Section 11.7, "Making Applications Highly Available" for <code>rhc app enable-ha</code> usage.		
Revision 2.1-7	Thu Oct 23 2014	Brice Fallon-Freeman
BZ 1152344: Updated Section 4.1.5, "Viewing a Server" to fix an example. BZ 1125298: Added Section 5.2.6, "Configuring Domain Gear Size" which explains how to control gear sizes on a domain. Updated Section 9.1.1, "Web Framework Cartridges" for the new JBoss A-MQ, JBoss Fuse, and JBoss Fuse Builder cartridges. BZ 1118766: Added Section 10.1.3, "Highly-Available Applications" and Section 11.7, "Making Applications Highly Available" .		
Revision 2.1-6	Thu Oct 2 2014	Alex Dellapenta
OpenShift Enterprise 2.1.7 release. BZ 1134034: Updated Section 13.5.4, "Logging Environment Variables" to include the <code>multi</code> option for the <code>LOGSHIFTER_OUTPUT_TYPE</code> environment variable.		
Revision 2.1-5	Tue Aug 26 2014	Bilhar Aulakh
OpenShift Enterprise 2.1.5 release. Updated Section 9.1.1, "Web Framework Cartridges" and Section 9.1.2, "Add-on Cartridges" with technology versions supported by each cartridge. BZ 1124980: Updated Section 11.2, "Creating an Application" with information on creating an application from code.		
Revision 2.1-3	Mon Aug 11 2014	Brice Fallon-Freeman
Added Chapter 4, Multiple OpenShift Servers . Added Section 8.1, "Introduction to Regions and Zones" . BZ 1119469: Removed "Embedding 10gen MMS Agent" section.		
Revision 2.1-2	Thu Jun 26 2014	Julie Wu
OpenShift Enterprise 2.1.2 release. BZ 999529: Updated Section 9.1.1, "Web Framework Cartridges" and Section 9.1.2, "Add-on Cartridges" with more information on cartridges. BZ 1102736: Updated Section 6.1, "Introduction to Teams" to remove note about team management support only being in the gem version of the client tools. BZ 1107753: Updated Section 9.1.2, "Add-on Cartridges" with correct add-on cartridge information. Updated various sections to highlight OpenShift Enterprise 2.1 features.		

Revision 2.1-1	Thu Jun 5 2014	Brice Fallon-Freeman
OpenShift Enterprise 2.1.1 release. Added Team Management section. Updated Section 9.1.2, "Add-on Cartridges" and Section 11.9.4, "Accessing a Database Cartridge" for the addition of the MongoDB cartridge.		
Revision 2.1-0	Fri May 16 2014	Brice Fallon-Freeman
OpenShift Enterprise 2.1 release. BZ 1086697 Edited SSH Key information. BZ 1058255 Added File Structure to Section 15.1, "Introduction to Snapshots" . Added team information to the topics on Domain Membership. Added Teams chapter and Section 6.1, "Introduction to Teams" . Added info on the --mine option to Section 11.5, "Viewing Applications for a User" . Added Section 13.5.4, "Logging Environment Variables" . Added Section 11.3, "Cloning an Existing Application" . BZ 1077965: Fixed image in Section 10.1.2, "Scalable Applications" . Added information about Watchman. BZ 1016151: Fixed command and example in Section 11.14.2, "Application Port Forwarding" . BZ 1033360: Removed references to OpenShift Online plans. BZ 1065804: Fixed command error. Updated Section 13.5.9, "JBoss Environment Variables" . Added Section 13.5.6, "Library Environment Variables" . Added Section 13.5.11, "Python Environment Variables" . BZ 1023944: Updated Section 11.9.4, "Accessing a Database Cartridge" . Updated Section 13.5.3, "Directory Environment Variables" . Added Section 13.4.5, "Metrics Action Hooks" . BZ 1076233: Fixed broken images. Restructured book.		
Revision 2.0-1	Thu Feb 27 2014	Bilhar Aulakh
OpenShift Enterprise 2.0.3 release. BZ 1051190: Added Section 13.4.4, "Scaling Action Hooks" . BZ 1033360: Removed references to OpenShift Online plans.		
Revision 2.0-0	Mon Dec 9 2013	Bilhar Aulakh
OpenShift Enterprise 2.0 release. Added two new topics about action hooks for cartridges and the build process. Added information on configuring application deployment. Edited 'How Scaling Works' section. Removed specific cartridge version numbers.		