OPENSHIFT

OpenShift All Versions User Guide

Using OpenShift to manage your applications in the cloud

OpenShift All Versions User Guide

Using OpenShift to manage your applications in the cloud

Legal Notice

Keywords

Abstract

This guide provides an introduction to OpenShift and documents its application management functions.

Table of Contents

Preface

1. Document Conventions

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

In PDF and paper editions, this manual uses typefaces drawn from the Liberation Fonts set. The Liberation Fonts set is also used in HTML editions if the set is installed on your system. If not, alternative but equivalent typefaces are displayed. Note: Red Hat Enterprise Linux 5 and later includes the Liberation Fonts set by default.

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 sub-menu titles. For example:

> Choose **System** → **Preferences** → **Mouse** from the main menu bar to launch **Mouse Preferences**. In the **Buttons** tab, click 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:

```
package org.jboss.book.jca.ex1;

import javax.naming.InitialContext;

public class ExClient
{
   public static void main(String args[])
       throws Exception
   {
      InitialContext iniCtx = new InitialContext();
      Object         ref    = iniCtx.lookup("EchoBean");
      EchoHome       home   = (EchoHome) ref;
      Echo           echo   = home.create();

      System.out.println("Created Echo");

      System.out.println("Echo.echo('Hello') = " + echo.echo("Hello"));
   }
}
```

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

2.1. Do You Need Help?

If you experience difficulty with a procedure or other information described in this documentation, visit the Red Hat Knowledgebase at http://kbase.redhat.com to search or browse through technical support articles about Red Hat products, or visit the Red Hat Customer Portal at http://access.redhat.com. You can also access the OpenShift web site at https://openshift.redhat.com/ to find blogs, FAQs, forums, and other sources of information.

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 or any other 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 Origin.**

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

OpenShift is Red Hat's Platform as a Service (PaaS) offering. OpenShift is an application platform in the cloud where application developers and teams can build, test, deploy, and run their applications with automatic scaling. OpenShift provides developers with a wide selection of programming languages and frameworks including Java, Ruby, PHP, Perl, Python, and Node.js. It also provides integrated developer tools to support the application lifecycle, including Eclipse integration, JBoss Developer Studio, Jenkins, Maven, and GIT. OpenShift uses an open source ecosystem to provide key platform services for mobile applications (Appcelerator), NoSQL services (MongoDB), SQL services (Postgres, MySQL), and more. JBoss provides an enterprise-class middleware platform for Java applications, providing support for Java EE6 and integrated services such as transactions and messaging, which are critical for enterprise applications.

The foundation of the OpenShift platform is Red Hat Enterprise Linux, which provides a secure and scalable multi-tenant operating system to address the needs of enterprise-class applications as well as providing integrated application runtimes and libraries.

This document describes how to navigate and utilize an OpenShift environment. It provides information on overall architecture, common application management tasks via web and command line interfaces, and basic troubleshooting. It is intended for developers and administrators of applications on OpenShift.

Chapter 2. OpenShift Overview

2.1. Platform Overview

OpenShift enables you to create, deploy and manage applications within the cloud. It provides disk space, CPU resources, memory, network connectivity, and an Apache or JBoss server. Depending on the type of application you are building, you also have access to a template file system layout for that type (for example, PHP, Python, and Ruby/Rails). OpenShift also generates a limited DNS for you.

OpenShift 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 not touched in these directories for any 10-day period are deleted.

The two basic functional units of OpenShift are the Broker, which provides the interface, and the Cartridges, which provide application frameworks.

Broker

The Broker is the single point of contact for all application management activities. It is responsible for managing user logins, DNS, application state, and general orchestration of the application. User interaction with the broker is generally performed using either the Web console, CLI tools, or JBoss tools, using a REST-based API.

Cartridges

Cartridges provide the actual functionality necessary 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.

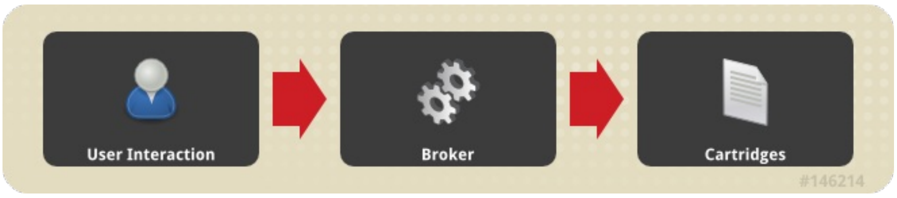


Figure 2.1. High-level OpenShift Platform Overview

2.2. System Resources and Application Containers

The system resources and security containers provided by the OpenShift platform are gears and nodes.

- Gears — Gears provide a resource-constrained container where you can run one or more cartridges. They limit the amount of RAM and disk space available to a cartridge.

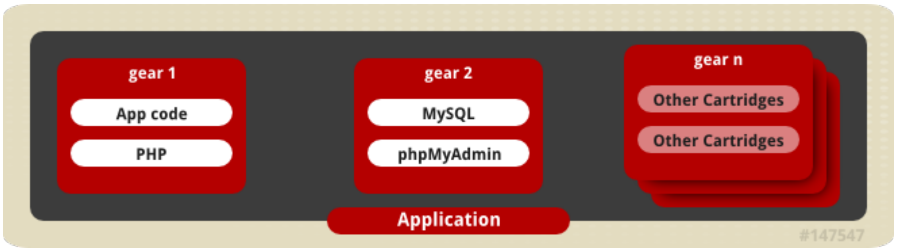


Figure 2.2. Application Cartridges on Gears

OpenShift currently provides two gear sizes:

  - Small — provides 512MB of RAM, 100MB of swap space, and 1GB of disk space
  - Medium — provides 1GB of RAM, 100MB of swap space, and 1GB of disk space

  By default, you can use up to three small gears (a total of 1.5GB of RAM and 3GB of disk space). OpenShift can assign these three gears to a single application and its cartridges (Cron, MySQL, etc.) or you can use each gear for a separate application.

- Nodes — To enable resource sharing, multiple gears run on a single physical or virtual machine. This machine is referred to as a node host.

Chapter 3. OpenShift User Interfaces

Note

Refer to the *Client Tools Installation Guide* to ensure that you have performed all of the necessary steps to set up your environment for OpenShift.

3.1. OpenShift Web Interface

3.1.1. Accessing the OpenShift Management Console

OpenShift applications can be created and managed using the *OpenShift Management Console*, a graphical user interface accessed with a browser. The Management Console also allows you to manage your OpenShift account settings and provides links to OpenShift documentation and community resources.

Procedure 3.1. To access the OpenShift Management Console:

1. On the OpenShift homepage, click **`SIGN IN TO MANAGE YOUR APPS`** in the upper-right corner.
2. On the sign in screen, enter your login name and password details then click **`Sign in`**.

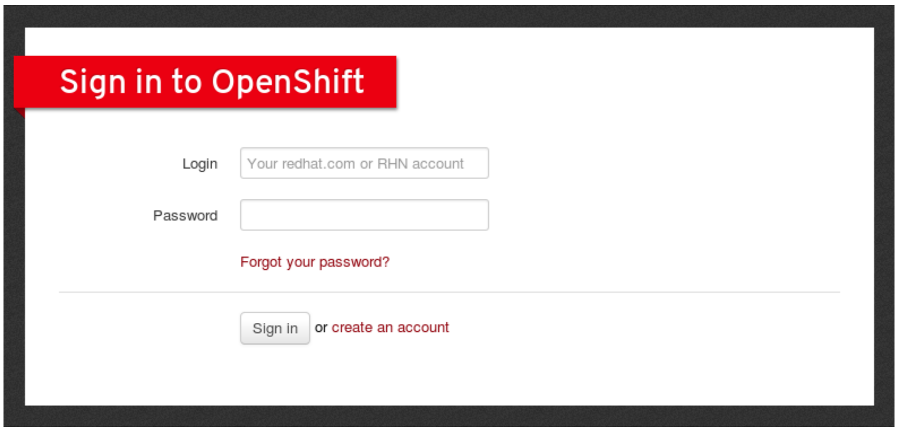


Figure 3.1. Sign In Screen

If you do not have any applications, the **`Create a New Application`** screen opens. If you have applications, you are taken to the **`My Applications`** screen.

3.2. OpenShift Command Line Interface

Refer to the *Client Tools Installation Guide* for instructions on how to install the command line OpenShift.

For a full list of client tool commands, run **`rhc --help`**.

Chapter 4. Account Management

4.1. View Account Details

For a quick overview of the currently logged in user and their capabilities on a given server, run **`rhc account`** from the command line:

```
$ rhc account
Login:               default
Plan:                FreeShift
Gears Used:          3
Gears Allowed:       3
Server               openshift.redhat.com
Allowed Gear Sizes: small
```

4.2. Changing Your Password

Your OpenShift user account password can be changed using the Management Console.

When choosing a password we recommend using a combination of numbers, symbols, and upper and lower case letters for extra security.

Procedure 4.1. To change your password:

1. Access the Management Console and click **`My Account`** in the navigation bar at the top of the page.
2. In the **`Personal Information`** section, click **`Change password`** then follow the on-screen instructions.

4.3. Resetting Your Password

If you forget your OpenShift user account password, you can have a new password sent to your email address.

Procedure 4.2. To reset your password:

1. On the OpenShift homepage, click **`SIGN IN TO MANAGE YOUR APPS`** in the upper-right corner.
2. On the sign in screen, click **`Forgot your password?`**
3. Enter your email address and click **`Reset Password`**.

A new password is sent to your email address. Use this password to access the OpenShift Management Console and change your password.

Chapter 5. Namespaces

You must create a namespace before you can create an application. OpenShift uses non-strict domain names (that is, there is no preceding period), and the domain name forms part of the application name. The syntax for the application name is ***`ApplicationName–namespace`***.example.com.

Each username can only support a single namespace, but you can create multiple applications within the namespace. If you need multiple namespaces, you need to create multiple accounts using different usernames.

5.1. Management Console

5.1.1. Altering a Namespace

Altering your namespace deletes the old namespace and creates a new one. It also automatically updates the public URLs and repository addresses for your applications. In order to **`git push`** future changes to your applications after changing your namespace, the git **`config`** file must be updated with the new repository address.

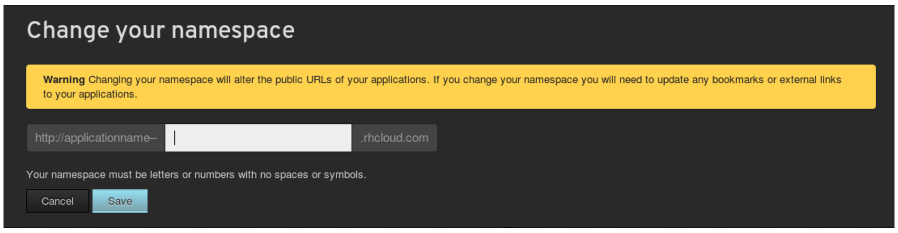
OpenShift uses a blacklist to restrict the list of available namespace and application names that you can use. This list is maintained on the server. If you try to change your namespace to any members of this blacklist, the command will fail.

Important

This procedure alters the URLs for your applications. You need to update any bookmarks and external links you have made to these URLs. Links made using an alias do not need to be changed. See Section 6.3.2.4, "Using Arbitrary DNS Names".

Procedure 5.1. To change your namespace:

1. Access the Management Console and click **`My Account`** in the navigation bar at the top of the page.
2. Scroll down to the **`Namespace`** section and click **`Change your namespace`**.
3. Enter your desired namespace in the box provided and click **`Save`**.



Change your namespace

Warning Changing your namespace will alter the public URLs of your applications. If you change your namespace you will need to update any bookmarks or external links to your applications.

http://applicationname– | .rhcloud.com

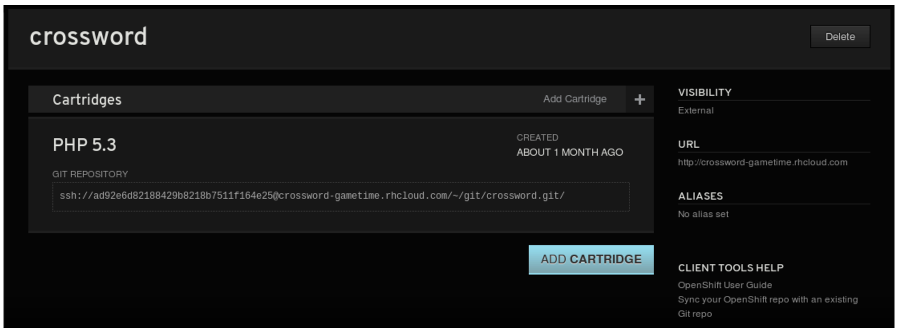
Your namespace must be letters or numbers with no spaces or symbols.

Figure 5.1. Change Namespace Dialog

The public URLs and repository addresses of your applications automatically update with the new namespace. To enable the **`git push`** command to function properly, update the git **`config`** file using the following procedure.

Procedure 5.2. To update the git config files:

1. Access the Management console and click **`My Applications`** in the navigation bar at the top of the page.
2. Click on your first application.
3. Copy the entire SSH address located in the **`GIT REPOSITORY`** box.
4. Open the git **`config`** file located in ***`path/to/appdirectory/`*****`.git/`** and replace the remote URL address with the new SSH address for your application.
5. Repeat the previous four steps for each of the applications in the altered namespace to update their git **`config`** files.


crossword

Delete

Cartridges

Add Cartridge +

PHP 5.3

CREATED
ABOUT 1 MONTH AGO

GIT REPOSITORY

```
ssh://ad92e6d82188429b8218b7511f164e25@crossword-gametime.rhcloud.com/~/git/crossword.git/
```

ADD CARTRIDGE

VISIBILITY
External

URL
http://crossword-gametime.rhcloud.com

ALIASES
No alias set

CLIENT TOOLS HELP
OpenShift User Guide
Sync your OpenShift repo with an existing Git repo

Figure 5.2. Application Details Screen Showing Git Repository Address

5.2. Command Line Interface

5.2.1. Creating a Namespace

Use the **`rhc domain create`** command to create a new namespace. This command uses the following syntax:

```
$ rhc domain create <namespace> -l <rhlogin> -p <password> [OptionalParameters]
```

- ***`namespace`*** — specifies the namespace that you want to create. This must contain a maximum of 16 alphanumeric characters.
- ***`rhlogin`*** — this can be your Red Hat Network login or the email address that you used to apply for your OpenShift account.
- ***`password`*** — the password that you used to apply for your OpenShift account.

Optional Parameters

Further optional parameters exist that you can pass to the **`rhc domain create`** command. Refer to **`rhc domain create --help`** for details.

Note

The **`rhc domain create`** command creates a local configuration file in **`~/.openshift/express.conf`** and updates it with your rhlogin. The rhc client tools use the login specified in the **`~/.openshift/express.conf`** file by default when you run further commands. If no default login is configured, or if you want to use a different login, include the **`-l`** ***`rhlogin`*** parameter as a command line argument.

Example 5.1. Creating a new domain

The following example demonstrates how to create a new domain.

```
$ rhc domain create automobile -l jsmith@mail.com

Creating domain with namespace 'automobile'

Password: *******

RESULT:
Success!
You may now create an application using the 'rhc app create' command
```

Note

You do not need SSH keys to create a domain. You only need to create SSH keys and upload them to the server when you want to create applications and communicate with your domain. OpenShift uses a *blacklist* to restrict the list of available namespace and application names that you can use. This list is maintained on the server. If you try to use **`rhc domain create`** or **`rhc app create`** with any members of this blacklist, the command will fail.

5.2.2. Viewing a Namespace

Use the **`rhc apps`** command to display information about all of the current applications in your namespace.

The following example demonstrates how to display application information for **`user`**:

```
$ rhc apps
Password: *******

racer @ http://racer-automobile.example.com/ (uuid:
926056f8845b4e388b37f6735c89d0eb)
--------------------------------------------------------------------------------
-----
  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

  php-5.3 (PHP 5.3)
  -----------------
    Gears: 1 small
```

5.2.3. Altering a Namespace

If you need to modify the name of an existing namespace, use the **`rhc domain update`** command as shown below.

Important

The **`rhc domain update`** command alters the URLs of your applications. You need to update any bookmarks and external links you have made to these URLs. Links made using an alias do not need to be changed. Refer to Section 6.3.2.4, “Using Arbitrary DNS Names” for more information.

```
$ rhc domain update <old_namespace> <new_namespace>
Password: *******

Changing namespace 'old_namespace' to 'new_namespace'...

RESULT:
Success!
You can use 'rhc domain show' to view any url changes. Be sure to update any links
including the url in your local git config: <local_git_repo>/.git/config
```

Note

If no SSH keys are found in the **`.ssh`** folder of your home directory, the **`rhc domain update`** command generates a new pair of keys for you. You will then be prompted to upload the public SSH key to the OpenShift server. Refer to the *SSH Authentication* chapter for more information.

5.2.4. Deleting a Namespace

If you no longer need a particular namespace, you can use the **`rhc domain delete`** command to delete that namespace. Before you can delete a namespace, however, you need to ensure that it does not contain any applications.

Procedure 5.3. How to delete a namespace

1. Ensure that the namespace does not contain any applications.

   ```
   $ rhc apps
   No applications. Use 'rhc app create'.
   ```

   If any applications are listed, use the **`rhc app delete`** command to remove them.

   ```
   $ rhc app delete <ApplicationName>
   ```

   Warning

   Application removal is irreversible. All remote data associated with the application will be removed.

2. Use the **`rhc domain delete`** command to delete the namespace.

```
$ rhc domain delete <namespace>
Password: **********

Deleting domain 'namespace'

RESULT:
Success!
```

After you have deleted your namespace, you need to create a new one before you can create any new applications or use the other client tools.

Chapter 6. Applications and Cartridges

Note

Before you start working through the procedures and examples in the following chapters, refer to the *Client Tools Installation Guide* to ensure that you have performed all of the necessary steps to set up your environment for OpenShift.

6.1. Applications Overview

When you create an OpenShift application, a domain name that is a combination of the application name and the namespace of your domain is registered in DNS. A copy of the application code is checked out locally in a folder with the same name as the application you create. OpenShift runs the components of your application on small virtual servers, referred to as "gears".

The following diagram illustrates the relationships between usernames, namespaces, domain names, and applications:

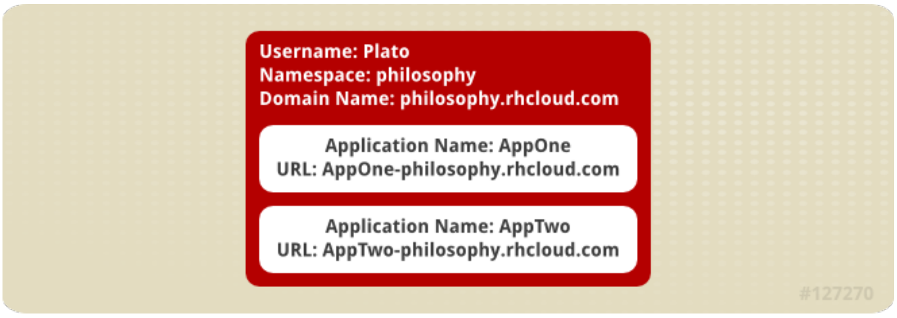

Username: Plato
Namespace: philosophy
Domain Name: philosophy.rhcloud.com
Application Name: AppOne
URL: AppOne-philosophy.rhcloud.com
Application Name: AppTwo
URL: AppTwo-philosophy.rhcloud.com
#127270

Figure 6.1. Name Relationships

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

Table 6.1. Application Components

Component	Description
Namespace	The namespace is not directly related to DNS, instead it provides a unique group identifier for all the applications of a specific user. The namespace is appended to the application name to form a final application URL of the form http://[APP NAME]-[NAMESPACE].example.com
Application Name	User-selected name of the application. The final URL to access the application is of the form http://[APP NAME]-[NAMESPACE].example.com
Alias	Users can provide their own DNS names for the application by registering an *alias* with OpenShift and pointing the DNS entry to the OpenShift servers.
Git repository	Allows you to modify your application code locally; you can then run the **`git push`** command to deploy the revised code.

6.1.1. Scaled Applications

Application scaling provides for the automatic allocation of resources based on demand. OpenShift monitors the resource requirements of scaled applications, and increases or decreases available resources accordingly. You need to specify whether an application is scaled or not when you create the application. A scaled application cannot be converted to a non-scaled application. Likewise, a non-scaled application cannot be converted to a scaled application.

6.1.1.1. How Scaling Works

Each application created on OpenShift always has the web cartridge associated with it. The web cartridge can, for example, be a PHP cartridge. When an application is scaled, a second cartridge, called HAProxy, is added to the application. The HAProxy cartridge listens to all incoming web page requests for an application and passes them on to the web cartridge, following defined guidelines for load monitoring.

As the number of web page requests to an application increase, the HAProxy will inform OpenShift when an overload of requests is detected. OpenShift will then create a copy of the existing web cartridge on a separate gear. In such a case, the web cartridge now has 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.

However, not all OpenShift applications can be scaled, as detailed in the table below.

Table 6.2. Applications that can or cannot be scaled

Type of Application	Scalable
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
Python	Yes
Perl	Yes
Ruby	Yes
Node.js	Yes
Jenkins	No
HAProxy	No
Zend Server	No
DIY	No

Note

You can add only MySQL 5.1, MongoDB, PostgreSQL, or Jenkins Client 1.4 cartridges to scaled applications.

6.1.1.2. Automatic and Manual Scaling

By default, when you create a scaled OpenShift application, it is automatically scaled based on the number of requests. But OpenShift also allows you to manually scale your application by disabling the automatic scaling feature.

Scaled applications can be created with the OpenShift client tools using the CLI.

6.1.2. Non-scaled Applications

If you create a non-scaled application, it only consumes one of the default quota of gears assigned to users. That is, it only consumes one of the available three gears. If you create a scaled application, it consumes two of the available gears; one for the high-availability proxy (HAProxy) itself, and one for your actual application. If you add MySQL to your application, it is installed in its own dedicated gear.

6.2. Cartridges Overview

Every OpenShift application you create must have one web cartridge to serve web requests. Web cartridges are available for a variety of programming languages, such as PHP, JBoss, and Ruby. You can add a number of other cartridges that provide enhanced capabilities for your applications.

Note

Web cartridges can only be added to new applications.

Use the **`rhc cartridge list`** command to view the current list of all cartridges available on OpenShift.

6.2.1. Addon Cartridges

After you create an OpenShift application with the required web cartridge, you can then add a number of other cartridges that provide capabilities like databases, scheduled jobs, or continuous integration. The table below describes the functionality of the different types of add-on cartridges available with OpenShift.

Table 6.3. Add-on Cartridge Functions

Function	Description
Database	Provide your application with one of several database back ends. Examples include MySQL and PostgreSQL.
Database management	Provide functionality for managing your application's database using third-party software. Examples include phpMyAdmin, RockMongo, and 10gen's MongoDB Monitoring Service (MMS).
Monitoring and Management	Provide a range of options for managing and monitoring your application. Examples include the Cron task scheduler, the Jenkins Client, and OpenShift Metrics.

Again, you can run the **`rhc cartridge list`** command to view the current list of available add-on cartridges. At the time of this writing, the following add-on cartridges are available:

Database Cartridges

- MySQL Database 5.1 — MySQL is a multi-user, multi-threaded SQL database server
- MongoDB NoSQL Database 2.2 — MongoDB is a scalable, high-performance, open source NoSQL database
- PostgreSQL Database 8.4 — PostgreSQL is an advanced object-relational database management system

Management Cartridges

- phpMyAdmin 3.4 — phpMyAdmin is a web-based MySQL administration tool
- RockMongo 1.1 — RockMongo is a web-based MongoDB administration tool
- 10gen MMS agent 0.1 — 10gen's MongoDB Monitoring Service (MMS)
- Jenkins Client 1.4 — a client for managing Jenkins-enabled applications
- HAProxy 1.4 — a high-performance TCP/HTTP load balancer
- Cron 1.4 — Cron is a daemon that runs specified programs at scheduled times
- OpenShift Metrics 0.1 — OpenShift Metrics is an experimental cartridge for monitoring applications
- SwitchYard 0.6 — SwitchYard is a lightweight service delivery framework providing full lifecycle support for developing, deploying, and managing service-oriented applications

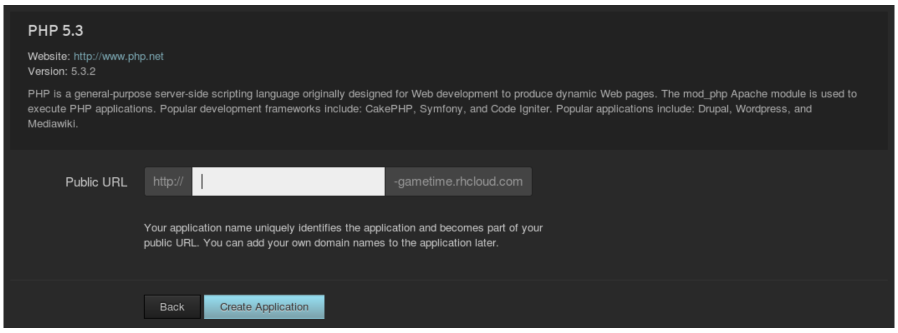
6.3. Creating an Application

6.3.1. Management Console

Creating applications using the OpenShift Management Console is a simple process.

Procedure 6.1. To create an application:

1. Access the Management Console and click **`Create Application`** in the navigation bar at the top of the page.
2. Click on the application type you wish to create from the available list of applications.
3. Type a name for your application in the box provided and configure the application to suit your requirements. For example, select whether you want to create a scaled or a non-scaled application. Click **`Create Application`**.



PHP 5.3

Website: http://www.php.net
Version: 5.3.2

PHP is a general-purpose server-side scripting language originally designed for Web development to produce dynamic Web pages. The mod_php Apache module is used to execute PHP applications. Popular development frameworks include: CakePHP, Symfony, and Code Igniter. Popular applications include: Drupal, Wordpress, and Mediawiki.

Public URL: http:// [ ] -gametime.rhcloud.com

Your application name uniquely identifies the application and becomes part of your public URL. You can add your own domain names to the application later.

Figure 6.2. Create Application Dialog

6.3.1.1. Creating Instant Applications

In addition to the standard application web cartridges, OpenShift provides several instant applications that allow you to create complete applications quickly and easily. Instant applications are automatically created with a web cartridge, any other required cartridges (such as a database), and all of the necessary code for a fully functioning application.

Procedure 6.2. To create an instant application:

1. Access the Management Console and click **`Create Application`** in the navigation bar at the top of the page.
2. Click **Browse by tag...** and select **All instant applications** from the pull-down menu. Then click on the instant application you wish to create.
3. Type a name for your application in the box provided and configure the application to suit your

requirements, and then click **`Create Application`**.

As instant applications are more complicated to build than basic applications, they can take longer to become available online. If you receive a 503 Service Unavailable error when attempting to view your instant application after you create it, wait a few minutes and try again.

6.3.1.2. Cloning Application Files

After you have created an application using the OpenShift Management Console, run the **`git clone`** command to copy the application's remote repository into your local working directory.

Procedure 6.3. To clone the remote repository:

1. Click **`My Applications`** in the navigation bar at the top of the page and click on the application you want to clone.
2. Copy the entire SSH address located in the **`GIT REPOSITORY`** box.
3. Open a terminal and use the following command to clone the remote repository to the working directory, replacing the example SSH address with the address for your application:

```
$ git clone ssh://a261d0fc2932413694456e3473fdc972@crossword-
gametime.example.com/~/git/crossword.git/
```

The **`git clone`** command copies the template application files from the remote repository into the working directory. Edit the template application files to develop your own application.

6.3.2. Command Line Interface

The **`rhc app create`** command from the OpenShift client tools is used to create a new application. When a new application is created, a new, remote git repository is created and automatically cloned to your current directory on your local machine. Further, the hostname and IP address of your application is added to the list of known hosts (**`~/.ssh/known_hosts`**).

Refer to **`rhc app create --help`** for a complete list of options.

Prerequisites

Application creation requires a reliable network connection. The **`rhc app create`** command only makes a single attempt to create your application. OpenShift makes up to seven checks for the DNS entry to exist, and returns a failure message if not found.

If you continue to experience timeout issues, you may need to use the **`--timeout`** option on the command line to override the default values. OpenShift 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* values from the default. You cannot set the timeout value to be less than the default. For example, if you use **`--timeout 50`**, it sets the connection timeout value to 50 seconds, but does not affect the read timeout value. If you use **`--timeout 150`**, it sets both the connection and read timeout values to 150 seconds.

6.3.2.1. Creating Non-scaled Applications

Use the **`rhc app create`** command without any options to create a non-scaled OpenShift application with the specified name, and in your domain namespace.

The following example demonstrates creating a PHP application called "racer" in the **`automobile`** domain that was created in Example 5.1, “Creating a new domain”:

```
$ rhc app create racer php-5.3
Password: *******

Creating application 'racer'
============================

Gear Size: default
Namespace: automobile
Cartridge: php-5.3
Scaling:   no

Your application's domain name is being propagated worldwide (this might take a
minute)...
Initialized empty Git repository in /root/apps/racer/.git/
done

racer @ http://racer-automobile.example.com/
============================================
Application Info
================
Git URL   = ssh://732b4727894f4afc3sep73796db80ac3@racer-
automobile.example.com/~/git/racer.git/
Created   = Dec 19  8:53 PM
SSH URL   = ssh://732b4727894f4afc3sep73796db80ac3@racer-automobile.example.com
UUID      = 732b4727894f4afc3sep73796db80ac3
Gear Size = small
Cartridges
==========
php-5.3

RESULT:
Application racer was created.
```

As mentioned in Section 5.2.1, “Creating a Namespace”, each domain can support multiple applications. By running **`rhc app create hybrid php-5.3`**, for example, you could create another application in the **`automobile`** namespace. The application URLs would appear as follows:

- http://racer-automobile.example.com
- http://hybrid-automobile.example.com

6.3.2.2. Creating Scaled Applications

Scaled applications automatically allocate resources based on demand. Use the **`rhc app create`** command with the **`-s`** (or **`--scaling`**) option to create a scaled application, as shown in the example below. Refer to Section 6.1.1, “Scaled Applications” for more information.

```
$ rhc app create hybrid php-5.3 -s
Password: **********

Creating application 'hybrid'
==============================

  Scaling:   yes
  Gear Size: default
  Cartridge: php-5.3
  Namespace: automobile

Your application's domain name is being propagated worldwide (this might take a
minute)...
Initialized empty Git repository in /apps/hybrid/.git/
done

hybrid @ http://hybrid-automobile.example.com/
====================================================
  Application Info
  ================
    SSH URL   = ssh://fjoe04cabdc4efa8f2513a21e2ed27d@hybrid-
automobile.example.com
    UUID      = fjoe04cabdc4efa8f2513a21e2ed27d
    Git URL   = ssh://fjoe04cabdc4efa8f2513a21e2ed27d@hybrid-
automobile.example.com/~/git/hybrid.git/
    Created   = 12:53 AM
    Gear Size = small
  Cartridges
  ==========
    php-5.3
    haproxy-1.4
  Scaling Info
  ============
    Scaled x2 (minimum: 2, maximum: available gears) with haproxy-1.4 on small
gears

RESULT:
Application hybrid was created.
```

When you create a scaled application, the automatic scaling feature is enabled by default. However, you can manually scale your application by disabling the automatic scaling feature.

6.3.2.2.1. Disabling Automatic Scaling

There may be cases where you may want to scale your application manually. Such cases may include:

- If you are anticipating a certain load on your application and wish to scale it accordingly.
- You have a fixed set of resources for your application.
- You want to manually control the cost.

You can disable the automatic scaling feature to manually control your application's scaling function. The instructions below describe how to disable the automatic scaling feature. It is assumed you have already created your scaled application.

Procedure 6.4. To disable the automatic scaling feature:

1. From your locally cloned Git repository, create a disable autoscaling marker, as shown in the example below.

```
$ touch <AppName>/.openshift/markers/disable_auto_scaling
```

For example, to create a disable autoscaling marker for an application named *myapp*, run the command as follows:

```
$ touch myapp/.openshift/markers/disable_auto_scaling
```

2. Add and commit your changes to your local Git repository.
   a. Change to your application's directory using the following command:

   ```
   $ cd <AppName>
   ```

   b. Use the **`git add`** command to add your changes to your local Git repository.

   ```
   $ git add .openshift/markers/disable_auto_scaling
   ```

   c. Use the **`git commit`** command to commit your changes to your local Git repository.

   ```
   $ git commit -m "disable auto-scaling"
   ```

3. Push your changes to your application's remote repository using the **`git push`** command.

```
$ git push
```

Now that you have disabled the automatic scaling feature of your application, the next section describes how you can manually scale your application.

6.3.2.2.2. Scaling an Application Manually

After disabling the automatic scaling feature of your application, you can manually control the scaling function of your application. The instructions below describe how you can manually scale your application up or down.

Procedure 6.5. To manually scale an application:

1. Create an SSH connection to your scaled application, as shown in the example below.

```
$ ssh <AppUUID>@<AppName>-<DomainName>.example.com
```

2. Use the **`add-gear`** or **`remove-gear`** command to manually scale your application up or down.
   a. Run the following command to scale up an application:

   ```
   [app-domain.example.com ~]\> add-gear -a <AppName> -u <AppUUID> -n
   <DomainName>
   ```

   b. Run the following command to scale down an application:

   ```
   [app-domain.example.com ~]\> remove-gear -a <AppName> -u <AppUUID> -n
   <DomainName>
   ```

6.3.2.3. Using DIY Cartridges

You can use OpenShift to create applications of no specific type, for build or other testing purposes. The syntax for using this cartridge type is the same as all other cartridge types:

```
$ rhc app create myapp diy-0.1
```

This creates an application that is not publicly available nor does it have anything running. You need to use **`git push`** and the **`.openshift/action_hooks/`** scripts to perform all required operations.

6.3.2.4. Using Arbitrary DNS Names

You can specify meaningful DNS names for your OpenShift applications so that you can use your own DNS entries instead of using the domain generated for you by the system.

For example, for the two applications in the previous section, you could create aliases that better suited your own domain and application purposes. You can use the **`rhc alias add <application> <alias>`** command to create these aliases, as follows:

```
$ rhc alias add racer fast.cars.com
```

Both **`http://racer-automobile.example.com`** and **`http://fast.cars.com`** point to the same DNS and display the same site.

If at any time you need to remove the alias, use the **`rhc alias remove <application> <alias>`** command:

```
$ rhc alias remove racer fast.cars.com
```

6.4. Adding and Managing Cartridges

6.4.1. Management Console

The OpenShift Management Console provides an intuitive interface for managing your application's cartridges.

You can view the cartridges associated with an application by clicking **`My Applications`** in the navigation bar at the top of Management Console then clicking on the application you want to view.

6.4.1.1. Adding Cartridges

The OpenShift Management Console allows you to add cartridges to your applications easily. Note that certain cartridges are only available to be added after a prerequisite cartridge is added. Cartridge descriptions in the Management Console detail these dependencies.

Procedure 6.6. To add a cartridge:

1. Access the OpenShift Management Console and click **`My Applications`** in the navigation bar at the top of the page.
2. Click on the application to which you want to add a cartridge.
3. Click **`Add Cartridge`**.
4. Choose the cartridge to add to your application and click **`Select`**.
5. Click **`Add Cartridge`** to confirm the selection.

6.4.2. Command Line Interface

The OpenShift client tools provide a full range of commands for managing cartridges that you add to your applications.

You can view the cartridges associated with your applications by running the **`rhc apps`** command. To view an individual application's information, use **`rhc app show`** ***`AppName`***. Installed cartridges are listed under the **`Cartridge`** heading.

6.4.2.1. Adding Cartridges

The current list of available cartridges can be viewed by running the **`rhc cartridge list`** command. Note that certain cartridges can only be added after a prerequisite cartridge is added. Cartridge descriptions in the OpenShift website Management Console detail these dependencies.

Note

You can add only MySQL 5.1, MongoDB, PostgreSQL, or Jenkins Client 1.4 cartridges to scaled applications.

Procedure 6.7. To add a cartridge using the CLI:

- Run the following command, replacing ***`AppName`*** and ***`CartType`*** with the name of the application to which you want to add a cartridge and the type of cartridge you want to add:

  ```
  $ rhc cartridge add CartType -a AppName
  ```

6.4.2.2. Managing Cartridges

You can use the following options with the **`rhc cartridge`** command, specifying the application to manage with the **`-a`** ***`AppName`*** argument:

Table 6.4. Application management command argument options

Option	Details
list	Lists supported cartridges.
add	Adds a cartridge.
remove	Removes a cartridge.
stop	Stops a cartridge.
start	Starts a cartridge.
restart	Restarts a cartridge.
status	Returns the current status of a cartridge.
reload	Reloads the configuration of a cartridge.
show	Shows information about a cartridge.
storage	View and manipulate storage on a cartrdige.
scale	Set the scaling range of a cartridge.

6.5. Working With Database Cartridges

OpenShift provides support for several database back ends, including PostgreSQL and MySQL. The procedures for adding these databases to your applications are the same in each case.

MongoDB provides some extended management features; these are discussed in Section 6.5.3, “Adding and Managing a MongoDB Database”.

6.5.1. Adding a MySQL Database

The following example illustrates how to add a MySQL database to the application "racer".

```
$ rhc cartridge add mysql-5.1 -a racer
Password: **********

Adding 'mysql-5.1' to application 'racer'
Success
mysql-5.1
=========
  Properties
  ==========
    Database Name  = racer
    Connection URL = mysql://127.5.32.1:3306/
    Password       = XdyRc3q9YbMF
    Username       = admin
```

6.5.2. Adding a PostgreSQL Database

To use a PostgreSQL database with your application, use the same procedure as described above for adding a MySQL database, but use the PostgreSQL cartridge instead:

```
$ rhc cartridge add postgresql-8.4 -a MyApp
```

You can use **SSH** to connect to your application and verify the integrity of your database. Use the following command to connect to your application:

```
$ ssh UUID@appname-namespace.example.com
```

Use the password provided as output to the **rhc cartridge add** command to connect to the database:

```
PGPASSWORD=password psql -h 127.0.251.129 -U admin -d appname
psql (8.4.9)
Type "help" for help.
appname=#
```

The last line in the output indicates a successful connection to the database. As a further check, you can run a simple query:

```
appname=# select 1 as a, 2 as b, 3 as c;
a | b | c
---+---+---
1 | 2 | 3
(1 row)
```

6.5.3. Adding and Managing a MongoDB Database

6.5.3.1. Overview

OpenShift supports **MongoDB,** a customizable back end database for web applications. OpenShift also supports **RockMongo,** a **MongoDB** administration tool. You can add **RockMongo** to your applications and use it to manage your **MongoDB** instances.

6.5.3.2. Using MongoDB with OpenShift Applications

Use the **rhc cartridge add** command to add a **MongoDB** database instance to your application and to add **RockMongo** to control that database instance. The following procedure demonstrates how to create a new application and how to use **MongoDB** and **RockMongo** with that application.

Procedure 6.8. How to use MongoDB and RockMongo with OpenShift applications

1. Create a new application.

```
$ rhc app create myMongo php-5.3
```

2. Add the **MongoDB** cartridge to your application.

```
$ rhc cartridge add mongodb-2.2 -a myMongo
```

   Take note of the credentials and other information that this command provides.

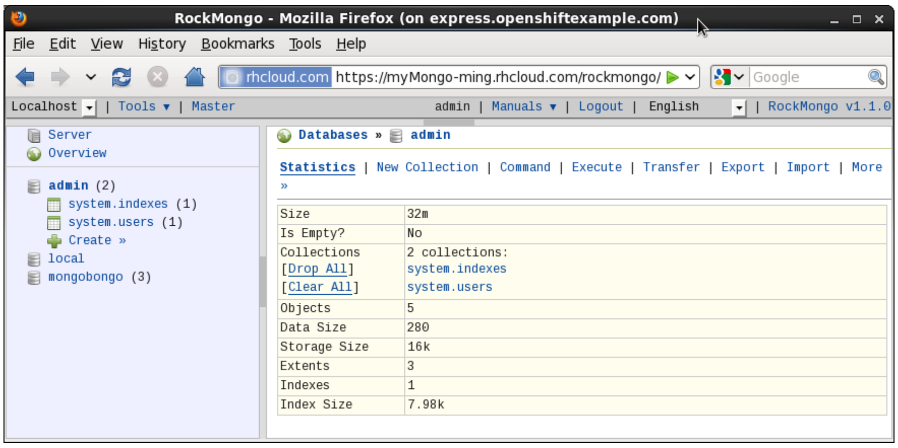
3. Add the **RockMongo** cartridge to your application.

```
$ rhc cartridge add rockmongo-1.1 -a myMongo
```

   Take note of the credentials and other information that this command provides.

6.5.3.3. Managing MongoDB Instances with a Browser

After you have added a **MongoDB** database and the **RockMongo** cartridge to your application, you can navigate to and start exploring your database. Use the URL and RockMongo User and RockMongo Password credentials provided as output from the last step in the Procedure 6.8, "How to use MongoDB and RockMongo with OpenShift applications" procedure. In this example, the URL is https://myMongo-myDomain.example.com/rockmongo/. Below is an example of the **RockMongo** web interface.


RockMongo - Mozilla Firefox (on express.openshiftexample.com)
File Edit View History Bookmarks Tools Help
rhcloud.com https://myMongo-ming.rhcloud.com/rockmongo/
Google
Localhost | Tools ▾ | Master
admin | Manuals ▾ | Logout | English | RockMongo v1.1.0
Server
Overview
admin (2)
system.indexes (1)
system.users (1)
Create »
local
mongobongo (3)
Databases » admin
Statistics | New Collection | Command | Execute | Transfer | Export | Import | More »

Size	32m
Is Empty?	No
Collections [Drop All] [Clear All]	2 collections: system.indexes system.users
Objects	5
Data Size	280
Storage Size	16k
Extents	3
Indexes	1
Index Size	7.98k

Figure 6.3. RockMongo Web Interface

Refer to **RockMongo** documentation for further information on how to use this interface to manage your database.

6.5.3.4. Managing MongoDB Instances in a Shell Environment

After you have added a **MongoDB** database and the **RockMongo** cartridge to your application as described in Procedure 6.8, "How to use MongoDB and RockMongo with OpenShift applications", you can manage your **MongoDB** instance in a shell environment.

Important

Shell access is quite powerful and it is possible to damage an application accidentally. Therefore it is recommended you only use shell access when necessary.

Use the **`rhc apps`** command to display information about all of your available applications. Take note of the UUID and public URL information that this command provides for your **MongoDB** instance.

The example below shows viewing information for the **myMongo** application:

Example 6.1. Viewing Application Information

```
$ rhc apps
Password: **********

myMongo @ http://myMongo-myDomain.example.com/ (uuid:
f74d7e4fffeb4w1c5abe5agr19e3d7f2)
--------------------------------------------------------------------------------
----------
  Created: 1:09 AM
  Gears:   1 (defaults to small)
  Git URL: ssh://f74d7e4fffeb4w1c5abe5agr19e3d7f2@myMongo-
myDomain.example.com/~/git/myMongo.git/
  SSH:     f74d7e4fffeb4w1c5abe5agr19e3d7f2@myMongo-myDomain.example.com

  php-5.3 (PHP 5.3)
  -----------------
    Gears: Located with mongodb-2.2, rockmongo-1.1

  mongodb-2.2 (MongoDB NoSQL Database 2.2)
  ----------------------------------------
    Gears:     Located with php-5.3, rockmongo-1.1
    Connection URL: mongodb://127.2.27.120:27327/
    Database Name:  myMongo
    Password:     z57C6HdU5JnL
    Username:     admin

  rockmongo-1.1 (RockMongo 1.1)
  -----------------------------
    Gears:          Located with php-5.3, mongodb-2.2
    Connection URL: https://myMongo-myDomain.example.com/rockmongo/
```

You will also need the *Root username* and the *Root password* for your **MongoDB** instance that was provided as output from Step 2 in Procedure 6.8, “How to use MongoDB and RockMongo with OpenShift applications”.

When you have the necessary information, use the **ssh** command to open a shell environment for your **MongoDB** instance, substituting the UUID and public URL parameters noted previously, as demonstrated below:

```
$ ssh 0bd9d81bfe0a4def9de47b89fe1d3543@myMongo-myDomain.example.com
```

When you have accessed your OpenShift application in the shell environment, you can type **`help`** at the shell prompt to get a list of the specialized shell commands.

6.5.3.4.1. Opening a Mongo Shell

From the shell prompt, use the following command to open an authenticated Mongo shell to run all Mongo shell commands, substituting the *Root Username* and *Root Password* obtained previously for your **MongoDB** instance:

```
[myMongo-myDomain.example.com~]\> mongo -u admin -p zVumUTBNKbXz
```

The example below demonstrates an open Mongo shell, and use of the **`show users`** Mongo shell command:

Example 6.2. Running Mongo Shell Commands

```
MongoDB shell version: 2.2.0
connecting to: 127.0.250.129:27017/admin
Welcome to the MongoDB shell.
> show users
{
 "_id" : ObjectId("4ee55d39078e94193206e157"),
 "user" : "admin",
 "readOnly" : false,
 "pwd" : "aba43436961fbc6145261a12ed94b8f7"
}
```

Refer to the **MongoDB** website and documentation for more information on Mongo shell commands.

6.6. Deploying Applications

To deploy your application to the cloud, you need to make any required changes to your application code base, commit those changes to your local repository, and then update the remote repository. Application files are stored in the local git repository that was cloned as part of the application creation process.

6.6.1. Preparing Your Application for Deployment

The **`rhc app create`** command creates a starting point, or template, for you to create and develop your own applications. To synchronize your application files with the remote cloud repository, you need to commit all of your files to the appropriate directories in the local git repository, and then push them to the remote repository. For example, you may be developing your own PHP application and need to add new files and directories to **`$_ENV['OPENSHIFT_APP_NAME']/php/`** or other directories.

Procedure 6.9. To prepare your application for deployment using the command line:

1. Use the following command to add each new file and directory to the git index:

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

2. Use the following command to commit your application to your local repository:

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

6.6.2. Deploying Your Application to the Cloud

After you have added your application files to the local repository, you need to push them to the remote repository. OpenShift will automatically stop, build, and restart your application with the committed changes.

Procedure 6.10. To deploy your application to the cloud using the command line:

- Use the following command to deploy your application to the remote repository:

```
$ git push
```

6.6.3. Hot Deploying Applications

When you run the **`git push`** command to upload code modifications, OpenShift stops, builds, deploys and restarts your application. This entire process takes time to complete and is unnecessary for many types of code changes. Hot deploying your application allows your changes to take effect without restarting the application cartridge, thus increasing deployment speed and minimizing application downtime.

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

Table 6.5. 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
Node.js	Yes
Zend Server	Yes
Jenkins	No
HAProxy	No
DIY	No

6.6.3.1. Hot Deployment Build Details

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

When you hot deploy JBoss AS, JBoss EAP, Tomcat 6, and Tomcat 7 applications, 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, Zend Server, Perl, Python, and Node.js

When you hot deploy PHP, Zend Server, Perl, Python, and Node.js applications, 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 for both Jenkins and non-Jenkins enabled applications.

For Jenkins enabled applications, the build is performed on a Jenkins slave instance and then synced to the gear(s) where the application server is running.

Ruby

When you hot deploy a Ruby application, the Passenger **restart.txt** file is touched, causing the application server to serve the new code without the need for a full server restart. For further details on this process, view the Passenger Documentation.

6.6.3.2. Enabling and Disabling Hot Deployment

Procedure 6.11. To enable hot deployment:

- Open a terminal and run the command applicable to your operating system from your application's root directory to create the **hot_deploy** marker file:

  Windows

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

  Linux and Mac

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

Procedure 6.12. To disable hot deployment:

- Hot deployment is disabled by deleting the **hot_deploy** marker file. To delete this file run the command applicable to your operating system from your application's root directory:

  Windows

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

  Linux and Mac

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

6.6.4. Deploying JBoss Applications

6.6.4.1. Deploying on Java 6 or Java 7

You can run JBoss AS 7 and JBoss EAP 6.0 applications on either Java 6 or Java 7. The Java version is specified by a **java7** marker file in your application's **markers** directory. By default, new JBoss applications have the **java7** marker enabled. Applications without the **java7** marker are deployed using Java 6.

Procedure 6.13. To use Java 7:

- Open a terminal and run the commands applicable to your operating system from your application's root directory to create the **java7** marker file and update the remote repository:

  Windows

```
C:\app_directory> copy NUL > .openshift\markers\java7
C:\app_directory> git add .openshift\markers\java7
C:\app_directory> git commit -a -m "Add Java 7 marker"
C:\app_directory> git push
```

Linux and Mac

```
[user@user app_directory]$ touch .openshift/markers/java7
[user@user app_directory]$ git add .openshift/markers/java7
[user@user app_directory]$ git commit -a -m "Add Java 7 marker"
[user@user app_directory]$ git push
```

Procedure 6.14. To use Java 6:

- Java 6 is enabled by deleting the **`java7`** marker file. Run the commands applicable to your operating system from your application's root directory to delete the **`java7`** marker file and update the remote repository:

  #### Windows

  ```
  C:\app_directory> del .openshift\markers\java7
  C:\app_directory> git commit -a -m "Remove Java 7 marker"
  C:\app_directory> git push
  ```

  #### Linux and Mac

  ```
  [user@user app_directory]$ rm .openshift/markers/java7
  [user@user app_directory]$ git commit -a -m "Remove Java 7 marker"
  [user@user app_directory]$ git push
  ```

6.6.4.2. Automatic Deployment

To automatically deploy your applications to the OpenShift server runtime, you can place your deployment content, for example, .war, .ear, .jar, and .sar files, in the JBoss Application Server (AS) or JBoss Enterprise Application Platform (EAP) **`deployments/`** directory.

The file system deployment scanner in JBoss AS 7 and JBoss EAP 6.0 relies on a system of marker files. You can add or remove various marker files and the scanner deploys, withdraws, or redeploys content based on the type of marker file.

These marker files always have the same name as the deployment content to which they relate, but with an additional file suffix to indicate the type of action to perform. For example, the marker file to indicate that the **`example.war`** file should be deployed is named **`example.war.dodeploy`**.

Option 1: Uploading content in a Maven src structure

You can upload your content in a Maven src structure similar to the example **`ROOT.war`** file. When you perform a **`git push`** the application is built and deployed. This is the typical deployment option, and requires that your **`pom.xml`** be stored at the root of your repository, and that you have a ***`maven-war-plugin`*** similar to the one in the example file to move the output from the build to the **`deployments/`** directory.

By default the ***`warName`*** is ROOT within the **`pom.xml`** file. This will render the webapp contents at http://***app_name-namespace***.example.com. If you change the warName in **`pom.xml`** to **`app_name`**, then your base URL would become http://***app_name-namespace***.example.com/app_name.

Important

If you are building locally, add any output **`.war`** and **`.ear`** files in the **`deployments/`** directory from the build to your **`.gitignore`** file.

Option 2: Uploading prebuilt content

You can use **`git push`** to push prebuilt **`.war`** files (with the corresponding **`.dodeploy`** file for exploded **`.war`** files) into the **`deployments/`** directory.

Important

If you use this method using the default repository, first run **`git rm -r src/ pom.xml`** from the root directory of your repository. If the **`pom.xml`** file still exists, the build will run again and replace any prebuilt content.

6.6.4.3. Types of Marker Files

Different marker file suffixes have different meanings. The relevant marker file types are as follows:

Table 6.6. Sample Table

Marker File Type	Description
.dodeploy	Placed by the user to indicate that the given content should be deployed into the runtime (or redeployed if already deployed in the runtime.)
.deploying	Placed by the deployment scanner service to indicate that it has detected a **`.dodeploy`** file and is in the process of deploying the content. This marker file is deleted when the deployment process completes.
.deployed	Placed by the deployment scanner service to indicate that the given content has been deployed to the runtime. If you delete this file, the content will be withdrawn.
.faileddeploy	Placed by the deployment scanner service to indicate that the given content failed to deploy to the runtime. The content of this file will include some information about the cause of the failure.
.undeploying	Placed by the deployment scanner service to indicate that it has detected a **`.deployed`** file has been deleted and the content is being withdrawn. This marker file is deleted when the withdrawal process completes.
.undeployed	Placed by the deployment scanner service to indicate that the given content has been withdrawn from the runtime. If you delete this file, it has no impact.

6.6.4.4. Example JBoss Application Deployment Workflows

The following sections describe the basic workflows involved in deploying, withdrawing, and redeploying JBoss prebuilt applications to OpenShift.

To add new, zipped content and deploy it, run the following command:

```
$ cp target/example.war deployments/
```

To add new, unzipped content and deploy it, run the following commands:

```
$ cp -r target/example.war/ deployments/
$ touch deployments/example.war.dodeploy
```

To withdraw deployed content, run the following command:

```
$ git rm deployments/example.war.dodeploy deployments/example.war
```

To replace currently deployed, zipped content with a new version and deploy it, run the following command:

```
$ cp target/example.war deployments/
```

To replace currently deployed, unzipped content with a new version and deploy it, run the following commands:

```
$ git rm -rf deployments/example.war/
$ cp -r target/example.war/ deployments/
$ touch deployments/example.war.dodeploy
```

6.7. Jenkins Online Build System

6.7.1. Introduction to Jenkins

Jenkins integrates with other OpenShift applications. To start building with Jenkins, you need to add the Jenkins Client to an existing application. From then on, running a **`git push`** command initiates a build process inside a Jenkins builder instead of inside your normal application compute space. For custom applications, or applications that have no upstream repositories, the build process can be initiated directly from the Jenkins web interface, without having to run the **`git push`** command.

Using the embedded Jenkins Client build system provides numerous benefits:

- Archived build information
- No application downtime during the build process
- Failed builds do not get deployed (leaving the previous working version in place)
- Jenkins builders have additional resources, for example memory and storage
- A large community of Jenkins plug-ins

6.7.2. Configuring Jenkins

This section describes how to set up Jenkins to rebuild your application automatically when you make changes to your local repository and then push those changes to the remote repository.

Note

The following procedures assume that you already have a valid user account for OpenShift.

6.7.2.1. Resource Requirements for Jenkins

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

6.7.3. Creating Jenkins-enabled Applications

You can enable Jenkins with new applications, either scaled or non-scaled. Use the **`rhc app create`** command to create a scaled or non-scaled application with an associated Jenkins application, and to add the Jenkins client to your application.

6.7.3.1. Creating a Scaled Jenkins-enabled Application

Use the **`rhc app create`** command with the ***`-s`*** (or ***`--scaling`***) option to create a scaled application, and the ***`--enable-jenkins`*** option to add the Jenkins client to your application, as shown below:

```
$ rhc app create App01 php-5.3 --enable-jenkins -s
```

Important

Take note of the login credentials that this command outputs to the screen. You will need these credentials to log in to the Jenkins home page.

6.7.3.2. Creating a Non-scaled Jenkins-enabled application

Use the **`rhc app create`** command as shown below to create a non-scaled application with an associated Jenkins application, and to add the Jenkins client to your application.

Ensure that you take note of the login credentials that this command outputs to the screen. You will need these credentials to log in to the Jenkins home page.

```
$ rhc app create App01 php-5.3 --enable-jenkins
```

6.7.3.3. Confirming Your Jenkins-enabled Application

You can now run **`rhc apps`** to confirm that your Jenkins-enabled application has been created. You can also navigate to the public URL returned by the **`rhc app create`** command to view your Jenkins home page.

Note

As part of the process of adding the Jenkins application to your domain, the Jenkins repository is downloaded to your local machine. This repository is not required for normal operations, and it is safe to delete this local copy.

6.7.4. Building Applications with Jenkins

Building 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. No further interaction is required by the user. The existing application is not affected until a new, successful build has been created. Should the build fail, the existing application continues to run, although a failure in the deployment process (deploy → start → post_deploy) may leave the application partially deployed or inaccessible.

The actual build and deployment process that Jenkins executes involves numerous steps, as described below.

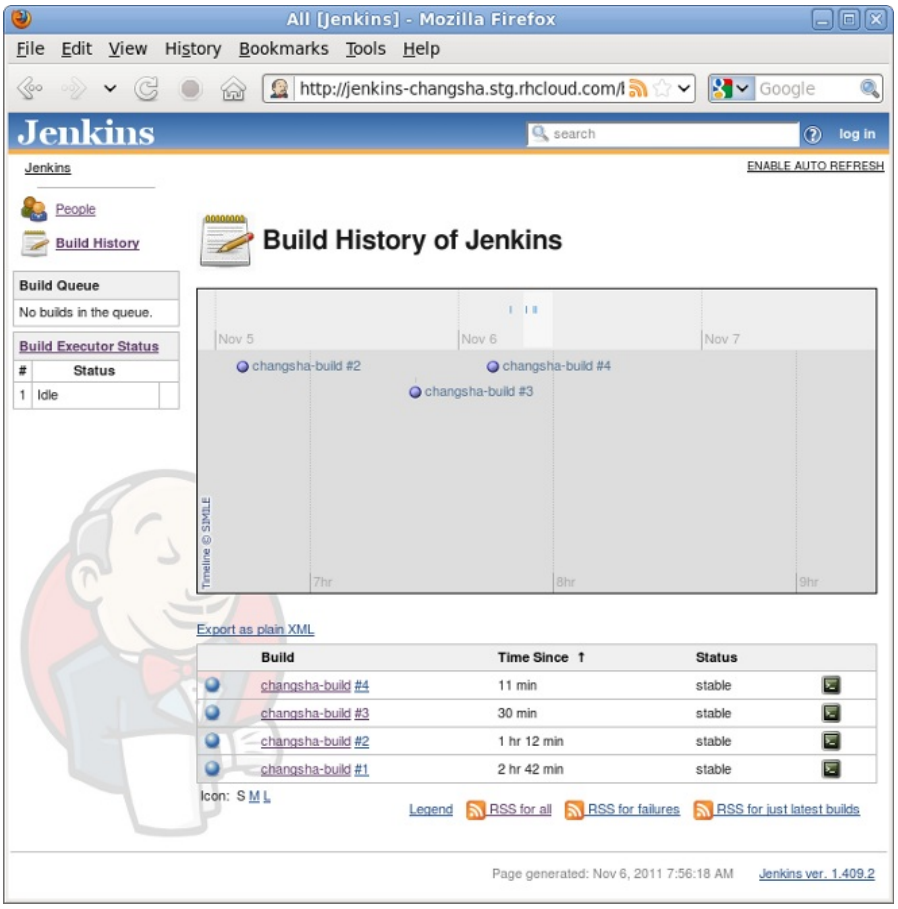
1. The user issues a **`git push`** command, and Jenkins is notified that a new push is ready.
2. A dedicated Jenkins slave (a *builder*) is created. You can use the **`rhc apps`** command to display this slave information. The application name is the same as the originating application, but with a "bldr" suffix.

   Important

   The first 28 characters of the application name must be unique or builders will be shared across applications. This can lead to build issues.

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.

You can monitor the build job 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. The following is an example build history of an application built using the embedded Jenkins Client.



Build History of Jenkins

Build Queue
No builds in the queue.

Build Executor Status

#	Status
1	Idle

changsha-build #2
changsha-build #4
changsha-build #3

	Build	Time Since ↑	Status	
●	changsha-build #4	11 min	stable	
●	changsha-build #3	30 min	stable	
●	changsha-build #2	1 hr 12 min	stable	
●	changsha-build #1	2 hr 42 min	stable	

Page generated: Nov 6, 2011 7:56:18 AM Jenkins ver. 1.409.2

Figure 6.4. Jenkins Build History Page

Logging for Jenkins-level errors (for example, DNS timeouts, builder configuration, etc.) is available in the Jenkins logs from the command line using the **`rhc tail`** command. Replace ***`jenkins`*** with your Jenkins application name in the following, if different:

```
$ rhc tail jenkins
```

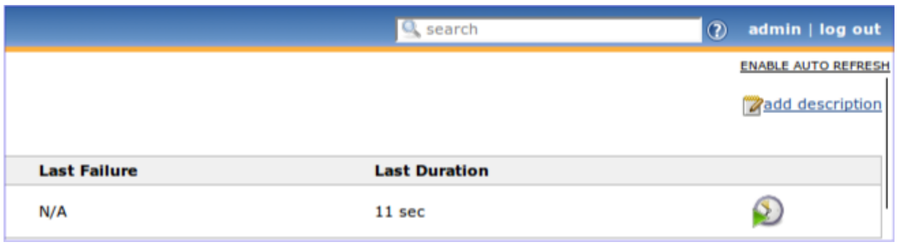
Logging for application-level errors (for example, compilation failures, test failures, etc.) is available via the Jenkins web interface under the corresponding build history, while logging for deployment-level errors is available in the application's logs from the command line:

```
$ rhc tail App01
```

6.7.4.1. Building Custom Applications

You can also build custom applications, or applications that have no upstream repositories, directly from the Jenkins web interface instead of using the **`git push`** command.

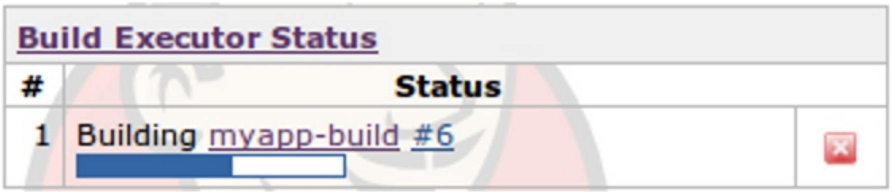
To build your application from the Jenkins web interface, click on the icon for the application you wish to build, located on the right side of the interface.



Last Failure	Last Duration
N/A	11 sec

Figure 6.5. Build from Jenkins Interface

The status of the build process can be viewed in the web interface under the section labeled **`Build Executor Status`**.



Build Executor Status

#	Status
1	Building myapp-build #6

Figure 6.6. Build Process Status

6.7.5. Building Applications Without Jenkins

Building without Jenkins uses your application space as part of the build and test process. This requires the currently running application be shut down in order to use its memory. This means that your application will be unavailable for the duration of the build and all associated tasks. These tasks are described below:

1. Pre-receive - This occurs when you run a **`git push`** command, but before the push is fully committed.
2. Build - This builds your application, downloads required dependencies, executes the **`.openshift/action_hooks/build`** script and prepares everything for deployment.
3. 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.
4. Post-deploy - This step allows for interaction with the running application, including running the **`.openshift/action_hooks/post_deploy`** script. This step occurs immediately after the application is restarted.

6.8. Deleting an Application

If you no longer need a particular application, you can choose to delete it.

6.8.1. Management Console

Warning

Application removal is irreversible. All remote data associated with the application will be removed.

Procedure 6.15. To delete an application:

1. Access the Management Console and click **`My Applications`** in the navigation bar at the top of the page.
2. Click on the application you want to delete.
3. Click **`Delete this application`**.
4. You are asked to confirm the request. Click **`Delete`** to confirm.

This process deletes your remote application data. If you want to delete application data stored on your local machine, you must do so manually.

6.8.2. Command Line Interface

Warning

Application removal is irreversible. All remote data associated with the application will be removed.

Procedure 6.16. To delete an application:

1. Run the following command to delete all the remote data for your application:

   ```
   $ rhc app delete AppName
   ```

2. At the prompt, type **`yes`** to confirm the deletion.

This process deletes your remote application data. If you want to delete application data stored on your local machine, you must do so manually.

6.8.3. Deleting Local Application Data

Warning

The following procedure deletes the selected directory and all the files it contains. Ensure you enter the correct directory and that you no longer need the files it contains before running this command.

Procedure 6.17. To delete local application data

- Open a terminal and use the following command to delete the application data stored on your local machine:

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

6.9. Managing Application Disk Space

As you develop your applications and push changes to the git repository, you will slowly reduce the amount of disk space that is available to run your applications. This is because git stores all repository information, whether it is still required or not. Other aspects of developing and running your applications also result in wasted disk space, such as old log files and unused application libraries.

OpenShift provides tools to help you optimize your disk space to help achieve the best possible performance of your applications.

6.9.1. The Disk Space Cleanup Tool

OpenShift provides a *Disk Space Cleanup Tool* to help you manage your application disk space. This tool is part of the **`rhc app`** command suite, and performs the following functions:

- Runs the **`git gc`** command on your application's git repository on the server
- Clears the application's **`/tmp`** and log file directories. These are specified by the application's **`OPENSHIFT_`*`<cartridge>`*`_LOG_DIR`** and **`OPENSHIFT_TMP_DIR`** environment variables.

  Important

  OpenShift does not automatically back up or rotate log files. The Disk Space Cleanup Tool runs the **`rm -rf`** command to clear the contents of these directories. If you want to save their contents, you should use the **`rhc snapshot save`** command first to create a snapshot of your system.

- Clears unused application libraries. This means that any library files previously installed by a **`git push`** command are removed.

The Disk Space Cleanup Tool uses the following syntax:

```
$ rhc app tidy <ApplicationName>
```

Note

The rhc client tools use the login specified in the **`~/.openshift/express.conf`** file by default. If no default login is configured, or if you want to use a different login, include the ***`-l`*** parameter as a command line argument.

Chapter 7. Managing Applications

7.1. Application Management Commands

Use the **`rhc app`** command to manage your applications, view application status, and start, stop, restart, reload and delete applications. Refer to **`rhc app --help`** for more information and a full list of the available options. This command uses the following syntax:

```
$ rhc app <action> ApplicationName [Optional Arguments]
```

The following table describes the application management command options that you can use to manage an existing application in the cloud.

Table 7.1. Application management command argument options

Option	Details
start	Start an application.
stop	Stop an application.
force-stop	Stops all application processes.
restart	Restart an application.
reload	Reload an application.
status	Display the current status of an application.
show	Show information about an application.
tidy	Clean out the application's logs and tmp directories and tidy up the git repo on the server.
create	Create an application and add it to a namespace.
git-clone	Clone and configure an application's repository locally.
delete	Remove an application.

7.2. Managing Applications in a Secure Shell Environment

You can manage your OpenShift applications in a shell environment to perform specialized operations and general debugging. The shell access provides specialized tools for managing your applications.

Important

Shell access is quite powerful and it is possible to accidentally damage an application. Therefore it is recommended you only use shell access when necessary.

7.2.1. Linux and Mac

Before you can access your application in a shell environment, ensure that you have completed the following:

- Create a domain as described in the *Creating a Namespace* section. The examples in this section assume you have created the **`automobile`** domain.
- Create an application as described in the *Creating an Application* section. The examples in this section assume you have created the "racer" application.

After you have created your domain and your application, use the **`rhc apps`** command to display information about all of the current applications:

Example 7.1. Viewing Application Information

```
$ rhc apps
Password: **********

racer @ http://racer-automobile.example.com/ (uuid:
926056f8845b4e388b37f6735c89d0eb)
--------------------------------------------------------------------------------
--------
  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

  php-5.3 (PHP 5.3)
  -----------------
    Gears: 1 small
```

Take note of the UUID and public URL information for the application you wish to manage in the shell environment. Use the **ssh** command to open a shell environment for your application, substituting the UUID and public URL parameters noted previously.

The example below demonstrates opening a shell environment to the "racer" application:

Example 7.2. Opening a Shell Environment for an Application Node

```
$ ssh 4j5k656a3f464d84b565d9c1d9b5283b@racer-automobile.example.com
Warning: Permanently added 'racer-automobile.example.com,174.129.151.26' (RSA)
to the list of known hosts.

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 ~]\>
```

Note

The shell environment to access your OpenShift applications is protected and restricted with Security-Enhanced Linux (SELinux) policies.

When you have accessed your OpenShift application in the shell environment, you can type **`help`** at the shell prompt to get a list of the specialized shell commands. As mentioned previously, you can also use general Linux commands to perform routine operations in the shell environment.

7.2.2. Windows

Managing OpenShift applications in a Secure Shell (SSH) environment on Windows requires the PuTTY SSH and Telnet client, and PuTTYgen SSH key generator. This section provides instructions on how you can use PuTTY to establish an SSH connection to your OpenShift applications. The instructions are provided in four easy steps, as outlined below:

- Step 1: Download PuTTY and PuTTYgen for Windows
- Step 2: Convert OpenSSH keys to PuTTY format
- Step 3: Locate application username and hostname
- Step 4: Establish SSH connection using PuTTY

Important

Secure Shell (SSH) access is quite powerful, and it is possible to accidentally damage an application. Therefore, it is recommended you only use SSH access when necessary.

Note

The screenshots shown in the instructions below were taken on Windows XP. Where necessary, the differences among other Windows operating systems have been noted.

It is assumed that the application you want to establish an SSH connection to has already been created. In the examples used in this section, an SSH connection will be established to the application named *testapp*.

Step 1: Download PuTTY and PuTTYgen

Go to *http://www.chiark.greenend.org.uk*. Download the executable files of the latest version of PuTTY and PuTTYgen to your desired directory. Alternatively, you can also download the PuTTY installer file that will install all required PuTTY packages on your computer.

Step 2: Convert OpenSSH Keys to PuTTY Format

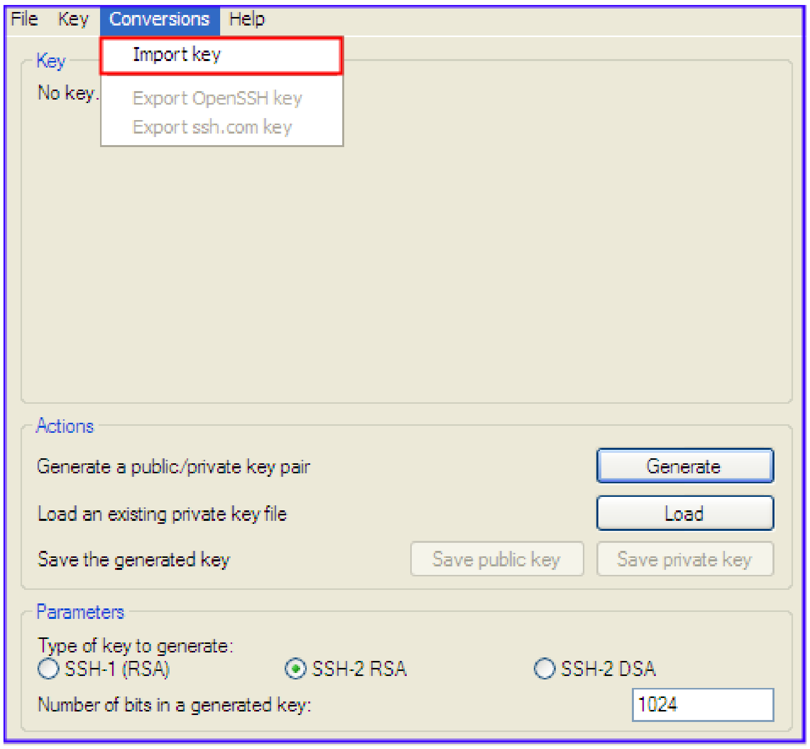
By default, the pair of SSH keys generated when installing client tools on Windows are in OpenSSH format. The pair consists of a private key, named **`id_rsa`**, and a public key, named **`id_rsa.pub`**. To establish an SSH connection to your application using PuTTY, the private SSH key must be converted to PuTTY format. The pair of SSH keys generated during initial configuration of OpenShift client tools are stored in the following folders:

- **`c:\Documents and Settings\user\.ssh`** for Windows XP

- **c:\Users\user\.ssh** for Windows 7

Follow the instructions below to convert your private OpenSSH key to PuTTY format.

1. Double-click **puttygen.exe** to launch PuTTYgen. If necessary, click **Run** in the *Security Warning* window. If you installed the PuTTY software with the installer file, click Start, point to All Programs, point to PuTTY, and then click **PuTTYgen**.
2. Click **Conversions**, then click **Import key** to select your private OpenSSH key, as shown in the figure below.



No key.

Actions

Generate a public/private key pair

Load an existing private key file

Save the generated key

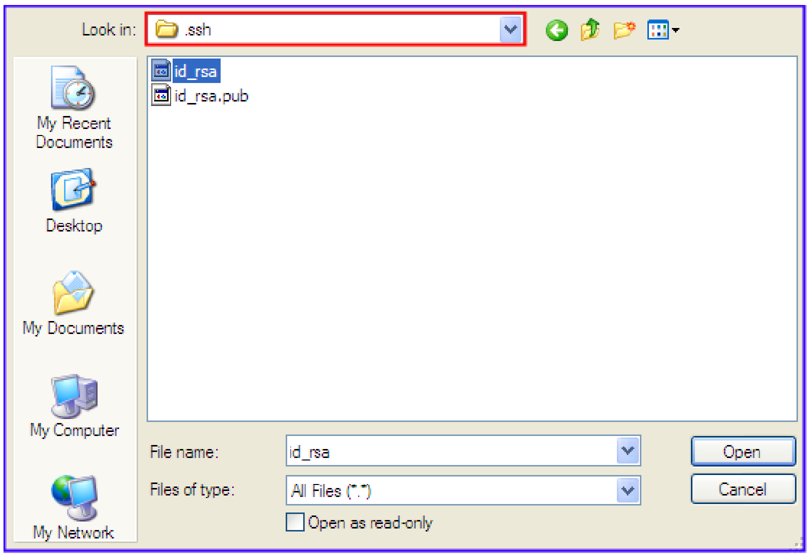
Parameters

Type of key to generate:
SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

Figure 7.1. Import SSH Key

3. Navigate to the **\user\.ssh** folder, and select the **id_rsa** key file to import, as shown in the figure below.



Look in: .ssh

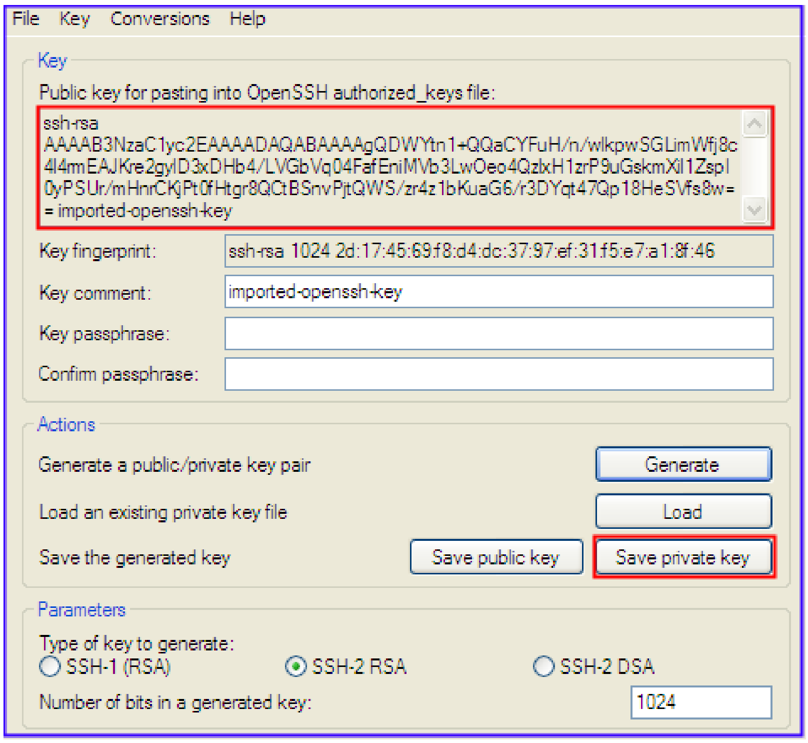
File name: id_rsa

Files of type: All Files (*.*)

Open as read-only

Figure 7.2. Select SSH Key to Import

4. Your private SSH key should now be imported to PuTTYgen, as shown in the figure below. Click **`Save private key`**, and save the **`id_rsa.ppk`** file in the **`\user\.ssh`** folder where the original keys are stored.



Public key for pasting into OpenSSH authorized_keys file:

```
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAAAgQDWYtn1+QQaCYFuH/n/wlkpwSGLimWfj8c
4l4mEAJKre2gylD3xDHb4/LVGbVq04FafEniMVb3LwOeo4QzlxH1zrP9uGskmXil1Zspl
0yPSUr/mHnrCKjPt0fHtgr8QCtBSnvPjtQWS/zr4z1bKuaG6/r3DYqt47Qp18HeSVfs8w=
= imported-openssh-key
```

Key fingerprint: ssh-rsa 1024 2d:17:45:69:f8:d4:dc:37:97:ef:31:f5:e7:a1:8f:46

Key comment: imported-openssh-key

Key passphrase:

Confirm passphrase:

Generate a public/private key pair

Load an existing private key file

Save the generated key

Type of key to generate: SSH-1 (RSA) [✓ SSH-2 RSA] SSH-2 DSA

Number of bits in a generated key: 1024

Figure 7.3. Save Key

5. Navigate to the **`\user\.ssh`** folder and verify that you now have three SSH key files. The **`id_rsa.ppk`** key will be used to establish an SSH connection to your OpenShift application. You can close PuTTYgen.

Step 3: Locate Application Username and Hostname

Each application created on OpenShift gets a unique UUID and gear name that is used to clone the remote Git repository to your local repository. You can use the **`rhc apps`** command or the OpenShift Management Console to get this information for the application to which you wish to establish an SSH connection. Both examples are shown below.

Use the **`rhc apps`** command and note the UUID and gear name listed in the *SSH* field.

```
$ rhc apps
Password: *******

testapp @ http://testapp-doctesting.rhcloud.com/ (uuid:
aa8d89ed311741e7b84d4edb82b11e0d)
--------------------------------------------------------------------------------
-----
  Created: Dec 19 10:20 PM
  Gears:   1 (defaults to small)
  Git URL: ssh://aa8d89ed311741e7b84d4edb82b11e0d@testapp-
docteseting.rhcloud.com/~/git/testapp.git/
  SSH:     aa8d89ed311741e7b84d4edb82b11e0d@testapp-doctesting.rhcloud.com

  php-5.3 (PHP 5.3)
  -----------------
    Gears: 1 small
```

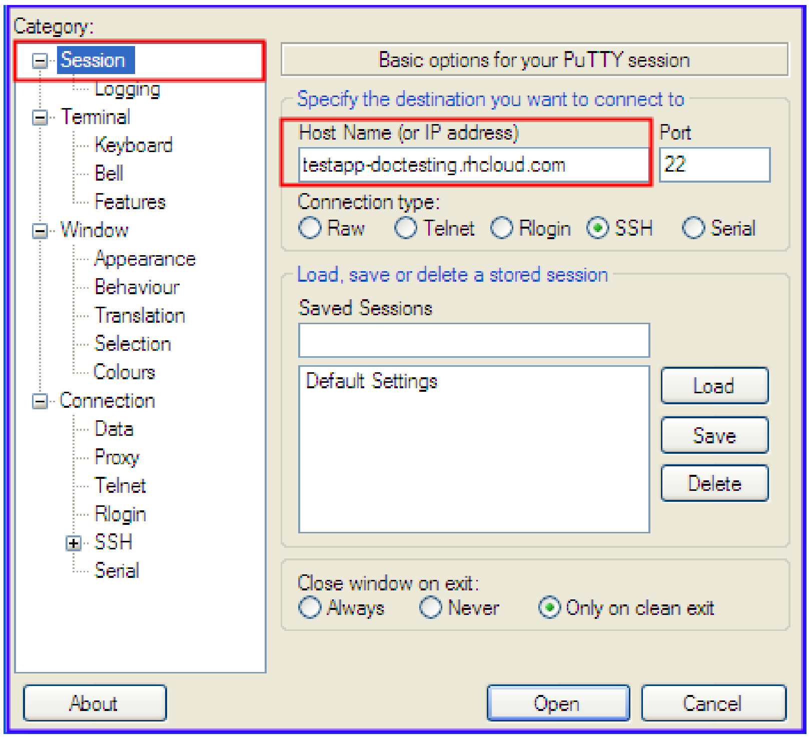
Alternatively, you can use the OpenShift Management Console to get the required information for your application. From the OpenShift Management Console, click on the **My Applications** tab, then click the name of the application you wish to access. The SSH parameters can be found by clicking and expanding **WANT TO LOG IN TO YOUR APPLICATION?** on this page; note the UUID and gear name used.

In the next step you will use the UUID as the username and the gear name as the hostname to configure PuTTY, and establish an SSH connection to your application. To avoid errors, it is recommended you cut and paste this information into PuTTY.

Step 4: Establish SSH Connection Using PuTTY

Now that you have the necessary information, you are ready to configure PuTTY to establish an SSH connection to your application. Follow the instructions below to configure PuTTY using the information from the previous step.

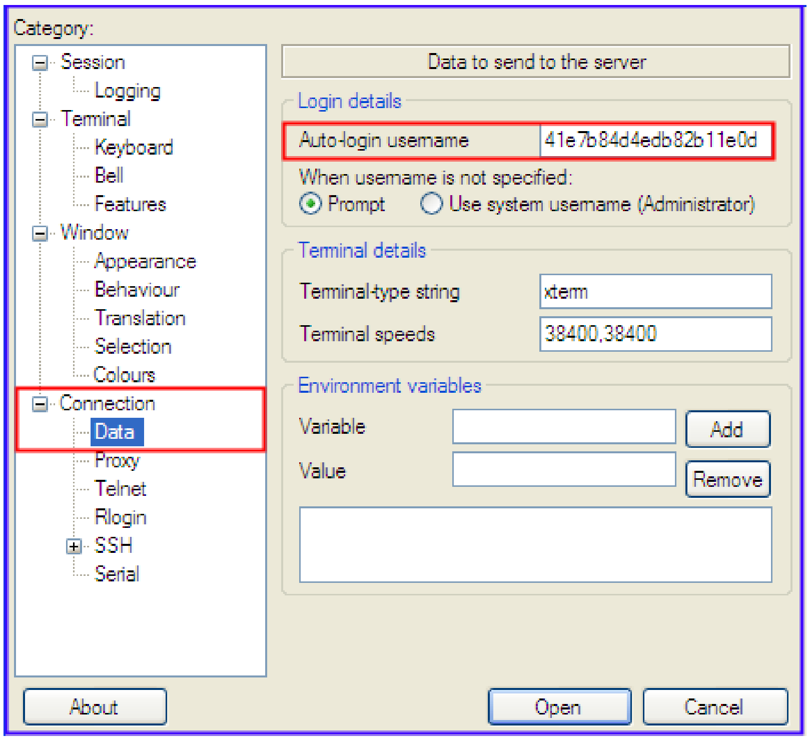
1. Double-click **putty.exe** to launch the PuTTY SSH and Telnet client. If necessary, click **Run** in the *Security Warning* window. If you installed the PuTTY software with the installer file, click Start, point to All Programs, point to PuTTY, and then click **PuTTY**.
2. In the left window pane under *Category*, click **Session**. Now copy and paste the gear name of your application in the *Host Name* text box, as highlighted in the figure below.



Host Name (or IP address): testapp-doctesting.rhcloud.com
Port: 22
Connection type: Raw Telnet Rlogin [✓ SSH] Serial
Saved Sessions:
Default Settings
Close window on exit: Always Never [✓ Only on clean exit]

Figure 7.4. Enter Hostname in PuTTY

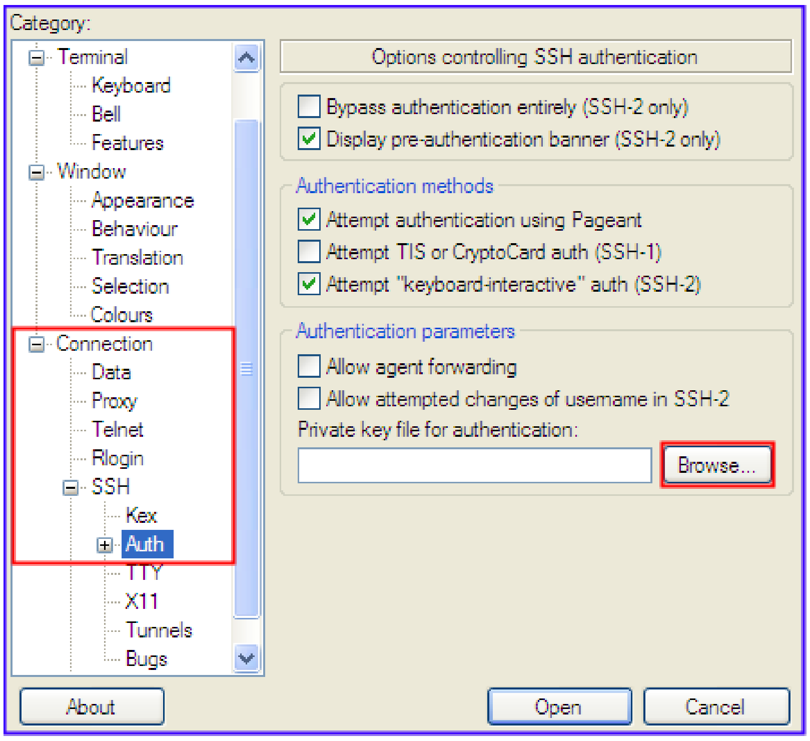
3. In the expanded list of options under **`Connection`**, click **`Data`**. Now copy and paste the UUID of your application in the *Auto-login username* text box, as highlighted in the figure below. Because the UUID is quite long, it may not be fully visible.



Auto-login username: 41e7b84d4edb82b11e0d
When username is not specified: [✓ Prompt] Use system username (Administrator)
Terminal-type string: xterm
Terminal speeds: 38400,38400

Figure 7.5. Enter UUID in PuTTY

4. In the expanded list of options under **`Connection|SSH`**, click **`Auth`**. Then click **`Browse`** to locate the **`id_rsa.ppk`** file you created earlier, as shown in the figure below.



Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

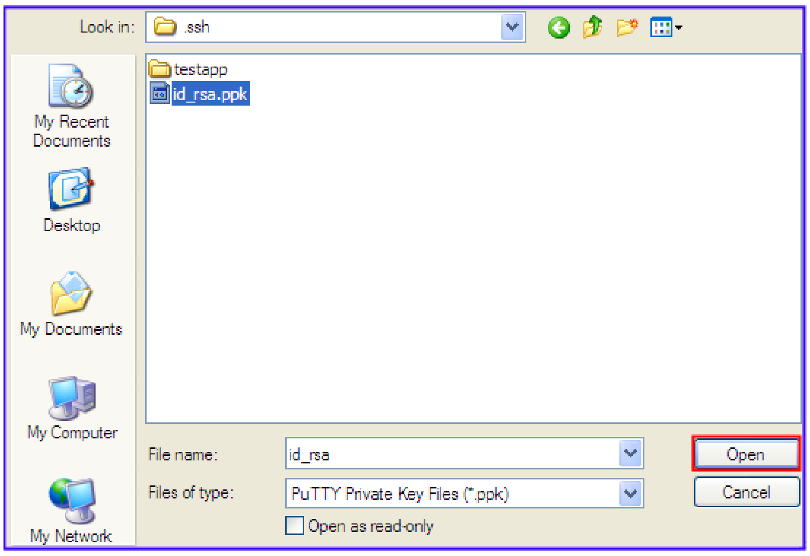
Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication:

Figure 7.6. Find SSH Key

When you have browsed to the **`id_rsa.ppk`** file, click **`Open`**, as shown in the figure below.



Look in: .ssh

testapp
id_rsa.ppk

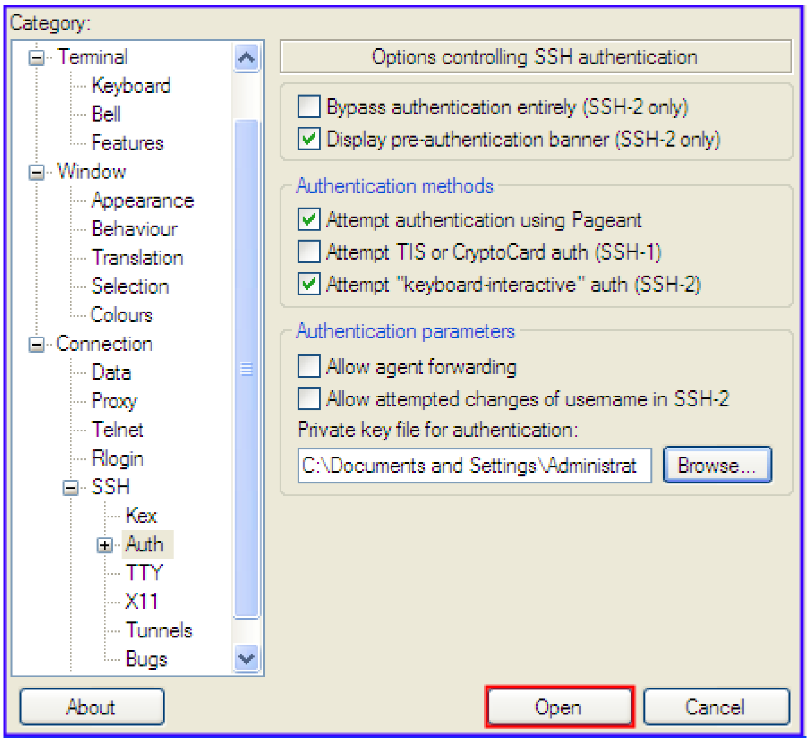
File name: id_rsa

Files of type: PuTTY Private Key Files (*.ppk)

- [ ] Open as read-only

Figure 7.7. Select SSH Key

5. With your **`id_rsa.ppk`** key selected, click **`Open`**, as shown in the figure below. This will open an SSH connection to your application gear.


Category:

Options controlling SSH authentication

- [ ] Bypass authentication entirely (SSH-2 only)
- [✓] Display pre-authentication banner (SSH-2 only)

Authentication methods

- [✓] Attempt authentication using Pageant
- [ ] Attempt TIS or CryptoCard auth (SSH-1)
- [✓] Attempt "keyboard-interactive" auth (SSH-2)

Authentication parameters

- [ ] Allow agent forwarding
- [ ] Allow attempted changes of username in SSH-2

Private key file for authentication: C:\Documents and Settings\Administrat

Figure 7.8. Connect to Application Gear

If a security warning is displayed, click **Yes** to accept it. An SSH terminal window will open, similar to the example shown below.

```
Using username "aa8d89ed311741e7b84d4edb82b11e0d".
Authenticating with public key "imported-openssh-key"

    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, delete your application with 'rhc app delete'
    and recreate it
    !!! IMPORTANT !!! IMPORTANT !!! IMPORTANT !!!

    Type "help" for more info.

[testapp-doctesting.example.com ~]\>
```

One of the useful features of secure shell access is the ability to view application logs. You can find the application log files in the ***App_Name*/logs** directory. The example below shows how to view the log file for the application named *testapp*.

```
[testapp-doctesting.example.com ~]\> cd testapp/logs
[testapp-doctesting.example.com ~]\> ls
[testapp-doctesting.example.com ~]\> view error_log-20120618-000000-EST
```

Note

The Secure Shell environment to access your OpenShift applications is protected and restricted with Security-Enhanced Linux (SELinux) policies.

With an SSH connection established to your OpenShift application, you can type **`help`** at the shell prompt to get a list of specialized shell commands. You can also use general Linux commands to perform routine operations in the shell environment.

7.3. Environment Variables

Environment variables are named values that change the way running processes behave in computing environments. They can be used to store a variety of different values: application names, commonly accessed directory names, usernames, passwords, hostnames, and IP addresses.

Depending on what cartridges you have enabled for your application, the number of variables you have available varies. Also, values may be slightly different for similar types of cartridges, for example if you have a PostgreSQL database as opposed to a MySQL databse.

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

1. Add an **`export`** statement to the **`<appname>/.openshift/action_hooks/build`** file, then run **`git push`**. The variables are listed in the git output and start with **`remote: declare -x`**.
2. Use SSH to gain shell access to your application and run the **`env`** command at the shell prompt.

7.3.1. Informational Environment Variables

These variables provide information about your application.

Table 7.2. Informational Environment Variables

Environment Variable Name	Example	Purpose
OPENSHIFT_APP_DNS	appname-namespace.example.com	The application's fully-qualified domain name
OPENSHIFT_APP_NAME	appname	The application's name
OPENSHIFT_APP_UUID	0123456789abcdef0123456789abcdef	The UUID of the application (32 hex characters)
OPENSHIFT_INTERNAL_IP	127.0.250.1	The IP address the application listens on
OPENSHIFT_INTERNAL_PORT	8080	The port the application receives requests from

7.3.2. Directory Environment Variables

These variables return the directories where your application resides. Note that the only directory that is guaranteed never to be deleted is **`OPENSHIFT_DATA_DIR`**. Except for **`OPENSHIFT_TMP_DIR`**, all of these directories are relative to **`$OPENSHIFT_HOMEDIR`**.

Table 7.3. Directory Environment Variables

Environment Variable Name	Example	Purpose
OPENSHIFT_DATA_DIR	app-root/data/	A persistent data directory
OPENSHIFT_<cartridge>_LOG_DIR	<cartridge>/logs/	Where cartridge-specific logs are stored
OPENSHIFT_<database>_DB_LOG_DIR	<database>/logs/	Where database-specific logs are stored
OPENSHIFT_REPO_DIR	app-root/runtime/repo/	Repository containing the currently deployed version of the application
OPENSHIFT_TMP_DIR	/tmp/	A temporary directory you can use; SELinux protects data in this directory from other users

7.3.3. Database Environment Variables

These variables pertain to your database (if you have one) and are used to connect your application to your database. The exact variable names depend on the database type; the value of ***<database>*** is MONGODB, MYSQL, or POSTGRESQL as appropriate. Note these connections are only available to your application internally; you cannot connect from an external source.

OpenShift does not currently support user changes to environment variables. This includes changing the default MySQL admin password (even outside of phpmyadmin). If you change your password, ensure that 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 password of your own choosing for these users.

Table 7.4. Database Environment Variables

Environment Variable Name	Example	Purpose
OPENSHIFT_<database>_DB_HOST	127.0.250.1	The hostname or IP address used to connect to the database
OPENSHIFT_<database>_DB_PORT	3306	The port the database server is listening on
OPENSHIFT_<database>_DB_USERNAME	admin	The database administrative username
OPENSHIFT_<database>_DB_PASSWORD	8ddTnst22X3Y	The database administrative user's password
OPENSHIFT_<database>_DB_SOCKET	$OPENSHIFT_HOMEDIR/mysql-5.1/socket/mysql.sock	An AF socket for connecting to the database (for non-scaled apps only)
OPENSHIFT_<database>_DB_URL	mysql://admin:8ddTnst22X3Y@127.0.250.1:3306/	Database connection URL

7.3.4. Jenkins Environment Variables

If your application includes Jenkins, you have access to the following variables.

Table 7.5. Jenkins Environment Variables

Environment Variable Name	Example	Purpose
JENKINS_USERNAME	system_builder	System builder account on the Jenkins server
JENKINS_PASSWORD	RnmXQlavsb4f	Password for the system builder account on the Jenkins server
JENKINS_URL	https://jenkins-namespace.example.com/	DNS name for the associated Jenkins server where builds occur

7.3.5. Gear Environment Variables

These variables apply to scaling applications.

Table 7.6. Gear Environment Variables

Environment Variable Name	Example	Purpose
OPENSHIFT_GEAR_DNS	gearname-namespace.example.com	The gear's fully-qualified domain name
OPENSHIFT_GEAR_NAME	gearname	The gear's name
OPENSHIFT_GEAR_UUID	0123456789abcdef0123456789abcdef	The gear's UUID

7.3.6. JBoss Environment Variables

OpenShift uses environment variables in the **`/<appname>/.openshift/config/standalone.xml`** file that is part of jbossas-7. The following example code from this **`standalone.xml`** file displays three environment variables:

```
<connection-
url>jdbc:mysql://${env.OPENSHIFT_MYSQL_DB_HOST}:${env.OPENSHIFT_MYSQL_DB_PORT}/${e
nv.OPENSHIFT_APP_NAME}</connection-url>
```

You can save environment variables on the server, which means that you do not have to pass sensitive information repeatedly to the command line.

Procedure 7.1. How to set environment variables on the server

1. Open the **`<appname>/.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 your environment variables is visible if you use the **`rhc snapshot`** commands.

7.4. Node.js

This section provides the basic information required for running Node.js applications on OpenShift.

7.4.1. Repository Layout

When you create a new Node.js application on OpenShift, the application directory is populated with several files. The following table displays this layout:

Table 7.7. Node.js Repository Layout

File	Use
node_modules/	Node modules packaged with the application
deplist.txt	Deprecated — List of npm modules to install with the application
npm_global_module_list	List of globally installed and available npm modules
server.js	OpenShift sample Node application
package.json	Package descriptor for npm
README	Information about using Node.js with OpenShift
.openshift/	Location for OpenShift specific files
.openshift/action_hooks/pre_build	Script that gets run every git push before the build
.openshift/action_hooks/build	Script that gets run every git push as part of the build process (on the CI system if available)
.openshift/action_hooks/deploy	Script that gets run every git push after build but before the application is restarted
.openshift/action_hooks/post_deploy	Script that gets run every git push after the application is restarted

You can create additional directories if needed, but do not delete the **`node_modules`** and **`.openshift`** directories.

Note

When you **`git push`**, everything in your remote repository directory is recreated. Place items that you want to persist between pushes, such as a database, in the **`../data`** directory on the gear where your OpenShift application is running.

7.4.2. Installing Node Modules

If you want to install Node modules from the npm registry, you can specify the modules in the **`package.json`** file. **`Package.json`** describes how your package is structured and is used by npm to install and manage your application. The dependencies hash in the **`package.json`** file can be used to specify the Node modules (and optionally the version) to install alongside your application ("locally") on the OpenShift environment when you **`git push`** your application. For more information, see https://npmjs.org/doc/json.html.

Example **`package.json`** entry:

```
"dependencies": {

  "coffee-script":  "1.3.3",

  "connect":  "*",

  "optimist": "<=0.3.4",

  "socket.io": ""

},
```

Node modules not included in the npm registry can be installed by packaging them in the **`node_modules`** directory.

The **`npm_global_module_list`** file contains a list of globally installed modules. You can override a globally available module by specifying it in **`package.json`** or by packaging it in the **`node_modules`** directory. Node gives preference to the "locally" installed version of that module.

7.5. Scheduling Timed Jobs with Cron

You can use the OpenShift cron scheduler to set up timed jobs for your applications. This consists of adding the cron scheduler cartridge to your application, adding the required cron jobs to the appropriate directories, and then updating the remote git repository. You can also use the **`rhc cartridge`** command to enable or disable your cron scripts.

The following example demonstrates how to create a new application and enable cron support.

Procedure 7.2. To create an application and enable cron support:

1. Create a new application:

   ```
   $ rhc app create holy php-5.3
   ```

2. Add the cron scheduler cartridge to your new application:

   ```
   $ rhc cartridge add cron-1.4 -a holy
   ```

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

   For example:

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

4. Commit your changes and push them to the remote repository.

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

This example appends a new line of date information to the **$OPENSHIFT_REPO_DIR/php/date.txt** file every minute.

You can use **curl** to verify the operation of this script:

```
$ 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 that you place in the **/cron** subdirectories are executed at the respective frequencies. Scripts in each subdirectory are executed sequentially, in alphabetical order. Scripts in the **/cron/hourly** directory are executed on the first minute of every hour. You can use the **rhc cartridge** command to enable or disable your cron scripts. For example, to disable all of your scripts:

```
$ rhc cartridge stop cron-1.4 -a holy
```

To enable all of your scripts:

```
$ rhc cartridge start cron-1.4 -a holy
```

Note

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

7.6. Sending and Receiving Email

7.6.1. Overview

OpenShift provides support for connecting to externally hosted email services (POP, IMAP, and SMTP). For developers, this means that you can establish a connection from your application to your own email server, or to one of the popular public email services, such as Gmail or YahooMail. If you are deploying popular blogging or wiki software on OpenShift, such as Drupal, Joomla, MediaWiki, or WordPress, you can also take advantage of this feature by altering the email settings of the software to point to the appropriate email service.

7.6.2. Email Port Configuration

The specific ports that have been enabled to support this are as follows:

- 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 will consume an extremely small share of the available bandwidth if congestion exists.

Important

Be aware that access to email servers from cloud providers may be blocked by *Realtime Blackhole Lists (RBLs)*. This may affect your ability to connect to some email servers. If you are unable to make a connection to one of these services, make sure that your email provider allows authenticated connections from Amazon AWS EC2 hosts.

Chapter 8. SSH Authentication

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

For successful authentication, the public SSH key on your computer must match the public key that has been uploaded to the OpenShift server. When you perform the initial configuration of the OpenShift client tools, 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 server.

If you have not performed the initial configuration of the OpenShift client tools, refer to the *Client Tools Installation Guide* for more information.

8.1. Resolving Authentication Issues

Occasionally your local public key may not match the public key for your account on the OpenShift server, or your key may 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, there are two ways you can generate a new pair of SSH keys:

- Use the interactive setup wizard (recommended)
- Generate and upload SSH keys manually

8.1.1. Using the Interactive Setup Wizard

The recommended method of resolving authentication issues is to use the interactive setup wizard to generate a new pair of SSH keys. The interactive setup wizard will also provide the option of automatically uploading your new public key to the OpenShift server. Use the command as shown below to launch the setup wizard and follow the onscreen instructions.

```
$ rhc setup
```

Refer to the *Client Tools Installation Guide* for more information about the OpenShift client tools interactive setup wizard.

8.2. Managing SSH Keys

8.2.1. Management Console

Using the OpenShift Management Console you can add, remove, and update public keys to control the access of other contributors to your OpenShift applications.

To view the public keys associated with your account, access the Management Console and click **`My Account`** in the navigation bar at the top of the page. Your current SSH public keys are listed in the **`Public Keys`** section.

8.2.1.1. Generating New Keys

The **`ssh-keygen`** command generates a new pair of RSA or DSA keys as specified. You can then use the Management Console to add the new keys to your account.

Procedure 8.1. To generate new SSH keys with the ssh-keygen command:

1. Manually generate a new pair of keys, replacing ***KeyType*** with the type of key you want to generate, either dsa or rsa:

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

2. By default the new SSH keys are located in the **`/home/username/.ssh/`** directory.

8.2.1.2. Adding a Key

Procedure 8.2. To add a key:

1. Access the Management Console and click **`My Account`** in the navigation bar at the top of the page.
2. In the **`Public Keys`** section, click **`Add a new key`**.
3. Enter a name for your key then paste the public key in the space provided.
4. Click **`Create`** to add your public key.

Important

If you copy and paste your SSH key from an editor or terminal with the word wrap function enabled, the key may include unnecessary line breaks. This causes the OpenShift web console to reject the SSH key and the upload process to fail. When you paste your key into the web console, confirm that the key contents are correct and do not contain any unnecessary line breaks.

8.2.1.3. Removing a Key

Procedure 8.3. To remove a key:

1. Access the Management Console and click **`My Account`** in the navigation bar at the top of the page.
2. In the **`Public Keys`** section, click **`Delete`** next to the key you want to remove.
3. A dialog box appears asking you to confirm the deletion. Click **`OK`** to confirm.

8.2.2. Command Line Interface

8.2.2.1. Generating and Uploading SSH Keys Manually

Although not recommended, advanced users can manually generate a new pair of SSH keys, and upload the public key to the OpenShift server. Follow the instructions below.

Procedure 8.4. To manually generate a new pair of SSH keys and upload the public key to the server:

1. Generate a new pair of keys:

   ```
   $ ssh-keygen -t <KeyType>
   ```

   where ***KeyType*** is the type of key you want to generate, either DSA or RSA.
2. Add the new public key to the user account:

```
$ rhc sshkey add <KeyName> <KeyPath> -l <UserName>
```

where ***<KeyPath>*** is the path and filename of the public key that you want to add, and ***<KeyName>*** is a name that you specify to identify this key.

3. Add the new public key to the SSH agent:

```
$ ssh-add <KeyPath>
```

8.2.2.2. Viewing Public Keys

Use the **`rhc sshkey list`** command to view all public keys that have been added to the OpenShift server for the specified user account:

```
$ rhc sshkey list -l <UserName>
Password:

RESULT:
Name: default
Type: ssh-rsa
Fingerprint: 43:f8:93:re:9f:a3:a8:f4:f3:34:g8:3d:1g:d8:3c:as
```

8.2.2.3. Adding a Key

Use the **`rhc sshkey add`** command to add a public key to the OpenShift server for the specified user account:

```
$ rhc sshkey add <KeyName> [<KeyPath>] -l <UserName>
Password:
Success
```

where ***<KeyName>*** is the user-specified identifier for the SSH key, and ***<KeyPath>*** is the path to an existing key (optional). If an existing key is not specified, a new key is generated and uploaded to the server.

Run **`rhc sshkey add --help`** for more information.

8.2.2.4. Removing a Key

Use the **`rhc sshkey remove`** command to remove an existing public key from the OpenShift server for the specified user account:

```
$ rhc sshkey remove <KeyName> -l <UserName>
```

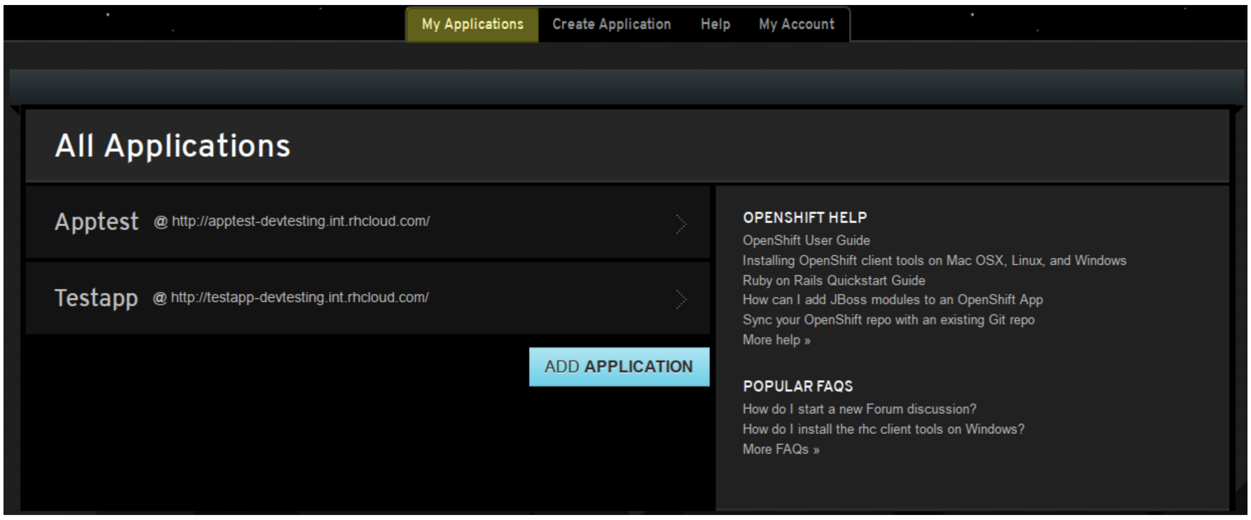
Chapter 9. Monitoring and Troubleshooting

9.1. Monitoring Application Resources

As described in Section 6.1.1, “Scaled Applications”, scaled applications are automatically allocated additional resources based on demand. The amount of resources consumed an application can be easily monitored and viewed from the OpenShift Management Console. Follow the instructions below to monitor and view application resources. The instructions below assume that a scaled application has already been created.

Procedure 9.1. To monitor application resources:

1. Access the OpenShift Management Console.
2. Click the **`My Applications`** tab to view all of your applications and click the name of the scaled application you want to examine.



All Applications

Apptest @ http://apptest-devtesting.int.rhcloud.com/

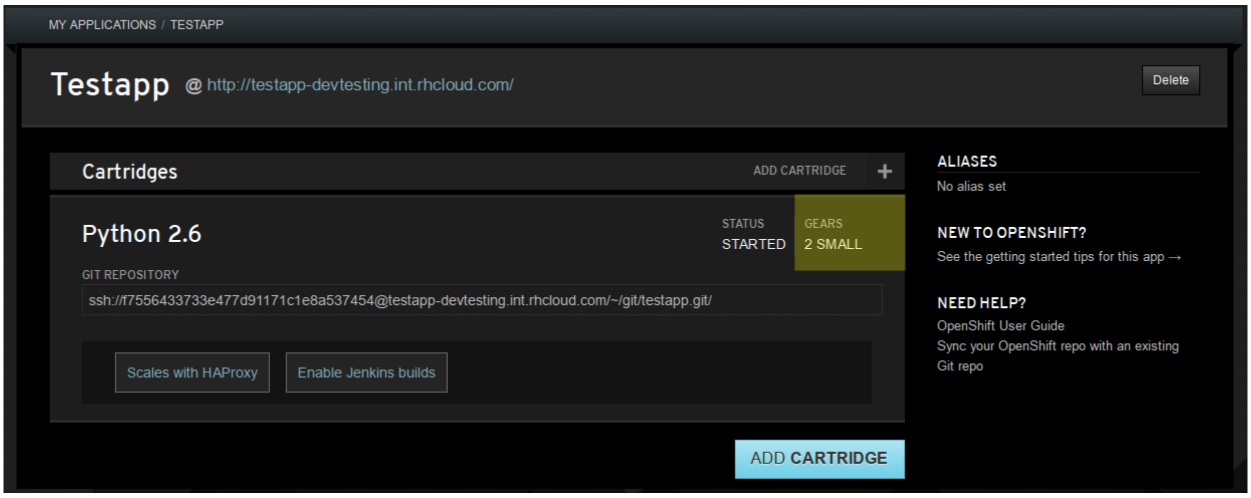
Testapp @ http://testapp-devtesting.int.rhcloud.com/

OPENSHIFT HELP
OpenShift User Guide
Installing OpenShift client tools on Mac OSX, Linux, and Windows
Ruby on Rails Quickstart Guide
How can I add JBoss modules to an OpenShift App
Sync your OpenShift repo with an existing Git repo
More help »

POPULAR FAQS
How do I start a new Forum discussion?
How do I install the rhc client tools on Windows?
More FAQs »

Figure 9.1. My Applications

3. The OpenShift Management Console displays the number of gears, along with the size of the gears, used by the selected application, as highlighted in the figure below.



Testapp @ http://testapp-devtesting.int.rhcloud.com/

Cartridges

Python 2.6

STATUS STARTED
GEARS 2 SMALL

GIT REPOSITORY
ssh://f7556433733e477d91171c1e8a537454@testapp-devtesting.int.rhcloud.com/~/git/testapp.git/

ALIASES
No alias set

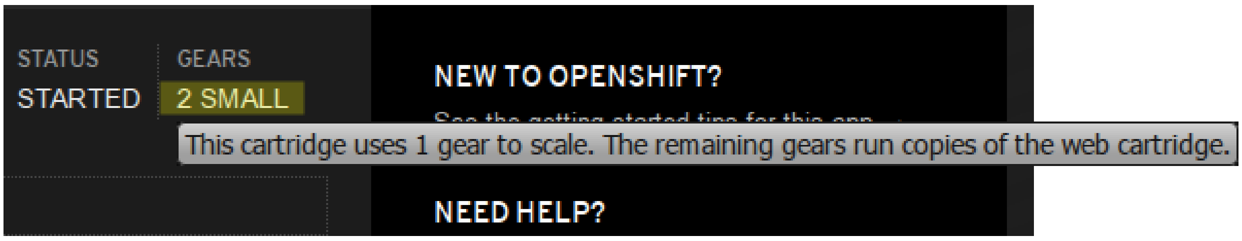
NEW TO OPENSHIFT?
See the getting started tips for this app →

NEED HELP?
OpenShift User Guide
Sync your OpenShift repo with an existing Git repo

Figure 9.2. Application Resource Information

4. Hover over the gear size information with your mouse to display a popup message with more

detailed information.



This cartridge uses 1 gear to scale. The remaining gears run copies of the web cartridge.

Figure 9.3. Application Gear Details

5. Scaling is performed by the HAProxy cartridge. Click **`Scales with HAProxy`** to get information about testing the scaling function of your application.

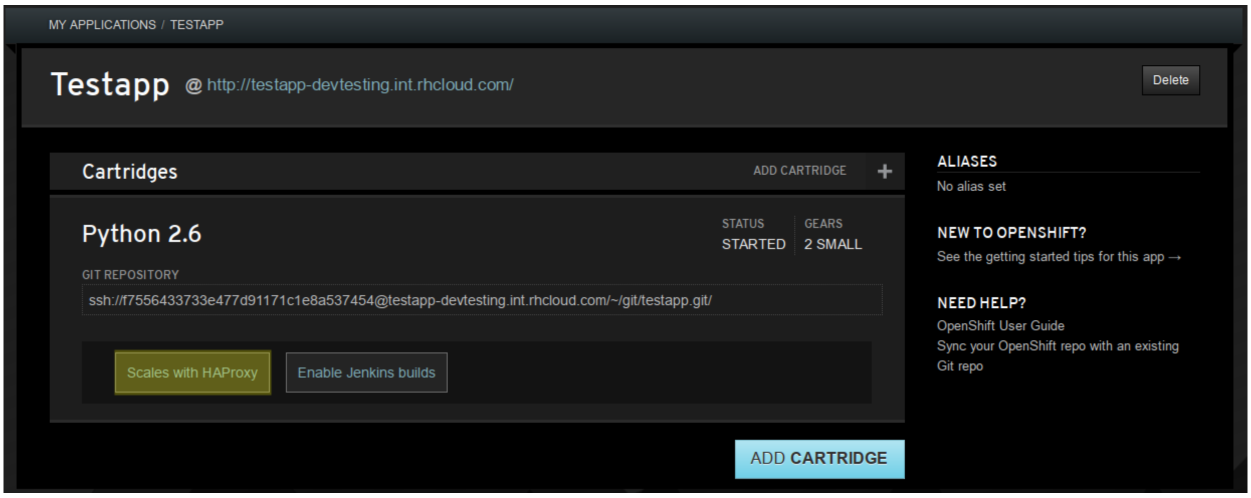

MY APPLICATIONS / TESTAPP
Testapp @ http://testapp-devtesting.int.rhcloud.com/
Delete
Cartridges
ADD CARTRIDGE +
Python 2.6
STATUS
STARTED
GEARS
2 SMALL
GIT REPOSITORY
ssh://f7556433733e477d91171c1e8a537454@testapp-devtesting.int.rhcloud.com/~/git/testapp.git/
Scales with HAProxy
Enable Jenkins builds
ADD CARTRIDGE
ALIASES
No alias set
NEW TO OPENSHIFT?
See the getting started tips for this app →
NEED HELP?
OpenShift User Guide
Sync your OpenShift repo with an existing Git repo

Figure 9.4. HAProxy Information

Note

At the time of this writing, the scaling function cannot be disabled from an application. The only way to disable scaling is to remove the scaled application, and create a new application without the scaling option.

9.2. MongoDB Monitoring Service (MMS)

The MongoDB Monitoring Service (MMS) is a cloud-based monitoring service, designed by 10gen, to monitor the health of MongoDB deployments. MMS provides an agent that runs on MongoDB servers, and this agent reports information such as opcounters, memory information, number of connections, network I/O, database storage size, and more. All of this information is available in customizable charts that are displayed in your web browser.

OpenShift provides built-in support for MMS; you can add the MMS agent to your application using the same commands as for other cartridges. The only pre-requisite for using MMS is that you have an account with 10gen; you can sign up for a free MMS account at https://mms.10gen.com/.

9.2.1. Configuring an Application with MMS

The following procedure describes how to configure your application to take advantage of the services provided by MMS. This procedure assumes that you have successfully created an application and added the MongoDB cartridge.

Procedure 9.2. How to set up your application to use MMS

1. Download the agent.

   When you log into MMS, you will see a "download the agent" link on the **Hosts** page. Click this link to download an agent preconfigured with your group credentials.
2. Create a directory named **mms** in your application's **.openshift** directory with the following command, replacing **app_directory** with the root directory for your application:

   ```
   $ mkdir ~/app_directory/.openshift/mms
   ```
3. Add the **settings.py** file to the **mms** directory:

   ```
   $ cp ~/mms-agent/settings.py ~/app_directory/.openshift/mms/
   ```

   Important

   The **settings.py** file contains the MMS agent settings that contain the API keys for connecting to your MMS account and updating the monitoring metrics. The MMS agent will not function without this file.
4. Use git to add and commit the **mms** directory to your local repository, then push it to your remote repository:

   ```
   $ cd app_directory/
   $ git add .openshift/mms/
   $ git commit -m "mms settings" -a
   $ git push
   ```
5. Use the following command to add the agent to your application:

   ```
   $ rhc cartridge add 10gen-mms-agent-0.1 -a app_name
   ```

After you have successfully completed this procedure, you can navigate to the https://mms.10gen.com/ page, enter your host's details and start monitoring your application.

9.2.2. Monitoring an Applications with MMS

9.2.2.1. Adding Hosts to MMS

After you have created an application and added the MMS agent, you can add the host to the **Hosts** page on https://mms.10gen.com/. To add a new host, you need the hostname, port number, and login credentials that were provided when you added MongoDB to your application. If you no longer have this information, you can retrieve it directly from your application, as follows:

1. Use SSH to connect to your application:

   ```
   $ ssh UUID@appname-namespace.example.com
   ```

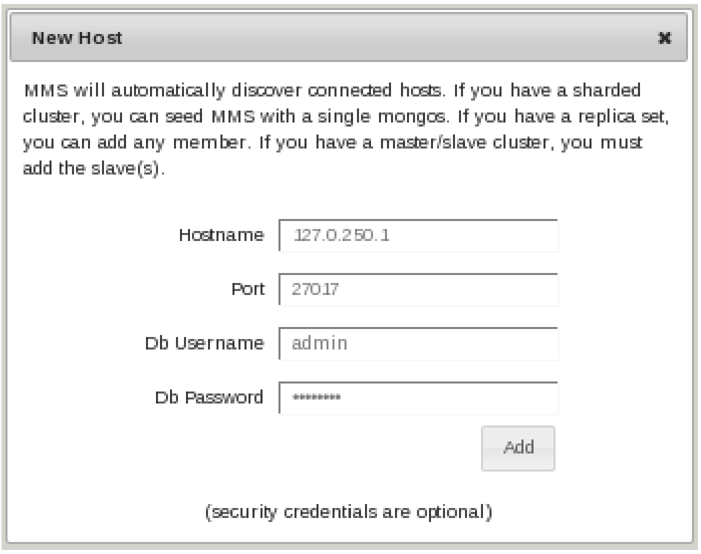
   where ***UUID*** is the UUID of your application. You can retrieve this using the **rhc apps** command. Replace ***appname*** and ***namespace*** with your application name and namespace, respectively.

2. Run the following command to retrieve all the necessary connection and credential information for your application:

```
> echo $OPENSHIFT_MONGODB_DB_URL
```

Procedure 9.3. How to add hosts to MMS

1. Navigate to https://mms.10gen.com/
2. Click the **`Hosts +`** button at the top of the page to display the **`New Host`** dialog box.
3. Enter the required details and then click **`Add`**. This adds the host details to the agent, which immediately starts collecting and storing data.



New Host

MMS will automatically discover connected hosts. If you have a sharded cluster, you can seed MMS with a single mongos. If you have a replica set, you can add any member. If you have a master/slave cluster, you must add the slave(s).

Hostname	127.0.250.1
Port	27017
Db Username	admin
Db Password	********

Add

(security credentials are optional)

Figure 9.5. Adding a New Host to MMS

9.2.2.2. MMS Monitoring with a Browser

After you have added the MMS agent to your application and added your host details to MMS, you can use the web interface to monitor your application's performance.

Procedure 9.4. How to monitor your applications with MMS

1. Navigate to https://mms.10gen.com/ and click the **`Hosts`** tab.
2. Click the name of the host that you want to monitor to view the available data collection streams. You can customize this page to suit your own requirements. Refer to the 10gen documentation for full details.

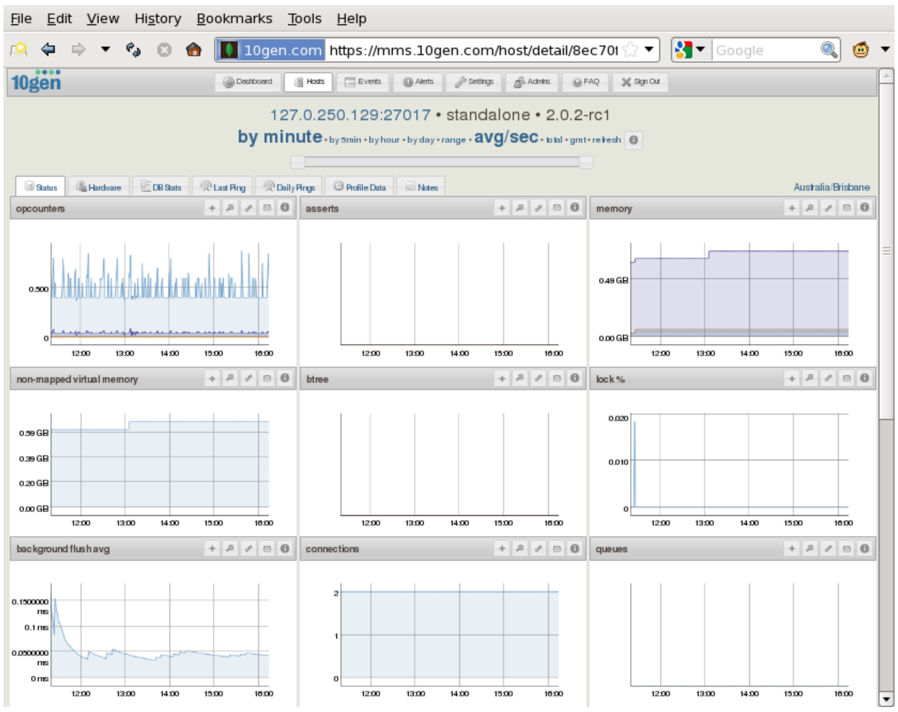

File Edit View History Bookmarks Tools Help
10gen.com https://mms.10gen.com/host/detail/8ec70
10gen
Dashboard Hosts Events Alerts Settings Admins FAQ Sign Out
127.0.250.129:27017 • standalone • 2.0.2-rc1
by minute • by 5min • by hour • by day • range • avg/sec • total • gmt • refresh
Status Hardware DB Stats Last Ping Daily Pings Profile Data Notes
Australia/Brisbane
opcounters
asserts
memory
non-mapped virtual memory
btree
lock %
background flush avg
connections
queues

Figure 9.6. Data Monitoring Using MMS

9.3. Troubleshooting JBoss Applications

OpenShift provides some basic tools to help you troubleshoot your JBoss applications should problems arise. This includes the ability to inspect the log files and also to trigger a thread dump for JBoss applications.

9.3.1. Using Thread Dumps

Thread dumps are a useful debugging tool. If an application appears to have stalled or is running out of resources, a thread dump can help to reveal the state of the server, to identify what might be causing any issues and ultimately to resolve the problem. You can use the **`rhc threaddump`** command to trigger a thread dump for a JBoss application.

The following example demonstrates the use of this command:

Note

This example assumes that you have a valid account and have already created a domain and JBoss application called "myJBoss".

```
$ rhc threaddump myJBoss
Password: **********

Application event 'thread-dump' successful
Success

The thread dump file will be available via: rhc tail myJBoss -f jbosseap-
6.0/jbosseap-6.0/standalone/tmp/jbosseap-6.0.log -o '-n 250'
```

You can now use the **`rhc tail`** command to inspect the resultant thread dump:

```
$ rhc tail myJBoss -f jbosseap-6.0/jbosseap-6.0/standalone/tmp/jbosseap-
6.0.log -o '-n 250'
Password: **********

<sample output>

Heap
 def new generation   total 18240K, used 4240K [0xdd580000, 0xde940000,
0xe2ad0000)
  eden space 16256K,  24% used [0xdd580000, 0xdd956930, 0xde560000)
  from space 1984K,  15% used [0xde750000, 0xde79d9a0, 0xde940000)

<output truncated>
```

9.3.2. Inspecting Server, Boot and Other Log Files

You can use the **rhc tail** command to inspect other JBoss application log files. If you include your application name as the only argument, this command will tail the contents of all files in the ***<cartridge>***/logs/ directory. The following shows abbreviated output from this command:

```
$ rhc tail myJBoss
Password: **********

==> jbosseap-6.0/logs/server.log <==
2013/01/02 02:06:37,144 INFO  [org.jboss.as.osgi] (MSC service thread 1-1)
JBAS011907: Register module: Module "deployment.ROOT.war:main" from Service
Module Loader

==> jbosseap-6.0/logs/boot.log <==
 user.name = 1ca72484953b4701a4d32be920bcefc1
 user.timezone = America/New_York
02:06:29,566 DEBUG [org.jboss.as.config] VM Arguments: -D[Standalone] -Xmx256m -
XX:MaxPermSize=128m -XX:+AggressiveOpts -Dorg.apache.tomcat.util.LOW_MEMORY=true -
Dorg.jboss.resolver.warning=true

<output truncated>
```

9.4. Performing Application Maintenance from Your Workstation

9.4.1. Port Forwarding

You can forward remote ports on your server to your workstation to make it easier to manage various services, such as MySQL. The **rhc port-forward** command provides a wrapper for the **ssh** command, and determines which remote ports should be forwarded to your workstation.

Note

Ensure that your application is running before attempting to configure port forwarding.

Refer to the following example:

```
$ rhc port-forward App01
Password: **********

Checking available ports...
Forwarding ports
  Service Connect to              Forward to
  ===== ================== ==== ==================
  httpd 127.4.182.129:8080  =>  127.4.182.129:8080
Press CTRL-C to terminate port forwarding
```

In this example, you could now open a browser and connect to your remote application as if it were running locally.

The current implementation of the **rhc port-forward** command forwards all open ports on your running application to your local workstation. If you have multiple cartridges added to your application, you will see further messages indicating which remote services are being bound to local ports. Refer to the following example:

```
$ rhc port-forward myJBoss
Password: **********

Checking available ports...
Forwarding ports
  Service Connect to            Forward to
  ====== ================ ==== ================
  java   127.4.188.1:3528  =>  127.4.188.1:3528
  java   127.4.188.1:4447  =>  127.4.188.1:4447
  java   127.4.188.1:5445  =>  127.4.188.1:5445
  java   127.4.188.1:5455  =>  127.4.188.1:5455
  java   127.4.188.1:8080  =>  127.4.188.1:8080
  java   127.4.188.1:9990  =>  127.4.188.1:9990
  java   127.4.188.1:9999  =>  127.4.188.1:9999
  mysqld 127.4.188.1:3306  =>  127.4.188.1:3306
Press CTRL-C to terminate port forwarding
```

If you need to forward only specific ports, you need to use the **ssh** command with the **-L** option. Refer to the **ssh** manual page for details.

9.4.1.1. 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, you may experience error messages similar to those shown below when attempting to configure port forwarding using the IP address for your OpenShift application.

```
$ rhc port-forward app01
Password: **********

Checking available ports...
Error trying to forward ports. You can try to forward manually by running:
ssh -N 70277280b8534c8a9fc76d2734393dfa@app01-namespace.example.com
```

The current workaround to enable port forwarding on Mac OS X is to configure an alias manually using the **ifconfig** command for each IP address used by your application, using the command as shown below:

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

For example, if the IP address used by your application is 127.10.51.129, run the command as shown below:

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

If your application uses multiple IP addresses, as shown in the example above, you must configure an alias for each IP address. For example, suppose you have a Node.js application with both MySQL and phpMyAdmin cartridges added, and it uses the IP addresses *127.11.25.1* and *127.11.25.2*. To correctly enable port forwarding, you must configure an alias for each IP address, as shown in the example below.

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

Note

The **`ifconfig`** command on Mac OS X requires root/administrative privileges to execute.

You can now use the **`rhc port-forward`** command to enable port forwarding.

Important

The IP address alias you configure for your OpenShift application is not persistent through system reboots. If you reboot your computer, you must repeat these steps to correctly enable port forwarding on Mac OS X.

Chapter 10. Backing Up and Restoring Applications

Use the **`rhc snapshot save`** command to create backups of your OpenShift applications, and also to restore those backups to the server. As a back-up tool, this command creates a gzipped tar file not only of your application but also of any locally-created log and other files, which is then downloaded to your local machine. The directory structure that exists on the server is maintained in the tar file.

Important

OpenShift does not maintain backups of your applications or user data on the OpenShift servers. The files created by this process are always downloaded to your local machine.

10.1. Creating Application Snapshots

The **`rhc snapshot save`** command uses the following syntax to create a snapshot:

```
$ rhc snapshot save Application_Name
```

The command will prompt for any missing information, such as your ***`rhlogin`*** and ***`password`***. The default filename for the snapshot is ***`$Application_Name`*`.tar.gz`**. You can override this path and filename with the **`--filepath`** option.

The following example demonstrates the process of creating and downloading a snapshot:

```
$ rhc snapshot save MasterApp
Password: **********

Pulling down a snapshot to MasterApp.tar.gz...
Waiting for stop to finish
Done
Creating and sending tar.gz
Done

RESULT:
Success
```

10.2. Restoring Application Snapshots

You can use the **`rhc snapshot restore`** command to restore snapshots to the server. This form of the command restores the git repository, the application data directories and the log files found in the specified archive. When the restoration is complete, OpenShift runs the deployment script on the newly-restored repository, which completes the deployment process in the same way as if you had run **`git push`**.

Important

Even though you can save your application using a different filename, you cannot restore that application to a different name. That is, if your application name is "MyApp", you can save it as **`OurApp.tar.gz`**, but when you restore it you must use the original application name.
The **`rhc snapshot restore`** command overwrites the remote git repository. Any changes that you might have made since you took the snapshot will be lost.

Warning

Importing snapshot data into a local environment may delete local content, for example a user table in your database. If you are unsure what effect a snapshot import may have on your local data, use SSH to access your application and make the backup directly.

The following example demonstrates the process of restoring a snapshot:

```
$ rhc snapshot restore MasterApp
Password: **********

Restoring from snapshot MasterApp.tar.gz...
restart_on_add=false
Waiting for stop to finish
Done
Removing old git repo: ~/git/MasterApp.git/
Removing old data dir: ~/app-root/data/*
Restoring ~/git/MasterApp.git and ~/app-root/data
restart_on_add=false
~/git/MasterApp.git ~
~
Running .openshift/action_hooks/pre_build
Running .openshift/action_hooks/build
Running .openshift/action_hooks/deploy
hot_deploy_added=false
Done
Running .openshift/action_hooks/post_deploy

RESULT:
Success
```

Revision History

Revision 2.0.25-0	Tue Apr 02 2013	Brian Moss

OpenShift Online 2.0-25 release.
Add Tomcat 7 to list of application types.
Add content for deleting an application using the CLI.
Add Jenkins-related logging info.
Fix typos and grammatical errors.
Add SwitchYard 0.6 to the list of supported cartridges.
Remove application version numbers.
Update Applications and Cartridges chapter.
Add Introduction chapter.
Remove Introduction-style content from Preface.
Changed all mentions of Getting Started Guide to Client Tools Installation Guide
Add 'rhc apps' and 'rhc account' usage

Revision 2.0.24-0.0	Thu Feb 28 2013	Brian Moss

OpenShift Online 2.0-24 release.
Restructure table of contents.
Remove author from pdf output.

Revision 2.0.22-0	Fri Feb 15 2013	Brian Moss

OpenShift Online 2.0-22 release.
Refactor rhc commands.
Add Tomcat 6 and Zend 5.6 to list of application types.
Update MongoDB 2.0 to MongoDB 2.2.
Update environment variables.
Add import warning to snapshot section.
Update web interface chapter.

Index