OpenShift Container Platform 3.11

Getting Started

Getting Started with OpenShift Container Platform 3.11

Last Updated: 2018-10-18

OpenShift Container Platform 3.11 Getting Started

Getting Started with OpenShift Container Platform 3.11

Legal Notice

Abstract

Whether you are a developer or a platform administrator, you can get started with OpenShift using the topics in this book. Administrators can use the installation utility and an interactive CLI tool, to quickly install and configure a new OpenShift instance across multiple hosts. Developers can use the OpenShift CLI or the web console to log in to an existing OpenShift instance and start creating applications.

Table of Contents

CHAPTER 1. OVERVIEW

1.1. INTRODUCTION

OpenShift Container Platform brings together Docker and Kubernetes, and provides an API to manage these services. OpenShift Container Platform allows you to create and manage containers.

1.1.1. Why Should I Use OpenShift?

Containers are standalone processes that run within their own environment, independent of operating system and the underlying infrastructure. OpenShift helps you to develop, deploy, and manage container-based applications. It provides you with a self-service platform to create, modify, and deploy applications on demand, thus enabling faster development and release life cycles.

Think of images as cookie cutters and containers as the actual cookies.

Think of OpenShift as an operating system, images as applications that you run on them, and the containers as the actual running instances of those applications.

If you already have OpenShift Container Platform installed, find the appropriate topic based on your role to get started:

I am a…	Links to relevant topics
Platform administrator	Install basic OpenShift Container Platform environment or Install production OpenShift Container Platform environment
Developer	Step through the basics of how to Create and Build an Image Using the Web Console and create your first project and application.

CHAPTER 2. INSTALL OPENSHIFT CONTAINER PLATFORM

2.1. OVERVIEW

This guide introduces you to the basic concepts of OpenShift Container Platform, and helps you install a basic application. This guide is not suitable for deploying or installing a production environment of OpenShift Container Platform.

2.1.1. Prerequisites

To install OpenShift Container Platform, you will need:

- At least two physical or virtual RHEL 7+ machines, with fully qualified domain names (either real world or within a network) and password-less SSH access to each other. This guide uses **`master.openshift.example.com`** and **`node.openshift.example.com`**. These machines must be able to ping each other using these domain names.
- A valid Red Hat subscription.
- Wildcard DNS resolution that resolves your domain to the IP of the node. So, an entry like the following in your DNS server:

```
master.openshift.example.com. 300 IN A <master_ip>
node.openshift.example.com. 300 IN A <node_ip>
*.apps.openshift.example.com. 300 IN A <node_ip>
```

WHY THE APPS IN YOUR DOMAIN NAME FOR THE WILDCARD ENTRY?

When using OpenShift Container Platform to deploy applications, an internal router needs to proxy incoming requests to the corresponding application pod. By using **apps** as part of the application domains, the application traffic is accurately marked to the right pod.

You can use anything other than **apps**.

```
*.cloudapps.openshift.example.com. 300 IN A <node_ip>
```

Once these are configured, use the following steps to set up a two-machine OpenShift Container Platform install.

2.1.2. Attach OpenShift Container Platform Subscription

1. As root on the target machines (both master and node), use **`subscription-manager`** to register the systems with Red Hat.

```
$ subscription-manager register
```

2. Pull the latest subscription data from RHSM:

```
$ subscription-manager refresh
```

3. List the available subscriptions.

```
$ subscription-manager list --available
```

4. Find the pool ID that provides OpenShift Container Platform subscription and attach it.

```
$ subscription-manager attach --pool=<pool_id>
```

5. Replace the string **<pool_id>** with the pool ID of the pool that provides OpenShift Container Platform. The pool ID is a long alphanumeric string.

These RHEL systems are now authorized to install OpenShift Container Platform. Now you need to tell the systems from where to get OpenShift Container Platform.

2.1.3. Set Up Repositories

On both master and node, use **subscription-manager** to enable the repositories that are necessary in order to install OpenShift Container Platform. You may have already enabled the first two repositories in this example.

```
$ subscription-manager repos --enable="rhel-7-server-rpms" \
    --enable="rhel-7-server-extras-rpms" \
    --enable="rhel-7-server-ose-3.11-rpms" \
    --enable="rhel-7-server-ansible-2.6-rpms"
```

This command tells your RHEL system that the tools required to install OpenShift Container Platform will be available from these repositories. Now we need the OpenShift Container Platform installer that is based on Ansible.

2.1.4. Install the OpenShift Container Platform Package

The installer for OpenShift Container Platform is provided by the **openshift-ansible** package. Install it using **yum** on both the master and the node, after running **yum update**.

```
$ yum -y install wget git net-tools bind-utils iptables-services bridge-
utils bash-completion kexec-tools sos psacct
$ yum -y update
$ yum -y install openshift-ansible
$ yum -y install docker
```

2.1.5. Set up Password-less SSH Access

Before running the installer on the master, set up password-less SSH access as this is required by the installer to gain access to the machines. On the master, run the following command.

```
$ ssh-keygen
```

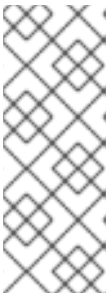
Follow the prompts and just hit enter when asked for pass phrase.

An easy way to distribute your SSH keys is by using a **bash** loop:

```
$ for host in master.openshift.example.com \
    node.openshift.example.com; \
    do ssh-copy-id -i ~/.ssh/id_rsa.pub $host; \
    done
```

2.1.6. Run the Installation Playbooks

1. See Example Inventory Files and select an example that closest matches your desired cluster configuration.



NOTE

Further example host files are available as references in the ***/usr/share/doc/openshift-ansible-docs-3.11.<version>/docs/example-inventories/*** directory (replacing **`<version>`** with your latest installed version of the **openshift-ansible-docs** package, which will update over time as the **openshift-ansible** parent package is upgraded). See Configuring Your Inventory File for full documentation on available inventory variables.

2. Edit the example inventory to use your host names, then save it to a file (default location is ***/etc/ansible/hosts***).

3. Change to the playbook directory and run the ***prerequisites.yml*** playbook using your inventory file:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i <inventory_file> playbooks/prerequisites.yml
```

4. Change to the playbook directory and run the ***deploy_cluster.yml*** playbook using your inventory file:

```
$ cd /usr/share/ansible/openshift-ansible
$ ansible-playbook -i <inventory_file> playbooks/deploy_cluster.yml
```

After a successful install, but before you add a new project, you must set up basic authentication, user access, and routes.

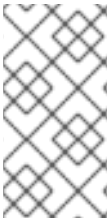
2.2. INTERACT WITH OPENSHIFT CONTAINER PLATFORM

OpenShift Container Platform provides two command line utilities to interact with it.

- **`oc`**: for normal project and application management
- **`oc adm`**: for administrative tasks
  When running **`oc adm`** commands, you should run them only from the first master listed in the Ansible host inventory file, by default ***/etc/ansible/hosts***.

Use **`oc --help`** and **`oc adm --help`** to view all available options.

In addition, you can use the web console to manage projects and applications. The web console is available at **`https://<master_fqdn>:8443/console`**. In the next section, you will see how to create user accounts for accessing the console.



NOTE

You can interact with your OpenShift Container Platform instance from a remote system as well, using these command line utilities. Bundled as the OpenShift CLI, you can download these utilities for Windows, Mac, or Linux environments in the CLI Reference section.

2.3. UNDERSTAND ROLES AND AUTHENTICATION

By default, when installed for the first time, there are no roles or user accounts created in OpenShift Container Platform, so you need to create them. You have the option to either create new roles or define a policy that allows anyone to log in (to start you off).

Before you do anything else, log in at least one time with the default **system:admin** user, on the master run the following command.

```
$ oc login -u system:admin
```

NOTE

All commands from now on should be executed on the master, unless otherwise indicated.

By logging in at least one time with this account, you will create the **system:admin** user's configuration file, which will allow you to log in subsequently.

There is no password for this system account.

Run the following command to verify that OpenShift Container Platform was installed and started successfully. You will get a listing of the master and node, in the **Ready** status.

```
$ oc get nodes
```

To continue configuring your basic OpenShift Container Platform environment, follow the steps outlined in Configure OpenShift Container Platform.

CHAPTER 3. CONFIGURE OPENSHIFT CONTAINER PLATFORM

3.1. OVERVIEW

This guide introduces you to the basic concepts of OpenShift Container Platform, and helps you configure a basic application. This guide provides the configuration steps following the installation of a basic OpenShift Container Platform environment, and is not suitable for deploying or installing a production environment of OpenShift.

3.2. CHANGE LOG IN IDENTITY PROVIDER

The default behavior of a freshly installed OpenShift Container Platform instance is to deny any user from logging in. To change the authentication method to HTPasswd:

1. Open the ***/etc/origin/master/master-config.yaml*** file in edit mode.
2. Find the **`identityProviders`** section.
3. Change **`DenyAllPasswordIdentityProvider`** to **`HTPasswdPasswordIdentityProvider`** provider.
4. Change the value of the name label to **`htpasswd_auth`** and add a new line **`file: /etc/origin/openshift-passwd`** in the provider section.
   An example **`identityProviders`** section with **`HTPasswdPasswordIdentityProvider`** would look like the following.

```
oauthConfig:
  ...
  identityProviders:
  - challenge: true
    login: true
    name: htpasswd_auth provider
    provider:
      apiVersion: v1
      kind: HTPasswdPasswordIdentityProvider
      file: /etc/origin/openshift-passwd
```

5. Save the file.

3.3. CREATE USER ACCOUNTS

Now that you are using the **`HTPasswdPasswordIdentityProvider`** provider, you need to generate these user accounts.

1. You can use the **httpd-tools** package to obtain the **htpasswd** binary that can generate these accounts.

```
# yum -y install httpd-tools
```

2. Create a user account.

```
# touch /etc/origin/openshift-passwd
# htpasswd -b /etc/origin/openshift-passwd admin redhat
```

You have created a user, **admin**, with the password, **redhat**.

3. Restart OpenShift before going forward.

```
# master-restart api
# master-restart controllers
```

4. Give this user account **cluster-admin** privileges, which allows it to do everything.

```
$ oc adm policy add-cluster-role-to-user cluster-admin admin
```

When running **oc adm** commands, you should run them only from the first master listed in the Ansible host inventory file, by default */etc/ansible/hosts*.

5. You can use this username/password combination to log in via the web console or the command line. To test this, run the following command.

```
$ oc login -u admin
```

Before going forward, change to the **default** project.

```
$ oc project default
```

For more details, see roles and authentication.

3.4. DEPLOY THE OPENSHIFT ROUTER

The OpenShift router is the entry point for external network traffic destined for OpenShift services. It supports HTTP, HTTPS, and any TLS-enabled traffic that uses SNI, which enables the router to send traffic to the correct service.

Without the router, OpenShift services and pods are unable to communicate with any resource outside of the OpenShift instance.

The installer creates a default router.

1. Delete the default router using the following command.

```
$ oc delete all -l router=router
```

2. Create a new default router.

```
$ oc adm router --replicas=1 --service-account=router
```

The OpenShift documentation contains detailed information on Router Overview.

3.5. DEPLOY AN INTERNAL REGISTRY

Openshift provides an internal, integrated container image registry that can be deployed to locally manage images. OpenShift uses the **docker-registry** to store, retrieve, and build container images, as well as deploy and manage them throughout their lifecycle.

The installer creates a default registry.

1. Delete the default registry using the following command.

```
$ oc delete all -l docker-registry=default
```

2. Create the **docker-registry** service in the **default** project using the **registry** service account.

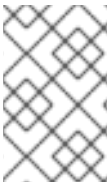
```
$ oc adm registry
```

3.6. CREATE PERSISTENT STORAGE FOR THE REGISTRY

The registry that you created in the previous step stores images and metadata, and uses an ephemeral volume for any pod deployment if persistent storage is not configured. This ephemeral volume is destroyed when the pod exits, losing all data, including any images built or pushed into the registry.

To configure persistent storage for the registry:

- Provision a volume that points to a storage server on your network (we will just create it on the master).
- Create a volume claim.
- Manually add the claim to the registry service.



NOTE

The following steps to configure persistent storage for the registry apply to storage for any image that requires persistent data and not just for the registry. The registry is just another image in the OpenShift environment.

3.6.1. Provision the Persistent Volume

1. Create a registry volume file on your master, as shown here, and call it ***registry-volume.yaml***.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: registry-volume
spec:
  capacity:
    storage: 3Gi
  accessModes:
  - ReadWriteMany
  nfs:
    path: /root/storage
    server: master.openshift.example.com
```

The folder */**root**/**storage*** must exist. Make sure to change the server entry to point to your master.

2. Create the registry persistent volume in OpenShift.

```
$ oc create -f registry-volume.yaml
```

3.6.2. Create the Persistent Volume Claim

Create a claim to bind the persistent volume created earlier. This claim is what ties the registry service to the persistent volume.

1. Create another file called ***registry-volume-claim.yaml***.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: registry-volume-claim
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 3Gi
```

2. Create the claim.

```
$ oc create -f registry-volume-claim.yaml
```

You have now created the Persistent Volume and the Persistent Volume Claim, and now need to add this claim to the registry.

3.6.3. Add the Persistent Volume Claim to the Registry

```
$ oc set volume dc/docker-registry \
    --add --overwrite -t persistentVolumeClaim \
    --claim-name=registry-volume-claim \
    --name=registry-storage
```

The **docker-registry** will now use the 3 GB persistent volume created for storing image and metadata.

CHAPTER 4. CREATE AND BUILD AN IMAGE USING THE WEB CONSOLE

4.1. OVERVIEW

This getting started experience walks you through the simplest way to get a sample project up and running on OpenShift Container Platform. There are a few different ways to launch images within a project, but this topic focuses on the quickest and easiest method.

If this is the first part of the documentation you have read, and you are unfamiliar with the core concepts of OpenShift Container Platform version 3 (v3), you might want to start by reading about what's new. This version of OpenShift Container Platform is significantly different from version 2 (v2).

OpenShift Container Platform 3 provides out of the box a set of languages and databases for developers with corresponding implementations and tutorials that allow you to kickstart your application development. Language support centers around the Quickstart templates, which in turn leverage builder images.

Language	Implementations and Tutorials
Ruby	Rails
Python	Django
Node.js	Node.js
PHP	CakePHP
Perl	Dancer
Java	

Other images provided by OpenShift Container Platform include:

- MySQL
- MongoDB
- PostgreSQL
- Jenkins

In addition, JBoss Middleware has put together a broad range of OpenShift Container Platform templates as well as images as part of their xPaaS services.

The technologies available with the xPaaS services in particular include:

- Java EE 6 Application Server provided by JBoss EAP 6
- Integration and Messaging Services provided by JBoss Fuse and JBoss A-MQ
- Data Grid Service provided by JBoss Data Grid

- Real Time Decision Service provided by JBoss BRMS
- Java Web Server 3.0 provided by Tomcat 7 and Tomcat 8

With each of these offerings, a series of combinations are provided:

- HTTP only versus HTTP and HTTPS
- No database required, or the use of either MongoDB, PostgreSQL, or MySQL
- If desired, integration with A-MQ

To help illustrate constructing such applications, the following sections guide you through creating a project that contains a sample Node.js application that will serve a welcome page and the current hit count (stored in a database).

NOTE

This topic discusses both Quickstart and Instant App templates and applications. Quickstarts provide a starting point for application development, but they rely on your development to create a useful application. In contrast, Instant Apps like Jenkins are instantly usable.

4.1.1. Browser Requirements

Review the browser versions and operating systems that can be used to access the web console.

4.2. BEFORE YOU BEGIN

Before you can get started:

- You must be able to access a running instance of OpenShift Container Platform. If you do not have access, contact your cluster administrator.
- Your instance must be pre-configured by a cluster administrator with the Instant App templates and builder images. If they are not available, direct your cluster administrator to the Loading the Default Image Streams and Templates topic.
- You must have the OpenShift Container Platform CLI downloaded and installed.

4.3. FORKING THE SAMPLE REPOSITORY

1. Visit the Ruby example page while you are logged in to GitHub.

NOTE

This topic follows the Ruby example, but you can follow along using any of the language examples provided in the OpenShift Container Platform GitHub project.

2. Fork the repository.
   You are redirected to your new fork.
3. Copy the clone URL for your fork.

4. Clone the repository to your local machine.

4.4. CREATING A PROJECT

To create an application, you must first create a new project, then select an InstantApp template. From there, OpenShift Container Platform begins the build process and creates a new deployment.

1. Visit the OpenShift Container Platform web console in your browser. The web console uses a self-signed certificate, so if prompted, continue past a browser warning.
2. Log in using the username and password recommended to you by your administrator.
3. To create a new project, click **New Project**.
4. Type a unique name, display name, and description for the new project.
5. Click **Create**.
   The web console's welcome screen loads.

4.5. CREATING AN APPLICATION

The Select Image or Template page gives you the option to create from a publicly accessible git repository, or from a template:

1. If creating a new project did not automatically redirect you to the Select Image or Template page, you might need to click **Add to Project**.
2. Click **Browse**, then select **ruby** from the drop-down list.
3. Click the **ruby:latest** builder image.
4. Type a **name** for your application, and specify the **Git Repository URL**, which is `https://github.com/<your_github_username>/ruby-ex.git`.
5. Optionally, click **Show advanced routing, build, and deployment options**, though by default this example application automatically creates a route, webhook trigger, and build change triggers.
6. Click **Create**.

NOTE

After creation, some of these settings can be modified from the web console by clicking **Browse**, **Builds**, select your build, then click **Actions**, and either **Edit** or **Edit YAML**.

Creating your application might take some time. You can follow along on the Overview page of the web console to see the new resources being created, and watch the progress of the build and deployment.

While the Ruby pod is being created, its status is shown as pending. The Ruby pod then starts up and displays its newly-assigned IP address. When the Ruby pod is running, the build is complete.

4.6. VERIFY THE APPLICATION IS RUNNING

If your DNS is correctly configured, then your new application can be accessed using a web browser. If you cannot access your application, then speak with your system administrator.

To view your new application:

1. In the web console, view the overview page to determine the web address for the application. For example, under **SERVICE: RUBY-EX** you should see something similar to: **`ruby-ex-my-test.example.openshiftapps.com`**.

2. Visit the web address for your new application.

4.7. CONFIGURING AUTOMATED BUILDS

You forked the source code for this application from the OpenShift Container Platform GitHub repository. Therefore, you can use a webhook to automatically trigger a rebuild of your application whenever you push code changes to your forked repository.

To set up a webhook for your application:

1. From the Web Console, navigate to the project containing your application.

2. Click the **Browse** tab, then click **Builds**.

3. Click your build name, then click the **Configuration** tab.

4. Click  next to **GitHub webhook URL** to copy your webhook payload URL.

5. Navigate to your forked repository on GitHub, then click **Settings**.

6. Click **Webhooks & Services**.

7. Click **Add webhook**.

8. Paste your webhook URL into the **Payload URL** field.

9. Click **Add webhook** to save.

GitHub now attempts to send a ping payload to your OpenShift Container Platform server to ensure that communication is successful. If you see a green check mark appear next to your webhook URL, then it is correctly configured. Hover your mouse over the check mark to see the status of the last delivery.

The next time you push a code change to your forked repository, your application will automatically rebuild.

4.8. WRITING A CODE CHANGE

To work locally and then push changes to your application:

1. On your local machine, use a text editor to change the sample application's source for the file ***ruby-ex/config.ru***

2. Make a code change that will be visible from within your application. For example: on line 229, change the title from **`Welcome to your Ruby application on OpenShift`** to **`This is my Awesome OpenShift Application`**, then save your changes.

3. Commit the change in git, and push the change to your fork.
   If your webhook is correctly configured, your application will immediately rebuild itself based on your changes. Once the rebuild is successful, view your updated application using the route that was created earlier.

Now going forward, all you need to do is push code updates and OpenShift Container Platform handles the rest.

4.8.1. Manually Rebuilding Images

You may find it useful to manually rebuild an image if your webhook is not working, or if a build fails and you do not want to change the code before restarting the build. To manually rebuild the image based on your latest committed change to your forked repository:

1. Click the **Browse** tab, then click **Builds**.

2. Find your build, then click **Start Build**.

CHAPTER 5. CREATE AND BUILD AN IMAGE USING THE CLI

5.1. OVERVIEW

This getting started experience walks you through the simplest way to get a sample project up and running on OpenShift Container Platform. There are a few different ways to launch images within a project, but this topic focuses on the quickest and easiest method.

If this is the first part of the documentation you have read, and you are unfamiliar with the core concepts of OpenShift Container Platform version 3 (v3), you might want to start by reading about what's new. This version of OpenShift Container Platform is significantly different from version 2 (v2).

OpenShift Container Platform 3 provides out of the box a set of languages and databases for developers with corresponding implementations and tutorials that allow you to kickstart your application development. Language support centers around the Quickstart templates, which in turn leverage builder images.

Language	Implementations and Tutorials
Ruby	Rails
Python	Django
Node.js	Node.js
PHP	CakePHP
Perl	Dancer
Java	

Other images provided by OpenShift Container Platform include:

- MySQL
- MongoDB
- PostgreSQL
- Jenkins

In addition, JBoss Middleware has put together a broad range of OpenShift Container Platform templates as well as images as part of their xPaaS services.

The technologies available with the xPaaS services in particular include:

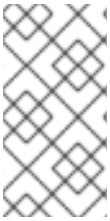
- Java EE 6 Application Server provided by JBoss EAP 6
- Integration and Messaging Services provided by JBoss Fuse and JBoss A-MQ
- Data Grid Service provided by JBoss Data Grid

- Real Time Decision Service provided by JBoss BRMS
- Java Web Server 3.0 provided by Tomcat 7 and Tomcat 8

With each of these offerings, a series of combinations are provided:

- HTTP only versus HTTP and HTTPS
- No database required, or the use of either MongoDB, PostgreSQL, or MySQL
- If desired, integration with A-MQ

To help illustrate constructing such applications, the following sections guide you through creating a project that contains a sample Node.js application that will serve a welcome page and the current hit count (stored in a database).



NOTE

This topic discusses both Quickstart and Instant App templates and applications. Quickstarts provide a starting point for application development, but they rely on your development to create a useful application. In contrast, Instant Apps like Jenkins are instantly usable.

5.2. BEFORE YOU BEGIN

Before you can get started:

- You must be able to access a running instance of OpenShift Container Platform. If you do not have access, contact your cluster administrator.
- Your instance must be pre-configured by a cluster administrator with the Instant App templates and builder images. If they are not available, direct your cluster administrator to theLoading the Default Image Streams and Templates topic.
- You must have the OpenShift Container Platform CLI downloaded and installed.

5.3. FORKING THE SAMPLE REPOSITORY

1. Visit the Ruby example page while you are logged in to GitHub.

NOTE

This topic follows the Ruby example, but you can follow along using any of the language examples provided in the OpenShift Container Platform GitHub project.

2. Fork the repository.
   You are redirected to your new fork.
3. Copy the clone URL for your fork.
4. Clone the repository to your local machine.

5.4. CREATING A PROJECT

To create an application, you must create a new project and specify the location of the source. From there, OpenShift Container Platform begins the build process and creates a new deployment.

1. Log into OpenShift Container Platform from the CLI:

   - With username and password:

     ```
     $ oc login -u=<username> -p=<password> --server=<your-openshift-
     server> --insecure-skip-tls-verify
     ```

   - With oauth token:

     ```
     $ oc login <https://api.your-openshift-server.com> --token=
     <tokenID>
     ```

2. To create a new project:

   ```
   $ oc new-project <projectname> --description="<description>" --
   display-name="<display_name>"
   ```

After creating the new project, you will be automatically switched to the new project namespace.

5.5. CREATING AN APPLICATION

To create a new application from the code in your forked repository:

1. Create the application by specifying the source of the code:

   ```
   $ oc new-app openshift/ruby-20-
   centos7~https://github.com/<your_github_username>/ruby-ex
   ```

   OpenShift Container Platform finds the matching builder image (which in this case is **ruby-20-centos7**) and then creates resources for the application (image stream, build configuration, deployment configuration, service). It also schedules the build.

2. Track the progress of the build:

   ```
   $ oc logs -f bc/ruby-ex
   ```

3. Once the build is complete and the resulting image has successfully pushed to your registry, check the status of your application:

   ```
   $ oc status
   ```

   Or you can view the build from the web console.

Creating your application might take some time. You can follow along on the Overview page of the web console to see the new resources being created, and watch the progress of the build and deployment. You can also use the **`oc get pods`** command to check when the pod is up and running, or the **`oc get builds`** command to see build statistics.

While the Ruby pod is being created, its status is shown as pending. The Ruby pod then starts up and displays its newly-assigned IP address. When the Ruby pod is running, the build is complete.

The **oc status** command tells you what IP address the service is running; the default port it deploys to is 8080.

5.6. CREATE A ROUTE

An OpenShift Container Platform route exposes a service at a host name, so that external clients can reach it by name. To create a route to your new application:

1. Expose a service for **ruby-ex**:

```
$ oc expose service ruby-ex
```

2. View your new route:

```
$ oc get route
```

3. Copy the route location, which will be something like **ruby-ex-my-test.example.openshiftapps.com**.

5.7. VERIFY THE APPLICATION IS RUNNING

To view your new application, paste the route location that you copied (in the previous section) into the address bar of your web browser and hit enter.

The example **ruby-ex** application is a simple welcome screen, and contains details on how to deploy code changes, manage your application, and other development resources.

Next, configure automated builds via a GitHub webhook trigger, so that code changes in your forked repository cause your application to rebuild.

5.8. CONFIGURING AUTOMATED BUILDS

You forked the source code for this application from the OpenShift Container Platform GitHub repository. Therefore, you can use a webhook to automatically trigger a rebuild of your application whenever you push code changes to your forked repository.

To set up a webhook for your application:

1. View the triggers section of the **BuildConfig** to verify that a GitHub webhook trigger exists:

```
$ oc edit bc/ruby-ex
```

You should see something similar to this:

```
triggers
- github:
    secret: Q1tGY0i9f1ZFihQbX07S
    type: GitHub
```

The secret ensures that only you and your repository can trigger the build.

2. Run the following command to display the webhook URLs associated with your **BuildConfig**:

```
$ oc describe bc ruby-ex
```

3. Copy the GitHub webhook payload URL output by the above command.
4. Navigate to your forked repository on GitHub, then click **Settings**.
5. Click **Webhooks & Services**.
6. Click **Add webhook**.
7. Paste your webhook URL into the **Payload URL** field.
8. Click **Add webhook** to save.

GitHub now attempts to send a ping payload to your OpenShift Container Platform server to ensure that communication is successful. If you see a green check mark appear next to your webhook URL, then it is correctly configured. Hover your mouse over the check mark to see the status of the last delivery.

The next time you push a code change to your forked repository, your application will automatically rebuild.

5.9. WRITING A CODE CHANGE

To work locally and then push changes to your application:

1. On your local machine, use a text editor to change the sample application's source for the file ***ruby-ex/config.ru***
2. Make a code change that will be visible from within your application. For example: on line 229, change the title from **`Welcome to your Ruby application on OpenShift`** to **`This is my Awesome OpenShift Application`**, then save your changes.
3. Commit the change in git, and push the change to your fork.
   If your webhook is correctly configured, your application will immediately rebuild itself based on your changes. Once the rebuild is successful, view your updated application using the route that was created earlier.

Now going forward, all you need to do is push code updates and OpenShift Container Platform handles the rest.

5.9.1. Manually Rebuilding Images

You may find it useful to manually rebuild an image if your webhook is not working, or if a build fails and you do not want to change the code before restarting the build. To manually rebuild the image based on your latest committed change to your forked repository:

```
$ oc start-build ruby-ex
```

5.10. TROUBLESHOOTING

Changing Projects

Although the **`oc new-project`** command automatically sets your current project to the one you've just created, you can always change projects by running:

```
$ oc project <project-name>
```

To view a list of projects:

```
$ oc get projects
```

Manually Triggering Builds

If the build does not start automatically, start a build and stream the logs:

```
$ oc start-build ruby-ex --follow
```

Alternatively, do not include **`--follow`** in the above command, and instead issue the following command after triggering the build, where **n** is the number of the build to track:

```
$ oc logs -f build/ruby-ex-n
```