## Table of Contents



# Deployment Life Cycle Lab

This lab includes the following sections:

- **Roll Back, Activate, and Code Life Cycle**

  In this section, you manage the various phases of the deployment's life cycle.

- **Create and Use Web Hooks**

  In this section, you create a Git webhook and start a new build and a new deployment automatically by pushing a code change in your Git repository.

# 1. Roll Back, Activate, and Code Life Cycle

No one codes correctly all the time. On occasion, you may want to revert to a previous incarnation of your application to restart a programming task. Other times, you may want to move to a newer version.

In this section, you take user `marina`'s `hello-ruby` application, modify its front end, and then rebuild. Afterwards, you revert to the original version and then go on to your rebuilt version.

The next sections require a GitHub account.

## 1.1. Fork Repository

If you have not done so already: from the Git web interface, click **Fork** in the upper right corner of the GitHub web UI to fork the Git repository `https://github.com/openshift/ruby-hello-world` into your own account.

## 1.2. Create Your Application

Remember that `buildconfig` (the build-configuration file) instructs OpenShift Enterprise on how to perform a build.

1. As `root`, create a project for user `marina`:

```
[root@master00 ~]# oadm new-project lifecycle --display-name="Lifecycle Lab" \
    --description="This is the project we use to learn about Lifecycle management" \
    --admin=marina --node-selector='region=primary'
```

2. Switch to user `marina` and use the `lifecycle` project:

```
[root@master00 ~]# su - marina
[marina@master00 ~]$ oc project lifecycle
```

3. Create an application from the `https://github.com/openshift/ruby-hello-world` repository:

```
[marina@master00~]$ oc new-app https://github.com/openshift/ruby-hello-world --
strategy=source
```

The `--strategy=source` option forces `oc new-app` to adopt the S2I strategy. A simpler alternative is the `new-app` command using your own repository, but you are intentionally picking the "wrong" repository as part of this learning exercise.

4. Run `oc env` to add the environment variables for a database to be used later:

```
[marina@master00~]$ oc env dc/ruby-hello-world MYSQL_USER=mysqluser \
  MYSQL_PASSWORD=redhat MYSQL_DATABASE=mydb
```

5. Waiting for the build to finish. Meanwhile, expose your service to the world so that you can test it from your local browser:

```
[marina@master00~]$ oc expose service ruby-hello-world \
  --name="ruby-hello-world" \
  --hostname=ruby-hello-world.lifecycle.cloudapps-${guid}.oslab.opentlc.com
```

6. View the current `buildconfig` for your application:

```
[marina@master00~]$ oc get buildconfig ruby-hello-world -o yaml
```

7. Verify that the output is similar to the following:

```
apiVersion: v1
kind: BuildConfig
```

```
  metadata:
    creationTimestamp: 2015-07-11T03:44:43Z
    name: ruby-hello-world
    namespace: lifecycle
    resourceVersion: "10546"
    selfLink: /osapi/v1beta3/namespaces/lifecycle/buildconfigs/ruby-hello-world
    uid: 2ad8d8bc-277f-11e5-a5f8-2cc260072896
  spec:
    output:
      to:
        kind: ImageStreamTag
        name: ruby-hello-world:latest
    resources: {}
    source:
      git:
        uri: https://github.com/openshift/ruby-hello-world
      type: Git
    strategy:
      dockerStrategy:
        from:
          kind: ImageStreamTag
          name: ruby-20-centos7:latest
      type: Docker
    triggers:
    - github:
        secret: jV5Ipwr7__4ae_sZG2Jm
      type: GitHub
    - generic:
        secret: ALNUyArydLb22JqdXYIb
      type: Generic
    - imageChange:
        lastTriggeredImageID: openshift/ruby-20-centos7:latest
      type: ImageChange
  status:
    lastVersion: 1
```

8. Observe that the current configuration points at the **`openshift/ruby-hello-world`** repository.
    - Because you forked this repository earlier, you can now repoint your configuration.

## 1.3. Repoint Your Configuration

1. Run **`oc edit`** to repoint the configuration.

```
[marina@master00~]$  oc edit bc ruby-hello-world
```

    a. Change the **`uri`** reference to match the name of your GitHub repository, which is based in part on your GitHub username: **`https://github.com/GitHubUsername/ruby-hello-world`**.

> Replace **`GitHubUsername`** with your actual GitHub username. For

example, if your GitHub username is `jeandeaux`, the name of your GitHub repository is `'https://github.com/jeandeaux/ruby-hello-world`.

b. Save and exit `vi` by typing **:wq**.

There are other ways to achieve this outcome, this way is used to cover the `oc edit` and the `oc start-build` commands.

2. Run `oc get buildconfig ruby-hello-world -o yaml` again. Notice that `uri` has been updated.
3. Run `oc get builds` to check if the new build has started:

```
[marina@master00~]$ oc get builds
```

If the build has not started yet, you can start it yourself and then follow `build-log`:

```
[marina@master00~]$ oc get bc
NAME                TYPE      SOURCE
ruby-hello-world    Docker    https://github.com/YOURUSERNAME/ruby-hello-world

[marina@master00~]$ oc start-build ruby-hello-world
ruby-hello-world-2

[marina@master00~]$ oc get builds -w
NAME                  TYPE      FROM  STATUS     STARTED              DURATION
ruby-hello-world-1    Source    Git   Complete   16 minutes ago   4m25s
ruby-hello-world-2    Source    Git   Complete   About a minute ago   1m46s

[marina@master00~]$ oc logs -f bc/ruby-hello-world
I0709 23:41:08.493756       1 docker.go:69] Starting Docker build from
justanother1/ruby-hello-world-7 BuildConfig ...
I0709 23:41:08.508448       1 tar.go:133] Adding to tar: /tmp/docker-
build062004796/.gitignore as .gitignore
I0709 23:41:08.509588       1 tar.go:133] Adding to tar: /tmp/docker-
build062004796/.sti/bin/README as .sti/bin/README
I0709 23:41:08.509953       1 tar.go:133] Adding to tar: /tmp/docker-
build062004796/.sti/environment as .sti/environment
I0709 23:41:08.510183       1 tar.go:133] Adding to tar: /tmp/docker-
build062004796/Dockerfile as Dockerfile
I0709 23:41:08.510548       1 tar.go:133] Adding to tar: /tmp/docker-
build062004796/Gemfile as Gemfile
.......
Cropped Output
.......
```

4. Search for the available `mysql` applications (templates):

```
[marina@master00-82bc ~]$ oc new-app --search mysql
Templates (oc new-app --template=<template>)
```

```
-----
mysql-persistent
  Project: openshift
  MySQL database service, with persistent storage. Scaling to more than one replica is
not supported
mysql-ephemeral
  Project: openshift
  MySQL database service, without persistent storage. WARNING: Any data stored will be
lost upon pod destruction. Only use this template for testing
eap64-mysql-s2i
  Project: openshift
  Application template for EAP 6 MySQL applications built using S2I.
jws30-tomcat7-mysql-persistent-s2i
  Project: openshift
  Application template for JWS MySQL applications with persistent storage built using
S2I.
jws30-tomcat8-mysql-s2i
  Project: openshift
  Application template for JWS MySQL applications built using S2I.
jws30-tomcat7-mysql-s2i
  Project: openshift
  Application template for JWS MySQL applications built using S2I.
cakephp-mysql-example
  Project: openshift
  An example CakePHP application with a MySQL database
dancer-mysql-example
  Project: openshift
  An example Dancer application with a MySQL database
jws30-tomcat8-mysql-persistent-s2i
  Project: openshift
  Application template for JWS MySQL applications with persistent storage built using
S2I.
eap64-mysql-persistent-s2i
  Project: openshift
  Application template for EAP 6 MySQL applications with persistent storage built using
S2I.

Image streams (oc new-app --image-stream=<image-stream> [--code=<source>])
-----
mysql
  Project: openshift
  Tags:    5.5, 5.6, latest

Docker images (oc new-app --docker-image=<docker-image> [--code=<source>])
-----
mysql
  Registry: Docker Hub
  Tags:     latest
```

5. Create the **`database`** application by running **`oc new-app`**:

```
[marina@master00~]$ oc new-app --template=mysql-ephemeral \
                    --
param=MYSQL_USER=mysqluser,MYSQL_PASSWORD=redhat,MYSQL_DATABASE=mydb,DATABASE_SERVICE_NA
ME=database
```

6. Verify that your values were processed correctly:

```
[marina@master00~]$ oc env dc/database --list
```

```
# deploymentconfigs database, container mysql
MYSQL_USER=mysqluser
MYSQL_PASSWORD=redhat
MYSQL_DATABASE=mydb
```

7. You must redeploy your front end so that it checks for the database again. You can either delete just the pod, or you can redeploy the application:

```
[marina@master00-GUID ~]$ oc deploy ruby-hello-world --latest
```

8. You can see the logs for your latest deployment if you use the `oc logs` command this way:

```
[marina@master00~]$  oc logs -f dc/ruby-hello-world
I1222 01:54:45.485814       1 deployer.go:198] Deploying from lifecycle/ruby-hello-
world-3 to lifecycle/ruby-hello-world-4 (replicas: 1)
I1222 01:54:46.913895       1 rolling.go:232] RollingUpdater: Continuing update with
existing controller ruby-hello-world-4.
I1222 01:54:47.019320       1 rolling.go:232] RollingUpdater: Scaling up ruby-hello-
world-4 from 0 to 1, scaling down ruby-hello-world-3 from 1 to 0 (keep 0 pods available,
don't exceed 2 pods)
I1222 01:54:47.020399       1 rolling.go:232] RollingUpdater: Scaling ruby-hello-world-4
up to 1
I1222 01:54:51.372703       1 rolling.go:232] RollingUpdater: Scaling ruby-hello-world-3
down to 0
```

# 2. Create and Use Web Hooks

With webhooks, you can integrate external systems into your OpenShift Enterprise environment so that they can start OpenShift Enterprise builds. Generally speaking, you make code changes and update the code repository, after which a process hits OpenShift Enterprise's webhook URL to start a build with the new code.

## 2.1. Create a Webhook

Your GitHub account can configure a webhook whenever you push a commit to a specific branch.

1. Find the webhook URL:
   a. Go to the web console.
   b. Navigate to your project.

c. Click **Browse** and then click **Builds**.

- Two webhook URLs are displayed.

2. Copy the generic URL, which looks like this:

```
https://master00-
GUID.oslab.opentlc.com:8443/osapi/v1/namespaces/lifecycle/buildconfigs/ruby-hello-
world/webhooks/ALNUyArydLb22JqdXYIb/generic
```

3. Obtain the `secret` password from `buildconfig`:

```
[marina@master00~]$ oc get bc ruby-hello-world -o yaml
```

- The output looks similar to the following.
- Note the `secret` value in your configuration in Git.

```
... Cropped Output ...
  triggers:
  - github:
      secret: xTah2lioO2Bz9JZT9dPf
    type: GitHub
  - generic:
      secret: B5h3ARS88HD7S3LOcbRZ
    type: Generic
... Cropped Output ...
```

4. In the GitHub repository, which you forked earlier, go to **Settings → Webhooks and Services**.
5. Paste the URL that you copied from the OpenShift Enterprise UI into the **Payload URL** field.
6. Fill in the `secret` field and disable SSL verification.
7. Click **Add Webhook**.

## 2.2. Test Your Webhook

To test your webhook, revise the code, commit, and then push the change into the Git repository. Do the following:

Alternatively, you can test the webhook the usual way by cloning your repository locally, making the required changes, and pushing them to the repository.

1. Go to your forked repository (`https://github.com/GitHubUsername/ruby-hello-world`) and find the `main.erb` file in the `views` folder.
   - You can edit files in the GitHub web UI.
2. Change this HTML code--

```
<div class="page-header" align=center>
  <h1> Welcome to an OpenShift v3 Demo App! </h1>
</div>
```

--to read as follows (including the deliberately misspelled `crustom`):

```
<div class="page-header" align=center>
  <h1> This is my crustom demo! </h1>
</div>
```

3. Commit the change to the repository.
4. Check if a build has started.

If another build is already running, this latest build may fail because both builds are pushing to the registry. Either run `oc delete build` to stop the earlier build or `oc start-build` to restart the failed build.

5. Log in as `marina` and check the web UI to verify that the build is running.
6. Wait for the build to complete. It can take a minute for your service endpoint to update.
7. Use your browser to go to the application at `http://ruby-hello-world.lifecycle.cloudapps-GUID.oslab.opentlc.com/`.
   - The output includes the deliberately misspelled `crustom`.
   - If you try to access the application before the update is complete, you may see a `503` error.

## 2.3. Roll Back Your Application

Because you failed to properly test your application and your typo made it into production, you must revert to the previous version of your application.

1. Log in to the web console as `marina`.
2. Locate the **Deployments** section of the **Browse** menu.
   - Two deployments are at your front end: `1` and `2`.

Alternatively, view this information from the CLI:

```
[marina@master00~]$ oc get replicationcontroller
```

The semantics of this syntax state that `DeploymentConfig` ensures that `ReplicationController` is created to manage the deployment of the built `Image` from `ImageStream`.

3. From the CLI, roll back the deployment:

a. Determine which builds are available:

```
[marina@master00~] oc get builds
```

b. Choose a deployment and see what a rollback to `ruby-hello-world-X` would look like:

```
[marina@master00~]$ oc rollback ruby-hello-world-X --dry-run # X is your desired
deployment
Name:           ruby-hello-world
Created:        39 minutes ago
Labels:         <none>
Latest Version: 9
Triggers:       Config, Image(ruby-hello-world@latest, auto=false)
Strategy:       Recreate
Template:
                        Selector:       deploymentconfig=ruby-hello-world
                        Replicas:       1
                        Containers:
                                NAME                    IMAGE
ENV
                                ruby-hello-world
172.30.119.73:5000/lifecycle/ruby-hello-
world@sha256:fcc9ce95e503429926dbe9e0cde304e0a0de19483e1cb79acada7334d7eb2504
MYSQL_DATABASE=mydb,MYSQL_PASSWORD=redhat,MYSQL_USER=root
Latest Deployment:      <none>
```

- From the above output, you can see that you can go ahead with the rollback.

c. Roll back the deployment:

```
[marina@master00~]$ oc rollback ruby-hello-world-X # X is your desired deployment
#oc get9 rolled back to ruby-hello-world-X
Warning: the following images triggers were disabled: ruby-hello-world
  You can re-enable them with: oc deploy ruby-hello-world --enable-triggers
```

4. Click the **Browse** tab of your project and note that you have a new pod in the **Pods** section.
5. After a few minutes, go back to the application in your browser.
   - The old "Welcome . . ." message is displayed.

## 2.4. Roll Your Application Forward

To roll forward (activate) the typo-enabled application:

```
[marina@master00~]$ oc rollback ruby-hello-world-X # X is your desired deployment
#11 rolled back to ruby-hello-world-X
Warning: the following images triggers were disabled: ruby-hello-world
  You can re-enable them with: oc deploy ruby-hello-world --enable-triggers
```

Build Version: 2.2.1 : Last updated 2016-01-19 02:58:45 EST