Microsoft

OPENSHIFT on Azure

Khaled Elbedri
Technology Solutions Professional - Global
Black Belt, Open Source Software
12/17/2016

TABLE OF CONTENTS

Objectives and initial setup 3

Introduction to openshift 4

EXERCISE-1: Deploy Openshift on azure 5

EXERCISE-2: Create and manage projects 11

EXERCISE-3: Create and manage Applications 13

EXERCISE-4: Configuring automated builds 17

EXERCISE-5: Continuous deployment 19

End the lab 23

References 23

- Useful links 23
- Redhat and Microsoft partnership 23

Figure 1: OpenShift capabilities and characteristics 4
Figure 2: OpenShift on Azure 5
Figure 3: OpenShift ARM template resources 6
Figure 4: Bash on Windows 6

OBJECTIVES AND INITIAL SETUP

This document describes the steps necessary to deploy and manage OpenShift.orig (Red Hat OpenShift container platform) environment on Azure. OpenShift.orig is the upstream project and is a test bed incubator for Red Hat OpenShift container platform. The lab is based on OpenShift version 3.3.

You don't need a Red Hat subscription to perform the lab instructions. But you will need a valid Azure account.

- Create your free azure account (https://azure.microsoft.com/en-us/free/), today.
- You need to have an account on GitHub, If you don't have one create a free account (https://github.com/).
- If you are using Windows 10, you can install Bash shell on Ubuntu on Windows (http://www.windowscentral.com/how-install-bash-shell-command-line-windows-10). To install Azure CLI, download and install the latest Node.js and npm for Ubuntu: (https://nodejs.org/en/download/package-manager/#debian-and-ubuntu-based-linux-distributions). Then, follow the instructions (**Option-1**): https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/
- If you are using MAC or another windows version, install Azure CLI, following (**Option-2)**: https://azure.microsoft.com/en-us/documentation/articles/xplat-cli-install/
- [Optional] Download and install Visual Studio Code: https://code.visualstudio.com/
- If you have Azure CLI, already, make sure you are running the latest release if not, upgrade your current install.

The Lab exercises cover:

- Introduction to OpenShift3
- Deployment of OpenShit.orig on Azure
- Creating and managing OpenShift projects on azure
- Creating and managing OpenShift applications on Azure
- Automating builds with Linux Containers on Azure.

INTRODUCTION TO OPENSHIFT

OpenShift containers platform is Red Hat's Platform-as-a-Service (PaaS) that allows developers to quickly develop, host, and scale applications in a cloud environment.

Microsoft and Red Hat have signed a partnership that includes support to run Red Hat OpenShift on Microsoft Azure and eventually Azure Stack.

OpenShift offers multiple access modes including: developer CLI, admin CLI, web console and IDE plugins. Click2cloud is a plugin that allows Visual studio to deploy code to OpenShift, directly.

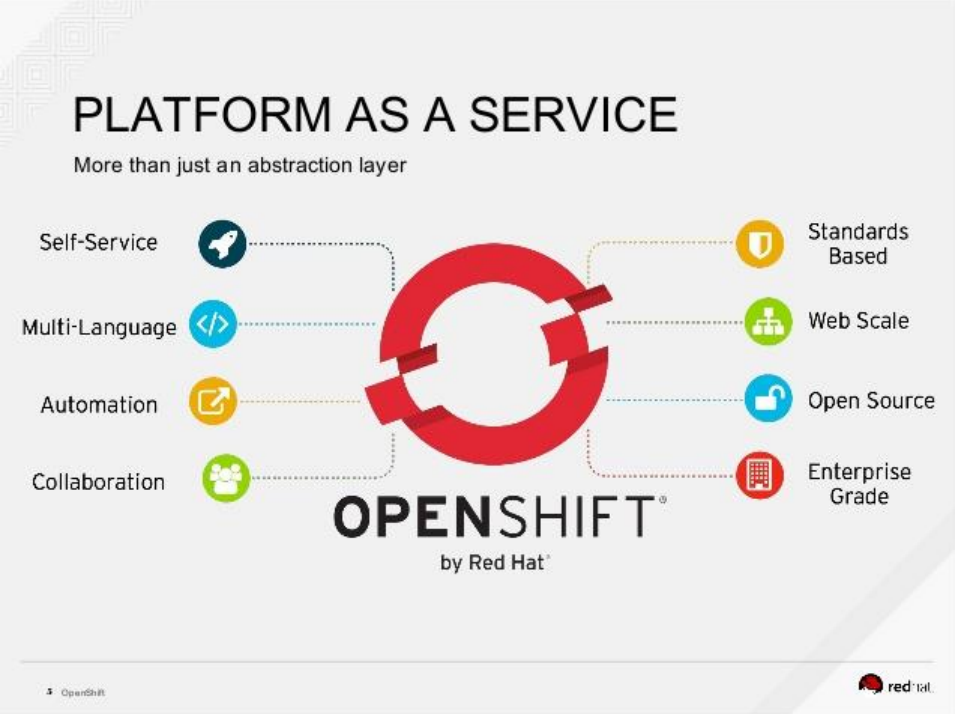

Figure 1: OpenShift capabilities and characteristics

EXERCISE-1: DEPLOY OPENSHIFT ON AZURE

OpenShift, leverages multiple Azure services such as VM *scale sets*, VM *extensions*, and *Key vaults* to provide an Enterprise grade offering for customers who would like to containerize and manage their applications, without investing long time and hard effort configuring and integrating various tools.

OpenShift offers another alternative to multiple CaaS (container as a service) solutions available on Azure, such as *Azure container service* and *Pivotel* from *CloudFoundry*.

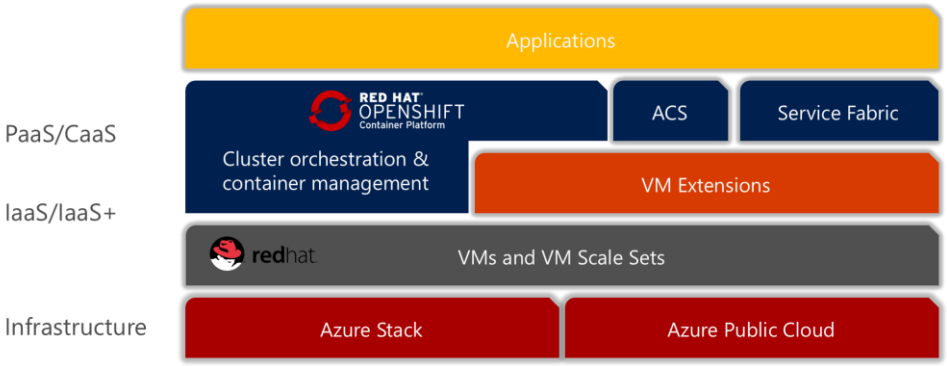

Figure 2: OpenShift on Azure

1. OpenShift is available as an Azure Resource Manager quick deploy template at https://github.com/khaled99b/azure-openshift. Click on the *Visualize* button and navigate the Azure resources that will be deployed by the template.

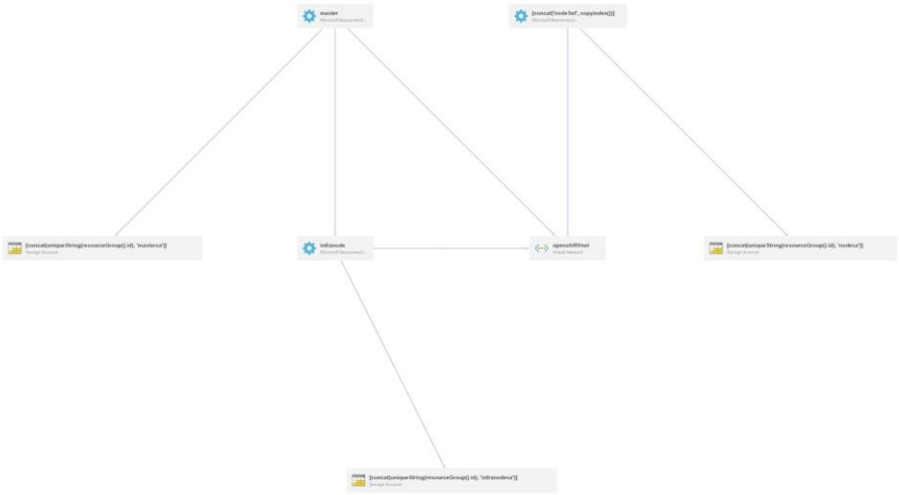

Figure 3: OpenShift ARM template resources

Windows10 allows to install and run Linux Bash shell, natively. We will leverage this new feature to communicate with Azure using the Linux CLI and Bash. Your local windows machine has been pre-configured with Bash and Azure CLI.

2. From the desktop of your local windows machine, launch "Bash on Ubuntu on Windows".

Figure 4: Bash on Windows

3. From the open bash terminal, use the CLI to login to your Azure account and follow the instructions.

```
[local]# azure login
```

4. Make sure the Azure CLI is using Resource Manager mode.

```
[local]# azure config mode arm
info:    Executing command config mode
info:    New mode is arm
info:    config mode command OK
```

5. List available account subscriptions and note the default one (Marked with the keyword *true*)

```
[local]# azure account list
```

6. Create an *ssh* keypair with a blank passphrase using *Bash*.

```
[local]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key
(/home/azureuser/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in
/home/azureuser/.ssh/id_rsa.
Your public key has been saved in
/home/azureuser/.ssh/id_rsa.pub.
The key fingerprint is:
... output omitted ...
```

7. To deploy from the template (https://github.com/khaled99b/azure-openshift), you have multiple options. You could use the graphical or command line interface:

   NB: Some parameters need to be universally unique. To avoid hitting existing names, you will need to replace the string ***CHANGEME*** by the string indicated in the up right corner of your local environment interface.

   NB: Make sure to change the instance size to Standard_DS2 to avoid hitting CPU core limits.

Click on the link "*Deploy to Azure*" from the link in step-1. Fill the form with the required information as per the bellow table, carefully! Note, that, if you have to redeploy because of any error, you will have to delete and recreate the resource group.

The lab assumes a deployment in North Europe Azure region. You are free to change the region to your preferred one.

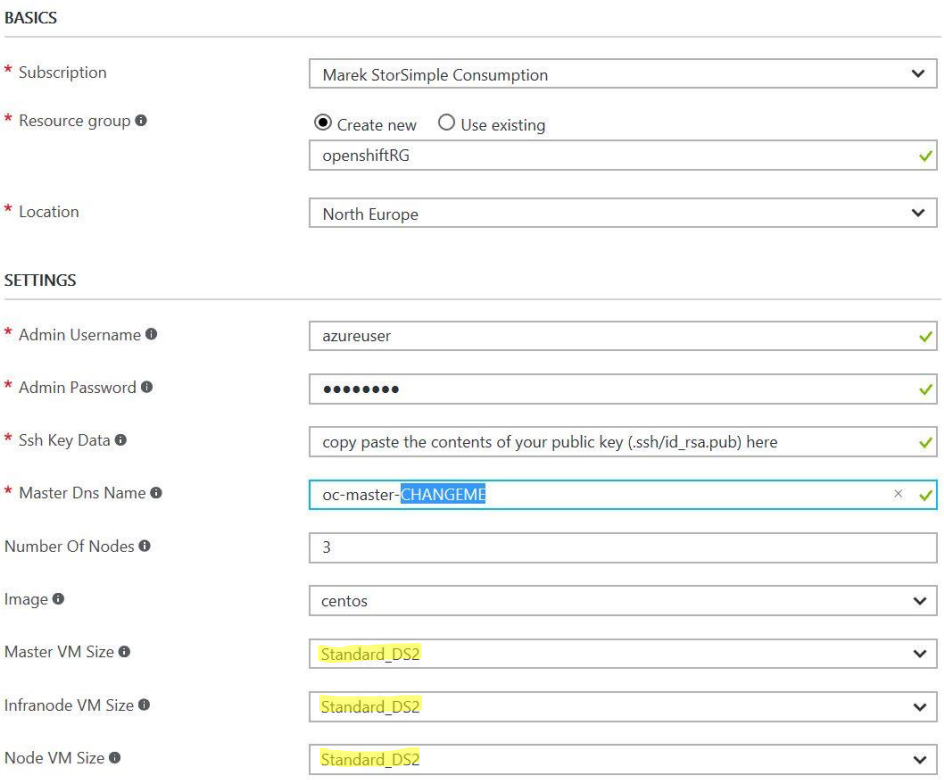

Figure 5: OpenShift ARM template deployment

Agree on the terms and conditions and purchase. The template will deploy the VMs infrastructure. The process will take around 20 minutes. From the Azure portal, click on the openshiftRG resource group and keep an eye on the progress of the deployment of the different resources.

You can leverage the waiting time to walk through the upcoming steps of the lab.

Once successfully finished. Navigate the newly created resource group in the Azure portal. Select Deployments -> OpenShiftDeployment and note the *OPENSHIFTMASTERPUBLICIPFQDN*

9. Ssh into the jumbox node and enable forwarding of the authentication agent connection.

```
[local-machine]# eval $(ssh-agent)
[local-machine]# ssh-add .ssh/id_rsa
[local-machine]# ssh -A -i
azureuser@OPENSHIFTJUMPBOXPUBLICIPFQDN
```

10. Install *Openshift* using *Ansible*

```
[jumpbox-node]#./openshift-install.sh
```

11. Escalate *azureuser*'s privileges to admin level in openshift

```
[jumbox-node]# sudo oadm policy add-cluster-role-to-user
cluster-admin azureuser
```

You have successfully deployed one Openshift (Kubernetes) master and three Openshift (Kubernetes) agents behind preconfigured load balancers, inside the same vNet (Azure subnet).

The following diagram explains the physical architecture of the deployed cluster.

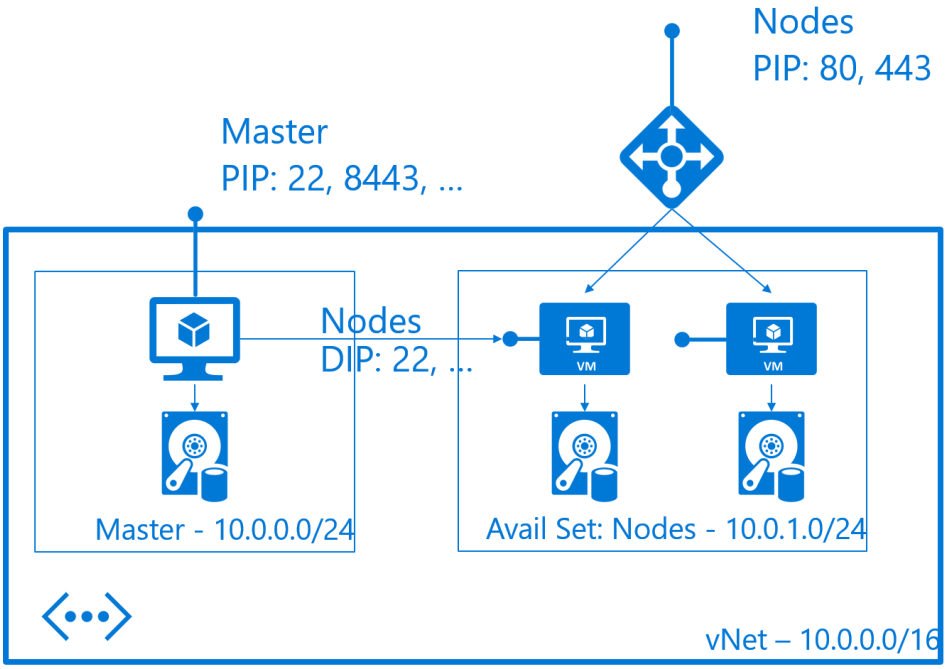

Figure 6: Deployment diagram

The next diagram, explains the role and tasks of the Openshift master/agents and the logical architecture of the solution.

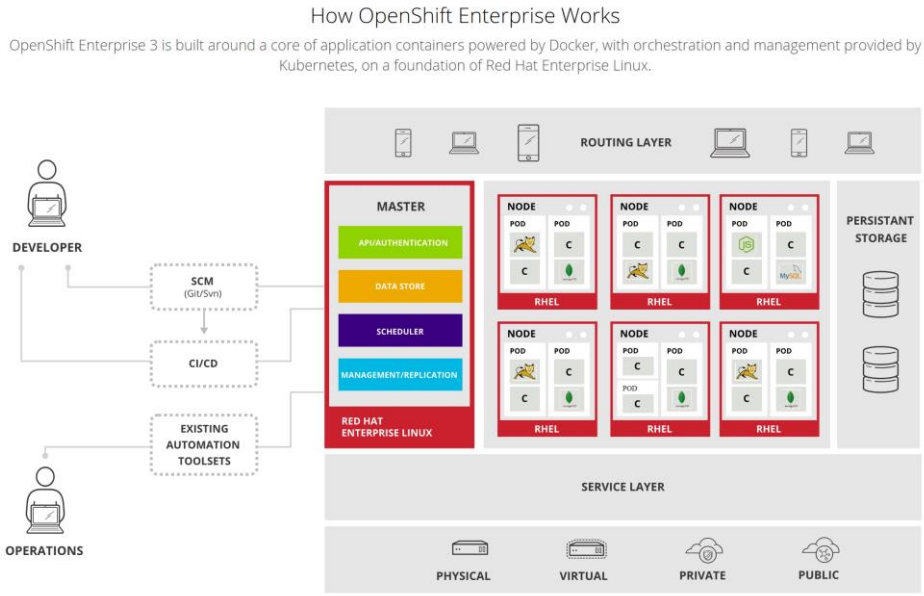

Figure 7: OpenShift logical architecture

EXERCISE-2: CREATE AND MANAGE PROJECTS

There are many ways to launch images within an OpenShift project. For the sake of simplicity, we will focus on the quickest and easiest method.

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

1. Login to your *github* account, or create one if you didn't.
2. Browse to *openshift/ruby-ex* repository and fork it into your *github* account

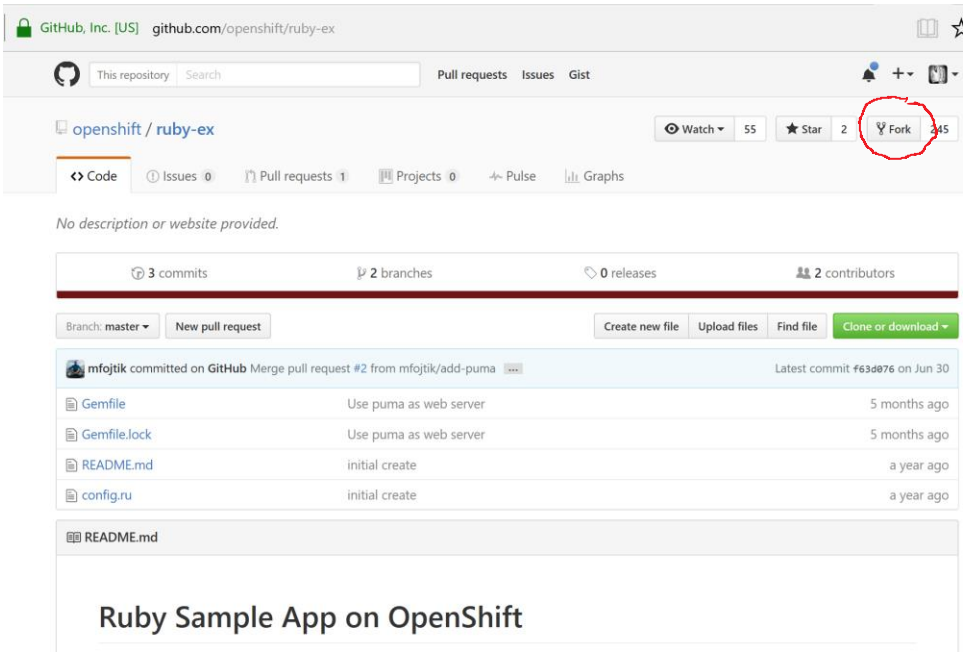

Figure 8: Fork ruby-ex sample application

3. From your browser, visit the OpenShift web console at *https://FQDN-master-node:8443*. The web site, uses a self-signed certificate, so if prompted, continue and ignore the browser warning.
4. Log in using your username and password.

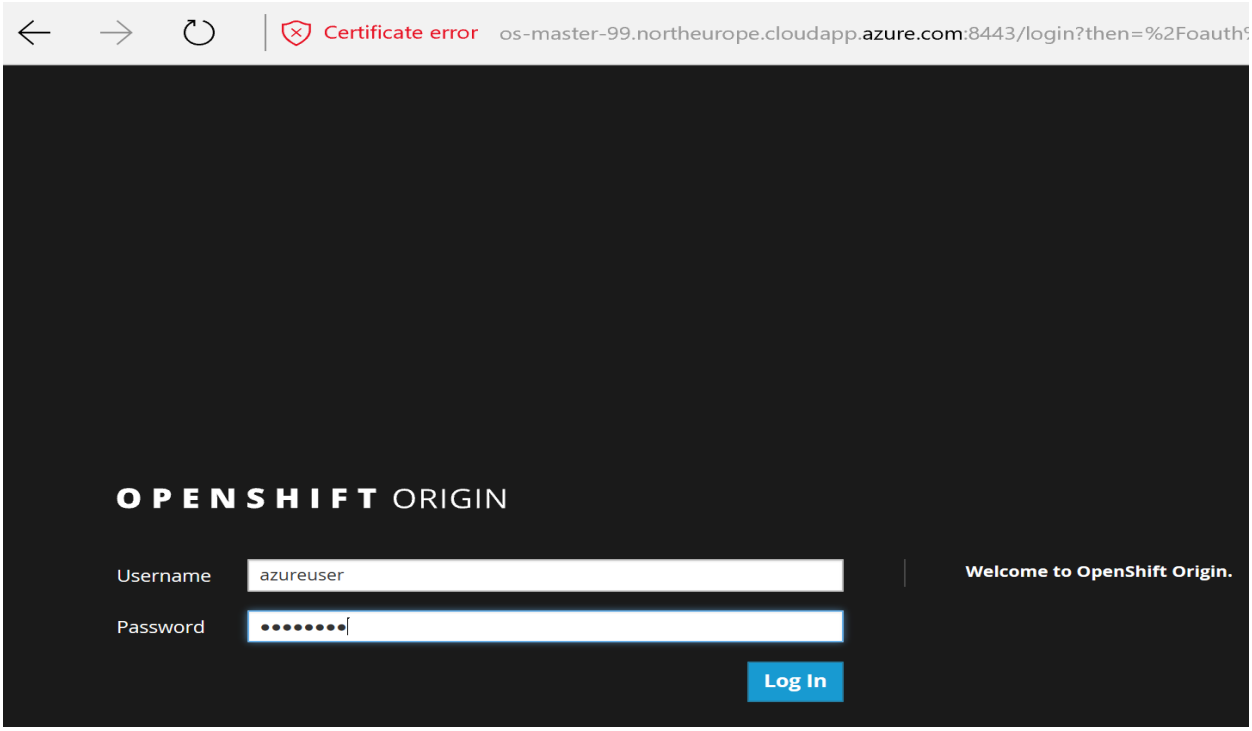

Figure 9: OepnShift login console

5. To create a new project, click **New Project**.
6. Type a unique name, display name, and description for the new project.
7. Click **Create**. The web console's welcome screen should start loading.

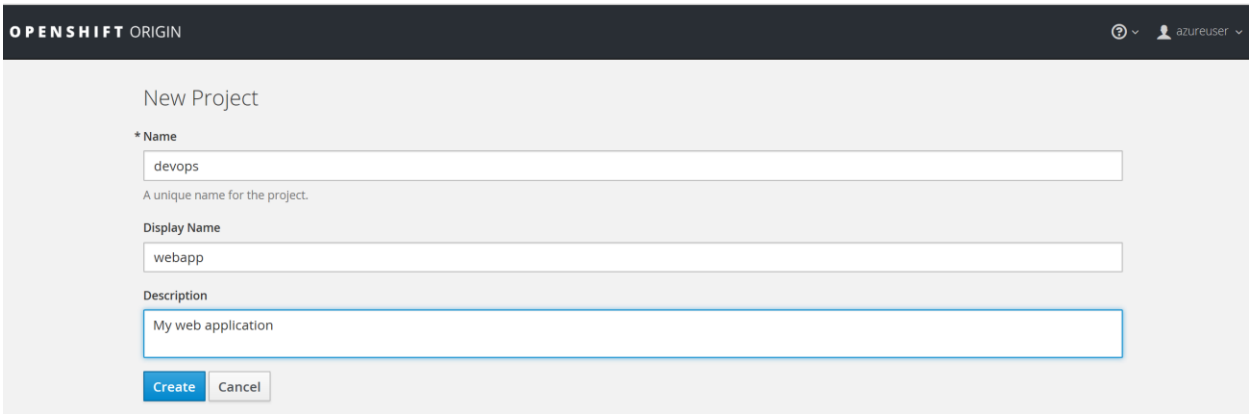

Figure 10: OepnShift new project

EXERCISE-3: CREATE AND MANAGE APPLICATIONS

The *Select Image* or *Template* page gives you the option to add an application to your project 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.

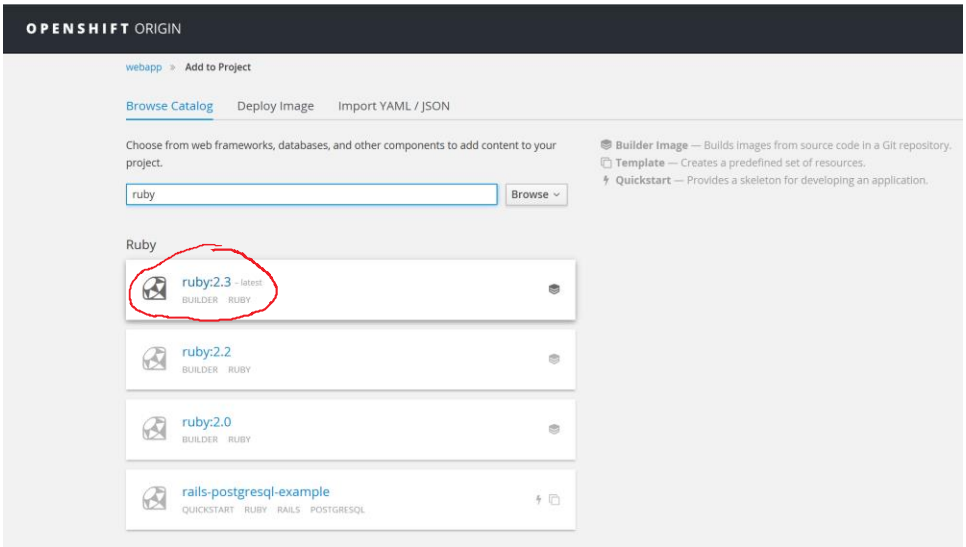

Figure 11: Create ruby builder

3. Click the **ruby:latest** builder image.
4. Type a **name** for your application, and specify the git repository URL you previously forked: https://github.com/<your_github_username>/ruby-ex.git.

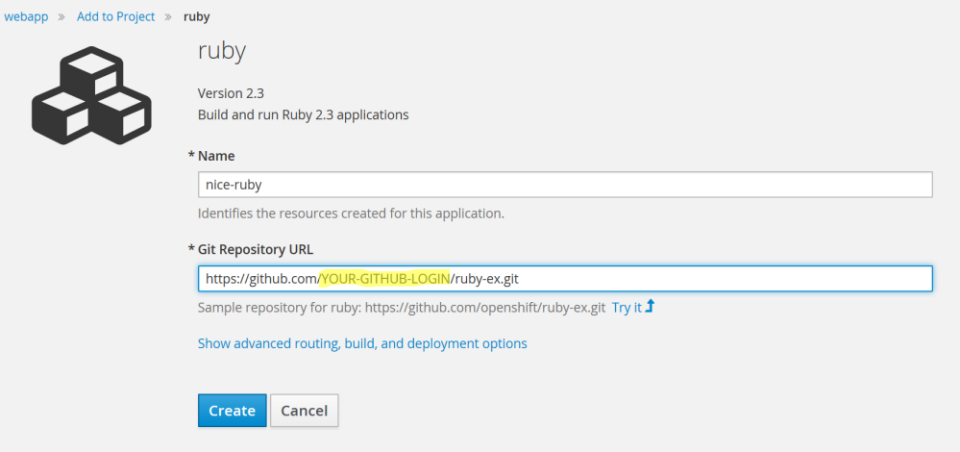

Figure 12: Build and run ruby

5. Optionally, click **Show advanced routing, build, and deployment options**. Explore the build configuration and other options and note that this example application automatically creates a route, *webhook* trigger, and builds change triggers. A *webhook* is an HTTP call-back triggered by a specific event.
6. Click **Create**. Creating your application might take some time. note the *payload url*, we will use it later to set a *webhook* in your *github* repository.

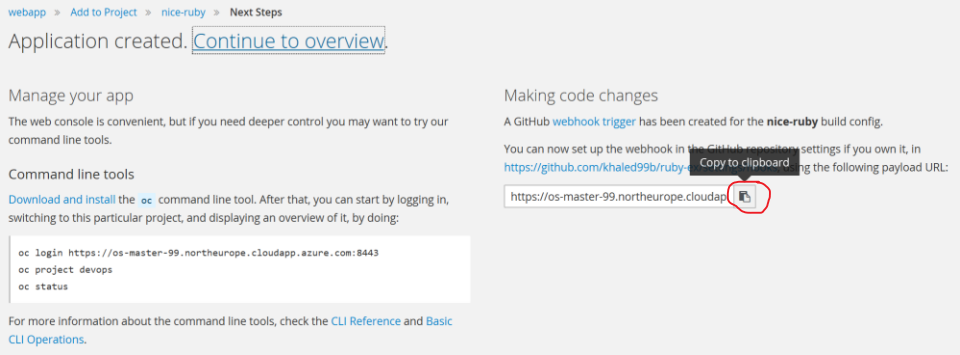

Figure 13: Create a ruby application

7. 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.

   OpenShift Origin leverages the Kubernetes concept of a pod, which is one or more containers deployed together on one host. A pod is the smallest compute unit that can be defined, deployed, and managed.

Pods are the rough equivalent of a machine instance (physical or virtual) to a container. Each pod is allocated its own internal IP address, therefore owning its entire port space. Containers within pods can share their local storage and networking.

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.

Pods have a lifecycle; they are defined, then they are assigned to run on a node, then they run until their container(s) exit or they are removed for some other reason. Pods, depending on policy and exit code, may be removed after exiting, or may be retained in order to enable access to the logs of their containers.

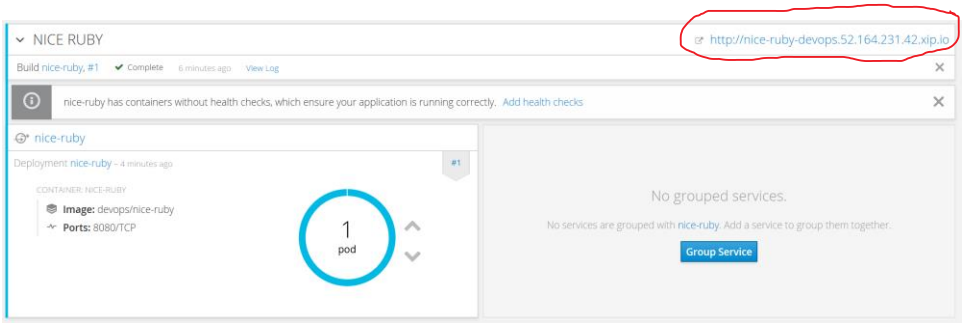

Figure 14: Ruby application pod

8. From the overview page, click the web address for the application in the up right corner. Verify that the web application is up and available.

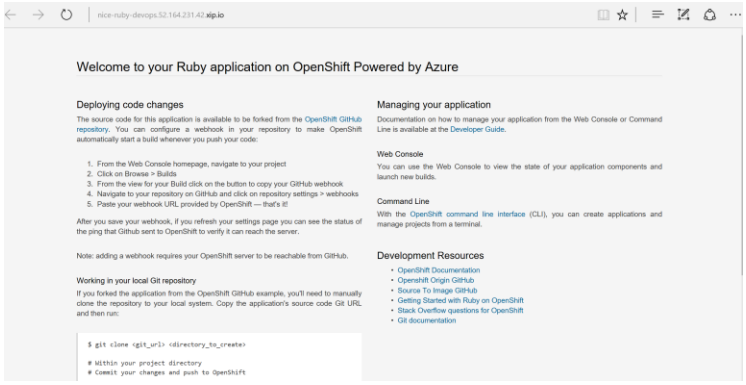

Figure 15: ruby sample application

9. Return to the *OpenShift* admin console. Browse to the project's overview page, and test scaling out and in your application by increasing or decreasing the number of *pods*, using the up and down arrow signs on the web console. Scale out the app into 3 pods and watch the progress.

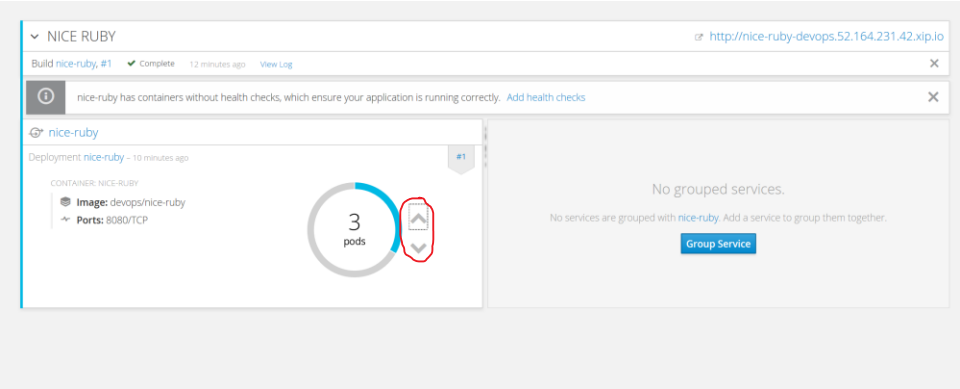

Figure 16: Scaling out the application

10. Browse to **Applications -> Pods**, and make sure 3 pods serving the same application are now up and running.

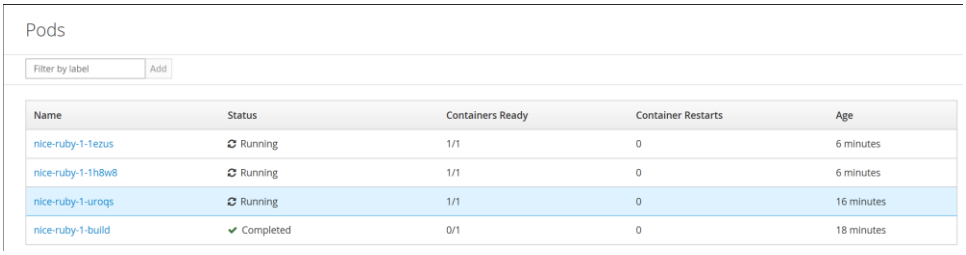

Figure 17: scaled out pods

EXERCISE-4: CONFIGURING AUTOMATED BUILDS

Since we forked the source code of the application from the OpenShift GitHub repository, we can use a *webhook* to automatically trigger a rebuild of the application whenever code changes are pushed to the 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.

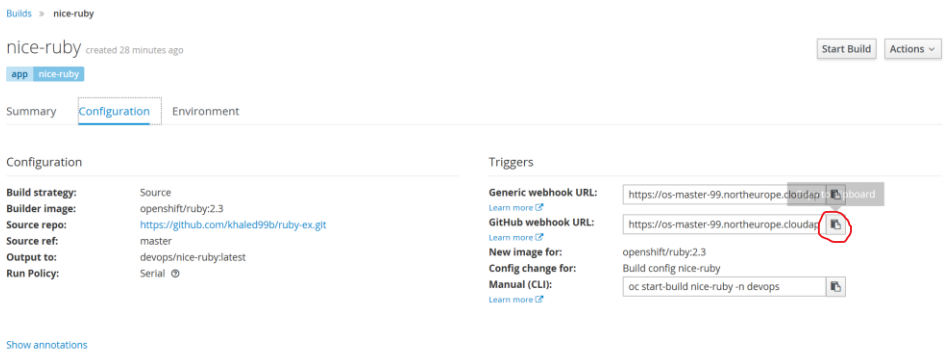

Figure 18: openshift github webhook url

5. Navigate to your forked repository on GitHub, then click **Settings**.
6. Click **Webhooks & Services** and Click **Add webhook**.

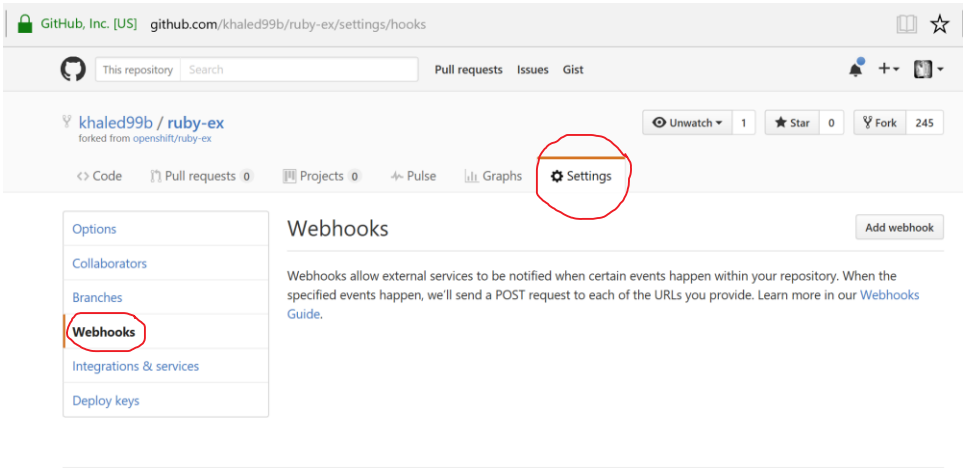

Figure 19: github webhook configuration

7. Paste your *webhook* URL into the **Payload URL** field.
8. Disable SSL verification and click **Add webhook** to save.

GitHub will now attempt to send a ping payload to your *OpenShift* 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.

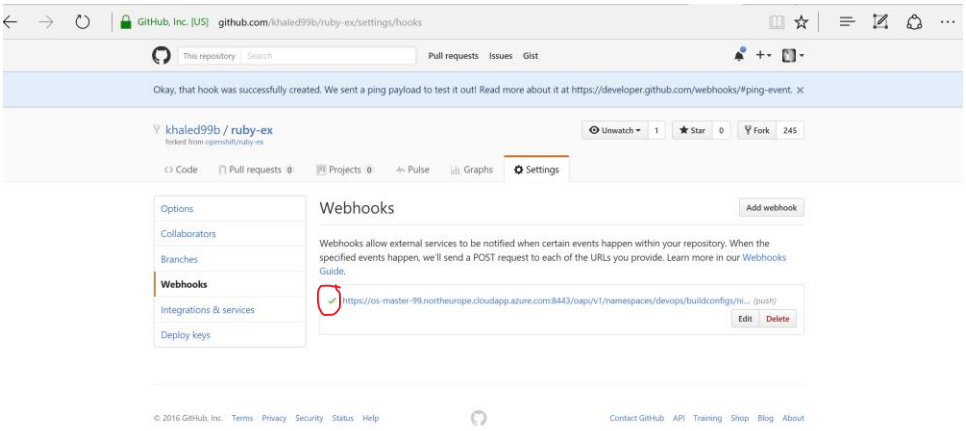

Figure 20: webhook ping

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

EXERCISE-5: CONTINUOUS DEPLOYMENT

In this section, we demonstrate one of the most powerful features of *OpenShift*. We will see how we can trigger a continuous deployment pipeline, just by committing code change to Github.

Once there is a code change, the Github *webhook* will trigger the build of a new container image that combines a blueprint image from the registry with the updated code and generate a new image. This feature is called ***S2I***, or source to image. Once the build finishes, *OpenShift* will automatically deploy the new application based on the new image. This capability enables multiple deployment strategies such as A/B testing, Rolling upgrades…

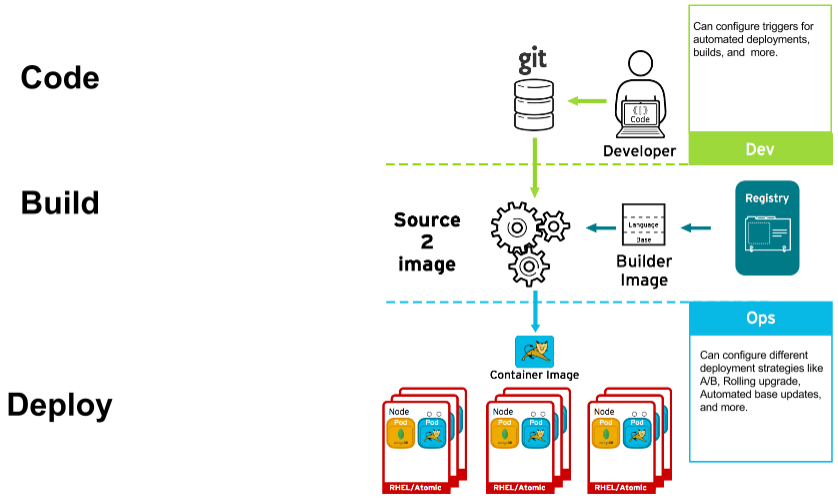

Figure 21: Continuous deployment pipeline

1. Install *Git* into your local machine

   PS: If you don't want to use *git*, you still can perform this exercise by moving to step-5 and editing the file *config.ru*, directly from the web interface of *github* and then go to step-8.

```
[local]# sudo apt-get install git
```

2. Create a "dev" folder and change into.

```
[local]# mkdir dev && cd dev
```

3. Clone the forked repository to your local system

```
[local dev]# git clone
https://github.com/<YourGithubUsername>/ruby-ex.git
```

4. Make sure your local *git* repository is referencing to your *ruby-ex git*, on *github*:

```
[local dev]# cd ruby-ex


[local ruby-ex]# git remote -v
```

5. On your local machine, use your preferred text editor to change the sample application's source for the file ***config.ru***

   Make a code change that will be visible from within your application. For example: on line 229, change the title to "Welcome to your Ruby application on OpenShift powered by Azure!", then save your changes.

6. Verify the working tree status

```
[local ruby-ex]# git status
```

7. Add config.ru content to the index, Commit the change in *git*, and push the change to your fork. You will need to authenticate with your *github* credentials

```
[local ruby-ex]# git add config.ru
[local ruby-ex]# git commit -m "simple message"
[local ruby-ex]# git status
[local ruby-ex]# git push
```

8. If your *webhook* is correctly configured, your application will immediately rebuild itself, based on your changes. Monitor the build from the graphical console. 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 handles the rest.

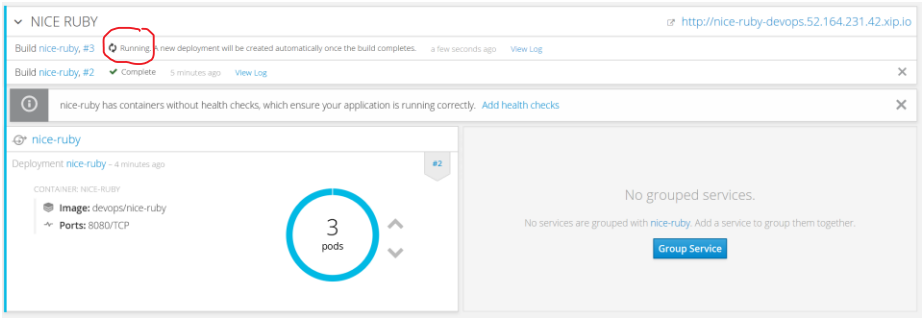

Figure 22: Automated build

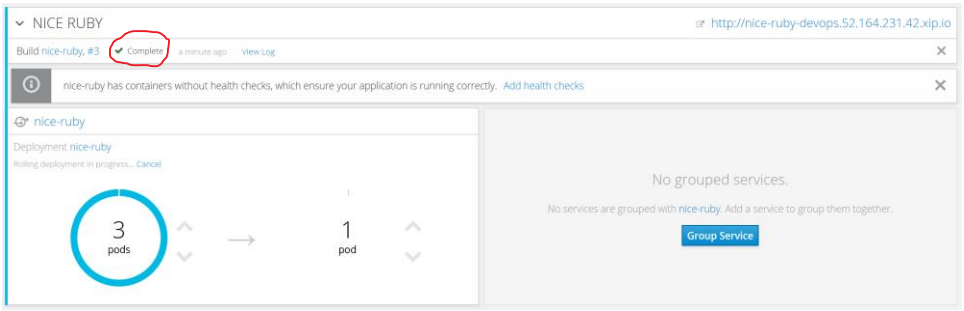

Figure 23: Automated deployment

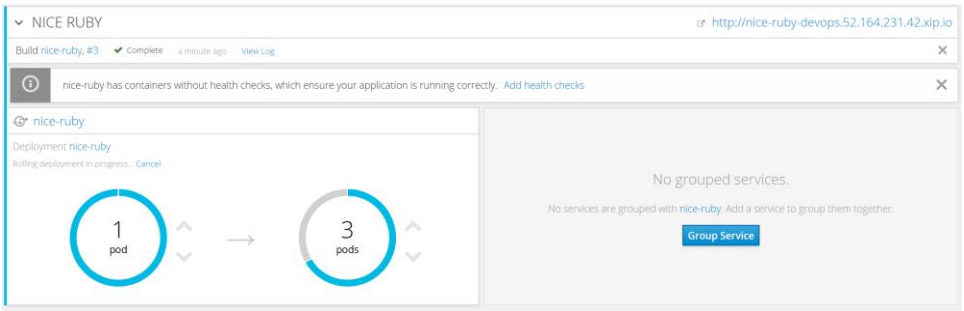

Figure 24: Automated deployment - progressive

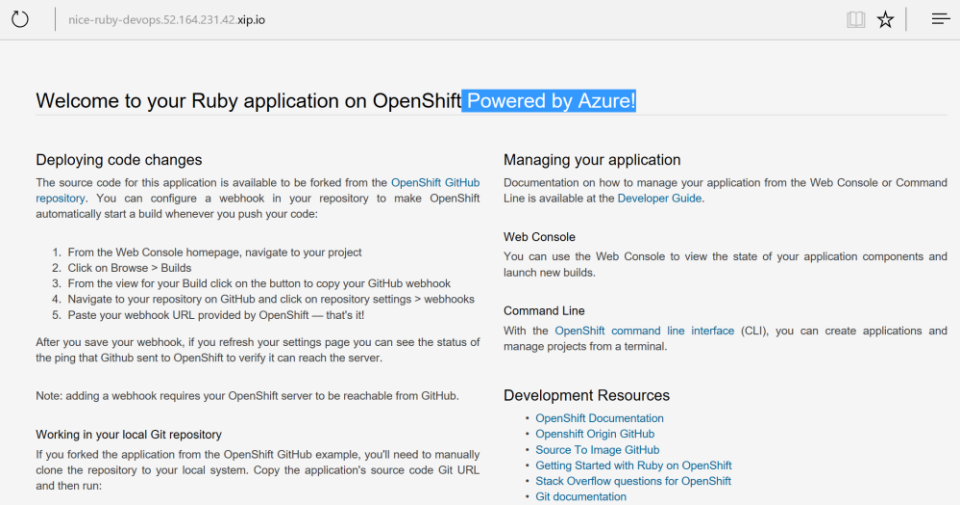

Figure 25: Updated application

9. 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:
    a. Click the **Browse** tab, then click **Builds**.
    b. Find your build, then click **Start Build**.

END THE LAB

To end the lab, simply delete the resource group *openshiftRG* from the Azure portal or from the Azure CLI. And delete the created *webhook* from your *git* repository.

```
[local]# azure group delete openshiftRG
```

REFERENCES

USEFUL LINKS

https://access.redhat.com/documentation/en/openshift-container-platform/

https://visualstudiogallery.msdn.microsoft.com/9a5b8b19-dadf-4b46-8712-527303d32231

http://open.microsoft.com/

https://github.com/microsoft/

REDHAT AND MICROSOFT PARTNERSHIP

http://openness.microsoft.com/2016/04/15/microsoft-red-hat-partnership-accelerating-partner-opportunities/

https://www.redhat.com/en/microsoft