Cloud 301

Optum

Student Lab Manual

Contents

This page intentionally left blank.

Lab: Getting Started with OpenShift Online

Overview
Time: 20 Minutes

Prerequisites:

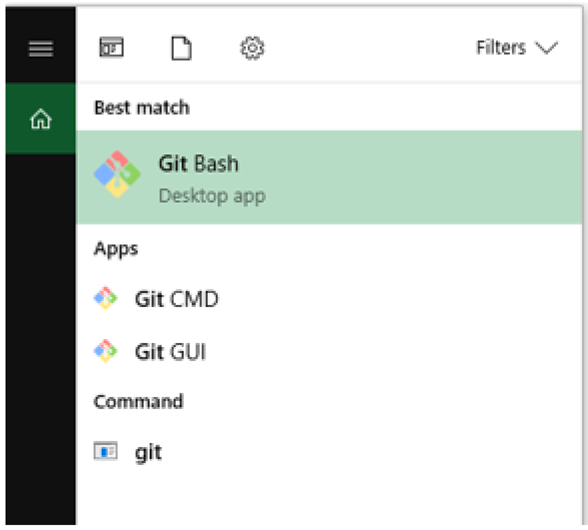
- Computer system with internet connectivity.
- GitHub account.
- Download Git software as described in Step 1.

In this lab, you will:

- Install Git software.
- Sign up for OpenShift Online starter plan.
- Get familiar with the OpenShift website.

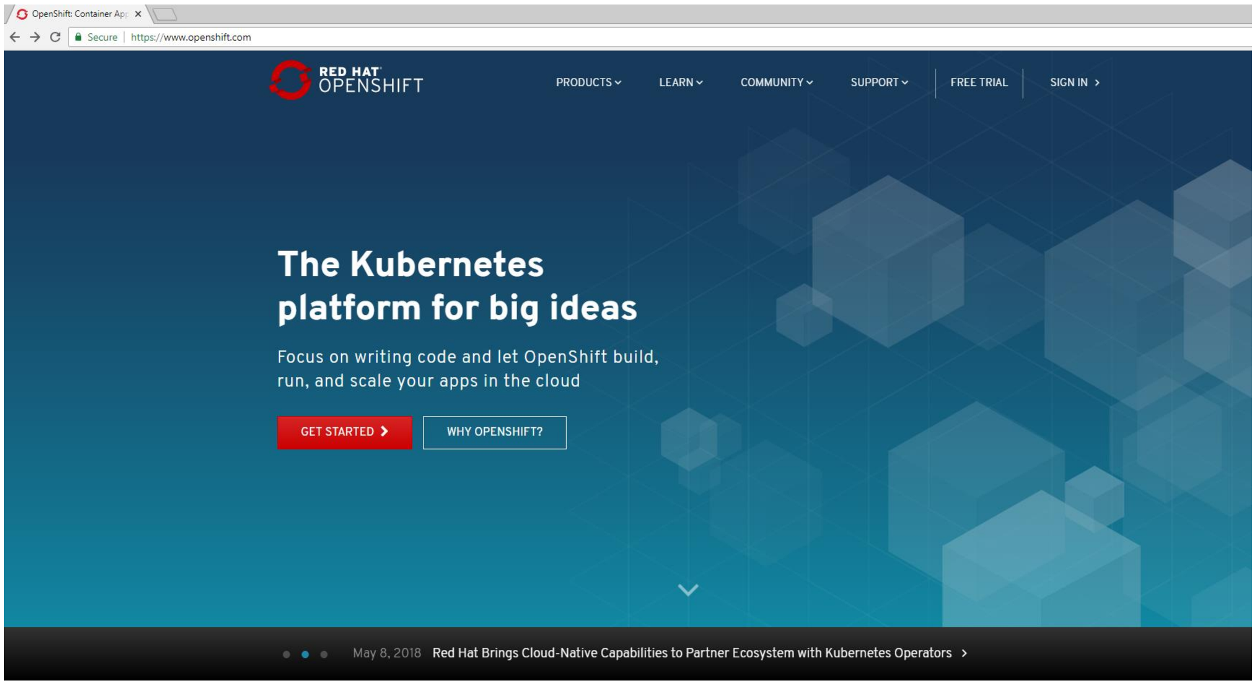
Step 1: Installing Git Software (if you do not have it already).

- Most Linux computers have Git installed, but it can be installed from your package manager.
- For MAC, Git is included in the Apple development tools or it can be downloaded.
- For Windows, download Git from the website.
- The download website for Git is: https://git-scm.com/downloads.
- After downloading the Git software, follow the installation prompts.
- If using firewall software, make sure it is configured to allow Git to access the internet.
- The name of the terminal on Windows is Git Bash. If you cannot locate the Git Bash software after installing, type "Git Bash" into the Windows search bar.



Step 2: Sign Up for OpenShift Online.

- Navigate your web browser to: **https://www.openshift.com.**
- Click on the red **Get Started** button as pictured.

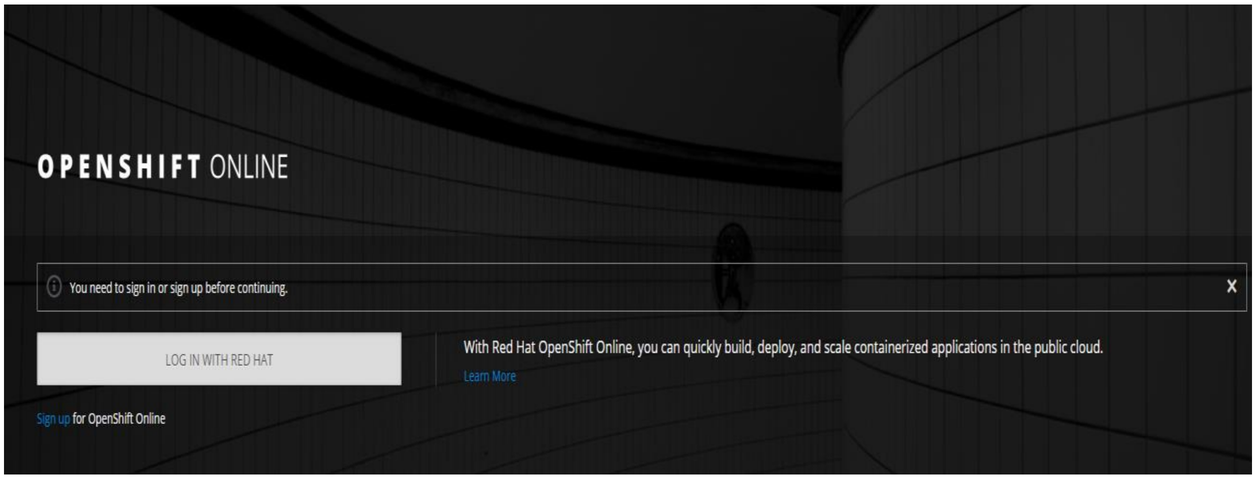


The Kubernetes platform for big ideas

Focus on writing code and let OpenShift build, run, and scale your apps in the cloud

May 8, 2018 Red Hat Brings Cloud-Native Capabilities to Partner Ecosystem with Kubernetes Operators

- Click the **red Create a Free account** button.
- On the next page, click the blue **Sign up** link in the left-lower corner.

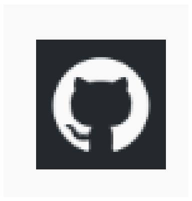


You need to sign in or sign up before continuing.

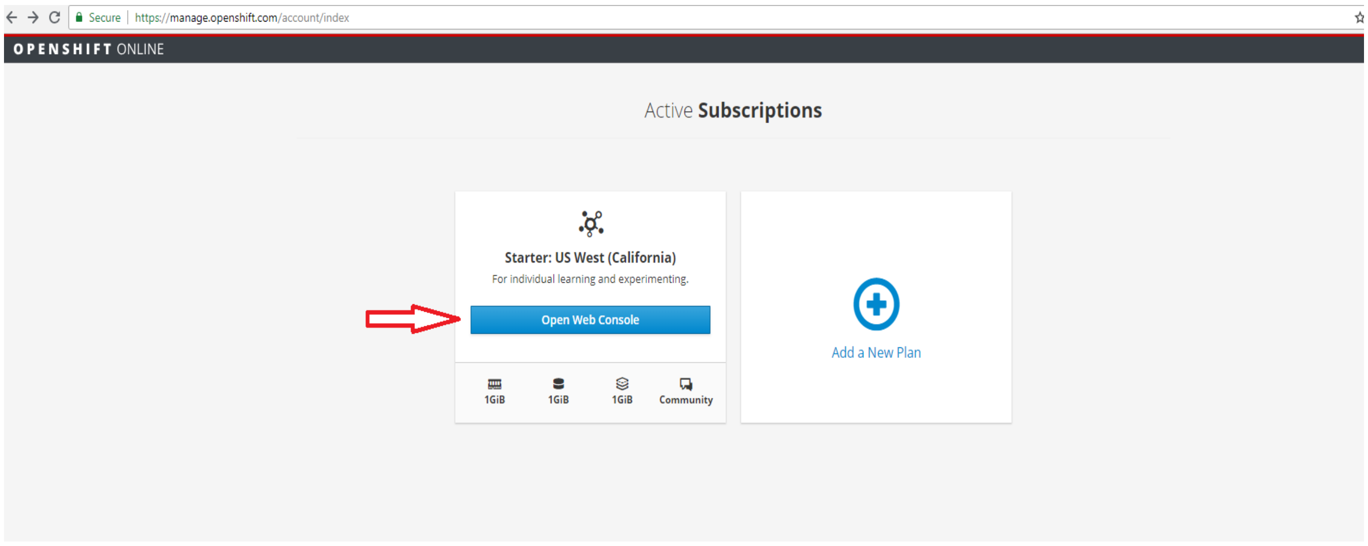
With Red Hat OpenShift Online, you can quickly build, deploy, and scale containerized applications in the public cloud.

Sign up for OpenShift Online

- Sign up or sign in:
  - If you have a Red Hat® account, you may use your Red Hat® user ID.
  - You may log in with your GitHub account by clicking the **GitHub** button at the bottom of the screen.

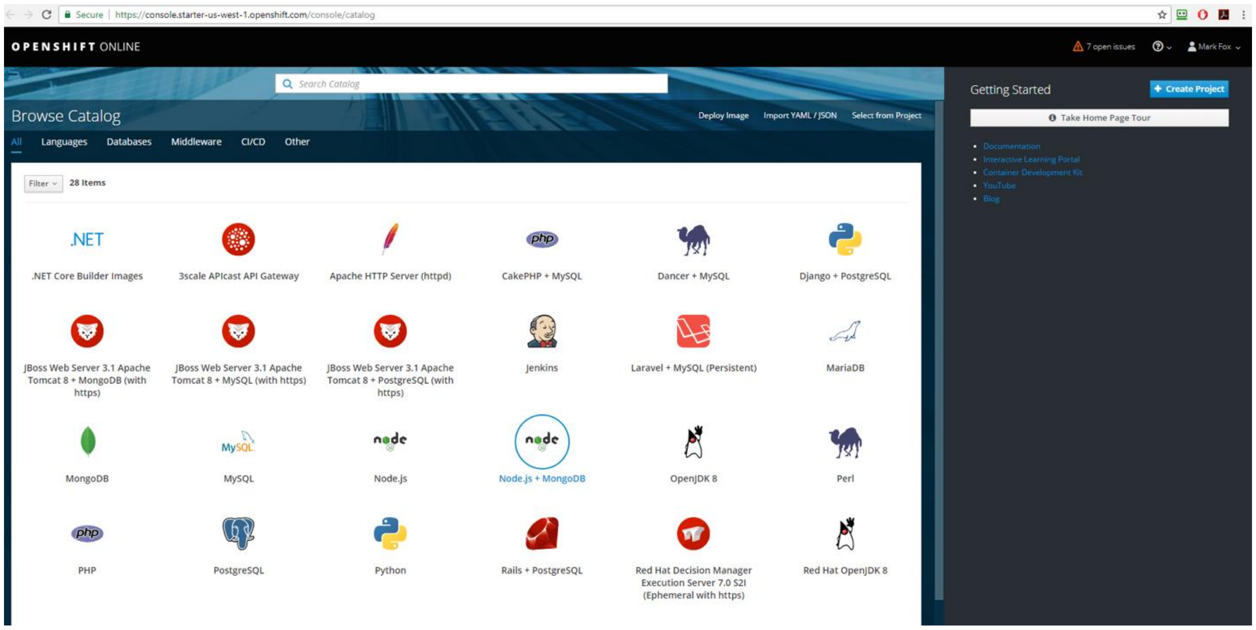


  - Or you may click on the **Create one Now** button to create a Red Hat® account.
- Once you have signed up for an account, chose the free starter plan and it will take you to this page below. Click the blue **Open Web Console** button.


Secure
https://manage.openshift.com/account/index
OPENSHIFT ONLINE
Active Subscriptions
Starter: US West (California)
For individual learning and experimenting.
Open Web Console
1GiB
1GiB
1GiB
Community
Add a New Plan

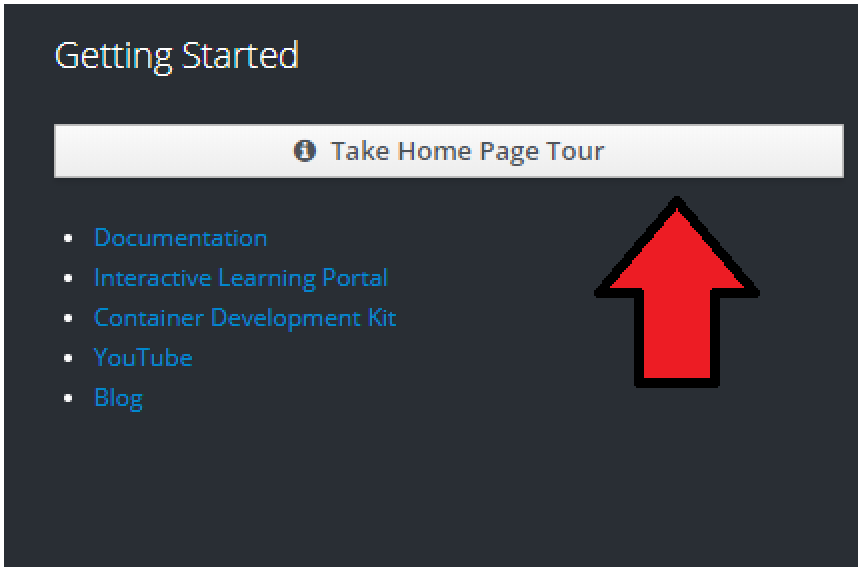
Step 3: Review the OpenShift Online Website.

- Once you click the blue button as above, you will see the web console.

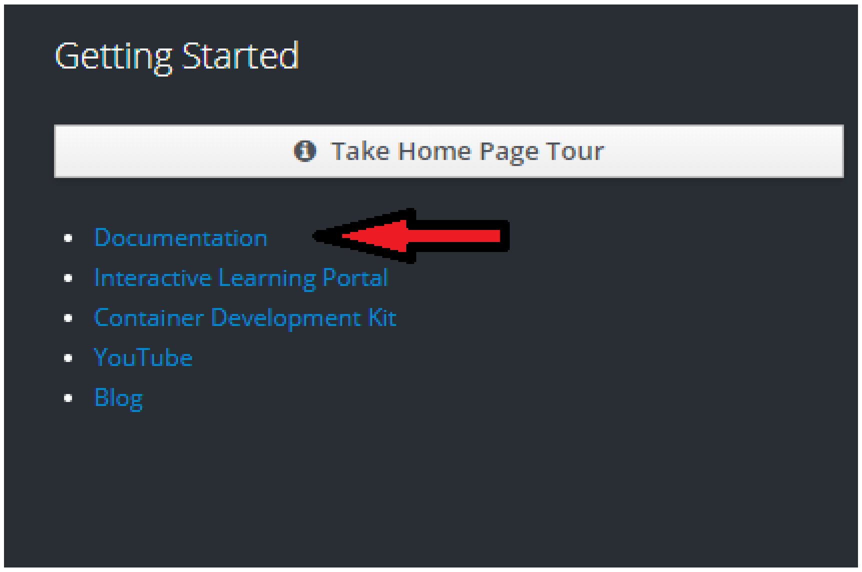


https://console.starter-us-west-1.openshift.com/console/catalog
OPENSHIFT ONLINE
7 open issues
Mark Fox
Search Catalog
Browse Catalog
Deploy Image
Import YAML / JSON
Select from Project
All
Languages
Databases
Middleware
CI/CD
Other
Filter
28 Items
.NET Core Builder Images
3scale APIcast API Gateway
Apache HTTP Server (httpd)
CakePHP + MySQL
Dancer + MySQL
Django + PostgreSQL
JBoss Web Server 3.1 Apache Tomcat 8 + MongoDB (with https)
JBoss Web Server 3.1 Apache Tomcat 8 + MySQL (with https)
JBoss Web Server 3.1 Apache Tomcat 8 + PostgreSQL (with https)
Jenkins
Laravel + MySQL (Persistent)
MariaDB
MongoDB
MySQL
Node.js
Node.js + MongoDB
OpenJDK 8
Perl
PHP
PostgreSQL
Python
Rails + PostgreSQL
Red Hat Decision Manager Execution Server 7.0 S2I (Ephemeral with https)
Red Hat OpenJDK 8
Getting Started
Create Project
Take Home Page Tour
Documentation
Interactive Learning Portal
Container Development Kit
YouTube
Blog

- Familiarize yourself with the web console by clicking the **Take Home Page Tour** button in the upper-right corner. This will introduce you to some of the parts of the web page.



- Click the blue **Documentation** link.



- The link to the documentation page is: https://docs.openshift.com/online/welcome/index.html.
- Click on the link on the page that says: **Check out a walkthrough on creating your first app**.
- Click on the **Basic Walkthrough** link and it will take you to this page: https://docs.openshift.com/online/getting_started/basic_walkthrough.html#getting-started-basic-walkthrough.
- Bookmark the link to use as a reference for the next lab.

Lab: OpenShift: Installing a Sample App

Overview
Time: 30 Minutes

Prerequisites:

- A computer system with Internet connectivity.
- A GitHub account.
- Completed Openshift Lab: *Getting Started with OpenShift Online*.
- A free Openshift Online account.
- Git installed locally on your system.

In this lab, you will:

- Fork an app on the GitHub repository.
- Use Git to clone the application on your local system.
- Create a project using the OpenShift web console.
- Install the application on the OpenShift online server.
- Configure automatic builds for your application.

Note:

- This lab is based on the OpenShift Basic Walkthrough web page but we will use a different application.
- Navigate to the Basic Walkthrough page below and keep it open for your reference: https://docs.openshift.com/online/getting_started/basic_walkthrough.html#getting-started-basic-walkthrough.
- The free starter plan for OpenShift Online allows you to only have one project at a time; so if you want to try another project later, you will need to delete the project installed in this tutorial.

Step 1: Select an App and Clone It on Your System.
Open up a separate web window/tab to **https://github.com/** and sign in using your GitHub credentials.

Navigate to the following URL: **https://github.com/elephantscale/hello-openshift-nodejs**. This will bring you to the *elephantscale/hello-openshift-nodejs* repository with an example Node.js application which will display a message on the site.

In the upper-right-hand corner of the GitHub repository, click on the Fork button. This will provide you with your own copy of the app in your GitHub account.

If using Windows, open the Git Bash console on your local computer as described in OpenShift Lab: *Getting Started with OpenShift Online*. If using Mac or Linux computers, you can use the terminal.

Enter the following commands into the Git Bash console or terminal to clone the example application to your local computer. Put your actual GitHub username in place of "username." (Do not type the $ sign.)

```
       bash  $ git clone https://github.com/username/hello-openshift-
nodejs  $ cd hello-openshift-nodejs
```

Step 2: Create a Project Using the OpenShift Web Console.
In a new window/tab, navigate to the OpenShift web console. The link is below. Sign in using your user ID and password if required. https://console.starter-us-west-1.openshift.com/console/catalog.

Click the blue **Create Project** link in the upper right of the screen.

Create a unique name for the project and type it in the **Name** field. For example, you could use "username-example" with no spaces or quotes. Replace username with your own username. Fill in brief details under the display name and description.

Click the blue **Create** button.

You should receive a small notification in the upper right of the screen saying that your project was created. After a couple of minutes, the name of your project should appear at the top right of the screen.

If you need to delete the project later, notice that you have that option if you click on the three white circles to the right of your project name. But don't delete it now.
Click on your project name -- in this case: "Example Project."

Step 3: Install the Application on the OpenShift Online Server.
Click the blue **Browse Catalog** button.

Under **Select an item to add to the current project**, click on **Languages**.
In the list of programming languages, click on **JavaScript**. Then click the **Node.js** button on the left.

A window will open that says "Nodejs." Click the blue **next** box.

A configuration page will open. You can use the latest version of Node.js.

– Type the application name as `hello-openshift-nodejs`

- Type the Git repository name as below but replace username with your actual GitHub user ID.

  `https://github.com/username/hello-openshift-nodejs.git`

Click the blue **Create** button. It may take a few minutes to complete.

Click the blue **Close** button.

Click the **Overview** button to view the status of the application.

It will take a couple of minutes to load the application on the OpenShift Online server, but the overview will show you the status of the app. When it is ready, there will be a web address of your application which you can click to see the application.

The application should show the text in your web browser with your unique web address.

Step 4: Configuring Your Application for Automatic Builds.

A GitHub webhook allows you to automatically rebuild your application on the OpenShift server whenever you update your application and push the changes to your GitHub repository.

From the project view, click on the **Builds** button in the upper-left part of the screen, then click the smaller **Builds** in the menu that opens up.

Next, click on the name of your application.

Click the **Configuration** tab.

Click the copy icon to the right of the GitHub webhook URL field. This will copy the URL to your clipboard.

Navigate your browser to your forked GitHub repository. If using the link below, replace "username" with your actual GitHub username: https://github.com/username/hello-openshift-nodejs.

Click the **Settings** button.

Click the **Webhook** tab on the left and then click the **Add webhook** button in the upper right.

Select the **Payload URL** field and press **Ctrl-V** to paste the URL you copied to the clipboard earlier. You can also paste by right-clicking over the field and left-clicking **Paste**.

Under Content type, select **application/json**.

Click the **Add webhook** button. A green check to the left of the address means it correctly configured. (It may not show a check until you use Git to push changes).

Notes:

- Please review the OpenShift basic walkthrough for a few more details and information on how to scale your application: https://docs.openshift.com/online/getting_started/basic_walkthrough.html#getting-started-basic-walkthrough.

Lab: Introduction to OpenShift Command Line Interface

Overview
Time: 20 Minutes

Prerequisites:

- A computer system with internet connectivity.
- A GitHub account.
- A free OpenShift Online account.
- Git software installed.
- Complete OpenShift Labs: *Getting Started with OpenShift Online* and *Installing a Sample App* before this step.
- Keep an open browser window logged in to your GitHub account.
- Open command prompt/terminal to use the OpenShift Command Line Interface (CLI).

In this lab, you will:

- Log in to OpenShift using CLI.
- Delete existing project or projects using the CLI.
- Create a project using the CLI.
- Deploy an example application from source code.
- Configure a route for your application.
- Log out of OpenShift CLI.

Step 1: Command Line Interface Login.
Open the command line interface (if it is not still open). Remember to open it as an administrator if you are using Windows. * In the command prompt type:

```
    oc help
```

Read through the list of available commands.

If you want help with a specific command, type "oc command -- help". Instead of the word command, type the actual command. Try typing the command below.

```
       oc login --help
```

This gives you helpful information for knowing how to login to your OpenShift service using CLI.

In order to log in, you will need to get some information from your OpenShift online account. For that, you will be using the 'oc' command.

Under the download links, copy the session token by clicking on the copy icon to the right of the top field (Under the Mac OS X download link) as pictured below.

Your login command with the session token should now be copied to your clipboard.

Go back to your Command Prompt or terminal and log in to your OpenShift account.

Paste the text into the Command Prompt or Terminal. You can use Ctrl - V if your O/S supports it. Press **Enter**.

If it works, it will say that you have logged in and it will provide basic information as below. It should say your actual user name instead of username.

> Logged into "https://api.starter-us-west-1.openshift.com:443" as "username" using the token provided. You have one project on this server: "username-example" Using project "username-example".
>
> C:32>

Step 2: Delete Existing Projects or Apps or Services.

In order to try to install another project, you must not have any projects installed already (if you are using the OpenShift Online starter free plan). We will learn how to delete existing projects by using the CLI.

Note that if you want to delete the entire project, then you only need to use the "oc delete project" command. The "oc delete" command can also delete specific objects or items from your project.

If you do not have any projects now, you do not need to delete a project below.

Type the following command into the command prompt or terminal:

```
        oc get projects
```

You will see a list of projects if you have any. To select a project use this command below. Replace username-example with the actual name of your project.

> oc project username-example

Delete the objects in your selected project with this command.

```
        oc delete all --all
```

It should list the names of the objects deleted. Check the names of any persistent volume claims PVC.

```
        oc get pvc
```

If it says "No resources found," you do not need to delete it. If you have any PVC, such as mongodb, delete it using:

```
        oc delete pvc mongodb
```

Now go ahead and delete the project with this command. Replace username-example with the actual name of your project.

```
        oc delete project username-example
```

Check the status of available projects; if you only had one project, you should not have any projects now.

```
        oc projects
```

To change to another project, you will type "oc project" then the project name. Just for your information, If you had another project named "project2" you could type the command below.

```
        oc project project2
```

Step 3: Create a Project Using the CLI.
In this step, create a project and then we will deploy an application in Step 4.

For examples below, instead of typing username-example, type a unique project name. As before, you can use replace the word "username" with your actual username.

- Option 1: Minimal description.

```
        oc new-project username-example
```

- Option 2: Full description. Use the following arguments with the command.
  - --description="" is for the project description.
  - --display-name="" is for the project display name.
  - Names in description and display name can have spaces but must be in quotes.

```
        oc new-project username-example --display-name="Example
Project" --description="Example Project made with CLI"
```

Note: The actions you use with oc affect the project you are currently on. If you have more than one project you will need to make sure you are on the right project. To check which project you are currently on, use this command.

```
               oc project
```

Note that oc projects tells you the list of projects, and oc project tells you which one you are on.

Step 4: Work with Labels in your Project

In this step, you will create and modify labels.

To update the label on your project, use this command

```
oc label <object_type> <object_name_or_id> \<label>
```

To list the labels, use the 'oc describe' command.

```
$ oc describe svc docker-registry
Name:                   docker-registry
Labels:                 docker-registry=default
Selector:               docker-registry=default
IP:                     172.30.78.158
Port:                   <unnamed>       5000/TCP
Endpoints:              10.1.0.2:5000
Session Affinity:       None
No events.
```

To act on projects, you can use labels. For example, to gracefully shutdown the project, do this:

`oc stop` *`<object_type>`* `-l` *`<label>`*

Step 5: Deploy an app from source code.
In this step, we will deploy the same application we used in OpenShift Lab: *Installing a Sample App*; but this time, we will use the command line. * In that lab Step 1, we instructed you to Fork an application to your GitHub account from: https://github.com/elephantscale/hello-openshift-nodejs

If you have already forked this application to your own GitHub account, and cloned the application to your local system, you do not need to do it again. If not, go to Lab: *Installing a Sample App* Step 1 and follow the directions until you get to Step 2, then return back here.

Using a separate web browser window/tab, log in to GitHub.com and keep that page/tab open.

Navigate to the example app hello-openshift-nodejs in your GitHub account, then copy the GitHub URL to your clipboard. You can select the text and use Ctrl - C if your O/S supports it.

The URL should look like this with your actual username instead of the word username: https://github.com/username/hello-openshift-nodejs

- In the command prompt or terminal, type this command to deploy the application:

```
          oc new-app https://github.com/cedarfox/hello-openshift-nodejs
```

Type this command to check the status of the running application:

```
          oc status
```

The app should be running but not exposed, so you cannot access the app via your browser yet.

Step 6: Configure a Route for Your Application.
You will not be able to view the running application in a web browser until you configure a route.
* In the command prompt or terminal type the command:

```
          oc get services
```

It should display something like the following. (Shown on windows computer. Do not type the path.)

```
        C:32>oc get services NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S)
AGE hello-openshift-nodejs ClusterIP 172.30.253.226  8080/TCP 13m
C:32>_
```

To expose the routes for your application so it can be accessible, type the following command:

```
        oc expose service/hello-openshift-nodejs
```

If successful, it will say "route "hello-openshift-nodejs" exposed". Now type this command:

    oc get routes

- This will display:
  - the name of the app - "hello-openshift-nodejs ".
  - HOST/PORT - a custom address you can use to access the app via browser.
  - Services - "hello-openshift-nodejs".
  - Port - 8080-tcp.

Select the address listed under HOST/PORT and copy it to the clipboard.

Open up a new browser/tab and paste the address into the address bar and load the page.

It should show the same screen as in Lab *Installing a Sample App*.

Navigate back to your web console to view the status of the application.

Logout of Openshift CLI:

```
          oc logout
```

When you log in again later, you will need to copy your session token login command again from the OpenShift web console as described in Step 1. The session token changes each time you log in.

After you set up your app using the CLI, you may set up to automatically reconfigure builds. It is recommended that you obtain the GitHub webhook URL from the OpenShift web console as described in OpenShift Lab: *Installing a Sample App* Step 4.

References:

- https://docs.openshift.com/online/getting_started/beyond_the_basics.html#getting-started-beyond-the-basics

Lab: OpenShift: Using Command Line Interface to Automatically Rebuild Your Application

Overview
Time: 30 Minutes

Prerequisites:

- A computer system with internet connectivity.
- A GitHub account.
- Keep an open browser window logged in to your GitHub account.
- OpenShift Online free account.
- Git software installed.
- Complete pervious OpenShift Labs before this step.
- A text editor of your choice installed to your system.
    - Note for Windows users: Do not use Notepad as the text editor. If you do not have another text editor installed, you should download and install a text editor.
    - Some example text editors:
        - Notepad ++ - https://notepad-plus-plus.org/
        - Atom - https://atom.io/
        - Brackets - http://brackets.io/
        - Visual studio code - https://code.visualstudio.com/
        - Vim text editor - https://www.vim.org/

Note:
At this point you should still have the project and sample application from the OpenShift Lab: *Introduction to OpenShift Command Line Interface*. If you do not, go through the steps with that lab again.

In this lab, you will:

- Clone an application from the GitHub repository.
- Log in to OpenShift using the CLI.
- Delete existing webhooks from your GitHub account.
- Obtain a webhook URL using the CLI.
- Add a webhook using the URL you have obtained.
- Edit the source code of your application.
- Deploy an example app from source code
- Use "git push" to update changes.

Step 1: Clone an Application from the GitHub Repository.

** Note: If you already did this in OpenShift Lab: *Installing a Sample App*, then you do not need to repeat it. **

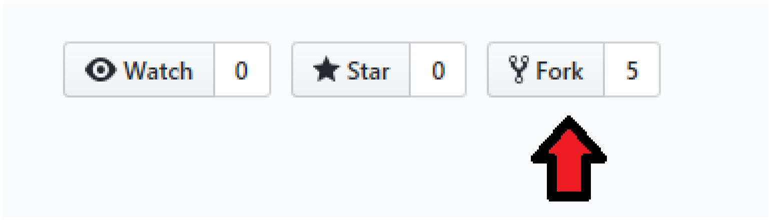
Open up a separate web window/tab to your GitHub account and sign in using your GitHub credentials.

Follow the link below.

https://github.com/elephantscale/hello-openshift-nodejs

– This will bring you to the elephantscale/hello-openshift-nodejs repository with an example Node.js application, which will display a message on the site.

In the upper-right-hand corner of the GitHub repository, click on the **Fork** button. This will provide you with your own copy of the application in your GitHub account.



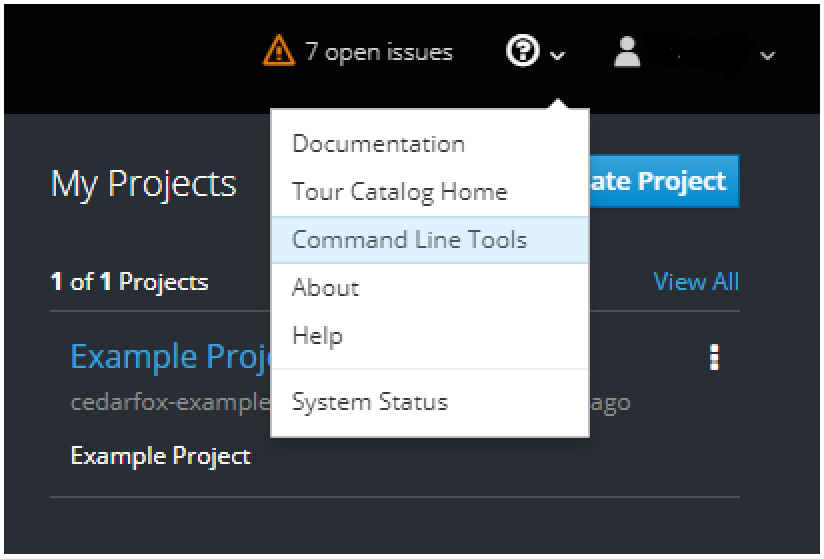
If using Windows, open the Git Bash console on your local computer as described in Openshift Lab: *Getting Started with OpenShift Online*. If using Mac or Linux computers, you can use the terminal.

Enter the following commands into the Git Bash console or terminal to clone the example application to your local computer and to change directories to that app. Put your actual GitHub username in place of "username."

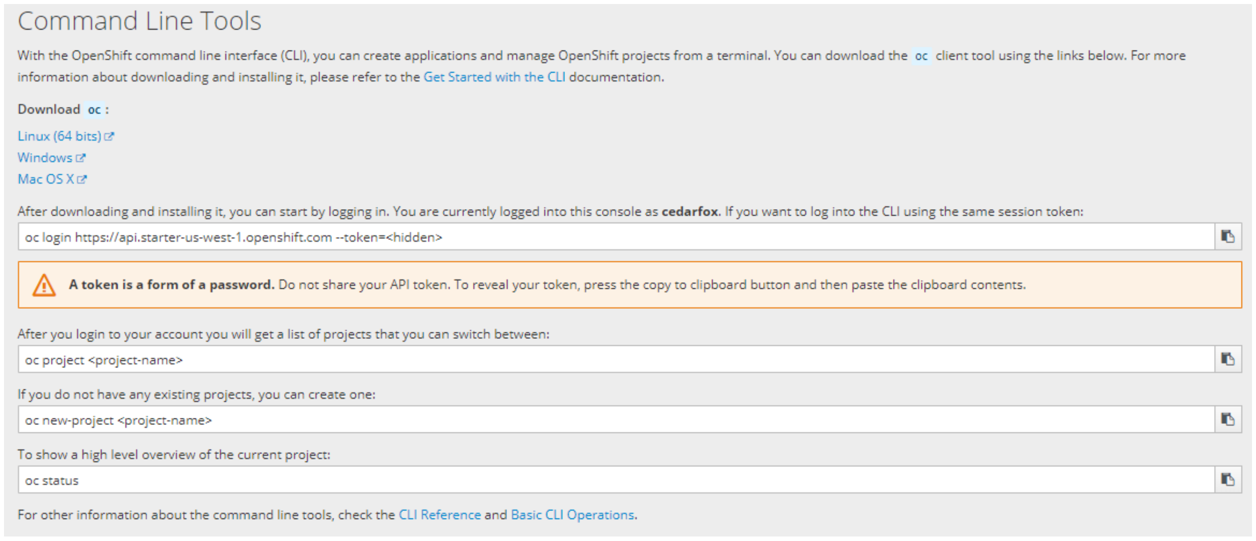
```
       bash   $ git clone https://github.com/username/hello-openshift-
nodejs   $ cd hello-openshift-nodejs ### Step 2 - Login to Openshift
using CLI
```

Log in to your OpenShift online account using a web browser and navigate to the web console.

At the top right of your screen, you will see a "?" with a down arrow as pictured below. Click on the down arrow and then click on **Command Line Tools**.



This will bring you to a screen as pictured below.



Command Line Tools

With the OpenShift command line interface (CLI), you can create applications and manage OpenShift projects from a terminal. You can download the oc client tool using the links below. For more information about downloading and installing it, please refer to the Get Started with the CLI documentation.

Download oc :

Linux (64 bits)
Windows
Mac OS X

After downloading and installing it, you can start by logging in. You are currently logged into this console as **cedarfox**. If you want to log into the CLI using the same session token:

```
oc login https://api.starter-us-west-1.openshift.com --token=<hidden>
```

A token is a form of a password. Do not share your API token. To reveal your token, press the copy to clipboard button and then paste the clipboard contents.

After you login to your account you will get a list of projects that you can switch between:

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

If you do not have any existing projects, you can create one:

```
oc new-project <project-name>
```

To show a high level overview of the current project:

```
oc status
```

For other information about the command line tools, check the CLI Reference and Basic CLI Operations.

Click the copy icon to the right of the "oc login" field.

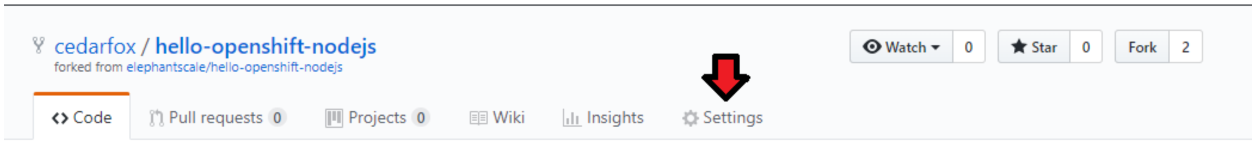
Open your terminal, or in Windows, run the command prompt "as an administrator."

Paste the oc login code into the command prompt or terminal and press **Enter**. This should log you in to the OpenShift CLI; keep the command prompt or terminal window open.

Step 2: Delete Existing Webhooks from Your GitHub Account.

Navigate to the following address, but replace "username" with your actual GitHub username: https://github.com/username/hello-openshift-nodejs

Click on the **Settings** button. Then, click on the **Webhook** tab on the left.


cedarfox / hello-openshift-nodejs
forked from elephantscale/hello-openshift-nodejs
Watch 0
Star 0
Fork 2
Code
Pull requests 0
Projects 0
Wiki
Insights
Settings

Look under webhooks and see if there is already a webhook installed. If there is, click **Delete**. Then, click the **Yes, delete webhook** button.

Keep this window open.

Step 3: Obtain a Webhook URL Using the CLI.

Open up a new text file with your preferred text editor.

Go back to your command prompt/terminal window.

Type the following command. Note that "bc" means BuildConfig.

```
            oc describe bc hello-openshift-nodejs
```

This will print up information from the application's BuildConfig file.

Look for the field that says "Webhook Github URL:".

Select the entire URL and copy it to your clipboard.

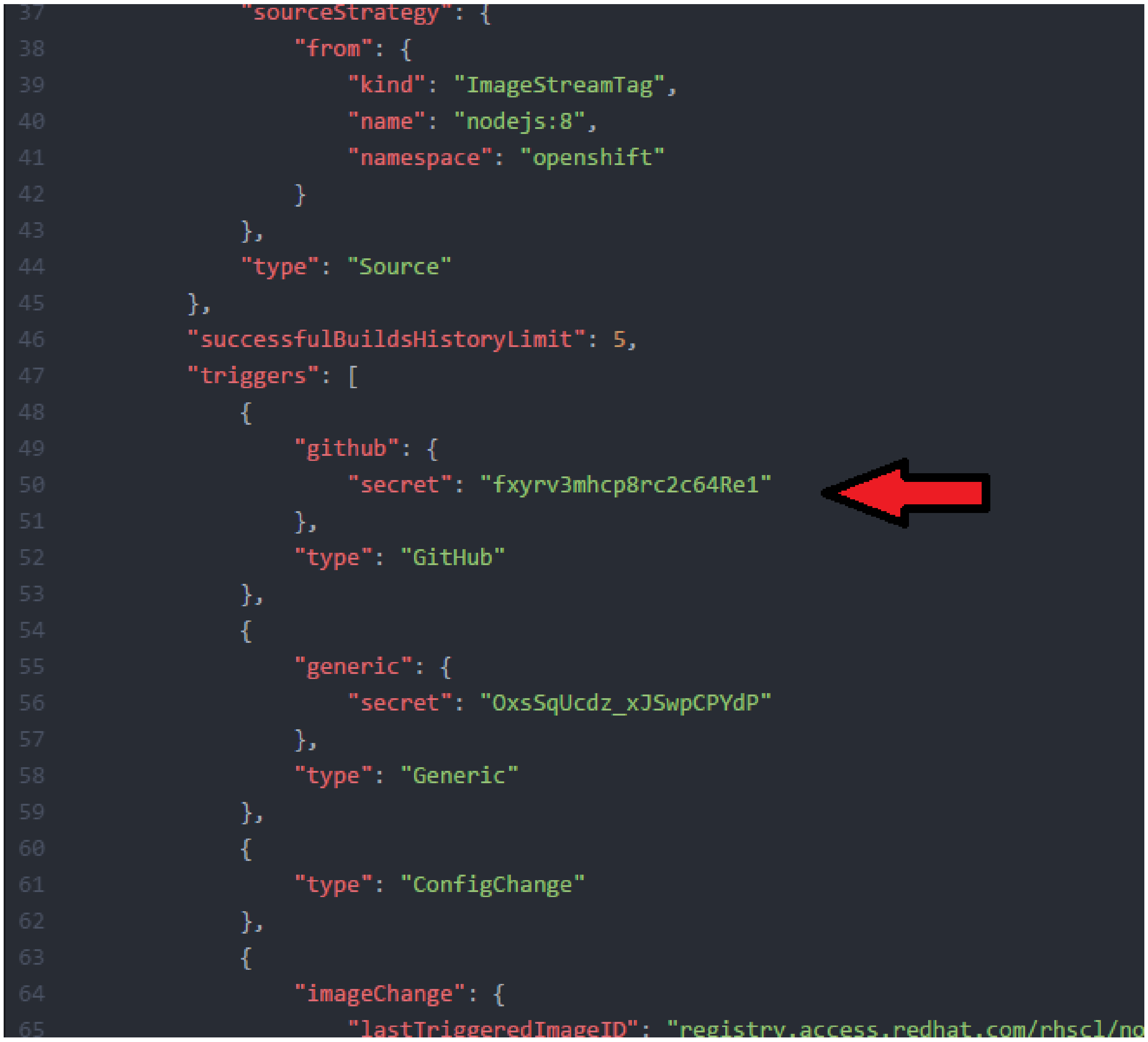
Paste the URL into your new text file.

Type the following command to save a copy of the BuildConfig file to your local directory.

```
oc get bc/hello-openshift-nodejs -o json > bc.json
```

Navigate to the current directory and open the bc.json file using a text editor. (If using Windows, you can use Windows Explorer to navigate to the folder. Do not use Notepad for the text editor.)

Scroll down the file until you see the word "github" and underneath, you see the word "secret".

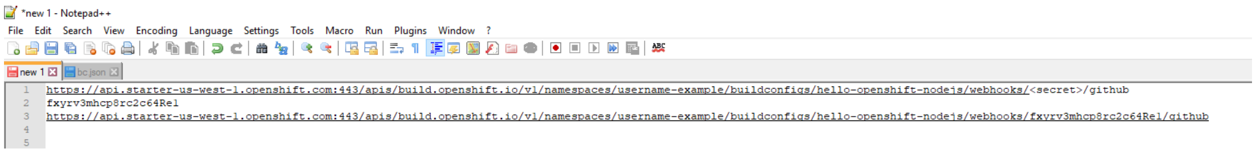


```
            "sourceStrategy": {
                "from": {
                    "kind": "ImageStreamTag",
                    "name": "nodejs:8",
                    "namespace": "openshift"
                }
            },
            "type": "Source"
        },
        "successfulBuildsHistoryLimit": 5,
        "triggers": [
            {
                "github": {
                    "secret": "fxyrv3mhcp8rc2c64Re1"
                },
                "type": "GitHub"
            },
            {
                "generic": {
                    "secret": "OxsSqUcdz_xJSwpCPYdP"
                },
                "type": "Generic"
            },
            {
                "type": "ConfigChange"
            },
            {
                "imageChange": {
                    "lastTriggeredImageID": "registry.access.redhat.com/rhscl/no
```

Copy the secret text under "github" without copying the quotes.

- Paste the secret characters in the new text file underneath the URL you pasted earlier.

- You will see that the URL ends in .../hello-openshift-nodejs/webhooks/<secret>/github
- Delete the characters: <secret>
- In between the slashes where <secret> was, paste the actual secret, as in the example below.


*new 1 - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
new 1
bc.json

```
https://api.starter-us-west-1.openshift.com:443/apis/build.openshift.io/v1/namespaces/username-example/buildconfigs/hello-openshift-nodejs/webhooks/<secret>/github
fxyrv3mhcp8rc2c64Rel
https://api.starter-us-west-1.openshift.com:443/apis/build.openshift.io/v1/namespaces/username-example/buildconfigs/hello-openshift-nodejs/webhooks/fxyrv3mhcp8rc2c64Rel/github
```

Now, select and copy the entire URL which has the secret in it.

Step 4: Add a Webhook Using the URL You Have Obtained.

Go back to your open GitHub browser window. Click the **Add webhook** button in the upper right.

Click the **Payload URL** field and press **Ctrl-V** to paste the URL you copied to the clipboard earlier.

Under Content type, select **application/json**.

Click on the green **Add webhook** button. A green check to the left of the address means it correctly configured. Note that it may not show a green check until you use "git push" to update your changes.

Step 5: Edit the Source Code of Your Application.

Navigate to the local cloned copy of hello-openshift-nodejs. Note that if using Git Bash, the cloned copy will be in the hello-openshift-nodejs folder inside the starting default folder.

In Windows, the default starting folder is C: your-username * (replace "your-username" with your Windows user account name).

Open the server.js file inside the hello-openshift-nodejs folder using a text editor.

On line 7, after the word "app." but before "", type your own custom text. You could type "Excellent!!!" without the quotes, or something else if you prefer.

Save your changes to the file.

Step 6: Use "git push" to Update Changes.

Open the Git Bash command line (terminal in Linux/Mac).

Change directories to the application. (Note: do not type the $ in these examples.)

```
        $ cd hello-openshift-nodejs`
```

Type `git status` and it should list server.js in red.

Now type the following:

```
        $ git add server.js
```

Type `git status` again and it should list server.js in green.

Type `git commit` and press **Enter**.

A new window opens. Type a description of your changes. Then follow the prompts to press Ctrl-X and then press "Y," and press **Enter**.

Type `git push` into the command prompt and press **Enter**. This will upload the changes into the server.

Open your command prompt/terminal where you were using the "oc" commands and type:

```
        oc get routes
```

Under HOST/PORT, it will tell your URL for your apps. Copy the entire URL to the clipboard.

Paste the URL into a new browser window and press **Enter**.

It should show you the application text. It may take a couple of minutes to update the app after you push the changes. But after a couple of minutes, it should read the new text.

If it shows the text which you added in the source code, then you have configured it correctly.

References:

- https://docs.openshift.com/online/getting_started/beyond_the_basics.html#getting-started-beyond-the-basics

Lab: Using Web Console and Command Line Interface to View and Edit Configuration Files

Overview
Time: 20 Minutes

Prerequisites:

- A computer system with internet connectivity.
- A GitHub account.
- Keep an open browser window logged in to your GitHub account.
- OpenShift Online free account.
- Git software installed.
- Complete pervious OpenShift Labs before this step.
- A text editor of your choice installed to your system.
    - Note for Windows users: Do not use notepad as the text editor. If you do not have another text editor installed, you should download and install a text editor.
    - Some example text editors:
        - Notepad ++ - https://notepad-plus-plus.org/
        - Atom - https://atom.io/
        - Brackets - http://brackets.io/
        - Visual studio code - https://code.visualstudio.com/
        - Vim text editor - https://www.vim.org/

In this lab, you will:

- Learn how to view and edit configuration files using the web console.
- Learn how to configure health checks using the web console.
- Learn how to view and edit configuration files using the CLI.
- Learn how to configure health checks using CLI.
- Learn how to create and edit secret files.
- Learn how to create a PersistentVolumeClaim (PVC) and edit the PVC object definition file.
- Learn how to create pods and edit pod object definition files.

Notes:

- If the example commands show a $ command prompt, then do not actually type the $.

Initial preparation:

Keep a browser window logged into your GitHub account.

Open a browser window and log in to your OpenShift Online account and enter the web console.

Open a terminal (or in Windows a command prompt "as an administrator".)

If you are not logged in to your OpenShift Online account through the command prompt/terminal, then log in using the `oc login` command copied from your web console. (Lab: Introduction to OpenShift Command Line Interface, describes how to log in.)

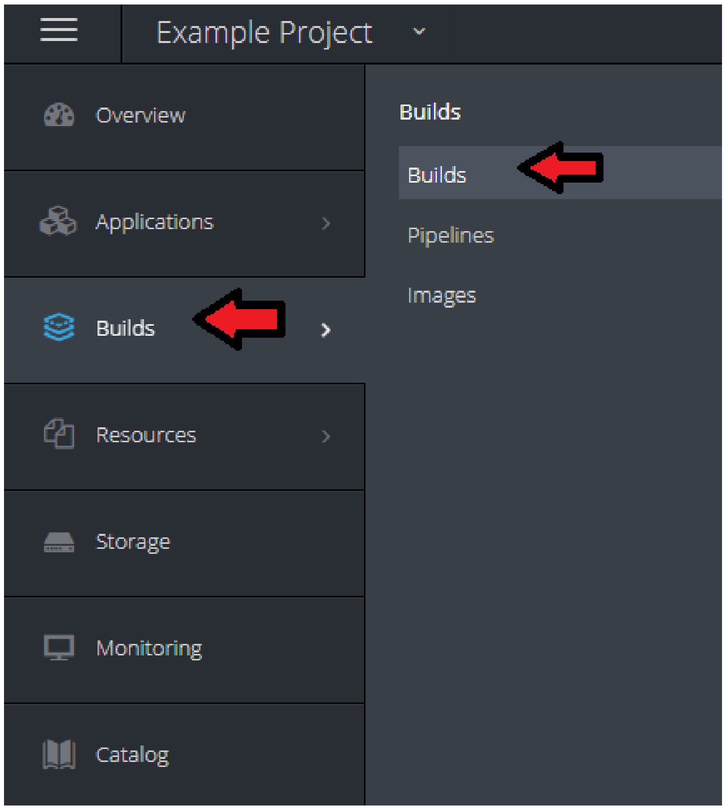
Type `oc project` into the command line. It will tell you the name of the project you are working on.

Topics to Cover

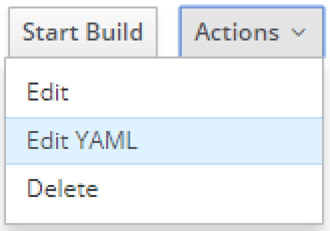
- How to view and edit configuration files using the web console.
- How to configure health checks using the web console.
- How to view and edit configuration files using the CLI.
- How to configure health checks using CLI.
- How to create and edit secret files.
- How to create a PersistentVolumeClaim (PVC) and edit the PVC object definition file.
- How to create pods and edit pod object definition files.

How to View and Edit Configuration Files Using the Web Console

- Log in to your Openshift online account and navigate to the web console.
- Click on your project name in the upper right hand of the screen.
- Open the BuildConfig file on the web console by doing the following steps.
  - On the left side of the screen, click the "Builds" tab, then click the word "Builds" in the menu.

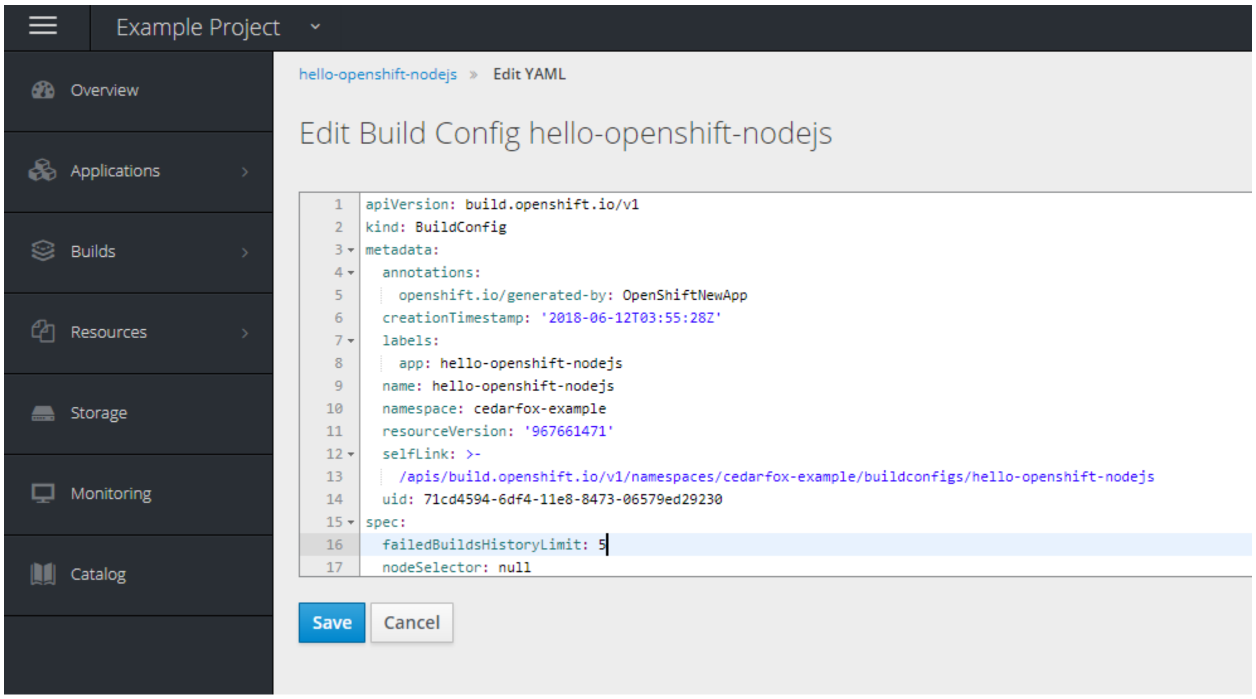


- Click on the name of the build. In this case it is "**hello-openshift-nodejs**."
- On the upper right hand of the screen click the "**Actions**" button, then click on "**Edit YAML**".



Edit the BuildConfig file.

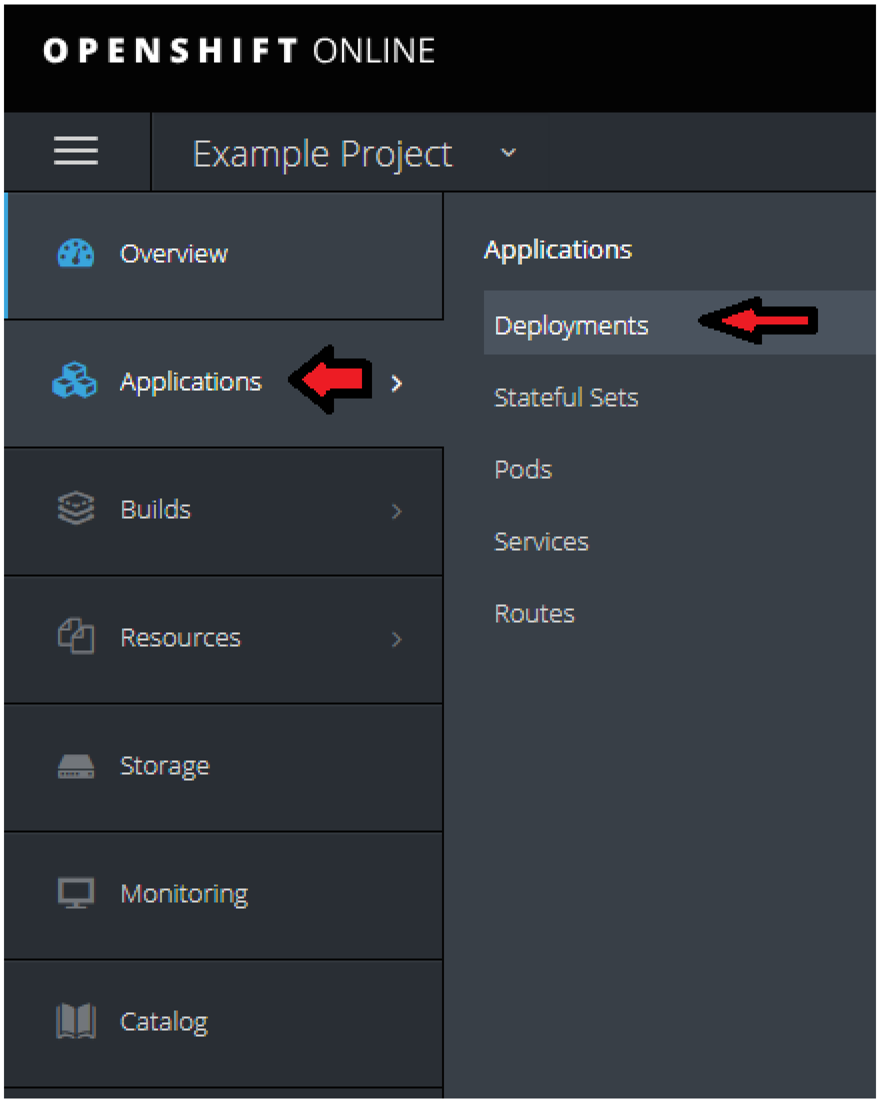
- Scroll down through the lines to see what is in the BuildConfig file.
- Scroll down to Line 16 where it says "failedBuildsHistoryLimit: 5".



```
1  apiVersion: build.openshift.io/v1
2  kind: BuildConfig
3  metadata:
4    annotations:
5      openshift.io/generated-by: OpenShiftNewApp
6    creationTimestamp: '2018-06-12T03:55:28Z'
7    labels:
8      app: hello-openshift-nodejs
9    name: hello-openshift-nodejs
10   namespace: cedarfox-example
11   resourceVersion: '967661471'
12   selfLink: >-
13     /apis/build.openshift.io/v1/namespaces/cedarfox-example/buildconfigs/hello-openshift-nodejs
14   uid: 71cd4594-6df4-11e8-8473-06579ed29230
15 spec:
16   failedBuildsHistoryLimit: 5
17   nodeSelector: null
```

- Change the number 5 to a **6**.
- Click the blue "**Save**" button. You should see a confirmation that the build configuration was updated.

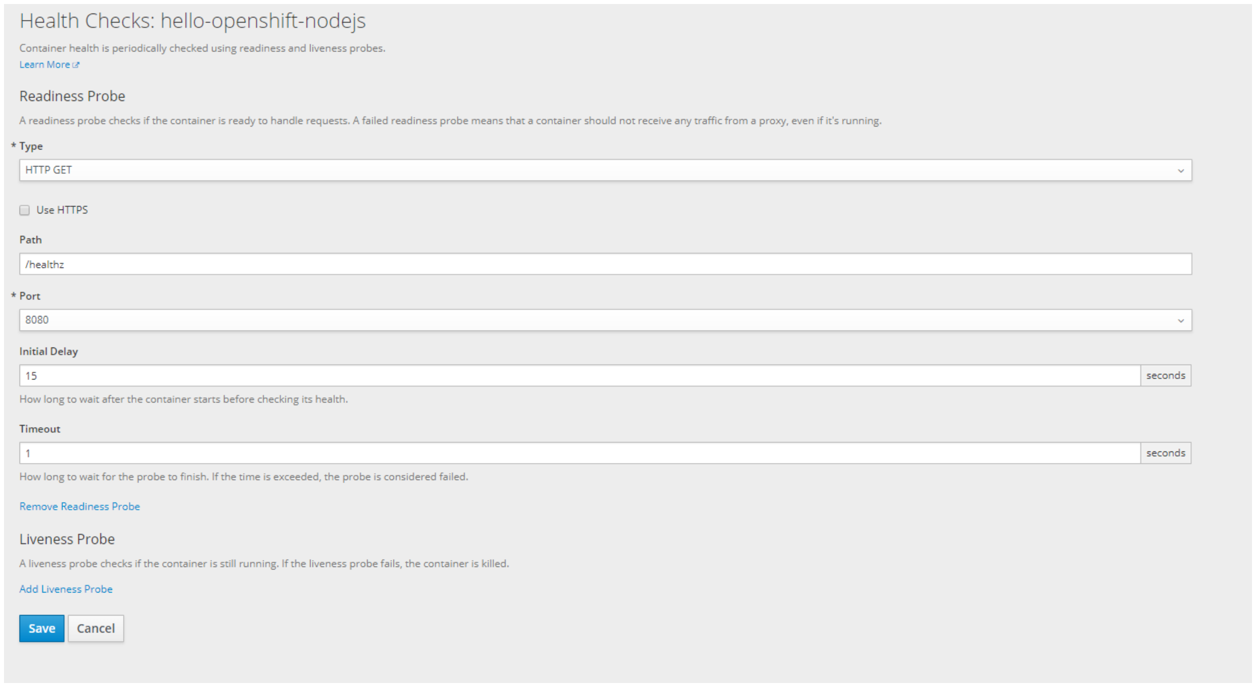
- Open the DeploymentConfig file on the web console doing the following.
  - On the left side of the screen, click the "**Applications**" tab, then click "**Deployments**" in the menu that opens.



- Click on the name of the application. In this case it is "**hello-openshift-nodejs**."
- Click on the button in the right upper hand of the screen that says "**Actions**." Then click "**Edit YAML**."
- Scroll down to review the contents and note how many lines are in the DeploymentConfig file. Remember the number.
- Then click "**cancel**" on the bottom to exit the screen.

How to Configure Health Checks using the Web Console

- Add a readiness Probe – a type of health check.
- On the right upper hand of the screen, click "**Actions**," then click "**Edit Health Checks**."
- Click the link that says "**Add readiness probe**."
- In the field that says Path, type `/healthz`.
- In the field that says Initial Delay, type `15`
- Ensure that port is set to **8080** and Timeout set to **1**.
- Click **Save** at the bottom. This will save the readiness probe to the DeploymentConfig file.



Health Checks: hello-openshift-nodejs

Container health is periodically checked using readiness and liveness probes.

Learn More

Readiness Probe

A readiness probe checks if the container is ready to handle requests. A failed readiness probe means that a container should not receive any traffic from a proxy, even if it's running.

* Type: HTTP GET

Use HTTPS

Path: /healthz

* Port: 8080

Initial Delay: 15 seconds

How long to wait after the container starts before checking its health.

Timeout: 1 seconds

How long to wait for the probe to finish. If the time is exceeded, the probe is considered failed.

Remove Readiness Probe

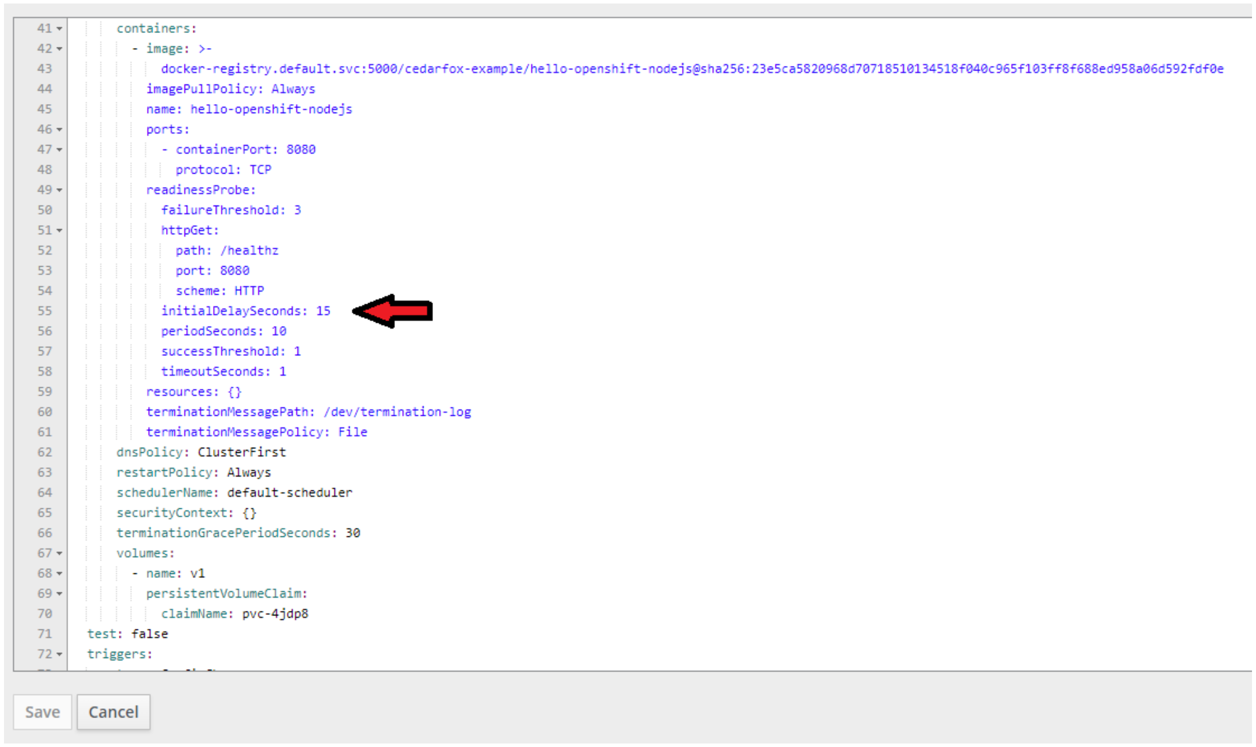
Liveness Probe

A liveness probe checks if the container is still running. If the liveness probe fails, the container is killed.

Add Liveness Probe

- Open the **DeploymentConfig** file again using "**Edit YAML**" and scroll down to view the contents.

  - In line 49 of the DeploymentConfig file, you should see the readiness probe added to the file with the specifications below.
  - In line 55, you will see "initialDelaySeconds: 15". Change the number 15 to **16**.



```
41      containers:
42        - image: >-
43            docker-registry.default.svc:5000/cedarfox-example/hello-openshift-nodejs@sha256:23e5ca5820968d70718510134518f040c965f103ff8f688ed958a06d592fdf0e
44          imagePullPolicy: Always
45          name: hello-openshift-nodejs
46          ports:
47            - containerPort: 8080
48              protocol: TCP
49          readinessProbe:
50            failureThreshold: 3
51            httpGet:
52              path: /healthz
53              port: 8080
54              scheme: HTTP
55            initialDelaySeconds: 15
56            periodSeconds: 10
57            successThreshold: 1
58            timeoutSeconds: 1
59          resources: {}
60          terminationMessagePath: /dev/termination-log
61          terminationMessagePolicy: File
62      dnsPolicy: ClusterFirst
63      restartPolicy: Always
64      schedulerName: default-scheduler
65      securityContext: {}
66      terminationGracePeriodSeconds: 30
67      volumes:
68        - name: v1
69          persistentVolumeClaim:
70            claimName: pvc-4jdp8
71  test: false
72  triggers:
```

  - Scroll down to the bottom, you will see that there are a few more lines total in the DeploymentConfig file.
  - Click the **Save** button.

- Now, we will add a liveness probe as well.

  - Click "**Actions**," then click "**Edit Health Checks**."
  - Click "**Add liveness probe**."
  - In the field that says Path, type **`/healthz`**.
  - In the field that says Initial Delay, type **`15`**
  - Ensure that port is set to 8080 and Timeout set to `1`.
  - Click the **Save** button.

- Open the **DeploymentConfig** file again using "**Edit YAML**" and scroll down to view the contents.

  - Look through the lines and identify the liveness probe details and readiness code details.
  - When you are done looking click "Cancel."

How to View and Edit Configuration Files using CLI

- Log in to Openshift by CLI (if not already logged in).
  - Open the terminal window or command prompt window "as an administrator."
  - If you are not logged in, use the "oc login" command as described in Lab: Introduction to OpenShift Command Line Interface

- View the BuildConfig file using the `oc edit` command.

```
              oc edit bc/hello-openshift-nodejs
```

- A text editor should open up with the contents of the BuildConfig.
  - Note: Windows will likely use Notepad by default for this option, so it is not advisable to save any changes with Notepad, since Notepad may not save the text in the correct format.

- Look through the contents of the BuildConfig file and then **close** the text editor.

- View and edit the BuildConfig file using the `oc get` command.
  - Save a local copy of the BuildConfig file in .json format by the following command.

```
      oc get bc/hello-openshift-nodejs -o json > test.json
```

  - Open the file **test.json** using a text editor (but do not use notepad).
  - Scroll down to the field "**failedBuildsHistoryLimit: 6**".
    - Change the number 6 to **7**.
    - **Save** the file and **close** your text editor.
    - Type the following command to replace BuildConfig file on the server with your local copy.

```
         oc replace -f test.json
```

- Edit it again using `oc edit`.

```
         oc edit bc/hello-openshift-nodejs
```

- Confirm that the "failedBuildsHistoryLimit:" now shows "7."
- Close the text editor without saving.
- Now practice this same process with the DeploymentConfig file as you did above with the BuildConfig file.
  - View the file in a text editor.

```
      oc edit dc/hello-openshift-nodejs
```

- Save a copy of the DeploymentConfig locally.

```
      oc get dc/hello-openshift-nodejs -o json > testdc.json
```

- Open the file testdc.json in a text editor.
- Scroll down to the text about "liveness probe" and look below for the text that says "**initialDelaySeconds : 15**"
- Change the number "15" to **17**.
- **Save** the changes with the text editor and **close** the editor.
- Type in the "`oc replace`" command to replace the file on the server.

```
      oc replace -f testdc.json
```

- Use the oc edit command to verify that changes were applied.

```
      oc edit dc/hello-openshift-nodejs
```

- Look in the code under "liveness probe" and it should read "initialDelaySeconds : 17".
- Close the text editor without saving.

How to Configure Health Checks using CLI

- Delete all existing readiness and liveness probes using `oc set probe` by typing the following command.

```
 oc set probe dc/hello-openshift-nodejs --remove --readiness --
liveness
```

- Use the `oc edit` command to verify that changes were applied.

```
    oc edit dc/hello-openshift-nodejs
```

- The information about readiness probes and liveness probes should be removed from the DeploymentConfig file.
- Close the text editor without saving.
- Create a readiness probe like the one we made in the web console by the following command.

```
    oc set probe dc/hello-openshift-nodejs --readiness --get-
url=http://:8080/healthz --initial-delay-seconds=20
```

- This time we will set the initial delay to 20 - a different number than before.
- Create a liveness probe by using this command.

```
     oc set probe dc/hello-openshift-nodejs --liveness --get-
url=http://:8080/healthz --initial-delay-seconds=20
```

- Create a readiness probe by using this command.

```
  oc set probe dc/hello-openshift-nodejs --readiness --get-
url=http://:8080/healthz --initial-delay-seconds=20
```

- You can change the parameters by running the command again. This command uses the `--open-tcp` argument to update the port to 3306 in the liveness probe.

```
   oc set probe dc/hello-openshift-nodejs --liveness --open-
tcp=3306
```

- Use the `oc edit` command to verify that changes were applied.

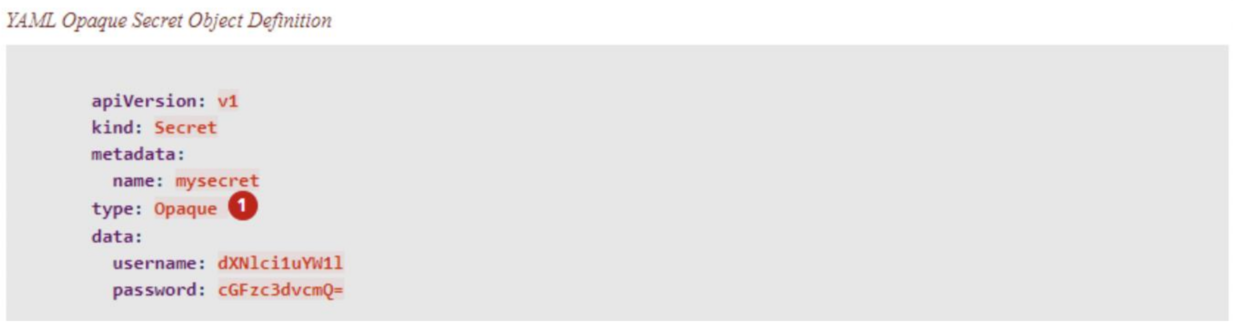
```
    oc edit dc/hello-openshift-nodejs
```

  - Close the text editor without saving.

How to Create and Edit Secret Files

- Navigate to this site to get an example secret file. https://docs.openshift.com/online/dev_guide/secrets.html#secrets-examples

- Locate the example secret as in the image below and select and copy the secret data starting with "apiVersion".


YAML Opaque Secret Object Definition

```
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque ❶
data:
  username: dXNlci1uYW1l
  password: cGFzc3dvcmQ=
```

- Open a text editor to a new file and copy the fields.
- Change the name of the secret from "mysecret" to `secret1`
- Save the file using the .yaml format and call it **secret1**. **Save** it in the directory you are working in with your terminal or command prompt.
- Type the following command to create a secret object using the file.

```
    oc create -f secret1.yaml
```

- It should say "secret 'secret1' created."
- Check the list of secrets by typing: `oc get secret;` "secret1" should be listed.
- To view the secret in the default editor, type this command.

```
     oc edit secret secret1
```

- View the file, then close the file without saving.
- To save a copy of the secret file to your computer, use the `oc get` command.

```
    oc get secret secret1 -o yaml > testsecret.yaml
```

- Now, open the `testsecret.yaml` using a text editor.
- We won't actually make changes, but for the sake of example, we will imagine the file has been modified. Close the text editor.
- Replace the secret on the server with the local file `testsecret.yaml` by typing the following command.

```
        oc replace -f testsecret.yaml
```

How to Create a PersistentVolumeClaim (PVC) and Edit the PVC Object Definition File

- Check if you have any PVCs in your project by typing: **`oc get pvc`**.
- It should say "No resources found," unless you have created one.
- Create a PVC using the `oc volume` command:

```
      oc volume dc/hello-openshift-nodejs --add --name=v1 -t pvc --
claim-size=1Gi --overwrite
```

- The --overwrite argument will cause the new PVC to overwrite an existing volume, if present.
- Now, type `oc get pvc` again to check the status. Check the name of the PVC.
- You may view the PVC object definition file by typing the following. Replace "`pvc-4jdp8`" with the actual name of your PVC.

```
        oc edit pvc/pvc-4jdp8
```

- You may save a local copy of the PVC object definition file using this command. Replace pvc-4jdp8 with the actual name of the PVC.

```
      oc get pvc pvc-4jdp8 -o yaml > pvc.yaml
```

- Open the file `pvc.yaml` with a text editor to view the contents. Then close the file.
- Imagine that we have modified the `pvc.yaml` file and now we will use the local copy to replace the `pvc.yaml` file on the server.

```
      oc replace -f pvc.yaml
```

How to Create Pods and Edit Pod Object Definition Files

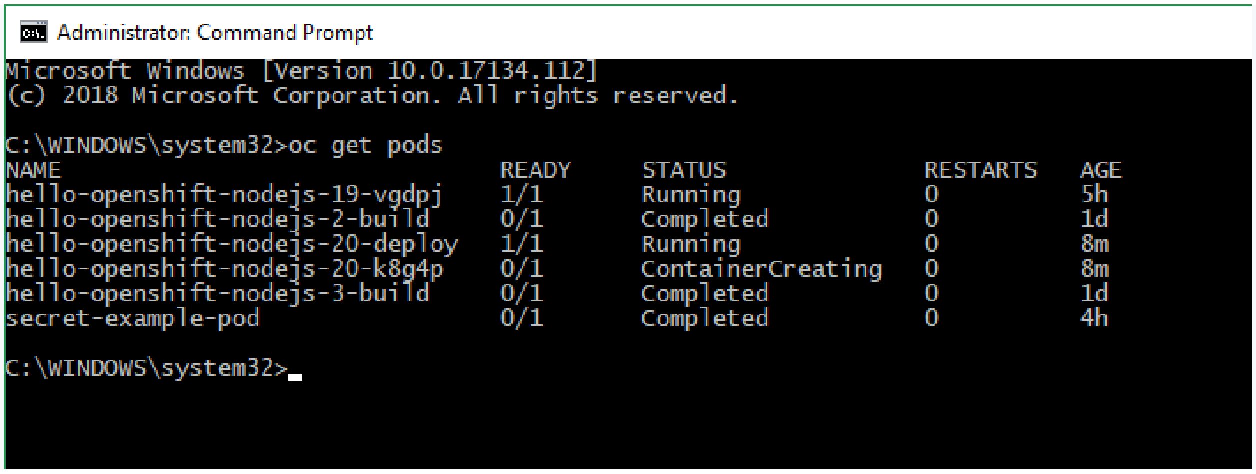
- Navigate to the following website to get the text for an example pod. Select and copy the pod text from example 2.
https://docs.openshift.com/online/dev_guide/secrets.html#secrets-examples

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-example-pod
spec:
  containers:
    - name: secret-test-container
      image: busybox
      command: [ "/bin/sh", "-c", "cat /etc/secret-volume/*" ]
      volumeMounts:
          # name must match the volume name below
          - name: secret-volume
            mountPath: /etc/secret-volume
            readOnly: true
  volumes:
    - name: secret-volume
      secret:
        secretName: test-secret
  restartPolicy: Never
```

- Open a text editor to a new file and paste the text.
- Change the name of the "secretName" from "test-secret" to **secret1**
- Save the file using the .yaml format and call it **`expod.yaml`** . **Save** it in the directory you are working in with your terminal or command prompt.
- Create a new pod using the `expod.yaml` file by the following command.

```
oc create -f expod.yaml
```

- Check the status of pods by typing oc get pods. You should get a response similar to the following:



Administrator: Command Prompt

```
Microsoft Windows [Version 10.0.17134.112]
(c) 2018 Microsoft Corporation. All rights reserved.

C:\WINDOWS\system32>oc get pods
NAME                             READY     STATUS              RESTARTS   AGE
hello-openshift-nodejs-19-vgdpj  1/1       Running             0          5h
hello-openshift-nodejs-2-build   0/1       Completed           0          1d
hello-openshift-nodejs-20-deploy 1/1       Running             0          8m
hello-openshift-nodejs-20-k8g4p  0/1       ContainerCreating   0          8m
hello-openshift-nodejs-3-build   0/1       Completed           0          1d
secret-example-pod               0/1       Completed           0          4h

C:\WINDOWS\system32>
```

- View the pod object definition file by typing this command.

```
oc edit pod secret-example-pod
```

- After viewing the file, **close** the text editor without saving.
- Save a local copy of the pod file by typing the following command.

```
oc get pod secret-example-pod -o yaml > expod2.yaml
```

- For the sake of the exercise, imagine that we have modified the `expod2.yaml` file.
- Now use the `oc replace` command to load our local copy of the file on to the server.

```
oc replace -f expod2.yaml
```

- You should get a response that says "pod 'secret-example-pod' replaced."

Lab: OpenShift: Working with Complex Deployments

Overview
Time: 20 Minutes

Prerequisites:

- A computer system with internet connectivity.
- A GitHub account.
- Keep an open browser window logged in to your GitHub account.
- OpenShift Online free account.
- Git software installed.
- Complete pervious OpenShift Labs before this step.
- A text editor of your choice installed to your system.
    - Note for Windows users: Do not use notepad as the text editor. If you do not have another text editor installed, you should download and install a text editor.
    - Some example text editors:
        - Notepad ++ - https://notepad-plus-plus.org/
        - Atom - https://atom.io/
        - Brackets - http://brackets.io/
        - Visual studio code - https://code.visualstudio.com/
        - Vim text editor - https://www.vim.org/

In this lab, you will:

- Scale an application.
- Autoscale an application.
- Practice creating pods with an affinity rule.
- Cluster an application.
- Create a route in Openshift.

Notes:

- If the example commands show a $ command prompt, then do not actually type the $.

Initial preparation:

Keep a browser window logged into your GitHub account.

Open a browser window and log in to your OpenShift Online account and enter the web console.

Open a terminal (or in Windows a command prompt "as an administrator".)

If you are not logged in to your OpenShift Online account through the command prompt/terminal, then log in using the `oc login` command copied from your web console. (Lab 2.1 describes how to log in.)

Type `oc project` into the command line. It will tell you the name of the project you are working on.

If you have the free OpenShift Online starter account, then you can only have one project. Delete the project by typing `oc delete project username-example`. Replace username-example with the name of your project.

Make a new project with a unique name using the following command. Replace username-project with the name you want to give the project:

```
       oc new-project username-project --display-name="Test Java
Application"
```

Install an Example Application

Step 1. Open the browser window to the following GitHub link: https://github.com/elephantscale/jee-start

Step 2. Fork the application by clicking the **Fork** button in the upper-right hand of the screen.

Step 3. Open your Git Bash or terminal window.

Step 4. You may optionally change directories to the directory where you want to clone the file.

Step 5. In your Git Bash window or terminal window, clone the forked application from your repository. Instead of "username," type your actual GitHub username.

```
       $  git clone https://github.com/username/jee-start
$  cd jee-start
```

Step 6. Install the jee-start application by typing this command in your command prompt/terminal. Replace the "username" in the GitHub URL with your actual GitHub username.

```
       oc new-app openshift/wildfly-101-
centos7~https://github.com/username/jee-start.git --name=jee-start
```

Step 7. Check the status of the new-app by typing oc status.

Step 8. Give the application a route with the following command.

```
         oc expose svc/jee-start
```

Step 9. Type oc get routes into the command line, and then copy the URL under the HOST/PORT field to your clipboard.

Step 10. Paste the URL into a new web browser window address bar.

Step 11. Note that this application has a hidden HelloWorld link. After the end of the existing URL in the web browser, add /HelloWorld and you should get a view of the HelloWorld screen.

Scaling an Application

Step 1. Check for information about replication controllers by typing `oc get rc`

Step 2. Check for information about pods by typing `oc get pods`

Step 3. Manually scale the application using the oc scale command.

```
        oc scale dc jee-start --replicas=5
```

Step 4. Check for information again about replication controllers by typing `oc get rc`, and about pods by typing `oc get pods`

It should show that there is "5" under "DESIRED", and "2" or more under "CURRENT." Notice in this example that it increased the number of pods from 1 to 2.

Step 5. Navigate to your project under the web console. Then click on the name of your application "jee-start" and the **Configuration** tab.

Step 6. Click on the pencil next to the "5 replicas," and change it to 10 and click the checkmark.

Step 7. Type `oc get rc` again in the command line, and it should show you 10 under the "DESIRED" field.

Autoscaling an Application

Step 1. Go back to the web console window and look on the same page where you changed the replicas, and on the right, click the **Add autoscaler** link.

Step 2. Fill in the details in the form as the following: Min pods: 1, Max pods: 8, CPU Request Target: 80%. Then, click **Save**.

Step 3. Go to your CLI and type oc get hpa to get information on the autoscaling. it will provide details regarding the numbers of pods and CPU usage.

Step 4. Turn off autoscale by deleting the hpa: `oc delete hpa --all`

Step 5. Now autoscale using the `oc autoscale` command.

```
        oc autoscale dc/jee-start --min=1 --max=10 --cpu-percent=75
```

Step 6. Type `oc get hpa` to check the status again. You have successfully scaled and autoscaled.

Practice Creating Pods with an Affinity Rule

In Lab: *Using Web Console and Command Line Interface to View and Edit Configuration Files*, we walked through the process of how to create a pod and edit the pod object definition file.

A pod affinity rule is a form of advanced pod scheduling which tries to put that type of pod on the same node. An Anti-affinity rule tries to put the pods on different nodes.

This is an example of a pod affinity configuration. (Note that on OpenShift Online Starter, there is not enough resources to run both copies of the hello-pod images, so in the second file we changed the name of the image file so it would install.)

Step 1. Select the following text and copy it to the clipboard.

```
apiVersion: v1
kind: Pod
metadata:
  name: team4
  labels:
     team: "4"
spec:
  containers:
  - name: ocp
    image: docker.io/ocpqe/hello-pod
```

Step 2. Open up a new text file in your text editor and paste the text in. Save it in yaml format with the name team4.yaml.

Step 3. Now select the following text and copy it to your clipboard.

```
apiVersion: v1
kind: Pod
metadata:
  name: team4a
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: team
            operator: In
            values:
            - "4"
        topologyKey: kubernetes.io/hostname
  containers:
  - name: pod-affinity
    image: openshift/origin-docker-registry:v0.6.2
```

Examples obtained from: https://docs.openshift.com/container-platform/3.9/admin_guide/scheduling/node_affinity.html

Step 4. Open up another new text file and paste the above text in. Save it in yaml format with the name team4a.yaml .

Step 5. Create the pods by typing oc create -f team4.yaml, enter, oc create -f team4a.yaml

Step 6. Type `oc get pods`. You should see the team4 and team 4a pods listed.

- This was just an example to show the process. You may have a CreateContainerError listed on team4a pod due to using the origin-docker-registry image.

Step 7. Delete the pods by typing oc delete pod team4, enter, oc delete pod team4a.

References:

Duncan, J., Osborne, J. (2018). Openshift In Action. Pages 147-167. Manning Publications: Shelter Island, NY. www.allitebooks.com

https://docs.openshift.com/container-platform/3.6/install_config/router/index.html#install-config-router-overviewhttps://docs.openshift.com/container-platform/3.6/install_config/router/default_haproxy_router.html#install-config-router-default-haproxy

https://docs.openshift.com/container-platform/3.9/admin_guide/scheduling/scheduler.html

https://docs.openshift.com/online/architecture/core_concepts/pods_and_services.htmlhttps://docs.openshift.com/container-platform/3.9/admin_guide/scheduling/node_affinity.html

Lab: OpenShift: Working with Application Clustering

Overview
Time: 20 Minutes

Prerequisites:

- A computer system with internet connectivity.
- A GitHub account.
- Keep an open browser window logged in to your GitHub account.
- OpenShift Online free account.
- Git software installed.
- Complete pervious OpenShift Labs before this step.
- A text editor of your choice installed to your system.
  - Note for Windows users: Do not use Notepad as the text editor. If you do not have another text editor installed, you should download and install a text editor.
  - Some example text editors:
    - Notepad ++ - https://notepad-plus-plus.org/
    - Atom - https://atom.io/
    - Brackets - http://brackets.io/
    - Visual studio code - https://code.visualstudio.com/
    - Vim text editor - https://www.vim.org/

In this lab, you will:

- Install an example application which uses application clustering.

Notes:

- If the example commands show a $ command prompt, then do not actually type the $.

Initial Preparation:

- Keep a browser window logged in to your GitHub account.
- Open a browser window and log in to your OpenShift Online account and enter the web console.
- Open a terminal (or in Windows a command prompt "as an administrator").
- If you are not logged in to your OpenShift Online account through the command prompt/terminal, then log in using the oc login command copied from your web console.
- Type `oc project` into the command line. It will tell you the name of the project you are working on.
- If you have the free OpenShift Online starter account, then you can only have one project. Delete the project by typing `oc delete project username-project`. Replace username-project with the name of your project.
- Make a new project with a unique name using the following command. Replace username-stateful with the name you want to give the project.

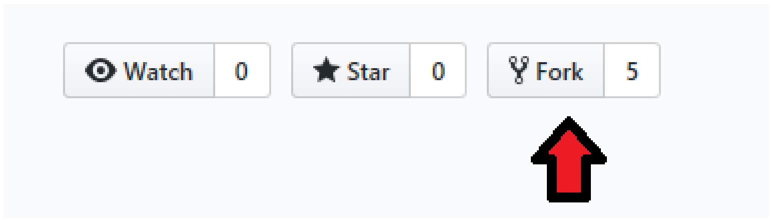
```
       oc new-project username-stateful --display-name="Test Wildfly
Application"
```

Install an Example Application which Uses Application Clustering.

Step 1. Navigate to the following GitHub repository.

https://github.com/elephantscale/openshift-cluster-example

Step 2. Fork the application by clicking the "**Fork**" button on the upper right of the screen.



Step 3. Open your Git Bash or terminal window.

Step 4. You may optionally change directories to the directory where you want to clone the file.

Step 5. In your Git Bash window or terminal window, clone the forked application from your repository. Instead of "username," type your actual GitHub username.

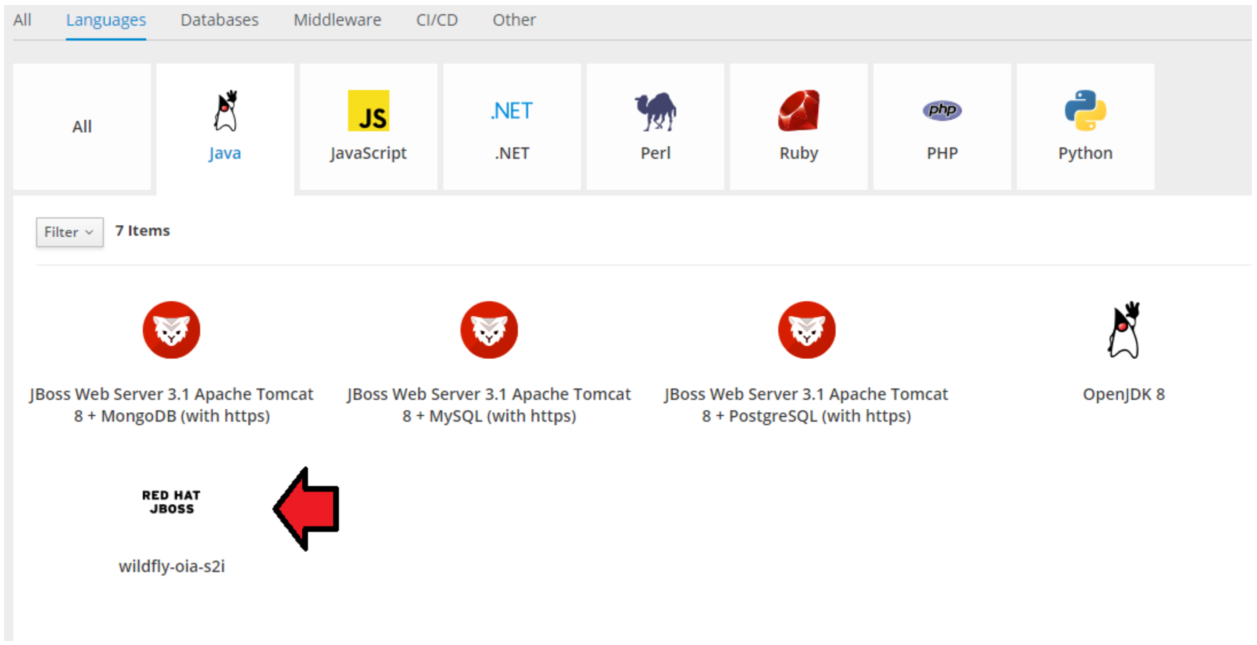
```
       $  git clone https://github.com/username/openshift-cluster-
example.git

$  cd openshift-cluster-example
```

Step 6. Copy the file wildfly-template.yaml to the directory you are working in with your command prompt/terminal. This example would copy the file back two directories.

```
       $ cp wildfly-template.yaml ../../wildfly-template.yaml
```

Step 7. In your command prompt/terminal make sure that the wildfly-template.yaml file is there by typing "dir" (or "ls").


All
Languages
Databases
Middleware
CI/CD
Other
All
Java
JavaScript
.NET
Perl
Ruby
PHP
Python
Filter
7 Items
JBoss Web Server 3.1 Apache Tomcat 8 + MongoDB (with https)
JBoss Web Server 3.1 Apache Tomcat 8 + MySQL (with https)
JBoss Web Server 3.1 Apache Tomcat 8 + PostgreSQL (with https)
OpenJDK 8
RED HAT JBOSS
wildfly-oia-s2i

Step 8. Load the template using the following command.

```
        oc create -f wildfly-template.yaml
```

Step 9. Navigate to your web console, click the project name, and then click on the blue **Browse catalog** button. Next, click on **languages**, **Java**, and then the **wildfly-oia-s2i** option.

Step 10. Click **next**, then change the GitHub source repository URL to the following. Instead of "username" in the URL it should be your GitHub username.

```
https://github.com/username/openshift-cluster-example
```

Step 11. Click on the blue **Create** button and it will start to build your app. Then click on the **Overview** tab on the top left. It will take a few minutes to build the application.

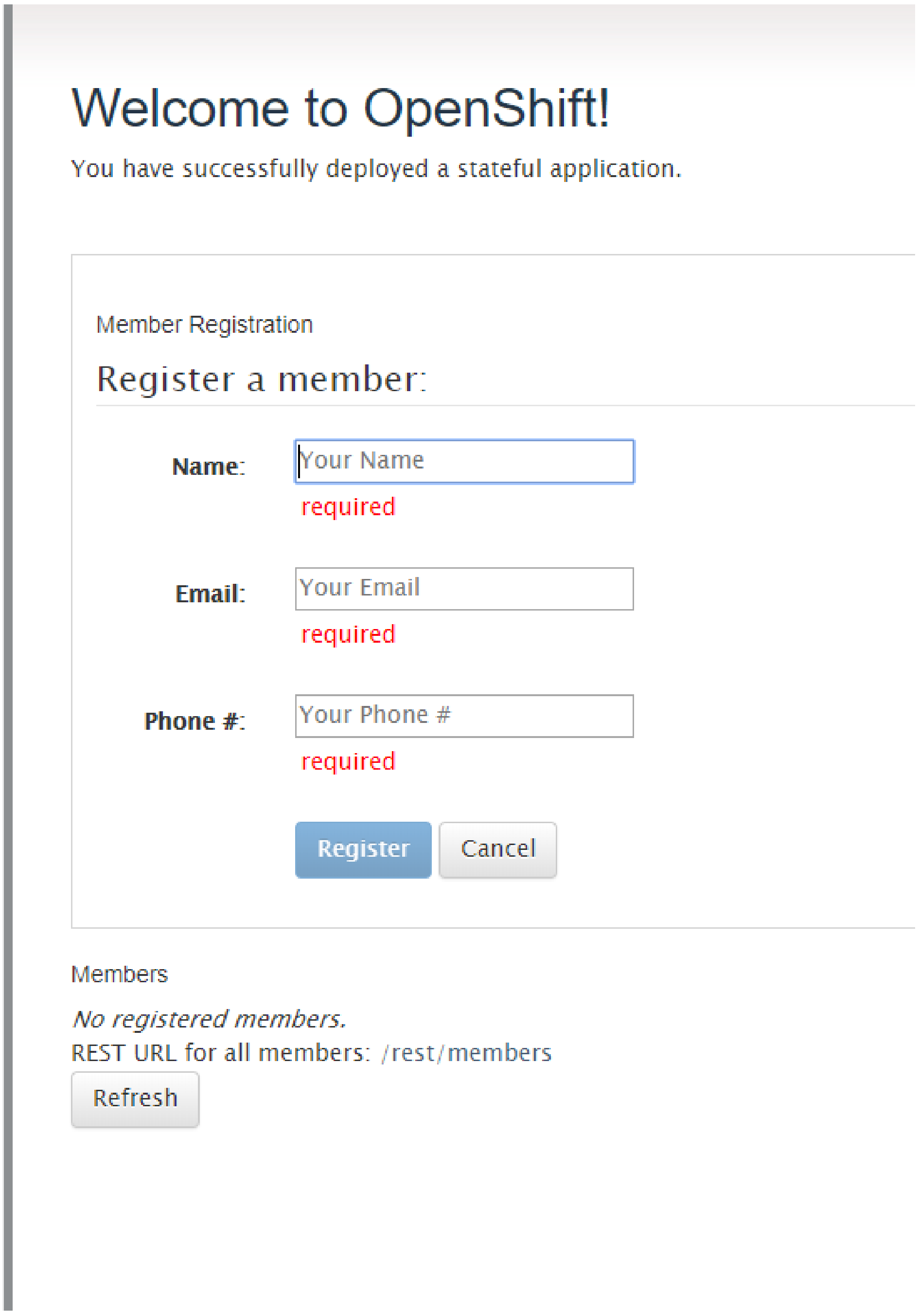
Step 12. Create a readiness probe using the web console using the following specifications. Then click **save**.

- Type: **HTTP GET**
- Path: **/**
- Port: **8080**
- Initial Delay: **20**
- Timeout **1**

Step 13. Use the following command to allow the pods to view the data. There are two places where it says username-stateful; you should replace that with the actual name of your project.

```
        oc policy add-role-to-user view system:serviceaccount:username-
stateful:default -n username-stateful
```

Step 14. When the build is ready, you will see a URL route on the top right on the **Overview** tab. Click on the URL to navigate to the page.


Welcome to OpenShift!

You have successfully deployed a stateful application.

Member Registration

Register a member:

Name:	Your Name required
Email:	Your Email required
Phone #:	Your Phone # required

Members

No registered members.

REST URL for all members: /rest/members

Step 15. In your command prompt, type `oc get pods` and note the name of the running pod.

Step 16. Open up 2 different tabs from the URL route from your app. In one tab, register 3-4 fake members by typing in the details and clicking **Register** for each. Click **Refresh** when you are done.

Step 17. On the duplicate tab, click **Refresh** and the user information you entered on the other tab should show up.

Step 18. In the web console, look under Applications / Deployments. Set the number of replicas to "2".

Step 19. Type `oc get rc` in the CLI to verify the number of replicas running. It should read 2.

Step 20. Type `oc get pods` again and then delete the original pod using the following command. Replace "pod-name" with the actual pod name. Note that you should be deleting the oldest running pod. There should still be another running pod left.

```
        oc delete pod pod-name
```

Step 21. It may take a minute for the system to update, then, open the browser tab where you entered registered members. Click **Refresh**. You should see the same screen as before, with your application in the member list.

In this example, we have configured an application which automatically shared data between pods, without storing it on a PVC.

This example was obtained from "Openshift in Action" chapter 8. For more information, the chapter goes on to describe how to install statefulsets and PVCs.

References:

Example application obtained from "Openshift in Action" chapter 8. * Duncan, J., Osborne, J. (2018). Openshift in Action. Pages 147-167. Manning Publications: Shelter Island, NY. www.allitebooks.com

https://docs.openshift.com/container-platform/3.6/install_config/router/index.html#install-config-router-overview

https://docs.openshift.com/container-platform/3.6/install_config/router/default_haproxy_router.html#install-config-router-default-haproxy

https://docs.openshift.com/container-platform/3.9/admin_guide/scheduling/scheduler.html

https://docs.openshift.com/online/architecture/core_concepts/pods_and_services.html

Lab: OpenShift: Troubleshooting Applications

Overview
Time: 20 Minutes

Prerequisites:

- A computer system with Internet connectivity.
- A GitHub account.
- Keep an open browser window logged in to your GitHub account.
- OpenShift Online free account.
- Git software installed.
- Complete pervious OpenShift Labs before this step.
- A text editor of your choice installed to your system.
  - Note for Windows users: Do not use Notepad as the text editor. If you do not have another text editor installed, you should download and install a text editor.
  - Some example text editors:
    - Notepad ++ - https://notepad-plus-plus.org/
    - Atom - https://atom.io/
    - Brackets - http://brackets.io/
    - Visual studio code - https://code.visualstudio.com/
    - Vim text editor - https://www.vim.org/
- For the optional tutorial, you would need Eclipse or another remote Java application debugging software installed. It would be recommended to already know how to use the debugging software prior to starting the tutorial.

In this lab, you will:

- Install an example Wildfly application.
- View events and logs.
- Use the `oc debug` and `oc rsh` commands.
- Take the optional tutorial about debugging using Eclipse.

Initial Preparation:

- Keep a browser window logged in to your GitHub account.
- Open a browser window and log in to your OpenShift Online account and enter the web console.
- Open a terminal (or in Windows a command prompt "as an administrator").
- If you are not logged in to your OpenShift Online account through the command prompt/terminal, then log in using the `oc login` command copied from your web console.

- Type `oc project` into the command line. The output of this command is the name of the project you are working on.
- If you have the free OpenShift Online starter account, then you can only have one project. Delete the project by typing `oc delete project username-example`. Replace "username-example" with the name of your project.
- Make a new project with a unique name using the following command. Replace "username-project" with the name you want to give the project.

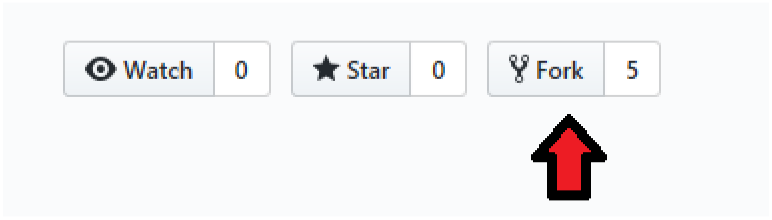
```
        oc new-project username-project --display-name="Test Java
Application"
```

Install an Example Wildfly Application.

Note: If you have already forked and cloned this application, then skip to step 6.

Step 1. Open the browser window to the following GitHub link: https://github.com/elephantscale/jee-start

Step 2. Fork the application by clicking the **Fork** button in the upper-right corner of the screen.



Step 3. Open your Git Bash or terminal window.

Step 4. You may optionally change directories to the directory where you want to clone the file.

Step 5. In your Git Bash window or terminal window, clone the forked application from your repository. Instead of "username," type your actual GitHub username.

```
        $  git clone https://github.com/username/jee-start
 $  cd jee-start
```

Step 6. Install the jee-start application by typing this command in your command prompt/terminal. Replace the "username" in the GitHub URL with your actual GitHub username.

```
       oc new-app openshift/wildfly-101-
centos7~https://github.com/username/jee-start.git --name=jee-start
```

Step 7. Check the status of the new-app by typing `oc status`

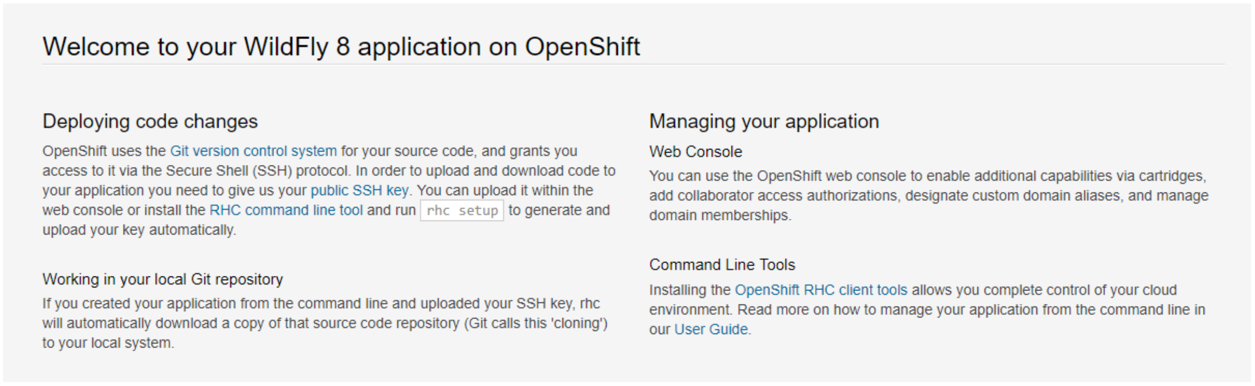
Step 8. Give the application a route with the following command.

```
         oc expose svc/jee-start
```

Step 9. Type `oc get routes` into the command line, and then copy the URL under the HOST/PORT field to your clipboard.

Step 10. Paste the URL into a new web browser window address bar.

Step 11. Note that this application has a hidden HelloWorld link. After the end of the existing URL in the web browser, add `/HelloWorld` and you should get a view of the HelloWorld screen.



Welcome to your WildFly 8 application on OpenShift

Deploying code changes

OpenShift uses the Git version control system for your source code, and grants you access to it via the Secure Shell (SSH) protocol. In order to upload and download code to your application you need to give us your public SSH key. You can upload it within the web console or install the RHC command line tool and run `rhc setup` to generate and upload your key automatically.

Working in your local Git repository

If you created your application from the command line and uploaded your SSH key, rhc will automatically download a copy of that source code repository (Git calls this 'cloning') to your local system.

Managing your application

Web Console

You can use the OpenShift web console to enable additional capabilities via cartridges, add collaborator access authorizations, designate custom domain aliases, and manage domain memberships.

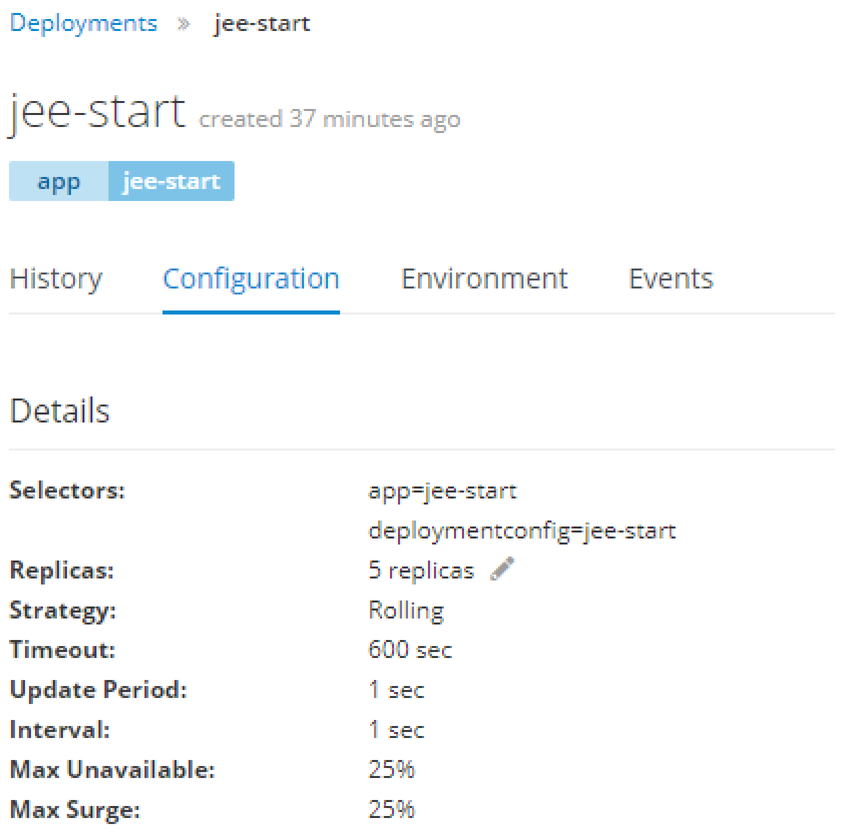
Command Line Tools

Installing the OpenShift RHC client tools allows you complete control of your cloud environment. Read more on how to manage your application from the command line in our User Guide.

View Events and Logs.

Step 1. Navigate to your OpenShift web console and select the project you made.

Step 2. Click **Applications / Deployments**. Then, click the **Event** tab and view the events.



Deployments » jee-start

jee-start created 37 minutes ago

app jee-start

History Configuration Environment Events

Details

Selectors:	app=jee-start
	deploymentconfig=jee-start

Replicas:	5 replicas
Strategy:	Rolling
Timeout:	600 sec
Update Period:	1 sec
Interval:	1 sec
Max Unavailable:	25%
Max Surge:	25%

Step 3. To obtain the list of all events from the command line, type:

```
        oc get ev
```

Step 4. Open your web console window and click the **History** tab. Then click the **View Log** link. This will show you the deployment log. (In case the deployment isn't running, click the **Deploy** button in the upper-right corner first, and then view logs.)

Step 5. View the pod log. Click **Applications** and then click **Pods** in the menu. Then, click the name of a pod that is running. Then, click the **Logs** tab.

Step 6. Look through the logs of the pod. The log will stream new information automatically if it becomes available.

Step 7. Click **Builds** in the upper left, and then click **Builds** in the menu. Then, click the name of the build, which should be "**jee-start**". Then, click the **View Log** link.

Step 8. Scroll down the log file of the Builds log.

Step 9. View the BuildConfig log from the CLI using this command. It will print the log in your command line.

```
        oc logs bc/jee-start
```

Step 10. Save a local copy of the DeploymentConfig log from the CLI using this command:

```
        oc logs dc/jee-start > log.txt
```

Step 11. View the log.txt file using your text editor.

Step 12. View the build log from the CLI.

- Type `oc get builds` to get the name of your build. Replace "jee-start-1" with the actual name of your build.
- Type this command to print the log onto the CLI screen:

```
        oc logs build/jee-start-1
```

Step 13. View a streaming log of a running pod.

- Type `oc get pods` into the CLI and note the name of a running pod.
- Type this command to print a streaming log (which updates automatically) of the pod onto the CLI. Replace "jee-start-3-hnzq2" with the name of your running pod.

```
        oc logs -f pod/jee-start-3-hnzq2
```

Step 14. When you are done viewing it, close the command prompt/window and open a new one (in Windows, open the command prompt "as an administrator").

Using the debug and rsh Commands

- `oc debug` creates a debugging pod to allow you to execute shell commands to the selected resource.

Step 1. In the CLI, type the following command to run the debug command to the DeploymentConfig.

```
        oc debug dc/jee-start
```

Step 2. This will open a shell prompt. Now, type `help` and press **Enter** for a list of commands.

Step 3. Type `exit` and press **Enter** to exit the shell.

- Note that you can also use the `debug` command for other resources and files.
- The `oc rsh` command allows you to run shell commands to a container in a pod.
- Type `oc get pods` and note the name of a running pod.
- Type the following command but replace "jee-start-4-q2pkj" with the name of your pod.

```
oc rsh jee-start-4-q2pkj
```

- Type `exit` to exit the shell.

Optional Tutorial: Debugging Using Eclipse.

- Note that you will need Eclipse or similar remote debugging software for Java installed on your system to complete this tutorial. This tutorial is recommended only if you are familiar with debugging using Eclipse or your other chosen debugging software.
- View the tutorial link in a web browser at this URL: https://blog.openshift.com/debugging-java-applications-on-openshift-kubernetes/
- You have already installed the app used for debugging in this online tutorial. But the name we have used is jee-start.
- Enable the debugging port with this command:

```
oc set env dc/jee-start DEBUG=true
```

- You should have already cloned the jee-start files to your repository. Add the source code to your workspace using Eclipse or other debugging software.
- Open the URL for your installed app by clicking on the URL from the web console.
- After the URL in the web browser add `/HelloWorld` and then press **Enter**.
- Type `oc get pods` to get the name of your running pods.
- Port Forward the name of the running pod using the following command. Replace "jee-start-4-q2pkj" with the actual name of your running pod.

```
oc port-forward jee-start-4-q2pkj 8787:8787
```

- Now, follow the steps in the tutorial in the link above. There is a 1-minute video showing the steps of configuring Eclipse to debug the application. This will show the remaining steps of this tutorial.