# Exercise Notes:

It is recommended to Delete and Re-Provision your Lab environment before working on these exercises. This will reset your lab environment and delete work that has already been done as part of the guided exercises and labs in DO288.

Follow the instructions in the Guided Exercise “Configuring the Classroom Environment” in Chapter 1 of DO288 v4.2 to set up your lab environment.

It is recommended that you start these exercises with a complete reset of your lab environment, github repository, and quay repositories. For Cleanup instructions, see the end of this document.

# Exercise 1 (Exam Question 7)

Expose the OpenShift internal registry outside the cluster and log in as developer to the OpenShift internal registry.

Pull an image from the Openshift to your workbench.

# Exercise 1 - Solution

1. source /usr/local/etc/ocp4.config
2. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}

***Note: The following step is required in the exam but fails in the lab environment with an error "Error from server (Forbidden): …!!!!***

1. oc patch configs.imageregistry.operator.openshift.io/cluster --type merge -p '{"spec":{"defaultRoute":true}}'
2. oc get route -n openshift-image-registry

*=> provides url of openshift image registry*

1. TOKEN=$(oc whoami -t)
2. *HOST=$(oc get route default-route -n openshift-image-registry --template=‘{{.spec.host}}’)*
3. sudo podman login -u ${RHT\_OCP4\_DEV\_USER} -p ${TOKEN} ${HOST}

* *should provide “Login Succeeded!” message*

**sudo podman pull ${HOST}/abc/xyz**

**sudo podman search ${HOST}/http**

*Note:*

*If you do not get a route from the command above*

*try this command:*

*oc auth can-i get routes -n openshift-image-registry*

*command should return “yes”*

*Also you can use:*

*HOST=$(oc get route default-route -n open shift-image-registry —template=‘{{ .spec.host}}’) then $HOST for hostname*

*Can also use this command if you retrieve HOST above*

1. podman login **-u $RHT\_OCP4\_DEV\_USER -p $TOKEN $HOST**

-

# Exercise 2 (Exam Question 1)

Deploy an application based on the following requirements:

• The project name for OpenShift is ***youruser-*apollo**.

• The application name for OpenShift is **lunar**.

• The application should be accessible from the default route:

**lunar-*youruser*-apollo.apps.<*cluster>*.com**

• The Git repository that contains the application sources is:

**https://github.com/*yourgithubuser*/DO288-apps#source-build**

• The context directory in the Git repository is **nodejs-helloworld**.

• Npm modules required to build the application are available from:

**http://nexus-common.apps.*cluster*.com/repository/nodejs**

• Use the npm\_config\_registry build environment variable to pass this information to the S2I builder image for Node.js

• The python -m json.tool filename.json command can be used to identify syntax

errors in JSON files.

# Exercise 2 - Solution

1. Make sure you have DO288-apps forked from Red Hat training into your personal github.com repository.
2. git clone <http://github.com/$RHT_OCP4_GITHUB_USER/DO288-apps>
3. cd DO288-apps
4. git checkout master
5. git checkout -b source-build
6. git push --set-upstream origin source-build
7. cd ~/DO288-apps/nodejs-helloworld
8. python –m json.tool package.json
9. vim package.json **=> fix json by adding ':' after express**

*Note:*

*python –m json.tool will return the line number of the error in package.json, in vim type : and the linenumber to go to the line of the error, line 12*

1. git add .
2. git commit -a -m 'Fixed json error'
3. git push --set-upstream origin source-build
4. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
5. oc new-project ${RHT\_OCP4\_DEV\_USER}-apollo
6. oc new-app -–name=lunar –-build-env npm\_config\_registry=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs/ nodejs:10~http://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps#source-build ––context-dir nodejs-helloworld
7. oc logs –f bc/lunar
8. oc expose svc lunar
9. oc get route **=> provides route**
10. curl *route*

# Exercise 4 (Exam Question 3)

The application image contains a PHP application that displays the web server environment

and the PHP interpreter configuration. Deploy the application as follows:

• Run the command “**lab expose-image start**” to set up this exercise.

• The project that hosts the image stream for the image built outside of OpenShift should be

named ***youruser-*colorado**.

• The application project should be named ***youruser-*texas**.

• Grant pull access to pods from the ***youruser*-texas** project to image streams in the **youruser**-**colorado** project.

• The container image name is php-info. The OCI-formatted image layers and manifest are in the **/home/student/DO288/labs/expose-image/php-info** folder. Push that image into a private image repository in your Quay.io account.

• Add the newly created **php-info** image as an image stream to the OpenShift internal registry.

• Deploy an application to the ***youruser*-texas** project based on the php-info image. The application name should be **dallas.**

• Access the application using the default host name assigned by OpenShift.

• Use the **/usr/local/etc/ocp4.config** configuration file to get classroom configuration data such as the OpenShift cluster’s Master API URL.

# Exercise 4 - Solution

1. lab expose-image start
2. ls ~/DO288/labs/expose-image/php-info *=> verify that container image exists*
3. source /usr/local/etc/ocp4.config
4. **podman login -u ${RHT\_OCP4\_QUAY\_USER} quay.io**
5. **oc create secret generic quayio \**

**–-from-file .dockerconfigjson=${XDG\_RUNTIME\_DIR}/containers/auth.json \**

**--type kubernetes.io/dockerconfigjson**

1. skopeo copy oci:/home/student/DO288/labs/expose-image/php-info docker://quay.io/${RHT\_OCP4\_QUAY\_USER}/php-info
2. skopeo inspect docker://quay.io/${RHT\_OCP4\_QUAY\_USER}/php-info
3. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
4. oc new-project ${RHT\_OCP4\_DEV\_USER}-colorado
5. **oc secrets link default quayio**
6. oc import-image php-info –-confirm **–-reference-policy local** -–from quay.io/${RHT\_OCP4\_QUAY\_USER}/php-info
7. oc get istag
8. oc new-project ${RHT\_OCP4\_DEV\_USER}-texas
9. **oc policy add-role-to-group \**

**-n ${RHT\_OCP4\_DEV\_USER}-colorado system:image-puller \**

**system:serviceaccounts:${RHT\_OCP4\_DEV\_USER}-texas**

1. oc new-app –-name dallas -i ${RHT\_OCP4\_DEV\_USER}-colorado/php-info
2. oc expose svc dallas
3. oc get route
4. curl http://<*route>*

# Exercise 6 (Exam Question 2)

Run an application with customized S2I scripts to copy all html files from the directory /tmp/src/. into the directory ./ and write today’s date into info.html file using format “yyyy-mm-dd” followed by a line with the text “*We have a liftoff!*”.

• Use the builder image **httpd:2.4** available in the Openshift internal registry to deploy the application.

• The Git repository that contains the application sources is:

**https://github.com/*yourgithubuser*/DO288-apps/s2i-scripts**

• The project name for OpenShift is ***youruser-*gemini**.

• The application name for OpenShift is **rocket**.

• The application should be accessible from the default route exposed by Openshift.

• The information page should be available at ***route*/info.html**

# Exercise 6 - Solution

1. source /usr/local/etc/ocp4.config
2. lab s2i-scripts start
3. cd ~/DO288-apps
4. git checkout master
5. git checkout -b s2i-scripts
6. git push –set-upstream origin s2i-scripts
7. cd s2i-scripts
8. chmod +x –Rf .s2i/bin
9. vim ./.s2i/bin/assemble

Add the **cp** to assemble so app code gets copied into dir indicated by exam

Add the date part to assemble and output to info.html

In the exam, use date format “+%Y-%m-%d”

Be sure to use grave accent/back tick above tab key for single quotes and add a space between date and “ or use: **DATE=$(date “+%Y-%m-%d”)**

A screenshot of a social media post

Description automatically generated

1. git commit –a -m “fixed assemble”
2. git push -u origin s2i-scripts
3. oc new-project ${RHT\_OCP4\_DEV\_USER}-gemini
4. oc new-app –-name rocket httpd:2.4~https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps#s2i-scripts –-context-dir s2i-scripts
5. oc expose svc rocket
6. oc get route
7. curl *route*/info.html

# Exercise 7 (Exam Question 6)

**Setup Instructions:**

Before starting this lab, please run the following commands in this sequence from a terminal in your lab environment to setup the required files for this lab:

mkdir ~/exercise7

cd ~/ex

**Download and extract to exercise7 folder using wget or download with browser:** wget <http://content.example.com/courses/do288/ocp4.1/materials/labs/review-template.tgz>

tar -xf review-template.yaml

cp ~/exercise7/review-template/todo-template.yaml ./todo-template.yaml

rm -rf review\*

In this exercise, you will deploy the To Do List application from a template that you will complete for reuse. The application is a To Do List application written in JavaScript. It consists of a web front end, based on the AngularJS web framework, and an HTTP API back end, based on Node.js. The back end uses the Restify and Sequelize frameworks.

Both the application front end and back end run on the same container. The application is deployed from source code stored in a Github repository. A MySQL database is used as the data store. The application initializes the database on startup.

The template takes the following parameters:

• **HOSTNAME**: The hostname used to access the To Do List application.

• **NPM\_PROXY**: The URL of the NPM repository server.

• **SECRET**: The secret for OpenShift webhooks.

• **PASSWORD**: The database connection password. The user name is fixed in the template.

• **CLEAN\_DATABASE**: A flag that indicates whether the application initializes the database on

startup.

• **APP\_GIT\_URL**: The github respository with the source code for DO288.

The template parameters obey the following restrictions:

• The **HOSTNAME, NPM\_PROXY, APP\_GIT\_URL, and CLEAN\_DATABASE** parameters are required.

• The **SECRET and PASSWORD** parameters have random default values unless specified otherwise during application creation.

• The **CLEAN\_DATABASE** parameter has the default value of "true".

In addition, the template should be set up so that it meets the following requirements:

• The template should be displayed in the Openshift console with the following display name: **Resistance To Do List**

• The Template name in OpenShift is **resistance-todo**

• The Template should be created in the project ***youruser-*resistance-common.**

• When the application is deployed, it should use the nodejs:6 and mysql:5.7 images.

• Replace the hardcoded value for the **npm\_config\_registry** with the parameterized value. The required npm module dependencies are downloaded from the following URL available at **http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs**

• The Github repository to pass as a parameter to the template is located at [**https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps**](https://github.com/$%7bRHT_OCP4_GITHUB_USER%7d/DO288-apps)

• The hostname to pass as a parameter to the template is: **${RHT\_OCP4\_DEV\_USER}-todo.${RHT\_OCP4\_WILDCARD\_DOMAIN}**

A starter template file called todo-template.yaml is provided in the **~/exercise7** folder. It contains the resources required to deploy the application.

The resources are in the correct order, and already cleaned up. You are required to add the

missing parameters and add references to the parameters in the resource list.

You must deploy the application according to the following requirements:

• The project name for OpenShift is ***youruser-*yoda**.

• The application name for OpenShift is **skywalker**.

• The application initializes the database on startup.

• The password to access the database is **r2d2**.

The application start up behavior should be as follows:

• If the application is attempted to be started with the command “oc new-app --name skywalker *youruser-*resistance-common/resistance-todo” it should not start andan error message should be displayed.

• If the application is started with the command “oc new-app --name skywalker ${RHT\_OCP4\_DEV\_USER}-resistance-common/resistance-todo -p HOSTNAME=${RHT\_OCP4\_DEV\_USER}-skywalker.${RHT\_OCP4\_WILDCARD\_DOMAIN} -p APP\_GIT\_URL=https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps -p NPM\_PROXY=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs”

it should start and use the values provided in the parameters.

• If the application is started with the command and all parameters it should start:

oc new-app –-name skywalker ${RHT\_OCP4\_DEV\_USER}-common/resistance-todo \

-p APP\_GIT\_URL=https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps \

-p NPM\_PROXY=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs \

-p PASSWORD=r2d2 \

-p CLEAN\_DATABASE=true \

-p HOSTNAME=${RHT\_OCP4\_DEV\_USER}-skywalker.${RHT\_OCP4\_WILDCARD\_DOMAIN}

• If you prefer to test the application with a web browser, use the web front end to add

some entries to the To Do List application interactively. The application user interface is

accessible from the following URL: **http://*youruser*-skywalker.apps.*wildcard-domain*/todo/index.html**

• If you prefer to test the application using the command line, use the following URL to

submit an HTTP GET request to the application back end:

**http://*youruser*-skywalker.apps.*wildcard-domain*/todo/api/items-count**

The reply includes a count attribute, even if there are no entries in the application. An

error reply means the database was not initialized correctly

# Exercise 7 - Solution:

1. vim todo-template.yaml => to edit the template
2. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
3. oc new-project ${RHT\_OCP4\_DEV\_USER}-resistance-common
4. oc create -f todo-template.yaml -n ${RHT\_OCP4\_DEV\_USER}-resistance-common
5. oc policy add-role-to-group \

-n ${RHT\_OCP4\_DEV\_USER}-resistance-common \

system:image-puller \

system:serviceaccounts:${RHT\_OCP4\_DEV\_USER}-yoda

1. oc new-project ${RHT\_OCP4\_DEV\_USER}-yoda

Attempt to start the application with no paramaters. It should fail indicating that HOSTNAME is a required parameter.

1. oc new-app --name skywalker ${RHT\_OCP4\_DEV\_USER}-resistance-common/resistance-todo

A screenshot of a cell phone

Description automatically generated

Attempt to start the application with the required HOSTNAME, APP\_GIT\_URL, and NPM\_PROXY parameters. It should start.

1. oc new-app --name skywalker ${RHT\_OCP4\_DEV\_USER}-resistance-common/resistance-todo -p HOSTNAME=${RHT\_OCP4\_DEV\_USER}-skywalker.${RHT\_OCP4\_WILDCARD\_DOMAIN} -p APP\_GIT\_URL=https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps -p NPM\_PROXY=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs
2. curl *route*/todo/api/items-count

After verification that pervious step worked, delete all application resources.

1. oc get pods => to see a list of running pods and their application name

oc delete all -l app=todoapp

oc delete all -l app=tododb

Attempt to start the application with the parameters as specified in the instructions. It should start.

1. oc new-app –-name skywalker ${RHT\_OCP4\_DEV\_USER}-common/resistance-todo \

-p APP\_GIT\_URL=https://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps \

-p NPM\_PROXY=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs \

-p PASSWORD=r2d2 \

-p CLEAN\_DATABASE=true \

-p HOSTNAME=${RHT\_OCP4\_DEV\_USER}-skywalker.${RHT\_OCP4\_WILDCARD\_DOMAIN}

1. oc get route
2. curl *route*/todo/api/items-count

Sample todo-template.yaml

apiVersion: template.openshift.io/v1

kind: Template

metadata:

annotations:

openshift.io/display-name: Todo Application

description: The Todo application provides an HTTP API back-end and an Angular front-end.

iconClass: icon-nodejs

tags: nodejs,mysql

name: resistance-todo

objects:

- apiVersion: image.openshift.io/v1

kind: ImageStream

metadata:

annotations:

labels:

app: todoapp

name: todoapp

spec:

lookupPolicy:

local: false

- apiVersion: build.openshift.io/v1

kind: BuildConfig

metadata:

annotations:

labels:

app: todoapp

name: todoapp

spec:

failedBuildsHistoryLimit: 5

nodeSelector: null

output:

to:

kind: ImageStreamTag

name: todoapp:latest

postCommit: {}

resources: {}

runPolicy: Serial

source:

contextDir: todo-single

git:

uri: ${APP\_GIT\_URL}

type: Git

strategy:

sourceStrategy:

env:

- name: npm\_config\_registry

value: ${NPM\_PROXY}

from:

kind: ImageStreamTag

name: nodejs:6

namespace: openshift

type: Source

successfulBuildsHistoryLimit: 5

triggers:

- github:

secret: ${SECRET}

type: GitHub

- generic:

secret: ${SECRET}

type: Generic

- type: ConfigChange

- imageChange: {}

type: ImageChange

- apiVersion: apps.openshift.io/v1

kind: DeploymentConfig

metadata:

annotations:

labels:

app: todoapp

name: todoapp

spec:

replicas: 1

selector:

app: todoapp

deploymentconfig: todoapp

strategy:

activeDeadlineSeconds: 21600

resources: {}

rollingParams:

intervalSeconds: 1

maxSurge: 25%

maxUnavailable: 25%

timeoutSeconds: 600

updatePeriodSeconds: 1

type: Rolling

template:

metadata:

annotations:

labels:

app: todoapp

deploymentconfig: todoapp

spec:

containers:

- env:

- name: DATABASE\_NAME

value: tododb

- name: DATABASE\_PASSWORD

value: ${PASSWORD}

- name: DATABASE\_SVC

value: tododb

- name: DATABASE\_USER

value: todoapp

- name: DATABASE\_INIT

value: ${CLEAN\_DATABASE}

imagePullPolicy: Always

name: todoapp

ports:

- containerPort: 8080

protocol: TCP

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

test: false

triggers:

- type: ConfigChange

- imageChangeParams:

automatic: true

containerNames:

- todoapp

from:

kind: ImageStreamTag

name: todoapp:latest

type: ImageChange

- apiVersion: apps.openshift.io/v1

kind: DeploymentConfig

metadata:

annotations:

labels:

app: tododb

name: tododb

spec:

replicas: 1

selector:

app: tododb

deploymentconfig: tododb

strategy:

activeDeadlineSeconds: 21600

resources: {}

rollingParams:

intervalSeconds: 1

maxSurge: 25%

maxUnavailable: 25%

timeoutSeconds: 600

updatePeriodSeconds: 1

type: Rolling

template:

metadata:

annotations:

labels:

app: tododb

deploymentconfig: tododb

spec:

containers:

- env:

- name: MYSQL\_DATABASE

value: tododb

- name: MYSQL\_PASSWORD

value: ${PASSWORD}

- name: MYSQL\_USER

value: todoapp

imagePullPolicy: IfNotPresent

name: tododb

ports:

- containerPort: 3306

protocol: TCP

resources: {}

terminationMessagePath: /dev/termination-log

terminationMessagePolicy: File

volumeMounts:

- mountPath: /var/lib/mysql/data

name: tododb-volume-1

dnsPolicy: ClusterFirst

restartPolicy: Always

schedulerName: default-scheduler

securityContext: {}

terminationGracePeriodSeconds: 30

volumes:

- name: tododb-volume-1

emptyDir: {}

test: false

triggers:

- type: ConfigChange

- imageChangeParams:

automatic: true

containerNames:

- tododb

from:

kind: ImageStreamTag

name: mysql:5.7

namespace: openshift

type: ImageChange

- apiVersion: v1

kind: Service

metadata:

annotations:

labels:

app: todoapp

name: todoapp

spec:

ports:

- name: 8080-tcp

port: 8080

protocol: TCP

targetPort: 8080

selector:

app: todoapp

deploymentconfig: todoapp

sessionAffinity: None

type: ClusterIP

- apiVersion: v1

kind: Service

metadata:

annotations:

labels:

app: tododb

name: tododb

spec:

ports:

- name: 3306-tcp

port: 3306

protocol: TCP

targetPort: 3306

selector:

app: tododb

deploymentconfig: tododb

sessionAffinity: None

type: ClusterIP

- apiVersion: route.openshift.io/v1

kind: Route

metadata:

labels:

app: todoapp

name: todoapp

spec:

host: ${HOSTNAME}

port:

targetPort: 8080-tcp

to:

kind: Service

name: todoapp

weight: 100

wildcardPolicy: None

parameters:

- name: APP\_GIT\_URL

displayName: Aplication source Git URL

description: Git URL where application source is stored

required: true

- name: HOSTNAME

displayName: Aplication Hostname

description: FQDN of the route that gives access the application

required: true

- name: NPM\_PROXY

displayName: Npm proxy server URL

description: URL to a server that provides npm dependencies

required: true

- name: SECRET

displayName: Webhook Secret

description: Secret for webhooks

generate: expression

from: '[a-zA-Z0-9]{40}'

- name: PASSWORD

displayName: Database Password

generate: expression

from: '[a-zA-Z0-9]{16}'

- name: CLEAN\_DATABASE

required: true

value: "true"

# Exercise 9 (Exam Question 4)

The application is written in JavaScript, using the Node.js runtime. You should build and deploy the application to an OpenShift cluster according to the following requirements:

• The project name is ***youruser-*saturn**.

• The application name for OpenShift is **titan**.

• The Git repository that contains the application sources is:

**https://github.com/*yourgithubuser*/DO288-apps#build-app**

• Npm modules required to build the application are available from:

**http://nexus-common.apps.*cluster*.com/repository/nodejs**

* Create a simple python script as follows and place it into the root directory of the git repository. Commit and push the file to git.

# This program prints Hello, world!

print('Hello, world!')

* Make changes to OpenShift so that the python script created in the previous step is invoked every time the application is built as a post-commit build hook. Build the application and verify that the script was invoked.

# Exercise 9 - Solution

1. source /usr/local/etc/ocp4.config
2. cd ~/DO288-apps
3. git checkout master
4. git checkout -b build-app
5. cd build-app
6. touch hello.py
7. Chmod 777 hello.py
8. vim hello.py (add the command to print hello world)
9. git add hello.py
10. git commit -a -m 'Added hello.py'
11. git push -u origin build-app
12. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
13. oc new-project ${RHT\_OCP4\_DEV\_USER}-saturn
14. oc new-app --name titan --build-env npm\_config\_registry=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs nodejs:10~http://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps#build-app --context-dir build-app
15. oc logs -f bc/titan
16. oc expose svc titan
17. oc get route
18. curl http://*route*
19. **oc set build-hook bc/titan --post-commit --command -- python hello.py**
20. oc start-build bc/titan
21. oc logs -f bc/titan

A screenshot of a social media post

Description automatically generated

# Exercise 10 (Exam Question 5)

The application is written in JavaScript, using the Node.js runtime. You should build and deploy the application to an OpenShift cluster according to the following requirements:

• The project name is ***youruser*-jupiter**.

• The application name for OpenShift is **europa**.

• The application should be accessible under the default route exposed by Openshift.

• The Git repository that contains the application sources is:

**https://github.com/*yourgithubuser*/DO288-apps**

• The context directory for the Git repository is **probes**

• Npm modules required to build the application are available from:

**http://nexus-common.apps.*cluster*.com/repository/nodejs**

• The application exposes two HTTP GET URLs at /healthz and /ready. The /

healthz URL will be used by the liveness probe, and the /ready URL will be used

by the readiness probe. The response when these URLs are invoked is an HTTP

status code of 200 if the pod is ready and in a healthy state, or status code 503

otherwise. Use the following values to set up Readiness Probe and Liveness Probe for the application:

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone

Description automatically generated

# Exercise 10 - Solution

1. source /usr/local/etc/ocp4.config
2. cd ~/DO288-apps
3. git checkout master
4. git checkout -b probes
5. cd probes
6. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
7. oc new-project ${RHT\_OCP4\_DEV\_USER}-jupiter
8. oc new-app --name europa --build-env npm\_config\_registry=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs nodejs:10~http://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps --context-dir probes
9. oc logs -f bc/europa
10. oc expose svc europa
11. oc get route
12. curl *route => or use http://…*
13. oc set probe dc/europa --readiness --get-url=http://:8080/ready --initial-delay-seconds=2
14. oc set probe dc/europa --liveness --get-url=http://:8080/healthz --initial-delay-seconds=2 --timeout-seconds=2
15. oc get pods

A screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

# Exercise 12 (Exam Question 4)

In this exercise, you will use configuration maps and secrets to externalize the configuration

for a containerized application.

• The project name is **uranus**.

• The application name for OpenShift is **oberon**.

• The application should be accessible from the default Openshift route.

• The Git repository that contains the application sources is:

**https://github.com/*yourgithubuser*/DO288-apps**

• The context directory for the Git repository is **app-config**

• Npm modules required to build the application are available from:

**http://nexus-common.apps.*cluster*.com/repository/nodejs**

• Inject configuration data into the container using configuration maps and secrets as follows:

1. Create a new configuration map resource called myappconf. Store a key called **APP\_MSG** with the value **"Test Message"** in this configuration map

• The application should be available at **http://oberon.apps.lab.example.com** and you should be able to see the following output:

Value in the APP\_MSG env var is => Test Message

# Exercise 12 - Solution

1. source /usr/local/etc/ocp4.config
2. oc login -u ${RHT\_OCP4\_DEV\_USER} -p ${RHT\_OCP4\_DEV\_PASSWORD} ${RHT\_OCP4\_MASTER\_API}
3. oc new-project ${RHT\_OCP4\_DEV\_USER}-uranus
4. oc new-app --name oberon --build-env npm\_config\_registry=http://${RHT\_OCP4\_NEXUS\_SERVER}/repository/nodejs http://github.com/${RHT\_OCP4\_GITHUB\_USER}/DO288-apps --context-dir app-config
5. oc logs -f bc/oberon
6. oc create configmap myappconf --from-literal APP\_MSG=”Test Message”
7. oc describe configmap/myappconf
8. oc set env dc/oberon --from configmap/myappconf
9. cd ~/app-config
10. oc status
11. oc expose svc oberon
12. oc get route
13. curl *route*
14. oc get configmap myappconf -o json > \

/home/student/greeterconf.json

CLEANUP

Perform the following steps to reset the lab environment, github, and quay.io to its original state.

1. Log in to the Openshift master and manually delete all projects

Note: "DELETE LAB" does not reset the Openshift environment on AWS.

A screenshot of a social media post

Description automatically generated

1. Perform a reset of the lab environment by selecting "DELETE LAB" on the Online Lab course webpage.

A screenshot of a cell phone

Description automatically generated

1. Delete the github repository

<https://help.github.com/en/github/administering-a-repository/deleting-a-repository>

Alternatively, simply rename this repository

1. Delete quay repositories

Log on to Quay

Select the repository to delete

Go to settings

Click "Delete Repository"