IBM Cloud

Introduction to Containers and Kubernetes with IBM Cloud Private (ICP)

Hands-on Workshop

Lab Guide

© Copyright IBM Corp. 2018. All rights reserved

Notices and Disclaimers

© Copyright IBM Corporation 2018.

The information contained in these materials is provided for informational purposes only, and is provided AS IS without warranty of any kind, express or implied. IBM shall not be responsible for any damages arising out of the use of, or otherwise related to, these materials. Nothing contained in these materials is intended to, nor shall have the effect of, creating any warranties or representations from IBM or its suppliers or licensors, or altering the terms and conditions of the applicable license agreement governing the use of IBM software. References in these materials to IBM products, programs, or services do not imply that they will be available in all countries in which IBM operates. This information is based on current IBM product plans and strategy, which are subject to change by IBM without notice. Product release dates and/or capabilities referenced in these materials may change at any time at IBM's sole discretion based on market opportunities or other factors, and are not intended to be a commitment to future product or feature availability in any way.

This document is current as of the initial date of publication and may be changed by IBM at any time. Not all offerings are available in every country in which IBM operates.

IBM, the IBM logo and ibm.com are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both. If these and other IBM trademarked terms are marked on their first occurrence in this information with a trademark symbol (® or ™), these symbols indicate U.S. registered or common law trademarks owned by IBM at the time this information was published. Such trademarks may also be registered or common law trademarks in other countries. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at ibm.com/legal/copytrade.shtml

Other company, product and service names may be trademarks or service marks of others

Document Revision History

Rev #	File Name	Date
1.0	Docker and Kubernetes - Lab	1/21/2018
2.0	Docker and Kubernetes - Lab	1/26/2018
3.0	Docker and Kubernetes - Lab	2/1/2018

Prepared & Revised by:
Louis V. Frolio - louis.frolio@ibm.com
David Solomon - dlsolomo@us.ibm.com

Table of Contents

© Copyright IBM Corp. 2018. All rights reserved

Lab Environment Overview

Environment	Access
Docker and Kubectl on Linux	ssh 169.46.99.30
IBM Cloud Private	http://169.46.33.190:8443

© Copyright IBM Corp. 2018. All rights reserved

Section 1: Container Basics

Purpose:

Throughout this lab, we will be using a sample application, a variation of the mobile game 2048. You will see how we create a Docker image from this application and run it as a container.

In later sections of this lab, you will learn how to deploy this container into a Kubernetes cluster on IBM Cloud Private.

This section introduces container basics. You will learn how to create, run, inspect and manage containers. Also, you will work through establishing console access within the container.

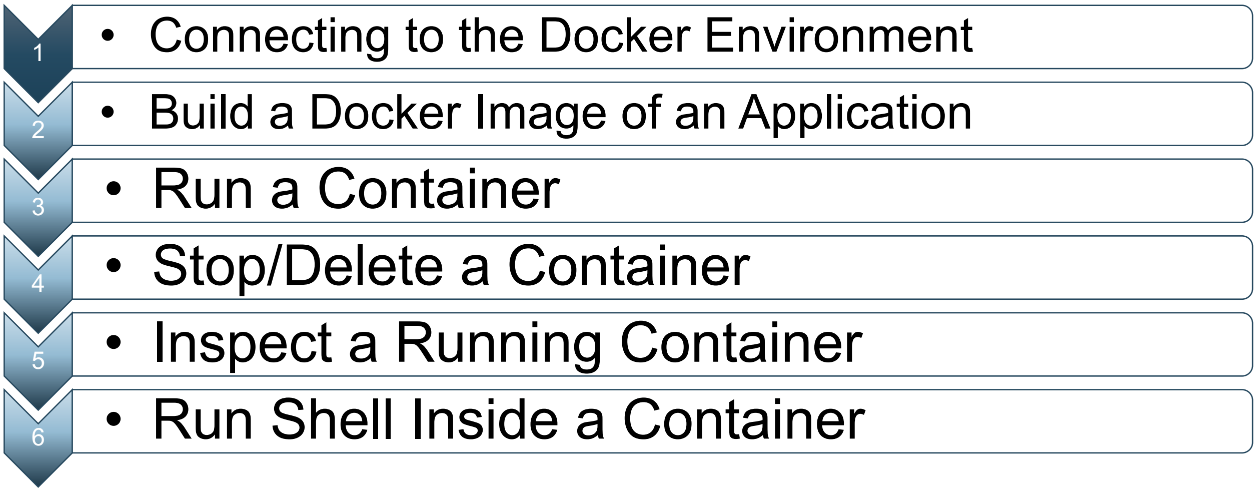
Your lab instructor will assign you a unique username. When you see ***<your username>*** in the instructions, please substitute with your assigned username.

Tasks:

Tasks you will complete in this lab exercise include:

- Connect to the Docker environment
- Creating a Docker Image for an Application
- Running containers
- Inspecting containers
- Container process monitoring
- Container shell access

Section 1: Lab Workflow Overview



1. Connecting to the Docker Environment
2. Build a Docker Image of an Application
3. Run a Container
4. Stop/Delete a Container
5. Inspect a Running Container
6. Run Shell Inside a Container

© Copyright IBM Corp. 2018. All rights reserved

Section 1: Lab Instructions

Step | Action

1 **Login to the Docker Environment**

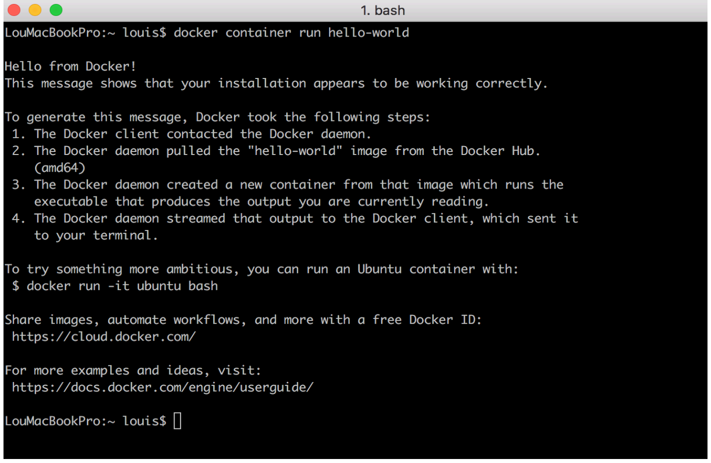
a. Our Docker environment is on a cloud hosted Linux server. In order to access this server, you will need to open an **ssh** session using either Putty (on Windows) or a terminal window (Mac or Linux) to the following address, port number, and user:

Server IP Address- 169.46.99.30
Port= 2222
Username= <your username> (e.g., user01 (if your number is 01))
Password= passw0rd

b. Once logged in, confirm that you can access Docker by running the following command:

```
~$ docker container run hello-world
```

Verify that the output is similar to the following:



```
LouMacBookPro:~ louis$ docker container run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
 1. The Docker client contacted the Docker daemon.
 2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
    (amd64)
 3. The Docker daemon created a new container from that image which runs the
    executable that produces the output you are currently reading.
 4. The Docker daemon streamed that output to the Docker client, which sent it
    to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
 $ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
 https://cloud.docker.com/

For more examples and ideas, visit:
 https://docs.docker.com/engine/userguide/

LouMacBookPro:~ louis$
```

2 **Build a Docker Image for an Application**

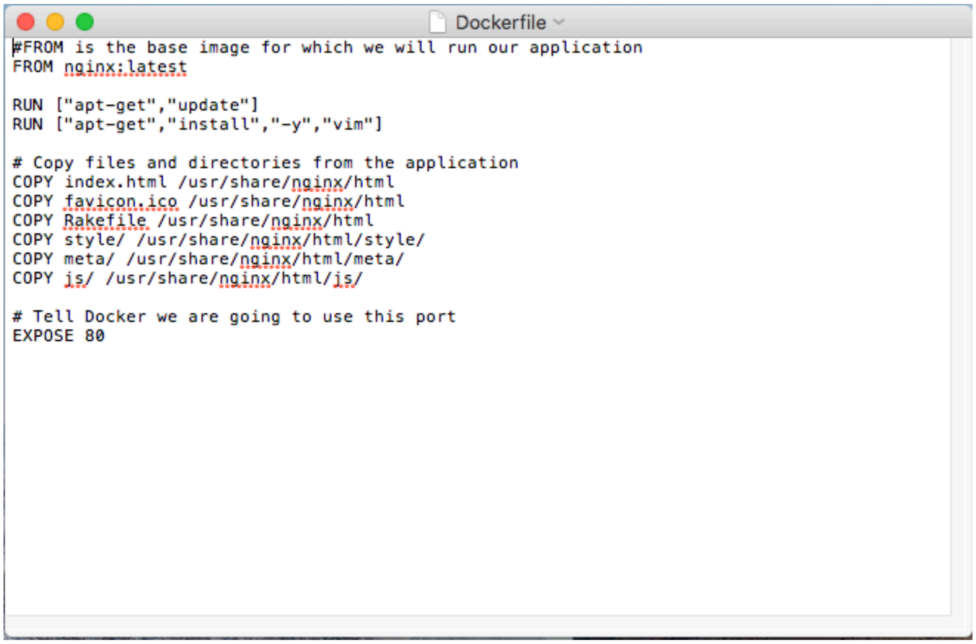
a. Before we can work with a container, we will need to first build an image for our 2048 application. First, we will make of copy of the application code to your home directory:

~$ cp -R /labs/2048_master . (don’t forget the “.” at the end)
~$ cd 2048_master

© Copyright IBM Corp. 2018. All rights reserved

Step	Action

b. These files are the application code required to run the game. Notice there is a file called “Dockerfile” in the top directory of the unzipped files. The Dockerfile is the file you create that instructs Docker how to create and package the application into a Docker image. In this case, the file has already been created for you. Open the file and browse its contents. It will look similar to the figure below:



```
#FROM is the base image for which we will run our application
FROM nginx:latest

RUN ["apt-get","update"]
RUN ["apt-get","install","-y","vim"]

# Copy files and directories from the application
COPY index.html /usr/share/nginx/html
COPY favicon.ico /usr/share/nginx/html
COPY Rakefile /usr/share/nginx/html
COPY style/ /usr/share/nginx/html/style/
COPY meta/ /usr/share/nginx/html/meta/
COPY js/ /usr/share/nginx/html/js/

# Tell Docker we are going to use this port
EXPOSE 80
```

The commands in this file instruct Docker to use a simple web service (nginx) as a base image (nginx is automatically pulled from Docker Hub when the image is built. The file then copies the application code into a directory structure within the image (in /usr/share). Finally, port 80 is exposed in order to enable access to the game from our Web Browser.

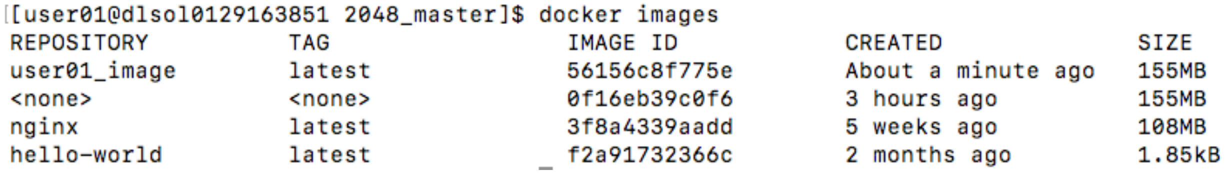
c. Now you can build the image by running the following command:

~ $ docker build -t <your username>_image . *(don’t forget the “.” at the end)*

d. Docker will now build the image. You can confirm this by running the following command and observing that an image named “<*your username*>*_image*” is listed:

~$ docker images

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td></td><td>



```
[user01@dlsol0129163851 2048_master]$ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED              SIZE
user01_image        latest              56156c8f775e        About a minute ago   155MB
<none>              <none>              0f16eb39c0f6        3 hours ago          155MB
nginx               latest              3f8a4339aadd        5 weeks ago          108MB
hello-world         latest              f2a91732366c        2 months ago         1.85kB
```

You have now successfully taken an existing application and created a Docker image from it.

</td></tr>
<tr><td>3</td><td>

Run a Container

a. Now that you have an image, we will now run the 2048 application as a container. To do this, run the following command:

Your instructor will assign you a port a unique port number to use for the remained of the lab.

~$ docker container run --name <*your username*>_*container* -p <*your port*>:80 <your username>_*image*

The container you just created is an instance of your image running as a process. There is no limit to the number of containers that can be run from an image.

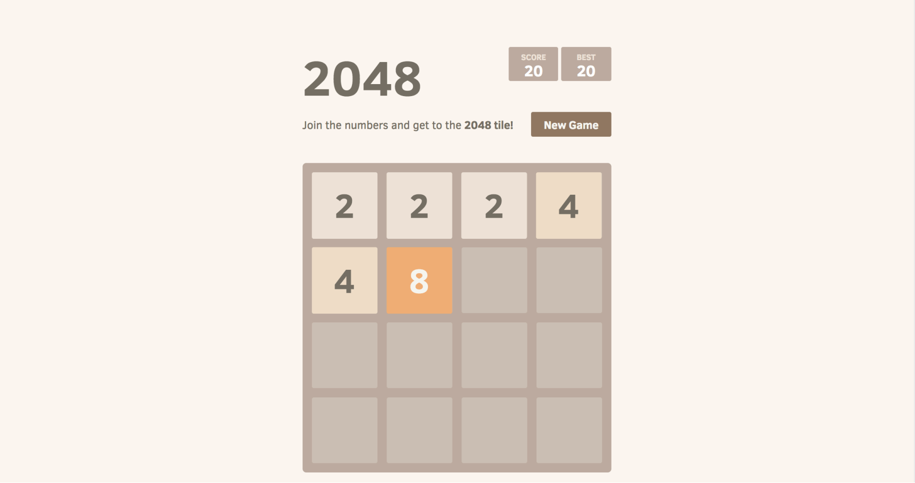
Commands:
--name – Specify a unique name for the container service. If omitted Docker will create a random, human readable name.
-p – Specify that the container internal port (80) be exposed to <your port> on the host.

b. Open a browser and navigate to: 169.46.99.30:<your port>. A page will open with the game, as shown below:

</td></tr>
</table>

© Copyright IBM Corp. 2018. All rights reserved

Step	Action



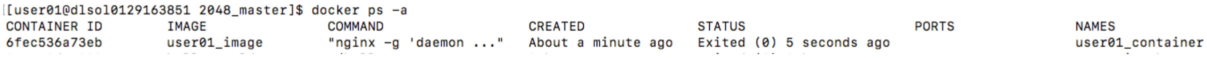
You have now successfully run your first container!!

3 Stop/Delete a Container

a. You can stop the container by typing cntrl-c
~$ <Cntrl-c>

b. Verify that the container is no longer running:
~$ docker container ps

c. Although the container is not running it still exists:
~$ docker container ps -a



```
[user01@dlsol0129163851 2048_master]$ docker ps -a
CONTAINER ID        IMAGE               COMMAND                  CREATED              STATUS                      PORTS               NAMES
6fec536a73eb        user01_image        "nginx -g 'daemon ..."   About a minute ago   Exited (0) 5 seconds ago                        user01_container
```

-a, --all: Show all containers (default shows just running)

d. Remove the container:
~$ docker container rm *<your username>_container*

Containers can be removed either by their name or container id

4 Inspect a Running Container

a. Run a new Docker container for the game:
~$ docker run --publish <your port>:80 --detach --name <your username>_container <your username>_image

Step	Action
	You should be brought back to the terminal prompt (the “detach” option runs the container as a background process) b. Open a browser and navigate to “169.46.99.30:<your port>”. You should be prompted with the game again. c. You can run a variety of commands to get information on the status of a running container. These commands. can be useful when troubleshoot an environment or application. For example, inspecting the meta-data for running container: ~$ docker container inspect <your username>_container and, Stream live performance container metrics: ~$ docker container stats <your username>_container d. Clean up ~$ docker container rm -f <your username>_container Commands: **-d, --detach** - Run the container in the background.
5	**Run Shell Inside a Container** a. We can also directly access a container via a command shell. It allows you to directly login to the container’s command prompt; enabling you to troubleshoot application issues or update the content of a running container. First run the container again: ~$ docker container run --name <your username>_container -d -p <your port>:80 <your username>_image b. Next, we will use the following command to open a shell prompt into the container: ~$ docker exec -it <your username>_container bash

© Copyright IBM Corp. 2018. All rights reserved

Step	Action
	c. Run Linux commands in container: For example, # ls -tal // List directories and files. # exit // Exit shell d. Delete the container: ~$ docker rm -f <your username>_container Commands: -i - Run interactively -t - Create pseudo tty -a - Attach to STDIN, STDOUT or STDERR exec - Run a command in a running container run - Run a command in a new container

© Copyright IBM Corp. 2018. All rights reserved

Section 1: Lab Summary

In this section you learned how to create new containers based on images stored in Docker Hub. You also learned how to interact with containers both from the outside (top, inspect, stats, …), and from the inside (docker exec and run). Access to the Docker service via tty was demonstrated and you learned how to run Linux commands inside the container just as if you were working with a Linux OS.

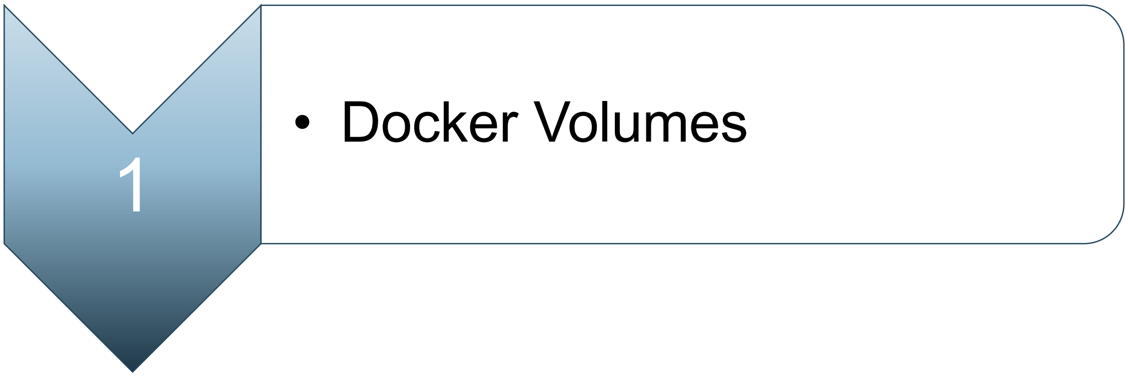
© Copyright IBM Corp. 2018. All rights reserved

Section 2: Data Persistence in Docker

Purpose:	In this section, you will see one method of how data from a container can be persisted, even after a container is removed. Unless such persistence is established, any changes made to a container’s data are deleted once the container is deleted. The method we will use below is Docker Volumes. With Volumes, Docker controls a location for persistent storage on your local machine that persists once a container is deleted.

Tasks:	Tasks you will complete in this lab exercise include: • Create and work with Docker volumes

Section 2: Lab Workflow Overview



1

- Docker Volumes

© Copyright IBM Corp. 2018. All rights reserved

Section 2: Lab Instructions

Step	Action
1	**Docker Volumes**

a. Let’s run our game application in a new container, except this time we will include an option (-v (or volume)) to instruct Docker to persist the content of a specific directory on your local machine:

```
~$ docker container run -d --name <your username>_container -p <your port>:80 -v myvol:/usr/share/nginx/html <your username>_image
```

b. Open bash shell on container and navigate the /usr/share/nginx/html directory:

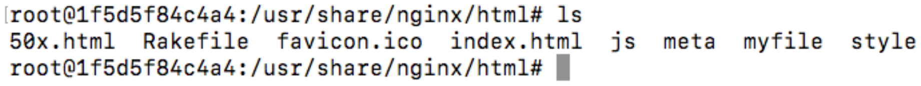
```
~$ docker container exec -it <your username>_container bash
# cd /usr/share/nginx/html
```

c. Create a new file in the html folder containing the phrase, “This is my file”.

```
# echo “This is my file” > myfile
```

Confirm the file “myfile” is listed in the directory and exit the container.

```
# ls
```



```
root@1f5d5f84c4a4:/usr/share/nginx/html# ls
50x.html  Rakefile  favicon.ico  index.html  js  meta  myfile  style
root@1f5d5f84c4a4:/usr/share/nginx/html# 
```

```
# exit
```

d. We will now remove the container using the command:

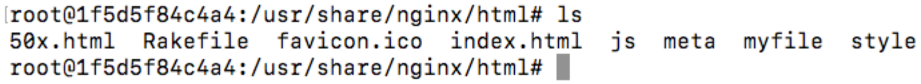
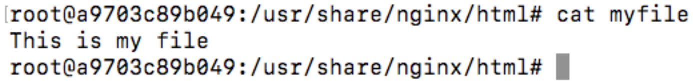
```
~$ docker rm -f <your username>_container
```

e. Now, we can create a new container, referencing the persistent volume and confirm that our file is still present:

```
~$ docker container run -d --name <your username>_container -p 8080:80 -v myvol:/usr/share/nginx/html <your username>_image

~$ docker container exec -it <your username>_container bash
```

Step	Action

	# cd /usr/share/nginx/html
	# ls
	
	`root@1f5d5f84c4a4:/usr/share/nginx/html# ls`
	`50x.html  Rakefile  favicon.ico  index.html  js  meta  myfile  style`
	`root@1f5d5f84c4a4:/usr/share/nginx/html#`
	# cat myfile
	
	`root@a9703c89b049:/usr/share/nginx/html# cat myfile`
	`This is my file`
	`root@a9703c89b049:/usr/share/nginx/html#`
	Volumes are extremely useful for local development projects. You can maintain several volumes to which you can attach a new directory or database that fits a specific purpose.

Section 2: Lab Summary

In this lab you were introduced to one way to persist data on the host file system. With volumes the container references a volume object on the local file system.

© Copyright IBM Corp. 2018. All rights reserved

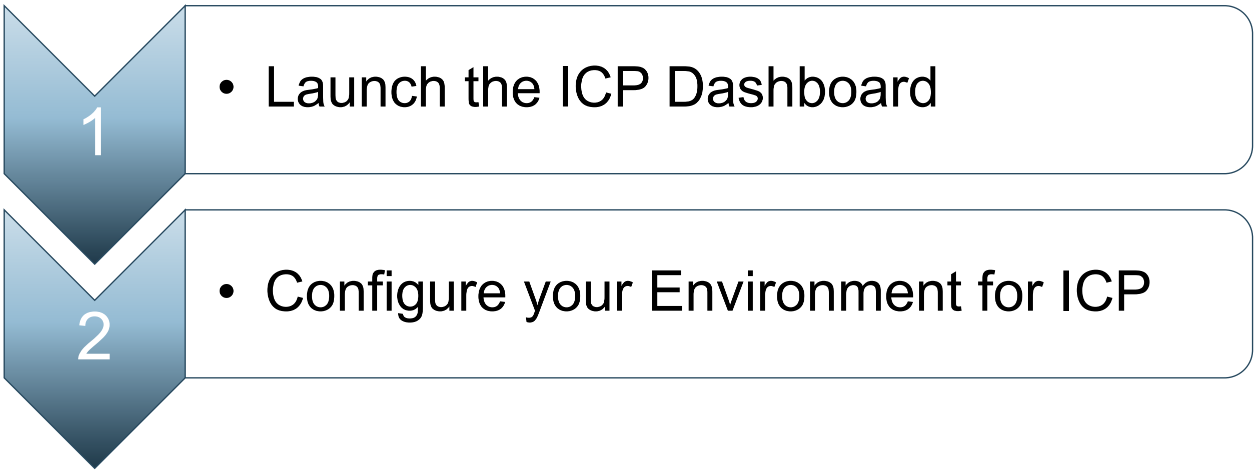
Section 3: Getting Started with Kubernetes in IBM Cloud Private

Purpose:	In this lab you will learn how to configure your environment to work with a Kubernetes cluster within IBM Cloud Private (ICP)

Tasks:	Tasks you will complete in this lab exercise include: • Access the IBM Cloud Private Dashboard • Access the ICP Kubernetes configuration settings • Configure your environment to use the ICP cluster

© Copyright IBM Corp. 2018. All rights reserved

Section 3: Getting Started with Kubernetes in IBM Cloud Private



1

- Launch the ICP Dashboard

2

- Configure your Environment for ICP

© Copyright IBM Corp. 2018. All rights reserved

Section 3: Lab Instructions

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>

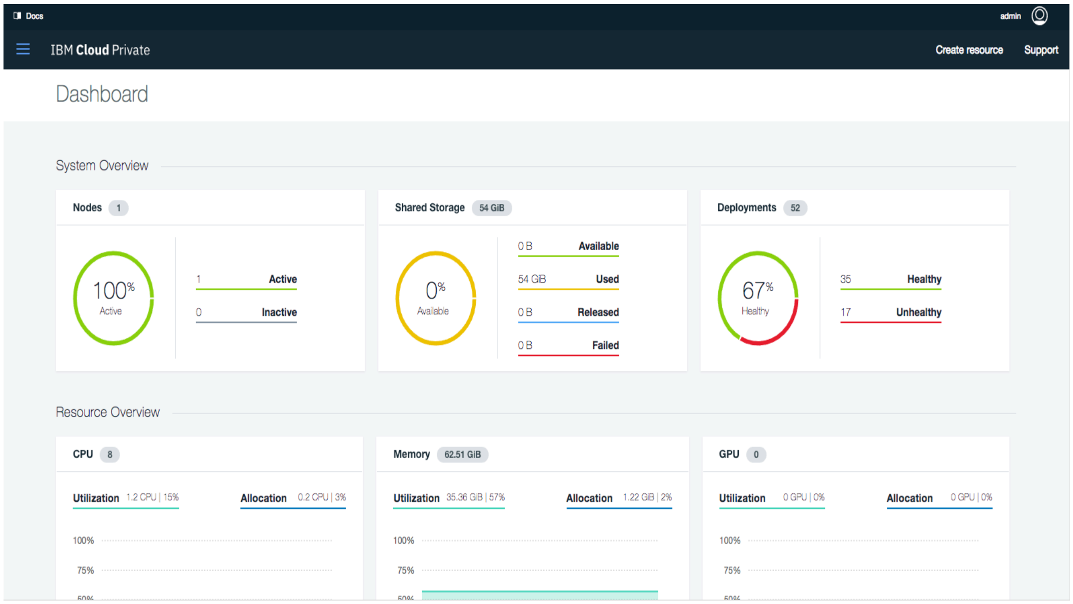
Launch the ICP Dashboard

a. ICP has a centralized dashboard and control center. This dashboard is similar to the classic Kubernetes dashboard but provides additional enterprise services and features (e.g, data science, security).

Open a browser and navigate to the following URL to open the dashboard:

https://169.46.33.190:8443/

Login with username: admin/ password: admin. Your will then be brought to the main ICP overview page, as shown below:



Docs

admin

IBM Cloud Private

Create resource

Support

Dashboard

System Overview

Nodes 1

100% Active

1 Active

0 Inactive

Shared Storage 54 GiB

0% Available

0 B Available

54 GiB Used

0 B Released

0 B Failed

Deployments 52

67% Healthy

35 Healthy

17 Unhealthy

Resource Overview

CPU 8

Utilization 1.2 CPU | 15%

Allocation 0.2 CPU | 3%

Memory 62.51 GiB

Utilization 35.36 GiB | 57%

Allocation 1.22 GiB | 2%

GPU 0

Utilization 0 GPU | 0%

Allocation 0 GPU | 0%

You will notice that this ICP instance is a basic 1-node Kubernetes cluster.

</td></tr>
<tr><td>2</td><td>

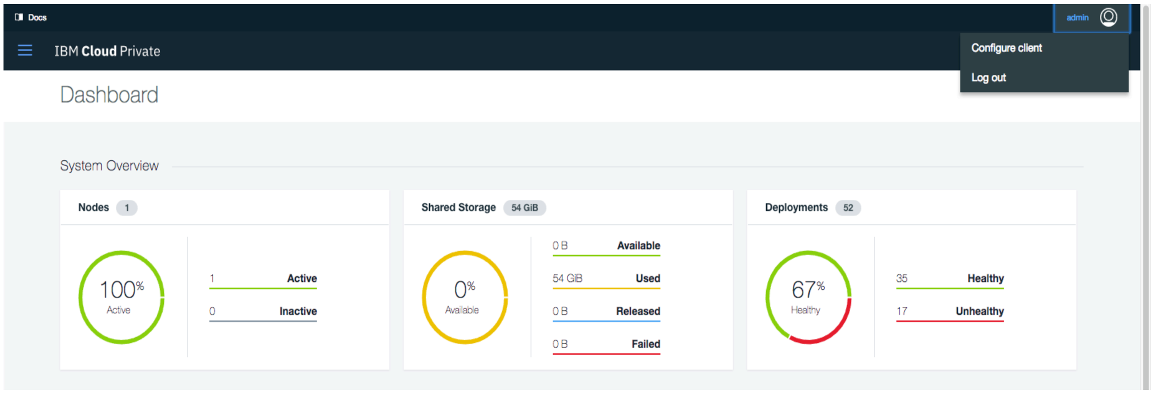
Configure your Environment for ICP

a. In order to interact with and control the ICP cluster remotely using kubectl, you will need to first configure your environment to direct all kubectl commands to the ICP cluster. Fortunately, ICP helps with this by quickly providing the appropriate configuration settings for the cluster.

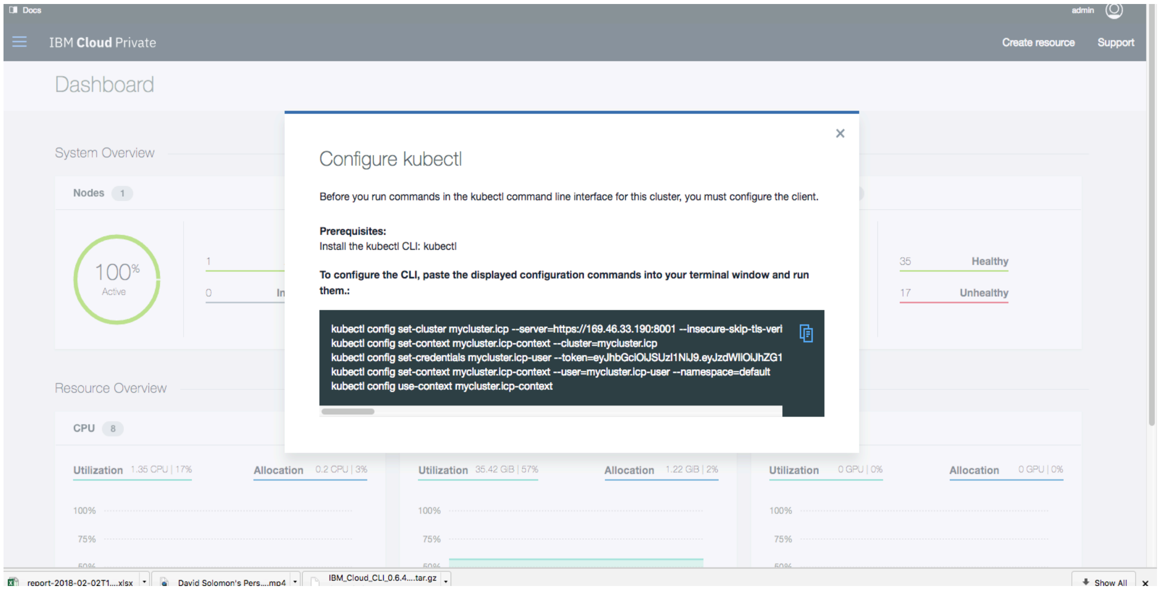
</td></tr>
</table>

© Copyright IBM Corp. 2018. All rights reserved

Step	Action
	On the ICP Dashboard, click on the word “admin” at the top left of the page next to the symbol. You will then see two options, “Configure Client” and “Logout”. Select “Configure Client”.



Once selected, a dialog box called “Configure kubectl” will appear. This box contains the commands that need to be run in your local environment (the Linux environment we used for the Docker portion of this Lab) in order to properly configure kubectl to interact with the ICP cluster.



Configure kubectl

Before you run commands in the kubectl command line interface for this cluster, you must configure the client.

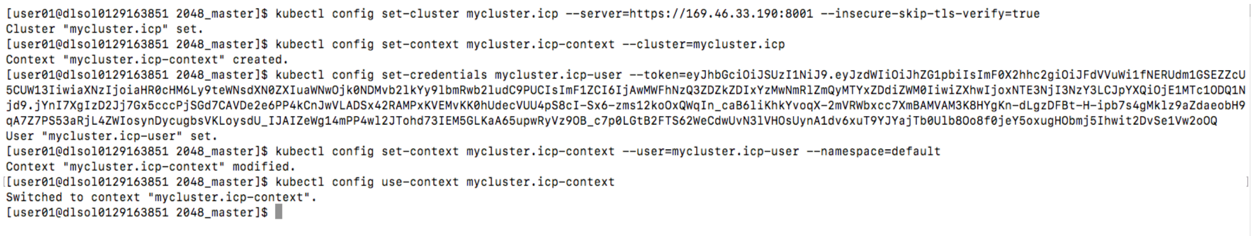
Prerequisites:
Install the kubectl CLI: kubectl

To configure the CLI, paste the displayed configuration commands into your terminal window and run them.:

```
kubectl config set-cluster mycluster.icp --server=https://169.46.33.190:8001 --insecure-skip-tls-veri
kubectl config set-context mycluster.icp-context --cluster=mycluster.icp
kubectl config set-credentials mycluster.icp-user --token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1
kubectl config set-context mycluster.icp-context --user=mycluster.icp-user --namespace=default
kubectl config use-context mycluster.icp-context
```

Now, copy these commands (either manually or using the blue copy symbol in the dialog box).

Step	Action

	b. Now, copy these commands (either manually or using the blue copy symbol on the upper right of the dialog box).
	c. Return to your terminal session to our Linux server and paste these commands at a command prompt (you may need to press Return for the last command to run).
	
	`[user01@dlsol0129163851 2048_master]$ kubectl config set-cluster mycluster.icp --server=https://169.46.33.190:8001 --insecure-skip-tls-verify=true`
	`Cluster "mycluster.icp" set.`
	`[user01@dlsol0129163851 2048_master]$ kubectl config set-context mycluster.icp-context --cluster=mycluster.icp`
	`Context "mycluster.icp-context" created.`
	`[user01@dlsol0129163851 2048_master]$ kubectl config set-credentials mycluster.icp-user --token=eyJhbGciOiJSUzI1NiJ9.eyJzdWIiOiJhZG1pbiIsImF0X2hhc2giOiJFdVVuWi1fNERUdm1GSEZZcU`
	`5CUW13IiwiaXNzIjoiaHR0cHM6Ly9teWNsdXN0ZXIuaWNwOjk0NDMvb2lkYy9lbmRwb2ludC9PUCIsImF1ZCI6IjAwMWFhNzQ3ZDZkZDIxYzMwNmRlZmQyMTYxZDdiZWM0IiwiZXhwIjoxNTE3NjI3NzY3LCJpYXQiOjE1MTc1ODQ1N`
	`jd9.jYnI7XgIzD2Jj7Gx5cccPjSGd7CAVDe2e6PP4kCnJwVLADSx42RAMPxKVEMvKK0hUdecVUU4pS8cI-Sx6-zms12koOxQWqIn_caB6liKhkYvoqX-2mVRWbxcc7XmBAMVAM3K8HYgKn-dLgzDFBt-H-ipb7s4gMklz9aZdaeobH9`
	`qA7Z7PS53aRjL4ZWIosynDycugbsVKLoysdU_IJAIZeWg14mPP4wl2JTohd73IEM5GLKaA65upwRyVz9OB_c7p0LGtB2FTS62WeCdwUvN3lVHOsUynA1dv6xuT9YJYajTb0Ulb8Oo8f0jeY5oxugHObmj5Ihwit2DvSe1Vw2oOQ`
	`User "mycluster.icp-user" set.`
	`[user01@dlsol0129163851 2048_master]$ kubectl config set-context mycluster.icp-context --user=mycluster.icp-user --namespace=default`
	`Context "mycluster.icp-context" modified.`
	`[user01@dlsol0129163851 2048_master]$ kubectl config use-context mycluster.icp-context`
	`Switched to context "mycluster.icp-context".`
	`[user01@dlsol0129163851 2048_master]$`
	You have now successfully configured your environment to start working with Kubernetes and IBM Cloud Private.

Section 3: Lab Summary

In this section, you learned how to access the ICP Dashboard and setup a your environment to interact with a Kubernetes cluster on ICP.

© Copyright IBM Corp. 2018. All rights reserved

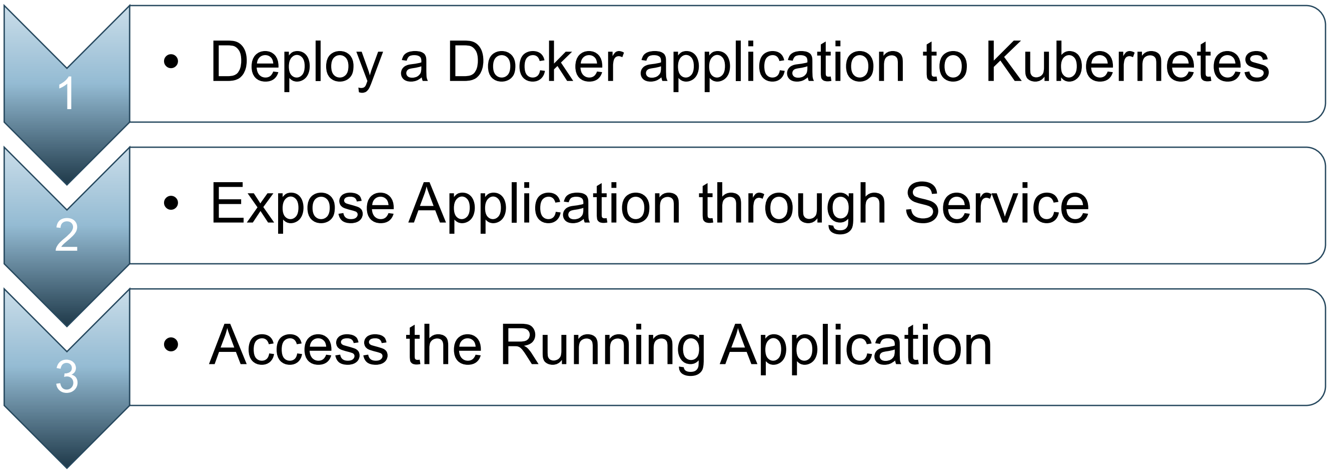
Section 4: Deploy your Application to Kubernetes

Purpose:	In this lab you will learn how to deploy an application to Kubernetes.

Tasks:	Tasks you will complete in this lab exercise include: • Deploy a Docker application to Kubernetes • Expose the application through a service • Access the running application

© Copyright IBM Corp. 2018. All rights reserved

Section 4: Deploy an Application to Kubernetes



- Deploy a Docker application to Kubernetes
- Expose Application through Service
- Access the Running Application

© Copyright IBM Corp. 2018. All rights reserved

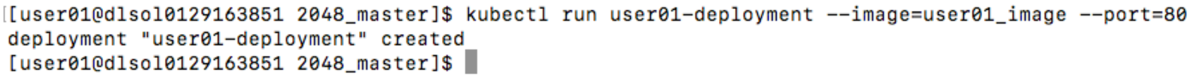
Section 4: Lab Instructions

Step	Action
1	**Deploy a Docker application to the Kubernetes cluster**

a. We will now deploy the same 2048 game application to your cluster. To do this, enter the following command to create a new deployment, using the you previously built.

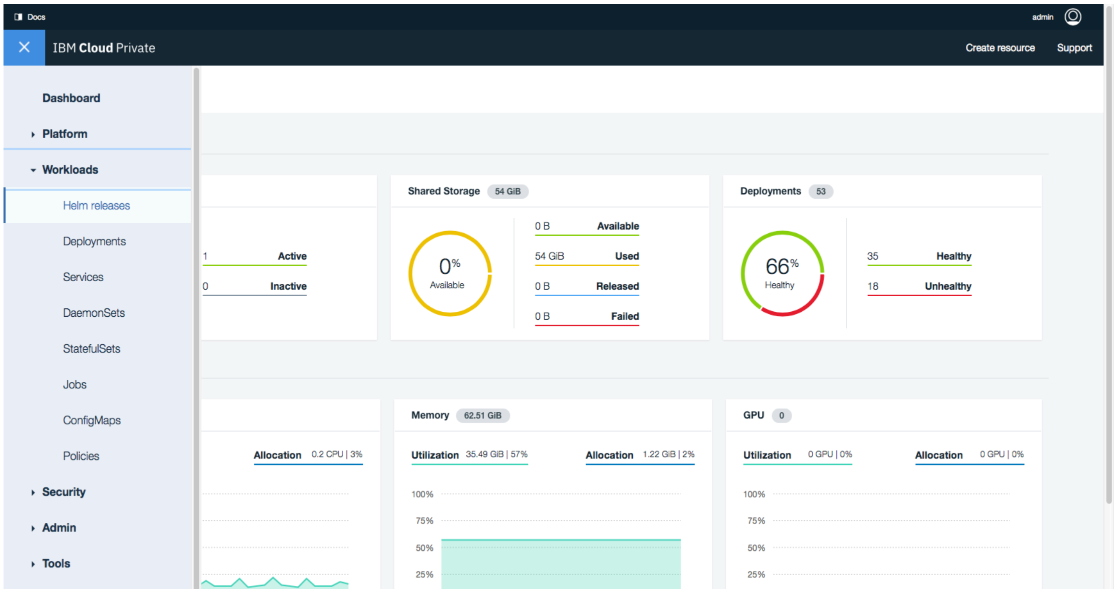
~$ kubectl run <your username>-deployment --image=<your username>_image --port=80

b. Confirm the output is as shown below:



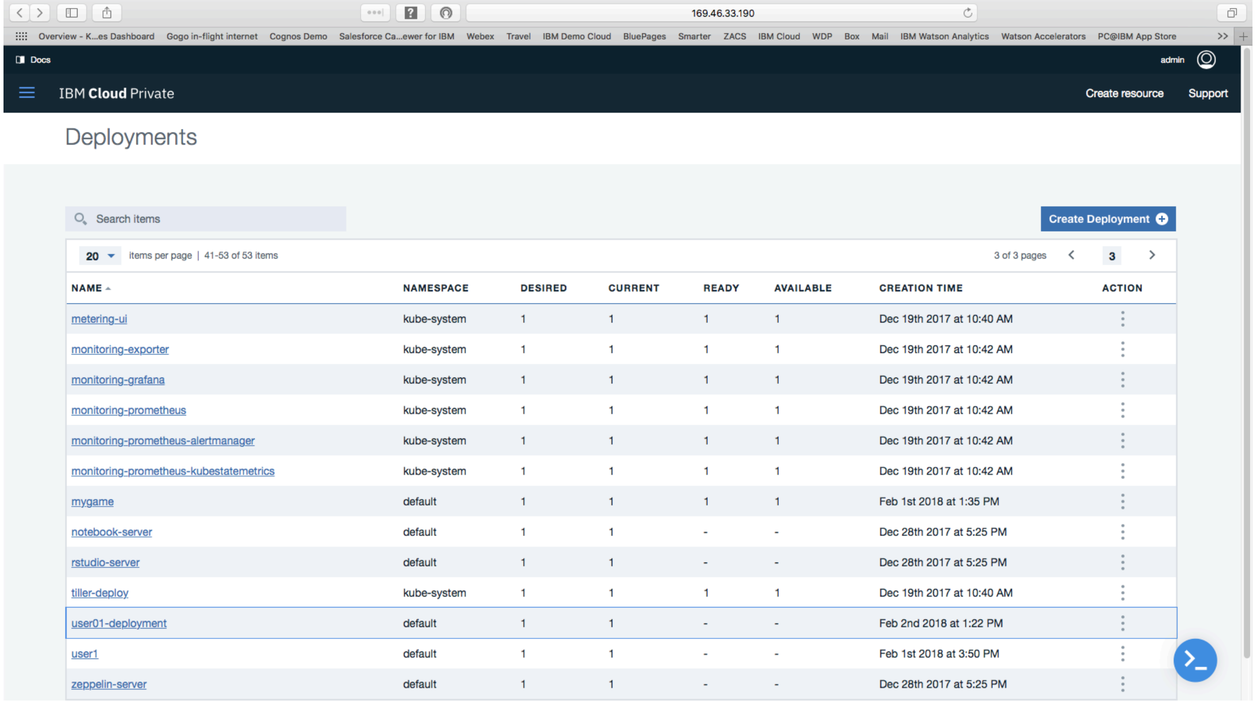
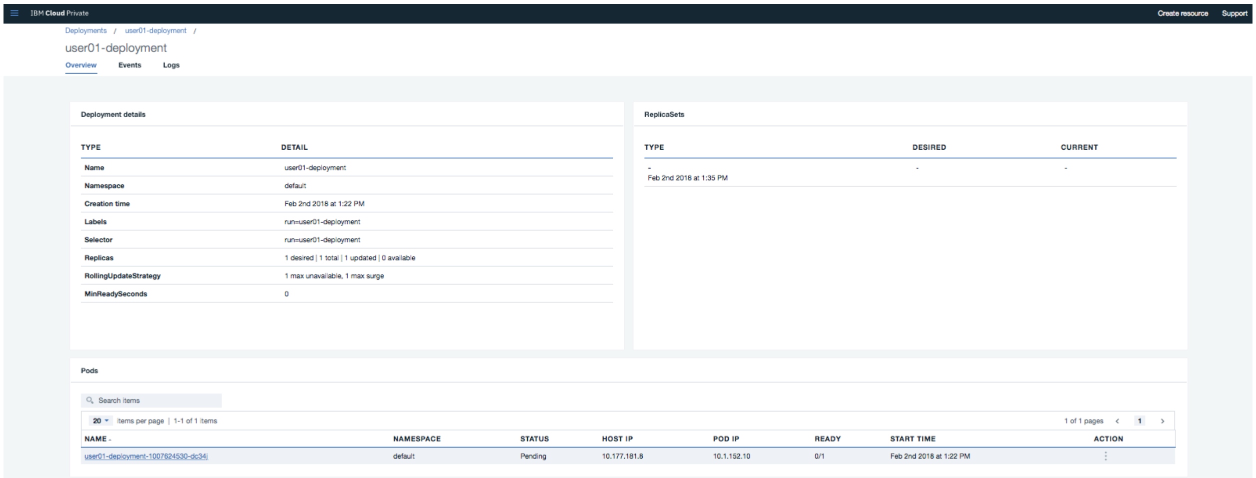
```
[user01@dlsol0129163851 2048_master]$ kubectl run user01-deployment --image=user01_image --port=80
deployment "user01-deployment" created
[user01@dlsol0129163851 2048_master]$ 
```

c. Return to the ICP dashboard. Select the menu icon on the upper left of the screen. Go to “Workloads” and select “Deployments”.



d. Find your deployment in the list of Deployments and select it (you may need to navigate to page 3). This will bring up details about your deployment; including the associated Pod and ReplicaSet.

© Copyright IBM Corp. 2018. All rights reserved

Step	Action
	 

Deployments

Search items

Create Deployment

20 items per page | 41-53 of 53 items — 3 of 3 pages — 3

NAME	NAMESPACE	DESIRED	CURRENT	READY	AVAILABLE	CREATION TIME	ACTION
metering-ui	kube-system	1	1	1	1	Dec 19th 2017 at 10:40 AM	⋮
monitoring-exporter	kube-system	1	1	1	1	Dec 19th 2017 at 10:42 AM	⋮
monitoring-grafana	kube-system	1	1	1	1	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus	kube-system	1	1	1	1	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus-alertmanager	kube-system	1	1	1	1	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus-kubestatemetrics	kube-system	1	1	1	1	Dec 19th 2017 at 10:42 AM	⋮
mygame	default	1	1	1	1	Feb 1st 2018 at 1:35 PM	⋮
notebook-server	default	1	1	-	-	Dec 28th 2017 at 5:25 PM	⋮
rstudio-server	default	1	1	-	-	Dec 28th 2017 at 5:25 PM	⋮
tiller-deploy	kube-system	1	1	1	1	Dec 19th 2017 at 10:40 AM	⋮
user01-deployment	default	1	1	-	-	Feb 2nd 2018 at 1:22 PM	⋮
user1	default	1	1	-	-	Feb 1st 2018 at 3:50 PM	⋮
zeppelin-server	default	1	1	-	-	Dec 28th 2017 at 5:25 PM	⋮

user01-deployment

Deployment details

TYPE	DETAIL
Name	user01-deployment
Namespace	default
Creation time	Feb 2nd 2018 at 1:22 PM
Labels	run=user01-deployment
Selector	run=user01-deployment
Replicas	1 desired | 1 total | 1 updated | 0 available
RollingUpdateStrategy	1 max unavailable, 1 max surge
MinReadySeconds	0

ReplicaSets

TYPE	DESIRED	CURRENT
- Feb 2nd 2018 at 1:35 PM	-	-

Pods

Search items

20 items per page | 1-1 of 1 items — 1 of 1 pages — 1

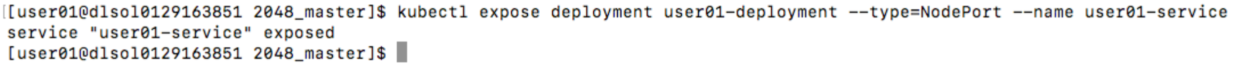
NAME	NAMESPACE	STATUS	HOST IP	POD IP	READY	START TIME	ACTION
user01-deployment-1007624530-dc34j	default	Pending	10.177.181.8	10.1.152.10	0/1	Feb 2nd 2018 at 1:22 PM	⋮

Step	Action
2	**<u>Exposing the application through a service</u>** a. In order to interact with your application from outside the cluster, you wll need to create a service which provide an endpoint to expose the application. To do this, enter the following command to create a new service. ~$ kubectl expose deployment <your username>-deployment --type=NodePort --name *<your username>-service*

© Copyright IBM Corp. 2018. All rights reserved

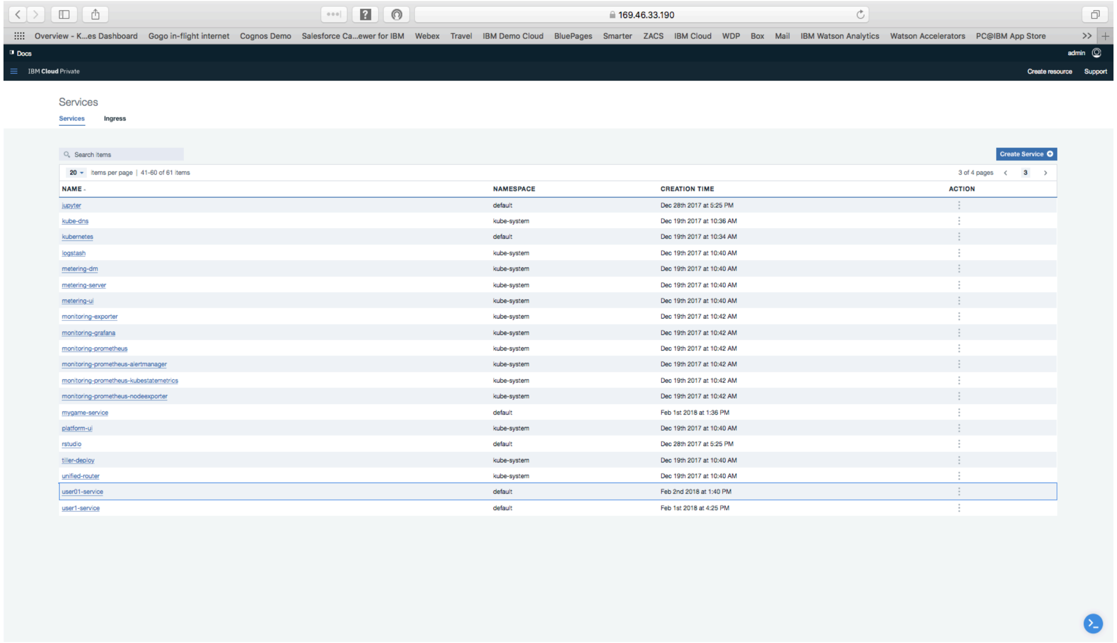
Step	Action

b. Confirm the output is as shown below:



```
[user01@dlsol0129163851 2048_master]$ kubectl expose deployment user01-deployment --type=NodePort --name user01-service
service "user01-service" exposed
[user01@dlsol0129163851 2048_master]$ █
```

c. Return to the ICP dashboard. Under the “Workloads” menu option, select “Services”. The list of services will appear. Confirm your service (you may have to navigate to the 3rd or 4th page) is listed.

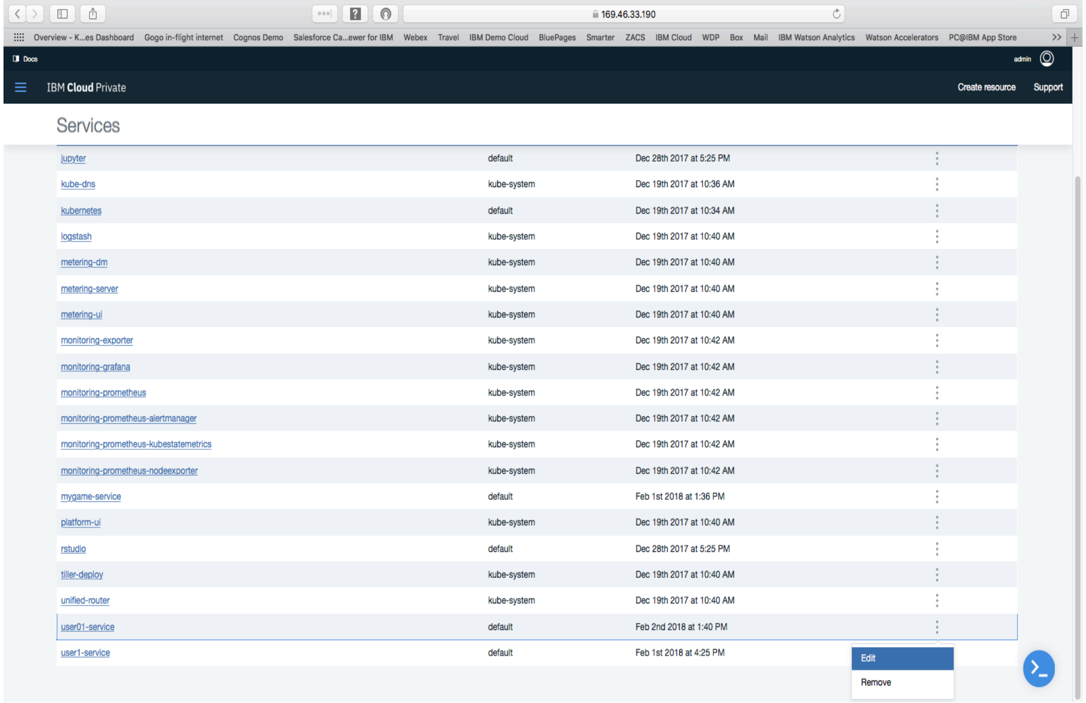


20 items per page | 41-60 of 61 items 3 of 4 pages 3

NAME	NAMESPACE	CREATION TIME	ACTION
jupyter	default	Dec 28th 2017 at 5:25 PM	⋮
kube-dns	kube-system	Dec 19th 2017 at 10:36 AM	⋮
kubernetes	default	Dec 19th 2017 at 10:34 AM	⋮
logstash	kube-system	Dec 19th 2017 at 10:40 AM	⋮
metering-dm	kube-system	Dec 19th 2017 at 10:40 AM	⋮
metering-server	kube-system	Dec 19th 2017 at 10:40 AM	⋮
metering-ui	kube-system	Dec 19th 2017 at 10:40 AM	⋮
monitoring-exporter	kube-system	Dec 19th 2017 at 10:42 AM	⋮
monitoring-grafana	kube-system	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus	kube-system	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus-alertmanager	kube-system	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus-kubestatemetrics	kube-system	Dec 19th 2017 at 10:42 AM	⋮
monitoring-prometheus-nodeexporter	kube-system	Dec 19th 2017 at 10:42 AM	⋮
mygame-service	default	Feb 1st 2018 at 1:36 PM	⋮
platform-ui	kube-system	Dec 19th 2017 at 10:40 AM	⋮
rstudio	default	Dec 28th 2017 at 5:25 PM	⋮
tiller-deploy	kube-system	Dec 19th 2017 at 10:40 AM	⋮
unified-router	kube-system	Dec 19th 2017 at 10:40 AM	⋮
user01-service	default	Feb 2nd 2018 at 1:40 PM	⋮
user1-service	default	Feb 1st 2018 at 4:25 PM	⋮

d. When you expose a service, Kubernetes automatically assigns a unique IP address that the cluster will listen to on behalf of your application. This address is typically in the 30000-32000 range. However, due to some of the open port limitations in our Data Center (nothing to do with ICP itself), we need to manually replace this with a new port. To do this, select the “Action” menu on the right side of your service’s listing in the screen shown above and select “Edit, as shown below.

© Copyright IBM Corp. 2018. All rights reserved

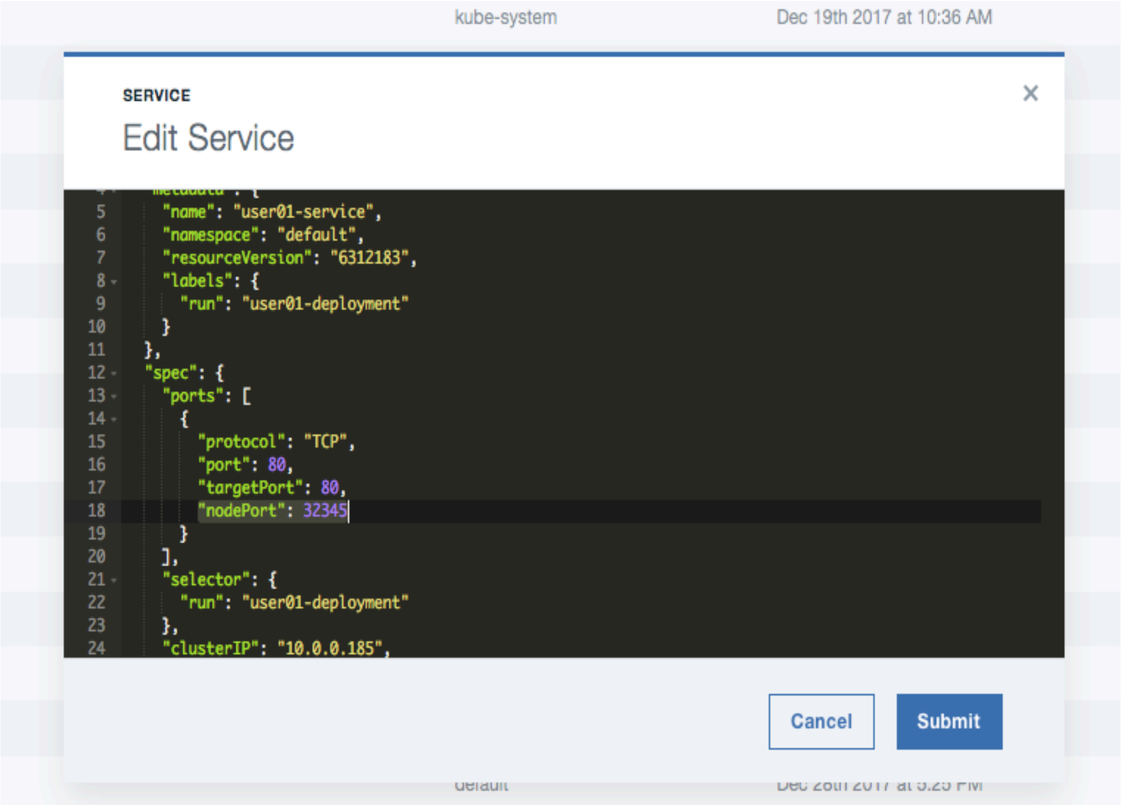
<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td></td><td>


Services

<table>
<tr><td>jupyter</td><td>default</td><td>Dec 28th 2017 at 5:25 PM</td></tr>
<tr><td>kube-dns</td><td>kube-system</td><td>Dec 19th 2017 at 10:36 AM</td></tr>
<tr><td>kubernetes</td><td>default</td><td>Dec 19th 2017 at 10:34 AM</td></tr>
<tr><td>logstash</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>metering-dm</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>metering-server</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>metering-ui</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>monitoring-exporter</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>monitoring-grafana</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>monitoring-prometheus</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>monitoring-prometheus-alertmanager</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>monitoring-prometheus-kubestatemetrics</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>monitoring-prometheus-nodeexporter</td><td>kube-system</td><td>Dec 19th 2017 at 10:42 AM</td></tr>
<tr><td>mygame-service</td><td>default</td><td>Feb 1st 2018 at 1:36 PM</td></tr>
<tr><td>platform-ui</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>rstudio</td><td>default</td><td>Dec 28th 2017 at 5:25 PM</td></tr>
<tr><td>tiller-deploy</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>unified-router</td><td>kube-system</td><td>Dec 19th 2017 at 10:40 AM</td></tr>
<tr><td>user01-service</td><td>default</td><td>Feb 2nd 2018 at 1:40 PM</td></tr>
<tr><td>user1-service</td><td>default</td><td>Feb 1st 2018 at 4:25 PM</td></tr>
</table>

e. An editing window will appear that will allow you to edit the YAML code that defines the service. Locate the “NodePort” field and replace the port number with the <your port> used previously, as shown below.
</td></tr>
</table>

Step	Action



```
  "metadata": {
    "name": "user01-service",
    "namespace": "default",
    "resourceVersion": "6312183",
    "labels": {
      "run": "user01-deployment"
    }
  },
  "spec": {
    "ports": [
      {
        "protocol": "TCP",
        "port": 80,
        "targetPort": 80,
        "nodePort": 32345
      }
    ],
    "selector": {
      "run": "user01-deployment"
    },
    "clusterIP": "10.0.0.185",
```

f. Click Submit.

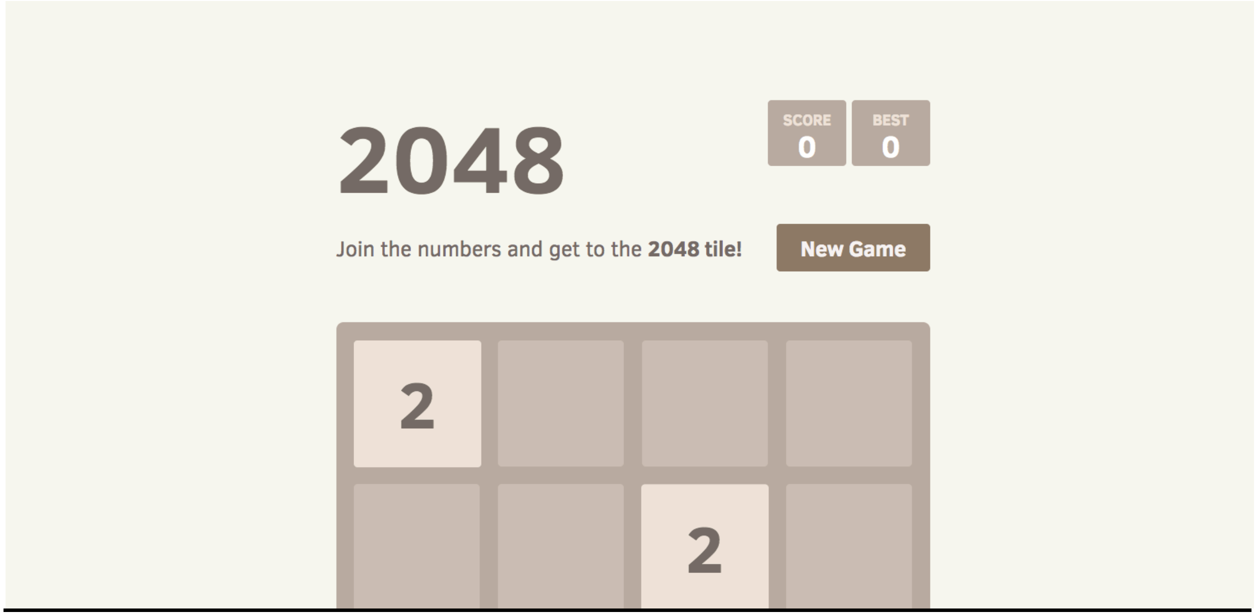
You have not successfully enabled your application running in the Kubernetes cluster to be accessed from the outside.

3 **Access the Running Application**

a. To access the application, go to your browser and enter the following URL and verify that you can access the application, as shown below:

169.46.33.190:<*your port* >

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td></td><td>



Join the numbers and get to the **2048 tile!**

b. Delete the deployment and the service, using the following commands:

```
~$ kubectl delete deployment <your username>-deployment
```

```
~$ kubectl delete service <your username>-service
```

</td></tr>
</table>

Section 4: Lab Summary

In this section, you learned how to deploy an Docker application to Kubernetes, how to enable it to be access from the outside world, and how to access it.

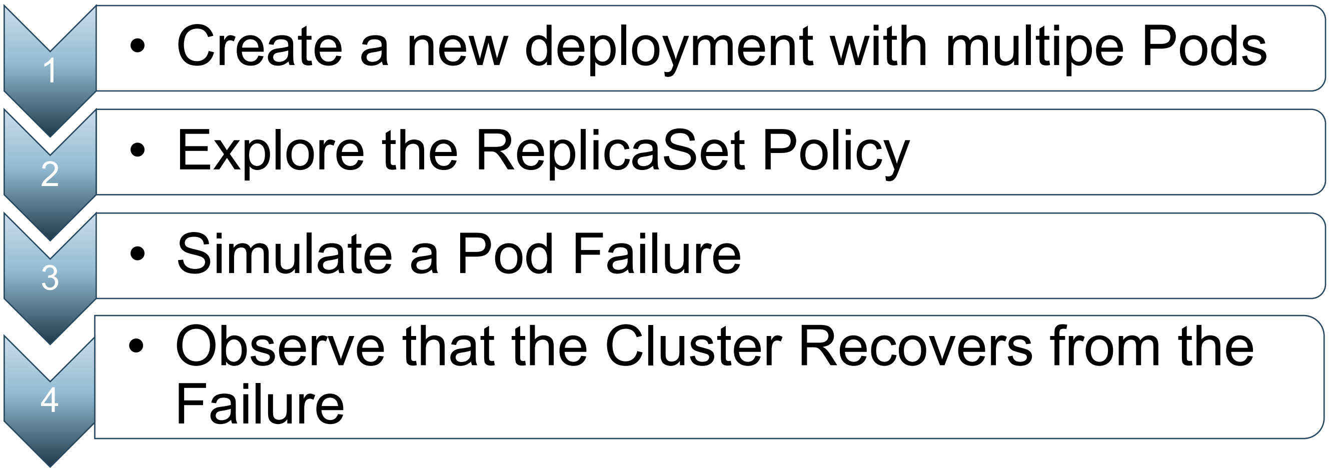
Section 5: Observing Kubernetes Resiliency

Purpose:	In this lab, you will learn how Kubernetes recovers from a container failure.

Tasks:	Tasks you will complete in this lab exercise include: • Create a new deployment with multiple Pods • Explore the ReplicaSet policy • Simulate a pod failure • Observer how the cluster quickly recovers from the failure to retain the number of available pods

© Copyright IBM Corp. 2018. All rights reserved

Section 5: Observing Kubernetes Resiliency



1. Create a new deployment with multipe Pods
2. Explore the ReplicaSet Policy
3. Simulate a Pod Failure
4. Observe that the Cluster Recovers from the Failure

© Copyright IBM Corp. 2018. All rights reserved

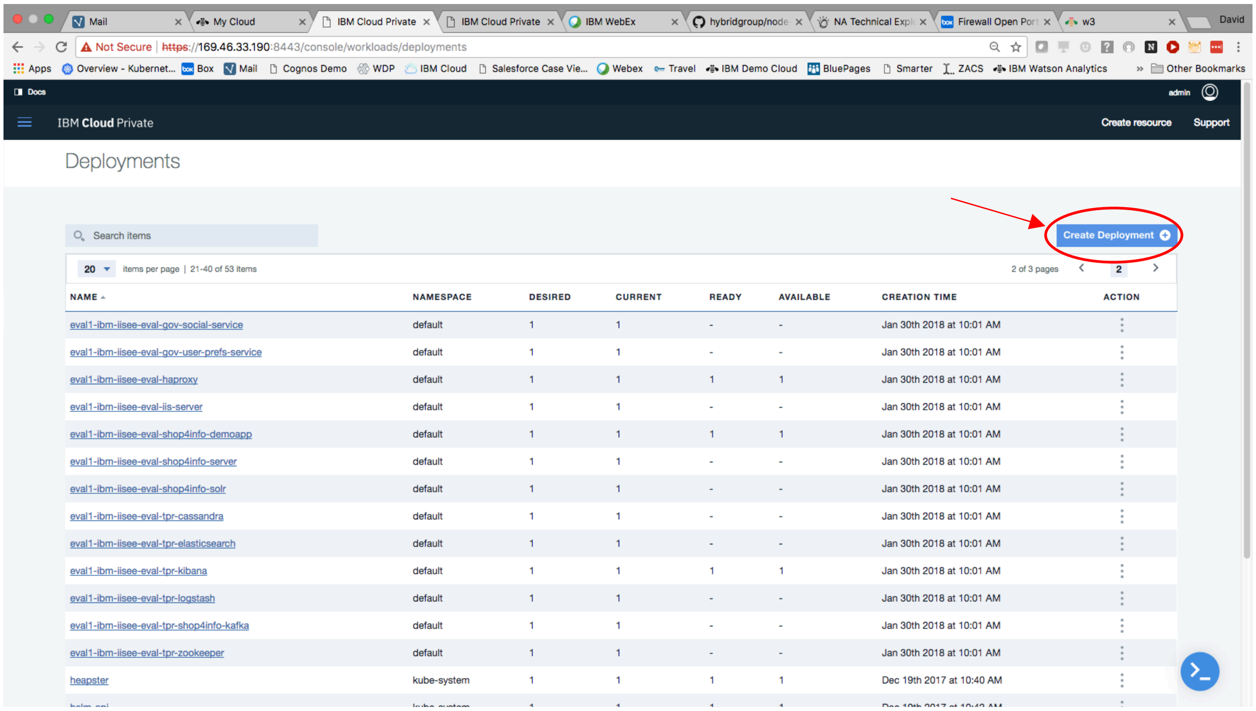
Section 5: Lab Instructions

Step	Action
1	**<u>Create a new deployment</u>**

a. We will now create a new deployment using the ICP web GUI interface. This time, however, we will specify the use of 2 Pods.

Access the ICP Dashboard and select “Workloads” and then “Deployment” in the menu on the left.

b. Select the “Create Deployment” button on the upper right of the page:



Deployments

Search items

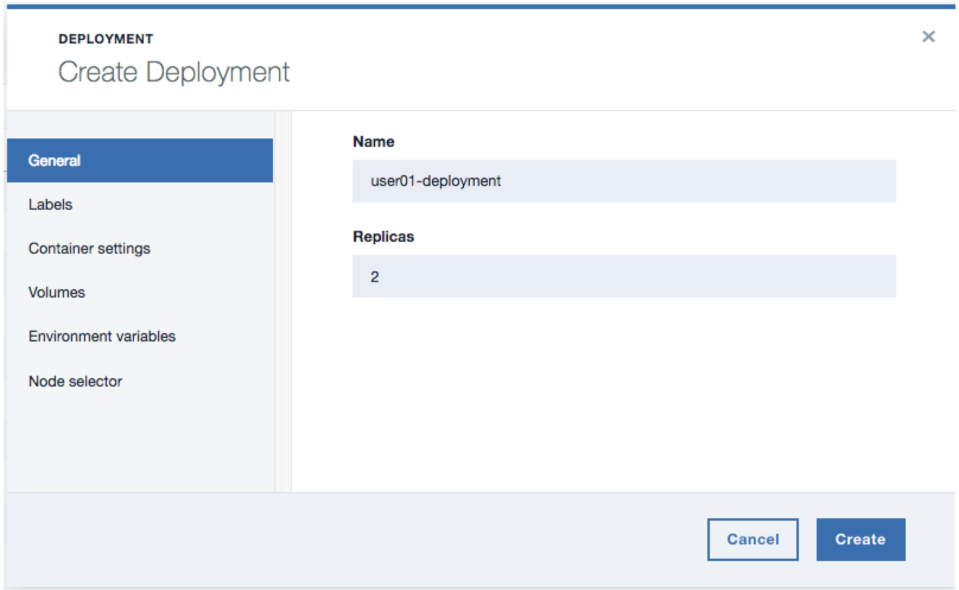
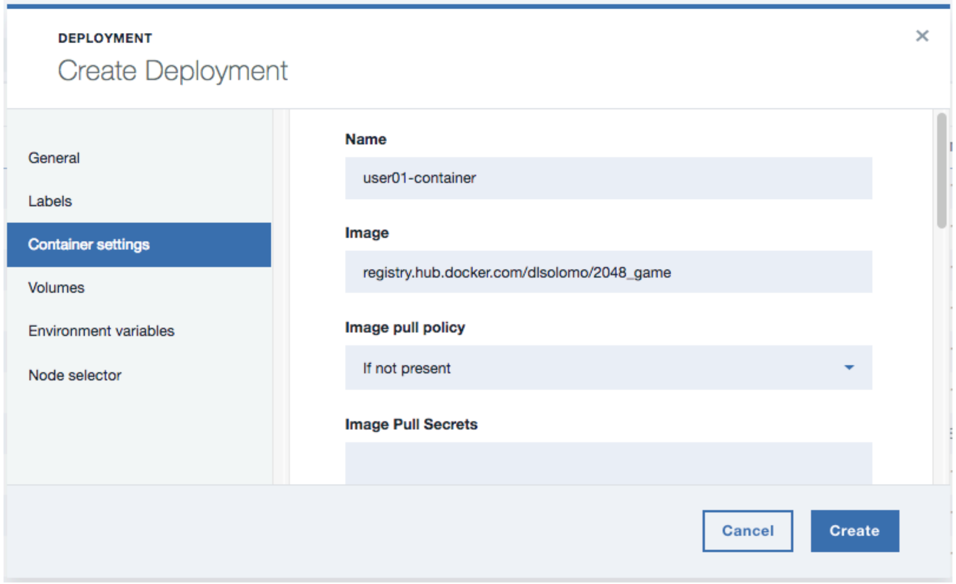
20 items per page | 21-40 of 53 items | 2 of 3 pages | 2

NAME	NAMESPACE	DESIRED	CURRENT	READY	AVAILABLE	CREATION TIME	ACTION
eval1-ibm-iisee-eval-gov-social-service	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-gov-user-prefs-service	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-haproxy	default	1	1	1	1	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-iis-server	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-shop4info-demoapp	default	1	1	1	1	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-shop4info-server	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-shop4info-solr	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-cassandra	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-elasticsearch	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-kibana	default	1	1	1	1	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-logstash	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-shop4info-kafka	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
eval1-ibm-iisee-eval-tpr-zookeeper	default	1	1	-	-	Jan 30th 2018 at 10:01 AM	⋮
heapster	kube-system	1	1	1	1	Dec 19th 2017 at 10:40 AM	⋮

c. A create deployment form will appear. Complete the form using the following settings; as shown below and click “Create”. This will create a deployment with 2 Pods:

In the “General” tab:
Name= <your username>-deployment
Replicas= 2

© Copyright IBM Corp. 2018. All rights reserved

Step	Action
	In the “Container settings” tab: Name=<your username>-container Image= registry.hub.docker.com/dlsolomo/2048_game  DEPLOYMENT Create Deployment Name user01-deployment Replicas 2 General Labels Container settings Volumes Environment variables Node selector Cancel Create Click “Create”.  DEPLOYMENT Create Deployment Name user01-container Image registry.hub.docker.com/dlsolomo/2048_game Image pull policy If not present Image Pull Secrets General Labels Container settings Volumes Environment variables Node selector Cancel Create

© Copyright IBM Corp. 2018. All rights reserved

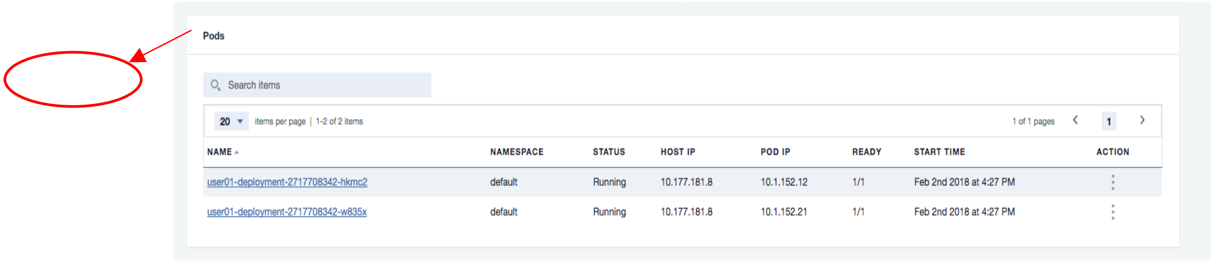
Step	Action

2 **Explore the ReplicaSet Policy**

a. We will now examine the ReplicaSet in more detail. As you may recall, a ReplicaSet manages a policy that governs the how and when Pods are deployed, including the recovery of a failed Pod. This recovery is based a policy established during or after a deployment.

Return to the ICP Dashboard. Go to the deployment list under "Workloads" and then "Deployments" and select the deployment you just created.

Note that there are now 2 PODs for this deployment.



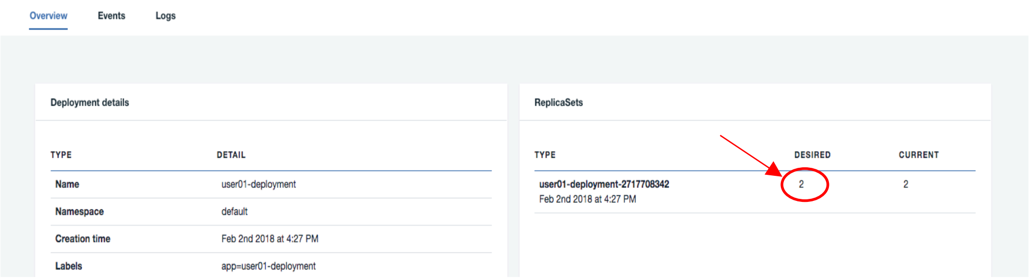
Pods

Search items

20 items per page | 1-2 of 2 items   1 of 1 pages 1

NAME	NAMESPACE	STATUS	HOST IP	POD IP	READY	START TIME	ACTION
user01-deployment-2717708342-hkmc2	default	Running	10.177.181.8	10.1.152.12	1/1	Feb 2nd 2018 at 4:27 PM	⋮
user01-deployment-2717708342-w835x	default	Running	10.177.181.8	10.1.152.21	1/1	Feb 2nd 2018 at 4:27 PM	⋮

Also note under the "ReplicaSets" section that the desired number of pods is set to 2. This means that the RepliSet will always attempt to maintain 2 pods up and running to service this application.



Deployment details

TYPE	DETAIL
Name	user01-deployment
Namespace	default
Creation time	Feb 2nd 2018 at 4:27 PM
Labels	app=user01-deployment

ReplicaSets

TYPE	DESIRED	CURRENT
user01-deployment-2717708342 Feb 2nd 2018 at 4:27 PM	2	2

3 **Simulate a Pod Failure**

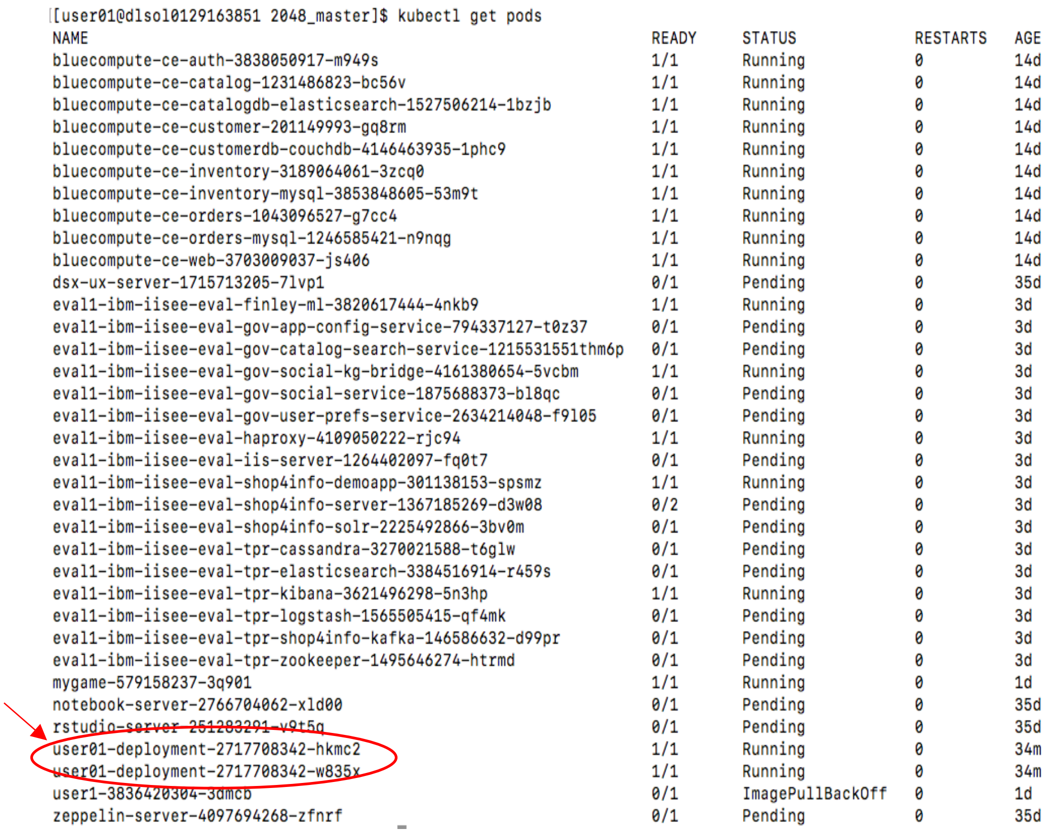
a. We will now use a kubectl command to simulate the failure of a pod. To do this, find the Pod IDs for the running Pods using the following command:

```
~$ kubectl get pods
```

© Copyright IBM Corp. 2018. All rights reserved

Step	Action

The command will list all the running pods and their names. Identify the 2 pods associated with your application, as shown below:



```
[user01@dlsol0129163851 2048_master]$ kubectl get pods
NAME                                                    READY     STATUS             RESTARTS   AGE
bluecompute-ce-auth-3838050917-m949s                    1/1       Running            0          14d
bluecompute-ce-catalog-1231486823-bc56v                 1/1       Running            0          14d
bluecompute-ce-catalogdb-elasticsearch-1527506214-1bzjb 1/1       Running            0          14d
bluecompute-ce-customer-201149993-gq8rm                 1/1       Running            0          14d
bluecompute-ce-customerdb-couchdb-4146463935-1phc9      1/1       Running            0          14d
bluecompute-ce-inventory-3189064061-3zcq0               1/1       Running            0          14d
bluecompute-ce-inventory-mysql-3853848605-53m9t         1/1       Running            0          14d
bluecompute-ce-orders-1043096527-g7cc4                  1/1       Running            0          14d
bluecompute-ce-orders-mysql-1246585421-n9nqg            1/1       Running            0          14d
bluecompute-ce-web-3703009037-js406                     1/1       Running            0          14d
dsx-ux-server-1715713205-7lvp1                          0/1       Pending            0          35d
eval1-ibm-iisee-eval-finley-ml-3820617444-4nkb9         1/1       Running            0          3d
eval1-ibm-iisee-eval-gov-app-config-service-794337127-t0z37  0/1  Pending            0          3d
eval1-ibm-iisee-eval-gov-catalog-search-service-1215531551thm6p  0/1  Pending        0          3d
eval1-ibm-iisee-eval-gov-social-kg-bridge-4161380654-5vcbm   1/1  Running            0          3d
eval1-ibm-iisee-eval-gov-social-service-1875688373-bl8qc     0/1  Pending            0          3d
eval1-ibm-iisee-eval-gov-user-prefs-service-2634214048-f9105 0/1  Pending            0          3d
eval1-ibm-iisee-eval-haproxy-4109050222-rjc94           1/1       Running            0          3d
eval1-ibm-iisee-eval-iis-server-1264402097-fq0t7        0/1       Pending            0          3d
eval1-ibm-iisee-eval-shop4info-demoapp-301138153-spsmz  1/1       Running            0          3d
eval1-ibm-iisee-eval-shop4info-server-1367185269-d3w08  0/2       Pending            0          3d
eval1-ibm-iisee-eval-shop4info-solr-2225492866-3bv0m    0/1       Pending            0          3d
eval1-ibm-iisee-eval-tpr-cassandra-3270021588-t6glw     0/1       Pending            0          3d
eval1-ibm-iisee-eval-tpr-elasticsearch-3384516914-r459s 0/1       Pending            0          3d
eval1-ibm-iisee-eval-tpr-kibana-3621496298-5n3hp        1/1       Running            0          3d
eval1-ibm-iisee-eval-tpr-logstash-1565505415-qf4mk      0/1       Pending            0          3d
eval1-ibm-iisee-eval-tpr-shop4info-kafka-146586632-d99pr 0/1      Pending            0          3d
eval1-ibm-iisee-eval-tpr-zookeeper-1495646274-htrmd     0/1       Pending            0          3d
mygame-579158237-3q901                                  1/1       Running            0          1d
notebook-server-2766704062-xld00                        0/1       Pending            0          35d
rstudio-server-251283291-v9t5q                          0/1       Pending            0          35d
user01-deployment-2717708342-hkmc2                      1/1       Running            0          34m
user01-deployment-2717708342-w835x                      1/1       Running            0          34m
user1-3836420304-3dmcb                                  0/1       ImagePullBackOff   0          1d
zeppelin-server-4097694268-zfnrf                        0/1       Pending            0          35d
```

b. Enter the following command to delete one of the Pods (it does not matter which one). Copy the name from the output of the previous step.

~$ kubectl delete pods <*the name of one of your Pods*>.

4

Observe that the Cluster Recovers from the Failure

a. Wait approximately 30 seconds and run the following command again and notice that one of the pods now has a different name. This is because when we deleted the other pod, the ReplicaSet rules immediately ensured that a new pod was created to ensure continuity, reliability, and quality of servicing the application.

~$ kubectl get pods

© Copyright IBM Corp. 2018. All rights reserved

Step	Action

	`user01-deployment-2717708342-hkmc2      1/1     Running     0     42m`
	`user01-deployment-2717708342-zn4qn      1/1     Running     0     12s`

Section 5: Lab Summary

In this section, you learned how Kubernetes can quickly recover from a Pod failure.