

IBM Cloud

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

Hands-on Workshop

Lab Guide





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

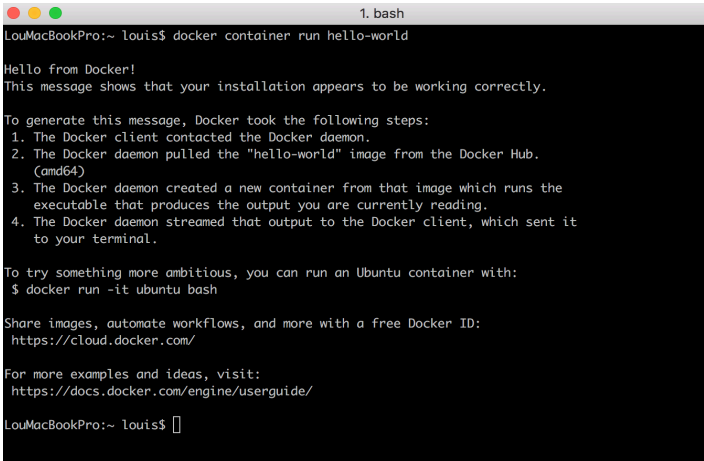
Table of Contents

Section 1: Container Basics	4
Section 1: Lab Instructions	5
Section 1: Lab Summary	10
Section 2: Data Persistence in Docker	11
Section 2: Lab Instructions	12
Section 2: Lab Summary	13
Section 3: Getting Started with Kubernetes in IBM Cloud Private	14
Section 3: Lab Instructions	15
Section 3: Lab Summary	17
Section 4: Deploy your Application to Kubernetes	18
Section 4: Lab Instructions	19
Section 4: Lab Summary	25
Section 5: Observing Kubernetes Resiliency	26
Section 5: Lab Instructions	27
Section 5: Lab Summary	28
Section 6: Deploying Services Using the ICP Catalog	29
Section 6: Lab Instructions	30
Section 6: Lab Summary	50
Section 7: Deploying a Microservices Application in ICP	51
Section 7: Lab Instructions	53

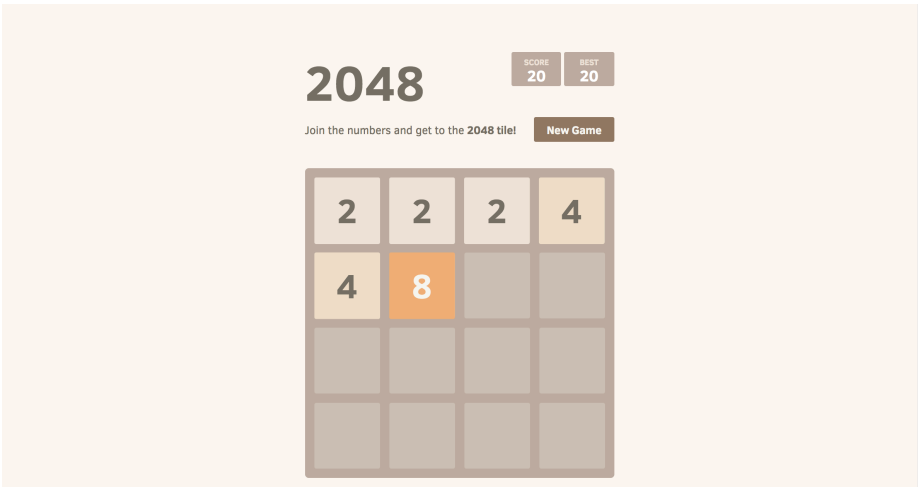
Section 1: Container Basics

Purpose:	<p>For the first 2 sections, 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.</p> <p>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.</p>
Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• 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 Instructions

Step	Action
1	<p><u>Login to the Docker Environment</u></p> <p>___ a. Your environment is on a cloud hosted Linux server. You can access this environment using the URL provided by your instructor.</p> <p>___ b. Once logged in, open a Gnome terminal window from the desktop. Next verify that docker is accessible by typing the following command:</p> <p style="text-align: center;">~\$ docker container run hello-world</p> <p>Verify that the output is similar to the following:</p> 
2	<p><u>Build a Docker Image for an Application</u></p> <p>___ a. Before we can work with a container, we will need to first build an image for our 2048 application. First, access the directory for our application code.</p> <p style="text-align: center;">~\$ cd 2048_master</p> <p>___ 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:</p>

Step	Action																									
	<div><div><div><div><div></div><div></div><div></div></div><div>Dockerfile</div><div>#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</div></div></div><div><p>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.</p><p>__c. Now you can build the image by running the following command:</p><pre>~ \$ docker build -t 2048_image . (don't forget the "." at the end)</pre><p>__d. Docker will now build the image. You can confirm this by running the following command and observing that an image named "2048_image" is listed:</p><pre>~\$ docker images</pre><div><pre>[[user01@d1sol0129163851 2048_master]\$ docker images</pre><table><tr><th>REPOSITORY</th><th>TAG</th><th>IMAGE ID</th><th>CREATED</th><th>SIZE</th></tr><tr><td>user01_image</td><td>latest</td><td>56156c8f775e</td><td>About a minute ago</td><td>155MB</td></tr><tr><td><none></td><td><none></td><td>0f16eb39c0f6</td><td>3 hours ago</td><td>155MB</td></tr><tr><td>nginx</td><td>latest</td><td>3f8a4339aadd</td><td>5 weeks ago</td><td>108MB</td></tr><tr><td>hello-world</td><td>latest</td><td>f2a91732366c</td><td>2 months ago</td><td>1.85kB</td></tr></table></div><p>You have now successfully taken an existing application and created a Docker image from it.</p></div></div>	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
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																						

Step	Action
3	<p><u>Run a Container</u></p> <p>__a. Now that you have an image, we will now run the 2048 application as a container. To do this, run the following command:</p> <p><i>Your instructor will assign you a port a unique port number to use for the remained of the lab.</i></p> <pre>~\$ docker container run --name 2048_container -p 31005:80 2048_image</pre> <p>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.</p> <p>Commands:</p> <p>--name – Specify a unique name for the container service. If omitted Docker will create a random, human readable name.</p> <p>-p – Specify that the container internal port (80) be exposed to <your port> on the host.</p> <p>__b. Open a browser and navigate to: http://localhost:31005. A page will open with the game, as shown below:</p>  <p>You have now successfully run your first container!!</p>
3	<p><u>Stop/Delete a Container</u></p> <p>__a. You can stop the container by typing cntrl-c</p> <pre>~\$ <Cntrl-c></pre>

Step	Action
	<p>__ b. Verify that the container is no longer running: ~\$ docker container ps</p> <p>__ c. Although the container is not running it still exists: ~\$ docker container ps -a</p> <pre data-bbox="232 533 1425 583">[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</pre> <p>-a, --all: Show all containers (default shows just running)</p> <p>__ d. Remove the container: ~\$ docker container rm 2048_container</p> <p>Containers can be removed either by their name or container id</p>
4	<p><u>Inspect a Running Container</u></p> <p>__ a. Run a new Docker container for the game: ~\$ docker run --publish <your port>:80 --detach --name 2048_container 2048_image</p> <p>You should be brought back to the terminal prompt (the “detach” option runs the container as a background process)</p> <p>__ b. Open a browser and navigate to “TBD”. You should be prompted with the game again.</p> <p>__ 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:</p> <p>~\$ docker container inspect 2048_container</p> <p>and,</p> <p>Stream live performance container metrics:</p> <p>~\$ docker container stats 2048_container</p> <p>__ d. Clean up ~\$ docker container rm -f 2048_container</p>

Step	Action
	<p>Commands:</p> <p>-d, --detach - Run the container in the background.</p>
5	<p><u>Run Shell Inside a Container</u></p> <p>__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.</p> <p>First run the container again:</p> <pre>~\$ docker container run --name 2048_container -d -p <your port>:80 2048_image</pre> <p>__b. Next, we will use the following command to open a shell prompt into the container:</p> <pre>~\$ docker exec -it 2048_container bash</pre> <p>__c. Run Linux commands in container: For example, # ls -tal // List directories and files. # exit // Exit shell</p> <p>__d. Delete the container:</p> <pre>~\$ docker rm -f 2048_container</pre> <p>Commands:</p> <ul style="list-style-type: none"> -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

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.

Section 2: Data Persistence in Docker

Purpose:	<p>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.</p> <p>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.</p>
Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• Create and work with Docker volumes

Section 2: Lab Instructions

Step	Action
1	<p><u>Docker Volumes</u></p> <p>__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:</p> <pre>~\$ docker container run -d --name 2048_container -p <your port>:80 -v myvol:/usr/share/nginx/html 2048_image</pre> <p>__b. Open bash shell on container and navigate the /usr/share/nginx/html directory:</p> <pre>~\$ docker container exec -it 2048_container bash # cd /usr/share/nginx/html</pre> <p>__c. Create a new file in the html folder containing the phrase, "This is my file".</p> <pre># echo "This is my file" > myfile</pre> <p>Confirm the file "myfile" is listed in the directory and exit the container.</p> <pre># ls</pre> <pre>[root@1f5d5f84c4a4:/usr/share/nginx/html# ls 50x.html Rakefile favicon.ico index.html js meta myfile style root@1f5d5f84c4a4:/usr/share/nginx/html# █</pre> <pre># exit</pre> <p>__d. We will now remove the container using the command:</p> <pre>~\$ docker rm -f 2048_container</pre> <p>__e. Now, we can create a new container, referencing the persistent volume and confirm that our file is still present:</p> <pre>~\$ docker container run -d --name 2048_container -p 8080:80 -v myvol:/usr/share/nginx/html 2048_image</pre> <pre>~\$ docker container exec -it 2048_container bash</pre>

Step	Action
	<pre># 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# █</pre> <p>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.</p>

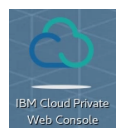
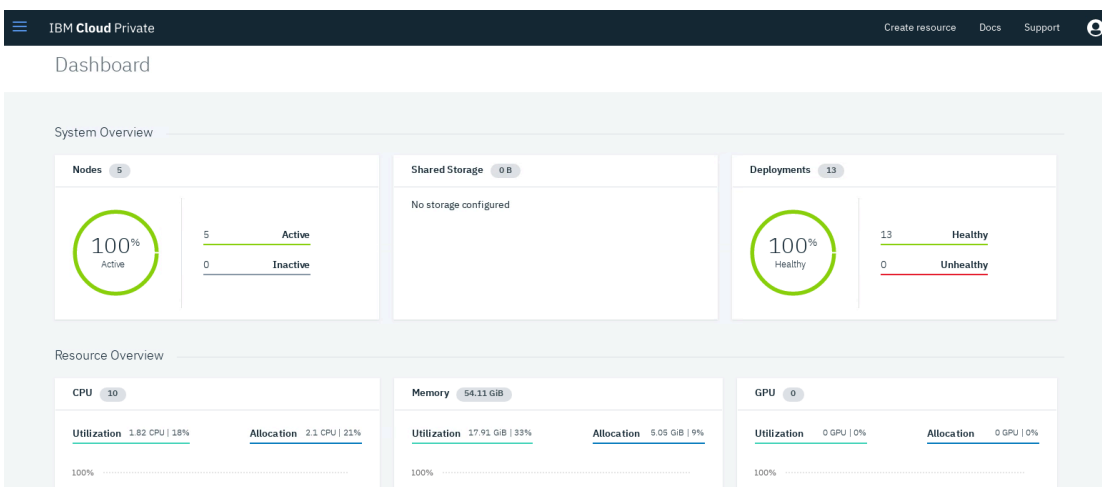
Section 2: Lab Summary


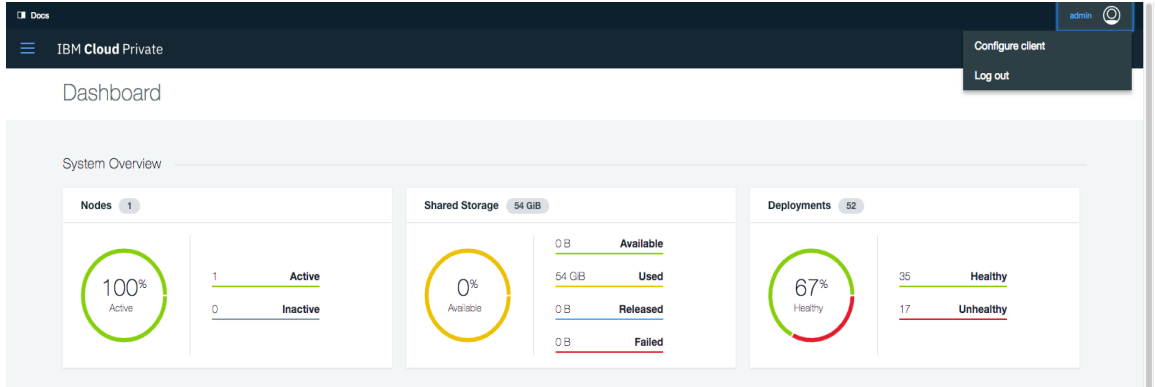
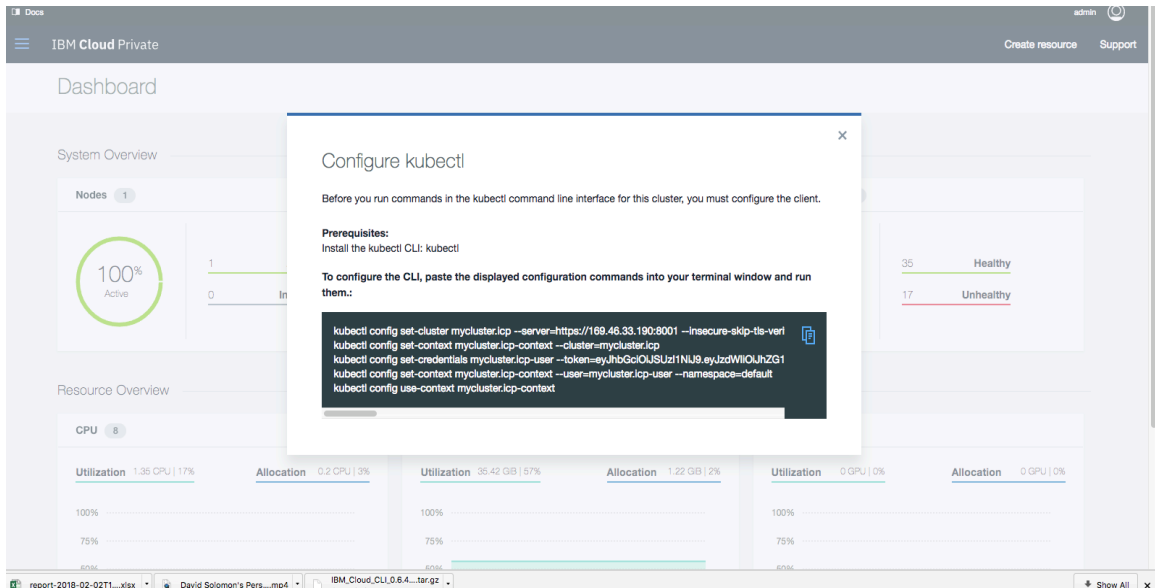
In this lab you were introduced to one way to persist data on the host file system. With volumes the container references a volume object on the local file system.

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:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• Access the IBM Cloud Private Dashboard• Access the ICP Kubernetes configuration settings• Configure your environment to use the ICP cluster

Section 3: Lab Instructions

Step	Action
1	<p><u>Launch the ICP Dashboard</u></p> <p>__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).</p> <p>Open the dashboard by double clicking on the Web Console icon on the desktop.</p>  <p>Login with username: admin/ password: admin. Click on the hamburger menu at the top and select “Dashboard”</p>  <p>You will notice that this ICP instance is a 5-node Kubernetes cluster.</p>
2	<p><u>Configure your Environment for ICP</u></p> <p>__a. In order to interact with and control the ICP cluster from a command line 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.</p>

Step	Action
	<p>On the ICP Dashboard, click on the word “admin” at the top right of the page next to the  symbol. You will then see two options, “Configure Client” and “Logout”. Select “Configure Client”.</p>  <p>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.</p>  <p>Now, copy these commands (either manually or using the blue copy symbol in the dialog box).</p>

Step	Action
__b.	Now, copy these commands (either manually or using the blue copy symbol on the upper right of the dialog box).
__c.	Open a Gnome terminal and paste these commands at a command prompt (you may need to press Return for the last command to run).
	<pre>[user01@d1sol0129163851 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@d1sol0129163851 2048_master]\$ kubectl config set-context mycluster.icp-context --cluster=mycluster.icp Context "mycluster.icp-context" created. [user01@d1sol0129163851 2048_master]\$ kubectl config set-credentials mycluster.icp-user --token=eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhZG1pb2IiImF0X2hhc2giOiJFdVZlVWV1fFNERUdm1GESEZzcU5GCUwzIiwiaXNzIjoiYWRRb2V7Y2EwN2XiaWVudWJkNDMwY2lkYy91bmRwb2ludC9PUCIsImF1ZC16IjAwMWFhbnZQ3ZDZkZDIxYzWmNmRlZmQyMTYxZDdlZW8iIiwiaWF0IjoiMTc1ODQ1Njdp9.1YnI7Xg1ZD2j7Gv5ccPj5Sb7CAVDh2e6PP4K3JwVLADSk4ZRAMpXKVEWxK0NUdecVUU4pS8c1-Sx6-zms12koXQWqIn_c8B61lKhtYvqX-c7VNRWdxcc7XNBaMVAHSHYgK-n-dLgzDFBT-H-1pb7s4gMk1z9aZdaebOH9qA7Z7P5S3aR1iKzWioyNDycugbsVLoydsU_IJA1ZeWg14mPP4w12JToh73IEM5GLKa6SUpwRyVz90B_c7pLGt8ZFTS62WeCdWuvN3lVHOsuYnA1dv6xuT9YJYajTb0U1b80o8f8jeY5oxugH0bmJ5IhwitZDvSe1Vw2oQq [user01@d1sol0129163851 2048_master]\$ kubectl config set-context mycluster.icp-context --user=mycluster.icp-user --namespace=default Context "mycluster.icp-context" modified. [user01@d1sol0129163851 2048_master]\$ kubectl config use-context mycluster.icp-context Switched to context "mycluster.icp-context". [user01@d1sol0129163851 2048_master]\$</pre>
	<p>You have now successfully configured your environment to start working with Kubernetes and IBM Cloud Private.</p>

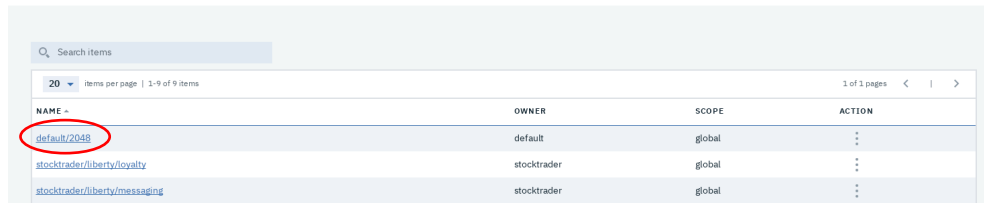
Section 3: Lab Summary

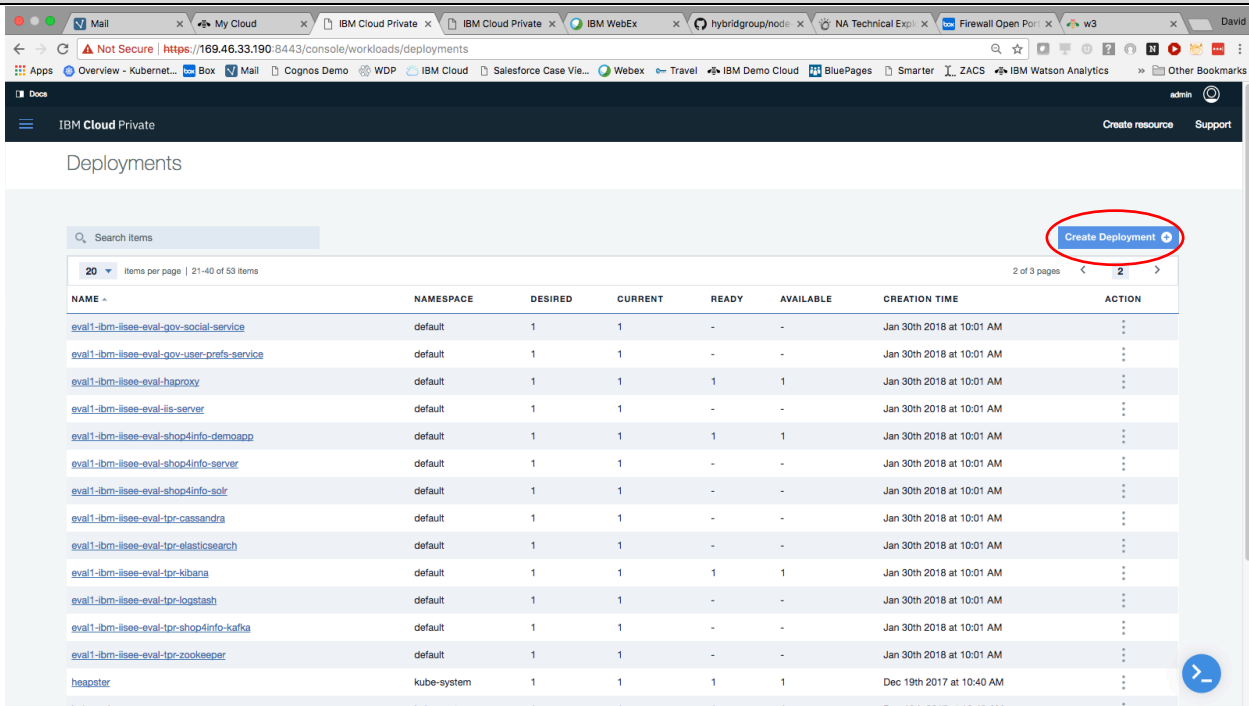
In this section, you learned how to access the ICP Dashboard and setup a your environment to interact with a Kubernetes cluster on ICP.

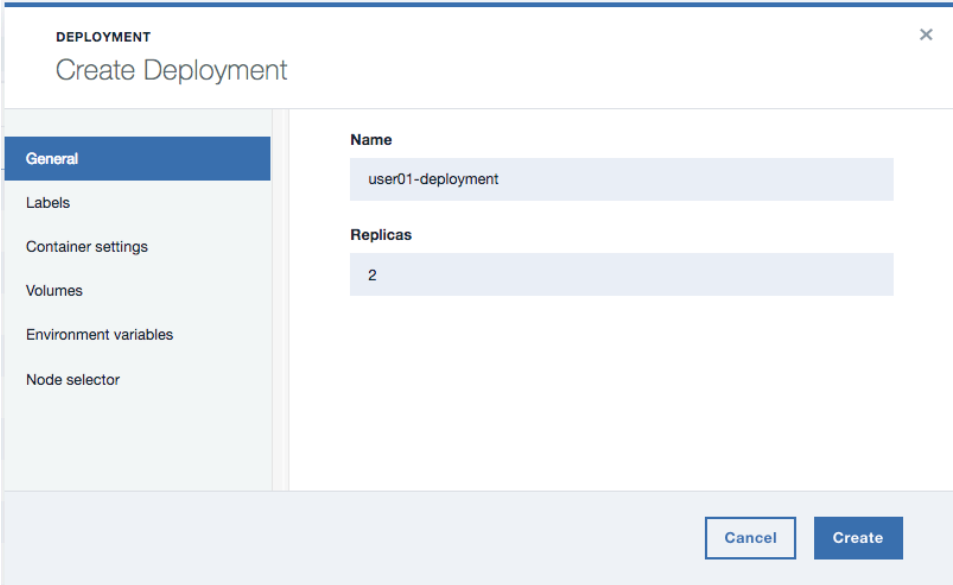
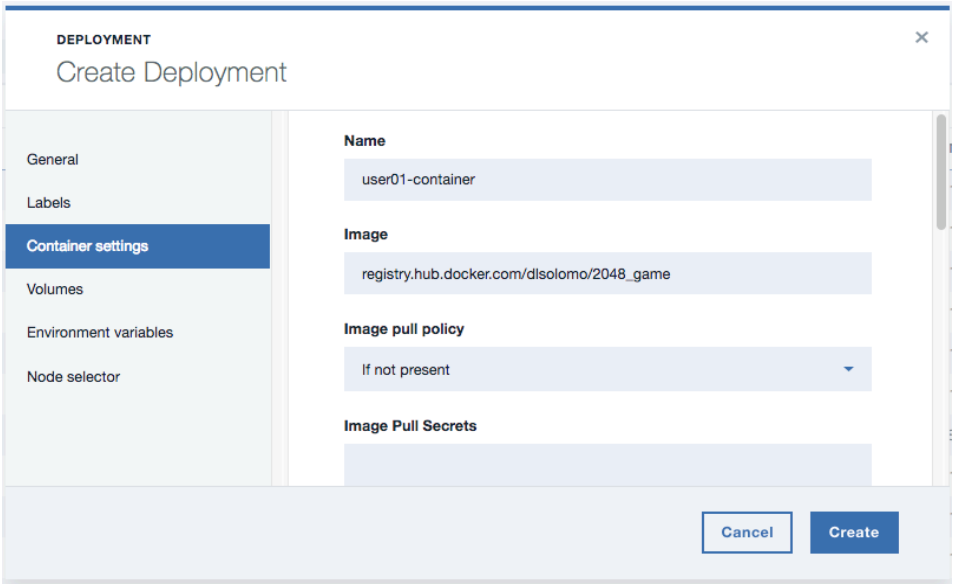
Section 4: Deploy your Application to Kubernetes

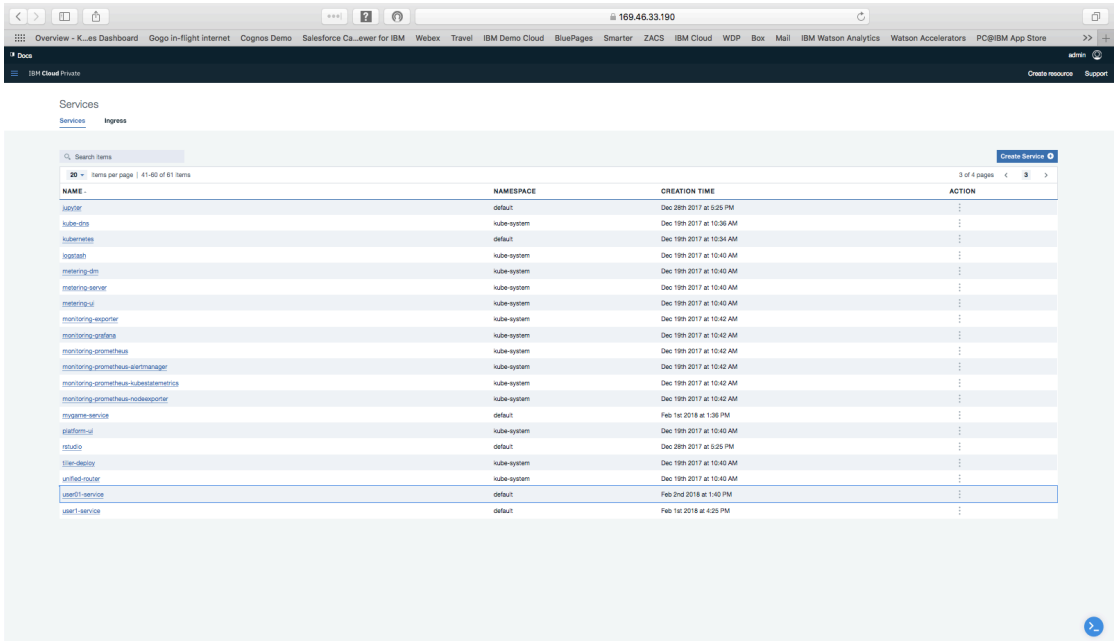
Purpose:	In this lab you will learn how to deploy an application to Kubernetes.
Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• Deploy a Docker application to Kubernetes• Expose the application through a service• Access the running application

Section 4: Lab Instructions

Step	Action
1	<p><u>Copy the 2048 Image to the ICP Private Docker Registry</u></p> <p>__a. In a real-world scenario, the enterprise applications you will deploy on ICP should only be accessible through a private registry, such as the one included with ICP. From there, we can then deploy our applications. In order to copy our 2048 image to this private registry, run the following commands from the same terminal window you used in the previous lab:</p> <pre>~\$ docker login poticpcluster.icp:8500 -u admin -p admin</pre> <pre>~\$ docker tag 2048_image poticpcluster.icp:8500/default/2048:1.0</pre> <pre>~\$ docker push poticpcluster.icp:8500/default/2048:1.0</pre> <p>__b. We will now confirm that the image is in the ICP registry. Return to the ICP dashboard and from the hamburger menu, select Catalog→Images.</p> <p>__c. Confirm that the image is listed.</p>  <p>The screenshot shows the 'Images' section of the ICP dashboard. It features a search bar at the top, a dropdown menu set to '20' items per page, and a table of images. The table has columns for NAME, OWNER, SCOPE, and ACTION. The first row, 'default/2048', is highlighted with a red circle. Below it are two other rows: 'stocktrader/liberty/loyalty' and 'stocktrader/liberty/messaging'.</p>
1	<p><u>Create a new deployment</u></p> <p>__a. We will now deploy our game application to your ICP Cluster. Access the ICP Dashboard and select “Workloads” and then “Deployment” from the hamburger menu.</p> <p>__b. Select the “Create Deployment” button on the upper right of the page:</p>

Step	Action
	 <p>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:</p> <p>In the “General” tab: Name= 2048-deployment Replicas= 2</p> <p>In the “Container settings” tab: Name=2048-container Image= poticpcluster.icp:8500/default/2048:1.0</p>

Step	Action
	<div data-bbox="394 275 1341 856">  </div> <p data-bbox="418 892 623 926">Click “Create”.</p> <div data-bbox="394 1005 1341 1587">  </div>
2	<p data-bbox="228 1625 881 1659"><u>Exposing the application through a service</u></p> <p data-bbox="228 1696 1471 1791">___a. In order to interact with your application from outside the cluster, you will 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.</p>

Step	Action
	<p>~\$ kubectl expose deployment 2048-deployment --type=NodePort --name <i>service-2048</i></p> <p>__b. Confirm the output is as shown below:</p> <pre>[user01@dlsol0129163851 2048_master]\$ kubectl expose deployment user01-deployment --type=NodePort --name user01-service service "user01-service" exposed [user01@dlsol0129163851 2048_master]\$</pre> <p>__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.</p>  <p>__d. When you expose a service, Kubernetes automatically assigns a unique port that the cluster will listen to on behalf of your application. This port is typically in the 30000-32000 range.</p>

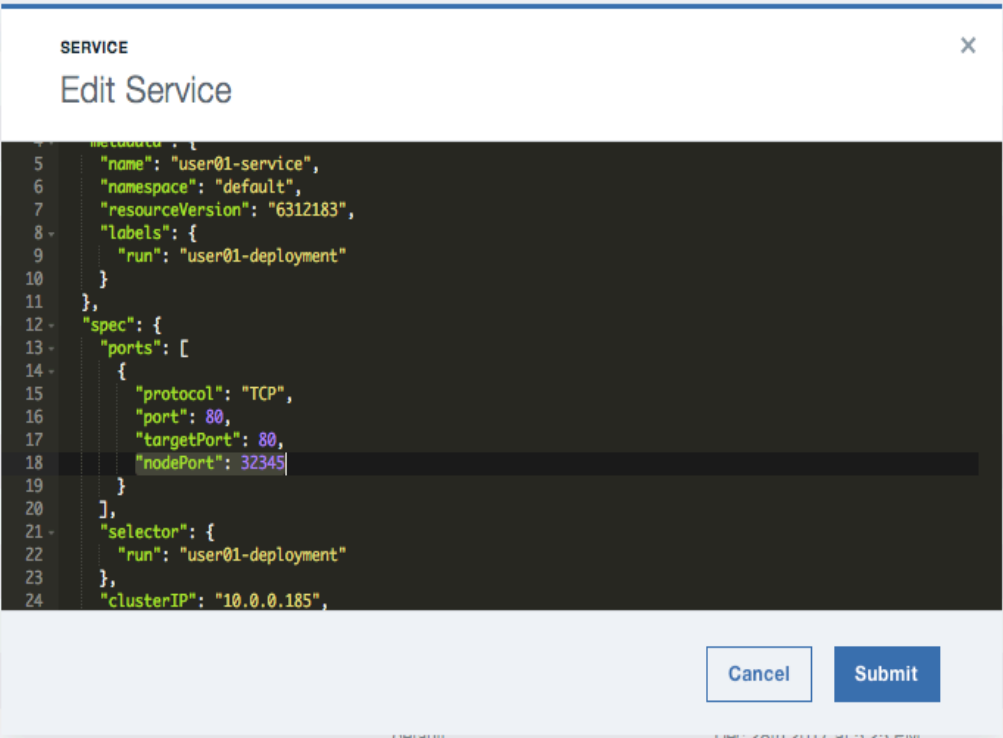
Step

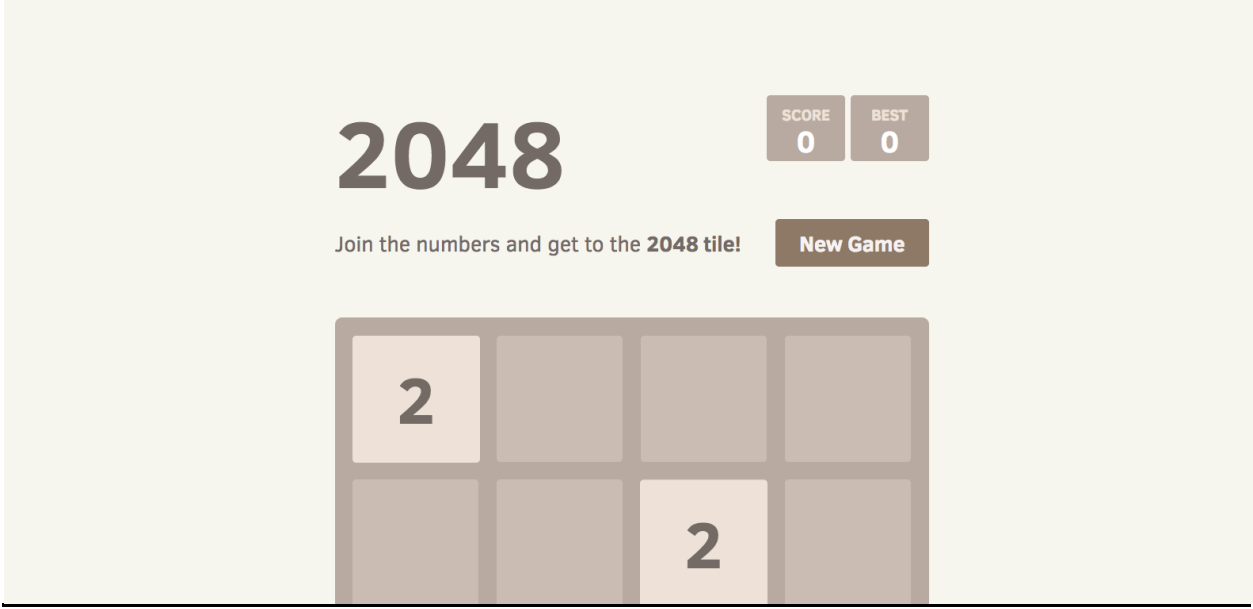
Action

The screenshot shows the IBM Cloud Private console interface. At the top, there's a navigation bar with the 'IBM Cloud Private' logo and links for 'Create resource' and 'Support'. Below this, the 'Services' tab is active, displaying a table of installed services. The table has three columns: Service Name, Namespace, and Created At. The 'user01-service' is highlighted in blue. To the right of the table, there are 'Edit' and 'Remove' buttons. A blue circular button with a right arrow is also visible at the bottom right of the table area.

Service Name	Namespace	Created At
juicytfr	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
rtstudio	default	Dec 28th 2017 at 5:25 PM
title-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

__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.

Step	Action
	<div data-bbox="386 279 1502 1075">  </div> <p>__f. Click Cancel.</p> <p>You have now successfully enabled your application running in the Kubernetes cluster to be accessed from the outside.</p>
3	<p><u>Access the Running Application</u></p> <p>__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:</p> <p style="text-align: center;">192.168.142.102:<your port ></p>

Step	Action
	

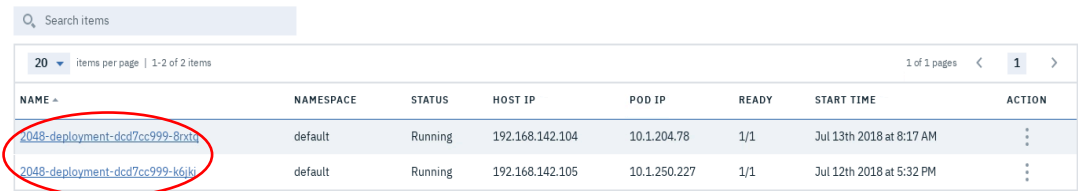

Section 4: Lab Summary

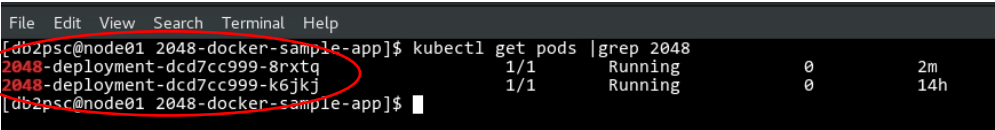
In this section, you learned how to deploy an Docker application to Kubernetes, how to enable it to be access from the outside world, and how to access it.

Section 5: Observing Kubernetes Resiliency

Purpose:	In this lab, you will learn how Kubernetes recovers from a container failure.
Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• 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

Section 5: Lab Instructions

Step	Action
1	<p><u>Explore the ReplicaSet Policy</u></p> <p>a. In the previous section, note that you deployed the application with 2 replicas. We will now examine these 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.</p> <p>Return to the ICP Dashboard. Go to the deployment list under “Workloads” and then “Deployments” and select the deployment you previously created.</p> <p>Note that there are now 2 PODs for this deployment.</p> <p>Pods</p>  <p>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.</p> <p>ReplicaSets</p> 
2	<p><u>Simulate a Pod Failure</u></p> <p>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:</p> <pre>~\$ kubectl get pods grep 2048</pre> <p>The command will list all the running pods and their names. Identify the 2 pods associated with your application, as shown below:</p>

Step	Action
	<div data-bbox="451 306 1440 434">  <pre> File Edit View Search Terminal Help [db2psc@node01 2048-docker-sample-app]\$ kubectl get pods grep 2048 2048-deployment-dcd7cc999-8rxtq 1/1 Running 0 2m 2048-deployment-dcd7cc999-k6jkj 1/1 Running 0 14h [db2psc@node01 2048-docker-sample-app]\$ </pre> </div> <p data-bbox="245 470 1422 537">__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.</p> <p data-bbox="402 573 1192 609">~\$ kubectl delete pods <i><the name of one of your Pods></i>.</p>
3	<p data-bbox="245 686 1016 720"><u>Observe that the Cluster Recovers from the Failure</u></p> <p data-bbox="245 758 1484 892">__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.</p> <p data-bbox="402 928 812 963">~\$ kubectl get pods grep 2048</p>


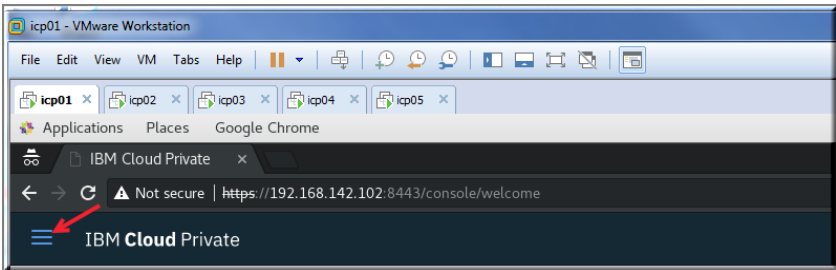
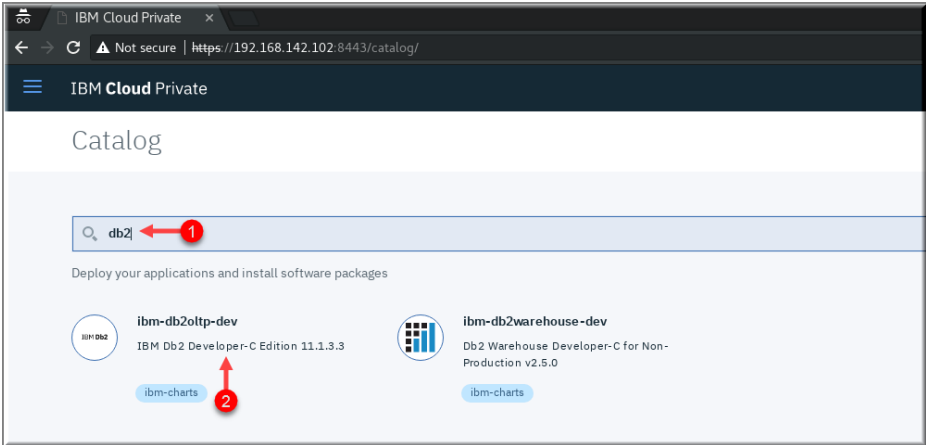
Section 5: Lab Summary


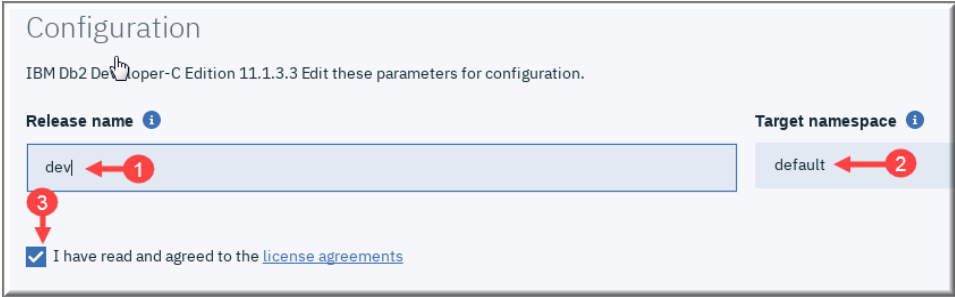
In this section, you learned how Kubernetes can quickly recover from a Pod failure.



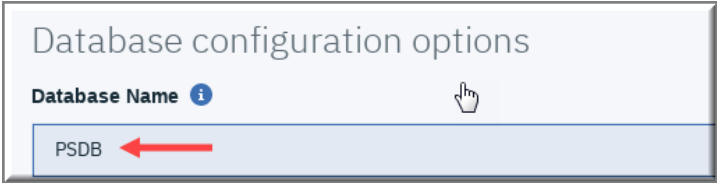

Section 6: Deploying Services Using the ICP Catalog

Purpose:	In this lab, you will learn how you can quickly deploy IBM and 3 rd -party services using the ICP catalog. The services you will be deploying will be used in a later lab to support our deploying of a full microservices application.
Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none">• Deploy db2• Deploy IBM MQ• Deploy Redis for in-memory cache

Section 6: Lab Instructions

Step	Action
1	<p><u>Deploy db2</u></p> <p>__a. Switch to the web GUI of IBM Cloud Private.</p> <p>__b. Click Hamburger  as shown. [Remember this for future reference.]</p>  <p>__c. Expand Catalog and click Helm Charts.</p> <p>__d. The Catalog page shows software offerings from IBM and as well as Open Source.</p> <p>__e. Type db2 in the search box to narrow down the search. Click ibm-db2oltp-dev</p>  <p>__f. Review the IBM Db2 Developer-C Helm Chart page as shown. Read the prerequisites – 1. Docker container and 2. Persistent Storage for Db2</p> <p>__g. Since Docker containers are ephemeral, we need persistent storage.</p>

Step	Action
	<p>The Db2 Helm Chart supports two options for persistent storage.</p> <ul style="list-style-type: none"> __1. Using predefined Persistent Volume and Persistent Volume Claim. __2. Dynamic provisioning <p>In IBM Cloud Private, the persistent volume support is for HostPath, NFS, vSphere cloud provider and Gluster. IBM Spectrum Scale (GPFS) will be added soon.</p> <p>In our lab exercise, we use Gluster dynamic provisioning.</p> <p>Note that Gluster can have raw volumes from Tier-1 SAN storage. This allows you to deploy production-grade, monolithic databases using the StatefulSet capability of the IBM Cloud Private to get all new features and functions of SVC (Service Volume Claim), PVC (Persistent Volume Claim), dynamic provisioning and loose coupling between applications and databases.</p> <p>The applications can still be microservices based, connecting to tier-1 scalable, production-grade database such as Db2 deployed in the IBM Cloud Private environment.</p> <p>Configure Db2</p> <p>__h. Scroll towards the bottom of the page and click Configure.</p>  <p>__i. Enter the Release name dev.</p> <p>__j. Select the Target namespace default.</p> <p>__k. Check the box I have read and agreed to the License agreements.</p>  <p>__l. Scroll to the Worker node architecture and select amd64 from the drop-down menu.</p>




Step	Action
	<div data-bbox="297 289 977 491">  <p>Worker node architecture</p> <p>Worker node architecture ⓘ</p> <p>amd64 ←</p> </div> <p data-bbox="191 520 1539 594">__m. Scroll to the <i>Db2 instance configuration</i>, type the <i>Db2 instance name</i> db2psc and the <i>Password for Db2 instance</i> password.</p> <div data-bbox="297 625 1300 848">  <p>Db2 instance configuration</p> <p>Db2 instance name ⓘ Password for Db2 instance ⓘ</p> <p>db2psc ← password</p> </div> <p data-bbox="191 877 1068 919">__n. In <i>Database configuration options</i>, type <i>Database Name</i> PSDB.</p> <div data-bbox="297 947 1010 1129">  <p>Database configuration options</p> <p>Database Name ⓘ</p> <p>PSDB ←</p> </div> <p data-bbox="191 1159 1531 1232">__o. Scroll to <i>Data volume configuration</i>, and type glusterfs-storage in <i>Existing storage class name</i>. [Note: You must not skip this step.]</p> <div data-bbox="297 1262 1399 1583">  <p>Data volume configuration</p> <p>Name of the persistent volume claim ⓘ Existing volume claim ⓘ</p> <p>data-stor Enter value</p> <p>Existing storage class name ⓘ Size of the volume claim ⓘ</p> <p>glusterfs-storage ← glusterfs-storage 20Gi</p> </div> <p data-bbox="191 1612 1580 1686">__p. Scroll to the <i>Resource configuration</i> and change the <i>Memory request</i> to 1Gi, <i>Memory limit</i> to 3Gi, <i>CPU request</i> to 1000m and <i>CPU limit</i> to 2000m.</p>

Step	Action										
	<div><div><div><div>Resource configuration</div><div><div><div>Memory request ⓘ</div><div>1Gi ← 1</div></div><div><div>CPU request ⓘ</div><div>1000m ← 3</div></div><div><div>Memory limit ⓘ</div><div>3Gi ← 2</div></div><div><div>CPU limit ⓘ</div><div>2000m ← 4</div></div></div></div></div></div> <p>__q. Click Install.</p> <div><div><div>Cancel</div><div>Install ←</div></div></div> <p>__r. Click View Helm Release.</p> <div><div><div>✓</div><div>Installation complete</div><div>↓</div><div>View Helm Release</div><div>Return to Catalog</div></div></div> <p>__s. Click the just-deployed Helm release dev.</p> <div><div><div>Helm releases</div><div><div>🔍 Search items</div><div>20 items per page 1-1 of 1 items</div><table><tr><th>NAME^</th><th>NAMESPACE^</th><th>STATUS^</th><th>CHART NAME^</th><th>CURRENT VERSION^</th></tr><tr><td>dev ↓</td><td>default</td><td>Deployed</td><td>ibm-db2oltp-dev</td><td>2.0.0</td></tr></table></div></div></div> <p>__t. Click Hamburger ☰ expand Workloads and click StatefulSets.</p> <p>__u. Click dev-ibm-db2oltp-dev.</p>	NAME^	NAMESPACE^	STATUS^	CHART NAME^	CURRENT VERSION^	dev ↓	default	Deployed	ibm-db2oltp-dev	2.0.0
NAME^	NAMESPACE^	STATUS^	CHART NAME^	CURRENT VERSION^							
dev ↓	default	Deployed	ibm-db2oltp-dev	2.0.0							

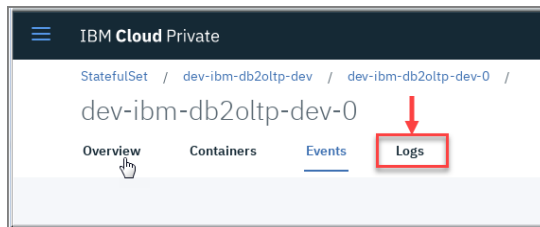
Step	Action																																																			
	<div><div>20 items per page 1-4 of 4 items</div><table><tr><th>NAME</th><th>NAMESPACE</th><th>DESIRED</th><th>CURRENT</th><th>CREATION TIME ^</th></tr><tr><td>dev-ibm-db2oltp-dev</td><td>default</td><td>1</td><td>1</td><td>Mar 30th 2018 at 4:00 PM</td></tr><tr><td>image-manager</td><td>kube-system</td><td>1</td><td>1</td><td>Mar 30th 2018 at 1:11 AM</td></tr><tr><td>elasticsearch-data</td><td>kube-system</td><td>1</td><td>1</td><td>Mar 30th 2018 at 1:10 AM</td></tr><tr><td>icp-ds</td><td>kube-system</td><td>1</td><td>1</td><td>Mar 30th 2018 at 12:56 AM</td></tr></table></div> <p>__v. In Pods section, notice that the Db2 Docker image was deployed. It runs on one of the worker nodes.</p> <div><div>Pods</div><div>Q Search items</div><div>20 items per page 1-1 of 1 items1 of 1 pages < 1 ></div><table><tr><th>NAME ^</th><th>NAMESPACE</th><th>STATUS</th><th>HOST IP</th><th>POD IP</th><th>READY</th><th>START TIME</th><th>ACTION</th></tr><tr><td>dev-ibm-db2oltp-dev-0</td><td>default</td><td>Running</td><td>192.168.142.103</td><td>10.1.177.70</td><td>1/1</td><td>Mar 28th 2018 at 11:27 AM</td><td></td></tr></table></div> <p>__w. Click dev-ibm-db2oltp-dev-0</p> <p>__x. The Overview screen displays.</p> <p>__y. Click Containers. Click dev-ibm-db2oltp-dev.</p> <div><div>dev-ibm-db2oltp-dev-0</div><div>OverviewContainersEventsLogs</div><div><div>1</div><table><tr><th>NAME</th><th>IMAGE</th><th>PORT</th><th>STATE</th><th>RESTART COUNT</th></tr><tr><td>dev-ibm-db2oltp-dev</td><td>store/ibmcorp/db2_developer_c:11.1.3.3-x86_64</td><td>50000/TCP,55000/TCP</td><td>Running</td><td>0</td></tr></table></div></div>	NAME	NAMESPACE	DESIRED	CURRENT	CREATION TIME ^	dev-ibm-db2oltp-dev	default	1	1	Mar 30th 2018 at 4:00 PM	image-manager	kube-system	1	1	Mar 30th 2018 at 1:11 AM	elasticsearch-data	kube-system	1	1	Mar 30th 2018 at 1:10 AM	icp-ds	kube-system	1	1	Mar 30th 2018 at 12:56 AM	NAME ^	NAMESPACE	STATUS	HOST IP	POD IP	READY	START TIME	ACTION	dev-ibm-db2oltp-dev-0	default	Running	192.168.142.103	10.1.177.70	1/1	Mar 28th 2018 at 11:27 AM		NAME	IMAGE	PORT	STATE	RESTART COUNT	dev-ibm-db2oltp-dev	store/ibmcorp/db2_developer_c:11.1.3.3-x86_64	50000/TCP,55000/TCP	Running	0
NAME	NAMESPACE	DESIRED	CURRENT	CREATION TIME ^																																																
dev-ibm-db2oltp-dev	default	1	1	Mar 30th 2018 at 4:00 PM																																																
image-manager	kube-system	1	1	Mar 30th 2018 at 1:11 AM																																																
elasticsearch-data	kube-system	1	1	Mar 30th 2018 at 1:10 AM																																																
icp-ds	kube-system	1	1	Mar 30th 2018 at 12:56 AM																																																
NAME ^	NAMESPACE	STATUS	HOST IP	POD IP	READY	START TIME	ACTION																																													
dev-ibm-db2oltp-dev-0	default	Running	192.168.142.103	10.1.177.70	1/1	Mar 28th 2018 at 11:27 AM																																														
NAME	IMAGE	PORT	STATE	RESTART COUNT																																																
dev-ibm-db2oltp-dev	store/ibmcorp/db2_developer_c:11.1.3.3-x86_64	50000/TCP,55000/TCP	Running	0																																																

__z. Click [Events](#).

Search items						
20 items per page 1-9 of 9 items		1 of 1 pages < 1 >				
TYPE	SOURCE	COUNT	REASON	MESSAGE	FIRST SEEN ^	LAST SEEN
Warning	default-scheduler	4	FailedScheduling	pod has unbound PersistentVolumeClaims (repeated 5 times)	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	default-scheduler	1	Scheduled	Successfully assigned dev-ibm-db2oltp-dev-0 to 192.168.142.103	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	SuccessfulMountVolume	MountVolume.SetUp succeeded for volume "dev-hadr-stor"	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	SuccessfulMountVolume	MountVolume.SetUp succeeded for volume "default-token-8cgf9"	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	SuccessfulMountVolume	MountVolume.SetUp succeeded for volume "pvc-7bbf7c1a-329c-11e8-aa9e-005056271837"	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	Pulled	Container image "store/ibmcorp/db2_developer_c:11.1.3.3-x86_64" already present on machine	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	Created	Created container	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Normal	kubelet 192.168.142.103	1	Started	Started container	Mar 28th 2018 at 11:27 AM	Mar 28th 2018 at 11:27 AM
Warning	kubelet 192.168.142.103	6	Unhealthy	Readiness probe failed: DB2 State : Operable	Mar 28th 2018 at 11:28 AM	Mar 28th 2018 at 11:30 AM

__aa. Note the events. **If there is any error**, delete the Helm chart through [Hamburger](#)  [Workload](#)  [Helm Releases](#)  [dev](#) and click the 3 vertical dots sign to delete and start all over. The error might be due to missing parameters before installing the chart.

__bb. If there is warning, ignore it. Let's examine the logs. Click [Logs](#).



__cc. These logs are directly coming from the Db2 Docker container running in Kubernetes pod. We will see, in later sections, how to open a Db2 command line shell inside the container and check the logs manually.

__dd. Scroll to the message **DB2START processing was successful**. The database creation happens automatically. Note this may take several minutes.

Step	Action
------	--------

```

become effective.
were not changed dynamically. For these configuration parameters, the database
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
must be shutdown and reactivated before the configuration parameter changes
### Restarting DB2
SQL1064N DB2STOP processing was successful.
03/28/2018 15:35:40 0 0 SQL1064N DB2STOP processing was successful.
03/28/2018 15:35:43 0 0 SQL1063N DB2START processing was successful.
SQL1063N DB2START processing was successful.
### Making backup directory and performing backup
Backup successful. The timestamp for this backup image is : 20180328153547
(*) Applying autoconfiguration for instance ...
Database server = DB2/LINUX8664 11.1.3.3
SQL authorization ID = DB2PSC
Local database alias = PSDB
Database Connection Information
SQL1363W One or more of the parameters submitted for immediate modification
were not changed dynamically. For these configuration parameters, the database
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
become effective.
must be shutdown and reactivated before the configuration parameter changes
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
become effective.
DB20000I The UPDATE DATABASE CONFIGURATION command completed successfully.
must be shutdown and reactivated before the configuration parameter changes
were not changed dynamically. For these configuration parameters, the database
SQL1363W One or more of the parameters submitted for immediate modification
DB20000I The SQL command completed successfully.
SQL1064N DB2STOP processing was successful.
03/28/2018 15:36:23 0 0 SQL1064N DB2STOP processing was successful.
SQL1063N DB2START processing was successful.
03/28/2018 15:36:27 0 0 SQL1063N DB2START processing was successful.

```

Explore Persistent Volumes

ee. Click **Hamburger**  **Platform**  **Storage**.

Storage

PersistentVolume

PersistentVolumeClaim

Search items

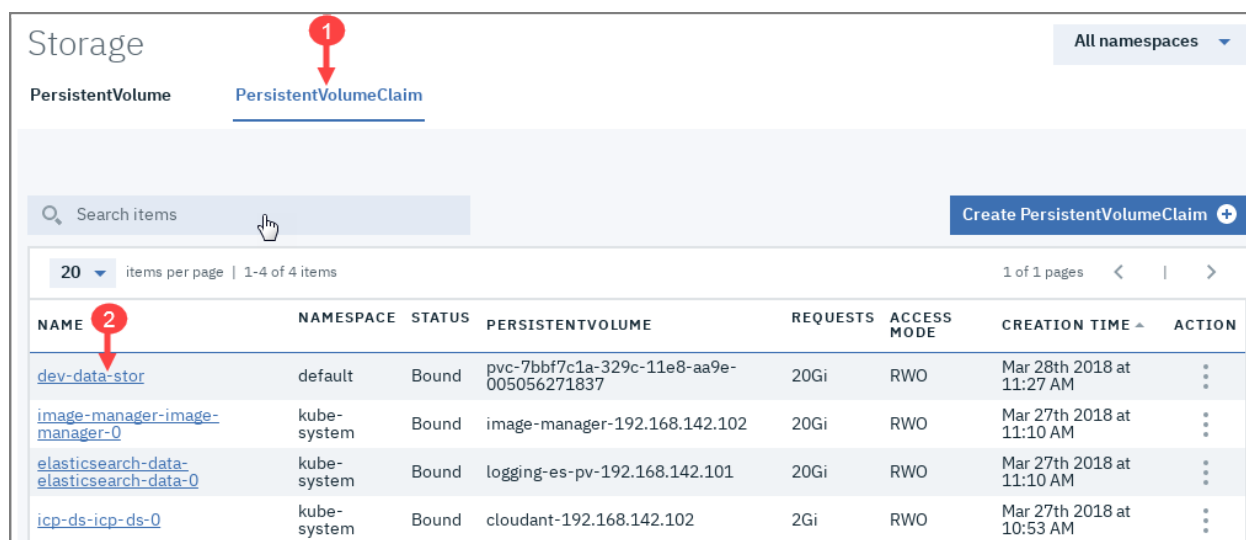
Create PersistentVolume

20 items per page | 1-4 of 4 items

1 of 1 pages

NAME	TYPE	CAPACITY	ACCESS MODE	RECLAIM POLICY	STATUS	CLAIM	CREATION TIME	ACTION
pvc-7bbf7c1a-329c-11e8-aa9e-005056271837	Glusterfs	22G	RWO	Delete	Bound	default/dev-data-stor	Mar 28th 2018 at 11:27 AM	<div></div>
image-manager-192.168.142.102	LocalVolume	20Gi	RWO	Retain	Bound	kube-system/image-manager-image-manager-0	Mar 27th 2018 at 11:10 AM	<div></div>
logging-es-pv-192.168.142.101	LocalVolume	20Gi	RWO	Delete	Bound	kube-system/elasticsearch-data-elasticsearch-data-0	Mar 27th 2018 at 11:10 AM	<div></div>
cloudant-192.168.142.102	LocalVolume	2Gi	RWO	Delete	Bound	kube-system/icp-ds-icp-ds-0	Mar 27th 2018 at 10:53 AM	<div></div>

__ff. Click [Persistent Volume Claim](#) ⇒ [dev-data-stor](#)



Storage

PersistentVolume PersistentVolumeClaim

Search items

Create PersistentVolumeClaim +

20 items per page | 1-4 of 4 items

NAME	NAMESPACE	STATUS	PERSISTENTVOLUME	REQUESTS	ACCESS MODE	CREATION TIME	ACTION
dev-data-stor	default	Bound	pvc-7bbf7c1a-329c-11e8-aa9e-005056271837	20Gi	RWO	Mar 28th 2018 at 11:27 AM	⋮
image-manager-image-manager-0	kube-system	Bound	image-manager-192.168.142.102	20Gi	RWO	Mar 27th 2018 at 11:10 AM	⋮
elasticsearch-data-elasticsearch-data-0	kube-system	Bound	logging-es-pv-192.168.142.101	20Gi	RWO	Mar 27th 2018 at 11:10 AM	⋮
icp-ds-icp-ds-0	kube-system	Bound	cloudant-192.168.142.102	2Gi	RWO	Mar 27th 2018 at 10:53 AM	⋮

__gg. Note that Type [GlusterFS](#) persistent volume claim ([dev-data-stor](#)) was created automatically by Kubernetes as we had defined the used storage class, which was created during the IBM Cloud Private install.



Note: The mechanics of dynamic provisioning of volumes is requested through REST API from Kubernetes Master node and the request is handled by [Heketi](#) REST API server that interfaces with the Gluster docker containers running in all worker nodes.

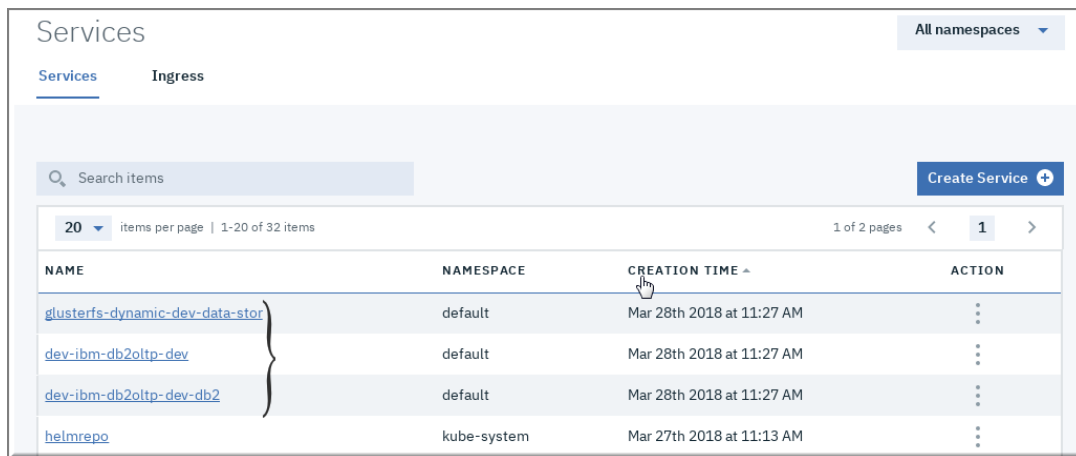
__hh. Click [Events](#) and check the message.

__ii. Note: Individual information can be obtained through [kubectl](#) commands. You can find the appropriate command from the GUI screen. We explore these in later sections of this lab.

Explore Network Service


__jj. Click [Hamburger](#) ⇒ [Network Access](#) ⇒ [Services](#)

Step	Action
------	--------



NAME	NAMESPACE	CREATION TIME	ACTION
glusterfs-dynamic-dev-data-stor	default	Mar 28th 2018 at 11:27 AM	...
dev-ibm-db2oltp-dev	default	Mar 28th 2018 at 11:27 AM	...
dev-ibm-db2oltp-dev-db2	default	Mar 28th 2018 at 11:27 AM	...
helmrepo	kube-system	Mar 27th 2018 at 11:13 AM	...

__kk. Notice three services created through the Helm chart for proper communication to and from pods running on workers and the external applications.



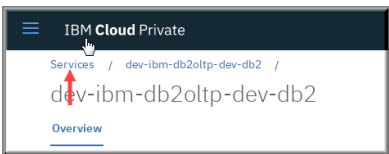




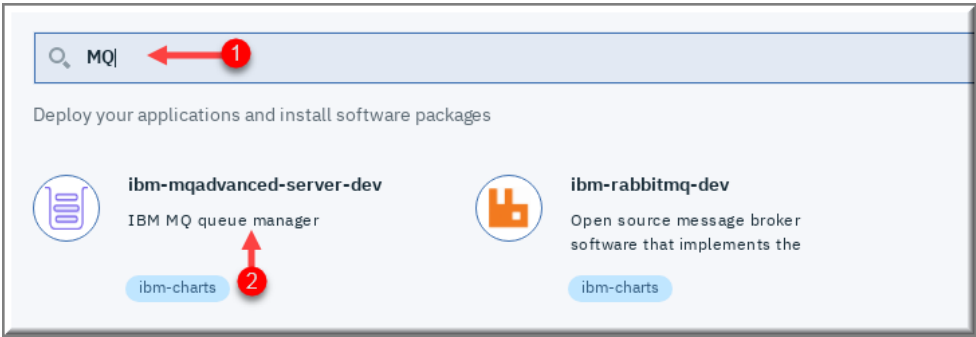

Note: The network services are provided by Kubernetes based on the type which can be [ClusterIP](#) or [NodePort](#). Note that Db2 can be deployed on any available worker node (affinity can be defined, if desired) and cluster IP addresses are private addresses visible within the cluster only.



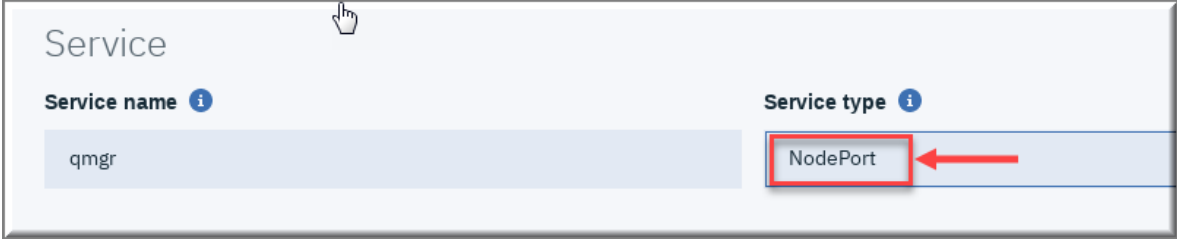
The services are interfaces between external and internal communication and are managed internally by the [iptables](#). [Refer to the second lab for details.]

__ll. Click [dev-ibm-db2oltp-dev-db2](#)

Service details	
Type	Detail
Name	dev-ibm-db2oltp-dev-db2
Namespace	default
Creation time	Mar 28th 2018 at 11:27 AM
Type	NodePort
Labels	app=dev-ibm-db2oltp-dev,chart=ibm-db2oltp-dev-2.0.0,component=db2,heritage=Tiller,release=dev
Selector	app=dev-ibm-db2oltp-dev
IP	10.0.0.52
Port	ibm-db2oltp-dev 50000/TCP; ibm-db2oltp-dev-text 55000/TCP
Node port	ibm-db2oltp-dev 30728/TCP ibm-db2oltp-dev-text 31664/TCP
Session affinity	None

Step	Action																												
__mm.	<p>Note that this service is of the type NodePort, which allows traffic to be routed from the external network through workers' IP addresses to the Docker containers running within the same pod.</p> <p>__nn. Review the Port and NodePort definition ibm-db2oltp-dev 30728/TCP.</p> <table border="1"> <tr> <td>IP</td><td>10.0.0.52</td></tr> <tr> <td>Port</td><td>ibm-db2oltp-dev 50000/TCP; ibm-db2oltp-dev-text 55000/TCP</td></tr> <tr> <td>Node port</td><td>ibm-db2oltp-dev 30728/TCP ibm-db2oltp-dev-text 31664/TCP</td></tr> </table> <p>__oo. Conclusions from above:</p> <ul style="list-style-type: none"> ✓ Db2 is running in a container with an IP address 10.0.0.52 (This could differ in your case.) which is on host 192.168.142.103 (This may also differ for you. You can determine this by looking at Hamburger  Workloads StatefulSets) ✓ Two TCP ports 50000 (db2 instance) and 55000 (Text search) are available. ✓ Through NodePort, access is available to 10.0.0.52:50000 and 10.0.0.52:55000 via any host IP of workers' node at port 30728 and 31664. <div>  <p>Note: The Db2 Docker container always uses port 50,000 and 55,000 and as many containers can be deployed. But each will have different high port through which connections from external network can be made. This abstraction provides agility and elasticity and helps in automation.</p> </div> <p>__pp. Click Services.</p>  <p>__qq. Click dev-ibm-db2oltp-dev</p> <table border="1"> <tr> <th>Type</th><th>Detail</th></tr> <tr> <td>Name</td><td>dev-ibm-db2oltp-dev</td></tr> <tr> <td>Namespace</td><td>default</td></tr> <tr> <td>Creation time</td><td>Mar 28th 2018 at 11:27 AM</td></tr> <tr> <td>Type</td><td>ClusterIP</td></tr> <tr> <td>Labels</td><td>app=dev-ibm-db2oltp-dev,chart=ibm-db2oltp-dev-2.0.0,component=db2,heritage=Tiller,release=dev</td></tr> <tr> <td>Selector</td><td>app=dev-ibm-db2oltp-dev</td></tr> <tr> <td>IP</td><td>None</td></tr> <tr> <td>Port</td><td>main 50000/TCP; text 55000/TCP; db2hadrp 60006/TCP; db2hadrh 60007/TCP</td></tr> <tr> <td>Node port</td><td>None</td></tr> <tr> <td>Session affinity</td><td>None</td></tr> </table>	IP	10.0.0.52	Port	ibm-db2oltp-dev 50000/TCP; ibm-db2oltp-dev-text 55000/TCP	Node port	ibm-db2oltp-dev 30728/TCP ibm-db2oltp-dev-text 31664/TCP	Type	Detail	Name	dev-ibm-db2oltp-dev	Namespace	default	Creation time	Mar 28th 2018 at 11:27 AM	Type	ClusterIP	Labels	app=dev-ibm-db2oltp-dev,chart=ibm-db2oltp-dev-2.0.0,component=db2,heritage=Tiller,release=dev	Selector	app=dev-ibm-db2oltp-dev	IP	None	Port	main 50000/TCP; text 55000/TCP; db2hadrp 60006/TCP; db2hadrh 60007/TCP	Node port	None	Session affinity	None
IP	10.0.0.52																												
Port	ibm-db2oltp-dev 50000/TCP; ibm-db2oltp-dev-text 55000/TCP																												
Node port	ibm-db2oltp-dev 30728/TCP ibm-db2oltp-dev-text 31664/TCP																												
Type	Detail																												
Name	dev-ibm-db2oltp-dev																												
Namespace	default																												
Creation time	Mar 28th 2018 at 11:27 AM																												
Type	ClusterIP																												
Labels	app=dev-ibm-db2oltp-dev,chart=ibm-db2oltp-dev-2.0.0,component=db2,heritage=Tiller,release=dev																												
Selector	app=dev-ibm-db2oltp-dev																												
IP	None																												
Port	main 50000/TCP; text 55000/TCP; db2hadrp 60006/TCP; db2hadrh 60007/TCP																												
Node port	None																												
Session affinity	None																												

Step	Action
	<p>__rr. This network service is assigned the type ClusterIP which restricts access to within the cluster and keeps it isolated.</p> <div data-bbox="313 373 1568 541">  <p>Note: We have seen all the essential ingredients for an automated install and deployments of Db2 in a private cloud environment run by IBM Cloud Private.</p> <p>This is a new paradigm that DBAs should adopt to stay current in a rapidly changing technological environment.</p> </div>
2	<p><u>Deploy IBM MQ</u></p> <p>__a. Click Hamburger  ⇒ Catalog ⇒ Helm Charts</p> <p>__b. Type MQ in the search box. Click IBM MQ queue manager.</p> <div data-bbox="297 779 1268 1113">  </div> <p>__c. Review the IBM MQ Helm Chart readme file.</p> <p>__d. Click Configure.</p> <p>__e. Please pay attention and enter the information correctly to avoid online download of the image.</p> <p>__f. Type <i>Release name</i> qdev, select <i>Target namespace</i> default and check I have read and agreed to the license agreement.</p> <div data-bbox="297 1444 1409 1785">  </div>

Step	Action
<p>__g. Scroll to the Image section.</p> <p>__h. For <i>Image Repository</i>, type <code>store/ibmcorp/mqadvanced-server-dev</code> and Tag <code>9</code>. [Type carefully. No typos, please.]</p>	<div data-bbox="297 455 1281 747">  </div> <p>__i. Scroll to <i>Persistence</i>, Enter <code>glusterfs-storage</code> in <i>Storage Class name</i>.</p> <div data-bbox="297 846 1206 1245">  </div> <p>__j. In the <i>Service</i> section, change <i>Service type</i> to <code>NodePort</code>.</p> <div data-bbox="297 1344 1468 1581">  </div> <p>__k. In <i>Queue manager</i>, type <i>Queue manager name</i> <code>qmgr</code>, <i>Admin password</i> as <code>password</code> and do not specify any password for <i>App</i>.</p>

Step	Action																		
	<div><div>Queue manager</div><div><div>Queue manager name ⓘ</div><div>Admin password ⓘ</div></div><div><div>qmgr</div><div>..... password</div></div></div>																		
__l.	Click Install .																		
	<div><div>Cancel</div><div>Install</div></div>																		
__m.	Click View Helm Release .																		
	<div><div>✓</div><div>Installation complete</div><div><div>View Helm Release</div></div><div>Return to Catalog</div></div>																		
__n.	Click qdev																		
	<div><div>20 items per page 1-2 of 2 items</div><table><tr><th>NAME^</th><th>NAMESPACE^</th><th>STATUS^</th><th>CHART NAME^</th><th>CURRENT VERSION^</th><th>AVAILABLE VERSION^</th></tr><tr><td>dev</td><td>default</td><td>Deployed</td><td>ibm-db2oltp-dev</td><td>2.0.0</td><td>Up To Date</td></tr><tr><td>qdev</td><td>default</td><td>Deployed</td><td>ibm-mqadvanced-server-dev</td><td>1.1.0</td><td>Up To Date</td></tr></table></div>	NAME^	NAMESPACE^	STATUS^	CHART NAME^	CURRENT VERSION^	AVAILABLE VERSION^	dev	default	Deployed	ibm-db2oltp-dev	2.0.0	Up To Date	qdev	default	Deployed	ibm-mqadvanced-server-dev	1.1.0	Up To Date
NAME^	NAMESPACE^	STATUS^	CHART NAME^	CURRENT VERSION^	AVAILABLE VERSION^														
dev	default	Deployed	ibm-db2oltp-dev	2.0.0	Up To Date														
qdev	default	Deployed	ibm-mqadvanced-server-dev	1.1.0	Up To Date														
__o.	Note four artifacts Secret , Service , StatefulSet and Pod were created.																		

__p. Click *StatefulSet* `qdev-ibm-mq`.

StatefulSet		
NAME	DESIRED	CURRENT
<code>qdev-ibm-mq</code>	1	1

__q. Click *Logs*. The MQ operational status should display as *running*.

qdev-ibm-mq

Overview

Events

Logs

Search

qdev-ibm-mq-0


qmgr

```


14 : SET CHLAUTH('DEV.APP.SVRCONN') TYPE(ADDRESSMAP) ADDRESS('*') USERSRC(CHANNEL) CHCKCLNT(REQUIRED) DESCR('Allows
AMQ8877: IBM MQ channel authentication record set.
15 : SET CHLAUTH('DEV.ADMIN.SVRCONN') TYPE(BLOCKUSER) USERLIST('nobody') DESCR('Allows admins on ADMIN channel') ACT
AMQ8877: IBM MQ channel authentication record set.
16 : SET CHLAUTH('DEV.ADMIN.SVRCONN') TYPE(USERMAP) CLNTUSER('admin') USERSRC(CHANNEL) DESCR('Allows admin user to c
AMQ8877: IBM MQ channel authentication record set.
:
: * Developer TLS
17 : ALTER QMGR CERTLABL('queuemanagercertificate')
AMQ8005: IBM MQ queue manager changed.
:
: * Developer listener
18 : DEFINE LISTENER('DEV.LISTENER.TCP') TRPTYPE(TCP) PORT(1414) CONTROL(QMGR) REPLACE
AMQ8021: Request to start IBM MQ listener accepted.
AMQ8626: IBM MQ listener created.
19 : START LISTENER('DEV.LISTENER.TCP')
No commands have a syntax error.
All valid MQSC commands were processed.
19 MQSC commands read.
-----
Monitoring Queue Manager qmgr
QMNAME(qmgr)
IBM MQ Queue Manager qmgr is now fully running
Server mqweb started with process ID 378.

```

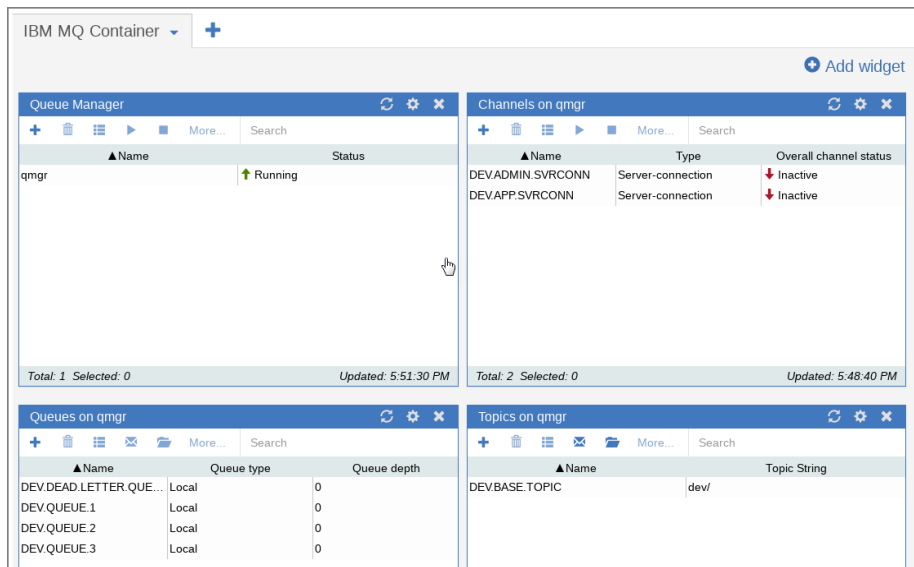
STATUS(Running)

__r. Click *Hamburger*  *Network Access* *Services*

__s. Locate and click `qdev-ibm-mq`

Step	Action																						
	<div data-bbox="297 287 1318 875"> <table> <thead> <tr> <th>Type</th><th>Detail</th></tr> </thead> <tbody> <tr> <td>Name</td><td>qdev-ibm-mq</td></tr> <tr> <td>Namespace</td><td>default</td></tr> <tr> <td>Creation time</td><td>Mar 28th 2018 at 5:18 PM</td></tr> <tr> <td>Type</td><td>NodePort</td></tr> <tr> <td>Labels</td><td>app=qdev-ibm-mq,chart=ibm-mqadvanced-server-dev-1.1.0,heritage=Tiller,release=qdev</td></tr> <tr> <td>Selector</td><td>app=qdev-ibm-mq</td></tr> <tr> <td>IP</td><td>10.0.0.123</td></tr> <tr> <td>Port</td><td>qmgr-server 1414/TCP; qmgr-web 9443/TCP</td></tr> <tr> <td>Node port</td><td>qmgr-server 30779/TCP qmgr-web 32404/TCP</td></tr> <tr> <td>Session affinity</td><td>None</td></tr> </tbody> </table> </div> <p>__t. Note the <i>Service type</i> NodePort and note the high port number for qmgr-web. In our case, this is 32404 (Your case might be different.)</p> <p>__u. Open the second tab in the browser.</p> <p>__v. Type the address <a href="https://192.168.142.101:<your port>">https://192.168.142.101:<your port> (Replace the port number with the one you see in your case.)</p> <p>__w. You are directed to IBM MQ Console. Type User Name: admin and Password: password.</p> <div data-bbox="297 1239 1023 1617"> <div>IBM MQ Console - Login</div> <div>Please enter your username and password</div> <div> <div>User Name: admin</div> <div>Password: password</div> </div> <div>Please note that after some time you will be signed out automatically and asked to sign in again</div> <div>Login</div> </div> <div>  <p>Note: Be patient. We have only one CPU and one SSD drive. We are running five VMs and more than 120+ docker containers.</p> </div>	Type	Detail	Name	qdev-ibm-mq	Namespace	default	Creation time	Mar 28th 2018 at 5:18 PM	Type	NodePort	Labels	app=qdev-ibm-mq,chart=ibm-mqadvanced-server-dev-1.1.0,heritage=Tiller,release=qdev	Selector	app=qdev-ibm-mq	IP	10.0.0.123	Port	qmgr-server 1414/TCP; qmgr-web 9443/TCP	Node port	qmgr-server 30779/TCP qmgr-web 32404/TCP	Session affinity	None
Type	Detail																						
Name	qdev-ibm-mq																						
Namespace	default																						
Creation time	Mar 28th 2018 at 5:18 PM																						
Type	NodePort																						
Labels	app=qdev-ibm-mq,chart=ibm-mqadvanced-server-dev-1.1.0,heritage=Tiller,release=qdev																						
Selector	app=qdev-ibm-mq																						
IP	10.0.0.123																						
Port	qmgr-server 1414/TCP; qmgr-web 9443/TCP																						
Node port	qmgr-server 30779/TCP qmgr-web 32404/TCP																						
Session affinity	None																						

__x. The IBM MQ console displays. Allow time for it to load all panes.



The screenshot shows the IBM MQ console interface with four panes:

- Queue Manager:** Displays the status of the queue manager. The table shows:

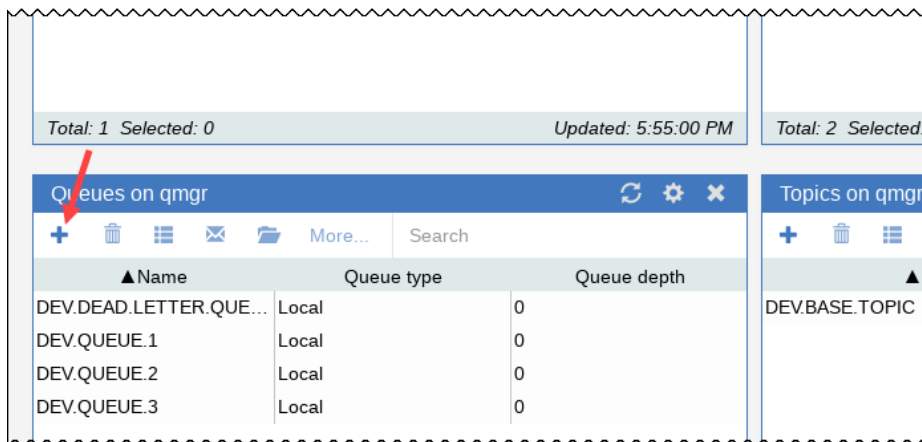
Name	Status
qmgr	Running
- Channels on qmgr:** Displays the status of channels. The table shows:

Name	Type	Overall channel status
DEV.ADMIN.SVRCONN	Server-connection	Inactive
DEV.APP.SVRCONN	Server-connection	Inactive
- Queues on qmgr:** Displays the status of queues. The table shows:

Name	Queue type	Queue depth
DEV.DEAD.LETTER.QUE...	Local	0
DEV.QUEUE.1	Local	0
DEV.QUEUE.2	Local	0
DEV.QUEUE.3	Local	0
- Topics on qmgr:** Displays the status of topics. The table shows:

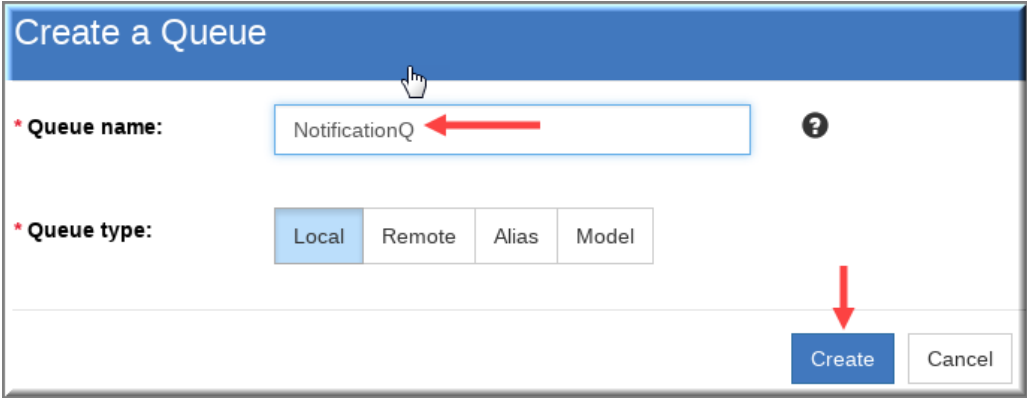
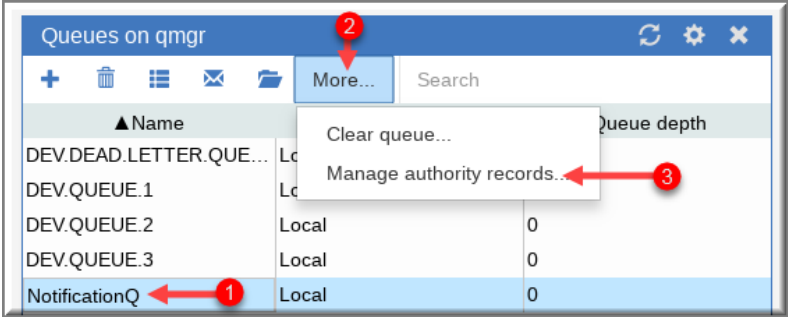
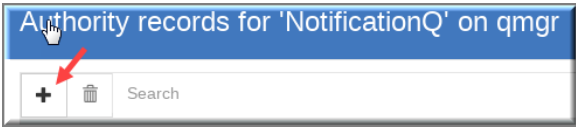
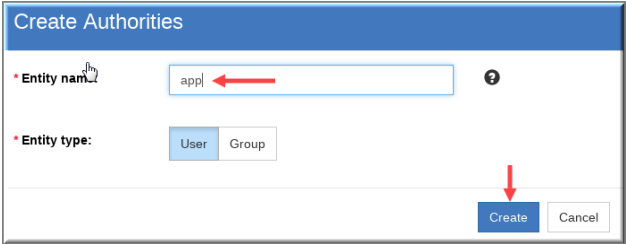
Name	Topic String
DEV.BASE.TOPIC	dev/

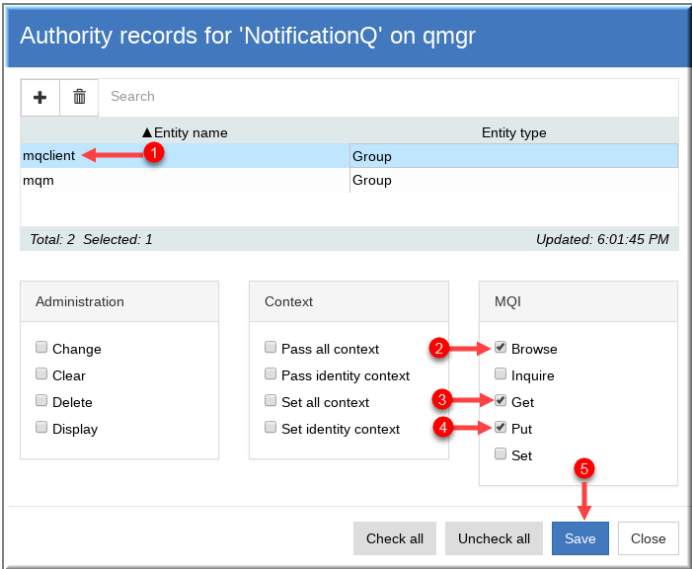

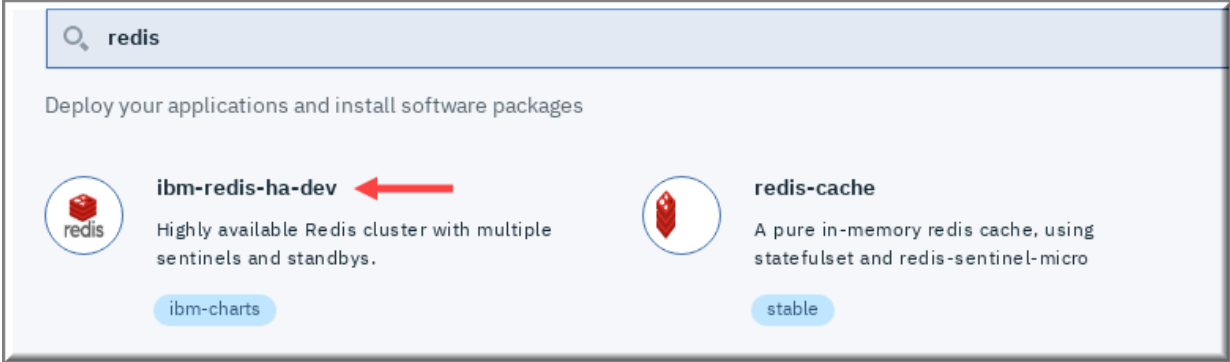
__y. Click + in the bottom left **Queues on qmgr** to add a Queue.


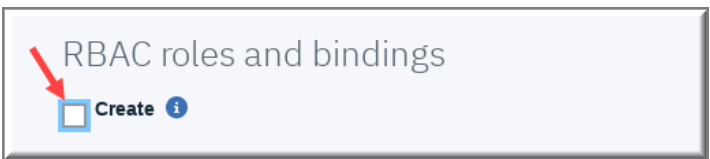
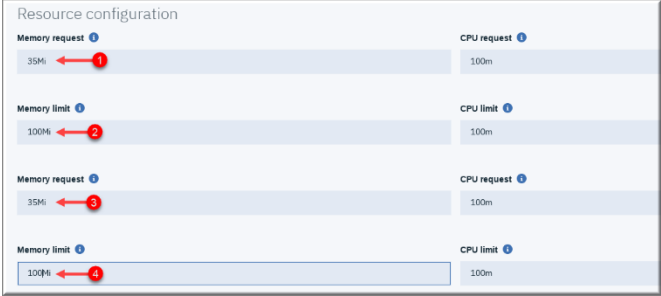
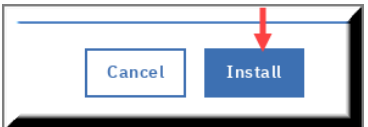


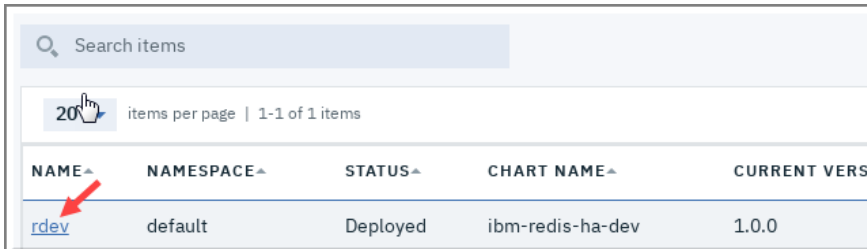
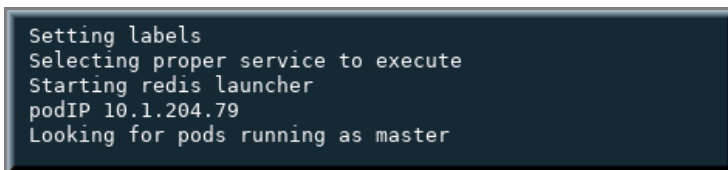


This close-up screenshot focuses on the 'Queues on qmgr' pane. A red arrow points to the '+' button in the top-left corner of the pane's toolbar, indicating where to click to add a new queue. The table below the toolbar shows the same queue data as in the previous screenshot.

__z. Type **NotificationQ** (Type carefully. No typos please.) and click **Create**.

Step	Action
	<div data-bbox="297 289 1318 684">  </div> <p data-bbox="191 716 1328 747">__aa. Select NotificationQ, click More... and select Manage authority records.</p> <div data-bbox="297 783 1081 1098">  </div> <p data-bbox="191 1129 383 1161">__bb. Click +</p> <div data-bbox="297 1194 870 1320">  </div> <p data-bbox="191 1352 821 1383">__cc. Type Entity name: appl and click Create.</p> <div data-bbox="297 1417 919 1659">  </div>

Step	Action
	<p data-bbox="191 289 1078 321">__dd. Click mqclient. Check Browse, Get and Put and click Save.</p> <div data-bbox="297 354 985 919">  </div> <p data-bbox="191 951 1179 982">__ee. Click Close. Click x to close the browser tab for IBM MQ dashboard.</p>
4	<p data-bbox="191 1083 675 1115"><u>Deploy Redis- In Memory Cache</u></p> <p data-bbox="191 1178 927 1209">__ff. Click Hamburger  Catalog Helm Charts</p> <p data-bbox="191 1251 1325 1283">__gg. Type Redis in the search box. Click ibm-redis-ha-dev for my-local-charts.</p> <div data-bbox="297 1316 1521 1675">  </div> <p data-bbox="191 1707 873 1738">__hh. Review the ibm-redis-ha-dev readme file.</p>

Step	Action
	<p data-bbox="191 289 519 321">__ii. Click Configure.</p> <p data-bbox="191 352 1498 426">__jj. Type <i>Release name</i> rdev, select <i>Target namespace</i> default and check I have read and agreed to the license agreement.</p> <div data-bbox="297 457 1490 814">  </div> <p data-bbox="191 846 743 877">__kk. Scroll to <i>RBAC roles and bindings</i>.</p> <p data-bbox="191 909 519 940">__ll. Uncheck Create.</p> <div data-bbox="297 972 1003 1129">  </div> <p data-bbox="191 1161 1498 1234">__mm. Scroll to <i>Resource configuration</i> and change Memory request to 35Mi and Memory limit to 100Mi.</p> <div data-bbox="297 1266 954 1560">  </div> <p data-bbox="191 1591 487 1623">__nn. Click Install.</p> <div data-bbox="297 1654 662 1780">  </div>

Step	Action
__oo.	Click View Helm Release .
__pp.	Click rdev .
	
__qq.	Please notice the artifacts <i>Service Account</i> , <i>Service</i> , <i>Deployment</i> and <i>Pods</i> created for Redis in memory cache database.
__rr.	Scroll to Deployment and click rdev-ibm-redis-ha-dev-server
__ss.	Click Logs .
	
	<div>  <p>Note: We have deployed Db2, MQ and Redis using IBM Cloud Private Catalog. We connect these deployments together in later labs to demonstrate a microservices application that uses IBM MQ, In-memory cache Redis and backend as Db2.</p> </div>
__tt.	Navigate to Hamburger ⇒ Platform ⇒ Nodes . Click each worker node to find the host in which Db2, MQ and Redis are running.
	
__uu.	Notice that Kubernetes scheduler has spread them across all three worker nodes.
__vv.	If any of the pods/workers is lost, Kubernetes starts the pod on other available worker nodes.

Step	Action
	<p>__ww. We have only one master node running, which is a single point of failure. The Enterprise edition allows the creation of multiple master nodes for high availability. Due to resource constraints, we created only one master node.</p>
	<h2 data-bbox="186 447 526 489"><u>Troubleshooting</u></h2> <p data-bbox="186 520 1580 590">__a. If your chart (db2, MQ or Liberty) did not install properly or it appears to be in perpetual pending mode, take the following steps.</p> <ul style="list-style-type: none"> <li data-bbox="293 621 1354 653">✓ Be sure you specified the image repository name correctly, as given in the lab. <li data-bbox="293 684 1092 716">✓ Be sure you specified the tag properly, as given in the lab. <li data-bbox="293 747 1370 779">✓ Be sure you specified the Storage Class Name correctly – glusterfs-storage <li data-bbox="293 810 1179 842">✓ Not specifying Storage Class properly is the most common error. <li data-bbox="293 873 1070 905">✓ If the pod is still in pending state, delete the deployment. <ul style="list-style-type: none"> <li data-bbox="370 936 1130 968">__i. Go to Hamburger ⇒ Workload ⇒ Helm release. <li data-bbox="370 999 1325 1031">__ii. Select the release, click the three vertical dots and click delete. <li data-bbox="370 1062 1568 1131">__iii. Go to Hamburger ⇒ Platform ⇒ Storage ⇒ Persistent Volume Claim and delete it. <li data-bbox="293 1167 610 1199">✓ Repeat the exercise. <p data-bbox="186 1230 1471 1262">__b. If you forgot to specify MQ password, you can obtain the password from the command line.</p> <p data-bbox="186 1293 505 1325">__c. Type command:</p> <pre data-bbox="293 1356 1455 1377"># kubectl get secret qdev-ibm-mq -o json jq -r .data.adminPassword base64 -d</pre>

Section 6: Lab Summary

In this section, you learned how to deploy services from Helm charts using the ICP Catalog feature.

Section 7: Deploying a Microservices Application in ICP

Purpose: In previous labs, you worked with a single container application (the 2048 game). However, in a real-world situation, the value of ICP is in being able to quickly deploy and manage complex applications which may consist of many microservices. In this lab, you will learn how you can use ICP to deploy such an application.

The example application used here is based on the work from the IBM Cloud team and is available at <https://github.com/IBMStockTrader>

The microservices stock trader application is based on the following Docker containers.

Component	Docker container
Db2	store/ibmcorp/db2_developer_c:11.1.3.3-x86_64
MQ	store/ibmcorp/mqadvanced-server-dev:9.0.3
Redis	ibmcom/redis-ha:4.0.6-r0
Liberty	store/ibmcorp/websphere-liberty:javaee7
Liberty-Portfolio	poticpcluster.icp:8500/stocktrader/liberty/portfolio:1.0.1
Liberty-Trader	poticpcluster.icp:8500/stocktrader/liberty/trader:1.0.1
Liberty-Loyalty	poticpcluster.icp:8500/stocktrader/liberty/loyalty:1.0.1
Liberty-Notify-Twitter	poticpcluster.icp:8500/stocktrader/liberty/notify-twitter:1.0.1
Liberty-Notify-Slack	poticpcluster.icp:8500/stocktrader/liberty/notify-slack:1.0.1
Liberty-Messaging	poticpcluster.icp:8500/stocktrader/liberty/messaging:1.0.1
Liberty-stockquote	poticpcluster.icp:8500/stocktrader/liberty/stockquote:1.0.1
Nodejs-Trader	poticpcluster.icp:8500/stocktrader/nodejs/trader:1.0.1

- In previous lab exercises, we saw the build process for Db2, MQ and Redis.
- In this lab exercise, we describe the process of building other containers for different microservices components and deploy them to the IBM Cloud Private cluster.
- The **Portfolio** microservice communicates with **Db2** for persistence storage of data in relational tables. This microservice receives HTTP requests (GET, PUT, POST and DELETE) from either Liberty based **Trader** GUI and Node.js-based **Trader** GUI.
- The **Portfolio** microservice using JMS puts messages in IBM **MQ** and **Messaging** microservice consumes those messages from the MQ.


	<ul style="list-style-type: none"> • The Loyalty microservice determines the loyalty level of a given portfolio owner, based on their total portfolio value. It provides notifications whenever the loyalty level changes. When it detects a change in level, it does a POST to an IBM Cloud Function (earlier aka OpenWhisk) action sequence, which builds a message and posts it to a Slack channel (#slack-test on ibm-cloud.slack.com) using notify-slack microservice. • The notify-twitter microservice sends a tweet via @IBMStockTrader account on Twitter. • Both notify-twitter and notify-slack microservices use the same network service and if both of these microservices are installed, the message to either Slack channel or Twitter will be random. The Itsio routing rules could be used to determine as which gets used and under what conditions. • The stockquote microservice gets the price of a specified stock. It hits an API in API Connect, which drives a call to 'Quandl.com' to get the actual data. This service uses Redis for caching. When a quote is requested, it first checks to see if the answer is in the cache, and if so, whether the quote is less than 24 hours old. (Quandl only returns the previous business day's closing price.) If so, just use that. Otherwise (or if any exceptions occur communicating with Redis), it drives the REST call to API Connect as usual, then adds it to Redis so it's there for next time. • Note: Due to the time constraints of this lab session, we have automated many of the deployment tasks in a series of scripts that you will review and run. • Note: The runtime components of this application have already been pre-built, since it is not the intent of this lab to focus on building the application. The github link above provides details on how the application can be built.
--	---

Tasks:	<p>Tasks you will complete in this lab exercise include:</p> <ul style="list-style-type: none"> • Build Docker Container for the Stock Trader Microservices • Push the Docker Containers to the IBM Private Registry • Create the Db2 tables for the application • Deploy the Microservices • Expose the microservice application • Check the Redis Server and MQ • Run the Stock Trader Application
--------	---

Section 7: Lab Instructions

Step	Action
1	<p><u>Building the Docker Containers for Each Microservice</u></p> <p>__a. Switch to the GNOME Terminal command line.</p> <p>__b. Login as root using the “su” command. The password is “password”.</p> <p>__c. In order to ensure that the environment is correctly setup, run “source ~/.bashrc”</p> <p>__d. Type cd8 to switch the lab directory to 08ms (microservices).</p> <pre>[root@node01 07ta]# cd8 [root@node01 08ms]#</pre> <p>__e. Review 00-build-docker-containers-DO-NOT-RUN-IF-NO-Internet script. [Do not run.]</p> <pre>CWD=\$PWD DIRS=\$(find . -mindepth 1 -maxdepth 1 -type d -printf '%f\n' sort) for dir in \$DIRS do echo ===== echo \$dir - Running dockerbuild echo ===== cd \$dir ./01-builddocker cd \$CWD echo ===== echo done</pre> <p>__f. The above script runs the 01-builddocker script from each of the subdirectories. This script creates the Docker container for each microservice.</p> <p>__g. Run ls -l</p> <pre>[root@node01 08ms]# ls -l total 32</pre>

Step	Action
	<pre>-rwxr-xr-x 1 root root 796 Apr 16 09:58 00-build-docker- containers-DO-NOT-RUN-IF-NO-Internet -rwxr-xr-x 1 root root 816 Apr 16 10:25 01-push-image-to- local-registry -rwxr-xr-x 1 db2psc db2psc 973 Apr 16 10:27 02-deploy-docker- containers drwxr-xr-x 3 db2psc db2psc 231 Apr 16 10:18 03-portfolio drwxr-xr-x 3 db2psc db2psc 147 Apr 16 10:18 04-trader drwxr-xr-x 3 db2psc db2psc 147 Apr 16 10:19 05-stock-quote drwxr-xr-x 3 db2psc db2psc 147 Apr 16 10:19 06-messaging drwxr-xr-x 3 db2psc db2psc 189 Apr 16 10:19 07-notification- slack drwxr-xr-x 3 db2psc db2psc 191 Apr 16 10:19 08-notification- twitter drwxr-xr-x 3 db2psc db2psc 147 Apr 16 10:19 09-loyalty-level drwxr-xr-x 3 db2psc db2psc 158 Apr 16 10:19 10-trader-nodejs -rwxr-xr-x 1 db2psc db2psc 3295 Apr 9 22:59 20-kg1 -rwxr-xr-x 1 db2psc db2psc 506 Apr 8 14:09 30- setImagePullAlways -rwxr-xr-x 1 db2psc db2psc 521 Apr 8 14:09 40- setImageIfNotPresent -rwxr-xr-x 1 db2psc db2psc 449 Apr 8 07:50 50-cleanall -rwxr-xr-x 1 db2psc db2psc 197 Apr 9 23:25 post</pre>
__h.	The directories from 03-portfolio through 10-trader-nodejs are microservices directories.
__i.	Run cd 03-portfolio
	<pre>[root@node01 08ms]# cd 03-portfolio/ [root@node01 03-portfolio]#</pre>
__j.	Run ls -l
	<pre>[root@node01 03-portfolio]# ls -l total 3852 -rwxr-xr-x 1 db2psc db2psc 865 Apr 16 10:08 01-builddocker -rwxr-xr-x 1 root root 1112 Apr 16 10:18 02-pushdocker -rwxr-xr-x 1 db2psc db2psc 1251 Apr 8 21:51 03-createsecrets -rwxr-xr-x 1 db2psc db2psc 696 Apr 6 14:08 04-deploydocker -rwxr-xr-x 1 db2psc db2psc 952 Apr 6 14:08 05-createtables drwxr-xr-x 5 db2psc db2psc 74 Apr 10 13:06 config -rw-r--r-- 1 db2psc db2psc 3905812 Apr 3 23:43 db2jcc4.jar -rw-r--r-- 1 db2psc db2psc 2324 Apr 9 22:24 deploy.yaml -rw-r--r-- 1 db2psc db2psc 141 Apr 4 00:16 Dockerfile -rw-r--r-- 1 db2psc db2psc 474 Apr 8 07:59 tables.sql</pre>

Step	Action
	<p>Build Containers</p> <p>__k. Review 01-builddocker [Do not run].</p> <pre data-bbox="345 394 1498 1129">[root@node01 03-portfolio]# cat 01-builddocker NAMESPACE=stocktrader echo ===== echo Create name space : \$NAMESPACE echo ===== cat << EOF kubectl apply -f - apiVersion: v1 kind: Namespace metadata: name: \$NAMESPACE EOF - - - - IMAGENAME=liberty/portfolio:1.0.1 echo CLUSTERNAME=\$CLUSTERNAME docker build -t \$CLUSTERNAME.icp:8500/\$NAMESPACE/\$IMAGENAME -f Dockerfile .</pre> <p>__l. The above script creates a stocktrader namespace and runs <code>docker build</code> command to build the container as per the name using <code>-t</code> switch.</p> <p>__m. Review Dockerfile.</p> <pre data-bbox="345 1318 1498 1491">[root@node01 03-portfolio]# cat Dockerfile FROM store/ibmcorp/websphere-liberty:javaee7 ADD config /config ADD db2jcc4.jar ./ RUN installUtility install --acceptLicense defaultServer</pre> <p>__n. The above Dockerfile uses the WebSphere Liberty base image. (We have already downloaded the base image.)</p> <div data-bbox="367 1612 1487 1759"> <div data-bbox="367 1612 500 1759">  </div> <div data-bbox="509 1612 1487 1759"> <p>Note: If the Docker image is not present, Docker downloads the image from Docker Store. Please refer to Appendix-A for the procedure to download IBM Docker containers.</p> </div> </div>

Step	Action
	<p>__o. It then adds the config folder (ADD config /config) to the base image, copies db2jcc4.jar to root of the image and runs InstallUtility to create the default server.</p> <p>__p. In the microservices environment, each bundled, similar components are stored in their own Docker container and just deploying an individual container serves the purpose of continuous improvement and delivery.</p> <p>__q. Run tree config.</p> <pre data-bbox="345 625 1040 989">[root@node01 03-portfolio]# tree config/ config/ ├── apps │ └── Portfolio.war ├── configDropins │ └── defaults │ └── keystore.xml ├── resources │ └── security │ └── key.jks └── server.xml</pre> <p>__r. Note that this directory comes from the development organization or the CICD (Continuous Improvement and Delivery) mechanism through GitHub (or any other source control) though Jenkins will trigger the build process, build container and deploy to the right environment.</p> <p>__s. In this session, we described those processes to show individual components so that you can build your pipeline using SCM (Source Control Mechanism) and Jenkins.</p> <p>__t. Note that we have Portfolio.war in apps directory. The security files key.jks and keystore.xml in their respective directories.</p> <p>__u. Review server.xml – through which Liberty uses features, security, JDBC data sources and more.</p> <pre data-bbox="345 1486 1187 1801">[root@node01 03-portfolio]# cat config/server.xml <server description="Portfolio server"> <featureManager> <feature>microProfile-1.3</feature> <feature>jdbc-4.1</feature> <feature>jndi-1.0</feature> <feature>appSecurity-2.0</feature> <feature>openapi-3.0</feature> </featureManager> - - - -</pre>

Step	Action
	<pre data-bbox="349 304 1469 1354"> <connectionManager id="DB2-Connections" minPoolSize="5" maxPoolSize="50"/> <dataSource id="PortfolioDB" jndiName="jdbc/Portfolio/PortfolioDB" connectionManagerRef="DB2-Connections" isolationLevel="TRANSACTION_READ_COMMITTED"> <jdbcDriver> <library name="DB2" description="DB2 JDBC driver jar"> <file id="db2jcc4" name="/db2jcc4.jar"/> </library> </jdbcDriver> <properties.db2.jcc serverName="\${env.JDBC_HOST}" portNumber="\${env.JDBC_PORT}" databaseName="\${env.JDBC_DB}" user="\${env.JDBC_ID}" password="\${env.JDBC_PASSWORD}"/> </dataSource> - - - <webApplication id="Portfolio" name="Portfolio" location="Portfolio.war" contextRoot="/portfolio"> <application-bnd> <security-role id="StockTrader" name="StockTrader"> <special-subject type="ALL_AUTHENTICATED_USERS" id="IBMid"/> </security-role> </application-bnd> </webApplication> </server> </pre> <p data-bbox="243 1396 1485 1701"> <u>v.</u> Note the features this Liberty server uses through featureManager section. <u>w.</u> Review the JDBC connection properties defined through environment variables. We use Kubernetes secrets to provide these values and then Kubernetes transfers them to the environment variables when starting the container. We will demonstrate this connection later in this section. <u>x.</u> We will not run <code>00-build-docker-containers-DO-NOT-RUN-IF-NO-Internet</code> since we already built containers to save time. </p>

Step	Action
2	<p><u>Push Docker Containers into the Private Registry</u></p> <p>__a. IBM Cloud Private provides a local private registry to which we push the Docker container. Usually, the CICD process (through Jenkins) pushes the Docker container to the IBM Cloud Private local registry.</p> <p>__b. Run <code>cd8</code> to change the lab directory.</p> <pre data-bbox="345 562 1498 636">[root@node01 03-portfolio]# cd8 [root@node01 08ms]#</pre> <p>__c. Run <code>cat 03-portfolio/02-pushcontainer</code></p> <pre data-bbox="345 720 1498 1024">[root@node01 08ms]# cat 03-portfolio/02-pushdocker IMAGENAME=liberty/portfolio:1.0.1 echo CLUSTERNAME=\$CLUSTERNAME docker login \$CLUSTERNAME.icp:8500 -u \$DEFAULTUSERNAME -p \$DEFAULTPASSWORD docker push \$CLUSTERNAME.icp:8500/\$NAMESPACE/\$IMAGENAME</pre> <p>__d. After logging in to the local Docker registry, the Docker push command is used to copy the image to the IBM Cloud Private registry.</p> <p>__e. Review <code>01-push-image-to-local-registry</code></p> <pre data-bbox="345 1213 1498 1770">[root@node01 08ms]# cat 01-push-image-to-local-registry - - - - CWD=\$PWD DIRS=\$(find . -mindepth 1 -maxdepth 1 -type d -printf '%f\n' sort) for dir in \$DIRS do echo ===== echo \$dir - Running dockerbuild echo ===== cd \$dir ./02-pushdocker cd \$CWD</pre>

Step	Action																																								
	<pre>echo ===== echo done</pre>																																								
__f.	The above script runs 02-pushdocker in all subdirectories to push the Docker image to the local registry.																																								
__g.	Run 01-push-image-to-local-registry <pre>[root@node01 08ms]# ./01-push-image-to-local-registry - --- CLUSTERNAME=poticpcluster ===== Push image to the local registry ===== WARNING! Using --password via the CLI is insecure. Use -- password-stdin. Login Succeeded The push refers to a repository [poticpcluster.icp:8500/stocktrader/liberty/portfolio]</pre>																																								
__h.	The above script pushes all containers to the IBM Cloud Private local registry.																																								
__i.	Switch to the web UI.																																								
__j.	Click Hamburger  Catalog Images .																																								
	<div><div>20 items per page 1-9 of 9 items1 of 1 pages < ></div><table><tr><th>NAME ^</th><th>OWNER</th><th>SCOPE</th><th>ACTION</th></tr><tr><td>stocktrader/liberty/loyalty</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/messaging</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/notify-slack</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/notify-twitter</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/portfolio</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/stockquote</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/liberty/trader</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>stocktrader/nodejs/trader</td><td>stocktrader</td><td>namespace</td><td>⋮</td></tr><tr><td>ta/liberty/employee</td><td>ta</td><td>namespace</td><td>⋮</td></tr></table></div>	NAME ^	OWNER	SCOPE	ACTION	stocktrader/liberty/loyalty	stocktrader	namespace	⋮	stocktrader/liberty/messaging	stocktrader	namespace	⋮	stocktrader/liberty/notify-slack	stocktrader	namespace	⋮	stocktrader/liberty/notify-twitter	stocktrader	namespace	⋮	stocktrader/liberty/portfolio	stocktrader	namespace	⋮	stocktrader/liberty/stockquote	stocktrader	namespace	⋮	stocktrader/liberty/trader	stocktrader	namespace	⋮	stocktrader/nodejs/trader	stocktrader	namespace	⋮	ta/liberty/employee	ta	namespace	⋮
NAME ^	OWNER	SCOPE	ACTION																																						
stocktrader/liberty/loyalty	stocktrader	namespace	⋮																																						
stocktrader/liberty/messaging	stocktrader	namespace	⋮																																						
stocktrader/liberty/notify-slack	stocktrader	namespace	⋮																																						
stocktrader/liberty/notify-twitter	stocktrader	namespace	⋮																																						
stocktrader/liberty/portfolio	stocktrader	namespace	⋮																																						
stocktrader/liberty/stockquote	stocktrader	namespace	⋮																																						
stocktrader/liberty/trader	stocktrader	namespace	⋮																																						
stocktrader/nodejs/trader	stocktrader	namespace	⋮																																						
ta/liberty/employee	ta	namespace	⋮																																						

Step	Action
	<p>__k. The images are now stored in IBM Cloud Private registry and through our deployment process, the images can be pulled by any worker node.</p> <p>__l. Switch back to the command line.</p>
3	<p><u>Create Database Tables</u></p> <p>__a. Change directory to 03-portfolio.</p> <pre data-bbox="345 558 1498 600">[root@node01 08ms]# cd 03-portfolio/</pre> <p>__b. Review script 05-createtables</p> <pre data-bbox="345 684 1498 888">[root@node01 03-portfolio]# cat 05-createtables DB2POD=\$(kubectl -n default get pods --selector app=dev-ibm-db2oltp-dev -o jsonpath='{.items[0].metadata.name}') kubectl -n default cp ./tables.sql \$DB2POD:/tmp kubectl -n default exec -it \$DB2POD -- /bin/bash -c "su - db2psc -c \"db2 -tvf /tmp/tables.sql\""</pre> <p>__c. The script determines the Db2 pod name using label <code>app= dev-ibm-db2oltp-dev</code>. We then copy the create table script to <code>/tmp</code> folder of the Db2 container and then run <code>kubect1 exec</code> command to run the Db2 command to create tables.</p> <p>__d. Run 05-createtables</p> <pre data-bbox="345 1108 1498 1770">[root@node01 03-portfolio]# ./05-createtables Get the db2 pod name Db2 pod name = dev-ibm-db2oltp-dev-0 CONNECT TO PSDB Database Connection Information Database server = DB2/LINUX8664 11.1.3.3 SQL authorization ID = DB2PSC Local database alias = PSDB CREATE TABLE Portfolio (owner VARCHAR(32) NOT NULL, total DOUBLE, loyalty VARCHAR(8), PRIMARY KEY(owner)) DB20000I The SQL command completed successfully. CREATE TABLE Stock (owner VARCHAR(32) NOT NULL, symbol VARCHAR(8) NOT NULL, shares INTEGER, price DOUBLE, total DOUBLE, dateQuoted DATE, FOREIGN KEY (owner) REFERENCES Portfolio(owner) ON DELETE CASCADE, PRIMARY KEY(owner, symbol)) DB20000I The SQL command completed successfully. CONNECT RESET</pre>

Step	Action
	<pre>DB20000I The SQL command completed successfully.</pre>
4	<p><u>Deploy Microservices</u></p> <p>__a. Review 02-deploy-docker-containers</p> <pre>[root@node01 08ms]# cat 02-deploy-docker-containers CWD=\$PWD DIRS=\$(find . -mindepth 1 -maxdepth 1 -type d -not -path ./07- notification-slack -printf '%f\n' sort) for dir in \$DIRS do cd \$dir ./03-createsecrets ./04-deploydocker cd \$CWD done</pre> <p>__b. The above script runs 03-createsecrets and 04-deploydocker.</p> <p>__c. Run <code>cat */03-createsecrets</code></p> <pre>[root@node01 08ms]# cat */03-createsecrets # jwt - json web token secret kubectl -n stocktrader \ create secret generic jwt \ --from-literal=audience=stock-trader \ --from-literal=issuer=http://stock-trader.ibm.com # Db2 secret kubectl -n stocktrader \ create secret generic db2 \ --from-literal=id=db2psc \ --from-literal=pwd=password \ --from-literal=host=dev-ibm-db2oltp-dev.default.svc.cluster.local \ --from-literal=port=50000 \ --from-literal=db=PSDB</pre> <p>__d. Scroll to see that we created a secret object for each microservice (if applicable), which provides runtime credentials.</p> <p>__e. For example, notice the Db2 secret, which has the name of the database, user ID, password, host name and the port number. These values from secret through <code>deploy.yaml</code> are passed to the container in the form of environments variables and then the server.xml picks up these values from the container environment variables.</p>

Step	Action
	<p data-bbox="245 289 1133 321">__f. Run <code>kubectl -n stocktrader get secret db2 -o yaml</code></p> <pre data-bbox="342 342 1489 940">[root@node01 08ms]# kubectl -n stocktrader get secret db2 -o yaml apiVersion: v1 data: db: UFNEQg== host: ZGV2LWlibS1kYjJvbHRwLWRldi5kZWZhdWx0LnN2Yy5jbHVzdGVyLmxvY2Fs id: ZGIychNj port: NTAwMDA= pwd: cGFzc3dvcmQ= kind: Secret metadata: creationTimestamp: 2018-04-10T02:33:45Z name: db2 namespace: stocktrader resourceVersion: "81980" selfLink: /api/v1/namespaces/stocktrader/secrets/db2 uid: 97abbd2f-3c67-11e8-970f-005056271837 type: Opaque</pre> <p data-bbox="245 972 1084 1003">__g. Note the values of the secret are stored in encoded form.</p> <p data-bbox="245 1035 1489 1108">__h. For example: If you want to see the Db2 password, run <code>echo cGFzc3dvcmQ= base64 -d</code></p> <pre data-bbox="342 1129 1489 1203">[root@node01 08ms]# echo cGFzc3dvcmQ= base64 -d password[root@node01 08ms]#</pre> <p data-bbox="245 1234 771 1266">__i. The encoded value is <code>password</code>.</p> <p data-bbox="245 1297 836 1329">__j. Review <code>03-portfolio/deploy.yaml</code></p> <pre data-bbox="342 1350 1489 1812">[root@node01 08ms]# cat 03-portfolio/deploy.yaml apiVersion: extensions/v1beta1 kind: Deployment metadata: name: portfolio spec: replicas: 1 template: metadata: labels: app: portfolio solution: stocktrader id: portfolio version: 1.0.1 spec: containers: - name: portfolio</pre>

Step	Action
	<pre> image: poticpcluster.icp:8500/stocktrader/liberty/portfolio:1.0.1 env: - name: JDBC_HOST valueFrom: secretKeyRef: name: db2 key: host - name: JDBC_PORT valueFrom: secretKeyRef: name: db2 key: port - name: JDBC_DB valueFrom: secretKeyRef: name: db2 key: db - name: JDBC_ID valueFrom: secretKeyRef: name: db2 key: id - name: JDBC_PASSWORD valueFrom: secretKeyRef: name: db2 key: pwd - name: JWT_AUDIENCE valueFrom: secretKeyRef: name: jwt key: audience - name: JWT_ISSUER valueFrom: secretKeyRef: name: jwt key: issuer ports: - containerPort: 9080 - containerPort: 9443 imagePullPolicy: Always --- #Deploy the service apiVersion: v1 kind: Service metadata: name: portfolio-service labels: app: portfolio spec: type: NodePort ports: - name: http protocol: TCP port: 9080 targetPort: 9080 - name: https </pre>


Step	Action
	<pre> protocol: TCP port: 9443 targetPort: 9443 selector: app: portfolio --- #Configure the ingress apiVersion: extensions/v1beta1 kind: Ingress metadata: annotations: kubernetes.io/ingress.class: "nginx" ingress.kubernetes.io/affinity: "cookie" ingress.kubernetes.io/session-cookie-name: "route" ingress.kubernetes.io/session-cookie-hash: "sha1" ingress.kubernetes.io/secure-backends: "true" ingress.kubernetes.io/app-root: "/portfolio" name: portfolio-ingress spec: rules: - host: http: paths: - path: /portfolio backend: serviceName: portfolio-service servicePort: 9443 </pre> <p>__k. The portfolio microservice is deployed in Kubernetes cluster through the aforesaid deploy.yaml file.</p> <p>__l. The salient features of the above deploy.yaml are as follows:</p> <ul style="list-style-type: none"> ✓ The docker container poticpcluster.icp:8500/stocktrader/liberty/portfolio:1.0.1 (from ICP registry) is used to deploy portfolio microservice. It has been given a label app set to portfolio. ✓ The JDBC_HOST environment variable to the docker container is mapped to Kubernetes secret db2 parameter host and other parameters as well. ✓ The Liberty application server is using two ports 9080 (HTTP) and 9553 (HTTPS). ✓ The network service is named portfolio-service and the selector label is set to app:portfolio – which is the glue between network service and the Docker container. This is how the network traffic is routed. The type of the service is NodePort – which allows connections from the proxy server (or any worker node) to these exposed ports.

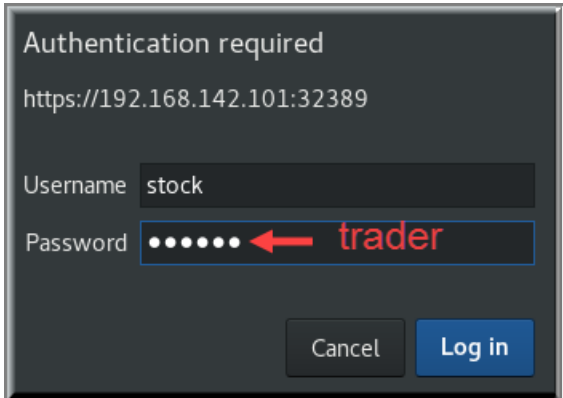
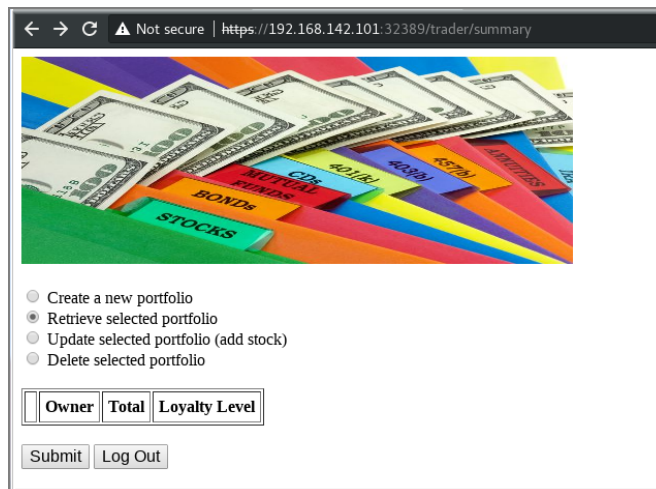
Step	Action
	<ul style="list-style-type: none"> ✓ The optional routing is done by defining ingress named as portfolio-ingress with path set to /portfolio and this ingress is tied to the network service portfolio-service. ✓ The advantage of using the ingress is to reach out to path without having to specify port number and this is done through an Ingress Controller which is a reverse proxy provided through nginx. <p>__m. Review 03-portfolio/04-deploydocker</p> <pre data-bbox="342 611 1466 884">[root@node01 08ms]# cat 03-portfolio/04-deploydocker - --- echo ===== echo Running command \"kubectl --namespace stocktrader apply -f deploy.yaml\" echo ===== kubectl --namespace stocktrader apply -f deploy.yaml</pre> <p>__n. After deploy.yaml is created for each microservice, the kubectl apply -f is used to deploy the objects.</p> <p>__o. After we have seen the above deployment procedure, we can now deploy all microservices.</p> <p>__p. Run 02-deploy-docker-containers</p> <pre data-bbox="342 1167 1466 1814">[root@node01 08ms]# ./02-deploy-docker-containers ===== Create Secrets and build Docker Containers ===== 03-portfolio - Running dockerbuild, create secrets and docker deploy ===== Create secrets for trader container secret "jwt" created secret "db2" created secret "ingress-host" created ===== Deploy Liberty Docker container for trader ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "portfolio" created service "portfolio-service" unchanged ingress "portfolio-ingress" configured ===== 04-trader - Running dockerbuild, create secrets and docker deploy ===== Create secrets for trader container secret "jwt" created secret "oidc" created ===== Deploy Liberty Docker container for trader</pre>

Step	Action
	<pre> ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "trader" created service "trader-service" unchanged ingress "trader-ingress" unchanged ===== 05-stock-quote - Running dockerbuild, create secrets and docker deploy ===== Create secrets for trader container ===== secret "redis" created ===== Deploy Liberty Docker container for stock ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "stockquote" created service "stock-quote-service" unchanged ingress "stock-quote-ingress" unchanged ===== 06-messaging - Running dockerbuild, create secrets and docker deploy ===== Create secrets for messaging container ===== secret "mq" created ===== Deploy Liberty Docker container for messaging ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "messaging" created ===== 08-notification-twitter - Running dockerbuild, create secrets and docker deploy ===== Create secrets for notification twitter container ===== secret "twitter" created ===== Deploy Liberty Docker container for notification twitter ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "notification-twitter" created service "notification-service" unchanged ingress "notification-ingress" unchanged ===== 09-loyalty-level - Running dockerbuild, create secrets and docker deploy ===== Create secrets for notify-level ===== Deploy Liberty Docker container for loyalty ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "loyalty-level" created service "loyalty-level-service" unchanged ingress "loyalty-ingress" unchanged ===== 10-trader-nodejs - Running dockerbuild, create secrets and docker deploy ===== Create secret for ingress-controller to switch to tradr instead of trader </pre>

Step	Action
	<pre>===== secret "ingress-host" created ===== Deploy Liberty Docker container for loyalty ===== Running command "kubectl --namespace stocktrader apply -f deploy.yaml" ===== deployment "tradr" created service "tradr-service" unchanged ingress "nodejs-trader-ingress" configured =====</pre>
__q.	Run the commands to see the status of deployments and pods.
__r.	Run <code>kubectl -n stocktrader get deployments</code>
	<pre>[root@node01 08ms]# kubectl -n stocktrader get deployments NAME DESIRED CURRENT UP-TO-DATE AVAILABLE AGE loyalty-level 1 1 1 1 2m messaging 1 1 1 1 2m notification-twitter 1 1 1 1 2m portfolio 1 1 1 1 2m stockquote 1 1 1 1 2m trader 1 1 1 1 2m tradr 1 1 1 1 2m</pre>
__s.	Run <code>kubectl -n stocktrader get pods</code>
	<pre>[root@node01 08ms]# kubectl -n stocktrader get pods NAME READY STATUS RESTARTS AGE loyalty-level-7b58569b9b-f62pt 1/1 Running 0 3m messaging-559cf6f4cf-rbgf4 1/1 Running 0 3m notification-twitter-585b96f845-kgjbx 1/1 Running 0 3m portfolio-7c568d6cb8-s7vc4 1/1 Running 0 3m stockquote-dbf546b67-fs55t 1/1 Running 0 3m</pre>

Step	Action																																																																																				
		trader-5c5ff75c5d-r42jh 3m	1/1	Running	0																																																																																
		tradr-84784b4d9f-j6vfx 3m	1/1	Running	0																																																																																
5	<u>Expose Microservice Application</u> __a. The entry point for the application that IBM Cloud Team has built starts with the trader microservice using path <code>/trader/summary</code> using secured HTTPS port – 9443. __b. There are multiple ways this application can be run – this is explained to demonstrate how network services work in Kubernetes. __c. Run <code>kubectl -n stocktrader get services</code> <div><pre>[root@node01 08ms]# kubectl -n stocktrader get services</pre><table><thead><tr><th>NAME</th><th>TYPE</th><th>CLUSTER-IP</th><th>EXTERNAL-IP</th><th>PORT(S)</th></tr></thead><tbody><tr><td colspan="5">AGE</td></tr><tr><td>loyalty-level-service</td><td>NodePort</td><td>10.0.0.114</td><td><none></td><td></td></tr><tr><td colspan="5">9080:32410/TCP,9443:30472/TCP 6d</td></tr><tr><td>notification-service</td><td>NodePort</td><td>10.0.0.211</td><td><none></td><td></td></tr><tr><td colspan="5">9080:30855/TCP,9443:32737/TCP 6d</td></tr><tr><td>portfolio-service</td><td>NodePort</td><td>10.0.0.195</td><td><none></td><td></td></tr><tr><td colspan="5">9080:32646/TCP,9443:30104/TCP 6d</td></tr><tr><td>stock-quote-service</td><td>NodePort</td><td>10.0.0.62</td><td><none></td><td></td></tr><tr><td colspan="5">9080:32224/TCP,9443:32306/TCP 6d</td></tr><tr><td>trader-service</td><td>NodePort</td><td>10.0.0.68</td><td><none></td><td></td></tr><tr><td colspan="5">9080:32388/TCP,9443:32389/TCP 6d</td></tr><tr><td>tradr-service</td><td>NodePort</td><td>10.0.0.99</td><td><none></td><td>3000:31007/TCP</td></tr><tr><td colspan="5">6d</td></tr></tbody></table></div> __d. Note the name of the trader service – which is <code>trader-service</code> using NodePort and <code>http</code> port <code>9080</code> is mapped to Node Port <code>32388</code> and HTTPS port <code>9443</code> mapped as <code>32389</code> . We have explicitly defined these ports through <code>deploy.yaml</code> and you will see the same values when you run the command in your lab environment. __e. The name <code>trader-service</code> is in name space <code>stocktrader</code> so the Kubernetes fully qualified domain name (FQDN) will be <code>trader-service.stocktrader.svc.cluster.local</code> __f. You can run this application from within ICP cluster as <code>https://trader-service.stocktrader.svc.cluster.local:9443/trader/summary</code> __g. Note that we have used the local port since we are using local service name. The local port and local Kubernetes FQDN are not visible outside the cluster. __h. You can use cluster IP address by examining the output of <code>kubectl -n stocktrader get service trader-service</code> <div><pre>[root@node01 08ms]# kubectl -n stocktrader get service trader-service</pre><table><thead><tr><th>NAME</th><th>TYPE</th><th>CLUSTER-IP</th><th>EXTERNAL-IP</th><th>PORT(S)</th></tr></thead><tbody><tr><td colspan="5">AGE</td></tr></tbody></table></div>					NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE					loyalty-level-service	NodePort	10.0.0.114	<none>		9080:32410/TCP,9443:30472/TCP 6d					notification-service	NodePort	10.0.0.211	<none>		9080:30855/TCP,9443:32737/TCP 6d					portfolio-service	NodePort	10.0.0.195	<none>		9080:32646/TCP,9443:30104/TCP 6d					stock-quote-service	NodePort	10.0.0.62	<none>		9080:32224/TCP,9443:32306/TCP 6d					trader-service	NodePort	10.0.0.68	<none>		9080:32388/TCP,9443:32389/TCP 6d					tradr-service	NodePort	10.0.0.99	<none>	3000:31007/TCP	6d					NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE				
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)																																																																																	
AGE																																																																																					
loyalty-level-service	NodePort	10.0.0.114	<none>																																																																																		
9080:32410/TCP,9443:30472/TCP 6d																																																																																					
notification-service	NodePort	10.0.0.211	<none>																																																																																		
9080:30855/TCP,9443:32737/TCP 6d																																																																																					
portfolio-service	NodePort	10.0.0.195	<none>																																																																																		
9080:32646/TCP,9443:30104/TCP 6d																																																																																					
stock-quote-service	NodePort	10.0.0.62	<none>																																																																																		
9080:32224/TCP,9443:32306/TCP 6d																																																																																					
trader-service	NodePort	10.0.0.68	<none>																																																																																		
9080:32388/TCP,9443:32389/TCP 6d																																																																																					
tradr-service	NodePort	10.0.0.99	<none>	3000:31007/TCP																																																																																	
6d																																																																																					
NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)																																																																																	
AGE																																																																																					

Step	Action										
	<table><tr><td>trader-service</td><td>NodePort</td><td>10.0.0.68</td><td><none></td><td>9080:32388/TCP,9443:32389/TCP</td></tr><tr><td>6d</td><td></td><td></td><td></td><td></td></tr></table> <p>__i. The cluster IP address is 10.0.0.68 (It may be different in your case).</p> <p>__j. The URL to access the application can be https://10.0.0.68:9443/trader/summary</p> <p>__k. Note that we use a local port as we have direct access to the local IP address of the pod.</p> <p><u>Host Names or Proxy Server</u></p> <p>__l. To access this application from outside the IBM Cloud Private cluster, one has to come through the proxy server. In our lab environment, the node01 (192.168.142.101) is the proxy server.</p> <p>__m. The URL to access the application is https://192.168.142.101:32389/trader/summary</p> <p>__n. Note that we are using the node port when accessing the application through proxy server.</p> <p>__o. Normally, access to the master and workers nodes is prohibited in the actual environment. But in our environment, you have access to these nodes from outside. You could use any nodes and the routing is handled by Kubernetes automatically. For example: You could use URL https://192.168.142.103:32389/trader/summary and the routing to the appropriate pod is automatic.</p> <div><div></div><div><p>Note: The internal routing is managed by iptables rules defined by the Kubernetes when a network service is defined.</p></div></div> <p>__p. An Ingress is a Kubernetes resource that lets you configure an HTTP load balancer for your Kubernetes services. Such a load balancer usually exposes your services to clients outside of your Kubernetes cluster. In other words, Kubernetes ingress is a collection of routing rules that govern how external users access services running in a Kubernetes cluster.</p>	trader-service	NodePort	10.0.0.68	<none>	9080:32388/TCP,9443:32389/TCP	6d				
trader-service	NodePort	10.0.0.68	<none>	9080:32388/TCP,9443:32389/TCP							
6d											

Step	Action
6	<p><u>Run the Microservice Application</u></p> <p>__a. Open a new browser tab to run the application.</p> <p>__b. Type URL: https://192.168.142.101:32389/trader/summary</p> <p>__c. Type Username stock and Password trader. Click Log in.</p> <div data-bbox="633 539 1192 934" data-label="Image">  </div> <p>__d. You should see the main summary page. Note that this is the server JSP with no use of client-side scripting and typically represents a legacy UI. This page is serviced by the trader microservice. Later, we will see Node.js-based web UI which can be plugged in to show the strengths of the microservices-based architecture in which the UI can be independent of the model and controller and easily replaceable.</p> <div data-bbox="532 1165 1183 1650" data-label="Image">  </div> <p>__d. Tick Create a new portfolio. Click Submit.</p> <p>__e. Type Owner PoT and click submit.</p>

Step	Action																																
	<div><div>Owner: <input type="text" value="PoT"/> <input type="button" value="Submit"/></div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$0.00</td><td>Basic</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div><div><div><div>__f.</div><div>Tick Update selected portfolio (add stock). Click Submit.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$0.00</td><td>Basic</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div><div><div><div>__g.</div><div>Note: You need an Internet connection as this request routes to https://www.quandl.com/ to retrieve the stock quote at the end of the previous day.</div></div></div><tr><td></td><td><div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div><tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr></td></tr></div></td></tr></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$0.00	Basic		<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$0.00</td><td>Basic</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div><div><div><div>__g.</div><div>Note: You need an Internet connection as this request routes to https://www.quandl.com/ to retrieve the stock quote at the end of the previous day.</div></div></div><tr><td></td><td><div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div><tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr></td></tr></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$0.00	Basic		<div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div> <tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr>		<div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div> <div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div> <tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr>		<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$151,910.00	Gold
	Owner	Total	Loyalty Level																														
<div><div></div><div></div></div>	PoT	\$0.00	Basic																														
	<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$0.00</td><td>Basic</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div><div><div><div>__g.</div><div>Note: You need an Internet connection as this request routes to https://www.quandl.com/ to retrieve the stock quote at the end of the previous day.</div></div></div><tr><td></td><td><div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div><tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr></td></tr></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$0.00	Basic		<div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div> <tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr>		<div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div> <div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div> <tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr>		<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$151,910.00	Gold										
	Owner	Total	Loyalty Level																														
<div><div></div><div></div></div>	PoT	\$0.00	Basic																														
	<div><div><div>__h.</div><div>Type Stock Symbol IBM and specify 1000 stocks. Click Submit.</div></div></div> <tr><td></td><td><div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div><div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div><tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr></td></tr>		<div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div> <div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div> <tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr>		<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$151,910.00	Gold																				
	<div><div><div>Owner: PoT Stock Symbol: <input type="text" value="IBM"/> Number of Shares: <input type="text" value="1000"/> <input type="button" value="Submit"/></div></div></div> <div><div><div>__i.</div><div>The Loyalty Level changes to Gold with the following screen.</div></div></div> <tr><td></td><td><div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div></td></tr>		<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$151,910.00	Gold																						
	<div><div><div><div><div></div><div>Create a new portfolio</div></div><div><div></div><div>Retrieve selected portfolio</div></div><div><div><div></div><div>Update selected portfolio (add stock)</div></div></div><div><div></div><div>Delete selected portfolio</div></div></div><div><table><tr><td></td><td>Owner</td><td>Total</td><td>Loyalty Level</td></tr><tr><td><div><div></div><div></div></div></td><td>PoT</td><td>\$151,910.00</td><td>Gold</td></tr></table><div><div><input type="button" value="Submit"/></div><div><input type="button" value="Log Out"/></div></div></div></div></div>		Owner	Total	Loyalty Level	<div><div></div><div></div></div>	PoT	\$151,910.00	Gold																								
	Owner	Total	Loyalty Level																														
<div><div></div><div></div></div>	PoT	\$151,910.00	Gold																														

- __j. Open a new tab in the browser and type URL <https://twitter.com/ibmstocktrader> and you should see the message posted at the Twitter site.



- __k. If you do not see the Twitter message, check the logs of the [messaging](#) microservice.

- __l. Run `kubectl -n stocktrader get pods`

```
[root@node01 08ms]# kubectl -n stocktrader get pods
```

NAME	READY	STATUS	AGE
loyalty-level-7b58569b9b-sfg8g	1/1	Running	2h
messaging-559cf6f4cf-h6kzx	1/1	Running	0m
notification-twitter-585b96f845-9nv85	1/1	Running	2h
portfolio-7c568d6cb8-6bkg4	1/1	Running	2h
stockquote-dbf546b67-4ldh7	1/1	Running	2h
trader-5c5ff75c5d-lgnpq	1/1	Running	2h
tradr-84784b4d9f-kpckw	1/1	Running	2h

- __m. Note the messaging pod name and get logs.

- __n. Run `kubectl -n stocktrader logs messaging-559cf6f4cf-h6kzx`

Step	Action
__o.	Change last two suffix verbs, as per your output.
__p.	Select Update selected portfolio (add stock)
	<div><div><div><div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div><div><div></div><div></div><div></div><div></div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></div><div></div><div></div></div> <div><div></div><div></</div></div>

Step	Action																																												
	<div><pre>[db2psc@dev-ibm-db2oltp-dev-0 ~]\$ db2 connect to PSDB</pre><p>Database Connection Information</p><p>Database server = DB2/LINUX8664 11.1.3.3</p><p>SQL authorization ID = DB2PSC</p><p>Local database alias = PSDB</p></div> <p><u>f.</u> Run the following commands: 1. db2 list tables, 2. db2 "select * from stock" and 3. db2 "select * from portfolio"</p> <div><pre>[db2psc@dev-ibm-db2oltp-dev-0 ~]\$ db2 list tables</pre><table><thead><tr><th>Table/View</th><th>Schema</th><th>Type</th><th>Creation time</th></tr></thead><tbody><tr><td>PORTFOLIO</td><td>DB2PSC</td><td>T</td><td>2018-04-17-00.07.51.819260</td></tr><tr><td>STOCK</td><td>DB2PSC</td><td>T</td><td>2018-04-17-00.07.52.795422</td></tr></tbody></table><p>2 record(s) selected.</p><pre>[db2psc@dev-ibm-db2oltp-dev-0 ~]\$ db2 "select * from stock"</pre><table><thead><tr><th>OWNER</th><th>SYMBOL</th><th>SHARES</th><th>PRICE</th><th>TOTAL</th><th>DATEQUOTED</th></tr></thead><tbody><tr><td>PoT</td><td>IBM</td><td>1000</td><td>+1.519100000000000E+002</td><td>+1.519100000000000E+005</td><td>03/27/2018</td></tr><tr><td>PoT</td><td>AAPL</td><td>1000</td><td>+1.683400000000000E+002</td><td>+1.683400000000000E+005</td><td>03/27/2018</td></tr></tbody></table><p>1 record(s) selected.</p><pre>[db2psc@dev-ibm-db2oltp-dev-0 ~]\$ db2 "select * from portfolio"</pre><table><thead><tr><th>OWNER</th><th>TOTAL</th><th>LOYALTY</th></tr></thead><tbody><tr><td>PoT</td><td>+3.202500000000000E+005</td><td>Gold</td></tr></tbody></table><p>1 record(s) selected.</p></div> <p><u>g.</u> Type exit to log out from the container.</p> <div><pre>[db2psc@dev-ibm-db2oltp-dev-0 ~]\$ exit</pre><p>logout</p></div>	Table/View	Schema	Type	Creation time	PORTFOLIO	DB2PSC	T	2018-04-17-00.07.51.819260	STOCK	DB2PSC	T	2018-04-17-00.07.52.795422	OWNER	SYMBOL	SHARES	PRICE	TOTAL	DATEQUOTED	PoT	IBM	1000	+1.519100000000000E+002	+1.519100000000000E+005	03/27/2018	PoT	AAPL	1000	+1.683400000000000E+002	+1.683400000000000E+005	03/27/2018	OWNER	TOTAL	LOYALTY	PoT	+3.202500000000000E+005	Gold								
Table/View	Schema	Type	Creation time																																										
PORTFOLIO	DB2PSC	T	2018-04-17-00.07.51.819260																																										
STOCK	DB2PSC	T	2018-04-17-00.07.52.795422																																										
OWNER	SYMBOL	SHARES	PRICE	TOTAL	DATEQUOTED																																								
PoT	IBM	1000	+1.519100000000000E+002	+1.519100000000000E+005	03/27/2018																																								
PoT	AAPL	1000	+1.683400000000000E+002	+1.683400000000000E+005	03/27/2018																																								
OWNER	TOTAL	LOYALTY																																											
PoT	+3.202500000000000E+005	Gold																																											
8	<p><u>Explore Redis Records</u></p> <p><u>a.</u> Run kubectl -n default get pods</p> <div><pre>[root@node01 08ms]# kubectl -n default get pods</pre><table><thead><tr><th>NAME</th><th>READY</th><th>STATUS</th><th>RESTARTS</th></tr></thead><tbody><tr><td>dev-ibm-db2oltp-dev-0</td><td>1/1</td><td>Running</td><td>2</td></tr><tr><td>4h</td><td></td><td></td><td></td></tr><tr><td>helm-local-repo-fj9cj</td><td>1/1</td><td>Running</td><td>4</td></tr><tr><td>8h</td><td></td><td></td><td></td></tr><tr><td>qdev-ibm-mq-0</td><td>1/1</td><td>Running</td><td>0</td></tr><tr><td>1h</td><td></td><td></td><td></td></tr><tr><td>rdev-ibm-redis-ha-dev-sentinel-68db4dc96-9lgkr</td><td>1/1</td><td>Running</td><td>0</td></tr><tr><td>57m</td><td></td><td></td><td></td></tr><tr><td>rdev-ibm-redis-ha-dev-sentinel-68db4dc96-g4zvd</td><td>1/1</td><td>Running</td><td>0</td></tr><tr><td>57m</td><td></td><td></td><td></td></tr></tbody></table></div>	NAME	READY	STATUS	RESTARTS	dev-ibm-db2oltp-dev-0	1/1	Running	2	4h				helm-local-repo-fj9cj	1/1	Running	4	8h				qdev-ibm-mq-0	1/1	Running	0	1h				rdev-ibm-redis-ha-dev-sentinel-68db4dc96-9lgkr	1/1	Running	0	57m				rdev-ibm-redis-ha-dev-sentinel-68db4dc96-g4zvd	1/1	Running	0	57m			
NAME	READY	STATUS	RESTARTS																																										
dev-ibm-db2oltp-dev-0	1/1	Running	2																																										
4h																																													
helm-local-repo-fj9cj	1/1	Running	4																																										
8h																																													
qdev-ibm-mq-0	1/1	Running	0																																										
1h																																													
rdev-ibm-redis-ha-dev-sentinel-68db4dc96-9lgkr	1/1	Running	0																																										
57m																																													
rdev-ibm-redis-ha-dev-sentinel-68db4dc96-g4zvd	1/1	Running	0																																										
57m																																													

Step	Action
	<pre>rdev-ibm-redis-ha-dev-sentinel-68db4dc96-qsgz6 1/1 Running 0 57m rdev-ibm-redis-ha-dev-server-85d8f665d-2vpfk 1/1 Running 0 57m rdev-ibm-redis-ha-dev-server-85d8f665d-77t55 1/1 Running 0 57m rdev-ibm-redis-ha-dev-server-85d8f665d-q85kw 1/1 Running 0 57m</pre>
__b.	We have three copies of the redis server running. How do we know which one is the master?
__c.	Run kubectl -n default get pods -l redis-role=master
	<pre>[root@node01 08ms]# kubectl -n default get pods -l redis-role=master NAME READY STATUS RESTARTS AGE rdev-ibm-redis-ha-dev-server-85d8f665d-77t55 1/1 Running 0 59m</pre>
__d.	Highlight the redis-ha-dev-server and select the full name to copy.
__e.	Run kubectl -n default exec -it rdev-ibm-redis-ha-dev-server-85d8f665d-77t55 bash
	<pre># kubectl -n default exec -it rdev-ibm-redis-ha-dev-server-85d8f665d-77t55 bash bash-4.4#</pre>
__f.	Replace the suffix in the above-mentioned name as per the output in your command line.
__g.	Run redis-cli ping and it should return the response as pong .
	<pre>bash-4.4# redis-cli ping pong 127.0.0.1:6379></pre>
__h.	Run redis-cli to get the command line prompt.
__i.	Type info
__j.	Scroll through the Redis server statistics.
__k.	Type keys *
	<pre>127.0.0.1:6379> keys * 1) "AAPL" 2) "IBM" 127.0.0.1:6379></pre>

Step	Action
	<p data-bbox="245 304 1149 338">__l. Note IBM and AAPL stock quotes cached in the Redis server.</p> <p data-bbox="245 373 1068 407">__m. Type get IBM and get AAPL to see the cached values.</p> <pre data-bbox="342 422 1487 562">127.0.0.1:6379> get IBM "{\"symbol\": \"IBM\", \"date\": \"2018-03-27\", \"price\": 151.91}" 127.0.0.1:6379> get AAPL "{\"symbol\": \"AAPL\", \"date\": \"2018-03-27\", \"price\": 168.34}"</pre> <p data-bbox="245 598 1292 632">__n. Type exit to quit redis-cli and exit again to quit the Redis container.</p> <pre data-bbox="342 646 704 787">127.0.0.1:6379> exit bash-4.4# exit exit [root@node01 08ms]#</pre> <p data-bbox="245 823 1448 890">__o. Note that we run only one Redis server and one sentinel (replicated) server. In actual environment, we would run minimum 3 Redis server and 3 sentinel servers.</p>