Exercise 4.3: More Complex Deployment

We will now deploy a more complex demo application to test the cluster. When completed it will be a sock shopping site. The short URL is shown below for:
https://raw.githubusercontent.com/microservices-demo/microservices-demo/master/deploy/kubernetes/complete-demo.yaml

1. Begin by downloading the pre-made YAML file from github.

```
student@lfs458-node-1a0a:~$ wget https://tinyurl.com/y8bn2awp -O complete-demo.yaml

Resolving tinyurl.com (tinyurl.com)... 104.20.218.42, 104.20.219.42,
Connecting to tinyurl.com (tinyurl.com)|104.20.218.42|:443... connected.
HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://raw.githubusercontent.com/microservices-demo/microservices-...
--2017-11-02 16:54:27--  https://raw.githubusercontent.com/microservices-dem...
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 151.101.5...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|151.101....
HTTP request sent, awaiting response... 200 OK
<output_omitted>
```

2. Find the expected namespaces inside the file. It should be sock-shop. Also note the various settings. This file will deploy several containers which work together, providing a shopping website. As we work with other parameters you could revisit this file to see potential settings.

```
student@lfs458-node-1a0a:~$ less complete-demo.yaml
```

complete-demo.yaml

```
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
  name: carts-db
  labels:
    name: carts-db
  namespace: sock-shop
spec:
  replicas: 1
<output_omitted>
```

3. Create the namespace and verify it was made.

```
student@lfs458-node-1a0a:~$ kubectl create namespace sock-shop

namespace/sock-shop created


student@lfs458-node-1a0a:~$ kubectl get namespace

NAME              STATUS    AGE
default           Active    4h
kube-public       Active    4h
kube-system       Active    4h
low-usage-limit   Active    15m
sock-shop         Active    5s
```

4. View the images the new application will deploy.

```
student@lfs458-node-1a0a:~$ grep image complete-demo.yaml
        image: mongo
        image: weaveworksdemos/carts:0.4.8
        image: weaveworksdemos/catalogue-db:0.3.0
        image: weaveworksdemos/catalogue:0.3.5
        image: weaveworksdemos/front-end:0.3.12
        image: mongo
<output_omitted>
```

5. Create the new shopping website using the YAML file. Use the namespace you recently created. Note that the deployments match the images we saw in the file.

```
student@lfs458-node-1a0a:~$ kubectl apply -n sock-shop -f complete-demo.yaml
deployment "carts-db" created
service "carts-db" created
deployment "carts" created
service "carts" created
<output_omitted>
```

6. Using the proper namespace will be important. This can be set on a per-command basis or as a shell parameter. Note the first command shows no pods. We must remember to pass the proper namespace. Some containers may not have fully downloaded or deployed by the time you run the command.

```
student@lfs458-node-1a0a:~$ kubectl get pods
No resources found.

student@lfs458-node-1a0a:~$ kubectl -n sock-shop get pods
NAME                            READY     STATUS              RESTARTS   AGE
carts-511261774-c4jwv           1/1       Running             0          71s
carts-db-549516398-tw9zs        1/1       Running             0          71s
catalogue-4293036822-sp5kt      1/1       Running             0          71s
catalogue-db-1846494424-qzhvk   1/1       Running             0          71s
front-end-2337481689-6s65c      1/1       Running             0          71s
orders-208161811-1gc6k          1/1       Running             0          71s
orders-db-2069777334-4sp01      1/1       Running             0          71s
payment-3050936124-2cn2l        1/1       Running             0          71s
queue-master-2067646375-vzq77   1/1       Running             0          71s
rabbitmq-241640118-vk3m9        0/1       ContainerCreating   0          71s
shipping-3132821717-lm7kn       0/1       ContainerCreating   0          71s
user-1574605338-24xrb           0/1       ContainerCreating   0          71s
user-db-2947298815-lx9kp        1/1       Running             0          71s
```

7. Verify the shopping cart is exposing a web page. Use the public IP address of your AWS node (not the one derived from the prompt) to view the page. Note the external IP is not yet configured. Find the `NodePort` service. First try port 80 then try port 30001 as shown under the `PORTS` column.

```
student@lfs458-node-1a0a:~$ kubectl get svc -n sock-shop
NAME           TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)        AGE
carts          ClusterIP   10.100.154.148   <none>        80/TCP         95s
carts-db       ClusterIP   10.111.120.73    <none>        27017/TCP      95s
catalogue      ClusterIP   10.100.8.203     <none>        80/TCP         95s
catalogue-db   ClusterIP   10.111.94.74     <none>        3306/TCP       95s
front-end      NodePort    10.98.2.137      <none>        80:30001/TCP   95s
orders         ClusterIP   10.110.7.215     <none>        80/TCP         95s
orders-db      ClusterIP   10.106.19.121    <none>        27017/TCP      95s
payment        ClusterIP   10.111.28.218    <none>        80/TCP         95s
queue-master   ClusterIP   10.102.181.253   <none>        80/TCP         95s
rabbitmq       ClusterIP   10.107.134.121   <none>        5672/TCP       95s
shipping       ClusterIP   10.99.99.127     <none>        80/TCP         95s
user           ClusterIP   10.105.126.10    <none>        80/TCP         95s
user-db        ClusterIP   10.99.123.228    <none>        27017/TCP      95s
```

8. Check to see which node is running the containers. Note that the webserver is answering on a node which is not hosting the all the containers. First we check the master, then the second node. The containers should have to do with **kube proxy** services and **calico**. The following is the **sudo docker ps** on both nodes. The output is truncated, you will see several lines of output per container.

```
student@lfs458-node-1a0a:~$ sudo docker ps

CONTAINER ID        IMAGE
d6b7353e5dc5        weaveworksdemos/user@sha256:2ffccc332963c89e035fea52201012208bf62df4...
6c18f030f15b        weaveworksdemosshipping@sha256:983305c948fded487f4a4acdeab5f898e89d5...
baaa8d67ebef        weaveworksdemos/queue-master@sha256:6292d3095f4c7aeed8d863527f8ef6d7...
<output_omitted>


student@lfs458-worker:~$ sudo docker ps

CONTAINER ID        IMAGE
9452559caa0d        weaveworksdemospayment@sha256:5ab1c9877480a018d4dda10d6dfa382776...
993017c7b476        weaveworksdemos/user-db@sha256:b43f0f8a76e0c908805fcec74d1ad7f4a...
1356b0548ee8        weaveworksdemos/orders@sha256:b622e40e83433baf6374f15e076b53893f...
<output_omitted>
```

9. Now we will shut down the shopping application. This can be done a few different ways. Begin by getting a listing of resources in all namespaces. There should be about 14 deployments.

```
student@lfs458-node-1a0a:~$ kubectl get deployment --all-namespaces

NAMESPACE       NAME          DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
kube-system     calico-typha  0         0         0            0           4h
kube-system     coredns       2         2         2            2           4h
low-usage-limit limited-hog   1         1         1            1           33m
sock-shop       carts         1         1         1            1           6m44s
sock-shop       carts-db      1         1         1            1           6m44s
sock-shop       catalogue     1         1         1            1           6m44s
<output_omitted>
```

10. Use the terminal on the second node to get a count of the current docker containers. It should be something like 30, plus a line for status counted by **wc**. The main system should have something like 26 running, plus a line of status.

```
student@lfs458-node-1a0a:~$ sudo docker ps | wc -l

26


student@lfs458-worker:~$ sudo docker ps | wc -l

30
```

11. In order to complete maintainence we may need to move containers from a node and prevent new ones from deploying. One way to do this is to **drain**, or `cordon`, the node. Currently this will not affect `DaemonSets`, an object we will discuss in greater detail in the future. Begin by getting a list of nodes. Your node names will be different.

```
student@lfs458-node-1a0a:~$ kubectl get nodes

NAME                 STATUS    ROLES    AGE       VERSION
lfs458-worker        Ready     <none>   4h        v1.13.1
lfs458-node-1a0a     Ready     master   4h        v1.13.1
```

12. Modifying your second, worker node, update the node to **drain** the pods. Some resources may not drain, expect an error which we will work with next. Note the error includes `aborting command` which indicates the drain did not take place. Were you to check it would have the same number of containers running, but will show a new taint preventing the scheduler from assigning new pods.

```
student@lfs458-node-1a0a:~$ kubectl drain lfs458-worker
```

```
node/lfs458-worker cordoned
error: unable to drain node "lfs458-worker", aborting command...

There are pending nodes to be drained:
 lfs458-worker
error: DaemonSet-managed pods (use --ignore-daemonsets to ignore):
calico-node-vndn7, kube-proxy-rjjls
```

```
student@lfs458-node-1a0a:~$ kubectl describe node |grep -i taint
```

```
Taints:             <none>
Taints:             node.kubernetes.io/unschedulable:NoSchedule
```

13. As the error output suggests we can use the **–ignore-daemonsets** options to ignore containers which are not intended to move. We will find a new error when we use this command, near the end of the output. The node will continue to have the same number of pods and containers running.

```
student@lfs458-node-1a0a:~$ kubectl drain lfs458-worker --ignore-daemonsets
```

```
node/worker cordoned
error: unable to drain node "lfs458-worker", aborting command...

There are pending nodes to be drained:
 lfs458-worker
error: pods with local storage (use --delete-local-data to override):
carts-55f7f5c679-ffkq2, carts-db-5c55874946-w728d, orders-7b69bf5686-vtkcn
```

14. Run the command again. This time the output should both indicate the node has already been cordoned, then show the eviction of several pods. Not all pods will be gone as `daemonsets` will remain. Note the command is shown on two lines. You can omit the backslash and type the command on a single line.

```
student@lfs458-node-1a0a:~$ kubectl drain lfs458-worker \
     --ignore-daemonsets --delete-local-data
```

```
node/lfs458-worker already cordoned
WARNING: Ignoring DaemonSet-managed pods: calico-node-vndn7, kube-proxy-rjjls;
Deleting pods with local storage: carts-55f7f5c679-ppv7p,
carts-db-5c55874946-h42v2, orders-7b69bf5686-t82lz, orders-db-7bc46bdb98-x5zrl
pod/carts-db-5c55874946-h42v2 evicted
pod/orders-db-7bc46bdb98-x5zrl evicted
pod/catalogue-db-66ff5bbbf5-2wmx4 evicted
pod/catalogue-5764fdf6d-8gk96 evicted
pod/orders-7b69bf5686-t82lz evicted
pod/front-end-f99dbcb9c-92q4p evicted
pod/carts-55f7f5c679-ppv7p evicted
```

15. Were you to look on your second, worker node, you would see there should be fewer pods and containers than before. These pods can only be evicted via a special taint which we will discuss in the scheduling chapter.

```
student@lfs458-worker:~$ sudo docker ps | wc -l
```

```
6
```

16. Update the node taint such that the scheduler will use the node again. Verify that no nodes have moved over to the worker node as the scheduler only checks when a pod is deployed.

```
student@lfs458-node-1a0a:~$ kubectl uncordon lfs458-worker
```

```
node/lfs458-worker uncordoned
```

```
student@lfs458-node-1a0a:~$ kubectl describe node |grep -i taint
```

```
Taints:             <none>
Taints:             <none>
```

```
student@lfs458-worker:~$ sudo docker ps | wc -l
```

```
6
```

17. As we clean up our sock shop let us see some differences between pods and deployments. Start with a list of the pods that are running in the `sock-shop` namespace.

```
student@lfs458-node-1a0a:~$ kubectl -n sock-shop get pod
NAME                             READY     STATUS    RESTARTS   AGE
carts-db-549516398-tw9zs         1/1       Running   0          6h
catalogue-4293036822-sp5kt       1/1       Running   0          6h
<output_omitted>
```

18. Delete a few resources using the pod name.

```
student@lfs458-node-1a0a:~$ kubectl -n sock-shop delete pod \
   catalogue-4293036822-sp5kt catalogue-db-1846494424-qzhvk \
   front-end-2337481689-6s65c orders-208161811-1gc6k \
   orders-db-2069777334-4sp01
pod "catalogue-4293036822-sp5kt" deleted
pod "catalogue-db-1846494424-qzhvk" deleted
<output_omitted>
```

19. Check the status of the pods. There should be some pods running for only a few seconds. These will have the same name-stub as the Pods you recently deleted. The `Deployment` controller noticed expected number of Pods was not proper, so created new Pods until the current state matches the Pod manifest.

```
student@lfs458-node-1a0a:~$ kubectl -n sock-shop get pod
NAME                             READY     STATUS        RESTARTS   AGE
catalogue-4293036822-mtz8m       1/1       Running       0          22s
catalogue-db-1846494424-16n2p    1/1       Running       0          22s
front-end-2337481689-6s65c       1/1       Terminating   0          6h
front-end-2337481689-80gwt       1/1       Running       0          22s
```

20. Delete some of the resources via deployments.

```
student@lfs458-node-1a0a:~$ kubectl -n sock-shop delete deployment \
       catalogue catalogue-db front-end orders
deployment.extensions "catalogue" deleted
deployment.extensions "catalogue-db" deleted
deployment.extensions "front-end" deleted
deployment.extensions "orders" deleted
```

21. Check and both the pods and deployments you removed have not been recreated.

```
student@lfs458-node-1a0a:~$ kubectl -n sock-shop get pods | grep catalogue

student@lfs458-node-1a0a:~$ kubectl -n sock-shop get deployment
NAME           DESIRED   CURRENT   UP-TO-DATE   AVAILABLE   AGE
carts          1         1         1            1           71m
carts-db       1         1         1            1           71m
orders-db      1         1         1            1           71m
payment        1         1         1            1           71m
queue-master   1         1         1            1           71m
rabbitmq       1         1         1            1           71m
shipping       1         1         1            1           71m
user           1         1         1            1           71m
user-db        1         1         1            1           71m
```

22. Delete the rest of the deployments. When no resources are found, examine the output of the `docker ps` command. None of the `sock-shop` containers should be found. Use the same file we created with to delete all of the objects made. You will get some errors because we deleted a few deployments by hand.

```
student@lfs458-node-1a0a:~$ kubectl delete -f complete-demo.yaml
<output_omitted>
```