Exercise A.3: Practicing Skills

This exercise is to help you practice your skills. It does not cover all the items listed in the domain review guide. You should develop your own steps to build a full list of skill tests and steps.

Also note that all the detailed steps are not included. You should be able to complete these steps without being told what to type.

In a work or exam environment you may not be told exactly what to do or how to do it. The following steps are meant to get you used to thinking about solutions when the exact need isn't clear.

1. Find and use the `review1.yaml` file included in the course tarball. Use the **find** output and copy the YAML file to your home directory. Use **kubectl create** to create the object. Determine if the pod is running. Fix any errors you may encounter. The use of **kubectl describe** may be helpful.

   ```
   student@cp:~$ find ~ -name review1.yaml

   student@cp:~$ cp <copy-paste-from-above>  .

   student@cp:~$ kubectl create -f review1.yaml
   ```

2. After you get the pod running remove any pods or services you may have created as part of the review before moving on to the next section. For example:

   ```
   student@cp:~$ kubectl delete -f review1.yaml
   ```

3. Use the `review2.yaml` file to create a non-working deployment. Fix the deployment such that both containers are running and in a `READY` state. The web server listens on port 80, and the proxy listens on port 8080.

4. View the default page of the web server. When successful verify the `GET` activity logs in the container log. The message should look something like the following. Your time and IP may be different.

   ```
   192.168.124.0 - - [3/Dec/2024:03:30:31 +0000] "GET / HTTP/1.1" 200 612 "-"
   "curl/7.58.0" "-"
   ```

5. Find and use the `review4.yaml` file to create a pod, and verify it's running

6. Edit the pod such that it only runs on your worker node using the `nodeSelector` label.

7. Determine the CPU and memory resource requirements of `design-pod1`.

8. Edit the pod resource requirements such that the CPU limit is exactly twice the amount requested by the container. (Hint: subtract .22)

9. Increase the memory resource limit of the pod until the pod shows a `Running` status. This may require multiple edits and attempts. Determine the minimum amount necessary for the `Running` status to persist at least a minute.

10. Use the `review5.yaml` file to create several pods with various labels.

11. Using **only** the –selector value `tux` to delete only those pods. This should be half of the pods. Hint, you will need to view pod settings to determine the key value as well.

12. Create a new cronjob which runs `busybox` and the `sleep 30` command. Have the cronjob run every three minutes. View the job status to check your work. Change the settings so the pod runs 10 minutes from the current time, every week. For example, if the current time was 2:14PM, I would configure the job to run at 2:24PM, every Monday.

13. Delete any objects created during this review. You may want to delete all but the `cronjob` if you'd like to see if it runs in 10 minutes. Then delete that object as well.

14. Create a new secret called `specialofday` using the key `entree` and the value `meatloaf`.
15. Create a new deployment called `foodie` running the `nginx` image.
16. Add the `specialofday` secret to pod mounted as a volume under the `/food/` directory.
17. Execute a bash shell inside a `foodie` pod and verify the secret has been properly mounted.
18. Update the deployment to use the `nginx:1.12.1-alpine` image and verify the new image is in use.
19. Roll back the deployment and verify the typical, current stable version of nginx is in use again.
20. Create a new 200M NFS volume called `reviewvol` using the NFS server configured earlier in the lab.
21. Create a new PVC called `reviewpvc` which will uses the `reviewvol` volume.
22. Edit the deployment to use the PVC and mount the volume under `/newvol`
23. Execute a bash shell into the nginx container and verify the volume has been mounted.
24. Delete any resources created during this review.
25. Create a new deployment which uses the `nginx` image.
26. Create a new `LoadBalancer` service to expose the newly created deployment. Test that it works.
27. Create a new NetworkPolicy called `netblock` which blocks all traffic to pods in this deployment only. Test that all traffic is blocked to deployment.
28. Create a pod running nginx and ensure traffic can reach that deployment.
29. Update the `netblock` policy to allow traffic to the pod on port 80 only. Test that you can now access the default nginx web page.
30. Find and use the `review6.yaml` file to create a pod.

```
student@cp:~$ kubectl create -f review6.yaml
```

31. View the status of the pod.
32. Use the following commands to figure out why the pod has issues.

```
student@cp:~$ kubectl get pod securityreview

student@cp:~$ kubectl describe pod securityreview

student@cp:~$ kubectl logs securityreview
```

33. After finding the errors, log into the container and find the proper id of the nginx user.
34. Edit the pod such that the `securityContext` is in place and allows the web server to read the proper configuration files.
35. Create a new `serviceAccount` called `securityaccount`.
36. Create a ClusterRole named `secrole` which only allows create, delete, and list of pods in all apiGroups.
37. Bind the new clusterRole to the new serviceAccount.
38. Locate the token of the `securityaccount`. Create a file called `/tmp/securitytoken`. Put only the value of `token:` is equal to, a long string that may start with eyJh and be several lines long. Careful that only that string exists in the file.
39. Remove any resources you have added during this review
40. Create a new pod called `webone`, running the `nginx` service. Expose port 80.
41. Create a new service named `webone-svc`. The service should be accessible from outside the cluster.
42. Update both the pod and the service with selectors so that traffic for to the service IP shows the web server content.

43. Change the type of the service such that it is only accessible from within the cluster. Test that exterior access no longer works, but access from within the node works.
44. Deploy another pod, called `webtwo`, this time running the `wlniao/website` image. Create another service, called `webtwo-svc` such that only requests from within the cluster work. Note the default page for each server is distinct.
45. Test DNS names and verify CoreDNS is properly functioning.
46. Install and configure an ingress controller such that requests for webone.com see the `nginx` default page, and requests for webtwo.org see the `wlniao/website` default page. It does not matter which ingress controller you use.
47. Remove any resources created in this review.
48. Install a new cluster using an recent, previous version of Kubernetes. Backup etcd, then properly upgrade the entire cluster.
49. Create a pod running `busybox` without the scheduler being consulted.
50. Continue to create objects, integrate them with other objects and troubleshoot until each domain item has been covered.