Exercise 3.2: Configure A Local Repo

While we could create an account and upload our application to https://hub.docker.com or https://artifacthub.io/, thus sharing it with the world, we will instead create a local repository and make it available to the nodes of our cluster.

1. Create a simple registry using the `easyregistry.yaml` file included in the course tarball. Use the path returned by **find**, which may be different than the one found in the output below.

```
student@cp:~/app1$ find $HOME -name easyregistry.yaml
```

```
<some_long_path>/easyregistry.yaml
```

```
student@cp:~/app1$ kubectl create -f <path_from_output_above>
```

```
service/nginx created
service/registry created
deployment.apps/nginx created
persistentvolumeclaim/nginx-claim0 created
deployment.apps/registry created
persistentvolumeclaim/registry-claim0 created
persistentvolume/vol1 created
persistentvolume/vol2 created
```

2. Take note of the `ClusterIP` for the new registry service. In the example below it is `10.97.40.62`

```
student@cp:~/app1$ kubectl get svc | grep registry
```

```
registry      ClusterIP   10.97.40.62    <none>      5000/TCP,8080/TCP   5m35s
```

3. Verify the repo is working. Please note that if the connection hangs it may be due to a firewall issue. If running your nodes using GCE ensure your instances are using VPC setup and all ports are allowed. If using AWS also make sure all ports are being allowed.

   Edit the IP address to that of your `localrepo` service found in the previous command, and the listed port of `5000`

```
student@cp:~/app1$ curl 10.97.40.62:5000/v2/_catalog
```

```
{"repositories":[]}
```

4. Configure **podman** to work with non-TLS repos. The way to do this depends on the container engine in use. In our setup we will edit the `/etc/containerd/config.toml` file and add the registry we just created.

   We have included a script in the tar ball **local-repo-setup.sh** to setup local repository.

```
student@cp:~/app1$ find $HOME -name local-repo-setup.sh
student@cp:~/app1$ cp LFD459/SOLUTIONS/s_03/local-repo-setup.sh $HOME
student@cp:~/app1$ chmod +x $HOME/local-repo-setup.sh
student@cp:~/app1$ . local-repo-setup.sh       # dot <space> scriptname to set correct variables
```

```
. local-repo-setup.sh
Configuring local repo, Please standby
[[registry]]
location = "10.97.40.62:5000"
insecure = true
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."*"]
```

```
endpoint = ["http://10.97.40.62:5000"]

Local Repo configured, follow the next steps
```

5. Download and tag a typical image from hub.docker.com. Tag the image using the IP and port of the registry, via the $repo variable.

```
student@cp:~app1/$ sudo podman pull docker.io/library/alpine
```

```
Resolved "alpine" as an alias (/etc/containers/registries.conf.d/000-shortnames.conf)
Trying to pull docker.io/library/alpine:latest...
Getting image source signatures
Copying blob 540db60ca938 done
Copying config 6dbb9cc540 done
Writing manifest to image destination
Storing signatures
6dbb9cc54074106d46d4ccb330f2a40a682d49dda5f4844962b7dce9fe44aaec
```

```
student@cp:~/app1$ sudo podman tag alpine $repo/tagtest
```

6. Push the newly tagged image to your local registry. If you receive an error about an HTTP request to an HTTPS client check that you edited the /etc/containers/registry.conf file correctly and restarted the service.

```
student@cp:~/app1$ sudo podman push $repo/tagtest
```

```
Getting image source signatures
Copying blob b2d5eeeaba3a done
Copying config 6dbb9cc540 done
Writing manifest to image destination
Storing signatures
```

7. We will test to make sure we can also pull images from our local repository. Begin by removing the local cached images.

```
student@cp:~/app1$ sudo podman image rm alpine
```

```
Untagged: docker.io/library/alpine:latest
```

```
student@cp:~/app1$ sudo podman image rm $repo/tagtest
```

```
Untagged: 10.97.40.62:5000/tagtest:latest
Deleted: 6dbb9cc54074106d46d4ccb330f2a40a682d49dda5f4844962b7dce9fe44aaec
```

8. Pull the image from the local registry. It should report the download of a newer image.

```
student@cp:~/app1$ sudo podman pull $repo/tagtest
```

```
Trying to pull 10.97.40.62:5000/tagtest:latest...
Getting image source signatures
Copying blob 74782b667c7d done
Copying config 6dbb9cc540 done
Writing manifest to image destination
Storing signatures
6dbb9cc54074106d46d4ccb330f2a40a682d49dda5f4844962b7dce9fe44aaec
```

9. Configure the worker (second) node to use the registry running on the cp server. Connect to the worker node.

```
student@worker:~$ find $HOME -name local-repo-setup.sh
student@worker:~$ cp LFD459/SOLUTIONS/s_03/local-repo-setup.sh $HOME
```

```
student@worker:~$ chmod +x $HOME/local-repo-setup.sh
student@worker:~$ . local-repo-setup.sh       # dot <space> scriptname to set correct variables
student@worker:~$ sudo podman pull $repo/tagtest # if needed export repo=10.97.40.62:5000
```

```
. local-repo-setup.sh
Configuring local repo, Please standby
[[registry]]
location = "10.97.40.62:5000"
insecure = true
[plugins."io.containerd.grpc.v1.cri".registry.mirrors."*"]
endpoint = ["http://10.97.40.62:5000"]

Local Repo configured, follow the next steps


Trying to pull 10.97.40.62:5000/tagtest:latest...
Getting image source signatures
Getting image source signatures
Copying blob 74782b667c7d done
Copying config 6dbb9cc540 done
Writing manifest to image destination
Storing signatures
a313e647ae05302fc54b57065c250ff58d2f580fab7b7625949651b141d9fca7
```

10. Now that we know **podman** on all nodes can use the repository we need to make sure Kubernetes knows about the new repository and settings as well. The simplest way is to reboot every node. Log back in after the connection closes.

```
student@cp:~$ sudo reboot
```

```
student@worker:~$ sudo reboot
```

11. Test that the repo works after the reboot, as good admins we want to know our configuration is persistent. Be aware it can take a minute or two after reboot for the kube-apiserver to fully start. If the `$repo` variable isn't set check the source statement in `.bashrc`.

```
student@cp:~$ curl $repo/v2/_catalog
```

```
{"repositories":[]}
```

12. Use **podman tag** to assign the `simpleapp` image and then push it to the local registry. The image and dependent images should be pushed to the local repository.

```
student@cp:~/app1$ sudo podman tag simpleapp $repo/simpleapp
```

```
student@cp:~/app1$ sudo podman push $repo/simpleapp
```

```
Getting image source signatures
Copying blob 47458fb45d99 done
Copying blob a3c1026c6bcc done
Copying blob f1d420c2af1a done
Copying blob 461719022993 done
Copying blob d35c5bda4793 done
Copying blob 46829331b1e4 done
Copying blob ceee8816bb96 done
Copying blob da7b0a80a4f2 done
Copying blob e571d2d3c73c done
Copying blob 5c2db76bc949 done
Copying config a313e647ae done
Writing manifest to image destination
Storing signatures
```

13. Test that the image can be found in the repository, from both the cp and the worker

```
student@cp:~$ curl $repo/v2/_catalog
```

```
{"repositories":["simpleapp"]}
```

14. Return to the cp node and deploy the `simpleapp` in Kubernetes with several replicas. We will name the deployment `try1`. Scale to have six replicas. Increase the replica count until pods are deployed on both cp and worker.

```
student@cp:~$ kubectl create deployment try1 --image=$repo/simpleapp
```

```
deployment.apps/try1 created
```

```
student@cp:~$  kubectl scale deployment try1 --replicas=6
```

```
deployment.apps/try1 scaled
```

```
student@cp:~$  kubectl get pod -o wide
```

```
NAME                      READY   STATUS    RESTARTS   AGE     IP                NODE  ....
basicpod                  1/1     Running   0          20m     192.168.95.15     worker ....
registry-ddf6bcb7c-b8wcr  1/1     Running   0          16m     192.168.95.16     worker ....
try1-55f675ddd-28vgs      1/1     Running   0          17s     192.168.95.30     worker ....
try1-55f675ddd-2nrsj      1/1     Running   0          17s     192.168.95.26     worker ....
try1-55f675ddd-4vzt7      1/1     Running   0          17s     192.168.219.100   cp ....
<output_omitted>
```

15. On the `second node` use **sudo crictl ps** to verify containers of `simpleapp` are running. Even though **podman** has the options, cri-o is running the containers on our behalf. The scheduler will usually balance pod count across nodes. As the cp already has several pods running the new pods may be on the worker, so the number of pods returned will vary.

    You may need to configure **crictl** first to avoid error messages. You can omit the backslash and type the command on one line.

```
student@worker:~$ sudo crictl config \
  --set runtime-endpoint=unix:///run/containerd/containerd.sock \
  --set image-endpoint=unix:///run/containerd/containerd.sock
```

```
student@worker:~$ sudo crictl ps | grep simple
```

```
855750df82cd4       10.97.177.111:5000/
simpleapp@sha256:f7fc297b3c20c47c506896ee13f4d7f13b4c4c0ee0fafd2ab88c9809b2a1aed6   15 minutes ago
Running             simpleapp                 0                   6e31ba7491117
237af1260265a       10.97.177.111:5000/
simpleapp@sha256:f7fc297b3c20c47c506896ee13f4d7f13b4c4c0ee0fafd2ab88c9809b2a1aed6   15 minutes ago
Running             simpleapp                 0                   bc0d63564cf49
d0ae2910cf8b8       10.97.177.111:5000/
simpleapp@sha256:f7fc297b3c20c47c506896ee13f4d7f13b4c4c0ee0fafd2ab88c9809b2a1aed6   15 minutes ago
Running             simpleapp                 0                   b9bb90e04bec1
046a0c439fa43       10.97.177.111:5000/
simpleapp@sha256:f7fc297b3c20c47c506896ee13f4d7f13b4c4c0ee0fafd2ab88c9809b2a1aed6   15 minutes ago
Running             simpleapp                 0                   fe464425e41b1
```

16. Return to the `cp node`. Save the `try1` deployment as YAML.

```
student@cp:~/app1$ cd $HOME/app1/
student@cp:~/app1$ kubectl get deployment try1 -o yaml > simpleapp.yaml
```

17. Delete and recreate the `try1` deployment using the YAML file. Verify the deployment is running with the expected six replicas.

```
student@cp:~$ kubectl delete deployment try1
```

```
deployment.apps "try1" deleted
```

```
student@cp:~/app1$ kubectl create -f simpleapp.yaml
```

```
deployment.apps/try1 created
```

```
student@cp:~/app1$ kubectl get deployment
```

```
NAME       READY   UP-TO-DATE   AVAILABLE   AGE
nginx      1/1     1            1           15m
registry   1/1     1            1           15m
try1       6/6     6            6           5s
```