Exercise 8.2: OPTIONAL LAB: Conformance Testing

The **cncf.io** group is in the process of formalizing what is considered to be a conforming Kubernetes cluster. While that project matures there is an existing tool provided by **Heptio** which can be useful. We will need to make sure a newer version of **Golang** is installed for it to work. You can download the code from github and look around with git or with go, depending on which tool you are most familiar. **Things change quickly these steps may not work....today**

1. Download a compiled binary. A shorter URL is shown first, then the longer, just in case the link changes and you need to navigate. They should download the same file.

```
student@ckad-1:~$ curl -sLO https://tinyurl.com/yyu5bs28

student@ckad-1:~$ mv yyu5bs28 sonobuoy.tar.gz

student@ckad-1:~$ tar -xvf sonobuoy.tar.gz
LICENSE
sonobuoy


student@ckad-1:~$ curl -sLO \
https://github.com/heptio/sonobuoy/releases/download/v0.15.4/sonobuoy_0.15.4_linux_amd64.tar.gz
```

2. Run the test. We will not use the `--wait` option, which will capture the screen until the test finishes. This could take a while to finish. You should get some output indicating testing objects being created.

```
student@ckad-1:~$ sudo mv sonobuoy /usr/local/bin/

student@ckad-1:~$ sonobuoy run
WARN[0000] The maximum supported Kubernetes version is 1.15.99, but
the server version is v1.16.1. Sonobuoy will continue but unexpected results may occur.
INFO[0000] created object                    name=sonobuoy namespace= resource=namespaces
INFO[0000] created object                    name=sonobuoy-serviceaccount namespace=sonobuoy ....
INFO[0000] created object                    name=sonobuoy-serviceaccount-sonobuoy namespace=....
INFO[0000] created object                    name=sonobuoy-serviceaccount namespace= resource....
INFO[0000] created object                    name=sonobuoy-config-cm namespace=sonobuoy resou....
INFO[0000] created object                    name=sonobuoy-plugins-cm namespace=sonobuoy reso....
INFO[0000] created object                    name=sonobuoy namespace=sonobuoy resource=pods
INFO[0000] created object                    name=sonobuoy-master namespace=sonobuoy resource....
```

3. View the results inside the `sonobuoy` pod.

```
student@ckad-1:~$ kubectl get pods --all-namespaces
<output_omitted>
sonobuoy      sonobuoy                                                1/1
     Running   0          90s
sonobuoy      sonobuoy-e2e-job-b3bcb52b4fd54367                       2/2
     Running   0          85s
sonobuoy      sonobuoy-systemd-logs-daemon-set-f7ca2bb9a7174908-h47kb  2/2     Running   0          85s
sonobuoy      sonobuoy-systemd-logs-daemon-set-f7ca2bb9a7174908-s22d6  2/2     Running   0          85s

student@ckad-1:~$ kubectl -n sonobuoy exec -it sonobuoy -- /bin/bash
```

On Container

4. View the files inside the container.

```
root@sonobuoy:/# ls
bin   home   mnt        root                        sbin      tmp
boot  lib    opt        run                         sonobuoy  usr
dev   lib64  plugins.d  run_master.sh               srv       var
etc   media  proc       run_single_node_worker.sh   sys
```

5. View the `run_master.sh` script. Note that it mentions both the **sonobuoy** command and where to find the results.

```
root@sonobuoy:/# cat run_master.sh
#!/bin/bash
################################################################################
# Copyright 2017 Heptio Inc.
#

<output_omitted>
RESULTS_DIR="${RESULTS_DIR:-/tmp/sonobuoy}"
# It's ok for these env vars to be unbound
RESULTS_DIR="${RESULTS_DIR}" SONOBUOY_CONFIG="${SONOBUOY_CONFIG}"
SONOBUOY_ADVERTISE_IP="${SONOBUOY_ADVERTISE_IP}" /sonobuoy master -v 3 --logtostderr

echo -n "${RESULTS_DIR}/$(ls -t "${RESULTS_DIR}" | grep -v done | head -n 1)" > "${RESULTS_DIR}"/done
```

6. View the contents of the `/tmp/sonobuoy` directory. Note the subdirectory is a generated number, yours will be different. The **Tab** key can be used to complete the path.

```
root@sonobuoy:/# ls /tmp/sonobuoy/
d39f2629-fa3c-4a0b-9b33-53080e78b57b

root@sonobuoy:/# cd /tmp/sonobuoy/d39f2629-fa3c-4a0b-9b33-53080e78b57b ; ls
meta  plugins

root@sonobuoy:...57b# find .
.
./plugins
./plugins/systemd-logs
./plugins/systemd-logs/results
./plugins/systemd-logs/results/e-6clr
./plugins/systemd-logs/results/e-6clr/systemd_logs
./plugins/systemd-logs/results/e-5c7t
./plugins/systemd-logs/results/e-5c7t/systemd_logs
./meta
./meta/run.log
./meta/config.json
```

7. The **sonobuoy** command has several options. We will use two to explore the test output.

```
root@sonobuoy:...57b# cd /

root@sonobuoy:/# ./sonobuoy status
         PLUGIN     STATUS   RESULT   COUNT
            e2e    running                1
   systemd-logs   complete                2

Sonobuoy is still running. Runs can take up to 60 minutes.

root@sonobuoy:/# ./sonobuoy logs
<output_omitted>
```

8. Continue to look through tests and results as time permits. Connect to the other pods in the `sonobuoy` namespace and look for log and result files.

   There is also an online, graphical scanner. In testing, inside GCE, the results were blocked and never returned. You may have different outcome in other environments.