Lab 8.1 - Troubleshooting

Monitor Applications

View the **secondapp** pod, it should show as running. This may not mean the application within is working properly, but that the pod is running. The restarts are due to the command we have written to run. The pod exists when done, and the controller restarts another container inside. The count depends on how long the labs have been running.

```
student@ckad-1/app1:~$ cd
student@ckad-1:~$ kubectl get pods secondapp
NAME                            READY     STATUS      RESTARTS    AGE
secondapp                       2/2       Running     49          2d
```

Look closer at the pod. Working slowly through the output, check each line. If you have issues, are other pods having issues on the same node or volume? Check the state of each container. Both **busy** and **webserver** should report **Running**. Note that **webserver** has a restart count of zero, while **busy** has a restart count of 49. We expect this, as in our case, the pod has been running for 49 hours.

```
student@ckad-1:~$ kubectl describe pod secondapp
Name:          secondapp
Namespace:     default
Node:          ckad-2-wdrq/10.128.0.2
Start Time:    Fri, 13 Apr 2018 20:34:56 +0000
Labels:        example=second
Annotations:   <none>
Status:        Running
IP:            192.168.55.91
Containers:
  webserver:
<output_omitted>
```

```
    State:          Running
      Started:      Fri, 13 Apr 2018 20:34:58 +0000
    Ready:          True
    Restart Count:  0
<output_omitted>

  busy:
<output_omitted>

    State:          Running
      Started:      Sun, 15 Apr 2018 21:36:20 +0000
    Last State:     Terminated
      Reason:       Completed
      Exit Code:    0
      Started:      Sun, 15 Apr 2018 20:36:18 +0000
      Finished:     Sun, 15 Apr 2018 21:36:18 +0000
    Ready:          True
    Restart Count:  49
    Environment:    <none>
```

There are three values for conditions. Check that the pod reports **`Initialized`**, **`Ready`** and **`Scheduled`**.

```
<output_omitted>
Conditions:
  Type           Status
  Initialized    True
  Ready          True
  PodScheduled   True
<output_omitted>
```

Check if there are any events with errors or warnings which may indicate what is causing any problems:

```
Events:
  Type    Reason    Age                  From                   Message
  ----    ------    ----                 ----                   -------
  Normal  Pulling   34m (x50 over 2d)  kubelet, ckad-2-wdrq  pulling image
"busybox"
  Normal  Pulled    34m (x50 over 2d)  kubelet, ckad-2-wdrq  Successfully
pulled image "busybox"
  Normal  Created   34m (x50 over 2d)  kubelet, ckad-2-wdrq  Created
container
```

```
  Normal  Started  34m (x50 over 2d)  kubelet, ckad-2-wdrq  Started
container
```

View each container log. You may have to sift errors from expected output. Some containers may have no output at all, as is found with **busy**.

```
student@ckad-1:~$ kubectl logs secondapp webserver
192.168.55.0 - - [13/Apr/2018:21:18:13 +0000] "GET / HTTP/1.1" 200 612 "-"
"curl/7.47.0" "-"
192.168.55.0 - - [13/Apr/2018:21:20:35 +0000] "GET / HTTP/1.1" 200 612 "-"
"curl/7.53.1" "-"
127.0.0.1 - - [13/Apr/2018:21:25:29 +0000] "GET" 400 174 "-" "-" "-"
127.0.0.1 - - [13/Apr/2018:21:26:19 +0000] "GET index.html" 400 174 "-" "-"
"-"
<output_omitted>

student@ckad-1:~$ kubectl logs secondapp busy
student@ckad-1:~$
```

Check to make sure the container is able to use DNS and communicate with the outside world. Remember that we still have limited the UID for **secondapp** to be UID 2000, which may prevent some commands from running. It can also prevent an application from completing expected tasks:

```
student@ckad-1:~$ kubectl exec  -it  secondapp -c busy -- sh
/ $ nslookup www.linux.com
Server:    10.96.0.10
Address 1: 10.96.0.10 kube-dns.kube-system.svc.cluster.local

Name:      www.linux.com
Address 1: 151.101.45.5

/ $ cat /etc/resolv.conf
nameserver 10.96.0.10
search default.svc.cluster.local svc.cluster.local cluster.local
c.endless-station-188822.internal google.internal
options ndots:5
```

Test access to a remote node using **NetCat**. There are several options to **nc** which can help troubleshoot if the problem is the local node, something between systems or the target. In the example below, the connect never completes and a **<ctrl>-c** was used to interrupt.

```
/ $ nc www.linux.com 25
```

```
^Cpunt!
```

Test using an IP address in order to narrow the issue to name resolution. In this case, the IP in use is a well known IP for Google's DNS servers. The following example shows that Internet name resolution is working, but our UID issue prevents access to the **index.html** file.

```
/ $ wget http://www.linux.com/
Connecting to www.linux.com (151.101.45.5:80)
Connecting to www.linux.com (151.101.45.5:443)
wget: can't open 'index.html': Permission denied

/ $ exit
```

Make sure traffic is being sent to the correct Pod. Check the details of both the service and endpoint. Pay close attention to ports in use, as a simple typo can prevent traffic from reaching the proper pod. Make sure labels and selectors don't have any typos as well.

```
student@ckad-1:~$ kubectl get svc
NAME         TYPE           CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
kubernetes   ClusterIP      10.96.0.1        <none>        443/TCP
10d
nginx        ClusterIP      10.108.95.67     <none>        443/TCP
10d
registry     ClusterIP      10.105.119.236   <none>        5000/TCP
10d
secondapp    LoadBalancer   10.109.26.21     <pending>     80:32000/TCP
1d
thirdpage    NodePort       10.109.250.78    <none>        80:31230/TCP
1h

student@ckad-1:~$ kubectl get svc secondapp -o yaml
<output_omitted>
  clusterIP: 10.109.26.21
  externalTrafficPolicy: Cluster
  ports:
  - nodePort: 32000
    port: 80
    protocol: TCP
    targetPort: 80
  selector:
    example: second
```

```
<output_omitted>
```

Verify an endpoint for the service exists and has expected values, including namespaces, ports and protocols.

```
student@ckad-1:~$ kubectl get ep
NAME          ENDPOINTS             AGE
kubernetes    10.128.0.3:6443       10d
nginx         192.168.55.68:443     10d
registry      192.168.55.69:5000    10d
secondapp     192.168.55.91:80      1d
thirdpage     192.168.241.57:80     1h

student@ckad-1:~$ kubectl get ep secondapp -o yaml
apiVersion: v1
kind: Endpoints
metadata:
  creationTimestamp: 2018-04-14T05:37:32Z
<output_omitted>
```

If the containers, services and endpoints are working, the issue may be with an infrastructure service like **kube-proxy**. Ensure it's running, then look for errors in the logs. As we have two nodes, we will have two proxies to look at. As we built our cluster with **kubeadm**, the proxy runs as a container. On other systems, you may need to use **journalctl** or look under **/var/log/kube-proxy.log**.

```
student@ckad-1:~$ ps -elf |grep kube-proxy
4 S root      2864  2847  0  80   0 - 14178 -      15:45 ?        00:00:56
/usr/local/bin/kube-proxy --config=/var/lib/kube-proxy/config.conf
0 S student  23513 18282  0  80   0 -  3236 pipe_w 22:49 pts/0    00:00:00
grep --color=auto kube-proxy

student@ckad-1:~$ journalctl -a | grep proxy
Apr 15 15:44:43 ckad-2-nzjr audit[742]: AVC apparmor="STATUS"
operation="profile_load" profile="unconfined"
name="/usr/lib/lxd/lxd-bridge-proxy" pid=742 comm="apparmor_parser"
Apr 15 15:44:43 ckad-2-nzjr kernel: audit: type=1400
audit(1523807083.011:11): apparmor="STATUS" operation="profile_load"
profile="unconfined" name="/usr/lib/lxd/lxd-bridge-proxy" pid=742
comm="apparmor_parser"
Apr 15 15:45:17 ckad-2-nzjr kubelet[1248]: I0415 15:45:17.153670    1248
reconciler.go:217] operationExecutor.VerifyControllerAttachedVolume started
for volume "xtables-lock" (UniqueName:
```

```
"kubernetes.io/host-path/e701fc01-38f3-11e8-a142-42010a800003-xtables-lock"
) pod "kube-proxy-t8k4w" (UID: "e701fc01-38f3-11e8-a142-42010a800003")
```

Look at both of the proxy logs. Lines which begin with the character `I` are info, `E` are errors. In this example, the last message says access to listing an endpoint was denied by RBAC. It was because a default installation via Helm wasn't RBAC-aware. If you are not using the command line completion, view the possible pod names first.

```
student@ckad-1:~$ kubectl -n kube-system get pod

student@ckad-1:~$ kubectl -n kube-system logs kube-proxy-fsdfr
I0405 17:28:37.091224       1 feature_gate.go:190] feature gates: map[]
W0405 17:28:37.100565       1 server_others.go:289] Flag proxy-mode=""
unknown, assuming iptables proxy
I0405 17:28:37.101846       1 server_others.go:138] Using iptables Proxier.
I0405 17:28:37.121601       1 server_others.go:171] Tearing down inactive
rules.
<output_omitted>
E0415 15:45:17.086081       1 reflector.go:205]
k8s.io/kubernetes/pkg/client/informers/informers_generated/internalversion/
factory.go:85: Failed to list *core.Endpoints: endpoints is forbidden: User
"system:serviceaccount:kube-system:kube-proxy" cannot list endpoints at the
cluster scope: [clusterrole.rbac.authorization.k8s.io "system:node-proxier"
not found, clusterrole.rbac.authorization.k8s.io "system:basic-user" not
found, clusterrole.rbac.authorization.k8s.io "system:discovery" not found]
```

Check that the proxy is creating the expected rules for the problem service. Find the destination port being used for the service, 30195 in this case.

```
student@ckad-1:~$ sudo iptables-save |grep secondapp
-A KUBE-NODEPORTS -p tcp -m comment --comment "default/secondapp:" -m tcp
--dport 30195 -j KUBE-MARK-MASQ
-A KUBE-NODEPORTS -p tcp -m comment --comment "default/secondapp:" -m tcp
--dport 30195 -j KUBE-SVC-DAASHM5XQZF5XI3E
-A KUBE-SEP-YDKKGXN54FN2TFPE -s 192.168.55.91/32 -m comment --comment
"default/secondapp:" -j KUBE-MARK-MASQ
-A KUBE-SEP-YDKKGXN54FN2TFPE -p tcp -m comment --comment
"default/secondapp:" -m tcp -j DNAT --to-destination 192.168.55.91:80
-A KUBE-SERVICES ! -s 192.168.0.0/16 -d 10.109.26.21/32 -p tcp -m comment
--comment "default/secondapp: cluster IP" -m tcp --dport 80 -j
KUBE-MARK-MASQ
```

```
-A KUBE-SERVICES -d 10.109.26.21/32 -p tcp -m comment --comment
"default/secondapp: cluster IP" -m tcp --dport 80 -j
KUBE-SVC-DAASHM5XQZF5XI3E
-A KUBE-SVC-DAASHM5XQZF5XI3E -m comment --comment "default/secondapp:" -j
KUBE-SEP-YDKKGXN54FN2TFPE
-A KUBE-SVC-NPX46M4PTMTKRN6Y -m comment --comment
"default/kubernetes:https" -m recent --rcheck --seconds 10800 --reap --name
KUBE-SEP-2QXHNT77UCWCSQLV --mask 255.255.255.255 --rsource -j
KUBE-SEP-2QXHNT77UCWCSQLV
```

Ensure the proxy is working by checking the port targeted by iptables. If it fails, open a second terminal and view the proxy logs when making a request as it happens.

```
student@ckad-1:~$ curl localhost:32000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<output_omitted>
```

Conformance Testing

The CNCF 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.

Create a new directory to hold the testing code:

```
      student@ckad-1:~$ mkdir test

      student@ckad-1:~$ cd test/
```

Use `git` to download the Sonobuoy code. View the resource after it downloads:

```
student@ckad-1:~/test$ git clone https://github.com/heptio/sonobuoy
Cloning into 'sonobuoy'...
remote: Counting objects: 7847, done.
remote: Compressing objects: 100% (23/23), done.
remote: Total 7847 (delta 2), reused 0 (delta 0), pack-reused 7824
```

```
Receiving objects: 100% (7847/7847), 10.19 MiB | 0 bytes/s, done.
Resolving deltas: 100% (3818/3818), done.
Checking connectivity... done.

student@ckad-1:~/test$ ls
sonobuoy

student@ckad-1:~/test$ cd sonobuoy/

student@ckad-1:~/test/sonobuoy$ ls
cmd                 Gopkg.toml                          scripts
CODE_OF_CONDUCT.md  heptio-images-ee4b0474b93e.json.enc SUPPORT.md
CONTRIBUTING.md     LICENSE                             test
Dockerfile          main.go                             travis-deploy.sh
docs                Makefile                            vendor
examples            pkg
Gopkg.lock          README.md

student@ckad-1:~/test/sonobuoy$ less README.md
```

The Heptio team suggests the use of an easy-to-use Golang tool **gimme**. We will follow their suggestion and use it to pull their code. Start by making sure you have a **/bin** directory under your home directory.

```
student@ckad-1:~/test/sonobuoy$ cd; mkdir ~/bin
```

Use **curl** to download the binary. Note the use of **-o** as in output to save the binary to the newly created directory:

```
student@ckad-1:~$ curl -sL -o ~/bin/gimme \
https://raw.githubusercontent.com/travis-ci/gimme/master/gimme
```

View the file. Note it is not yet executable. Make it so:

```
student@ckad-1:~$ ls -l ~/bin/gimme
-rw-rw-r-- 1 student student 27035 Apr 15 20:46 /home/student/bin/gimme

student@ckad-1:~$ chmod +x ~/bin/gimme
```

Use the **gimme** tool to download the stable version of Go:

```
     student@ckad-1:~$ ~/bin/gimme stable
     unset GOOS;
     unset GOARCH;
     export GOROOT='/home/student/.gimme/versions/go1.10.1.linux.amd64';
     export
     PATH="/home/student/.gimme/versions/go1.10.1.linux.amd64/bin:${PATH}";
     go version >&2;

     export GIMME_ENV="/home/student/.gimme/envs/go1.10.1.env"
```

Ensure the expected path has been set and exported. You can copy the second from the previous output:

```
student@ckad-1:~$ export PATH=$GOROOT/bin:$GOPATH/bin:$PATH

student@ckad-1:~$ export \
PATH="/home/student/.gimme/versions/go1.10.1.linux.amd64/bin:${PATH}"

student@ckad-1:~$ export GOPATH=$PATH
```

Use the **go** command to download the **sonobuoy** code. You may need to install the software package **golang-go**, if it is not already installed:

```
student@ckad-1:~$ go get -u -v github.com/heptio/sonobuoy
github.com/heptio/sonobuoy (download)
created GOPATH=/home/student/go; see 'go help gopath'
github.com/heptio/sonobuoy/pkg/buildinfo
<output_omitted>
```

Execute the newly downloaded tool with the **run** option. Review the output. Take note of interesting tests in order to search for particular output in the logs. The binary has moved over time, so you may need to use the **find** command to locate it. The long path wraps, but you would type the command as one line.

```
student@ckad-1:~/test/sonobuoy$
         ~/.gimme/versions/go1.10.3.linux.amd64/bin/bin/sonobuoy run
Running plugins: e2e, systemd-logs
INFO[0000] created object
name=heptio-sonobuoy namespace= resource=namespaces
INFO[0000] created object
name=sonobuoy-serviceaccount namespace=heptio-sonobuoy
resource=serviceaccounts
```

```
INFO[0000] created object
name=sonobuoy-serviceaccount-heptio-sonobuoy namespace=
resource=clusterrolebindings
INFO[0000] created object
name=sonobuoy-serviceaccount namespace= resource=clusterroles
INFO[0000] created object
name=sonobuoy-config-cm namespace=heptio-sonobuoy resource=configmaps
INFO[0000] created object
name=sonobuoy-plugins-cm namespace=heptio-sonobuoy resource=configmaps
INFO[0000] created object                               name=sonobuoy
namespace=heptio-sonobuoy resource=pods
INFO[0000] created object
name=sonobuoy-master namespace=heptio-sonobuoy resource=services
```

Check the status of **`sonobuoy`**. It can take up to an hour to finish on large clusters. On our two-node cluster, it will take about two minutes.

```
student@ckad-1:~/test/sonobuoy$
~/.gimme/versions/go1.10.3.linux.amd64/bin/bin/sonobuoy \
 status
PLUGIN          STATUS          COUNT
e2e              running         1
systemd_logs    complete        2

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

Look at the logs. If the tests are ongoing, you will see incomplete logs.

```
student@ckad-1:~/test/sonobuoy$
~/.gimme/versions/go1.10.3.linux.amd64/bin/bin/sonobuoy logs
namespace="heptio-sonobuoy"
pod="sonobuoy-systemd-logs-daemon-set-e322ef32b0804cd2-d48np"
container="sonobuoy-worker"
time="2018-04-15T20:50:48Z" level=info msg="Waiting for waitfile"
waitfile=/tmp/results/done
time="2018-04-15T20:50:49Z" level=info msg="Detected done file,
transmitting result file" resultFile=/tmp/results/systemd_logs
namespace="heptio-sonobuoy" pod="sonobuoy" container="kube-sonobuoy"
<output_omitted>
```

Change into the **`client`** directory and look at the tests and results generated:

```
student@ckad-1:~/test/sonobuoy$ cd /home/student/test/sonobuoy/pkg/client/

student@ckad-1:~/test/sonobuoy/pkg/client$ ls
defaults.go  doc.go  example_interfaces_test.go  gen_test.go    logs.go
mode.go       results      run.go
delete.go    e2e.go  gen.go                      interfaces.go
logs_test.go  preflight.go  retrieve.go  status.go

student@ckad-1:~/test/sonobuoy/pkg/client$ cd results/

student@ckad-1:~/test/sonobuoy/pkg/client/results$ ls
doc.go  e2e  junit_utils.go  reader.go  reader_test.go  testdata  types.go

student@ckad-1:~/test/sonobuoy/pkg/client/results$ cd testdata/
student@ckad-1:~/test/sonobuoy/pkg/client/results/testdata$ ls -l
total 644
-rw-rw-r-- 1 student student 407010 Apr 15 20:43 results-0.10.tar.gz
-rw-rw-r-- 1 student student  32588 Apr 15 20:43 results-0.8.tar.gz
-rw-rw-r-- 1 student student 215876 Apr 15 20:43 results-0.9.tar.gz

student@ckad-1:~/test/sonobuoy/pkg/client/results/testdata$ tar -xf
results-0.8.tar.gz

student@ckad-1:~/test/sonobuoy/pkg/client/results/testdata$ ls
config.json  hosts  plugins  resources  results-0.10.tar.gz
results-0.8.tar.gz  results-0.9.tar.gz  serverversion

student@ckad-1:~/test/sonobuoy/pkg/client/results/testdata$ less \
plugins/e2e/results/e2e.log
Dec 13 17:06:53.480: INFO: Overriding default scale value of zero to 1
Dec 13 17:06:53.481: INFO: Overriding default milliseconds value of zero to
5000
Running Suite: Kubernetes e2e suite
===================================
Random Seed: 1513184813 - Will randomize all specs
Will run 1 of 698 specs

Dec 13 17:06:54.705: INFO: >>> kubeConfig:
Dec 13 17:06:54.707: INFO: Waiting up to 4h0m0s for all (but 0) nodes to be
schedulable
Dec 13 17:06:54.735: INFO: Waiting up to 10m0s for all pods (need at least
0) in namespace 'kube-system' to be running a
nd ready
```

```
Dec 13 17:06:54.895: INFO: 14 / 14 pods in namespace 'kube-system' are
running and ready (0 seconds elapsed)
Dec 13 17:06:54.895: INFO: expected 3 pod replicas in namespace
'kube-system', 3 are Running and Ready.
<output_omitted>
```

Find other files which have been generated, and their size:

```
student@ckad-1:~/t…./testdata$ find .
.
./results-0.8.tar.gz
./results-0.10.tar.gz
./hosts
./hosts/ip-10-0-9-16.us-west-2.compute.internal
<output_omitted>
```

Continue to look through tests and results as time permits. There is also an online, graphical scanner. In testing inside GCE, the results were blocked and never returned. You may have a different outcome in other environments.