LFS260

Kubernetes Security Essentials

Version 2022-03-25

Version 2022-03-25

Published by:

the **Linux Foundation**
https://www.linuxfoundation.org

Nondisclosure of Confidential Information

"Confidential Information" shall not include any of the following, even if marked confidential or proprietary: (a) information that relates to the code base of any open source or open standards project (collectively, "Open Project"), including any existing or future contribution thereto; (b) information generally relating or pertaining to the formation or operation of any Open Project; or (c) information relating to general business matters involving any Open Project.

This course does not include confidential information, nor should any confidential information be divulged in class.

Contents

List of Figures

Chapter 1

Introduction

1.1 Labs

Exercise 1.1: Configuring the System for sudo

It is very dangerous to run a **root shell** unless absolutely necessary: a single typo or other mistake can cause serious (even fatal) damage.

Thus, the sensible procedure is to configure things such that single commands may be run with superuser privilege, by using the **sudo** mechanism. With **sudo** the user only needs to know their own password and never needs to know the root password.

If you are using a distribution such as **Ubuntu**, you may not need to do this lab to get **sudo** configured properly for the course. However, you should still make sure you understand the procedure.

To check if your system is already configured to let the user account you are using run **sudo**, just do a simple command like:

```
$ sudo ls
```

You should be prompted for your user password and then the command should execute. If instead, you get an error message you need to execute the following procedure.

Launch a root shell by typing **su** and then giving the **root** password, not your user password.

On all recent **Linux** distributions you should navigate to the `/etc/sudoers.d` subdirectory and create a file, usually with the name of the user to whom root wishes to grant **sudo** access. However, this convention is not actually necessary as **sudo** will scan all files in this directory as needed. The file can simply contain:

```
student ALL=(ALL)  ALL
```

if the user is `student`.

An older practice (which certainly still works) is to add such a line at the end of the file `/etc/sudoers`. It is best to do so using the **visudo** program, which is careful about making sure you use the right syntax in your edit.

You probably also need to set proper permissions on the file by typing:

```
$ sudo chmod 440 /etc/sudoers.d/student
```

(Note some **Linux** distributions may require `400` instead of `440` for the permissions.)

After you have done these steps, exit the root shell by typing `exit` and then try to do `sudo ls` again.

There are many other ways an administrator can configure **sudo**, including specifying only certain permissions for certain users, limiting searched paths etc. The `/etc/sudoers` file is very well self-documented.

However, there is one more setting we highly recommend you do, even if your system already has **sudo** configured. Most distributions establish a different path for finding executables for normal users as compared to root users. In particular the directories `/sbin` and `/usr/sbin` are not searched, since **sudo** inherits the `PATH` of the user, not the full root user.

Thus, in this course we would have to be constantly reminding you of the full path to many system administration utilities; any enhancement to security is probably not worth the extra typing and figuring out which directories these programs are in. Consequently, we suggest you add the following line to the `.bashrc` file in your home directory:

```
PATH=$PATH:/usr/sbin:/sbin
```

If you log out and then log in again (you don't have to reboot) this will be fully effective.

Chapter 2

Cloud Security Overview

2.1 Labs

Exercise 2.1: Building Your Security Team

There is some paperwork and policy writing required to improve security. Part of the process is to write down necessary tasks, who is responsible, how often the task is to be performed, and details of remediation should there be an issue. This checklist is to get started. There are more resources in the NIST documents in the next section.

1. Identity management, credentials, and access
2. Ongoing Administration
3. Risk management process
4. Responding to an security issue
5. Securing data
6. Network Security
7. System Security
8. Application Security

Exercise 2.2: User Identification and Credentials

We will download and take a look at some **FIPS** guides. Each of these guides would take many hours to read and digest. Implementing the security improvements may take quite a while. Plan time after class to read through the documents mentioned in detail.

1. Download the **FIPS 200** and **201-2** guide onto your local system. Start here: https://csrc.nist.gov/publications/fips and locate both titles.
2. Read through the table of contents of **FIPS 200** then section 4 `SECURITY CONTROL SELECTION`. Locate the **NIST SP 800-53** mentioned in the section.
3. Read the table of contents of **NIST SP 800-53**. Take note of total number of pages and how many pages are part of `Appendix F`. Skim through some of the catalog entries such as `AC_4 INFORMATION FLOW ENFORCEMENT`

4. Make a plan to categorize all of your systems as `low-impact`, `moderate-impact` or `high-impact` information systems.
5. Read through the table of contents of **FIPS 201-2**. Then find section `2.1 Control Objectives` and understand the four control objectives and a manner to ensure that objective in your environment.
6. Download **NIST 800-171**. Scan through the sections. Take a look at `Appendix D`, and the mappings to NIST and ISO/IEC Relevant Security Controls.

> If the head of your publicly traded company was **criminally liable** for cybersecurity issues defined in 800-171, would you be able to assure `every item` is being followed?

Exercise 2.3: Tracking Known Issues

1. View https://nvd.nist.gov/vuln/search and type in `Kubernetes` as the Keyword Search.
2. Select a record which indicates a `CRITICAL` vulnerability, and read through current description and the product that it effects. Continue to read through other references, weakness enumeration, and known affected software configurations.
3. Determine who in your organization will be responsible for keeping track of CVE updates. Would it be one group, or a different person in working with individual project software.

Exercise 2.4: CIS Benchmarks

In this exercise we will download a free benchmark for Kubernetes from the Center for Internet Security ®. You may want to schedule a regular return for new or different information.

1. Open a local browser and visit https://www.cisecurity.org/cis-benchmarks/
2. Scroll to expand information on Kubernetes.

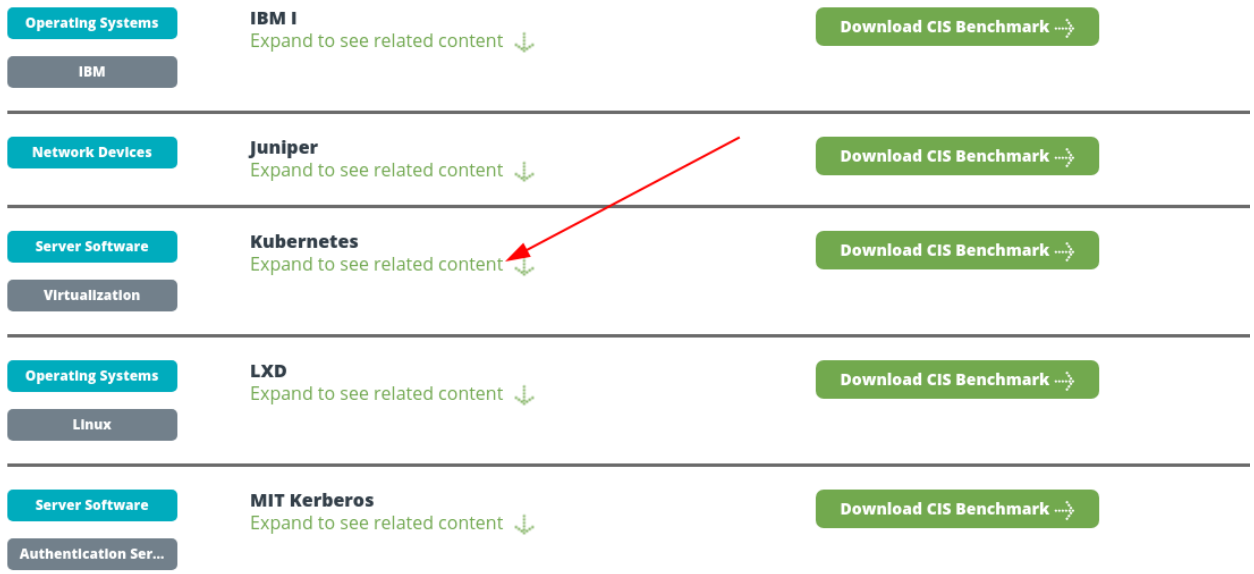


Figure 2.1: **Main CIS Page**

3. Among the expanded list you may note that only one version is considered current. Select the current benchmark.

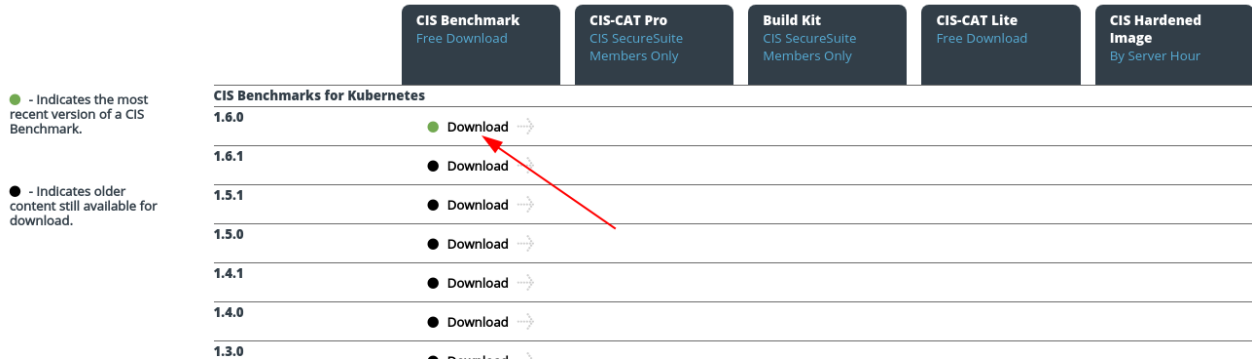


Figure 2.2: **Main CIS Page Kubernetes Detail**

4. Fill out your contact information and accept the terms. An email will be sent to the email given in a minute or two. Inside the email is a link to `Access the PDFs`, which will open a website to download the freely available PDFs.
5. Scroll down the list to find the Kubernetes content. Download the current `CIS Kubernetes Benchmark`.

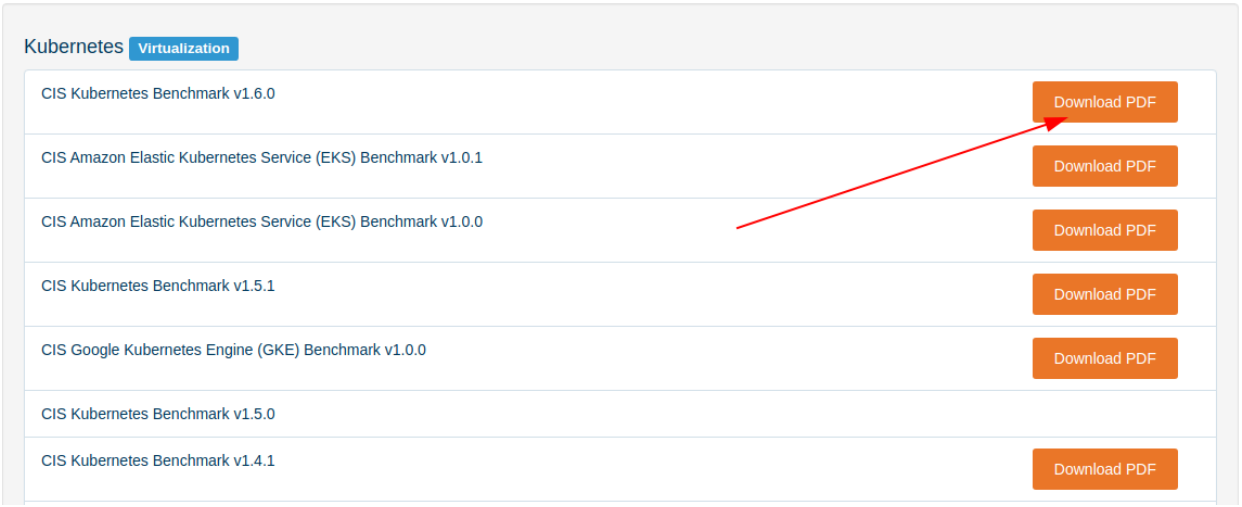
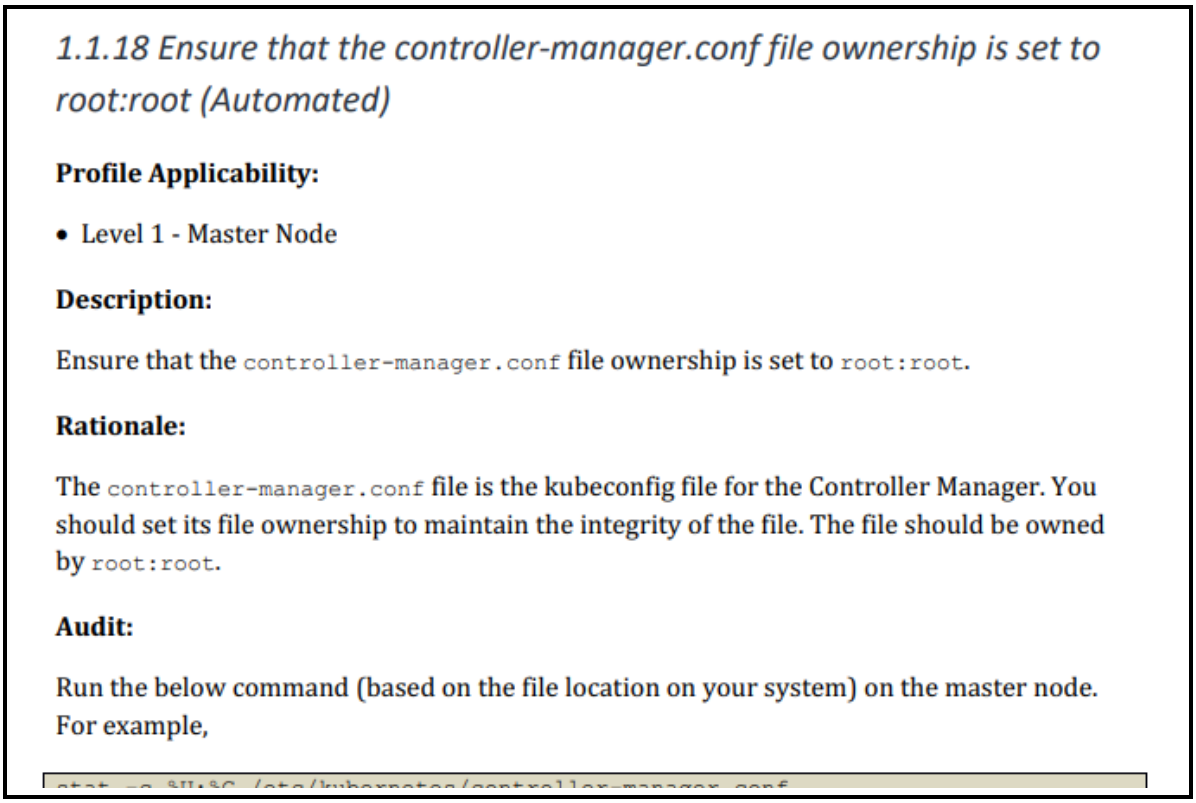


Figure 2.3: **List of PDFs**

6. View the PDF. There are 250+ pages in the document. Instead of reading all of it, find a particular item, such as 1.1.18. Read through the provided information for the item.



1.1.18 Ensure that the controller-manager.conf file ownership is set to root:root (Automated)

Profile Applicability:

- Level 1 - Master Node

Description:

Ensure that the `controller-manager.conf` file ownership is set to `root:root`.

Rationale:

The `controller-manager.conf` file is the kubeconfig file for the Controller Manager. You should set its file ownership to maintain the integrity of the file. The file should be owned by `root:root`.

Audit:

Run the below command (based on the file location on your system) on the master node. For example,

Figure 2.4: **Benchmark Item Details**

7. Skim the rest of the document. Plan a time after class to review the information in detail.

Exercise 2.5: Use First Node

We will create a single-node **Ubuntu 18.04** cluster. Currently 2 vCPU and 8G of memory allows for quick labs. This will be the first of two installs during the course. The exercises will call this node **single**, your node name may be different. Other Linux distributions should work in a similar manner, but have not been tested.

Very Important

Regardless of the platform used (**VirtualBox**, **VMWare**, **AWS**, **GCE** or even bare metal) please remember that security software like **SELinux**, **AppArmor**, and firewall configurations can prevent the labs from working. While not something to do in production consider disabling the firewall and security software, until used in a lab.

The **kubeadm** utility currently requires that swap be turned off on every node. The **swapoff -a** command will do this until the next reboot, with various methods to disable swap persistently. Cloud providers typically deploy instances with swap disabled.

Download shell scripts and YAML files

To assist with setting up please download the tarball of shell scripts and YAML files onto your exercise node.

```
$ wget https://training.linuxfoundation.org/cm/LFS260/LFS260_V2022-03-25_SOLUTIONS.tar.xz \
  --user=LFtraining --password=Penguin2014

$ tar -xvf LFS260_V2022-03-25_SOLUTIONS.tar.xz
```

(Note: depending on your software, if you are cutting and pasting the above instructions, the **underscores may disappear and be replaced by spaces**, so you may have to edit the command line by hand!)

Exercise 2.6: Free CIS Assessment Tool

1. Return to your local browser and navigate to the CIS homepage, https://cisecurity.org. Scroll down to the bottom of the page locate the CIS-CAT Lite under the `Tools` section.

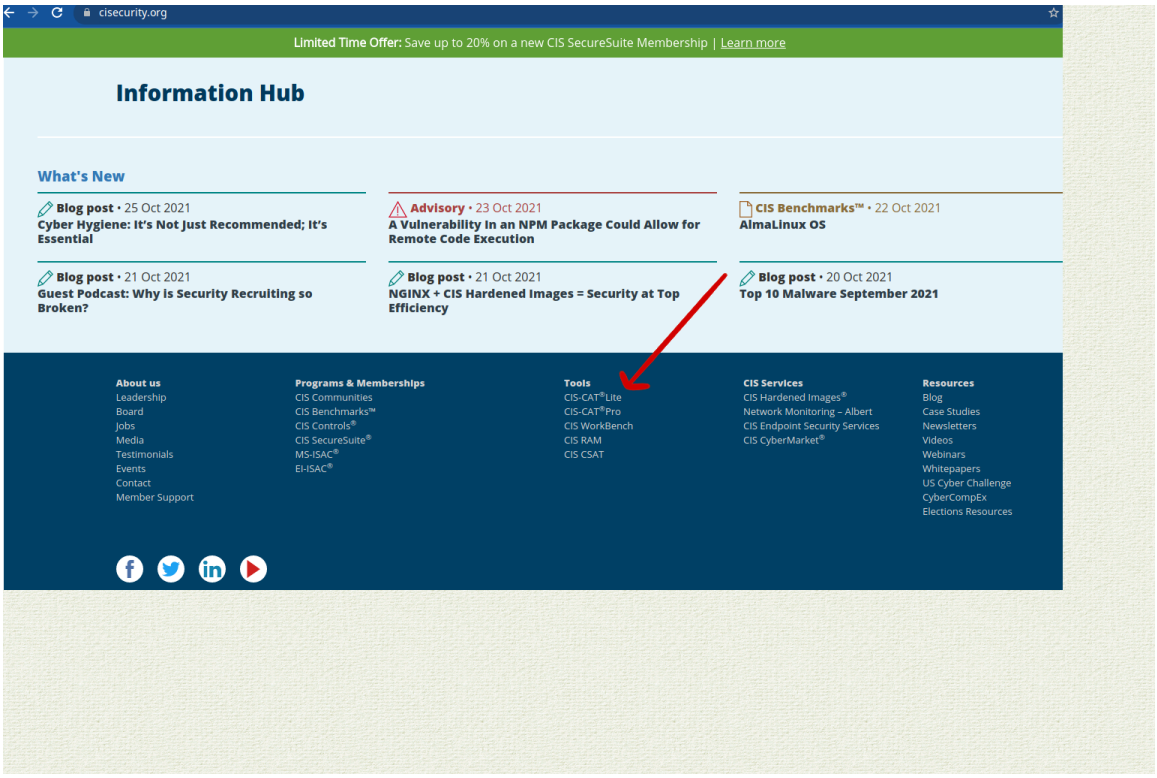


Figure 2.5: **CIS-CAT Tool**

2. Select the `CIS-CAT Lite` automated assessment tool. Note that under the `Compare Key Features` section the `CIS Benchmarks supported` only allows a select number of tools. The pro version for those with memberships has all available tools. At the moment Kubernetes is not part of the select group. We will test **Ubuntu**, both to ensure we are secure as well as to see a sample of what the assessment does and eventually returns. Fill out the form on the right and submit. This will cause an email to be sent to you.

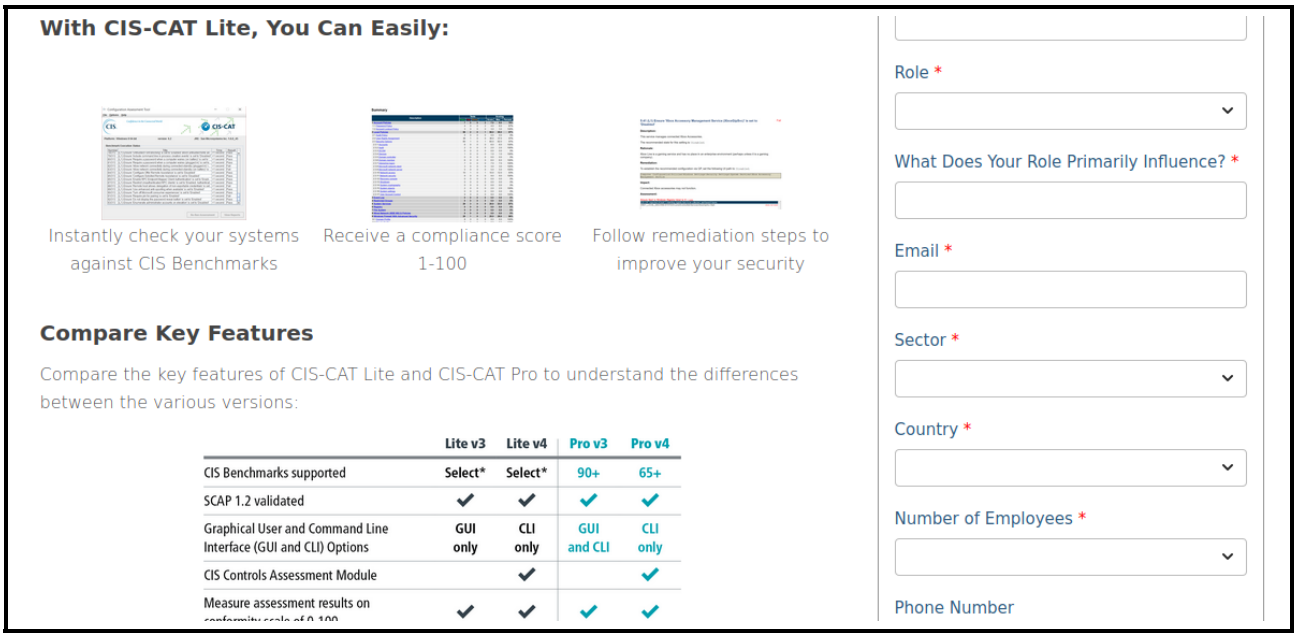


Figure 2.6: **CIS Select Assessments**

3. Inside the email you will eventually receive you will find the option to download version 3 or version 4. While you may want to download the zip file locally to your workstation, let us also download it to our exercise node. Instead of clicking on the green button, hover with the mouse over version `v4` and copy the link. Then log into your exercise node and use the **wget** command to retrieve the file.

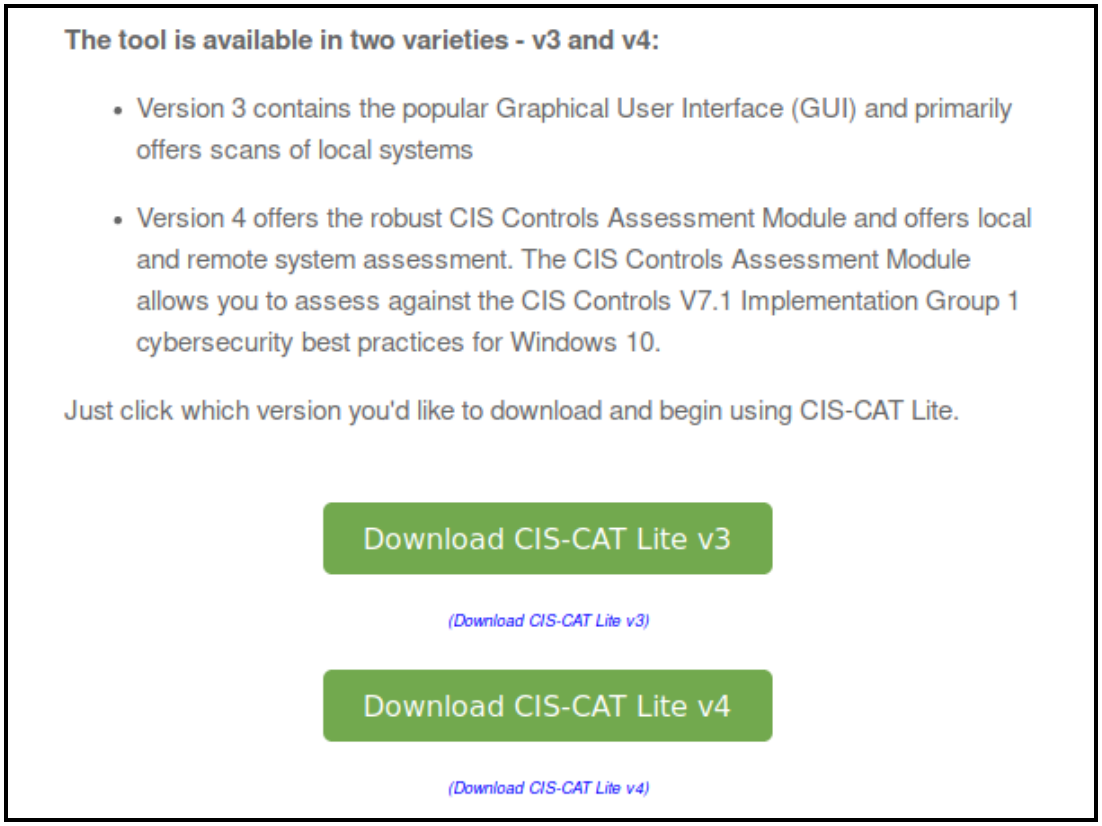


Figure 2.7: **CIS Email**

```
student@single:~$ wget https://learn.cisecurity.org/e/799323/1-799323-2019-11-15-3v7x/2mnnf/\
```

79038343?h=xWidc0ywLqO6rHOWMcM1VXE9q1_WfdjCoVQ-tL2jXks

```
<output_omitted>
```

4. As this manner of download creates a long and difficult file name you can use the **mv** command and tab to rename it to something easier, like `CIS-Cat.zip`.

```
student@single:~$ mv 79038343?h=xWidc0ywLqO6rHOWMcM1VXE9q1_WfdjCoVQ-tL2jXks CIS-Cat.zip
```

5. To extract the files we may first have to install the **unzip** command.

```
student@single:~$ sudo apt-get update ; sudo apt-get install unzip
```

```
<output_omitted>
```

6. Use the newly installed command to extract all the files. You may note there are files to run the assessment tool from a variety of operating systems. After extraction change into the newly created directory.

```
student@single:~$ unzip CIS-Cat.zip
```

```
Archive:  CIS-Cat.zip
   creating: Assessor-CLI/
   creating: Assessor-CLI/benchmarks/
   creating: Assessor-CLI/config/
   creating: Assessor-CLI/custom/
<output_omitted>
```

```
student@single:~$ cd Assessor-CLI
```

7. The assessment tool requires **JAVA**. Install the software then set the `JAVA_PATH` variable.

```
student@single:~/Assessor-CLI$ sudo apt-get install openjdk-11-jdk -y
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
<output_omitted>
```

```
student@single:~/Assessor-CLI$ export JAVA_PATH=/usr/lib/jvm/java-11-openjdk-amd64/bin/
```

8. Run the `Assessor-CLI.sh` script without any options. You should see some warnings and a text graphic, followed by help information. Among that information note the levels of verbosity as well as the `-i` option to run an interactive assessment. Use the **sudo** command as the tool requires root ability.

```
student@single:~/Assessor-CLI$ sudo bash Assessor-CLI.sh
```

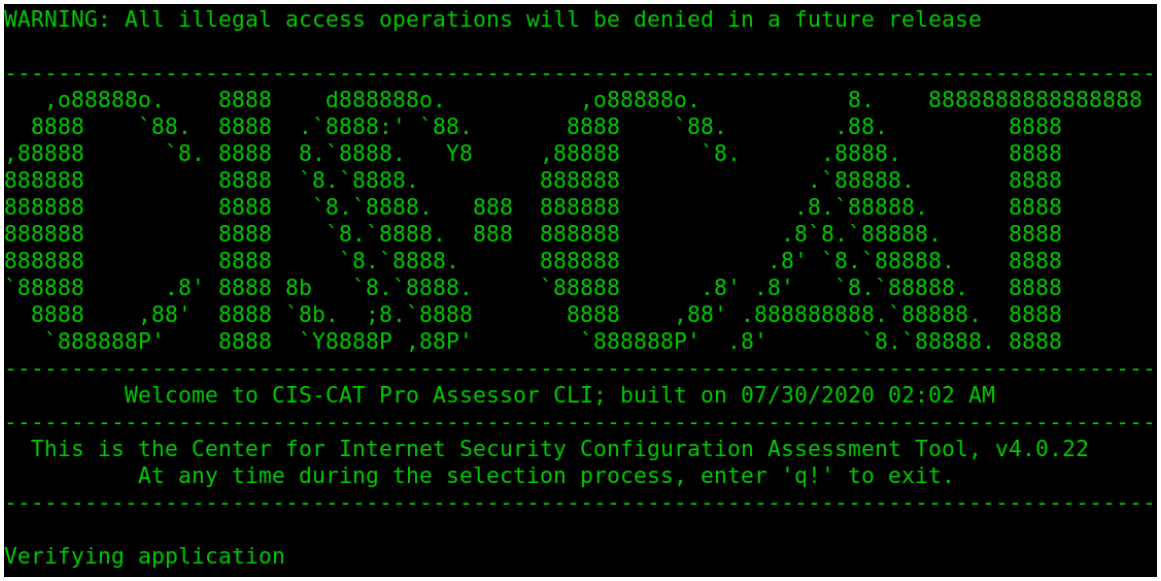


Figure 2.8: **Assessor Help**

9. Run the program again, this time pass the `-i` option. When the `Select Content` prompt appears enter the number `5`, to assess Ubuntu.

```
student@single:~/Assessor-CLI$ sudo bash Assessor-CLI.sh -i
```

```
Verifying application

Attempting to load the default sessions.properties, bundled with the application.

Loading Benchmarks/Data-Stream Collections

Available Benchmarks/Data-Stream Collections:
 1. CIS Controls Assessment Module - Implementation Group 1 for Windows 10 v1.0.3
 2. CIS Controls Assessment Module - Implementation Group 1 for Windows Server v1.0.0
 3. CIS Google Chrome Benchmark v2.0.0
 4. CIS Microsoft Windows 10 Enterprise Release 1909 Benchmark v1.8.1
 5. CIS Ubuntu Linux 18.04 LTS Benchmark v2.0.1
 > Select Content # (max 5): 5                              #<-- Enter 5 here
```

10. You should then see options about what level of testing you want to do. Chose option `1`. There will be a lot of output following. Take a moment to scan through the hundreds of tests. Some will pass, some will fail. At the end of the output you should see total assessment time and a location for the HTML report.

```
Selected 'CIS Ubuntu Linux 18.04 LTS Benchmark'

Assessment File CIS_Ubuntu_Linux_18.04_LTS_Benchmark_v2.0.1-xccdf.xml has a valid Signature.
Profiles:
1. Level 1 - Server
2. Level 2 - Server
3. Level 1 - Workstation
4. Level 2 - Workstation
 > Select Profile # (max 4): 1                 #<<-- Enter 1 here.

Selected Profile 'Level 1 - Server'

Obtaining session connection --> Local
Connection established.
Selected Checklist 'CIS Ubuntu Linux 18.04 LTS Benchmark'
Selected Profile 'Level 1 - Server'
```

```
Starting Assessment
----------------------- ASSESSMENT TARGET ----------------------------------
       Hostname: single
        OS Name: linux
     OS Version: 5.4.0-1028-gcp
OS Architecture: x86_64

<output_omitted>

Ending Assessment - Date & Time: 11-07-2020 04:14:01
Total Assessment Time: 31 seconds
- Generating Asset Reporting Format.
- Collecting Checklist Results.
- Combining Results.
- Saving Results.
- Generating Data-Stream Collection.
- Data-Stream Collection Generated.
- Asset Reporting Format Generated.

 ***** Writing Assessment Results *****
 - Reports saving to /home/student/Assessor-CLI/reports
 -- single-CIS_Ubuntu_Linux_18.04_LTS_Benchmark-20201107T041401Z.html
Assessment Complete for Checklist: CIS Ubuntu Linux 18.04 LTS Benchmark
------------------------------------------------------------
Disconnecting Session.
Exiting; Exit Code: 0
```

11. Copy the listed HTML file to your local machine using **scp**, **sftp**, or some other tool. Your information will be different, such as the IP, hostname, and filename. Copy the path from the Results output. Use a local browser to examine the file. The example below is split to fit on the page, your path would be continuous.

```
local$ scp -i LF-Class.pem student@35.193.19.105:/home/student/Assessor-CLI/reports/\
single-CIS_Ubuntu_Linux_18.04_LTS_Benchmark-20201107T041401Z.html .

local$ firefox single-CIS_Ubuntu_Linux_18.04_LTS_Benchmark-20201107T041401Z.html
```

12. Scroll through the document. The links will take you further into the document for details. Choose a failed test. Review the description, remediation, and the Assessment Evidence.

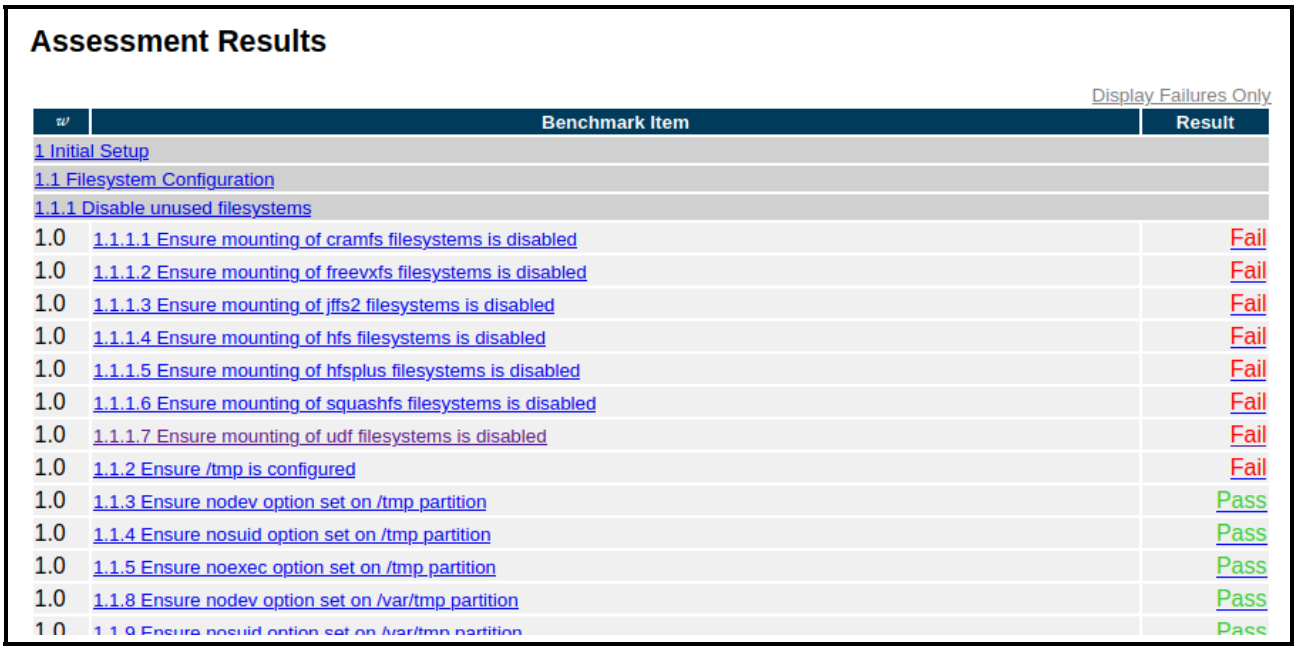


Figure 2.9: **Assessor Help**

Chapter 3

Preparing to Install

3.1 Labs

Exercise 3.1: Investigate Image Supply Chain

Planning should be done for the operating system, the Kubernetes software, and the images which will be deployed. Part of a process of approval will be using tools to scan images, prior to running in Kubernetes.

For each of these tools, investigate these questions:

1. Open source or closed?
2. If open source, does anyone offer support?
3. If closed, what is the pricing scheme?
4. If they have a **github** or **gitlab** repo, how active is the community? How many open issues exist, and how many of them are critical?

- **Trow**: https://trow.io/
- **Prisma Cloud**: https://www.paloaltonetworks.com/prisma/cloud
- **NeuVector**: https://neuvector.com/
- **Clair**: https://github.com/quay/clair
- **Aqua**: https://www.aquasec.com/ and **Trivy**: https://github.com/aquasecurity/trivy
- **Notary**: https://github.com/theupdateframework/notary

Exercise 3.2: Getting Started with Trivy

We can begin checking images before deploying Kubernetes by using **Trivy**. While there are many options and abilities of the tool, we will begin with installation and a basic check of a package with lots of issues, as well as a commonly used image, which we would hope has no issues.

1. Install dependency and other software to get started:

```
student@single:~$ sudo apt-get install wget apt-transport-https gnupg lsb-release -y
```

```
<output_omitted>
```

2. Pull down the public key from Aquasecurity:

```
student@single:~$ wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
```

```
OK
```

3. Update the local repo to get access to the packages. If you do not want to type out the long command line, you can create the file using a text editor. (As shown here, the command has a continuation backslash for space reasons. You can type it on one line without the backslash.)

```
student@single:~$  echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | \
  sudo tee -a /etc/apt/sources.list.d/trivy.list
```

```
deb https://aquasecurity.github.io/trivy-repo/deb bionic main
```

4. Update the cache from the repository:

```
student@single:~$ sudo apt-get update
```

```
<output_omitted>
```

5. Install the **Trivy** software:

```
student@single:~$ sudo apt-get install trivy -y
```

```
<output_omitted>
```

6. Test against a known flawed image. There will be quite a bit of output. Read through, noticing the red words like "critical" which indicate a serious issue. About 15 lines in you will find totals for issues. Note that we are checking the `1.2.3` version of the package and that there are several sections in the output.

```
student@single:~$ trivy image knqyf263/vuln-image:1.2.3
```

```
2020-12-06T04:09:13.502Z        INFO        Need to update DB
2020-12-06T04:09:13.502Z        INFO        Downloading DB...
19.36 MiB / 19.36 MiB [------------------------------] 100.00% 22.80 MiB p/s 1s
2020-12-06T04:09:17.915Z        INFO        Detecting Alpine vulnerabilities...
2020-12-06T04:09:17.921Z        INFO        Detecting  vulnerabilities...
2020-12-06T04:09:17.922Z        INFO        Detecting  vulnerabilities...
2020-12-06T04:09:17.932Z        INFO        Detecting  vulnerabilities...
2020-12-06T04:09:17.932Z        INFO        Detecting  vulnerabilities...
2020-12-06T04:09:17.940Z        WARN        version error (1.9.0.post1): malformed version: 1.9.0.post1
2020-12-06T04:09:17.940Z        WARN        version error (1.9.0.post1): malformed version: 1.9.0.post1
2020-12-06T04:09:17.941Z        INFO        Detecting  vulnerabilities...
2020-12-06T04:09:17.942Z        WARN        This OS version is no longer supported by the distribution: alpine 3.7.1
2020-12-06T04:09:17.942Z        WARN        The vulnerability detection may be insufficient because security updates are not provided

knqyf263/vuln-image:1.2.3 (alpine 3.7.1)
========================================
Total: 76 (UNKNOWN: 0, LOW: 1, MEDIUM: 25, HIGH: 25, CRITICAL: 25)

+---------------------+------------------+----------+-------------------+---------------+---------------------------------------+--------------------------------------+
|       LIBRARY       | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |                 TITLE                 |                 URL                  |
+---------------------+------------------+----------+-------------------+---------------+---------------------------------------+--------------------------------------+
| curl                | CVE-2018-14618   | CRITICAL | 7.61.0-r0         | 7.61.1-r0     | curl: NTLM password overflow          | avd.aquasec.com/nvd/cve-2018-14618   |
|                     |                  |          |                   |               | via integer overflow                  |                                      |
+                     +------------------+          +                   +               +---------------------------------------+--------------------------------------+
|                     | CVE-2018-16839   |          |                   | 7.61.1-r1     | curl: Integer overflow leading        | avd.aquasec.com/nvd/cve-2018-16839   |
|                     |                  |          |                   |               | to heap-based buffer overflow in      |                                      |
|                     |                  |          |                   |               | Curl_sasl_create_plain_message()      |                                      |
+                     +------------------+          +                   +               +                                       +                                      +

<output_omitted>

node-app/package-lock.json
==========================
Total: 8 (UNKNOWN: 0, LOW: 1, MEDIUM: 4, HIGH: 2, CRITICAL: 1)

+---------+------------------+----------+-------------------+---------------+-----------------------------+-----------------------------------+
| LIBRARY | VULNERABILITY ID | SEVERITY | INSTALLED VERSION | FIXED VERSION |            TITLE            |                URL                |
+---------+------------------+----------+-------------------+---------------+-----------------------------+-----------------------------------+
| jquery  | CVE-2019-11358   | MEDIUM   | 3.3.9             | 3.4.0         | jquery: Prototype pollution in | avd.aquasec.com/nvd/cve-2019-11358 |
|         |                  |          |                   |               | object's prototype leading to  |                                   |
|         |                  |          |                   |               | denial of service, remote...   |                                   |
+         +------------------+          +                   +               +-----------------------------+-----------------------------------+
|         | CVE-2020-11022   |          |                   | 3.5.0         | jquery: Cross-site             | avd.aquasec.com/nvd/cve-2020-11022 |
|         |                  |          |                   |               | scripting due to improper      |                                   |
|         |                  |          |                   |               | injQuery.htmlPrefilter method  |                                   |
```

```
+        +-----------------+         +                 +             +------------------------------
```

7. Now check an often used package. Notice that there are 151 total issues with 25 high and one critical. Scan through the list to find the high and critical issues. Note that there are several sections within the output.

```
student@single:~$ trivy image nginx
```

```
2020-12-06T04:09:54.635Z        WARN        You should avoid using the :latest tag as it is cached. You need to specify '--clear-cache' option when :latest image is changed
2020-12-06T04:09:56.962Z        INFO        Detecting Debian vulnerabilities...
2020-12-06T04:09:56.981Z        INFO        Trivy skips scanning programming language libraries because no supported file was detected

nginx (debian 10.6)
===================
Total: 151 (UNKNOWN: 0, LOW: 108, MEDIUM: 17, HIGH: 25, CRITICAL: 1)

+-----------------+--------------------+----------+---------------------------+-------------------+--------------------------------+----------------------------------------+
|     LIBRARY     |  VULNERABILITY ID  | SEVERITY |     INSTALLED VERSION     |    FIXED VERSION  |             TITLE              |                  URL                   |
+-----------------+--------------------+----------+---------------------------+-------------------+--------------------------------+----------------------------------------+
| apt             | CVE-2011-3374      | LOW      | 1.8.2.1                   |                   | It was found that apt-key      | avd.aquasec.com/nvd/cve-2011-3374      |
|                 |                    |          |                           |                   | in apt, all versions, do not   |                                        |
|                 |                    |          |                           |                   | correctly...                   |                                        |
+-----------------+--------------------+----------+---------------------------+-------------------+--------------------------------+----------------------------------------+
<output_omitted>
```

8. Finally we will check a package without vulnerabilities.

```
student@single:~$ trivy image alpine:3.13.1
```

```
2021-01-29T04:27:17.992Z        WARN        This OS version is not on the EOL list: alpine 3.13
2021-01-29T04:27:17.992Z        INFO        Detecting Alpine vulnerabilities...
2021-01-29T04:27:17.993Z        INFO        Trivy skips scanning programming language libraries because no supported file was detected
2021-01-29T04:27:17.993Z        WARN        This OS version is no longer supported by the distribution: alpine 3.13.1
2021-01-29T04:27:17.993Z        WARN        The vulnerability detection may be insufficient because security updates are not provided

alpine:3.13.1 (alpine 3.13.1)
=============================
Total: 0 (UNKNOWN: 0, LOW: 0, MEDIUM: 0, HIGH: 0, CRITICAL: 0)
```

Exercise 3.3: Consider Policy Based Control Organization

1. Determine what tools, if any, will be used to manage access policies.
2. Who will be responsible for maintaining the policy: devops, system admins, the security team, or some combination?
3. Will the default policy be mostly closed, or open?
4. What is the process to allow a new permission? Who has final say?
5. Who is responsible for removing abilities no longer needed? How often will they check?

Exercise 3.4: Investigate Open Policy Agent

Centralized tools may need to be configured prior to installing the cluster, such that they can fully integrate and prevent redoing configuration.

The **Open Policy Agent** is gaining maturity and may replace **Pod Security Policies**, though there are other options. We will deploy **OPA** after we install our cluster. Until then get a feel for working with it using the online play environment.

1. Use https://www.openpolicyagent.org/docs/latest/kubernetes-primer/ for examples and https://play.openpolicyagent.org/ to experiment with some basic rules.
2. On the left side of the Kubernetes primer page you can see information for **SSH**, **sudo**, and **Envoy**. Read through each as if you decide to deploy **OPA**, you should probably centralize as much as you can.
3. Use the Kubernetes examples and create a rule which denies any image with `latest` image tag. Use the playground to evaluate and ensure the rule works as designed. Compare the `REGO` rules shown in the tool compared to the CRDs used by **Gatekeeper** shown in the chapter.

Exercise 3.5: Investigate Sandbox Tools

As the sandbox and more secure tools don't run with standard **kubeadm** installed clusters, you may want to investigate if you require one, and what cluster setup that tool requires.

Evaluate each tool using the same questions as above.

1. **gVisor**: https://github.com/google/gvisor
2. **Kata**: https://katacontainers.io/
3. **Pouch**: http://pouchcontainer.io/
4. **Firecracker**: https://github.com/firecracker-microvm/firecracker-containerd
5. **UniK**: https://github.com/solo-io/unik

Exercise 3.6: Leverage Secure Runtimes

Default **kubeadm** clusters do not yet easily deploy other secure runtimes like **gVisor** or **Kata**. Without these installed and properly configured you can still create the objects, but the pod will remain in pending state.

1. Install Kubernetes using `containerd`. Find and follow `containerd-setup.txt`, which is in the course tarball. (You can find directions to download and extract the tarball earlier in the course.) While you may be able to run it as a **bash** script, it may be of more value to copy and paste each line to see the differences between a typical **Docker** or **cri-o** setup. If you copy and paste, do it from the script, not a pdf file. The path may be different, use the one from the output of the **find** command, not copied from the book.

```
student@single:~$ find $HOME -name containerd-setup.txt
```

```
    <some_long_path>/containerd-setup.txt
```

2. Now that we have a working cluster create the runtime to use `runsc` which is the setting to leverage **gVisor**:

```
student@single:~$ vim runtimeclass.yaml
```

runtimeclass.yaml

```
1 apiVersion: node.k8s.io/v1
2 kind: RuntimeClass
3 metadata:
4   name: gvisor
5 handler: runsc
```

3. Create a pod which uses the runtimeclass:

```
student@single:~$ vim gvisor-pod.yaml
```

gvisor-pod.yaml

```
1 apiVersion: v1
2 kind: Pod
3 metadata:
4   name: simple-gvisor
5 spec:
6   runtimeClassName: gvisor
7   containers:
8   - name: secure-nginx
```

```
9       image: nginx
```

```
student@single:~$ kubectl create -f runtimeclass.yaml
```

```
runtimeclass.node.k8s.io/gvisor created
```

```
student@single:~$ kubectl create -f gvisor-pod.yaml
```

```
pod/simple-gvisor created
```

4. Check the state of the pod.

```
student@single:~$ kubectl get pod
```

```
NAME            READY   STATUS    RESTARTS   AGE
simple-gvisor   1/1     Running   0          46s
```

5. Investigate the details of the pod with **kubectl describe** and **get -o yaml**.

```
student@single:~$ kubectl describe pod simple-gvisor
```

```
Name:         simple-gvisor
Namespace:    default
Priority:     0
Node:         single/10.128.0.44

<output_omitted>
```

6. Remove the `simple-gvisor` pod as we will not be using it.

Please Note

You can shutdown or reset the `single` node. We will use two fresh nodes for the rest of the course.

Chapter 4

Installing the Cluster

4.1 Labs

Exercise 4.1: Overview and Preliminaries

We will create a two-node **Ubuntu 20.04** cluster. Using two nodes allows an understanding of some issues and configurations found in a production environment. The rest of the exercises will use node names **cp** and **worker**, your node names may be different. Currently 2 vCPU and 8G of memory allows for quick labs. Other Linux distributions should work in a similar manner, but have not been tested.

Very Important

Regardless of the platform used (**VirtualBox**, **VMWare**, **AWS**, **GCE** or even bare metal) please remember that security software like **SELinux**, **AppArmor**, and firewall configurations can prevent the labs from working. While not something to do in production consider disabling the firewall and security software.

GCE requires a new `VPC` to be created and a rule allowing all traffic to be included. The use of **wireshark** can be a helpful place to start with troubleshooting network and connectivity issues if you are unable to open all ports.

The **kubeadm** utility currently requires that swap be turned off on every node. The **swapoff -a** command will do this until the next reboot, with various methods to disable swap persistently. Cloud providers typically deploy instances with swap disabled.

Download shell scripts and YAML files

To assist with setting up your cluster please download the tarball of shell scripts and YAML files onto your exercise nodes. The `k8scp.sh` and `k8sWorker.sh` scripts deploy a Kubernetes cluster using **kubeadm** and use `Project Calico` for networking. Should the file not be found you can always use a browser to investigate the parent directory.

```
$ wget https://training.linuxfoundation.org/cm/LFS260/LFS260_V2022-03-25_SOLUTIONS.tar.xz \
  --user=LFtraining --password=Penguin2014

$ tar -xvf LFS260_V2022-03-25_SOLUTIONS.tar.xz
```

(Note: depending on your software, if you are cutting and pasting the above instructions, the underscores may disap-

pear and be replaced by spaces, so you may have to edit the command line by hand!)

Typing out the commands is helpful to the learning process, and is encouraged. Use the included YAML files, as typing them out leads to frustration and does not add value. The labs are not written just for copy and paste. Use the included YAML files and type out the commands leveraging your shell recall features.

Exercise 4.2: Deploy a New Cluster

Deploy a Control Plane Node using Kubeadm

1. Log into your nodes using **PuTTY** or using **SSH** from a terminal window. Log in as a non-root user. The labs assume the user name is `student`. You can use a different account name as long as you remember that when looking at the prompt in the book. It may be a good idea to use different text or window colors as both nodes should be installed the same way. Upon logging into the node you should see something like this:

```
student@cp:~$
```

2. Use the **wget** command shown in the overview to download the course files tarball onto the control plane node, then extract the files using **tar**.

3. Review the script to install and begin the configuration of the control plane (cp) Kubernetes server. You may need to change the **find** command search directory for your home directory depending on how and where you downloaded the tarball of course files mentioned previously.

A **find** command is shown if you want to locate and copy to the current directory instead of creating the file. Mark the command for reference as it may not be shown for future commands.

```
student@cp:~$ find $HOME -name <YAML File>

student@cp:~$ cp LFS260/<Some Path>/<YAML File> .
```

```
student@cp:~$ find $HOME -name k8scp.sh
```

```
student@cp:~$ more LFS260/SOLUTIONS/s_04/k8scp.sh
```

LFS260/SOLUTIONS/s_04/k8scp.sh

```
#!/bin/bash
....
## TxS 03-2022
## v1.23.1 CKA/CKAD/CKS
echo "This script is written to work with Ubuntu 20.04"
sleep 3
echo
echo "Disable swap until next reboot"
echo
sudo swapoff -a

echo "Update the local node"
sudo apt-get update && sudo apt-get upgrade -y
echo
echo "Install Docker"
sleep 3
```

```
sudo apt-get install -y docker.io
echo
echo "Install kubeadm, kubelet, and kubectl"
sleep 3
<output_omitted>
```

4. Run the script as an argument to the **bash** shell. You will need the `kubeadm join` command (shown near the end of the output) when you add the worker/minion node in a future step. Use the **tee** utility to save the output of the script, in case you cannot scroll back to find the `kubeadm join` in the script output. Please note the following is one command and then its output.

   Using **Ubuntu 18** you will be asked questions during the installation. Allow restarts and use the local, installed software if asked during the update (usually option 2.)

   Copy files to your home directory first:

```
student@cp:~$ cp LFS260/SOLUTIONS/s_04/k8scp.sh .

student@cp:~$ bash k8scp.sh | tee $HOME/cp.out
```

```
<output_omitted>

Your Kubernetes cp has initialized successfully!

      To start using your cluster, you need to run the
      following as a regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

      You should now deploy a pod network to the cluster.
      Run \verb?kubectl apply -f [podnetwork].yaml? with one
      of the options listed at:
      https://kubernetes.io/docs/concepts/cluster-administration/addons/

      You can now join any number of machines by running the
      following on each node as root:

  kubeadm join 10.128.0.3:6443 --token 69rdjq.2x20l2j9ncexy37b
  --discovery-token-ca-cert-hash

sha256:72143e996ef78301191b9a42184124416aebcf0c7f363adf9208f9fa599079bd

<output_omitted>

NAME   STATUS     ROLES                  AGE     VERSION
cp     NotReady   control-plane,cp       18s     v1.23.1



Script finished. Move to the next step
```

Deploy a Minion Node

5. Open a separate terminal into your **second node**. Having both terminal sessions allows you to monitor the status of the cluster while adding the second node. Change the color or other characteristic of the second terminal to make it visually distinct from the first. This will keep you from running commands on the incorrect instance, which probably won't work.

Use the previous **wget** command to download the tarball to the second node. Extract the files with **tar** as before. Find and copy `k8sWorker.sh` to `student`'s home directory, then view it. You should see the same early steps as on the cp system.

```
student@worker:~$ more k8sWorker.sh
```

SH **k8sWorker.sh**

```
#!/bin/bash -x
## TxS 01-2022
## CKA/CKAD/CKS for 1.23.1
##
echo "  This script is written to work with Ubuntu 18.04"
echo
sleep 3
echo "  Disable swap until next reboot"
echo
sudo swapoff -a

echo "  Update the local node"
sleep 2
sudo apt-get update && sudo apt-get upgrade -y
echo
sleep 2

echo "  Install and configure Docker"
sleep 2
....
```

6. Run the script on the **second node**. Again, please note you may have questions during the update. Allow daemons to restart and use the local installed version (usually option 2.)

```
student@worker:~$ bash k8sWorker.sh | tee $HOME/worker.out
```

```
<output_omitted>
```

7. When the script is done, the minion node is ready to join the cluster. The `kubeadm join` statement can be found near the end of the `kubeadm init` output on the cp node. It should also be in the file `cp.out` as well. Your nodes will use a different IP address and hashes than the example below. You will need to pre-pend **sudo** to run the script copied from the cp node. Also note that some non-**Linux** operating systems and tools insert extra characters when multi-line samples are copied and pasted. Copying one line at a time solves this issue.

```
student@worker:~$ sudo kubeadm join --token 118c3e.83b49999dc5dc034 \
 10.128.0.3:6443 --discovery-token-ca-cert-hash \
sha256:40aa946e3f53e38271bae24723866f56c86d77efb49aedeb8a70cc189bfe2e1d
```

```
<output_omitted>
```

Exercise 4.3: Configure the Cluster

1. Use a configuration script to install a text editor. While the lab uses **vim**, any text editor such as **emacs** or **nano** will work. Be aware that Windows editors may have issues with special characters.

   Take a look at the script, to see other configuration steps. You can use **find** to locate the script; your location may be different. You can then copy, paste and execute the script. You may need to make the script executable with `chmod +x`, depending on how you extracted the file.

```
student@cp:~$ find $HOME -name setupscript.sh
```

```
/home/student/LFS260/SOLUTIONS/s_04/setupscript.sh
```

```
student@cp:~$ /home/student/LFS260/SOLUTIONS/s_04/setupscript.sh
```

```
<response_omitted>
```

2. By default the cp node will not allow general containers to be deployed **for security reasons**, This is done via a `taint` by **kubeadm**.

   This is not typically done in a production environment for security and resource contention reasons. The following command will remove the taint from all nodes, so you should see one success and one `not found` error. The worker/minion node does not have the taint to begin with. Note the **minus sign** at the end of the command, which removes the preceding value.

```
student@cp:~$  kubectl describe nodes | grep -i Taint
```

```
Taints:             node-role.kubernetes.io/master:NoSchedule
Taints:             <node>
```

```
student@cp:~$ kubectl taint nodes --all node-role.kubernetes.io/master-
```

```
node/cp untainted
taint "node-role.kubernetes.io/master:" not found
```

3. Check that both nodes are without a `Taint`. If they both are without taint the `nodes` should now show as `Ready`. It may take a minute or two for all pods to enter `Ready` state.

```
student@cp:~$ kubectl describe nodes | grep -i taint
```

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

```
student@cp:~$ kubectl get nodes
```

```
NAME     STATUS   ROLES                  AGE     VERSION
cp       Ready    control-plane,master   10m     v1.23.1
worker   Ready    <none>                 2m13s   v1.23.1
```

Exercise 4.4: Using the kube-bench tool

Now that we have a cluster and have already viewed the CIS Benchmark in a previous chapter, you can also use the **kube-bench** tool from Aquasecurity which shows a text output of the issues followed by commands to run. This may allow an easier path to keeping your environment secure.

1. Download the Aquasecurity **kube-bench** binary. The original URL is so long the command is being split. There is no need to include the backslashes if running as a single command. You can also navigate to github.com/aquasecurity/kube-bench/releases to find newer or other architecture versions.

```
student@cp:~$ curl -L \
 https://github.com/aquasecurity/kube-bench/releases/download/v0.6.6/kube-bench_0.6.6_linux_amd64.deb \
 -o  kube-bench_0.6.6_linux_amd64.deb
```

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   656  100   656    0     0   2699      0 --:--:-- --:--:-- --:--:--  2699
100 6473k  100 6473k    0     0  8246k      0 --:--:-- --:--:-- --:--:-- 11.9M
```

2. Install and verify the binary is in your search path.

```
student@cp:~$ sudo apt install ./kube-bench_0.6.6_linux_amd64.deb
```

```
<output_omitted>
```

```
student@cp:~$ which kube-bench
```

```
/usr/local/bin/kube-bench
```

3. Use **sudo** to run the **kube-bench** command and expect quite a bit of output. There may be differences from the output below as the software is dynamic.

```
student@cp:~$ sudo kube-bench
```

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or ....
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root ....
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to....

<output_ommitted>

== Remediations ==
1.1.9 Run the below command (based on the file location on your system) on the cp node.

For example,
chmod 644 <path/to/cni/files>

1.1.10 Run the below command (based on the file location on your system) on the cp node.
For example,
chown root:root <path/to/cni/files>


<output_omitted>

== Summary policies ==
0 checks PASS
0 checks FAIL
24 checks WARN
0 checks INFO

== Summary total ==
69 checks PASS
12 checks FAIL
41 checks WARN
0 checks INFO
```

4. Compare the results with what you saw with the official CIS benchmark PDF we looked at in a previous exercise. They should be very similar to each other.

Exercise 4.5: Deploy OPA Gatekeeper

We discussed using **Open Policy Agent (OPA)** in the preparation chapter, and will now deploy the tool inside of our cluster and ensure it works.

1. To keep track of the YAML files we will use create a new directory and then change directory into it.

```
student@cp:~$ mkdir gk

student@cp:~$ cd gk
```

2. We will be using a few YAML files included in the course tarball. Use **find** to locate and copy the files to the current directory. Read through each file before use and become familiar with their contents. The files are from https://github.com/open-policy-agent/gatekeeper.

   Scan through then use the `gatekeeper.yaml` file to deploy **OPA**. You will get a warning about some depredations and upcoming changes in API version, you can safely ignore the warning at this time.

```
student@cp:~/gk$ more gatekeeper.yaml
```

```
<output_omitted>
```

```
student@cp:~/gk$ kubectl create -f gatekeeper.yaml
```

```
namespace/gatekeeper-system unchanged
resourcequota/gatekeeper-critical-pods configured
customresourcedefinition.apiextensions.k8s.io/assign.mutations.gatekeeper.sh configured
customresourcedefinition.apiextensions.k8s.io/assignmetadata.mutations.gatekeeper.sh configured
customresourcedefinition.apiextensions.k8s.io/configs.config.gatekeeper.sh configured
customresourcedefinition.apiextensions.k8s.io/constraintpodstatuses.status.gatekeeper.sh
↪  configured
....
```

3. Investigate the new namespace and ensure the pods are all running properly.

```
student@cp:~/gk$ kubectl get ns
```

```
NAME                STATUS   AGE
default             Active   10m
gatekeeper-system   Active   43s
kube-node-lease     Active   10m
kube-public         Active   10m
kube-system         Active   10m
```

```
student@cp:~/gk$ kubectl -n gatekeeper-system get pod
```

```
NAME                                             READY   STATUS    RESTARTS   AGE
gatekeeper-audit-54b5f86d57-k6qnf                1/1     Running   0          52s
gatekeeper-controller-manager-5b96bd668-cxvs8    1/1     Running   0          52s
gatekeeper-controller-manager-5b96bd668-fhv6g    1/1     Running   0          52s
gatekeeper-controller-manager-5b96bd668-hvkp8    1/1     Running   0          52s
```

4. Add a new template and constraint to only allow new namespaces which have a particular label. These YAML files are in the course tarball.

```
student@cp:~/gk$ kubectl create -f gk-ns-constraintTemplate.yaml
```

```
constrainttemplate.templates.gatekeeper.sh/k8srequiredlabels created
```

```
student@cp:~/gk$ kubectl create -f gk-ns-constraint.yaml
```

```
k8srequiredlabels.constraints.gatekeeper.sh/ns-require-label created
```

5. Test that the new constraint prevents new namespace creation if the namespace does not have the proper label.

```
student@cp:~/gk$ kubectl create ns nolabel
```

```
Error from server ([denied by ns-require-label] You must provide labels:
{"gk-ns"}): admission webhook "validation.gatekeeper.sh" denied the
request: [denied by ns-require-label] You must provide labels: {"gk-ns"}
```

6. Create a new namespace, this time with the required label of `gk-ns`.

```
student@cp:~/gk$ kubectl create -f newNS.yaml
```

```
namespace/haslabel created
```

```
student@cp:~/gk$ kubectl get ns haslabel
```

```
NAME       STATUS   AGE
haslabel   Active   46s
```

7. Add another template and constraint, this time limiting the source image repository.

```
student@cp:~/gk$ kubectl create -f gk-image-constraintTemplate.yaml
```

```
constrainttemplate.templates.gatekeeper.sh/k8srequiredregistry created
```

```
student@cp:~/gk$ kubectl create -f gk-image-constraint.yaml
```

```
k8srequiredregistry.constraints.gatekeeper.sh/only-quay-images created
```

8. Now test creating two new deployments, one of which uses a `Quay.io` image.

```
student@cp:~/gk$ kubectl create deployment dockerhub --image=nginx
```

```
deployment.apps/dockerhub created
```

```
student@cp:~/gk$ kubectl create deployment quay.io --image=quay.io/prometheus/prometheus
```

```
deployment.apps/quay.io created
```

9. Investigate the status of Pods, Deployments, and ReplicaSets. Note the error does not show for the Deployment, nor is there a pod with the error. The ReplicaSet shows the error.

```
student@cp:~/gk$ kubectl get pod
```

```
NAME                       READY   STATUS    RESTARTS   AGE
quay.io-fbc5b687d-8tlch    1/1     Running   0          30s
```

```
student@cp:~/gk$ kubectl get deploy
```

```
NAME        READY   UP-TO-DATE   AVAILABLE   AGE
dockerhub   0/1     0            0           87s
quay.io     1/1     1            1           63s
```

```
student@cp:~/gk$ kubectl describe deploy dockerhub
```

```
Name:                   dockerhub
Namespace:              default
CreationTimestamp:      Tue, 02 Mar 2021 21:22:35 +0000
Labels:                 app=dockerhub
Annotations:            deployment.kubernetes.io/revision: 1
Selector:               app=dockerhub
Replicas:               1 desired | 0 updated | 0 total | 0 available | 1 unavailable
<output_omitted>
Conditions:
  Type             Status  Reason
  ----             ------  ------
```

```
  Progressing      True    NewReplicaSetCreated
  Available        False   MinimumReplicasUnavailable
  ReplicaFailure   True    FailedCreate
<output_omitted>
```

```
student@cp:~/gk$ kubectl describe rs dockerhub-fccf5c68f
```

```
<output_omitted>
Events:
  Type     Reason        Age                    From                   Message
  ----     ------        ----                   ----                   -------
  Warning  FailedCreate  106s (x17 over 7m13s)  replicaset-controller  Error
creating: admission webhook "validation.gatekeeper.sh" denied the request:
[denied by only-quay-images] Forbidden registry: nginx
```

10. Now that we know Gatekeeper is working, let us recover resources on and remove Gatekeeper to ensure it doesn't prevent deploying other tools in the course.

```
student@cp:~/gk$ kubectl delete deploy dockerhub quay.io

student@cp:~/gk$ kubectl delete ns haslabel

student@cp:~/gk$ kubectl delete -f gatekeeper.yaml
```

```
<outputs_omitted>
```

11. Test that you can create a namespace without a label and a deployment from any registry. Then remove the new resources.

Chapter 5

Securing the kube-apiserver

5.1 Labs

Exercise 5.1: Limiting Users with RBAC

1. Create a new role, `limitone`, which allows users to `get`, `watch`, and `delete` pods and secrets in the `prod-a` namespace:

```
student@cp:~$ vim limitonerole.yaml
```

limitonerole.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: prod-a
  name: limitone
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "delete"]
```

2. Add the new role leveraging the **kubectl auth reconcile** command. Then verify the role has been created:

```
student@cp:~$ kubectl auth reconcile -f limitonerole.yaml
```

```
role.rbac.authorization.k8s.io/limitone reconciled
        reconciliation required create
        missing rules added:
                {Verbs:[get watch delete] APIGroups:[] Resources:[pods] ResourceNames:[]
NonResourceURLs:[]}
```

```
student@cp:~$ kubectl -n prod-a get role
```

```
NAME       CREATED AT
limitone   2020-12-03T06:42:30Z
```

3. Bind the role to the user `paul` in the same namespace:

```
student@cp:~$ vim limitonebind.yaml
```

limitonebind.yaml

```
1  apiVersion: rbac.authorization.k8s.io/v1
2  kind: RoleBinding
3  metadata:
4    name: manage-pods
5    namespace: prod-a
6  subjects:
7  - kind: User
8    name: paul
9    apiGroup: rbac.authorization.k8s.io
10 roleRef:
11   kind: Role
12   name: limitone
13   apiGroup: rbac.authorization.k8s.io
```

4. Add the rolebinding again using the **kubectl auth reconcile** command. Then test that it has been properly added:

```
student@cp:~$ kubectl auth reconcile -f limitonebind.yaml
```

```
rolebinding.rbac.authorization.k8s.io/manage-pods reconciled
        reconciliation required create
        missing subjects added:
                {Kind:User APIGroup:rbac.authorization.k8s.io Name:paul Namespace:}
```

```
student@cp:~$ kubectl -n prod-a get rolebindings.rbac.authorization.k8s.io
```

```
NAME          ROLE           AGE
manage-pods   Role/limitone   26s
```

5. Create another role without the ability to delete and bind it to the user `dan` in the `dev-ns` namespace. See how quickly you can do it, perhaps using bookmarked YAML files from kubernetes.io/doc.

Exercise 5.2: Service Accounts

Just as we have allowed certain permissions for users, we often need to grant privileges to containers running inside of a pod, for which we use **Service Accounts**. This time we will create the objects in the `prod-b` namespace:

1. Create a simple pod which allows shell access:

```
student@cp:~$ vim simplepod.yaml
```

simplepod.yaml

```
1  apiVersion: v1
2  kind: Pod
3  metadata:
4    name: simple-pod
```

```
  namespace: prod-b
spec:
  containers:
  - name: simple
    image: nginx
```

2. Create and verify the pod is running in the proper namespace:

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

```
pod/simple-pod created
```

```
student@cp:~$ kubectl -n prod-b get pod
```

```
NAME          READY   STATUS    RESTARTS   AGE
simple-pod    1/1     Running   0          45s
```

3. View the current service account for the newly created pod, and where the information is kept inside the pod. We will need the directory later, when we use the included token.

```
student@cp:~$ kubectl -n prod-b get pod simple-pod -o yaml | grep service
```

```
    - mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  serviceAccount: default
  serviceAccountName: default
```

4. Make a call to the API server and observe the results. Log into the container and query the API server, which should return an error. You should find the IP address of your API server before logging into the container, as it may be different than the one in the example below.

```
student@cp:~$ kubectl -n prod-b exec -it simple-pod -- bash
```

On Container

(a) Use **curl** to check the `v1` API on the server. You can either find and pass the IP of the cp server and port 6443 or leverage the DNS name `kubernetes.default` and port 443.

```
root@simple-pod:/# curl https://kubernetes.default:443/api/v1 --insecure
```

```
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {

  },
  "status": "Failure",
  "message": "forbidden: User \"system:anonymous\" cannot get path \"/api/v1\"",
  "reason": "Forbidden",
  "details": {

  },
  "code": 403
```

(b) View the secret provided files mounted in the service account directory:

```
root@simple-pod:/# ls /var/run/secrets/kubernetes.io/serviceaccount
```

```
ca.crt        namespace  token
```

(c) Use the `token` file to query the API server again. Set the `TOKEN` variable first, then pass the variable in the header of the **curl** command. You can put the command on one line with the backslash.

```
root@simple-pod:/# export TOKEN=$(cat /run/secrets/kubernetes.io/serviceaccount/token)

root@simple-pod:/# curl -H "Authorization: Bearer $TOKEN" \
 https://kubernetes.default:443/api/v1 --insecure
```

```
{
  "kind": "APIResourceList",
  "groupVersion": "v1",
  "resources": [
    {
      "name": "bindings",
      "singularName": "",
      "namespaced": true,
      "kind": "Binding",
<output_omitted>
```

(d) Now try to view pods first in the `default` namespace and next in the `prod-b` namespace. Both should fail, at the moment. Remember you can use up-arrow and edit the previous command.

```
root@simple-pod:/# curl -H "Authorization: Bearer $TOKEN" \
 https://kubernetes.default:443/api/v1/namespaces/default/pods/ --insecure
```

```
....
  "status": "Failure",
  "message": "pods is forbidden: User \"system:anonymous\" cannot list
   resource \"pods\" in API group \"\" in the namespace \"default\"",
  "reason": "Forbidden",
....
```

```
root@simple-pod:/# curl -H "Authorization: Bearer $TOKEN" \
 https://kubernetes.default:443/api/v1/namespaces/prod-b/pods/ --insecure
```

```
....
  "status": "Failure",
  "message": "pods is forbidden: User \"system:anonymous\" cannot list
   resource \"pods\" in API group \"\" in the namespace \"prod-b\"",
  "reason": "Forbidden",
....
```

(e) Exit from the container.

```
root@simple-pod:/# exit
```

5. View all service accounts in the `prod-b` namespace, then at the YAML, to make it easier to create a new one. If you decide to write the output to a file remember to edit the namespace to avoid overwriting the service account and causing yourself a headache.

```
student@cp:~$ kubectl -n prod-b get sa
```

```
NAME      SECRETS   AGE
default   1         20h
```

```
student@cp:~$ kubectl -n prod-b get sa default -o yaml
```

```
apiVersion: v1
kind: ServiceAccount
metadata:
  creationTimestamp: "2020-12-02T07:26:07Z"
  name: default
  namespace: prod-b
  resourceVersion: "3559"
  selfLink: /api/v1/namespaces/prod-b/serviceaccounts/default
  uid: 4936be35-8df2-4c19-a4c9-7bcbf0fa24e2
secrets:
- name: default-token-w6lm9
```

6. Create a new service account called `simple-sa`, then add it to the cluster.

```
student@cp:~$ vim prodbSA.yaml
```

prodbSA.yaml

```
1 apiVersion: v1
2 kind: ServiceAccount
3 metadata:
4   name: simple-sa
5   namespace: prod-b
6 secrets:
```

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

```
serviceaccount/simple-sa created
```

7. Create a role which we can bind to the new service account. In this case we will use the `list` verb only.

```
student@cp:~$ vim SArole.yaml
```

SArole.yaml

```
1 apiVersion: rbac.authorization.k8s.io/v1
2 kind: Role
3 metadata:
4   namespace: prod-b
5   name: sa-role
6 rules:
7 - apiGroups: [""]
8   resources: ["pods"]
9   verbs: ["list"]
```

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

```
role.rbac.authorization.k8s.io/sa-role created
```

8. Bind the new role to the service account.

```
student@cp:~$ vim SArolebind.yaml
```

SArolebind.yaml

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: sa-role-bind
  namespace: prod-b
subjects:
- kind: ServiceAccount
  name: simple-sa
  namespace: prod-b
roleRef:
  kind: Role
  name: sa-role
  apiGroup: rbac.authorization.k8s.io
```

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

```
rolebinding.rbac.authorization.k8s.io/sa-role-bind created
```

9. Add the service account to the pod. Edit the yaml and recreate the pod.

```
student@cp:~$ vim simplepod.yaml
```

```
....
  namespace: prod-b
spec:
  serviceAccountName: simple-sa     #<<-- Add this line
  containers:
  - name: simple
....
```

```
student@cp:~$ kubectl delete -f simplepod.yaml ; kubectl create -f simplepod.yaml
```

```
pod "simple-pod" deleted
pod/simple-pod created
```

10. Log into the container and test the new permissions. As before check the `default` namespace, then the `prod-b` namespace. This time you should be able to view pods in `prod-b`.

```
student@cp:~$ kubectl -n prod-b exec -it simple-pod -- bash
```

On Container

(a) Set the TOKEN value and try to view the pods inside the `default` namespace. Remember to edit the IP of the server to match your IP.

```
root@simple-pod:/# export TOKEN=$(cat /run/secrets/kubernetes.io/serviceaccount/token)

root@simple-pod:/# curl -H "Authorization: Bearer $TOKEN"  \
  https://10.128.0.39:6443/api/v1/namespaces/default/pods/ --insecure
```

```
....
  "status": "Failure",
  "message": "pods is forbidden: User \"system:serviceaccount:prod-b:simple-sa\" cannot
   list resource \"pods\" in API group \"\" in the namespace \"default\"",
....
```

(b) Now view pods in the `prod-b` namespace. You should see at least `simple-pod`.

```
root@simple-pod:/# curl -H "Authorization: Bearer $TOKEN" \
  https://10128.0.39:6443/api/v1/namespaces/prod-b/pods/ --insecure
```

```
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "selfLink": "/api/v1/namespaces/prod-b/pods/",
    "resourceVersion": "420384"
  },
  "items": [
    {
      "metadata": {
        "name": "simple-pod",
        "namespace": "prod-b",
<output_omitted>
```

(c) If you were able to see pods then exit out of the container.

```
root@simple-pod:/# exit
```

11. Delete the pod to reclaim resources.

```
student@cp:~$ kubectl -n prod-b delete pod simple-pod
```

```
pod "simple-pod" deleted
```

Exercise 5.3: Researching Pod Security Policies

In this lab we will learn about some of the many things that can be controlled via Pod Security Policies. Use online resources to find the answers to the following items. Solutions to each follow in the Solutions section to check your work.

1. Which policy would I use to limit pods using hostPath to a directory, such as `/data` and all subdirectories?
2. Which policy and what YAML stanza with CAP_????? would be used if you want to allow a pod from fully controlling the node's networking?
3. If a developer requires known unsafe `sysctls`, such as what high-performance computing may require, what yaml would you need to put into the pod spec to allow it?

Solution 5.3

1. AllowedHostPaths
2. AllowedCapabilities policy with:

```yaml
securityContext:
      capabilities:
        add: ["NET_ADMIN"]
```

3. Pod spec would include:

```yaml
allowedUnsafeSysctls:
  - kernel.msg*
```

Exercise 5.4: Enable Pod Security Policies

1. View the existing pod security policies in the cluster. If you find one for `gatekeeper-admin` you may want to remove all resources using the `gatekeeper.yaml` file. You'll also see a deprecation warning.

```
student@cp:~$ kubectl get psp --all-namespaces
```

```
Warning: policy/v1beta1 PodSecurityPolicy is deprecated in v1.21+, unavailable in v1.25+
No resources found
```

2. Create a new PSP to prevent creating privileged pods and disallows pods which run as root.

```
student@cp:~$ vim nopriv.yaml
```

nopriv.yaml

```yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
  name: no-priv
spec:
  privileged: false
  runAsUser:
    rule: MustRunAsNonRoot
  seLinux:
    rule: RunAsAny
  fsGroup:
    rule: RunAsAny
  supplementalGroups:
    rule: RunAsAny
  volumes:
  - '*'
```

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

```
podsecuritypolicy.policy/restrict-root created
```

3. View the newly created pod security policy. Ensure that privilege enhancement is not allowed, and pods must run as a non-root user.

```
student@cp:~$ kubectl get psp
```

```
NAME            PRIV    CAPS   SELINUX    RUNASUSER          FSGROUP    SUPGROUP   READONLYROOTFS
↪  VOLUMES
restrict-root   false          RunAsAny   MustRunAsNonRoot   RunAsAny   RunAsAny   false
↪  *
```

4. Create a new database pod running `mariadb`.

```
student@cp:~$ kubectl run --image=mariadb db-one --port=3306 --env="MYSQL_ROOT_PASSWORD=LFtr@in"
```

```
pod/db-one created
```

5. Check the condition of the pod after a short while, it should be running, even though the PSP would otherwise prevent it starting. We will remedy the situation in the next step.

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

```
NAME     READY   STATUS    RESTARTS   AGE
db-one   1/1     Running   0          41s
```

6. Add the `PodSecurityPolicy` admission controller to the kube-apiserver. The pod should restart and be ready within a minute unless there is a typo.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
- --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt
    - --enable-admission-plugins=NodeRestriction,PodSecurityPolicy  #<--Add admission controller
    - --enable-bootstrap-token-auth=true
    - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
```

7. View the pod. It should continue to be running.

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

```
NAME     READY   STATUS    RESTARTS   AGE
db-one   1/1     Running   0          4m29s
```

8. Delete the pod, then recreate it. After a minute check the status of the pod.

```
student@cp:~$ kubectl delete pod db-one
```

```
pod "db-one" deleted
```

```
student@cp:~$ kubectl run --image=mariadb db-one --port=3306 --env="MYSQL_ROOT_PASSWORD=LFtr@in"
```

```
pod/db-one created
```

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

```
NAME     READY   STATUS                       RESTARTS   AGE
db-one   0/1     CreateContainerConfigError   0          95s
```

9. View the details of the pod, you should find an error as to why the pod is not running.

```
student@cp:~$ kubectl describe pod db-one
```

```
....
  Normal   Pulled     60s                kubelet            Successfully pulled
image "mariadb" in 598.824901ms
  Warning  Failed     47s (x8 over 116s)  kubelet            Error: container has runAsNonRoot
and image will run as root (pod: "db-one_default(0cfaefba-90c2-4204-837d-
e6ff571de452)", container: db-one)
```

```
  Normal   Pulled     47s                kubelet            Successfully pulled
image "mariadb" in 624.584689ms
  Normal   Pulling    36s (x9 over 116s)  kubelet            Pulling image "mariadb"
....
```

10. Edit the `nopriv.yaml` file and change the `runAsUser` value to be `RunAsAny`.

```
student@cp:~$ vim nopriv.yaml
```

```
name: no-priv
spec:
  privileged: false
  runAsUser:
    rule: RunAsAny         #<-- Edit this to allow RunAsAny
  seLinux:
    rule: RunAsAny
```

11. Replace the configuration in use and recreate the pod.

```
student@cp:~$ kubectl replace -f nopriv.yaml
```

```
podsecuritypolicy.policy/restrict-root replaced
```

```
student@cp:~$ kubectl delete pod db-one
```

```
pod "db-one" deleted
```

```
student@cp:~$  kubectl run --image=mariadb db-one --port=3306 --env="MYSQL_ROOT_PASSWORD=LFtr@in"
```

```
pod/db-one created
```

12. Verify the pod is now able to run properly.

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

```
NAME     READY   STATUS    RESTARTS   AGE
db-one   1/1     Running   0          83s
```

13. Now create a new deployment using YAML which will use the same image and settings. The `db-two.yaml` file is part of the course tar ball. Then create the deployment.

```
student@cp:~$ vim db-two.yaml
```

db-two.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: db-two
  name: db-two
  namespace: default
spec:
  selector:
    matchLabels:
```

```
      app: db-two
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: db-two
    spec:
      containers:
      - image: mariadb
        name: mariadb
        ports:
          - containerPort: 3306
            name: db-port
        env:
          - name: MYSQL_ROOT_PASSWORD
            value: LFtrain
```

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

```
deployment.apps/db-two created
```

14. Check on status of the deployment. You should find that the deployment shows no pods running. Examine the deployment and then the replicaset, which will have a different name than the example below, looking for errors or indications as to why the pod is not running. The deployment may not indicate much, the error will be seen with the replicaSet.

```
student@cp:~$ kubectl get deploy
```

```
NAME     READY   UP-TO-DATE   AVAILABLE   AGE
db-two   0/1     0            0           19s
```

```
student@cp:~$ kubectl describe deploy db-two
```

```
<output_omitted>
```

```
student@cp:~$ kubectl describe rs db-two-6fd7fc85c9
```

```
....
Conditions:
  Type             Status  Reason
  ----             ------  ------
  ReplicaFailure   True    FailedCreate
Events:
  Type     Reason        Age                    From                   Message
  ----     ------        ----                   ----                   -------
  Warning  FailedCreate  3m12s (x17 over 8m40s)  replicaset-controller  Error creating: pods
"db-two-6fd7fc85c9-" is forbidden: PodSecurityPolicy: unable to admit pod: []
```

15. Following the concept of least privilege we could continue to create serviceAccounts until all required steps are allowed. In our case remove the `PodSecurityPolicy` admission controller and the pods and deployments used for this lab. The commands are shown without details below as the steps should be familiar.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml

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

student@cp:~$ kubectl delete pod db-one
```

```
student@cp:~$ kubectl delete deploy db-two
```

16. **Challenge Step:** Use what you have learned about service accounts and RBAC roles to limit a psp to the `dev-ns` namespace.

Exercise 5.5: Enabling API Server Auditing

1. Create a simple policy file for auditing which collects all messages of `metadata` level.

```
student@cp:~$ sudo vim /etc/kubernetes/simple-policy.yaml
```

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
- level: Metadata
```

2. Edit the manifest file for `kube-apiserver` and add three sections. One section to declare the audit policy and audit log location. We will follow with convention and put the entries in alphabetical order. The next section will add `mountPath:` for both, and the third section will be two `hostPath:` stanzas. Align the new stanzas with existing entries of the same type.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```yaml
....
  - command:
    - kube-apiserver
    - --advertise-address=10.128.0.61
    - --allow-privileged=true
    - --audit-log-maxage=7                                    #<-- Retain age in days
    - --audit-log-maxbackup=2                                 #<-- Max number to retain
    - --audit-log-maxsize=50                                  #<-- Meg size when to rotate
    - --audit-log-path=/var/log/audit.log                     #<-- Where to log
    - --audit-policy-file=/etc/kubernetes/simple-policy.yaml  #<-- Audit policy file
    - --authorization-mode=Node,RBAC
    - --client-ca-file=/etc/kubernetes/pki/ca.crt

....

volumeMounts:
  - mountPath: /etc/kubernetes/simple-policy.yaml  #<-- Use same file names here
    name: audit
    readOnly: true
  - mountPath: /var/log/audit.log                  #<-- And here
    name: audit-log
    readOnly: false

....

- hostPath:                                        #<-- Add these eight lines
    path: /etc/kubernetes/simple-policy.yaml
    type: File
  name: audit
- hostPath:
    path: /var/log/audit.log
    type: FileOrCreate
  name: audit-log
```

3. As soon as the edit is made to the file **kubelet** will restart the pod with new values. As it is restarting the `kube-apiserver` the **kubectl** command won't properly function. If you wait for a minute and the pod is still not running then perhaps you have a typo or a missing component. Use the **docker** command to see if the container is running. If not running check the yaml files for typos.

   You can also look at the container logs, which may indicate the error.

   ```
   student@cp:~$ sudo docker ps |grep apiserver
   ```

   ```
   800f047e147a    1b74e93ece2f          "kube-apiserver --ad..."   About a minute ago  Up About a
   ↪  minute\
             k8s_kube-apiserver_kube-apiserver-cp_kube-system_ad21eb0ae9148bfb17931da01d10663a_0
   ed8775f64ff8    k8s.gcr.io/pause:3.2  "/pause"                   About a minute ago  Up About a
   ↪  minute\
             k8s_POD_kube-apiserver-cp_kube-system_ad21eb0ae9148bfb17931da01d10663a_0
   ```

   If you cannot tell why the container is not starting, also look if there is a log.

   ```
   student@cp:~$ sudo sudo cat /var/log/containers/kube-apiserver-cp-<YOUR-CONTAINER-NAME>
   ```

4. Use the **tail -f** command to view the ongoing messages in the audit file we set, `/var/log/audit.log`. Use the **ctrl-c** to end the tail, which will be rather verbose.

   ```
   student@cp:~$ tail -f /var/log/audit.log
   ```

   ```
   {"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Metadata","auditID":
   "54bdc490-366d-44c8-bc33-a5ad92bb1e72","stage":"ResponseComplete","requestURI":
   "/apis/crd.projectcalico.org/v1/clusterinformations/default","verb":"get","user":
   {"username":"system:serviceaccount:kube-system:calico-kube-controllers","uid":
   "503bae91-4679-4bda-9050-79ccc1550cf9","groups":["system:serviceaccounts","system:
   serviceaccounts:kube-system","system:authenticated"]},"sourceIPs":
   ["192.168.219.65"],"userAgent":"Go-http-client/2.0","objectRef":{"resource":
   "clusterinformations","name":"default","apiGroup":"crd.projectcalico.org","apiVersion":
   "v1"},"responseStatus":{"metadata":{},"code":200},"requestReceivedTimestamp":
   "2020-09-22T20:25:57.507036Z","stageTimestamp":"2020-09-22T20:25:57.510191Z","annotations":
   {"authentication.k8s.io/legacy-token":"system:serviceaccount:kube-system:
   calico-kube-controllers","authorization.k8s.io/decision":"allow","authorization.k8s.io/reason":
   "RBAC: allowed by ClusterRoleBinding \"calico-kube-controllers\" of ClusterRole
    \"calico-kube-controllers\" to ServiceAccount \"calico-kube-controllers/
   kube-system\""}}
   {"kind":"Event","apiVersion":"audit.k8s.io/v1","level":"Metadata","auditID":
   "7832f6ac-3dab-48d2-bc0e-ee08cd2d7d36","stage":"RequestReceived","requestURI":
   "/healthz?timeout=32s","verb":"get","user":{"username":"system:serviceaccount:
   kube-system:calico-kube-controllers","uid":"503bae91-4679-4bda-9050-79ccc1550cf9","groups":
   ["system:serviceaccounts","system:serviceaccounts:kube-system","system:
   authenticated"]},"sourceIPs":["192.168.219.65"],"userAgent":"kube-controllers/v0.0.0
   (linux/amd64) kubernetes/$Format","requestReceivedTimestamp":"2020-09-22T20:25:
   57.511648Z","stageTimestamp":"2020-09-22T20:25:57.511648Z","annotations":
   {"authentication.k8s.io/legacy-token":"system:serviceaccount:kube-system:
   calico-kube-controllers"}}
   <output_omitted>
   ```

5. This file will get big fast, and planning should be made for the storage. Take a look at how big the file is in this short time, almost 2M over seconds in the example below.

   ```
   student@cp:~$ ls -lh /var/log/audit.log
   ```

   ```
   -rw-r--r-- 1 root root 1.7M Sep 22 20:29 /var/log/audit.log
   ```

6. To temporarily stop auditing, edit the manifest file again and comment out the audit-policy line. After that audit file should no longer get larger.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
    - --allow-privileged=true
#    - --audit-policy-file=/etc/kubernetes/simple-policy.yaml   #<-- Comment this line
    - --audit-log-path=/var/log/audit.log
```

7. Verify that the size and the modify line of the audit file does not change. You can also run **tail -f** again and note that no new events are logged.

```
student@cp:~$ ls -l /var/log/audit.log
```

```
-rw-r--r-- 1 root root 15698868 Sep 22 20:33 /var/log/audit.log
```

8. Create a new Audit policy file, this time with more complex settings.

```
student@cp:~$ sudo vim /etc/kubernetes/moderate-policy.yaml
```

```yaml
apiVersion: audit.k8s.io/v1
kind: Policy
omitStages:
  - "RequestReceived"
rules:
  - level: RequestResponse
    resources:
    - group: ""
      resources: ["pods"]
  - level: Metadata
    resources:
    - group: ""
      resources: ["pods/log", "pods/status"]
  - level: Metadata
    userGroups: ["system:authenticated"]
    nonResourceURLs:
    - "/api*"
    - "/version"
  - level: Request
    resources:
    - group: ""
      resources: ["configmaps"]
    namespaces: ["kube-system"]
  - level: Metadata
    resources:
    - group: ""
      resources: ["secrets", "configmaps"]
    omitStages:
      - "RequestReceived"
```

9. Edit the apiserver configuration file again. This time uncomment and edit the line to point at the new `moderate-policy.yaml` file. Also edit the `hostPath` statement to include the new name as well.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```yaml
....
    - --allow-privileged=true
    - --audit-policy-file=/etc/kubernetes/moderate-policy.yaml  #<- Edit this line
    - --audit-log-path=/var/log/audit.log
....
```

```
    volumeMounts:
    - mountPath: /etc/kubernetes/moderate-policy.yaml #<- Edit this line
      name: audit
....
  - name: audit
    hostPath:
      path: /etc/kubernetes/moderate-policy.yaml      #<- Edit this line
      type: File
```

Exercise 5.6: Encrypting Secrets

As secrets are base64 encoded, and not encrypted by default they may not be as secure as desired. In this lab we will configure the API server to encrypt the secrets.

1. Begin by verifying that secrets are only encoded, and not encrypted. Create a new secret with a key and string easy to find in the output.

```
student@cp:~$ kubectl create secret generic first -n default --from-literal=somekey=findme
```

```
secret/first created
```

2. Verify that you can view the secret, and know that the value saved is what was expected.

```
apiVersion: v1
data:
  somekey: ZmluZG1l               # Note the value is encoded
kind: Secret
metadata:
  creationTimestamp: "2020-12-11T18:45:27Z"
  managedFields:
....
```

3. Use **base64** to check the encoded value of the secret we created, which should match the value returned by **kubectl get secret**. Note that as there is no `newline` in the output your next prompt will immediately follow the value.

```
student@cp:~$ echo 'ZmluZG1l' | base64 --decode
```

```
findmestudent@cp:~$
```

4. Run a command inside the **etcd** container to view the secret. You may want to ensure the location of the pki files to copy and paste the long command. It is presented below with comments, the backslashes and comments do not need to be typed out. Look through the output to find the key and value of the secret, verify it is encoded or clear text.

```
student@cp:~$ sudo grep etcd /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
- --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
- --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
- --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
- --etcd-servers=https://127.0.0.1:2379
```

```
student@cp:~$ kubectl -n kube-system exec -it etcd-cp -- sh -c \
"ETCDCTL_API=3                                  # Open quote, then set version
ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt  # pki files from previous output
ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt
ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key
```

```
etcdctl --endpoints=https://127.0.0.1:2379          # Command and setting endpoint
get /registry/secrets/default/first"                # Arguments to the command, close quote
```

```
/registry/secrets/default/first
k8s


v1Secret

firstdefault"*$01619fd7-a850-47b7-adfb-bf3021f8fe072----z-b
kubectl-createUpdatev-----FieldsV1:0
.{"f:data":{".":{},"f:somekey":{}},"f:type":{}}
somekeyfindme-Opaque-"
```

5. Generate a random key to use for encryption using **base64**. Your output will be different from the example below.

```
student@cp:~$ head -c 32 /dev/urandom | base64
```

```
EzqaiOSIGChGOwf0VdbkFtYPUM2EYf1TAAQbDrfizJQ=
```

6. Create an encryption configuration yaml file. Add the encoded string created in the previous command.

```
student@cp:~$ vim encryptionconfig.yaml
```

encryptionconfig.yaml

```
1  apiVersion: apiserver.config.k8s.io/v1
2  kind: EncryptionConfiguration
3  name: newsetup
4  resources:
5    - resources:
6      - secrets
7      providers:
8      - aescbc:
9          keys:
10         - name: firstkey
11           secret: EzqaiOSIGChGOwf0VdbkFtYPUM2EYf1TAAQbDrfizJQ=
12     - identity: {}
```

7. In order for the configuration file to be seen by the **kube-apiserver** container it must be in a directory mounted to the pod. While we could create a new volume, for now let's copy the file to a known and typically protected directory.

```
student@cp:~$ sudo cp encryptionconfig.yaml /etc/kubernetes/pki/
```

8. Edit the manifest file for kube-apiserver and add the `--encryption-provider-config=` option among the others.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
1  ....
2      - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
3      - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
4      - --encryption-provider-config=/etc/kubernetes/pki/encryptionconfig.yaml #<--Add this
5      image: k8s.gcr.io/kube-apiserver:v1.19.0
6      imagePullPolicy: IfNotPresent
7  ....
```

9. When an edit to files in the manifests directory takes place the **inotify** event will cause **kubelet** to update the API server pod. Should there be a typo or other issue there will not be a server to reply to any APIs. If **kubectl** commands fail, find and view the log file. Typically errors near the bottom of the file indicated the issue. Use <TAB> for the long filename.

```
student@cp:~$ sudo ls /var/log/containers/kube-apiserver-*
```

```
/var/log/containers/kube-apiserver-cp_kube-system_kube-apiserver-\
a1b3f379b7929e2143aecda7533045d0776c7f22370c15993ed2a8d298f78dd8.log
```

```
student@cp:~$ sudo tail /var/log/containers/kube-apiserver-<TAB>
```

10. Create another secret named `second` and verify you can see it using **kubectl**. It may be easiest to leverage earlier commands and just edit the secret name.

```
student@cp:~$ kubectl create secret generic second -n default --from-literal=anotherkey=hidden
```

```
secret/second created
```

```
student@cp:~$ kubectl -n kube-system exec -it etcd-cp -- sh -c\
                  "ETCDCTL_API=3 ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt\
                   ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt\
                   ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key etcdctl\
                   --endpoints=https://127.0.0.1:2379 get /registry/secrets/default/first"
```

```
/registry/secrets/default/first
k8s


v1Secret-
-
first-default"*$01619fd7-a850-47b7-adfb-bf3021f8fe072----z-b
kubectl-createUpdate-v----FieldsV1:0
.{"f:data":{".":{},"f:somekey":{}},"f:type":{}}
somekeyfindme-Opaque-"
```

11. Now let us check to see how the data is kept inside the **etcd** database. Run the command from our previous check, but change out the last part to be `second` instead of `first`.

```
student@cp:~$ kubectl -n kube-system exec -it etcd-cp -- \
sh -c "ETCDCTL_API=3 ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt \
ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt \
ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key etcdctl --endpoints=https://127.0.0.1:2379 \
get /registry/secrets/default/second"
```

```
/registry/secrets/default/second
<Several lines of odd and non-printable characters>
```

12. Check the `first` secret again inside of **etcd** database. It should be clear text as encryption happens when the secret is created.

```
student@cp:~$ kubectl -n kube-system exec -it etcd-cp -- \
sh -c "ETCDCTL_API=3 ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt \
ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt \
ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key etcdctl --endpoints=https://127.0.0.1:2379 \
get /registry/secrets/default/first"
```

```
/registry/secrets/default/first
k8s


v1Secret
```

```
firstdefault"*$01619fd7-a850-47b7-adfb-bf3021f8fe072----z-b
kubectl-createUpdatev-----FieldsV1:0
.{"f:data":{".":{},"f:somekey":{}},"f:type":{}}
somekeyfindme-Opaque-"
```

13. Re-write the all the secrets and verify `first` is encrypted, along with the rest.

```
student@cp:~$ kubectl get secrets --all-namespaces -o json | kubectl replace -f -
```

```
secret/default-token-gj9k6 replaced
secret/first replaced
secret/second replaced
secret/default-token-wmxfz replaced
secret/paul-token-ssfb7 replaced
<output_omitted>
```

```
student@cp:~$ kubectl -n kube-system exec -it etcd-cp -- \
sh -c "ETCDCTL_API=3 ETCDCTL_CACERT=/etc/kubernetes/pki/etcd/ca.crt \
ETCDCTL_CERT=/etc/kubernetes/pki/etcd/server.crt \
ETCDCTL_KEY=/etc/kubernetes/pki/etcd/server.key etcdctl \
--endpoints=https://127.0.0.1:2379 get /registry/secrets/default/first"
```

```
/registry/secrets/default/first
<Several lines of odd and non-printable characters>
```

Chapter 6

Networking

6.1 Labs

Exercise 6.1: Implement Network Security Policy

In this exercise we will create three new pods and services in different namespaces, and then create policies to allow some traffic and block other traffic.

1. Find and apply the Please see SOLUTIONS/s_06/netpods.yaml script to create pods and services for the lab. You path to the file may be different, use the path returned by **find**

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

/home/student/LFS260/SOLUTIONS/s_06/netpods.yaml

student@cp:~$ kubectl create -f /home/student/LFS260/SOLUTIONS/s_06/netpods.yaml
  pod/devapp created
  pod/frontend created
  pod/backend created
  service/devapp created
  service/frontend created
  service/backend created
```

2. Take a look at all pods and services, and note that the newly created resources are spread across three namespaces. Excluding resources from `kube-system` namespace may be helpful.

```
student@cp:~$ kubectl get pod,svc --all-namespaces |grep -v kube-system
  NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
  dev-ns        pod/devapp                        1/1     Running   0          159m
  prod-a        pod/frontend                      1/1     Running   0          159m
  prod-b        pod/backend                       1/1     Running   0          159m

  NAMESPACE     NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE
  default       service/kubernetes   ClusterIP   10.96.0.1        <none>        443/TCP         9d
  dev-ns        service/devapp       ClusterIP   10.108.70.106    <none>        80/TCP,22/TCP   159m
```

```
prod-a      service/frontend    ClusterIP   10.98.182.27     <none>      80:30854/TCP   159m
prod-b      service/backend     ClusterIP   10.110.101.117   <none>      80/TCP         159m
```

3. On devapp install network utilities, and determine default access between pods in different namespaces.

```
student@cp:~$ kubectl -n dev-ns exec -it devapp -- bash
```

On Container

(a) Install some network testing software, in case you would like to further investigate network traffic between namespaces. The command is split, but could be run as a single command without the backslash.

```
root@devapp:/# apt-get update ; \
  DEBIAN_FRONTEND=noninteractive apt-get install iputils-ping netcat ssh curl iproute2 -y
```

```
  <output_omitted>
```

(b) Test access to the frontend pod in the prod-a namespace and backend pod in the prod-b. The output should be the same for both commands.

```
root@devapp:/# curl frontend.prod-a.svc.cluster.local
```

```
  <!DOCTYPE html>
  <html>
  <head>
  <title>Welcome to nginx!</title>
  <style>
  <output_omitted>
```

```
root@devapp:/# curl backend.prod-b.svc.cluster.local
```

(c) Test access to the outside world, using **ping** and any other commands you may like. Test **ping** to another pod via its service.

```
root@devapp:/# ping -c3 www.linux.com
```

```
  ING www.linux.com (23.185.0.3) 56(84) bytes of data.
  64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=1 ttl=54 time=11.3 ms
  64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=2 ttl=54 time=11.2 ms
  64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=3 ttl=54 time=11.8 ms

  --- www.linux.com ping statistics ---
  3 packets transmitted, 3 received, 0% packet loss, time 4ms
  rtt min/avg/max/mdev = 11.241/11.459/11.809/0.278 ms
```

```
root@devapp:/# ping -c3 frontend.prod-a.svc.cluster.local
```

```
  PING frontend.prod-a.svc.cluster.local (10.98.182.27) 56(84) bytes of data.

  --- frontend.prod-a.svc.cluster.local ping statistics ---
  3 packets transmitted, 0 received, 100% packet loss, time 2045ms
```

```
root@devapp:/# exit
```

4. Also test from outside the cluster using a local curl, or using the public IP and high port in a local browser.

```
student@cp:~$ curl ifconfig.io
```

```
34.68.4.238
```

```
local$ curl 34.68.4.238:30854
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
```

5. From our testing we can see that all traffic with a configured port is allowed. Now create a new network policy which denies all traffic in the `prod-a` namespace.

```
student@cp:~$ vim denyall.yaml
```

denyall.yaml

```
1 kind: NetworkPolicy
2 apiVersion: networking.k8s.io/v1
3 metadata:
4   name: prod-deny-all
5   namespace: prod-a
6 spec:
7   podSelector: {}
8   policyTypes:
9   - Ingress
```

6. Create the resource.

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

```
networkpolicy.networking.k8s.io/prod-deny-all created
```

7. First ingress test access from `devapp` using the same command we ran before.

```
student@cp:~$ kubectl -n dev-ns exec -it devapp -- bash
```

On Container

(a) Test with **curl** to the `frontend` service. It may take a while to time out, you may want to use **control-c** to exit the command.

```
root@devapp:/# curl frontend.prod-a.svc.cluster.local
curl: (28) Failed to connect to frontend.prod-a.svc.cluster.local port 80: Connection timed out
```

(b) Test access to the `backend`, to make sure it still works, as well as to the outside world. Then exit back to the node.

```
root@devapp:/# curl backend.prod-b.svc.cluster.local
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
<output_omitted>
```

```
root@devapp:/# ping -c3 linux.com
```

```
PING linux.com (23.185.0.3) 56(84) bytes of data.
64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=1 ttl=54 time=11.5 ms
64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=2 ttl=54 time=11.3 ms
64 bytes from 23.185.0.3 (23.185.0.3): icmp_seq=3 ttl=54 time=14.1 ms

--- linux.com ping statistics ---
3 packets transmitted, 3 received, 0% packet loss, time 2003ms
rtt min/avg/max/mdev = 11.300/12.308/14.128/1.289 ms
```

```
root@devapp:/# exit
```

8. Finally test from outside the cluster using curl, or a local browser. It should fail.

```
local$ curl 34.68.4.238:30854
```

9. Log back into the `frontend` pod and test network access. As no egress rules have been applied it all should work.

```
student@cp:~$ kubectl -n prod-a exec -it frontend -- bash
```

On Container

(a) Test access to the `devapp`, `backend`, and outside world. The commands are shown without output.

```
root@frontend:/# curl backend.prod-b.svc.cluster.local
```

```
root@frontend:/# ping -c3 linux.com
```

10. To complete the testing log into `backend` and test access to `frontend`, which should fail, after a timeout.

```
student@cp:~$ kubectl -n prod-b exec -it backend -- bash
```

On Container

```
root@backend:/# curl frontend.prod-a.svc.cluster.local
```

```
curl: (7) Failed to connect to frontend.prod-a.svc.cluster.local port 80: Connection timed
↪  out
```

11. Create a new network policy to allow ingress access for pods with a label of `app: front`, in the `prod-a` namespaces but no other pods.

```
student@cp:~$ vim allowinternet.yaml
```

allowinternet.yaml

```
1 apiVersion: networking.k8s.io/v1
2 kind: NetworkPolicy
3 metadata:
4   name: internet-access
```

```yaml
spec:
  podSelector:
    matchLabels:
      app: front
  policyTypes:
  - Ingress
  ingress:
  - {}
```

```
student@cp:~$ kubectl -n prod-a create -f allowinternet.yaml
```

```
networkpolicy.networking.k8s.io/internet-access created
```

12. Log into the `backend` pod and test access to the `frontend` pod.
13. Remove all of the network policies, in all namespaces. Then remove the pods and services we created in this lab.

Exercise 6.2: Configure an Ingress Controller

If you are unable to configure the missing service account, covered in previous content, you may consider using a Helm chart. A topic covered in the CKA course, or use the `Getting Started` topic of **Helm** https://helm.sh/docs/intro/install/#from-the-binary-releases and https://artifacthub.io

1. Begin by installing the HAProxy ingress software, which include security settings, a configmap, and a daemonset. No pods will run until a node has been labeled, you will need to edit you cp node and add the `role: ingress-controller:` label to the existing labels. You may see some errors in the `create` output as well.

```
student@cp:~$ kubectl edit node cp-XXYYZZ    #<<---- Change to your cp name
```

```yaml
....
    node-role.kubernetes.io/master: ""
    node.kubernetes.io/exclude-from-external-load-balancers: ""
    role: ingress-controller                          #<<<<----  Add this line
  name: cp-XXYYZ
  resourceVersion: "200423"
....
```

```
student@cp:~$ kubectl create -f https://haproxy-ingress.github.io/resources/haproxy-ingress.yaml
```

```
namespace/ingress-controller created
serviceaccount/ingress-controller created
Warning: rbac.authorization.k8s.io/v1beta1 ClusterRole is deprecated in v1.17+,
unavailable in v1.22+; use rbac.authorization.k8s.io/v1 ClusterRole
clusterrole.rbac.authorization.k8s.io/ingress-controller created
Warning: rbac.authorization.k8s.io/v1beta1 Role is deprecated in v1.17+,
unavailable in v1.22+; use rbac.authorization.k8s.io/v1 Role
role.rbac.authorization.k8s.io/ingress-controller created
Warning: rbac.authorization.k8s.io/v1beta1 ClusterRoleBinding is deprecated in v1.17+,
unavailable in v1.22+; use rbac.authorization.k8s.io/v1 ClusterRoleBinding
clusterrolebinding.rbac.authorization.k8s.io/ingress-controller created
Warning: rbac.authorization.k8s.io/v1beta1 RoleBinding is deprecated in v1.17+,
unavailable in v1.22+; use rbac.authorization.k8s.io/v1 RoleBinding
rolebinding.rbac.authorization.k8s.io/ingress-controller created
configmap/haproxy-ingress created
```

```
daemonset.apps/haproxy-ingress created
```

2. View the current daemonset and pod status.

```
student@cp:~$ kubectl -n ingress-controller get daemonset
```

```
NAME              DESIRED   CURRENT   READY   UP-TO-DATE   AVAILABLE   NODE SELECTOR
↪  AGE
haproxy-ingress   1         1         1       1            1           role=ingress-controller
↪  5m5s
```

```
student@cp:~$ kubectl -n ingress-controller get pod
```

```
NAME                    READY   STATUS    RESTARTS   AGE
haproxy-ingress-hxjvs   1/1     Running   0          2m44s
```

3. Create and expose a deployment for testing.

```
student@cp:~$ kubectl create deployment tester --image nginx:alpine
```

```
deployment.apps/tester created
```

```
student@cp:~$ kubectl expose deployment tester --port=80
```

```
service/nginx exposed
```

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

```
NAME                      READY   STATUS    RESTARTS   AGE
tester-565785f75c-vgcjm   1/1     Running   0          12s
```

4. Find the main IP of the node. We will use this to test the ingress controller. You can use **hostname -i** or **ip** if that is unavailable.

```
student@cp:~$ hostname -i
10.128.0.57
```

```
student@cp:~$ ip add show ens4
```

```
2: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group default qlen 1000
    link/ether 42:01:0a:80:00:39 brd ff:ff:ff:ff:ff:ff
    inet 10.128.0.57/32 scope global dynamic ens4
       valid_lft 65653sec preferred_lft 65653sec
    inet6 fe80::4001:aff:fe80:39/64 scope link
       valid_lft forever preferred_lft forever
```

5. Now we can create the ingress object.

```
student@cp:~$ vim ha-proxy.yaml
```

ha-proxy.yaml

```
1 apiVersion: networking.k8s.io/v1
2 kind: Ingress
3 metadata:
4   name: tester
```

```
spec:
  rules:
  - host: nginx.10.128.0.57.nip.io   #<-- Edit this to by YOUR IP
    http:
      paths:
      - pathType: Prefix
        path: "/"
        backend:
          service:
            name: tester
            port:
              number: 80
```

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

```
Warning: extensions/v1beta1 Ingress is deprecated in v1.14+, unavailable in v1.22+; use networking.k8s.io/v1 Ingress
ingress.extensions/nginx created
```

```
student@cp:~$ kubectl get ingresses.networking.k8s.io
```

```
NAME    CLASS    HOSTS                       ADDRESS   PORTS   AGE
nginx   <none>   nginx.10.128.0.57.nip.io              80      18s
```

6. Watch the ingress pod for a minute. You may notice that it will crash over and over. Use the output of the pod logs, and information covered in a previous exercise to allow the pod to run. There is also a ”working” YAML file the course tarball with one of the ways to fix the issue.

7. Once the ingress pod runs without error you should be able to query it for traffic. Remember to use your IP address in the command, not the example below.

```
student@cp:~$ curl http://10.128.0.57 -H 'Host: nginx.10.128.0.57.nip.io'
```

```
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Exercise 6.3: Configure mTLS

We can use **Linkerd** to provide a service mesh, which also offers us mTLS on traffic.

1. Begin by downloading the `Edge` or most featured filled version instead of the most stable version.

```
student@cp:~$ wget https://run.linkerd.io/install-edge
```

```
--2020-09-16 04:06:19--  https://run.linkerd.io/install-edge
Resolving run.linkerd.io (run.linkerd.io)... 104.24.113.2, 172.67.223.146, 104.24.112.2, ...
Connecting to run.linkerd.io (run.linkerd.io)|104.24.113.2|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 3270 (3.2K) [application/octet-stream]
Saving to: 'install-edge'

install-edge            100%[=============================>]   3.19K  --.-KB/s    in 0s

2020-09-16 04:06:19 (36.9 MB/s) - 'install-edge' saved [3270/3270]
```

2. Edit to ensure the following version and installation root are in use. While there are newer versions, the newest Edge version may also have undocumented features, which may cause issues.

```
student@cp:~$ vim install-edge
```

```
....
LINKERD2_VERSION=${LINKERD2_VERSION:-edge-20.9.2}
INSTALLROOT=${INSTALLROOT:-"${HOME}/.linkerd2"}
....
```

3. Run the **install-edge** script. It should suggest a PATH variable to add, as well as some commands which may be useful when checking to see if the installation was successful.

```
student@cp:~$ sh install-edge
```

```
Downloading linkerd2-cli-edge-20.9.2-linux-amd64...
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   662  100   662    0     0   3412      0 --:--:-- --:--:-- --:--:--  3412
100 38.2M  100 38.2M    0     0  26.3M      0  0:00:01  0:00:01 --:--:-- 47.6M
Download complete!
a
Validating checksum...
Checksum valid.

Linkerd edge-20.9.2 was successfully installed


Add the linkerd CLI to your path with:

  export PATH=$PATH:/home/student/.linkerd2/bin

Now run:

  linkerd check --pre                     # validate that Linkerd can be installed
  linkerd install | kubectl apply -f -    # install the control plane into the 'linkerd' namespace
  linkerd check                           # validate everything worked!
  linkerd dashboard                       # launch the dashboard

Looking for more? Visit https://linkerd.io/2/next-steps
```

4. Set the PATH and then validate that the installation should work.

```
student@cp:~$ export PATH=$PATH:/home/student/.linkerd2/bin
student@cp:~$ linkerd check --pre
```

```
kubernetes-api
--------------
√ can initialize the client
√ can query the Kubernetes API

kubernetes-version
------------------
√ is running the minimum Kubernetes API version
√ is running the minimum kubectl version

pre-kubernetes-setup
--------------------
√ control plane namespace does not already exist
√ can create non-namespaced resources
√ can create ServiceAccounts
√ can create Services
√ can create Deployments
√ can create CronJobs
√ can create ConfigMaps
√ can create Secrets
√ can read Secrets
√ can read extension-apiserver-authentication configmap
√ no clock skew detected

pre-kubernetes-capability
-------------------------
√ has NET_ADMIN capability
√ has NET_RAW capability

linkerd-version
---------------
√ can determine the latest version
√ cli is up-to-date                #You may get a warning here. Which is okay.

Status check results are √
```

5. If all items are green check marks, except for the most recent version then proceed with the installation of Linkerd into the Kubernetes cluster.

```
student@cp:~$ linkerd install | kubectl apply -f -
```

```
namespace/linkerd created
clusterrole.rbac.authorization.k8s.io/linkerd-linkerd-identity created
clusterrolebinding.rbac.authorization.k8s.io/linkerd-linkerd-identity created
serviceaccount/linkerd-identity created
<output_omitted>
```

6. As there could be issues not checked by the pre-validation check, ensure the installation was successful. You should again receive all green checkmarks. Read through the tests. It may take a few minutes for the script to complete, and you may see several duplicate messages as it waits for pods to become available.

```
student@cp:~$ linkerd check
```

```
kubernetes-api
--------------
√ can initialize the client
√ can query the Kubernetes API

kubernetes-version
```

```
------------------
√ is running the minimum Kubernetes API version
√ is running the minimum kubectl version

linkerd-existence
-----------------
√ 'linkerd-config' config map exists
√ heartbeat ServiceAccount exist
√ control plane replica sets are ready
√ no unschedulable pods
√ controller pod is running
√ can initialize the client
√ can query the control plane API

linkerd-config
--------------
√ control plane Namespace exists
√ control plane ClusterRoles exist
√ control plane ClusterRoleBindings exist
√ control plane ServiceAccounts exist
√ control plane CustomResourceDefinitions exist
√ control plane MutatingWebhookConfigurations exist
√ control plane ValidatingWebhookConfigurations exist
√ control plane PodSecurityPolicies exist

linkerd-identity
----------------
√ certificate config is valid
√ trust anchors are using supported crypto algorithm
√ trust anchors are within their validity period
√ trust anchors are valid for at least 60 days
√ issuer cert is using supported crypto algorithm
√ issuer cert is within its validity period
√ issuer cert is valid for at least 60 days
√ issuer cert is issued by the trust anchor

linkerd-webhooks-and-apisvc-tls
-------------------------------
√ tap API server has valid cert
√ proxy-injector webhook has valid cert
√ sp-validator webhook has valid cert

linkerd-api
-----------
√ control plane pods are ready
√ control plane self-check
√ [kubernetes] control plane can talk to Kubernetes
√ [prometheus] control plane can talk to Prometheus
√ tap api service is running

linkerd-version
---------------
√ can determine the latest version
√ cli is up-to-date

control-plane-version
---------------------
√ control plane is up-to-date
√ control plane and cli versions match

linkerd-addons
--------------
√ 'linkerd-config-addons' config map exists
```

```
linkerd-prometheus
------------------
√ prometheus add-on service account exists
√ prometheus add-on config map exists
√ prometheus pod is running

linkerd-grafana
---------------
√ grafana add-on service account exists
√ grafana add-on config map exists
√ grafana pod is running

Status check results are √
```

7. View the status of all the newly created Linkerd pods. Check both the status, as well as the state of each container. You may have to label the nodes to allow the daemonset to run.

```
student@cp:~$ kubectl label nodes --all role=ingress-controller

student@cp:~$ kubectl -n linkerd get pod
```

```
NAME                                     READY   STATUS    RESTARTS   AGE
linkerd-controller-68bc8789cb-prspl      2/2     Running   0          5m30s
linkerd-destination-64f8d75697-544b4     2/2     Running   0          5m29s
linkerd-grafana-88556b8bd-62kbr          2/2     Running   0          5m28s
linkerd-identity-7bc7758cc6-5bmzg        2/2     Running   0          5m30s
linkerd-prometheus-78df669bd7-pckzd      2/2     Running   0          5m27s
linkerd-proxy-injector-55b5db76c7-clnz9  2/2     Running   0          5m29s
linkerd-sp-validator-5f76db989b-cpffw    2/2     Running   0          5m29s
linkerd-tap-c6947cd49-qk2bx              2/2     Running   0          5m28s
linkerd-web-7cb88c7bf-b48mm              2/2     Running   0          5m29s
```

8. Edit the linkerd-web deployment to allow more than only loopback as the `enforced-host`. We will remove all values, everything after the equal sign, but typically you would want to use your corporate domain, such as `-enforced-host=^dashboard\.Linux\.com$`

```
student@cp:~$ kubectl -n linkerd edit deployments.apps linkerd-web
```

```
deployment.apps/linkerd-web edited
```

9. Check to make sure both of the containers in the linkerd-web pod are running. If there is any problem, return to the previous step and make sure there is nothing after the equal sign for the enforced host setting.

```
student@cp:~$ kubectl -n linkerd get pod
```

```
NAME                                     READY   STATUS    RESTARTS   AGE
linkerd-controller-68bc8789cb-prspl      2/2     Running   0          8m56s
linkerd-destination-64f8d75697-544b4     2/2     Running   0          8m55s
linkerd-grafana-88556b8bd-62kbr          2/2     Running   0          8m54s
linkerd-identity-7bc7758cc6-5bmzg        2/2     Running   0          8m56s
linkerd-prometheus-78df669bd7-pckzd      2/2     Running   0          8m53s
linkerd-proxy-injector-55b5db76c7-clnz9  2/2     Running   0          8m55s
linkerd-sp-validator-5f76db989b-cpffw    2/2     Running   0          8m55s
linkerd-tap-c6947cd49-qk2bx              2/2     Running   0          8m54s
linkerd-web-5d546786b5-qh5nl             2/2     Running   0          2m9s
```

10. Start the dashboard as a background process. You should get some output telling you the dashboard is running, and the correct port to use.

```
student@cp:~$ linkerd dashboard &
```

```
[1] 3303

student@cp:~$ Linkerd dashboard available at:
http://localhost:50750
Grafana dashboard available at:
http://localhost:50750/grafana
Opening Linkerd dashboard in the default browser
Failed to open Linkerd dashboard automatically
Visit http://localhost:50750 in your browser to view the dashboard
```

11. Edit the `linkerd-web` service and change the service type to be `NodePort`.

```
student@cp:~$ kubectl -n linkerd edit svc linkerd-web
```

```
service/linkerd-web edited
```

12. Verify the service is working and view the high-numbered port in use. Yours may be different than the example below.

```
student@cp:~$ kubectl -n linkerd get svc
```

```
NAME                      TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                  AGE
linkerd-controller-api    ClusterIP   10.104.52.2      <none>        8085/TCP                 13m
linkerd-dst               ClusterIP   10.107.53.87     <none>        8086/TCP                 13m
linkerd-dst-headless      ClusterIP   None             <none>        8086/TCP                 13m
linkerd-grafana           ClusterIP   10.106.160.246   <none>        3000/TCP                 13m
linkerd-identity          ClusterIP   10.96.153.143    <none>        8080/TCP                 13m
linkerd-identity-headless ClusterIP   None             <none>        8080/TCP                 13m
linkerd-prometheus        ClusterIP   10.106.123.121   <none>        9090/TCP                 13m
linkerd-proxy-injector    ClusterIP   10.97.135.136    <none>        443/TCP                  13m
linkerd-sp-validator      ClusterIP   10.96.70.249     <none>        443/TCP                  13m
linkerd-tap               ClusterIP   10.96.51.66      <none>        8088/TCP,443/TCP         13m
linkerd-web               NodePort    10.106.197.144   <none>        8084:31150/TCP,9994:30902/TCP
↪ 13m
```

13. Open a browser locally to view the dashboard. Use the public IP, the one you use to SSH to your nodes, not an internal IP address, and the high numbered port as found in the previous output. The command should show you that IP address. Yours will be different from the one below.

```
student@cp:~$ curl ifconfig.io
```

```
34.70.213.59
```

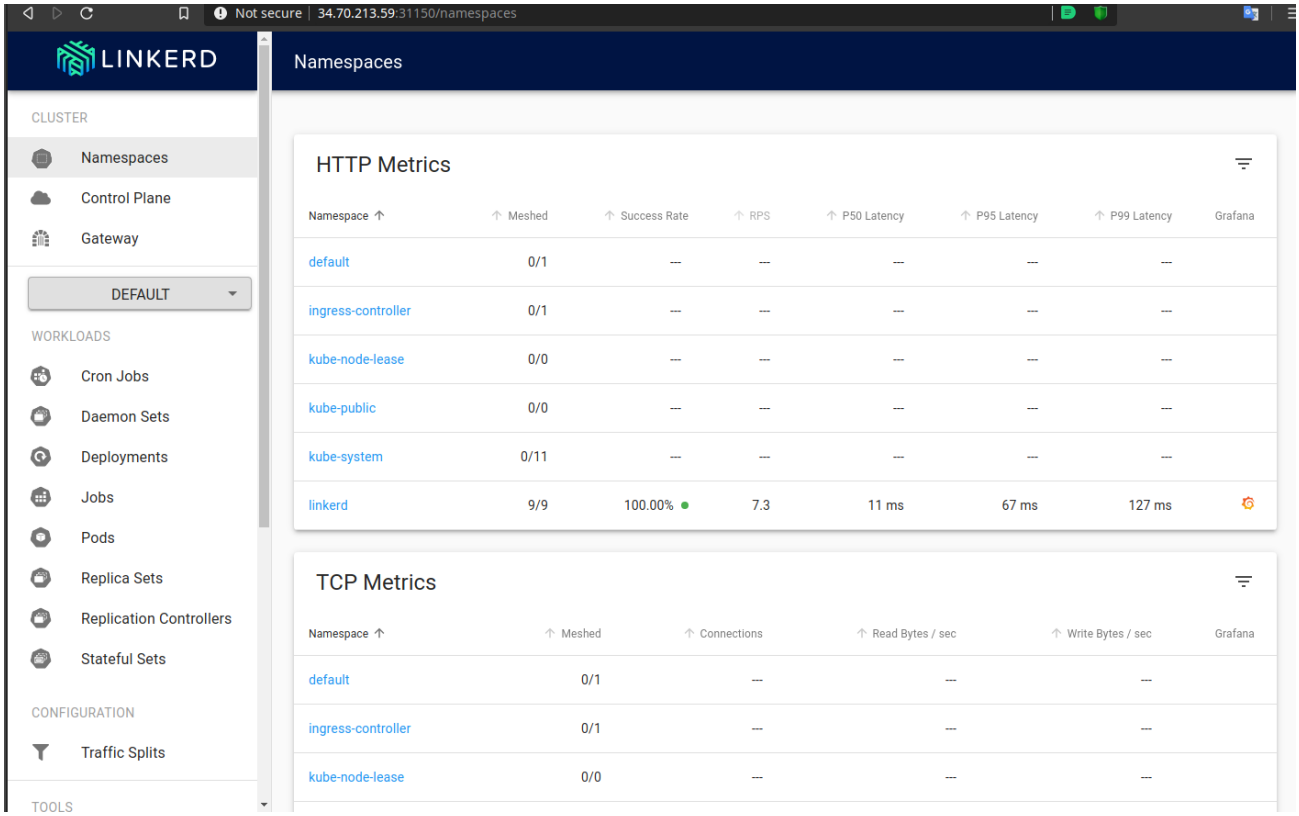


Figure 6.1: **Opening Linkerd Dashboard**

14. Return to the command line on the cp. We will create a new deployment which is enmeshed. We will first view a dry-run of creating the deployment, then view what is injected, and finally create the deployment.

```
student@cp:~$ kubectl create deployment test --image=nginx --dry-run=client -o yaml
```

```
  <output_omitted>
```

```
student@cp:~$ kubectl create deployment test --image=nginx --dry-run=client -o yaml | linkerd inject -
```

```
  ....
    template:
      metadata:
        annotations:
          linkerd.io/inject: enabled
        labels:
  ....
```

```
student@cp:~$ kubectl create deployment test --image=nginx --dry-run=client -o yaml | linkerd inject -\
        | kubectl create -f -
```

```
  deployment "test" injected

  deployment.apps/test created
```

15. After enmeshing the test deployment you can view the traffic, as well as `Grafana` page.

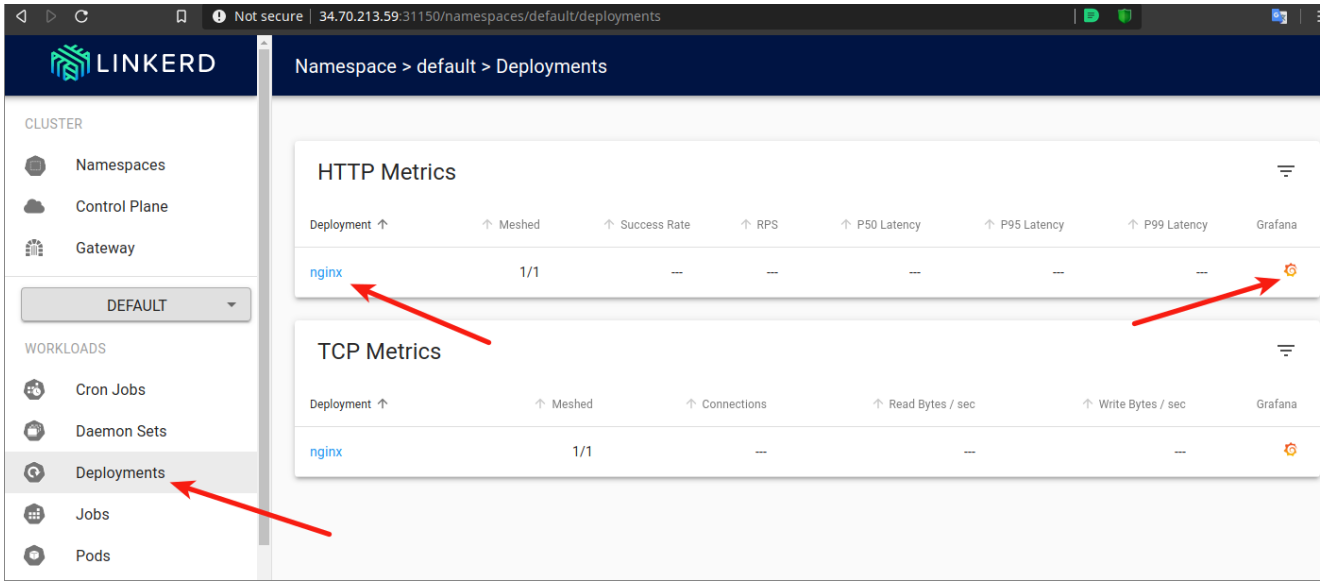


Figure 6.2: **Viewing Test Deployment**

16. With the time remaining expose the deployment and send traffic, both HTTP as well as other traffic and watch how the information is presented both via Linkerd as well as the Grafana page. Clean up the `text` deployment when done.

Chapter 7

Workload Considerations

7.1 Labs

Exercise 7.1: Using Docker to Scan Images

There are several tools available for scanning images. We will begin with **docker**, which you can use on multiple operating systems to check for security issues.

1. Clone the git repository and run the **docker-bench-security** tool against your cluster. There will be quite a bit of output, some of it similar to that from **kube-bench**, but focused on Docker instead. Scan through all the output, but take a close look at section `4 - Container Images and Build File` and `5 - Container Runtime`.

```
student@cp:~$ git clone https://github.com/docker/docker-bench-security.git
```

```
Cloning into 'docker-bench-security'...
remote: Enumerating objects: 23, done.
remote: Counting objects: 100% (23/23), done.
remote: Compressing objects: 100% (21/21), done.
remote: Total 2085 (delta 8), reused 6 (delta 2), pack-reused 2062
Receiving objects: 100% (2085/2085), 2.95 MiB | 16.87 MiB/s, done.
Resolving deltas: 100% (1455/1455), done.
```

```
student@cp:~$ cd docker-bench-security
student@cp:~$ sudo ./docker-bench-security.sh
```

```
[INFO] 1.2 - Linux Hosts Specific Configuration
WARNING: No swap limit support
[WARN] 1.2.1 - Ensure a separate partition for containers has been created (Scored)
[INFO] 1.2.2  - Ensure only trusted users are allowed to control Docker daemon (Scored)
[INFO]        * docker:x:116:
[WARN] 1.2.3  - Ensure auditing is configured for the Docker daemon (Scored)
[WARN] 1.2.4  - Ensure auditing is configured for Docker files and directories - /var/lib/docker
↪  (Scored)

....
[INFO] 4 - Container Images and Build File
[WARN] 4.1  - Ensure that a user for the container has been created (Scored)
[WARN]      * Running as root: k8s_kube-apiserver_kube-apiserver-cp_kube-system_32a2dd0...
[WARN]      * Running as root: k8s_POD_kube-apiserver-cp_kube-system_32a2dd06b850a9a3fc...
```

```
....

[INFO] 5 - Container Runtime
[WARN] 5.1  - Ensure that, if applicable, an AppArmor Profile is enabled (Scored)
[WARN]      * No AppArmorProfile Found: k8s_calico-node_calico-node-t8cvv_kube-system_bc8f442...
[WARN]      * No AppArmorProfile Found: k8s_kube-proxy_kube-proxy-4fl9l_kube-system_e5998ed4-...
[PASS] 5.2  - Ensure that, if applicable, SELinux security options are set (Scored)

....
```

Exercise 7.2: Image Analysis With Clair

1. Install some necessary software:

```
student@cp:~$ sudo apt-get install docker-compose -y
```

```
<output_omitted>
```

2. Use **git** to clone a handy repository. While the main **Clair** repository has similar contents, at the moment it generates some errors and has some hiccups. In this case Charlie Belmer, who has an interesting cloud security blog, has made easy to use files available:

```
student@cp:~$ git clone https://github.com/Charlie-belmer/Docker-security-example.git
```

```
Cloning into 'Docker-security-example'...
remote: Enumerating objects: 27, done.
remote: Total 27 (delta 0), reused 0 (delta 0), pack-reused 27
Unpacking objects: 100% (27/27), done.
```

3. Change into the newly created directory and then into the `clair` subdirectory:

```
student@cp:~$ cd Docker-security-example/
student@cp:~/Docker-security-example$ ls
```

```
LICENSE  README.md  clair  wazuh
```

```
student@cp:~/Docker-security-example$ cd clair/
student@cp:~/Docker-security-example/clair$ ls
```

```
docker-compose-data  docker-compose.yml
```

4. Ensure you have an `nginx` image downloaded to view in a following command.

```
student@cp:~/Docker-security-example/clair$ sudo docker pull nginx
```

```
<output_omitted>
```

5. This command will capture the current terminal window. Use **docker-compose up** to run the configuration found in the file, `docker-compose.yml`. You will open a second terminal or **PuTTY** session to the cp node in the next step. **DO NOT TYPE control-c until a later step**.

```
student@cp:~/Docker-security-example/clair$ sudo docker-compose up
```

```
Creating network "clair_default" with the default driver
Pulling postgres (postgres:9.6)...
9.6: Pulling from library/postgres
e50c3c9ef5a2: Pull complete
```

```
....

Creating clair_clairctl_1 ... done
Attaching to clair_postgres_1, clair_clair_1, clair_clairctl_1
postgres_1  | The files belonging to this database system will be owned by user "postgres".
postgres_1  | This user must also own the server process.
postgres_1  |
postgres_1  | The database cluster will be initialized with locale "en_US.utf8".
postgres_1  | The default database encoding has accordingly been set to "UTF8".
postgres_1  | The default text search configuration will be set to "english".
.....
<Ongoing_output>
```

6. **Open a second terminal** to the cp node using a terminal or another PuTTY session. Change to the clair subdirectory of the newly cloned git repo. Then execute the **clairctl** command with the **analyze** option. It will take about a minute to see output as the tool examines the layers of the image. Be aware as images change the number of vulnerabilities will change. Also when an image updates it may have none.

```
student@cp:~$ cd Docker-security-example/clair
student@cp:.../clair$ sudo docker-compose exec clairctl clairctl analyze -l nginx
```

```
Image: /nginx:latest
 5 layers found

  -> Analysis [6635deff86a3] found 92 vulnerabilities.
  -> Analysis [87e11ff4740b] found 92 vulnerabilities.
  -> Analysis [6121d6a75c8b] found 92 vulnerabilities.
  -> Analysis [614a9161e8d3] found 92 vulnerabilities.
  -> Analysis [12f575862bab] found 52 vulnerabilities.

cleaning temporary local repository: remove /tmp: device or resource busy
```

7. Use **docker pull** to get other images, such as `alpine` and `couchbase` in case `nginx` happens to have zero vulnerabilities. At the moment there are some hiccups when looking at versions other than `latest`.

8. Create an HTML report on a package with vulnerabilities.

```
student@cp:...clair$ sudo docker-compose exec clairctl clairctl report -l nginx
```

```
HTML report at /reports/html/analysis-nginx-latest.html
cleaning temporary local repository: remove /tmp: device or resource busy
```

9. Find where the file has been created so that you can download it to your local system and view it via a browser:

```
student@cp:...clair$ find $HOME -name analysis-nginx-latest.html
```

```
/home/student/Docker-security-example/clair/docker-compose-data/\
        clairctl-reports/html/analysis-nginx-latest.html
```

10. Exit back to your local node and pull down the file. If you are using PuTTY you may have to add a different program to get the file. The **scp** command would use your own pem key and the command would be on one line. It has been modified to wrap properly. Change the browser for whichever browser you prefer. Take a moment and look through the report.

```
student@cp:...clair$ exit
```

```
logout
Connection to 34.68.246.128 closed.
```

```
local$ scp -i LF-Class.pem \
  student@34.68.246.128:/home/student/Docker-security-example/clair/
  docker-compose-data/clairctl-reports/html/analysis-nginx-latest.html $HOME
```

```
  analysis-nginx-latest.html                    100%  152KB 307.0KB/s   00:00
```

```
local$ firefox analysis-nginx-latest.html
```

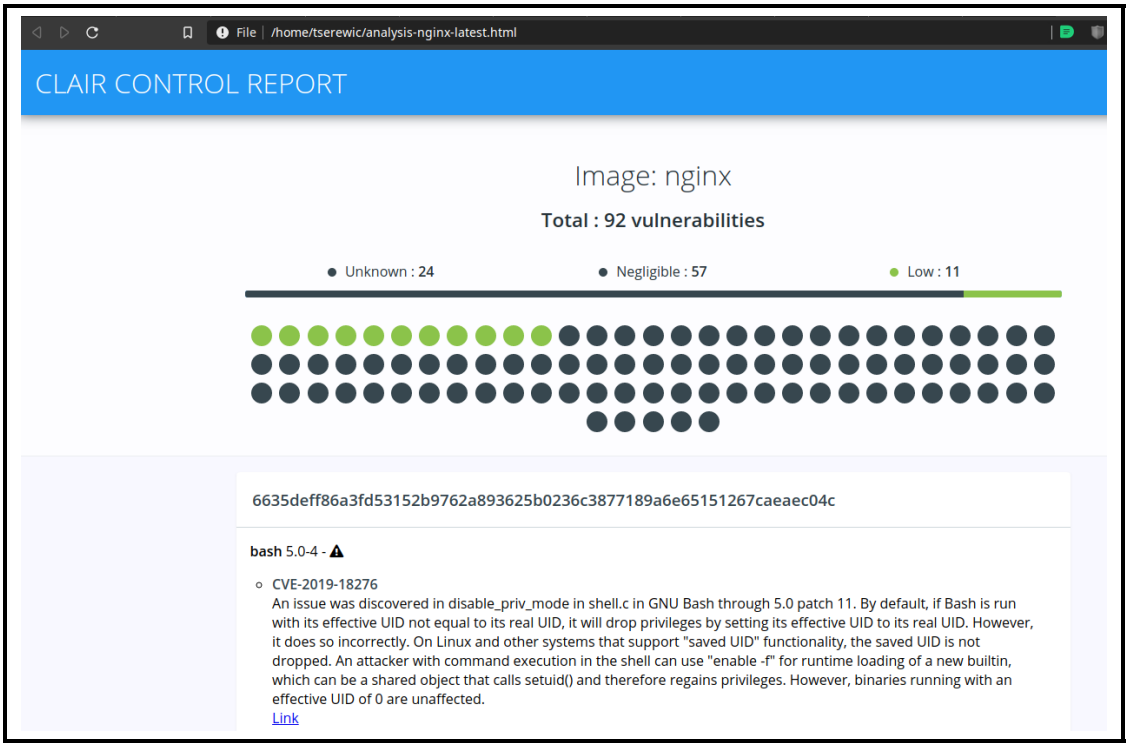


Figure 7.1: **HTML Clair Report**

11. **BACK ON THE FIRST TERMINAL**, you can now use **control-c** to stop **Clair** from running. You will use the second terminal again in a following section.

```
control-c
```

```
  ^CGracefully stopping... (press Ctrl+C again to force)
  Stopping clair_clairctl_1 ... done
  Stopping clair_clair_1    ... done
  Stopping clair_postgres_1 ... done
```

Exercise 7.3: Image Analysis With Trivy

1. Begin by adding a new repository for **Trivy**, and install the software.

```
student@cp:~$ sudo apt-get install wget apt-transport-https gnupg lsb-release -y
```

```
  Reading package lists... Done
  Building dependency tree
  Reading state information... Done
  <output_omitted>
```

```
student@cp:~$ wget -qO - https://aquasecurity.github.io/trivy-repo/deb/public.key | sudo apt-key add -
```

```
OK
```

```
student@cp:~$ echo deb https://aquasecurity.github.io/trivy-repo/deb $(lsb_release -sc) main | \
  sudo tee -a /etc/apt/sources.list.d/trivy.list
```

```
deb https://aquasecurity.github.io/trivy-repo/deb bionic main
```

```
student@cp:~$ sudo apt-get update
```

```
<output_omitted>
```

```
student@cp:~$ sudo apt-get install trivy
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  trivy
0 upgraded, 1 newly installed, 0 to remove and 6 not upgraded.
Need to get 9163 kB of archives.
After this operation, 26.3 MB of additional disk space will be used.
Get:1 https://aquasecurity.github.io/trivy-repo/deb bionic/main amd64 trivy amd64 0.14.0 [9163 kB]
Fetched 9163 kB in 1s (13.1 MB/s)
Selecting previously unselected package trivy.
(Reading database ... 65906 files and directories currently installed.)
Preparing to unpack .../trivy_0.14.0_amd64.deb ...
Unpacking trivy (0.14.0) ...
Setting up trivy (0.14.0) ...
```

2. Pull down an image from hub.docker.com, we will use a common one such as **nginx**.

```
student@cp:~$ sudo docker pull nginx
```

```
Using default tag: latest
latest: Pulling from library/nginx
6ec7b7d162b2: Pull complete
cb420a90068e: Pull complete
2766c0bf2b07: Pull complete
e05167b6a99d: Pull complete
70ac9d795e79: Pull complete
Digest: sha256:4cf620a5c81390ee209398ecc18e5fb9dd0f5155cd82adcbae532fec94006fb9
Status: Downloaded newer image for nginx:latest
docker.io/library/nginx:latest
```

3. Create a local tar file to interrogate, in case you are getting images from other sources such as developers prior to uploading to a repository.

```
student@cp:~$ sudo docker save nginx -o nginx.tar
```

4. Use **trivy** to check known vulnerabilities. There will be a lot of output.

```
student@cp:~$ sudo trivy -i nginx.tar
```

```
2020-12-19T00:17:54.538Z        INFO        Detecting Debian vulnerabilities...
2020-12-19T00:17:54.568Z        INFO        Trivy skips scanning programming
language libraries because no supported file was detected

nginx.tar (debian 10.7)
=======================
Total: 147 (UNKNOWN: 3, LOW: 107, MEDIUM: 6, HIGH: 30, CRITICAL: 1)
```

```
+------------------+--------------------+----------+-------------------------

|     LIBRARY      |  VULNERABILITY ID   | SEVERITY |     INSTALLED VERSION
 | FIXED VERSION |                            TITLE
    |                           URL                          |
<output_omitted>
```

5. From the report above we can see there is a CRITICAL issue in the package. Use **grep** to find the issue, then look it up on https://nvd.nist.gov, to find the exploit and possible fix.

```
student@cp:~$ sudo trivy -i nginx.tar  | grep CRITICAL
```

```
Total: 147 (UNKNOWN: 3, LOW: 107, MEDIUM: 6, HIGH: 30, CRITICAL: 1)
| libbsd0          | CVE-2019-20367      | CRITICAL | 0.9.1-2
    |               | nlist.c in libbsd before
       | avd.aquasec.com/nvd/cve-2019-20367         |
```

6. Now check an older version of the package, and use the built in filters of **trivy**.

```
student@cp:~$ sudo docker pull nginx:1.9.1
```

```
<output_omitted>
```

```
student@cp:~$ sudo trivy image --severity CRITICAL nginx:1.9.1
```

```
2020-12-19T00:37:14.640Z        INFO        Detecting Debian vulnerabilities...
2020-12-19T00:37:14.654Z        INFO        Trivy skips scanning programming
language libraries because no supported file was detected
2020-12-19T00:37:14.654Z        WARN        This OS version is no longer
supported by the distribution: debian 8.1
2020-12-19T00:37:14.654Z        WARN        The vulnerability detection may be
insufficient because security updates are not provided

nginx:1.9.1 (debian 8.1)
========================
Total: 54 (CRITICAL: 54)

+------------------+-----------------+----------+-------------------+------
-----------------+--------------------------------+-------------------------
----------+
|      LIBRARY      | VULNERABILITY ID | SEVERITY | INSTALLED VERSION |
FIXED VERSION     |              TITLE              |                URL
+------------------+-----------------+----------+-------------------+------
-----------------+--------------------------------+-------------------------
----------+
| libbz2-1.0        | CVE-2019-12900   | CRITICAL | 1.0.6-7           |
1.0.6-7+deb8u1        | bzip2: out-of-bounds write in  |
avd.aquasec.com/nvd/cve-2019-12900 |
                       | function BZ2_decompress        |                |
<output_omitted>
```

Exercise 7.4: Use Tracee to Scan Running Container

Tracee allows for dynamic tracing of a running container. We will use two terminal sessions on the cp for this lab. As well we will pin the particular version of Tracee to run as it updates on a regular basis and has changed dramatically with each version, so far.

1. View the help output for the **tracee** command. You will need to pass three volumes for necessary kernel files and output

location. The first time that you use the image it will be pulled from hub.docker.com. Pin the image used to version 0.4.0.

```
student@cp:~$ sudo docker run --name tracee --rm --privileged --pid=host \
 -v /lib/modules/:/lib/modules/:ro -v /usr/src:/usr/src:ro -v /tmp/tracee:/tmp/tracee \
 aquasec/tracee:0.4.0 -h
```

```
   NAME:
      Tracee - Trace OS events and syscalls using eBPF

   USAGE:
      tracee [global options] command [command options] [arguments...]

   VERSION:
      v0.4.0
   <output_omitted>
```

2. Start running **tracee** and look for newly created containers. You won't see any traffic at first. Leave the command running to capture the terminal. You'll run the following command in a second terminal window.

```
student@cp:~$ sudo docker run --name tracee --rm --privileged --pid=host \
 -v /lib/modules/:/lib/modules/:ro -v /usr/src:/usr/src:ro -v /tmp/tracee:/tmp/tracee \
 aquasec/tracee:0.4.0 --trace container=new
```

3. **In a second terminal session on the cp** create a new deployment with a unique name like special to ensure we recognize what is being traced.

```
student@cp:~$ kubectl create deployment special --image=nginx
```

```
   deployment.apps/special created
```

4. To make tracing of IO to the pod easier we will ensure that a replica is running on the current node. If there isn't one, an easy solution is to scale up.

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

```
   NAME                       READY   STATUS    RESTARTS   AGE   IP
             NODE     NOMINATED NODE   READINESS GATES
   special-7497d5bcfc-drddz   1/1     Running   0          15s   192.168.171.79
      worker   <none>           <none>
```

5. As the existing pod is on the other node in the above example, you may not need to do the following step. There are a few ways to manage where the pod is deployed. Take note of the ephemeral IP of the local pod.

```
student@cp:~$ kubectl scale deployment special --replicas=2
```

```
   deployment.apps/special scaled
```

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

```
   NAME                       READY   STATUS    RESTARTS   AGE   IP
     NODE     NOMINATED NODE   READINESS GATES
   special-7497d5bcfc-drddz   1/1     Running   0          34s   192.168.171.79
     worker   <none>           <none>
   special-7497d5bcfc-qhvr9   1/1     Running   0          4s    192.168.219.72
     cp   <none>           <none>
```

6. From the second window generate traffic to the `special` deployment pod. Check for new output in the first window. Then log into the pod and you should see details of each command

```
student@cp:~$ curl 192.168.219.72
```

```
<!DOCTYPE html>
<html>
<output_omitted>
```

```
student@cp:~$ kubectl exec -it special-7497d5bcfc-qhvr9 -- bash
root@special-7497d5bcfc-qhvr9:/# ls
```

```
<!DOCTYPE html>
<html>
<output_omitted>
```

7. Return to the **first terminal**. You should see output from the traced pod.

```
TIME(s)        UTS_NAME         UID    COMM             PID/host        TID/host        RET
EVENT               ARGS
61637.939777   special-7497d5b  0      bash             38     /32358   38     /32358   46
clone               flags: CLONE_CHILD_CLEARTID|CLONE_CHILD_SETTID, stack: 0x0, parent_tid: 0x0,
↪  child_tid: 0x7FA6CB2C87BE, tls: 0
61637.940689   special-7497d5b  0      bash             38     /32358   38     /32358   0
close               fd: 3
61637.941973   special-7497d5b  0      bash             38     /32358   38     /32358   0
close               fd: 4
61637.943682   special-7497d5b  0      bash             46     /619     46     /619     0
execve              pathname: /bin/ls, argv: [ls]
61637.945160   special-7497d5b  0      bash             46     /619     46     /619     0
security_bprm_check  pathname: /bin/ls, dev: 230, inode: 779638
61637.946548   special-7497d5b  0      ls               46     /619     46     /619     -2
access              pathname: /etc/ld.so.preload, mode: R_OK
61637.946644   special-7497d5b  0      ls               46     /619     46     /619     0
security_file_open   pathname: /etc/ld.so.cache, flags: O_RDONLY|O_LARGEFILE, dev: 230, inode:
↪  784361
<output_omitted>
```

8. Leave the trace running, perhaps with some blank space to see new lines added. On **the second terminal**, scale the deployment to zero, then back up to enough replicas that one runs locally. You should see more traffic in the first terminal.

```
student@cp:~$ kubectl scale deployment special --replicas=10
```

```
deployment.apps/special scaled
```

```
student@cp:~$ kubectl scale deployment special --replicas=2
```

```
deployment.apps/special scaled
```

9. **CHALLENGE STEP:** Save the output of the trace to a file of the entire process of nginx being created, a page request and shutdown.

Exercise 7.5: Using Falco to Monitor Audit Events

To further track and monitor cluster activity we will deploy **Falco**. Using an audit webhook will collect data from the **kube-apiserver** audit facility.

1. Begin by installing the package key for the **Falco** repository.

```
student@cp:~$ curl -s https://falco.org/repo/falcosecurity-3672BA8F.asc | sudo apt-key add -
```

```
OK
```

2. Next we will add the **Falco** repository for APT. You could edit the file by hand or run the command on a single line without the backslash.

```
student@cp:~$ echo "deb https://download.falco.org/packages/deb stable main" | \
  sudo tee -a /etc/apt/sources.list.d/falcosecurity.list
```

```
deb https://download.falco.org/packages/deb stable main
```

3. Update the repository information.

```
student@cp:~$ sudo apt-get update -y
```

```
<output_omitted>
```

4. During the **Falco** installation modules are created. Install software to ensure that it works. Your operating system could already have this package installed.

```
student@cp:~$ sudo apt-get -y install linux-headers-$(uname -r)
```

```
<output_omitted>
```

5. Install the **Falco** software. The installation process will take a few minutes, and generate a lot of output, as dependencies are installed and configured.

```
tudent@cp:~$ sudo apt-get install -y falco
```

```
<output_omitted>
Secure Boot not enabled on this system.
Done.

falco:
Running module version sanity check.
 - Original module
   - No original module exists within this kernel
 - Installation
   - Installing to /lib/modules/5.4.0-1029-gcp/updates/dkms/

depmod....

DKMS: install completed.
Setting up build-essential (12.4ubuntu1) ...
Processing triggers for systemd (237-3ubuntu10.43) ...
Processing triggers for man-db (2.8.3-2ubuntu0.1) ...
Processing triggers for ureadahead (0.100.0-21) ...
Processing triggers for libc-bin (2.27-3ubuntu1.4) ...
```

6. Enable and start **Falco**. Ensure that it is running without errors. Note the initialization information provided and that an internal webserver is also running.

```
student@cp:~$ sudo systemctl enable falco
```

```
falco.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable falco
```

```
student@cp2:~$ sudo systemctl start falco
```

```
student@cp:~$ sudo systemctl status falco
```

```
- falco.service - LSB: Falco syscall activity monitoring agent
   Loaded: loaded (/etc/init.d/falco; generated)
   Active: active (running) since Wed 2020-12-30 06:41:31 UTC; 5s ago
     Docs: man:systemd-sysv-generator(8)
  Process: 11878 ExecStart=/etc/init.d/falco start (code=exited, status=0/SUCCES
    Tasks: 10 (limit: 4915)
   CGroup: /system.slice/falco.service
           |- 11947 /usr/bin/falco --daemon --pidfile=/var/run/falco.pid

Dec 30 06:41:30 cp2 falco[11878]: Wed Dec 30 06:41:30 2020: Falco initialize
Dec 30 06:41:30 cp2 falco[11878]: Wed Dec 30 06:41:30 2020: Loading rules fr
Dec 30 06:41:30 cp2 falco[11897]: Loading rules from file /etc/falco/falco_r
Dec 30 06:41:30 cp2 falco[11897]: Loading rules from file /etc/falco/falco_r
Dec 30 06:41:30 cp2 falco[11878]: Wed Dec 30 06:41:30 2020: Loading rules fr
Dec 30 06:41:30 cp2 falco[11897]: Loading rules from file /etc/falco/k8s_aud
Dec 30 06:41:30 cp2 falco[11878]: Wed Dec 30 06:41:30 2020: Loading rules fr
Dec 30 06:41:31 cp2 systemd[1]: Started LSB: Falco syscall activity monitori
Dec 30 06:41:31 cp2 falco[11947]: Starting internal webserver, listening on
Dec 30 06:41:31 cp2 falco[11947]: 06:41:31.172437000: Notice Privileged cont
```

7. Read through the main configuration file for **Falco**. Note the section on the order rules files are read, that syslog is enabled by default (which will be seen via **journalctl**, and that the default port and location **Falco** will look for audit information.

```
student@cp:~$ sudo less /etc/falco/falco.yaml
```

```
....
# The files will be read in the order presented here, so make sure if
# you have overrides they appear in later files.
rules_file:
  - /etc/falco/falco_rules.yaml
  - /etc/falco/falco_rules.local.yaml
  - /etc/falco/k8s_audit_rules.yaml
  - /etc/falco/rules.d
....
# The ssl_certificate is a combination SSL Certificate and corresponding
# key contained in a single file. You can generate a key/cert as follows:
#
# $ openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
# $ cat certificate.pem key.pem > falco.pem
# $ sudo cp falco.pem /etc/falco/falco.pem

webserver:
  enabled: true
  listen_port: 8765
  k8s_audit_endpoint: /k8s-audit
  ssl_enabled: false
  ssl_certificate: /etc/falco/falco.pem
```

8. Take a look at the mentioned rules files, including `/etc/falco/k8s_audit_rules.yaml`. There are several hundred lines, most with comments. Notice the `source: k8s_audit` entry, among others.

```
student@cp:~$ sudo less /etc/falco/k8s_audit_rules.yaml
```

```
....
# If you wish to restrict activity to a specific set of users, override/append to this list.
# users created by kops are included
- list: vertical_pod_autoscaler_users
  items: ["vpa-recommender", "vpa-updater"]
```

```
- list: allowed_k8s_users
  items: [
    "minikube", "minikube-user", "kubelet", "kops", "admin", "kube", "kube-proxy",
    ↪  "kube-apiserver-healthcheck",
    "kubernetes-admin",
    vertical_pod_autoscaler_users,
    cluster-autoscaler,
    "system:addon-manager"
    ]
....
```

9. Follow the steps in documentation to generate a new SSL certificate and key. Feel free to enter your information or press enter to use default values during key generation.

```
student@cp:~$ openssl req -newkey rsa:2048 -nodes -keyout key.pem -x509 -days 365 -out certificate.pem
```

```
an't load /home/student/.rnd into RNG
139934645899712:error:2406F079:random number generator:RAND_load_file:Cannot open file:..
/crypto/rand/randfile.c:88:Filename=/home/student/.rnd
Generating a RSA private key
.....................................+++++
.....................+++++
writing new private key to 'key.pem'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
<output_omitted>
```

```
student@cp:~$ cat certificate.pem key.pem > falco.pem
```

```
student@cp:~$ sudo cp falco.pem /etc/falco/falco.pem
```

10. Restart **Falco** and ensure it returns to active status.

```
student@cp:~$ sudo systemctl restart falco.service
```

```
student@cp:~$ sudo systemctl status falco.service
```

```
- falco.service - LSB: Falco syscall activity monitoring agent
   Loaded: loaded (/etc/init.d/falco; generated)
   Active: active (running) since Wed 2020-12-30 06:44:59 UTC; 6s ago
<output_omitted>
```

11. Open a second PuTTY or terminal session to your cp node. We will use the second terminal to view messages as they are created. Leave the terminal up as we will use it for several commands. The command will capture the terminal. We will close it in a few commands.

```
second$ sudo journalctl -u falco.service -f
```

```
Dec 30 06:44:59 cp2 falco[15123]: Loading rules from file /etc/falco/falco_...
Dec 30 06:44:59 cp2 falco[15104]: Wed Dec 30 06:44:59 2020: Loading rules f...
Dec 30 06:44:59 cp2 falco[15123]: Loading rules from file /etc/falco/k8s_au...
Dec 30 06:44:59 cp2 falco[15104]: Wed Dec 30 06:44:59 2020: Loading rules f...
Dec 30 06:44:59 cp2 systemd[1]: Started LSB: Falco syscall activity monitor...
Dec 30 06:44:59 cp2 falco[15153]: Starting internal webserver, listening on...
Dec 30 06:44:59 cp2 falco[15153]: 06:44:59.648360000: Notice Privileged container started
(user=<NA> user_loginuid=0 command=container:84deda667de4
```

```
k8s_calico-node_calico-node-8tsqz_kube-system_99bb6e07-f540-4059-8f30-605c73bc6e
40_0 (id=84deda667de4) image=calico/node:v3.17.1)
```

12. Back on the first terminal session copy a sensitive file, and then open the `kube-apiserver.yaml` for editing. You should see a message when the file is opened and when the file is saved and closed.

    The response information shown will be in the second terminal window. Some information may be in green, some in red font.

```
student@cp2:~$ sudo cat /etc/shadow > /dev/null

student@cp2:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
Dec 30 06:52:59 cp falco[15153]: 06:52:59.174715196: Warning Sensitive file opened
for reading by non-trusted program (user=root user_loginuid=1001 program=cat
command=cat /etc/shadow file=/etc/shadow parent=sudo gparent=bash
ggparent=sshd gggparent=sshd container_id=host image=<NA>)

Dec 30 06:54:15 cp falco[15153]: 06:54:15.603342585: Error File below /etc
opened for writing (user=root user_loginuid=1001 command=vim
/etc/kubernetes/manifests/kube-apiserver.yaml parent=sudo pcmdline=sudo vim
/etc/kubernetes/manifests/kube-apiserver.yaml file=/etc/kubernetes/manifests/4913
program=vim gparent=bash ggparent=sshd gggparent=sshd container_id=host image=<NA>)
```

13. To determine if API events are being monitored, create and delete a simple deployment and a namespace. You should not see corresponding messages in the second terminal.

```
student@cp:~$ kubectl create deploy test --image=nginx
```

```
deployment.apps/test created
```

```
student@cp:~$ kubectl delete deploy test
```

```
deployment.apps "test" deleted
```

```
student@cp:~$ kubectl create ns test-ns
```

```
namespace/test-ns created
```

```
student@cp:~$ kubectl delete ns test-ns
```

```
namespace "test-ns" deleted
```

14. Now that we are sure that Falco is monitoring the operating system, let us add our own rules for Kubernetes. To begin, ensure that API server auditing has been enabled. Check back to the previous lab "Enabling API Server Auditing", which is part of the Securing the API Server chapter exercise.

    In case you had experimented with auditing, use a typical simple audit file and replace the one in use. An example has been included in the tarball called `falco-audit.yaml`. Use **find** and whatever path is returned, which may be different than what is found in the command below, to copy the file to the `/etc/kubernetes` directory.

```
student@cp:~$ sudo find $HOME -name falco-audit.yaml
```

```
/home/student/LFS460/SOLUTIONS/s_07/falco-audit.yaml     #Your path may be different
```

```
student@cp:~$ cp <path from find command> /etc/kubernetes/
```

15. Create a new admission controller webhook file. Edit the file to use IP address of the server.

```
student@cp:~$ sudo vim /etc/kubernetes/audit-webhook-kubeconfig
```

```yaml
apiVersion: v1
kind: Config
clusters:
- cluster:
    server: http://10.128.0.47:8765/k8s-audit   #<-- Edit to by YOUR IP
  name: falco
contexts:
- context:
    cluster: falco
    user: ""
  name: default-context
current-context: default-context
preferences: {}
users: []
```

16. Add the new webhook file to the `kube-apiserver.yaml` manifest. You will need to pass a new parameter and add the file volume as well. Also edit the file using the entry for your previous audit file and change the file name to the new `falco-audit.yaml` file, which is three areas to edit for the file declaration, `volumeMounts` and `volumes`.

```
student@cp:~$ sudo vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```yaml
....
    - --audit-log-path=/var/log/audit.log
    - --audit-policy-file=/etc/kubernetes/falco-audit.yaml                #<-- Add this entry
    - --audit-webhook-config-file=/etc/kubernetes/audit-webhook-kubeconfig #<-- and this entry
    - --authorization-mode=Node,RBAC
....
volumeMounts:
    - mountPath: /etc/kubernetes/audit-webhook-kubeconfig               #<-- Add these
      name: webhook
      readOnly: true
....
volumes:
  - hostPath:                                                           #<-- And these
      path: /etc/kubernetes/audit-webhook-kubeconfig
      type: File
    name: webhook
...
```

17. Ensure the API server restarted after the edit. You should see two entries, a few seconds old.

```
student@cp:~$ sudo docker ps |grep apiserver
```

```
e9387fe49b09        1b74e93ece2f              "kube-apiserver --ad..."   3 seconds ago
Up 2 seconds
k8s_kube-apiserver_kube-apiserver-cp2_kube-system_38080cbf3010f40598f263245a76721e_0
52454e6609b6        k8s.gcr.io/pause:3.2      "/pause"                   3 seconds ago
Up 2 seconds
k8s_POD_kube-apiserver-cp2_kube-system_38080cbf3010f40598f263245a76721e_0
```

18. Create the same deployment and namespaces as before. You should see new errors show up in the second terminal still running **journalctl**. It may take a few seconds for the message to arrive.

```
student@cp2:~$ kubectl create ns test-ns
```

```
namespace/test-ns created
```

In the second terminal:

```
Dec 30 08:44:19 cp falco[15153]: 08:44:08.961896960: Warning Disallowed
namespace created (user=kubernetes-admin ns=test-ns)
Dec 30 08:44:19 cp falco[15153]: 08:44:08.988226048: Notice K8s Serviceaccount
Created (user=system:serviceaccount:kube-system:service-account-controller
user=default ns=test-ns resp=201 decision=allow reason=RBAC: allowed by
ClusterRoleBinding "system:controller:service-account-controller" of
ClusterRole "system:controller:service-account-controller" to
ServiceAccount "service-account-controller/kube-system")
```

19. Reference the k8s_audit_rules.yaml file again. Create objects both on the list and not, and determine if each are being reported via **Falco**.

Exercise 7.6: Working with AppArmor Profiles

1. As AppArmor is typically enabled in **Ubuntu**, we will install AppArmor utilities and dependencies:

```
student@cp:~$ sudo apt-get install apparmor-utils -y
```

```
<output_omitted>
```

2. Begin by looking at which profiles are loaded on the current node, which are in `enforce mode`, and which are `complain mode`:

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

```
apparmor module is loaded.
25 profiles are loaded.
20 profiles are in enforce mode.
   /sbin/dhclient
   /snap/snapd/10492/usr/lib/snapd/snap-confine
   /snap/snapd/10492/usr/lib/snapd/snap-confine//mount-namespace-capture-helper
   /usr/bin/lxc-start
   /usr/bin/man
<output_omitted>
```

3. For the next few steps we will use **two terminal sessions**. For ease of use the prompt will show which node instead of the typical `user@node:directory`. Run **aa-genprof** and pass it the full path of the binary. This is so it knows what program to track. The output will stop giving you an option to **S**can or **F**inish. Don't type either, continue to the next step.

```
one$  sudo aa-genprof /usr/bin/kubectl
```

```
Writing updated profile for /usr/bin/kubectl.
Setting /usr/bin/kubectl to complain mode.

Before you begin, you may wish to check if a
profile already exists for the application you
wish to confine. See the following wiki page for
more information:
http://wiki.apparmor.net/index.php/Profiles

Profiling: /usr/bin/kubectl

Please start the application to be profiled in
another window and exercise its functionality now.
```

```
Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

[(S)can system log for AppArmor events] / (F)inish
```

4. In the second terminal window run a task, such as creating a new deployment. You will not see output in the first window yet.

```
two$ kubectl create deployment genprof --image=fluentd/fluent
```

```
deployment.apps/genprof created
```

5. Return to the first window and use **S** to scan for new profile information. After several lines of ouput you should be asked what to do with the new profile, path, and mode. Use **A** to allow.

```
S
```

```
Reading log entries from /var/log/syslog.
Updating AppArmor profiles in /etc/apparmor.d.
Complain-mode changes:

Profile:  /usr/bin/kubectl
Path:     /sys/kernel/mm/transparent_hugepage/hpage_pmd_size
New Mode: r
Severity: 4

 [1 - #include <abstractions/lxc/container-base>]
  2 - #include <abstractions/lxc/start-container>
  3 - /sys/kernel/mm/transparent_hugepage/hpage_pmd_size r,
(A)llow / [(D)eny] / (I)gnore / (G)lob / Glob with (E)xtension / (N)ew / Audi(t) / Abo(r)t /
↪  (F)inish
```

```
A
```

```
Adding #include <abstractions/lxc/container-base> to profile.
Deleted 1 previous matching profile entries.

= Changed Local Profiles =

The following local profiles were changed. Would you like to save them?

 [1 - /usr/bin/kubectl]
(S)ave Changes / Save Selec(t)ed Profile / [(V)iew Changes] / View Changes b/w (C)lean profiles /
↪  Abo(r)t
```

6. This time use the **V** option to view the changes. After scanning the output use **q** to return to the previous question.

```
V
```

```
--- /etc/apparmor.d/usr.bin.kubectl     2020-12-19 22:10:39.962062281 +0000
+++ /tmp/tmpuru9i7sr    2020-12-19 22:33:12.460860777 +0000
@@ -4,6 +4,7 @@
 /usr/bin/kubectl flags=(complain) {
   #include <abstractions/base>

-  /usr/bin/kubectl mr,
+
```

```
+  #include <abstractions/lxc/container-base>

 }
(END)
```

7. This time use **S**ave. It should return, ready to scan the next event.

```
S
```

```
Writing updated profile for /usr/bin/kubectl.

Profiling: /usr/bin/kubectl

Please start the application to be profiled in
another window and exercise its functionality now.

Once completed, select the "Scan" option below in
order to scan the system logs for AppArmor events.

For each AppArmor event, you will be given the
opportunity to choose whether the access should be
allowed or denied.

[(S)can system log for AppArmor events] / (F)inish
```

8. Return to the first window, and this time use **kubectl edit** to change the `replicas:` count to two. Depending on the current profiles you may see an entry having to do with `/usr/bin/vim.basic` in the output as well.

```
one$ kubectl edit deploy genprof
```

```
deployment.apps/genprof edited
```

9. Scan for changes again in the first window. As there would be no change to the profile, you should be quickly asked if you want to scan or finish again. The second time choose **F**inish. Review the questions

```
S
```

```
<output_omitted>
```

```
F
```

10. Delete the deployment. Then profile the **docker run** command as you run an `nginx` container.

```
one$ sudo aa-genprof /usr/bin/docker
```

```
<output_omitted>
```

```
two$ sudo docker run --name profile-nginx \
 -v /my/webpage:/usr/share/nginx/html:ro -d nginx
```

```
6567c1d29c556f12472749b1535c4fd2200321310c1d696d213b95d5ba674d6d
```

11. After you have scanned and saved the profile, stop and remove the container.

```
two$ sudo docker stop 6567c1d29c556f12472749b1535c4fd2200321310c1d696d213b95d5ba674d6d

two$ sudo docker rm 6567c1d29c556f12472749b1535c4fd2200321310c1d696d213b95d5ba674d6d
```

12. Change to the `/etc/apparmor.d/` directory. There should be two new files, one called `usr.bin.kubectl` and another `usr.bin.docker`. Take a look at each.

```
student@cp:~$ cd /etc/apparmor.d ; ls -l
```

```
total 96
drwxr-xr-x 5 root root  4096 Dec 11 09:30 abstractions
drwxr-xr-x 2 root root  4096 Dec 18 23:27 cache
drwxr-xr-x 2 root root  4096 Dec 11 09:27 disable
drwxr-xr-x 2 root root  4096 Apr 24  2018 force-complain
drwxr-xr-x 2 root root  4096 Dec 11 09:36 local
drwxr-xr-x 2 root root  4096 Dec 11 09:31 lxc
-rw-r--r-- 1 root root   198 Nov 23  2018 lxc-containers
-rw-r--r-- 1 root root  3194 Mar 26  2018 sbin.dhclient
drwxr-xr-x 5 root root  4096 Dec 11 09:30 tunables
-rw------- 1 root root   167 Dec 19 22:49 usr.bin.docker     #<-- Here
-rw------- 1 root root   185 Dec 19 22:35 usr.bin.kubectl    #<-- and here.
-rw-r--r-- 1 root root   125 Nov 23  2018 usr.bin.lxc-start
<output_omitted>
```

13. Take a look at a newly created profile. It should be similar to what you saw while running **aa-genprof**.

```
student@cp:/etc/apparmor.d$ sudo less usr.bin.kubectl
```

```
# Last Modified: Sat Dec 19 22:35:04 2020
#include <tunables/global>

/usr/bin/kubectl flags=(complain) {
  #include <abstractions/base>
  #include <abstractions/lxc/container-base>

}
```

14. Now look through a more complex profile, such as `usr.sbin.tcpdump`.

```
student@cp:/etc/apparmor.d$ sudo less usr.sbin.tcpdump
```

```
# vim:syntax=apparmor
#include <tunables/global>

/usr/sbin/tcpdump {
  #include <abstractions/base>
  #include <abstractions/nameservice>
  #include <abstractions/user-tmp>

  capability net_raw,
  capability setuid,
  capability setgid,
<output_omitted>
```

15. Disable the newly created **docker** profile.

```
student@cp:/etc/apparmor.d$ cd

student@cp:~$ sudo ln -s /etc/apparmor.d/usr.bin.docker /etc/apparmor.d/disable/
student@cp:~$ sudo apparmor_parser -R /etc/apparmor.d/usr.bin.docker
```

16. Check that the profile is not among enabled profiles.

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

```
apparmor module is loaded.
28 profiles are loaded.
20 profiles are in enforce mode.
....
```

17. Enable and verify the profile is again in use. Notice the count of loaded and enforce mode profiles changes, as well as the addition of the binary.

```
student@cp:~$ sudo rm /etc/apparmor.d/disable/usr.bin.docker
student@cp:~$ sudo apparmor_parser -r /etc/apparmor.d/usr.bin.docker
student@cp:~$ sudo apparmor_status
```

```
apparmor module is loaded.
29 profiles are loaded.
21 profiles are in enforce mode.
....
   /usr/bin/docker
....
```

18. Now make sure the `worker` node(s) have all the same commands and profiles as the `cp`.

19. **BONUS STEP 1:** If a developer emailed you a profile to use, could you properly add it to the cluster and test it was able to run the container?

20. **BONUS STEP 2:** If you are already quite familiar with profiles, take a look at using the **Bane** program.

Chapter 8

Issue Detection

8.1 Labs

Exercise 8.1: Getting Started With Suricata

1. Install **Suricata**. Reading and compiling the source code may be best, found here https://github.com/OISF/suricata. To quickly start using **Suricata** there are existing binaries. Be ready to press `Enter` after reading about what Suricata can provide. You may need to add the `software-properties-common` package in order to have the **add-apt-repository** command.

```
student@cp:~$ sudo add-apt-repository ppa:oisf/suricata-stable
```

```
 Suricata IDS/IPS/NSM stable packages
https://www.openinfosecfoundation.org/
https://planet.suricata-ids.org/
https://suricata-ids.org/

Suricata IDS/IPS/NSM - Suricata is a high performance Intrusion Detection
and Prevention System and Network Security Monitoring engine.

Open Source and owned by a community run non-profit foundation, the Open
Information Security Foundation (OISF). Suricata is developed by the OISF,
its supporting vendors and the community.

This Engine supports:

- Multi-Threading - provides for extremely fast and flexible operation on multi-core systems.

<output_omitted>

- Lua scripting

and many more great features -
https://suricata-ids.org/features/all-features/
 More info: https://launchpad.net/~oisf/+archive/ubuntu/suricata-stable
```

```
Press [ENTER] to continue or Ctrl-c to cancel adding it.
```

```
<output_omitted>
```

2. Add the **suricata** software. You may need to add **jq** as well, depending on the node image in use.

```
student@cp:~$ sudo apt-get install suricata jq -y
```

```
<output_omitted>
```

3. View the current information about the product.

```
student@cp:~$ sudo suricata --build-info
```

```
This is Suricata version 6.0.4 RELEASE
Features: NFQ PCAP_SET_BUFF AF_PACKET HAVE_PACKET_FANOUT LIBCAP_NG....
IMD support: none
Atomic intrinsics: 1 2 4 8 byte(s)
64-bits, Little-endian architecture
GCC version 9.3.0, C version 201112
compiled with _FORTIFY_SOURCE=2
L1 cache line size (CLS)=64
thread local storage method: _Thread_local
compiled with LibHTP v0.5.39, linked against LibHTP v0.5.39

Suricata Configuration:
  AF_PACKET support:                       yes
<output_omitted>
```

4. Ensure that Suricata is properly running.

```
student@cp:~$ sudo systemctl status suricata
```

```
- suricata.service - LSB: Next Generation IDS/IPS
   Loaded: loaded (/etc/init.d/suricata; generated)
   Active: active (exited) since Mon 2022-03-24 01:45:49 UTC; 14s ago
     Docs: man:systemd-sysv-generator(8)

Dec 21 01:45:49 cp systemd[1]: Starting LSB: Next Generation IDS/IPS...
Dec 21 01:45:49 cp suricata[22259]: Starting suricata in IDS (af-packet) mod
Dec 21 01:45:49 cp systemd[1]: Started LSB: Next Generation IDS/IPS.
```

5. Find the interface you would like to monitor. There are several choices, and a production environment may have a more complex configuration. For now, use the primary interface of the cp node. In the example below the primary interface would be `ens4`.

```
student@cp:~$ ip addr show
```

```
....
2: ens4: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc mq state UP group default qlen 1000
    link/ether 42:01:0a:80:00:2d brd ff:ff:ff:ff:ff:ff
    inet 10.128.0.45/32 scope global dynamic ens4
....
```

6. Work through the configuration file. Some settings have been pre-populated, such as `HOME_NET`. There are quite a few variables to see. While the file is almost two thousand lines long there are little over six hundred items which could be set. We will start with just a few to change or uncomment.

```
student@cp:~$ sudo vim /etc/suricata/suricata.yaml
```

/etc/suricata/suricata.yaml

```
....
# Linux high speed capture support
af-packet:
  - interface: ens4                                     #<-- Edit the interface
    # Number of receive threads. "auto" uses the number of cores
    #threads: auto
    # Default clusterid. AF_PACKET will load balance packets based on flow.
    cluster-id: 99                                      #<-- Check this one
....
    # to yes, the kernel will do the needed defragmentation before sending the packets.
    defrag: yes                                         #<-- Check this value
    # To use the ring feature of AF_PACKET, set 'use-mmap' to yes
    use-mmap: yes                                       #<-- Uncomment this value
    # Lock memory map to avoid it being swapped. Be careful that over
....
    # Don't use it in IPS or TAP mode as it causes severe latency
   tpacket-v3: yes                                      #<-- Uncomment this
    # Ring size will be computed with respect to "max-pending-packets" and number
....
```

7. There are a few interfaces mentioned in the yaml file. Now that we have changed one of the interface settings, change the rest of the `eth0` settings over to our primary interface `ens4`. Your interface may be different.

```
student@cp:~$ sudo sed -i s/eth0/ens4/g /etc/suricata/suricata.yaml
```

8. Update Suricata rules. It may take a bit to download all the files.

```
student@cp:~$ sudo suricata-update
```

```
24/3/2022 -- 02:03:59 - <Info> -- Using data-directory /var/lib/suricata.
24/3/2022 -- 02:03:59 - <Info> -- Using Suricata configuration /etc/suricata/suricata.yaml
24/3/2022 -- 02:03:59 - <Info> -- Using /etc/suricata/rules for Suricata provided rules.
24/3/2022 -- 02:03:59 - <Info> -- Found Suricata version 6.0.1 at /usr/bin/suricata.
24/3/2022 -- 02:03:59 - <Info> -- Loading /etc/suricata/suricata.yaml
24/3/2022 -- 02:03:59 - <Info> -- Disabling rules for protocol http2
<output_omitted>
```

9. View the newly added rules.

```
student@cp:~$ sudo ls -l /var/lib/suricata/rules
```

```
total 15500
-rw-r--r-- 1 root root     3207 Dec 21 02:04 classification.config
-rw-r--r-- 1 root root 15864786 Dec 21 02:04 suricata.rules
```

10. Restart the service, then take a look at the general log. The `-f` option will capture the terminal. After watching the output for a minute use **ctrl-c** to interrupt the tail command.

```
student@cp:~$ sudo systemctl restart suricata

student@cp:~$ sudo tail -f /var/log/suricata/suricata.log
```

```
....
24/3/2022 -- 20:16:10 - <Perf> - using shared mpm ctx' for udp-packet
24/3/2022 -- 20:16:10 - <Perf> - using shared mpm ctx' for other-ip
24/3/2022 -- 20:16:10 - <Info> - 25001 signatures processed. 1245 are IP-only rules, 4098 are ....
24/3/2022 -- 20:16:10 - <Config> - building signature grouping structure, stage 1: preprocessing
↪  rules... complete
24/3/2022 -- 20:16:10 - <Perf> - TCP toserver: 41 port groups, 40 unique SGH's, 1 copies
```

```
24/3/2022 -- 20:16:10 - <Perf> - TCP toclient: 21 port groups, 21 unique SGH's, 0 copies
....
```

11. View the current statistics. With the **tail** command continuing to run use a second terminal create and delete a deployment. Something to generate traffic. You should see the stats increase.

```
student@cp:~$ sudo tail -f /var/log/suricata/stats.log
```

```
flow.mgr.rows_maxlen                    | Total                    | 2
flow.mgr.flows_checked                  | Total                    | 4641
flow.mgr.flows_notimeout                | Total                    | 1457
flow.mgr.flows_timeout                  | Total                    | 3184
flow.mgr.flows_evicted                  | Total                    | 3184
flow.mgr.flows_evicted_needs_work       | Total                    | 2091
tcp.memuse                              | Total                    | 1146880
tcp.reassembly_memuse                   | Total                    | 466944
http.memuse                             | Total                    | 34950
flow.memuse                             | Total                    | 7394304
```

12. Now generate a known alert. In the first window run **tail -f** on /var/log/suricata/fast.log. Let it run and we will use a test URL in the following command.

```
student@cp:~$ sudo tail -f /var/log/suricata/fast.log
```

```
12/21/2020-02:08:17.234342  [**] [1:2013028:5] ET POLICY curl User-Agent Outbound [**]
[Classification:
 Attempted Information Leak] [Priority: 2] {TCP} 10.128.0.45:34238 ->
31.3.245.133:80
12/21/2020-02:08:17.334667  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned
root [**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 31.3.245.133:80
-> 10.128.0.45:34238
12/21/2020-02:08:33.823085  [**] [1:2402000:5763] ET DROP Dshield Block Listed Source
group 1 [**] [Classification: Misc Attack] [Priority: 2] {TCP} 185.175.93.104:52206 ->
10.128.0.45:5588
12/21/2020-02:08:48.380545  [**] [1:2013031:9] ET POLICY Python-urllib/ Suspicious
User Agent [**] [Classification: Attempted Information Leak] [Priority: 2] {TCP}
10.128.0.45:38738 -> 169.254.169.254:80
```

13. In a second window use **curl** to go to https://testmyids.com/. Shortly after the command runs you should see new traffic on the other terminal window.

```
student@cp:~$ curl https://testmyids.com/
```

```
uid=0(root) gid=0(root) groups=0(root)
```

14. View the first terminal session. You should see a similar error in the log to what is seen below. After you view the error you can use **control-c** to exit.

```
12/21/2020-03:46:35.488738  [**] [1:2013028:5] ET POLICY curl User-Agent Outbound [**]
 [Classification: Attempted Information Leak] [Priority: 2] {TCP} 10.128.0.45:53602
-> 31.3.245.133:80
12/21/2020-03:46:35.586844  [**] [1:2100498:7] GPL ATTACK_RESPONSE id check returned root
[**] [Classification: Potentially Bad Traffic] [Priority: 2] {TCP} 31.3.245.133:80
-> 10.128.0.45:53602
```

15. View more in-depth information by leveraging the **jq** command and the eve.json file. After you have seen some output, use **control-c** to quit back to a prompt.

```
student@cp:~$ sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="alert")'
```

```
{
  "timestamp": "2020-12-21T03:47:26.329687+0000",
  "flow_id": 1087808287199203,
  "in_iface": "ens4",
  "event_type": "alert",
  "src_ip": "10.128.0.45",
  "src_port": 58262,
  "dest_ip": "169.254.169.254",
  "dest_port": 80,
  "proto": "TCP",
  "tx_id": 0,
  "alert": {
    "action": "allowed",
....
```


16. Now view all statistics.

```
student@cp:~$ sudo tail -f /var/log/suricata/eve.json | jq 'select(.event_type=="stats")'
```


```
{
  "timestamp": "2020-12-21T03:51:09.637558+0000",
  "event_type": "stats",
  "stats": {
    "uptime": 1393,
    "capture": {
      "kernel_packets": 11173,
      "kernel_drops": 0,
      "errors": 0
    },
    "decoder": {
      "pkts": 11211,
      "bytes": 2130680,
      "invalid": 0,
      "ipv4": 11164,
      "ipv6": 1,
      "ethernet": 11211,
      "chdlc": 0,
      "raw": 0,
      "null": 0,
....
```


17. Read more about alerts and more information. Using a local browser navigate to https://suricata.readthedocs.io/en/latest/make-sense-alerts.html.

Exercise 8.2: Getting Started With OSSEC

Very Important

The OSSEC software has a wide install base and is a dynamic project. Available documentation may not be as useful as exploring the tool. As well there are three levels of features and tools from the basic download, the plus version which requires registration, and the enterprise version which requires purchasing a license.

We will use the basic version for this lab exercise.

1. Begin by installing the software using the **atomic** installer. Part of the output will be questions asking if you are willing to accept the terms. Read through the terms carefully before accepting. Also note there are three channels of software available, with only the more stable tree activated.

```
student@cp:~$  wget -q -O - https://updates.atomicorp.com/installers/atomic | sudo bash
```

```
<output_omitted>
OK
Adding [atomic] to /etc/apt/sources.list.d/atomic.list: OK

The Atomic repo has now been installed and configured for your system
The following channels are available:
  atomic          - [ACTIVATED] - contains the stable tree of ART packages
  atomic-testing  - [DISABLED]  - contains the testing tree of ART packages
  atomic-bleeding - [DISABLED]  - contains the development tree of ART packages
```

2. Update the repository.

```
student@cp:~$ sudo apt-get update
```

```
<output_omitted>
```

3. Now install the server software and answer any questions asked using the default values.

```
student@cp:~$ sudo apt-get install ossec-hids-server
```

```
<output_omitted>
```

4. Check the status of the `ossec.service`. Restart if not running. Note there are five or more processes created.

```
student@cp:~$ sudo systemctl status ossec.service
```

```
- ossec.service - LSB: Start and stop OSSEC HIDS
   Loaded: loaded (/etc/init.d/ossec; generated)
   Active: active (running) since Mon 2020-12-21 06:33:07 UTC; 6s ago
     Docs: man:systemd-sysv-generator(8)
<output_omitted>
```

If showing inactive then run these two commands and check again:

```
student@cp:~$ sudo systemctl enable ossec.service

student@cp:~$ sudo systemctl start ossec.service
```

5. Take a closer look at the new OSSEC processes being run, and what little included help documentation they present. Some run as the user root, others as the user ossec, which could be helpful when working with security logs and configuration. Note the directory where default configuration files are kept.

```
student@cp:~$ ps -ef |grep ossec
```

```
root     22787     1  0 16:06 ?        00:00:00 /var/ossec/bin/ossec-execd
ossec    22791     1  0 16:06 ?        00:00:00 /var/ossec/bin/ossec-analysisd
root     22796     1  0 16:06 ?        00:00:00 /var/ossec/bin/ossec-logcollector
root     22820     1  0 16:06 ?        00:00:00 /var/ossec/bin/ossec-syscheckd
ossec    22827     1  0 16:06 ?        00:00:00 /var/ossec/bin/ossec-monitord
student  25211 15345  0 16:09 pts/0    00:00:00 grep --color=auto ossec
```

```
student@cp:~$ sudo /var/ossec/bin/ossec-execd -h
```

```
OSSEC HIDS v3.6.0 - OSSEC Foundation (contact@ossec.net)
https://www.ossec.net
  ossec-execd: -[Vhdtf] [-g group] [-c config]
    -V          Version and license message
    -h          This help message
    -d          Execute in debug mode. This parameter
                can be specified multiple times
```

```
                to increase the debug level.
    -t          Test configuration
    -f          Run in foreground
    -g <group>  Group to run as (default: ossec)
    -c <config> Configuration file to use (default: /var/ossec/etc/ossec.conf)
```

6. View files associated with OSSEC. Take a look at the configuration files of `etc`, the various included `rules/`, and `logs/`.

```
student@cp:~$ sudo ls -R /var/ossec
```

```
/var/ossec:
active-response  agentless  bin  contrib  etc  logs  queue  rules  stats  tmp  var

/var/ossec/active-response:
bin

/var/ossec/active-response/bin:
cloudflare-ban.sh   firewalld-drop.sh  ipfw.sh            ossec-pagerduty.sh        pf.sh
disable-account.sh  host-deny.sh       ipfw_mac.sh  ossec-slack.sh        restart-ossec.sh
firewall-drop.sh    ip-customblock.sh  npf.sh             ossec-tweeter.sh        route-null.sh
<output_omitted>
```

7. Take a look at the main configuration file. As the directory is password protected, it may be easier in class to become root first. Note there are no email notifications by default, the various rules files included, files to check or exclude, and other settings.

```
student@cp:~$ sudo -i
```

```
root@cp:~$ cd /var/ossec/etc ; less ossec.conf
```

```
<ossec_config>
  <global>
    <email_notification>no</email_notification>
  </global>

  <rules>
    <include>rules_config.xml</include>
    <include>pam_rules.xml</include>
    <include>sshd_rules.xml</include>
<output_omitted>
```

8. View a couple of the included rules files. Note that there is not a file for Kubernetes by default, but the syntax makes adding the rule straightforward.

```
root@cp:~$ cd ../rules ; less sshd_rules.xml
```

```
....

  <rule id="5705" level="10" frequency="4" timeframe="360">
    <if_matched_sid>5704</if_matched_sid>
    <description>Possible scan or breakin attempt </description>
    <description>(high number of login timeouts).</description>
  </rule>

....
```

9. View the binaries associated with an active response to an issue. Read through a few of the scripts. Consider Kubernetes actions you may want to include in the list.

```
root@cp:~$ cd ../active-response/bin/ ; ls
```

```
cloudflare-ban.sh   host-deny.sh       npf.sh              pf.sh
disable-account.sh  ip-customblock.sh  ossec-pagerduty.sh  restart-ossec.sh
firewall-drop.sh    ipfw.sh            ossec-slack.sh      route-null.sh
firewalld-drop.sh   ipfw_mac.sh        ossec-tweeter.sh
```

```
root@cp:~$ less firewall-drop.sh
```

```
<output_omitted>
```

10. View some of the included text files, including some CIS based information. Take a look at a few of the included files.

```
root@cp:~$ cd ../../etc/shared ; ls
```

```
acsc_office2016_rcl.txt          cis_win2012r2_domainL2_rcl.txt
ar.conf                          cis_win2012r2_memberL1_rcl.txt
cis_apache2224_rcl.txt           cis_win2012r2_memberL2_rcl.txt
cis_debian_linux_rcl.txt         cis_win2016_domainL1_rcl.txt
<output_omitted>
```

```
root@cp:~$ less cis_debian_linux_rcl.txt
```

```
<output_omitted>
```

11. View a recently updated log file. Now that the software has been running for a while you should have a few logs to review.

```
root@cp:~$ cd ../../logs/ ; ls -R
```

```
.:
alerts  archives  firewall  ossec.log

./alerts:
2020  alerts.log

./alerts/2020:
Dec
<output_omitted>
```

```
root@cp:~$ less alerts/alerts.log
```

```
** Alert 1608532394.0: mail  - syslog,sudo
2020 Dec 21 06:33:14 cp->/var/log/auth.log
Rule: 5403 (level 4) -> 'First time user executed sudo.'
User: student
Dec 21 06:33:14 cp sudo:  student : TTY=pts/1 ; PWD=/home/student ;
USER=root ; COMMAND=/bin/syste
mctl status ossec.service

** Alert 1608532394.290: - pam,syslog,authentication_success,
2020 Dec 21 06:33:14 cp->/var/log/auth.log
Rule: 5501 (level 3) -> 'Login session opened.'
Dec 21 06:33:14 cp sudo: pam_unix(sudo:session): session opened for users
root by student(uid=0)
<output_omitted>
```

12. Continue to explore the tools, logs, and configuration files as time permits.

Chapter 9

Domain Reviews

9.1 Labs

Exercise 9.1: Finding Exam Information

1. Navigate to https://www.cncf.io/certification/cks/. Read through the page looking at what you may find in the exam, down to the section entitled `Exam resources`.
2. Select the link for `Candidate Handbook` which should take you to https://docs.linuxfoundation.org/tc-docs/certification/lf-candidate-handbook. This page contains information about all the **Linux Foundation** certifications. Read through the general information about taking an online performance-based exam. Locate the link to the browser `Compatibility Check Tool`. Ensure that you have an allowed browser and setup.
3. Expand the `Certification` tab on the left, then scroll down to find the `Important Instructions: CKS` link. Read through the particular information about the `CKS` exam.
4. Return to the main `CKS` exam page. This time select the `Curriculum Overview` link in the `Exam resources`. You should be brought to the github.com/cncf/curriculum page. Select the link for `CKS_Curriculum` for the most recent version if there is more than one. The new page should show a PDF. Scroll down to find the bullet point list of knowledge, skills, and abilities that a CKS specialist should be able to demonstrate.
5. Copy each bullet point over to a word processor, such as **libreoffice**. Under each bullet point write out the commands related to that action.
6. Test, then bookmark, known working YAML or other configuration files from one of the allowed web domains, as listed in the Candidate Handbook, on a browser you have already checked and are able to use during the exam.

Very Important

Ensure you have tested every bullet point steps and every YAML or source file on the system and browser you will be using for the exam and for the exam version you will be using.

Exercise 9.2: Practice Steps

These steps are included without detailed steps. You should be able to use the course book, bookmarked online resources, and experience to complete the steps. As some of these products **are dynamic and quick changing** you may have to research and test that the tool works, and continues to work with the resources you have bookmarked.

1. Use the CIS benchmark to evaluate your cluster. Remedy any major issues.
2. In an earlier lab we assigned the `create` verb to the `ops` group. Remove that permission, update the necessary roles and service accounts. Test that users in the group can no longer perform that task.
3. Create a new role that allows `Paul` to use the `list` and `watch`, but only in the `prod-b` namespace.
4. Execute the `review1.sh` script. The script will cause the **kube-apiserver** to become less secure. Cause the **kube-apiserver** to become more secure, and make sure the change would be in place should the node reboot.
5. Make sure that all pods with label `workgroup: dev` can communicate from `prod-a` to `prod-b`, but no traffic is allowed from either to `dev-ns`. Block ingress traffic from example.com to the `dev-ns` namespace.
6. Create an immutable pod in the `prod-b` namespace.
7. Enable API server auditing. Log everything in the `dev-ns` namespace. Log metadata for any changes to `secrets` in the `prod-a` namespace. Create a new secret and test that you can see the log in the audit file.
8. Configure **Falco**. Use the **Falco** events generator inside the `dev-ns` namespace to create events, and verify that **Falco** is working. Using the example from https://falco.org/docs/event-sources/kubernetes-audit/ create a configMap for AWS, and make sure that **Falco** alerts to its existence.
9. Use **Trivy** or other tool to scan for known vulnerabilities without running the image.
10. Use **Tracee**, **Falco**, or other tool to analyze a running container. Document what proper startup looks like, or a common function and save the output to a text file.
11. Create a new **AppArmor** profile for a container to run, and ensure that only the required permissions are included, but other functions are otherwise closed down.
12. Enable `kube-apiserver` auditing. Add a location to keep files. Ensure that files are kept for three days and are rotated when they reach 50M in size. Log all events having to do with secrets. Only log the request metadata for configmap objects. Include a catch all function to log the metadata for all other events.
13. Understand which admission controllers are currently in use. Determine how you would enable or disable an admission controller.
14. If your cluster was able to use `gVisor` or `kata`, what would you add to the pod to take advantage of that sandbox. How would you test that it is working as desired?
15. As this list is not exhaustive, rather an approach to some of the listed items, reference the PDF of knowledge, skills, and abilities. Create your own steps to satisfy each item. Remember to bookmark any code from an allowed website domain in the browser you will be using for the exam.
16. Set a timer and practice getting all the steps done in two hours.