ch #3 Installing Cluster

LFS 260

Installing Cluster

Choosing the hardware to run Kubernetes on may include several hardware options, which can either improve the security of the platform, or provide resources to containers, helping increase their security. Once the physical equipment has been chosen, the operating system should be secured during installation. Regular updates and monitoring are also important to limit exploits. After operating systems installation, you may want to implement Linux Security Modules (LSM) settings and install sandbox tools, should the design of the cluster call for their use.

By the end of this chapter, you should be able to:

- Plan on regular cluster and project software updates.
- Harden the kernel via various tools.
- Discuss about hardware options to increase security.
- Implement Linux Security Modules and management tools.

Cluster Version

With single-vendor products, the complexity of package interaction lies with the vendor.

Using a wide range of open source packages, without particular requirements for what is or is not used, requires a lot of documentation and testing by the customer.

Each of the products installed should also have an ongoing process to track security issues; not only that the issues are discovered, but also who is responsible for the fix, how long is allowable, and who will verify the fix fully deals with the vulnerability.

Protect the Kernel

As you build your cluster, there are several decisions to be made, and required configuration steps to perform prior to installing the Kubernetes packages.

Once an operating system has been chosen, you should determine if the vendor-supplied (and supported) kernel will be used, or if there are special circumstances which require a compiled kernel.

Another consideration is if advanced PAM modules will be used, as is sometimes the case for OS security.

Protect the Kernel

As vulnerabilities are discovered, you may need to update Kubernetes on a regular basis. As Kubernetes, and other projects, have been rather dynamic, this leads to an important decision:

- update whenever a vulnerability is found (but then have disruption when the API changes and existing configurations may need to be edited and redeployed)
- stay with the known working cluster (and hope the vulnerability is not exploited.)

Protect the Kernel

In general, the approach to the operating system and software installed should follow the concept of least privilege, in which the only software installed or permission allowed is the minimum necessary to do the job.

Instead of installing all the packages in a distribution, you would start with the minimal and add packages only as necessary. Every user would begin with no special access, and then add needed abilities one at a time.

Finding Kernel Vulnerabilities

The Linux Exploit Suggester script is a simple way to compare the kernel version running on a system with a public kernel vulnerability database (CVE).

Just clone the script from GitHub:

```
git clone https://github.com/jondonas/linux-exploit-suggester-2.git
```

and then run the script

```
./linux-exploit-suggester-2.pl
```

Mitigate kernel vulnerabilities

Blocking Dynamic Module

Several rootkits contain kernel modules which, once loaded, completely hide the rootkit from the typical system tools such as ps, top and others. In order to address this problem, the kernel community came up with a mechanism that disables kernel module loading completely.

Rootkits usually work as follows:

- Escalate privileges to root.
- Load kernel modules to hide the rootkit.
- Remove all traces of the rootkit.

The solution is to remove the ability to load modules.

Mitigate kernel vulnerabilities

kernel.modules disabled

System administrators can use a one-way method to block module loading after the system has booted and all modules have been loaded. This mechanism consists in writing "1" to kernel.modules_disabled.

This requires root or CAP_SYS_MODULE capabilities. No other module-related operations will be allowed from that point on.

A reboot is required in order to get the kernel back to its default *kernel.modules_disabled* setting.

Mitigate kernel vulnerabilities

Adaptive Stack Layout Randomization

All address spaces used to be build in the same way: Memory regions showed up at known addresses.

This created some attack vectors. Address Space Layout Randomization (ASLR) places various memory areas of a user-land executable in random locations, which helps prevent certain classes of attacks.

This is set with **kernel.randomize_va_space sysctl**:
0: Disabled
1: Randomized stack, VDSO (Virtual Dynamic Shared Object), shared memory addresses
2: Randomized stack, VDSO, shared memory and data addresses.

This was adapted from the external PaX/grsecurity projects, along with several other software-based hardening features.

Mitigate kernel vulnerabilities

Hardware Security Features

The Linux kernel also supports hardware security features where available, such as NX (No eXecute), VT-d (virtualization), the TPM (Trusted Platform Module), TXT (Trusted Execution Technology), and SMAF (Secure Memory Allocation Feature), along with various support for hardware cryptographic devices.

Mitigate kernel vulnerabilities

No eXecute

Most modern CPUs can protect against the execution of memory regions that should not contain executable code (heap, stack, etc), which helps block the exploitation of many security vulnerabilities.

Since this is implemented at a hardware level, applications typically do not need to suffer from any performance overhead (as opposed to a software-based approach, as discussed next).

Look for the nx flag in /proc/cpuinfo. It may be enabled/disabled in BIOS.

Installing Cluster

Create a two-nodeUbuntu 18.04 cluster.

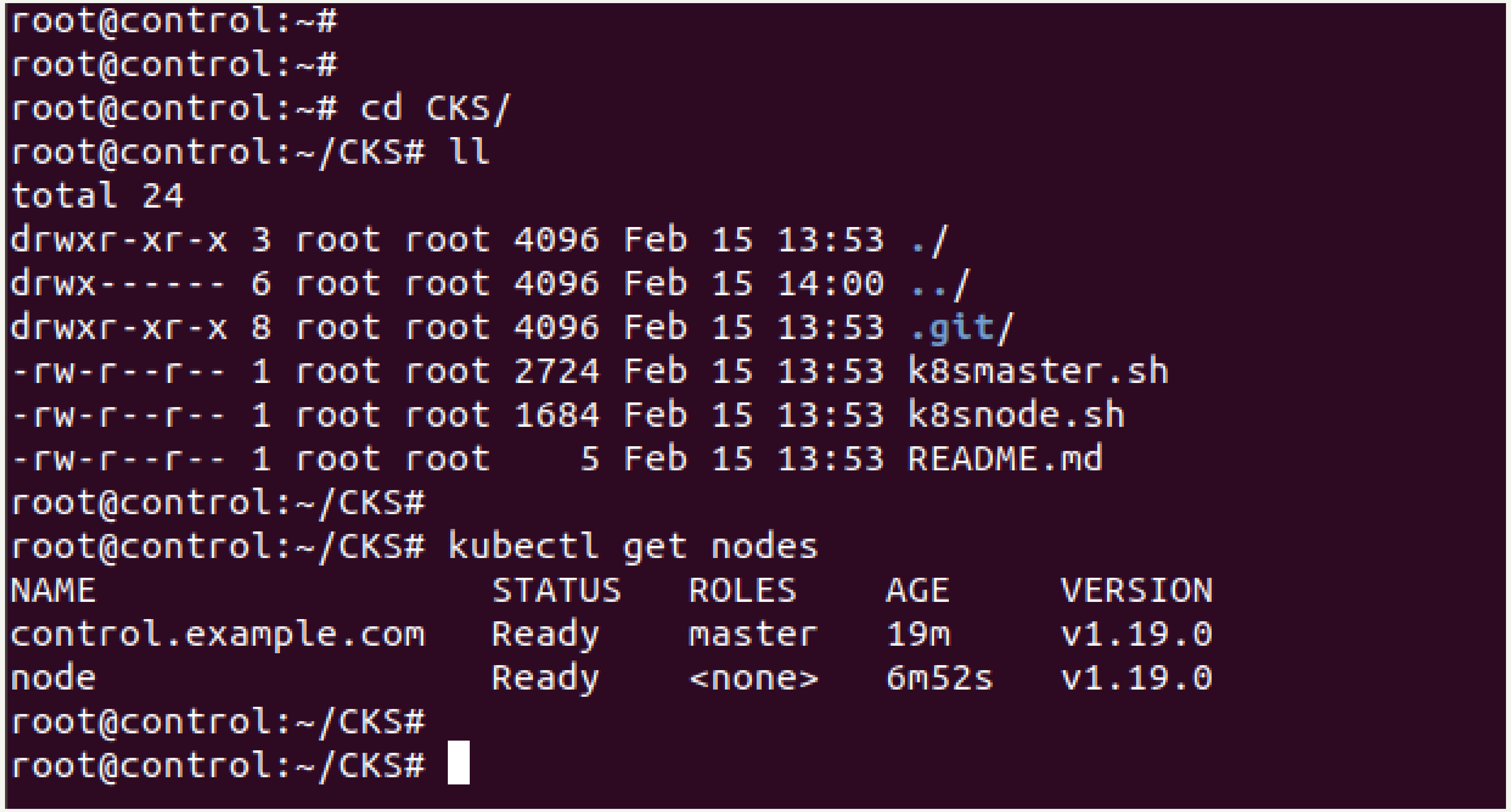
Control - control.example.com - 10.0.0.100
Node - node.example.com - 10.0.0.200

Using two nodes allows an understanding of some issues and configurations found in a production environment.

Install "vim" on both machines and update /etc/hosts

10.0.0.100 control.example.com control
10.0.0.200 node.example.com node

Installing Cluster



```
root@control:~#
root@control:~#
root@control:~# cd CKS/
root@control:~/CKS# ll
total 24
drwxr-xr-x 3 root root 4096 Feb 15 13:53 ./
drwx------ 6 root root 4096 Feb 15 14:00 ../
drwxr-xr-x 8 root root 4096 Feb 15 13:53 .git/
-rw-r--r-- 1 root root 2724 Feb 15 13:53 k8smaster.sh
-rw-r--r-- 1 root root 1684 Feb 15 13:53 k8snode.sh
-rw-r--r-- 1 root root    5 Feb 15 13:53 README.md
root@control:~/CKS#
root@control:~/CKS# kubectl get nodes
NAME                   STATUS   ROLES    AGE     VERSION
control.example.com    Ready    master   19m     v1.19.0
node                   Ready    <none>   6m52s   v1.19.0
root@control:~/CKS#
root@control:~/CKS#
```

Configure Cluster

By default the master 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. The worker/minion node does not have the taint to begin with.

```
# kubectl taint nodes --all node-role.kubernetes.io/master-
```

Check that both nodes are without Taint

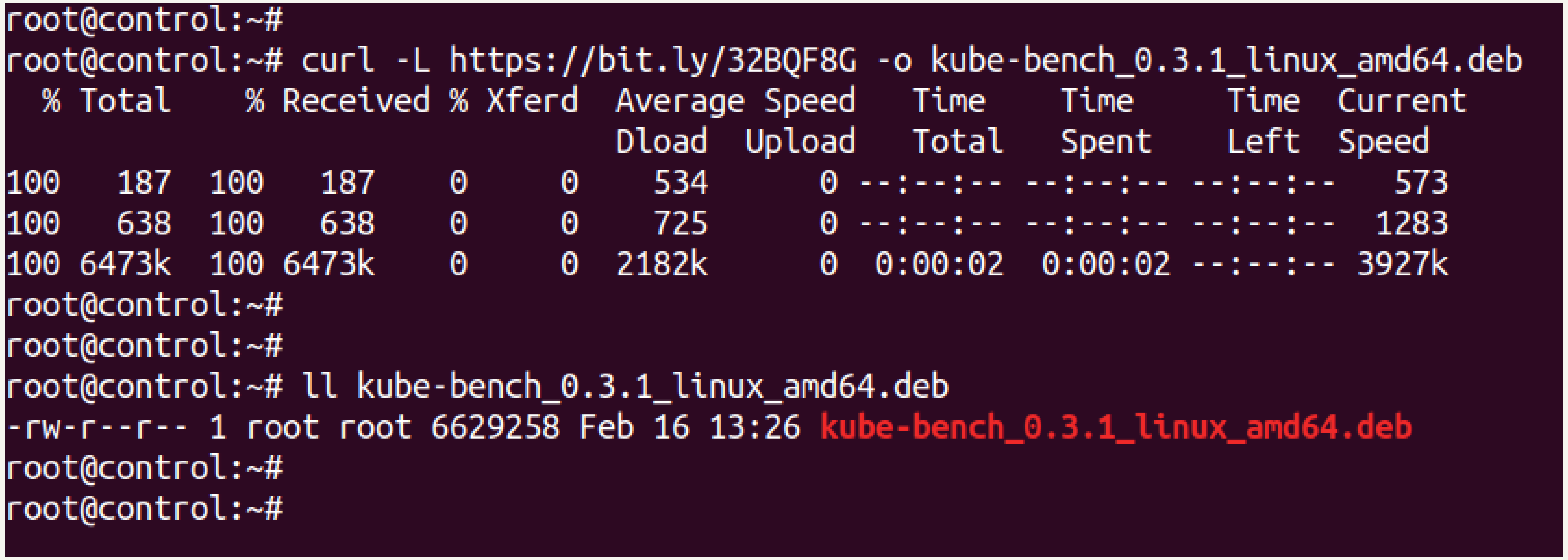
```
# kubectl describe nodes | grep -i taint
```

Also run the script - **setupscript.sh** on the control node.

Using kube-bench tool

Use the kube-bench tool from Aquasecurity which shows a text output of the issues followed by commands to run. This may allow an easierpath to keeping your environment secure. Download the Aquasecurity kube-bench binary.

# curl -L https://bit.ly/32BQF8G -o kube-bench_0.3.1_linux_amd64.deb

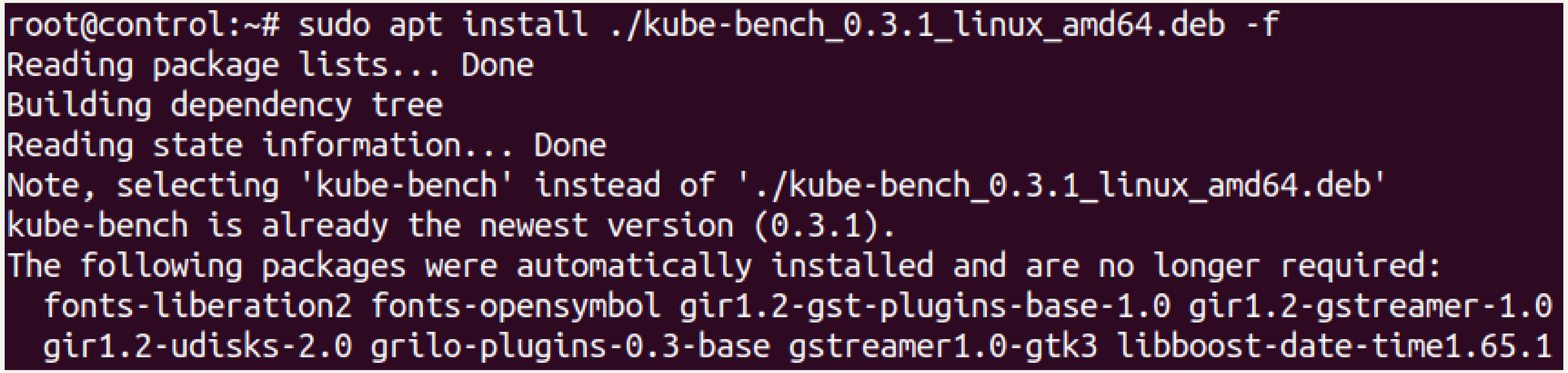


```
root@control:~#
root@control:~# curl -L https://bit.ly/32BQF8G -o kube-bench_0.3.1_linux_amd64.deb
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   187  100   187    0     0    534      0 --:--:-- --:--:-- --:--:--   573
100   638  100   638    0     0    725      0 --:--:-- --:--:-- --:--:--  1283
100 6473k  100 6473k    0     0  2182k      0  0:00:02  0:00:02 --:--:-- 3927k
root@control:~#
root@control:~#
root@control:~# ll kube-bench_0.3.1_linux_amd64.deb
-rw-r--r-- 1 root root 6629258 Feb 16 13:26 kube-bench_0.3.1_linux_amd64.deb
root@control:~#
root@control:~#
```

Using kube-bench tool

Install the kube-bench binary.

#sudo apt install ./kube-bench_0.3.1_linux_amd64.deb -f



```
root@control:~# sudo apt install ./kube-bench_0.3.1_linux_amd64.deb -f
Reading package lists... Done
Building dependency tree
Reading state information... Done
Note, selecting 'kube-bench' instead of './kube-bench_0.3.1_linux_amd64.deb'
kube-bench is already the newest version (0.3.1).
The following packages were automatically installed and are no longer required:
  fonts-liberation2 fonts-opensymbol gir1.2-gst-plugins-base-1.0 gir1.2-gstreamer-1.0
  gir1.2-udisks-2.0 grilo-plugins-0.3-base gstreamer1.0-gtk3 libboost-date-time1.65.1
```

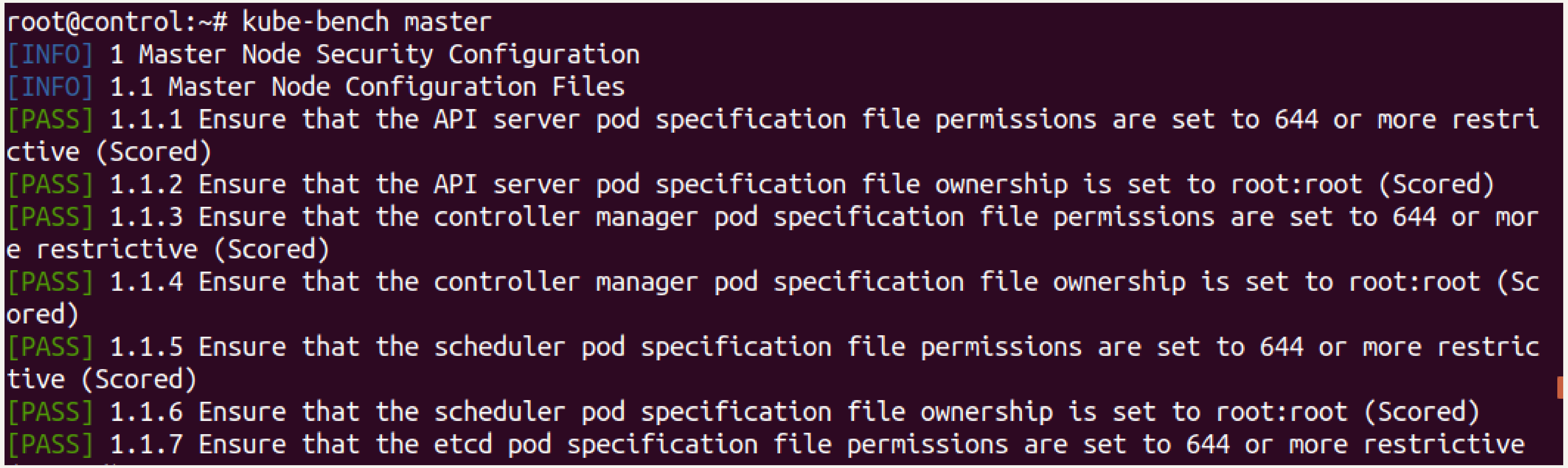
Using kube-bench tool

Check the binary path

#which kube-bench

Run the kube-bench command to check master.

#kube-bench master



```
root@control:~# kube-bench master
[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 more restri
ctive (Scored)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Scored)
[PASS] 1.1.3 Ensure that the controller manager pod specification file permissions are set to 644 or mor
e restrictive (Scored)
[PASS] 1.1.4 Ensure that the controller manager pod specification file ownership is set to root:root (Sc
ored)
[PASS] 1.1.5 Ensure that the scheduler pod specification file permissions are set to 644 or more restric
tive (Scored)
[PASS] 1.1.6 Ensure that the scheduler pod specification file ownership is set to root:root (Scored)
[PASS] 1.1.7 Ensure that the etcd pod specification file permissions are set to 644 or more restrictive
```

Using kube-bench tool

Check for "Remediations" and "Summary"