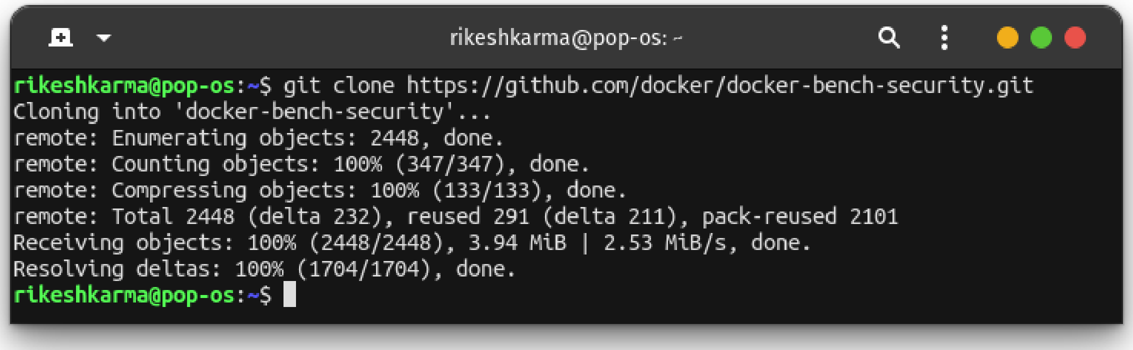
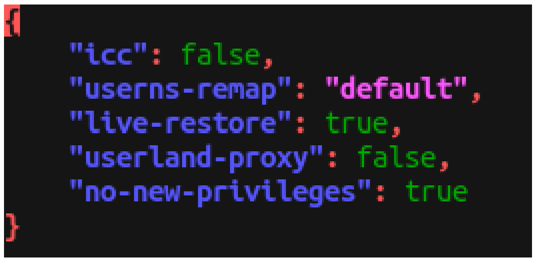
1. Install the Docker security tools and scan all the created images and containers and optimize the vulnerabilities.
   - ➔ There are quite a few Docker Security tools, and for this task, we will be installing a tool called **Docker Bench.** This docker security tool can run an audit of our docker's host machine to find any possible issues we might not even know about.
   - ➔ According to Docker documentation, "the Docker Bench for Security is a script that checks for dozens of common best-practices around deploying Docker containers in production."
   - ➔ Now let's install and run Docker Bench:
     - ◆ To install Docker Bench, we need to clone the latest version from the official GitHub repository, we won't find this tool in the standard repositories. So we need to clone the Docker Bench repository with the following command:
       - *git clone https://github.com/docker/docker-bench-security.git*



```
rikeshkarma@pop-os:~$ git clone https://github.com/docker/docker-bench-security.git
Cloning into 'docker-bench-security'...
remote: Enumerating objects: 2448, done.
remote: Counting objects: 100% (347/347), done.
remote: Compressing objects: 100% (133/133), done.
remote: Total 2448 (delta 232), reused 291 (delta 211), pack-reused 2101
Receiving objects: 100% (2448/2448), 3.94 MiB | 2.53 MiB/s, done.
Resolving deltas: 100% (1704/1704), done.
rikeshkarma@pop-os:~$
```

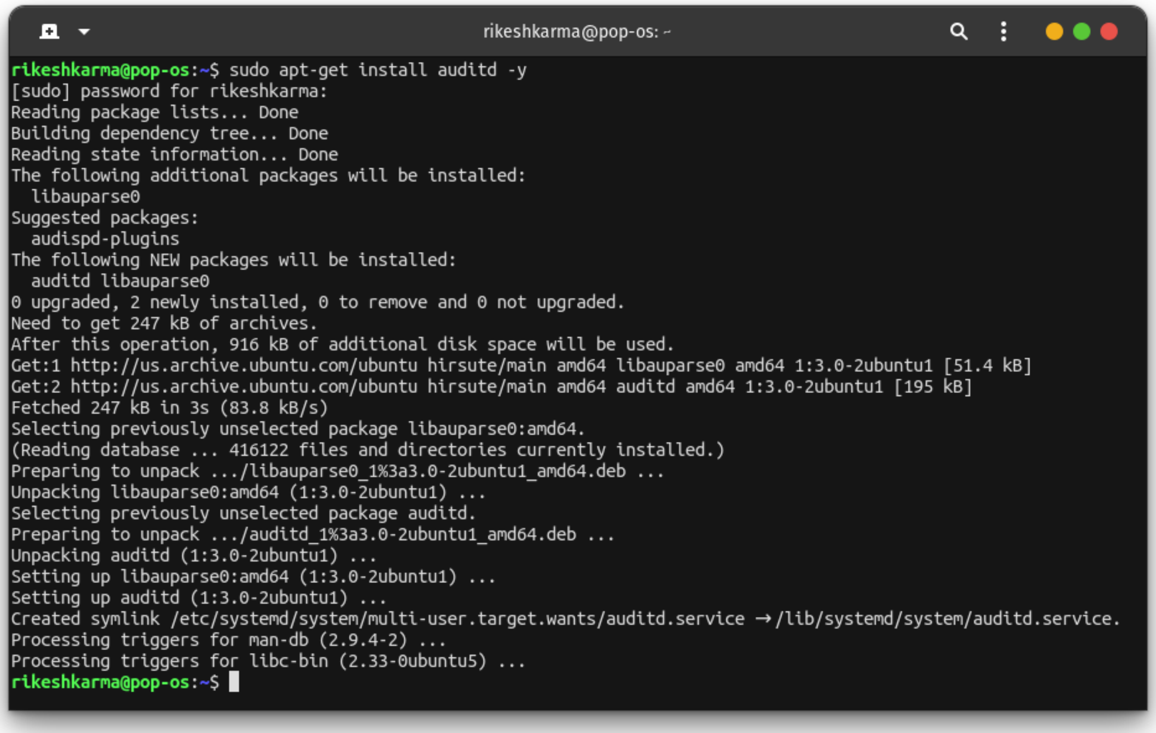
     - ◆ Now we need to configure the Docker daemon, by modifying the Docker daemon configuration file so that it can be accessed by Docker Bench. To open the configuration file use the following command:
       - *sudo nano /etc/docker/daemon.json*
       - And in file, we need to add the following lines:



```
{
    "icc": false,
    "userns-remap": "default",
    "live-restore": true,
    "userland-proxy": false,
    "no-new-privileges": true
}
```

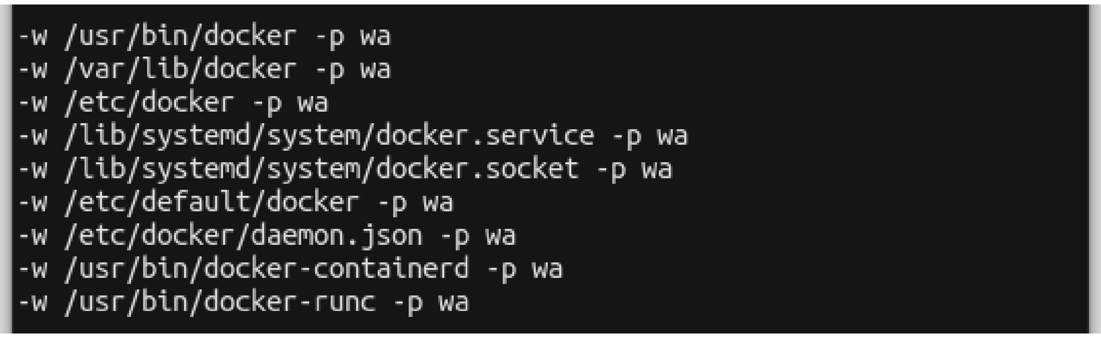
- Now we can close the file after saving it.

◆ For Docker Bench, we also need to install *Auditd,* which is the Linux userspace component of the Linux Auditing System which is responsible for writing audit records to the disk. We can use the following command to install this software:
  - *sudo apt-get install auditd -y*



```
rikeshkarma@pop-os:~$ sudo apt-get install auditd -y
[sudo] password for rikeshkarma:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libauparse0
Suggested packages:
  audispd-plugins
The following NEW packages will be installed:
  auditd libauparse0
0 upgraded, 2 newly installed, 0 to remove and 0 not upgraded.
Need to get 247 kB of archives.
After this operation, 916 kB of additional disk space will be used.
Get:1 http://us.archive.ubuntu.com/ubuntu hirsute/main amd64 libauparse0 amd64 1:3.0-2ubuntu1 [51.4 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu hirsute/main amd64 auditd amd64 1:3.0-2ubuntu1 [195 kB]
Fetched 247 kB in 3s (83.8 kB/s)
Selecting previously unselected package libauparse0:amd64.
(Reading database ... 416122 files and directories currently installed.)
Preparing to unpack .../libauparse0_1%3a3.0-2ubuntu1_amd64.deb ...
Unpacking libauparse0:amd64 (1:3.0-2ubuntu1) ...
Selecting previously unselected package auditd.
Preparing to unpack .../auditd_1%3a3.0-2ubuntu1_amd64.deb ...
Unpacking auditd (1:3.0-2ubuntu1) ...
Setting up libauparse0:amd64 (1:3.0-2ubuntu1) ...
Setting up auditd (1:3.0-2ubuntu1) ...
Created symlink /etc/systemd/system/multi-user.target.wants/auditd.service → /lib/systemd/system/auditd.service.
Processing triggers for man-db (2.9.4-2) ...
Processing triggers for libc-bin (2.33-0ubuntu5) ...
rikeshkarma@pop-os:~$
```

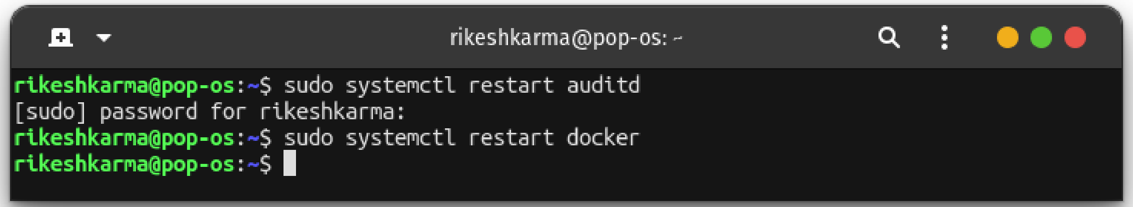
- Now again we need to open Auditd configuration file so that it can work with Docker. Let's use the following command:
  - *sudo nano /etc/audit/audit.rules*
- And at the bottom of this file, add the following lines:



```
-w /usr/bin/docker -p wa
-w /var/lib/docker -p wa
-w /etc/docker -p wa
-w /lib/systemd/system/docker.service -p wa
-w /lib/systemd/system/docker.socket -p wa
-w /etc/default/docker -p wa
-w /etc/docker/daemon.json -p wa
-w /usr/bin/docker-containerd -p wa
-w /usr/bin/docker-runc -p wa
```

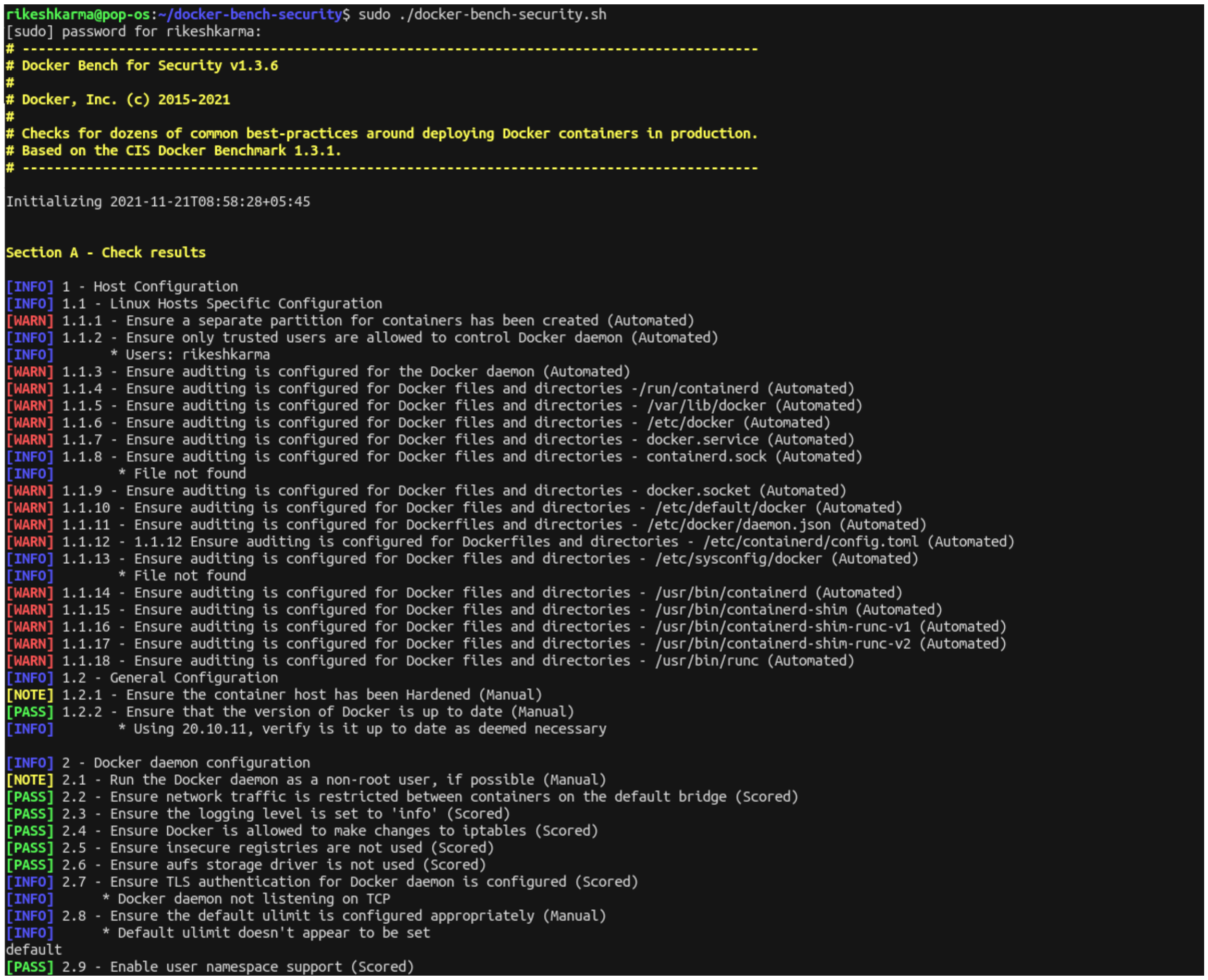
- Now we can close the file after saving it.

◆ Now let's run out the first audit. But before that let's restart Auditd and Docker using the following commands:
  - *sudo systemctl restart auditd*
  - *sudo systemctl restart docker*

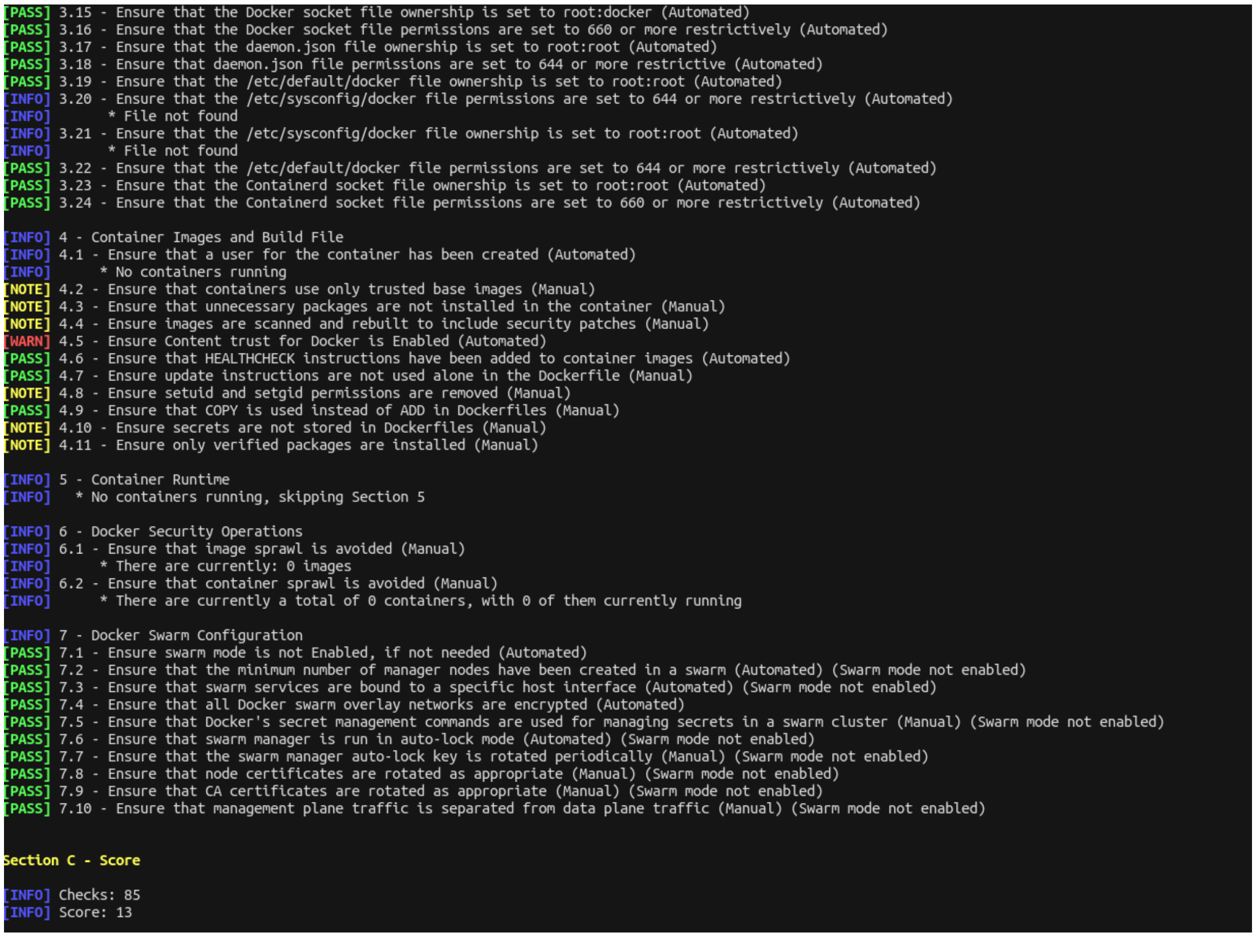


```
rikeshkarma@pop-os:~$ sudo systemctl restart auditd
[sudo] password for rikeshkarma:
rikeshkarma@pop-os:~$ sudo systemctl restart docker
rikeshkarma@pop-os:~$
```

- Now let's navigate inside the repository we cloned earlier and run the following command:
  - *sudo ./docker-bench-security.sh*



```
rikeshkarma@pop-os:~/docker-bench-security$ sudo ./docker-bench-security.sh
[sudo] password for rikeshkarma:
# --------------------------------------------------------------------------------------------
# Docker Bench for Security v1.3.6
#
# Docker, Inc. (c) 2015-2021
#
# Checks for dozens of common best-practices around deploying Docker containers in production.
# Based on the CIS Docker Benchmark 1.3.1.
# --------------------------------------------------------------------------------------------

Initializing 2021-11-21T08:58:28+05:45


Section A - Check results

[INFO] 1 - Host Configuration
[INFO] 1.1 - Linux Hosts Specific Configuration
[WARN] 1.1.1 - Ensure a separate partition for containers has been created (Automated)
[INFO] 1.1.2 - Ensure only trusted users are allowed to control Docker daemon (Automated)
[INFO]       * Users: rikeshkarma
[WARN] 1.1.3 - Ensure auditing is configured for the Docker daemon (Automated)
[WARN] 1.1.4 - Ensure auditing is configured for Docker files and directories -/run/containerd (Automated)
[WARN] 1.1.5 - Ensure auditing is configured for Docker files and directories - /var/lib/docker (Automated)
[WARN] 1.1.6 - Ensure auditing is configured for Docker files and directories - /etc/docker (Automated)
[WARN] 1.1.7 - Ensure auditing is configured for Docker files and directories - docker.service (Automated)
[INFO] 1.1.8 - Ensure auditing is configured for Docker files and directories - containerd.sock (Automated)
[INFO]        * File not found
[WARN] 1.1.9 - Ensure auditing is configured for Docker files and directories - docker.socket (Automated)
[WARN] 1.1.10 - Ensure auditing is configured for Docker files and directories - /etc/default/docker (Automated)
[WARN] 1.1.11 - Ensure auditing is configured for Dockerfiles and directories - /etc/docker/daemon.json (Automated)
[WARN] 1.1.12 - 1.1.12 Ensure auditing is configured for Dockerfiles and directories - /etc/containerd/config.toml (Automated)
[INFO] 1.1.13 - Ensure auditing is configured for Docker files and directories - /etc/sysconfig/docker (Automated)
[INFO]        * File not found
[WARN] 1.1.14 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd (Automated)
[WARN] 1.1.15 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim (Automated)
[WARN] 1.1.16 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v1 (Automated)
[WARN] 1.1.17 - Ensure auditing is configured for Docker files and directories - /usr/bin/containerd-shim-runc-v2 (Automated)
[WARN] 1.1.18 - Ensure auditing is configured for Docker files and directories - /usr/bin/runc (Automated)
[INFO] 1.2 - General Configuration
[NOTE] 1.2.1 - Ensure the container host has been Hardened (Manual)
[PASS] 1.2.2 - Ensure that the version of Docker is up to date (Manual)
[INFO]        * Using 20.10.11, verify is it up to date as deemed necessary

[INFO] 2 - Docker daemon configuration
[NOTE] 2.1 - Run the Docker daemon as a non-root user, if possible (Manual)
[PASS] 2.2 - Ensure network traffic is restricted between containers on the default bridge (Scored)
[PASS] 2.3 - Ensure the logging level is set to 'info' (Scored)
[PASS] 2.4 - Ensure Docker is allowed to make changes to iptables (Scored)
[PASS] 2.5 - Ensure insecure registries are not used (Scored)
[PASS] 2.6 - Ensure aufs storage driver is not used (Scored)
[INFO] 2.7 - Ensure TLS authentication for Docker daemon is configured (Scored)
[INFO]      * Docker daemon not listening on TCP
[INFO] 2.8 - Ensure the default ulimit is configured appropriately (Manual)
[INFO]      * Default ulimit doesn't appear to be set
default
[PASS] 2.9 - Enable user namespace support (Scored)
```



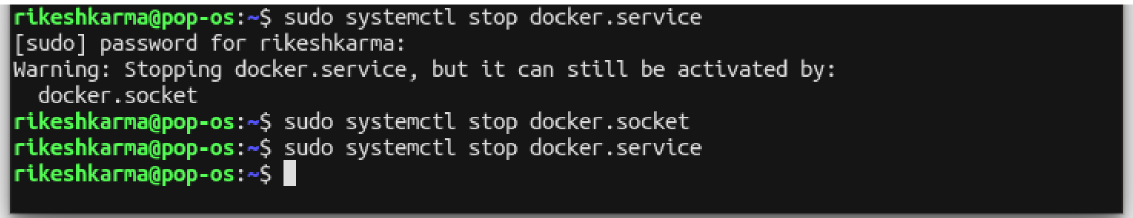
```
[PASS] 3.15 - Ensure that the Docker socket file ownership is set to root:docker (Automated)
[PASS] 3.16 - Ensure that the Docker socket file permissions are set to 660 or more restrictively (Automated)
[PASS] 3.17 - Ensure that the daemon.json file ownership is set to root:root (Automated)
[PASS] 3.18 - Ensure that daemon.json file permissions are set to 644 or more restrictive (Automated)
[PASS] 3.19 - Ensure that the /etc/default/docker file ownership is set to root:root (Automated)
[INFO] 3.20 - Ensure that the /etc/sysconfig/docker file permissions are set to 644 or more restrictively (Automated)
[INFO]      * File not found
[INFO] 3.21 - Ensure that the /etc/sysconfig/docker file ownership is set to root:root (Automated)
[INFO]      * File not found
[PASS] 3.22 - Ensure that the /etc/default/docker file permissions are set to 644 or more restrictively (Automated)
[PASS] 3.23 - Ensure that the Containerd socket file ownership is set to root:root (Automated)
[PASS] 3.24 - Ensure that the Containerd socket file permissions are set to 660 or more restrictively (Automated)

[INFO] 4 - Container Images and Build File
[INFO] 4.1 - Ensure that a user for the container has been created (Automated)
[INFO]      * No containers running
[NOTE] 4.2 - Ensure that containers use only trusted base images (Manual)
[NOTE] 4.3 - Ensure that unnecessary packages are not installed in the container (Manual)
[NOTE] 4.4 - Ensure images are scanned and rebuilt to include security patches (Manual)
[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)
[PASS] 4.6 - Ensure that HEALTHCHECK instructions have been added to container images (Automated)
[PASS] 4.7 - Ensure update instructions are not used alone in the Dockerfile (Manual)
[NOTE] 4.8 - Ensure setuid and setgid permissions are removed (Manual)
[PASS] 4.9 - Ensure that COPY is used instead of ADD in Dockerfiles (Manual)
[NOTE] 4.10 - Ensure secrets are not stored in Dockerfiles (Manual)
[NOTE] 4.11 - Ensure only verified packages are installed (Manual)

[INFO] 5 - Container Runtime
[INFO]   * No containers running, skipping Section 5

[INFO] 6 - Docker Security Operations
[INFO] 6.1 - Ensure that image sprawl is avoided (Manual)
[INFO]      * There are currently: 0 images
[INFO] 6.2 - Ensure that container sprawl is avoided (Manual)
[INFO]      * There are currently a total of 0 containers, with 0 of them currently running

[INFO] 7 - Docker Swarm Configuration
[PASS] 7.1 - Ensure swarm mode is not Enabled, if not needed (Automated)
[PASS] 7.2 - Ensure that the minimum number of manager nodes have been created in a swarm (Automated) (Swarm mode not enabled)
[PASS] 7.3 - Ensure that swarm services are bound to a specific host interface (Automated) (Swarm mode not enabled)
[PASS] 7.4 - Ensure that all Docker swarm overlay networks are encrypted (Automated)
[PASS] 7.5 - Ensure that Docker's secret management commands are used for managing secrets in a swarm cluster (Manual) (Swarm mode not enabled)
[PASS] 7.6 - Ensure that swarm manager is run in auto-lock mode (Automated) (Swarm mode not enabled)
[PASS] 7.7 - Ensure that the swarm manager auto-lock key is rotated periodically (Manual) (Swarm mode not enabled)
[PASS] 7.8 - Ensure that node certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.9 - Ensure that CA certificates are rotated as appropriate (Manual) (Swarm mode not enabled)
[PASS] 7.10 - Ensure that management plane traffic is separated from data plane traffic (Manual) (Swarm mode not enabled)


Section C - Score

[INFO] Checks: 85
[INFO] Score: 13
```

- We can see a lot of information being reported. The ones we need to pay close attention to are the ones that are marked as [WARN] because they might be a security issue. In the end, we can also see the number of checks performed and our score.

- We can also save the output to a file if we want to, for this, we can use the following command:
    - *sudo ./docker-bench-security.sh -l scan-results*
    - We can view the scanned filed later, if we need to using following command:
        - *less scan_results*
        - We will be warned that it may be a binary file, but it is still viewable.

- ➔ We can observe that there are alot of warnings. Let’s try optimizing:
  - ◆ We can find one warning that says:
    - *[WARN] 4.5 - Ensure Content trust for Docker is Enabled (Automated)*
  - ◆ We can optimize this warning by enabling content trust with the following command in the terminal:
    - *sudo echo "DOCKER_CONTENT_TRUST=1" | sudo tee -a /etc/environment*
  - ◆ Then we need to restart the Docker and the Content trust warning will not be shown.

2. Migrate the default working directory of docker i.e /var/lib/docker to /app/docker/.
   - ➔ By default, Docker stores most of its data inside */var/lib/docker* directory. So for this task, we need to migrate this default directory to /app/docker.
     - ◆ So now to change we have to sync or migrate files from the default directory to our preferred directory, let’s start by stopping Docker from running because making the changes while docker is running is certain to cause errors. We can use the following command to stop Docker:
       - *sudo systemctl stop docker.service*
       - *sudo systemctl stop docker.socket*

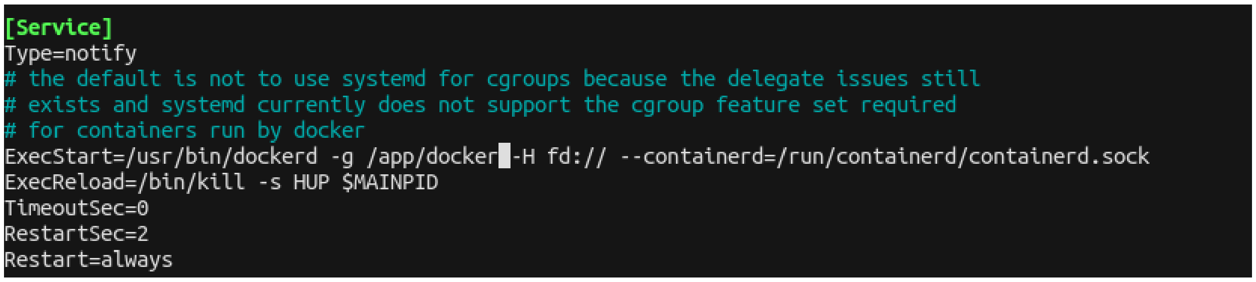


```
rikeshkarma@pop-os:~$ sudo systemctl stop docker.service
[sudo] password for rikeshkarma:
Warning: Stopping docker.service, but it can still be activated by:
  docker.socket
rikeshkarma@pop-os:~$ sudo systemctl stop docker.socket
rikeshkarma@pop-os:~$ sudo systemctl stop docker.service
rikeshkarma@pop-os:~$
```

- ◆ Firstly let’s copy our contents from */var/lib/docker* to our new desired location i.e */app/docker/.* Let’s create our desired directory using the following command:
  - sudo mkdir -p /app/docker
- ◆ After we are done creating our desired directory, let’s use the *rsync* command to transfer the files from */var/lib/docker/* to */app/docker* using the following command:
  - *sudo rsync -avxP /var/lib/docker/ /app/docker*
  - This might take some few minutes for completion.
- ◆ Now we need to edit the Docker unit file in path */lib/systemd/system/docker.service*. Here we need to enter the new

location inside the file, this is the systemd file that relates to Docker. For this let's use nano to edit the file. Use the following command:

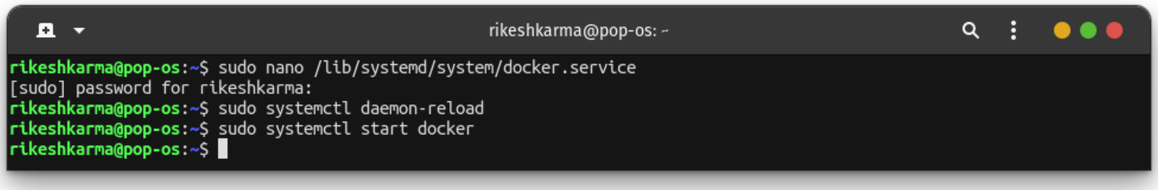
- *sudo nano /lib/systemd/system/docker.service*
- Here we need to add our new desired location in *ExecStart* line by putting *-g* and the location of the new Docker directory, which will look like:
    - ExecStart=/usr/bin/dockerd -g /app/docker -H fd:// --containerd=/run/containerd/containerd.sock



```
[Service]
Type=notify
# the default is not to use systemd for cgroups because the delegate issues still
# exists and systemd currently does not support the cgroup feature set required
# for containers run by docker
ExecStart=/usr/bin/dockerd -g /app/docker -H fd:// --containerd=/run/containerd/containerd.sock
ExecReload=/bin/kill -s HUP $MAINPID
TimeoutSec=0
RestartSec=2
Restart=always
```

◆ Lastly, let's reload the systemd configuration for Docker, after making all these changes. To reload the system daemons let's use the following command:

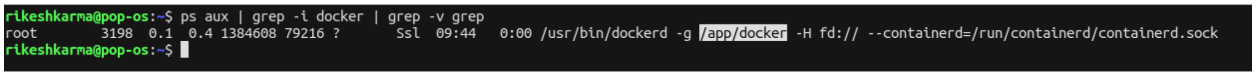
- *sudo systemctl daemon-reload*
- And to finally start Docker use the following command:
    - *sudo systemctl start docker*



```
rikeshkarma@pop-os:~$ sudo nano /lib/systemd/system/docker.service
[sudo] password for rikeshkarma:
rikeshkarma@pop-os:~$ sudo systemctl daemon-reload
rikeshkarma@pop-os:~$ sudo systemctl start docker
rikeshkarma@pop-os:~$ 
```

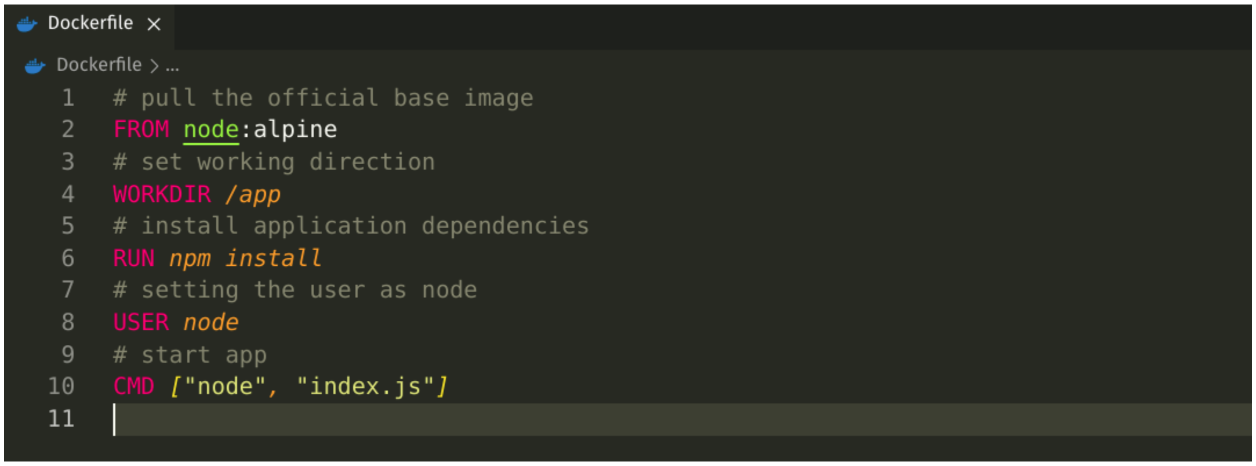
◆ Now to confirm if we have successfully migrated our default working directory of docker to /app/docker, we can use the *ps command* to make sure if the Docker service is utilizing our new directory using the following command:

- *ps aux | grep -i docker | grep -v grep*



```
rikeshkarma@pop-os:~$ ps aux | grep -i docker | grep -v grep
root        3198  0.1  0.4 1384608 79216 ?       Ssl  09:44   0:00 /usr/bin/dockerd -g /app/docker -H fd:// --containerd=/run/containerd/containerd.sock
rikeshkarma@pop-os:~$ 
```

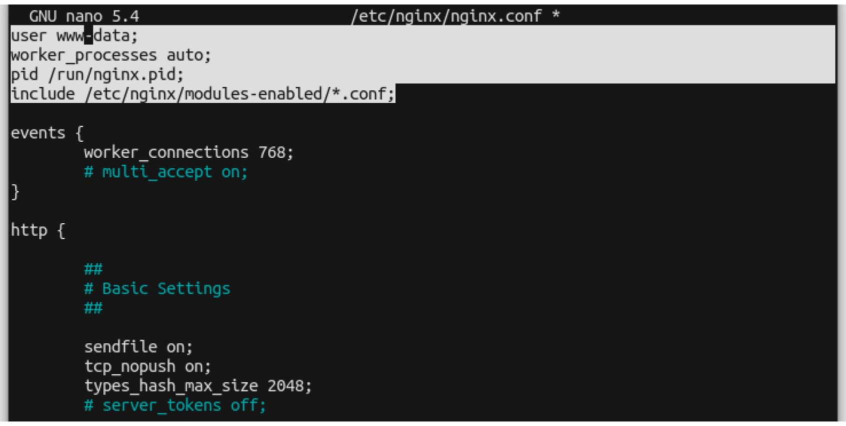
3. Provide the least privileges to each container and read-only access in mount volumes.
   - ➔ To provide least privileges to a container, we need to add a line in the Docker file which will set a non root user node and will manage the privilege.
     - ◆ Let's take node js first, in the Docker file of node js we need to add the following line so that it has non root user node. Check the snapshot:



```
Dockerfile ×
Dockerfile > ...
# pull the official base image
FROM node:alpine
# set working direction
WORKDIR /app
# install application dependencies
RUN npm install
# setting the user as node
USER node
# start app
CMD ["node", "index.js"]

```

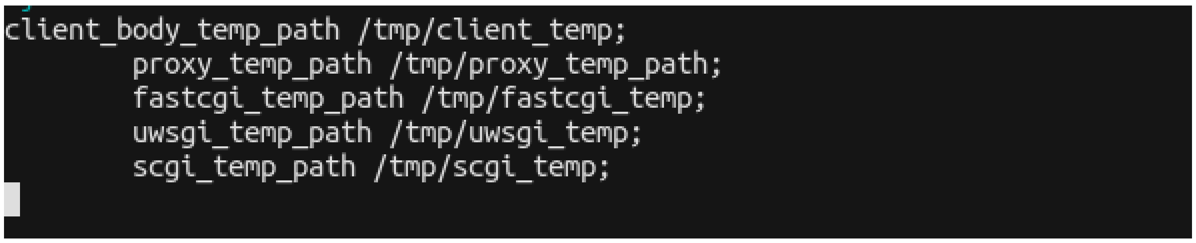
     - ◆ If the container does not provide non root user, we can create it after installing the dependencies and moving the source files into the container. Then run the container as a new user.

   - ➔ Alternatively we can also grant the least privileges to the already existing containers.
     - ◆ Let's take Nginx as an example. Firstly we need to remove the user directive on top of the nginx.conf file.
     - ◆ Then provide temporary pid to Nginx.



```
  GNU nano 5.4                          /etc/nginx/nginx.conf *
user www-data;
worker_processes auto;
pid /run/nginx.pid;
include /etc/nginx/modules-enabled/*.conf;

events {
        worker_connections 768;
        # multi_accept on;
}

http {

        ##
        # Basic Settings
        ##

        sendfile on;
        tcp_nopush on;
        types_hash_max_size 2048;
        # server_tokens off;
```

- And finally using the temporary paths to run the services uswgi, fastci, scfi as on snapshot below:



```
client_body_temp_path /tmp/client_temp;
        proxy_temp_path /tmp/proxy_temp_path;
        fastcgi_temp_path /tmp/fastcgi_temp;
        uwsgi_temp_path /tmp/uwsgi_temp;
        scgi_temp_path /tmp/scgi_temp;
```

➔ And finally to give read only permissions to the docker volumes:
  - Read only permissions can be given to docker containers by setting the *read-only* flag in docker images volume section to to *true*. This flag makes sure that the container has read only access in the mounted volumes.

Findings

1. To confirm for migration, we can also inspect one of our images. For this let's first get the *IMAGE ID* using the following command:
   - *docker images*
   - Then let's inspect the image using the *IMAGE ID* and look for WORKDIR using following command:
     - *docker inspect <image_id> | grep WorkDir*