This scenario will explore the different ways you can handle logging output from your application and services when running as containers.

Step 1 - Docker Logs
When you start a container, Docker will track the Standard Out and Standard Error outputs from the process and make them available via the client.

Example
In the background, there is an instance of Redis running with the name redis-server. Using the Docker client, we can access the standard out and standard error outputs using docker logs redis-server

```
$
$
$ docker logs redis-server
1:C 17 Apr 2023 09:37:36.012 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 17 Apr 2023 09:37:36.015 # Redis version=7.0.10, bits=64, commit=00000000, modified=0, pid=1, just started
1:C 17 Apr 2023 09:37:36.015 # Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf
1:M 17 Apr 2023 09:37:36.015 * monotonic clock: POSIX clock_gettime
1:M 17 Apr 2023 09:37:36.016 * Running mode=standalone, port=6379.
1:M 17 Apr 2023 09:37:36.016 # WARNING: The TCP backlog setting of 511 cannot be enforced because /proc/sys/net/core/somaxconn is set to the lower value of 128.
1:M 17 Apr 2023 09:37:36.016 # Server initialized
1:M 17 Apr 2023 09:37:36.016 # WARNING Memory overcommit must be enabled! Without it, a background save or replication may fail under low memory condition. Being disabled,
 it can can also cause failures without low memory condition, see https://github.com/jemalloc/jemalloc/issues/1328. To fix this issue add 'vm.overcommit_memory = 1' to /et
c/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
1:M 17 Apr 2023 09:37:36.016 * Ready to accept connections
$
```

Step 2 - SysLog
By default, the Docker logs are outputting using the json-file logger meaning the output stored in a JSON file on the host. This can result in large files filling the disk. As a result, you can change the log driver to move to a different destination.

Syslog
The Syslog log driver will write all the container logs to the central syslog on the host. "syslog is a widely used standard for message logging. It permits separation of the software that generates messages, the system that stores them, and the software that reports and analyses them." Wikipedia

This log-driver is designed to be used when syslog is being collected and aggregated by an external system.

Example
The command below will redirect the redis logs to syslog.

docker run -d --name redis-syslog --log-driver=syslog redis

Accessing Logs
If you attempted to view the logs using the client you'll recieve the error FATA[0000] "logs" command is supported only for "json-file" logging driver

Instead, you need to access them via the syslog stream.

```
$ docker run -d --name redis-server redis                                                                    Press
7af5b3c99e4f3ac374c642862f13167f3207c539a22d1fab5f30473c8088b4ab
$
$
$ docker logs redis-server
1:C 17 Apr 2023 09:37:36.012 # oO0OoO0OoO0Oo Redis is starting oO0OoO0OoO0Oo
1:C 17 Apr 2023 09:37:36.015 # Redis version=7.0.10, bits=64, commit=00000000, m
1:C 17 Apr 2023 09:37:36.015 # Warning: no config file specified, using the defa
1:M 17 Apr 2023 09:37:36.015 * monotonic clock: POSIX clock_gettime
1:M 17 Apr 2023 09:37:36.016 * Running mode=standalone, port=6379.
1:M 17 Apr 2023 09:37:36.016 # WARNING: The TCP backlog setting of 511 cannot be
1:M 17 Apr 2023 09:37:36.016 # Server initialized
1:M 17 Apr 2023 09:37:36.016 # WARNING Memory overcommit must be enabled! Withou
 it can can also cause failures without low memory condition, see https://github
c/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1'
1:M 17 Apr 2023 09:37:36.016 * Ready to accept connections
$ docker run -d --name redis-syslog --log-driver=syslog redis
06303a6fef00052a6365f7534ecdd79429793fe0701931e38e468d09e48316c0
$ docker logs redis-syslog
Error response from daemon: configured logging driver does not support reading
$
```

Step 3 - Disable Logging
The third option is to disable logging on the container. This is particularly useful for containers which are very verbose in their logging.

Example
When the container is launched simply set the log-driver to none. No output will be logged.

docker run -d --name redis-none --log-driver=none redis

```
$ docker run -d --name redis-none --log-driver=none redis
688356c9c86109169c3b378139abd7a742e59165069c722c821b41eb18f7eff7
```

Which Config?
The inspect command allows you to identify the logging configuration for a particular container. The command below will output the LogConfig section for each of the containers.

Server created in step 1

docker inspect --format '{{ .HostConfig.LogConfig }}' redis-server

Server created in step 2

docker inspect --format '{{ .HostConfig.LogConfig }}' redis-syslog

Server created in this step

docker inspect --format '{{ .HostConfig.LogConfig }}' redis-none

```
$ docker inspect --format '{{ .HostConfig.LogConfig }}' redis-server
{json-file map[]}
$ docker inspect --format '{{ json .HostConfig.LogConfig }}' redis-server
{"Type":"json-file","Config":{}}
$ docker inspect --format '{{ .HostConfig.LogConfig }}' redis-syslog
{syslog map[]}
$ docker inspect --format '{{ json .HostConfig.LogConfig }}' redis-syslog
{"Type":"syslog","Config":{}}
$ docker inspect --format '{{ .HostConfig.LogConfig }}' redis-none
{none map[]}
$ docker inspect --format '{{ json .HostConfig.LogConfig }}' redis-none
{"Type":"none","Config":{}}
```