View logs for a container or service

The docker logs command shows information logged by a running container. The docker service logs command shows information logged by all containers participating in a service. The information that is logged and the format of the log depends almost entirely on the container’s endpoint command.

By default, docker logs or docker service logs shows the command’s output just as it would appear if you ran the command interactively in a terminal. UNIX and Linux commands typically open three I/O streams when they run, called STDIN, STDOUT, and STDERR. STDIN is the commmand’s input stream, which may include input from the keyboard or input from another command. STDOUT is usually a command’s normal output, and STDERR is typically used to output error messages. By default, docker logsshows the command’s STDOUT and STDERR. To read more about I/O and Linux, see the [Linux Documentation Project article on I/O redirection](http://www.tldp.org/LDP/abs/html/io-redirection.html).

In some cases, docker logs may not show useful information unless you take additional steps.

* If you use a [logging driver](https://docs.docker.com/v17.09/engine/admin/logging/overview/) which sends logs to a file, an external host, a database, or another logging back-end, docker logsmay not show useful information.
* If your image runs a non-interactive process such as a web server or a database, that application may send its output to log files instead of STDOUT and STDERR.

In the first case, your logs are processed in other ways and you may choose not to use docker logs. In the second case, the official nginx image shows one workaround, and the official Apache httpd image shows another.

The official nginx image creates a symbolic link from /dev/stdout to /var/log/nginx/access.log, and creates another symbolic link from /dev/stderr to /var/log/nginx/error.log, overwriting the log files and causing logs to be sent to the relevant special device instead. See the [Dockerfile](https://github.com/nginxinc/docker-nginx/blob/8921999083def7ba43a06fabd5f80e4406651353/mainline/jessie/Dockerfile" \l "L21-L23).

The official httpd driver changes the httpd application’s configuration to write its normal output directly to /proc/self/fd/1(which is STDOUT) and its errors to /proc/self/fd/2 (which is STDERR). See the [Dockerfile](https://github.com/docker-library/httpd/blob/b13054c7de5c74bbaa6d595dbe38969e6d4f860c/2.2/Dockerfile" \l "L72-L75).

# Configure logging drivers

*Estimated reading time: 6 minutes*

Docker includes multiple logging mechanisms to help you [get information from running containers and services](https://docs.docker.com/v17.09/engine/admin/logging/view_container_logs/). These mechanisms are called logging drivers.

Each Docker daemon has a default logging driver, which each container uses unless you configure it to use a different logging driver.

In addition to using the logging drivers included with Docker, you can also implement and use [logging driver plugins](https://docs.docker.com/v17.09/engine/admin/logging/plugins/). Logging driver plugins are available in Docker 17.05 and higher.

## Configure the default logging driver

To configure the Docker daemon to default to a specific logging driver, set the value of log-driver to the name of the logging driver in the daemon.json file, which is located in /etc/docker/ on Linux hosts or C:\ProgramData\docker\config\ on Windows server hosts. The default logging driver is json-file. The following example explicitly sets the default logging driver to syslog:

{

"log-driver": "syslog"

}

If the logging driver has configurable options, you can set them in the daemon.json file as a JSON array with the key log-opts. The following example sets two configurable options on the json-file logging driver:

{

"log-driver": "json-file",

"log-opts": {

"labels": "production\_status",

"env": "os,customer"

}

}

If you do not specify a logging driver, the default is json-file. Thus, the default output for commands such as docker inspect <CONTAINER> is JSON.

To find the current default logging driver for the Docker daemon, run docker info and search for Logging Driver. You can use the following command on Linux, macOS, or PowerShell on Windows:

$ docker info |grep 'Logging Driver'

Logging Driver: json-file

## Configure the logging driver for a container

When you start a container, you can configure it to use a different logging driver than the Docker daemon’s default, using the --log-driver flag. If the logging driver has configurable options, you can set them using one or more instances of the --log-opt <NAME>=<VALUE> flag. Even if the container uses the default logging driver, it can use different configurable options.

The following example starts an Alpine container with the none logging driver.

$ docker run -it --log-driver none alpine ash

To find the current logging driver for a running container, if the daemon is using the json-file logging driver, run the following docker inspect command, substituting the container name or ID for <CONTAINER>:

$ docker inspect -f '{{.HostConfig.LogConfig.Type}}' <CONTAINER>

json-file

## Configure the delivery mode of log messages from container to log driver

Docker provides two modes for delivering messages from the container to the log driver:

* (default) direct, blocking delivery from container to driver
* non-blocking delivery that stores log messages in an intermediate per-container ring buffer for consumption by driver

The non-blocking message delivery mode prevents applications from blocking due to logging back pressure. Applications will likely fail in unexpected ways when STDERR or STDOUT streams block.

**WARNING**: When the buffer is full and a new message is enqueued, the oldest message in memory is dropped. Dropping messages is often preferred to blocking the log-writing process of an application.

The mode log option controls whether to use the blocking (default) or non-blocking message delivery.

The max-buffer-size log option controls the size of the ring buffer used for intermediate message storage when mode is set to non-blocking. max-buffer-size defaults to 1 megabyte.

The following example starts an Alpine container with log output in non-blocking mode and a 4 megabyte buffer:

$ docker run -it --log-opt mode=non-blocking --log-opt max-buffer-size=4m alpine ping 127.0.0.1

### Use environment variables or labels with logging drivers

Some logging drivers add the value of a container’s --env|-e or --label flags to the container’s logs. This example starts a container using the Docker daemon’s default logging driver (let’s assume json-file) but sets the environment variable os=ubuntu.

$ docker run -dit --label production\_status=testing -e os=ubuntu alpine sh

If the logging driver supports it, this adds additional fields to the logging output. The following output is generated by the json-filelogging driver:

"attrs":{"production\_status":"testing","os":"ubuntu"}

## Supported logging drivers

The following logging drivers are supported. See the link to each driver’s documentation for its configurable options, if applicable. If you are using [logging driver plugins](https://docs.docker.com/v17.09/engine/admin/logging/plugins/), you may see more options.

| **Driver** | **Description** |
| --- | --- |
| none | No logs will be available for the container and docker logs will not return any output. |
| [json-file](https://docs.docker.com/v17.09/engine/admin/logging/json-file/) | The logs are formatted as JSON. The default logging driver for Docker. |
| [syslog](https://docs.docker.com/v17.09/engine/admin/logging/syslog/) | Writes logging messages to the syslog facility. The syslog daemon must be running on the host machine. |
| [journald](https://docs.docker.com/v17.09/engine/admin/logging/journald/) | Writes log messages to journald. The journald daemon must be running on the host machine. |
| [gelf](https://docs.docker.com/v17.09/engine/admin/logging/gelf/) | Writes log messages to a Graylog Extended Log Format (GELF) endpoint such as Graylog or Logstash. |
| [fluentd](https://docs.docker.com/v17.09/engine/admin/logging/fluentd/) | Writes log messages to fluentd (forward input). The fluentd daemon must be running on the host machine. |
| [awslogs](https://docs.docker.com/v17.09/engine/admin/logging/awslogs/) | Writes log messages to Amazon CloudWatch Logs. |
| [splunk](https://docs.docker.com/v17.09/engine/admin/logging/splunk/) | Writes log messages to splunk using the HTTP Event Collector. |
| [etwlogs](https://docs.docker.com/v17.09/engine/admin/logging/etwlogs/) | Writes log messages as Event Tracing for Windows (ETW) events. Only available on Windows platforms. |
| [gcplogs](https://docs.docker.com/v17.09/engine/admin/logging/gcplogs/) | Writes log messages to Google Cloud Platform (GCP) Logging. |
| [logentries](https://docs.docker.com/v17.09/engine/admin/logging/logentries/) | Writes log messages to Rapid7 Logentries. |