Amazon CloudWatch Logs logging driver

- Usage
- Amazon CloudWatch Logs options
  - awslogs-region
  - awslogs-endpoint
  - awslogs-group
  - awslogs-stream
  - awslogs-create-group
  - awslogs-create-stream
  - awslogs-datetime-format
  - awslogs-multiline-pattern
  - tag
  - awslogs-force-flush-interval-seconds
  - awslogs-max-buffered-events
- Credentials

The `awslogs` logging driver sends container logs to Amazon CloudWatch Logs. Log entries can be retrieved through the AWS Management Console or the AWS SDKs and Command Line Tools.

Usage

To use the `awslogs` driver as the default logging driver, set the `log-driver` and `log-opt` keys to appropriate values in the `daemon.json` file. For more about configuring Docker using `daemon.json`, see daemon.json.

{{% include "daemon-cfg-desktop.md" %}}

The following example sets the log driver to `awslogs` and sets the `awslogs-region` option.

```
{
  "log-driver": "awslogs",
  "log-opts": {
    "awslogs-region": "us-east-1"
  }
}
```

Restart Docker for the changes to take effect.

You can set the logging driver for a specific container by using the `--log-driver` option to `docker run`:

```
$ docker run --log-driver=awslogs ...
```

If you are using Docker Compose, set `awslogs` using the following declaration example:

```
myservice:
  logging:
    driver: awslogs
    options:
      awslogs-region: us-east-1
```

Amazon CloudWatch Logs options

You can add logging options to the `daemon.json` to set Docker-wide defaults, or use the `--log-opt NAME=VALUE` flag to specify Amazon CloudWatch Logs logging driver options when starting a container.

awslogs-region

The `awslogs` logging driver sends your Docker logs to a specific region. Use the `awslogs-region` log option or the `AWS_REGION` environment variable to set the region. By default, if your Docker daemon is running on an EC2 instance and no region is set, the driver uses the instance's region.

```
$ docker run --log-driver=awslogs --log-opt awslogs-region=us-east-1 ...
```

awslogs-endpoint

By default, Docker uses either the `awslogs-region` log option or the detected region to construct the remote CloudWatch Logs API endpoint. Use the `awslogs-endpoint` log option to override the default endpoint with the provided endpoint.

> [!NOTE]
>
> The `awslogs-region` log option or detected region controls the region used for signing. You may experience signature errors if the endpoint you've specified with `awslogs-endpoint` uses a different region.

awslogs-group

You must specify a log group for the `awslogs` logging driver. You can specify the log group with the `awslogs-group` log option:

```
$ docker run --log-driver=awslogs --log-opt awslogs-region=us-east-1 --log-opt awslogs-group=myLogGroup ...
```

awslogs-stream

To configure which log stream should be used, you can specify the `awslogs-stream` log option. If not specified, the container ID is used as the log stream.

> [!NOTE]
>
> Log streams within a given log group should only be used by one container at a time. Using the same log stream for multiple containers concurrently can cause reduced logging performance.

awslogs-create-group

Log driver returns an error by default if the log group doesn't exist. However, you can set the `awslogs-create-group` to `true` to automatically create the log group as needed. The `awslogs-create-group` option defaults to `false`.

```
$ docker run \
    --log-driver=awslogs \
    --log-opt awslogs-region=us-east-1 \
    --log-opt awslogs-group=myLogGroup \
    --log-opt awslogs-create-group=true \
    ...
```

> [!NOTE]
>
> Your AWS IAM policy must include the `logs:CreateLogGroup` permission before you attempt to use `awslogs-create-group`.

awslogs-create-stream

By default, the log driver creates the AWS CloudWatch Logs stream used for container log persistence.

Set `awslogs-create-stream` to `false` to disable log stream creation. When disabled, the Docker daemon assumes the log stream already exists. A use case where this is beneficial is when log stream creation is handled by another process avoiding redundant AWS CloudWatch Logs API calls.

If `awslogs-create-stream` is set to `false` and the log stream does not exist, log persistence to CloudWatch fails during container runtime, resulting in `Failed to put log events` error messages in daemon logs.

```
$ docker run \
    --log-driver=awslogs \
    --log-opt awslogs-region=us-east-1 \
    --log-opt awslogs-group=myLogGroup \
    --log-opt awslogs-stream=myLogStream \
    --log-opt awslogs-create-stream=false \
    ...
```

awslogs-datetime-format

The `awslogs-datetime-format` option defines a multi-line start pattern in Python `strftime` format. A log message consists of a line that matches the pattern and any following lines that don't match the pattern. Thus the matched line is the delimiter between log messages.

One example of a use case for using this format is for parsing output such as a stack dump, which might otherwise be logged in multiple entries. The correct pattern allows it to be captured in a single entry.

This option always takes precedence if both `awslogs-datetime-format` and `awslogs-multiline-pattern` are configured.

> [!NOTE]
>
> Multi-line logging performs regular expression parsing and matching of all log messages, which may have a negative impact on logging performance.

Consider the following log stream, where new log messages start with a timestamp:

```
[May 01, 2017 19:00:01] A message was logged
[May 01, 2017 19:00:04] Another multi-line message was logged
Some random message
with some random words
[May 01, 2017 19:01:32] Another message was logged
```

The format can be expressed as a `strftime` expression of `[%b %d, %Y %H:%M:%S]`, and the `awslogs-datetime-format` value can be set to that expression:

```
$ docker run \
    --log-driver=awslogs \
    --log-opt awslogs-region=us-east-1 \
    --log-opt awslogs-group=myLogGroup \
    --log-opt awslogs-datetime-format='\[%b %d, %Y %H:%M:%S\]' \
    ...
```

This parses the logs into the following CloudWatch log events:

```
# First event
[May 01, 2017 19:00:01] A message was logged

# Second event
[May 01, 2017 19:00:04] Another multi-line message was logged
Some random message
with some random words

# Third event
```

```
[May 01, 2017 19:01:32] Another message was logged
```

The following `strftime` codes are supported:

Code	Meaning	Example
`%a`	Weekday abbreviated name.	Mon
`%A`	Weekday full name.	Monday
`%w`	Weekday as a decimal number where 0 is Sunday and 6 is Saturday.	0
`%d`	Day of the month as a zero-padded decimal number.	08
`%b`	Month abbreviated name.	Feb
`%B`	Month full name.	February
`%m`	Month as a zero-padded decimal number.	02
`%Y`	Year with century as a decimal number.	2008
`%y`	Year without century as a zero-padded decimal number.	08
`%H`	Hour (24-hour clock) as a zero-padded decimal number.	19
`%I`	Hour (12-hour clock) as a zero-padded decimal number.	07
`%p`	AM or PM.	AM
`%M`	Minute as a zero-padded decimal number.	57
`%S`	Second as a zero-padded decimal number.	04
`%f`	Microseconds as a zero-padded decimal number.	000345
`%z`	UTC offset in the form +HHMM or -HHMM.	+1300
`%Z`	Time zone name.	PST
`%j`	Day of the year as a zero-padded decimal number.	363

In addition, the following non-`strftime` codes are supported:

Code	Meaning	Example

`%L`	Milliseconds as a zero-padded decimal number preceded with a period.	.123

awslogs-multiline-pattern

The `awslogs-multiline-pattern` option defines a multi-line start pattern using a regular expression. A log message consists of a line that matches the pattern and any following lines that don't match the pattern. Thus the matched line is the delimiter between log messages.

This option is ignored if `awslogs-datetime-format` is also configured.

> [!NOTE]
>
> Multi-line logging performs regular expression parsing and matching of all log messages. This may have a negative impact on logging performance.

Consider the following log stream, where each log message should start with the pattern `INFO`:

```
INFO A message was logged
INFO Another multi-line message was logged
     Some random message
INFO Another message was logged
```

You can use the regular expression of `^INFO`:

```
$ docker run \
    --log-driver=awslogs \
    --log-opt awslogs-region=us-east-1 \
    --log-opt awslogs-group=myLogGroup \
    --log-opt awslogs-multiline-pattern='^INFO' \
    ...
```

This parses the logs into the following CloudWatch log events:

```
# First event
INFO A message was logged

# Second event
INFO Another multi-line message was logged
```

```
    Some random message

# Third event
INFO Another message was logged
```

tag

Specify `tag` as an alternative to the `awslogs-stream` option. `tag` interprets Go template markup, such as `{{.ID}}`, `{{.FullID}}` or `{{.Name}}` `docker.{{.ID}}`. See the tag option documentation for details on supported template substitutions.

When both `awslogs-stream` and `tag` are specified, the value supplied for `awslogs-stream` overrides the template specified with `tag`.

If not specified, the container ID is used as the log stream.

> [!NOTE]
>
> The CloudWatch log API doesn't support `:` in the log name. This can cause some issues when using the `{{ .ImageName }}` as a tag, since a Docker image has a format of `IMAGE:TAG`, such as `alpine:latest`. Template markup can be used to get the proper format. To get the image name and the first 12 characters of the container ID, you can use:
>
> ```
> --log-opt tag='{{ with split .ImageName ":" }}{{join . "_"}}{{end}}-{{.ID}}'
> ```
>
> the output is something like: `alpine_latest-bf0072049c76`

awslogs-force-flush-interval-seconds

The `awslogs` driver periodically flushes logs to CloudWatch.

The `awslogs-force-flush-interval-seconds` option changes log flush interval seconds.

Default is 5 seconds.

awslogs-max-buffered-events

The `awslogs` driver buffers logs.

The `awslogs-max-buffered-events` option changes log buffer size.

Default is 4K.

Credentials

You must provide AWS credentials to the Docker daemon to use the `awslogs` logging driver. You can provide these credentials with the `AWS_ACCESS_KEY_ID`, `AWS_SECRET_ACCESS_KEY`, and `AWS_SESSION_TOKEN` environment variables, the default AWS shared credentials file (`~/.aws/credentials` of the root user), or if you are running the Docker daemon on an Amazon EC2 instance, the Amazon EC2 instance profile.

> [!NOTE] Docker reads AWS credentials when the container starts. If you use a shared AWS credentials file with temporary credentials, updating the file later does not automatically update the credentials used by the running container. When the temporary credentials expire, log delivery to Amazon CloudWatch Logs can fail. Restart the container after refreshing the credentials so Docker can load the updated values.

Credentials must have a policy applied that allows the `logs:CreateLogStream` and `logs:PutLogEvents` actions, as shown in the following example.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Action": ["logs:CreateLogStream", "logs:PutLogEvents"],
      "Effect": "Allow",
      "Resource": "*"
    }
  ]
}
```

□□□ #2
□ □□□ □□□ 2026-10-31 19:24:22 UTC
□ □□□ □□□ 2026-10-31 19:24:23 UTC