

After you [set up debugging](/debugger/docs/setup) in Stackdriver Debugger and deploy or start your app, you can add logpoints in the [source console](https://source.cloud.google.com).

Logpoints let you inject logging into running services without restarting or interfering with the normal function of the service. Every time any instance runs code at the logpoint location, Debugger logs a message. The log output is sent to the appropriate log for the target environment. For App Engine, for example, output is sent to the request log in Stackdriver Logging.

Logpoints remain active for 24 hours after creation, or until they are deleted or the service is redeployed.

ints might not be available during startup while debugger initializes.

1. In the Google Cloud Console, open Cloud Source Repositories.

[Open Cloud Source Repositories](https://source.cloud.google.com/repos)

The **All repositories** page opens. Alternatively, you can open the [My source](/source-repositories/docs/browsing-repositories#browsing_to_a_repository) view.

2. Click the name of a repository.
3. Go to the file that contains source code you want to watch.
4. Click the line number of the source code location.

```

80 def draw_mandelbrot(width, height,
81                     left, right, top, bottom,
82                     iterations,
83                     z0):
84     """Returns a PIL.Image representing the given portion of the set."""
85     x, y = numpy.meshgrid(numpy.linspace(left, right, width),
86                          numpy.linspace(top, bottom, height))
87     c = x + 1j*y
88     z = c.copy()
89     z[:] = z0

```

5. When prompted, select the app where you want to add the logpoint.

Select an application to start debugging

Select an application ▼

i If you can't see your application, verify that it's associated with a version of this repository. [Learn more](#) ↗

Cancel
Continue

6. Click **Create logpoint**.


[Create snapshot](#)
Capture variables at each point in the call stack

[Create logpoint](#)
Generate a message each time a line is executed

7. Where prompted, enter the logpoint condition and message.

if (x>100) logpoint("Value of x has exceeded limit.") i Info ▼ Cancel Add

8. Click **Add**.

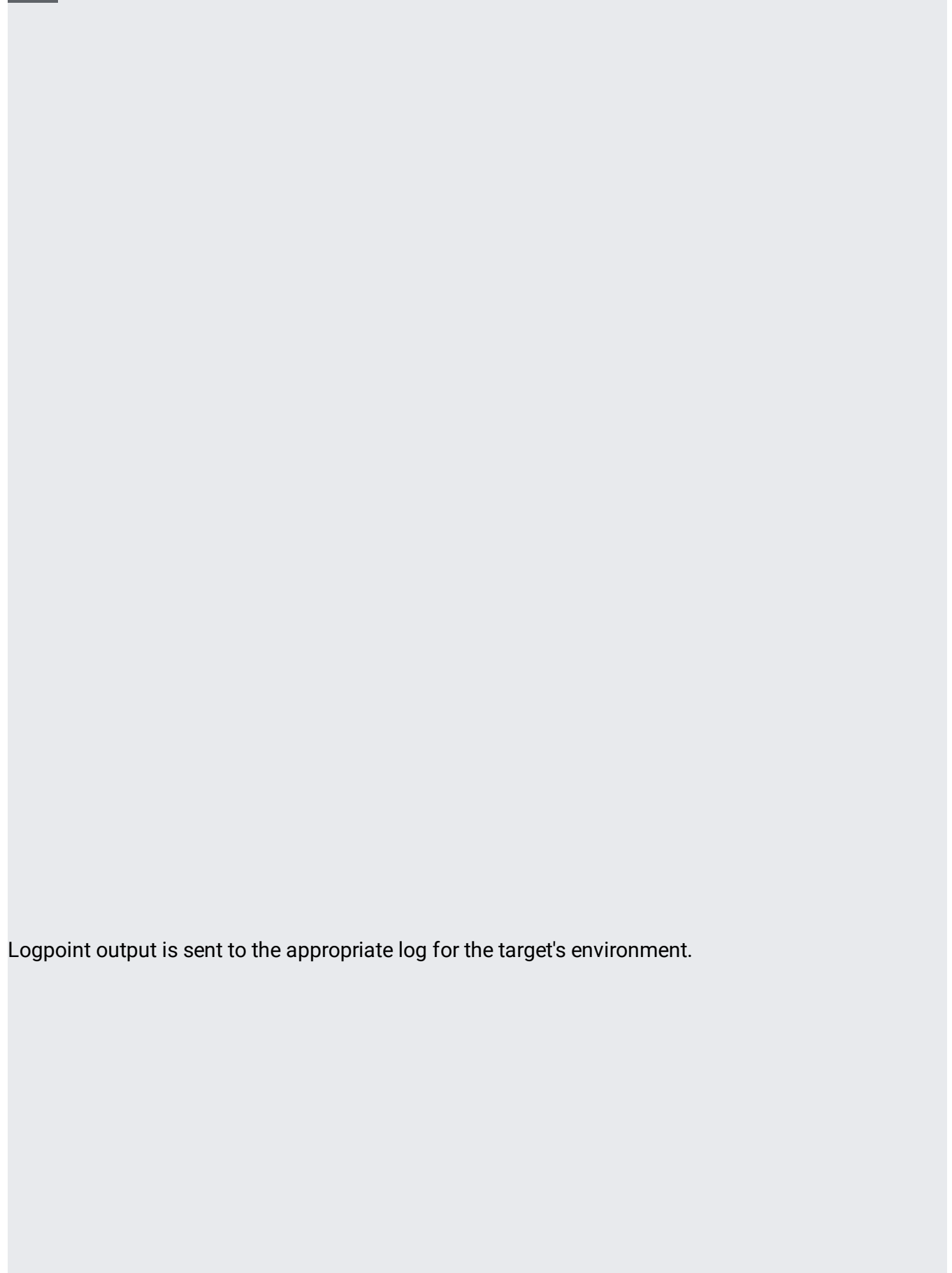
After you add a logpoint, the expression appears inline in the file view in the source console. To edit the logpoint, hold the pointer over it and click **Edit** .

A logpoint condition is a simple expression that must evaluate to `true` for the logpoint to be logged. Logpoint conditions are evaluated each time an instance runs the line of code until the logpoint expires or is deleted.

The condition is a full boolean expression that can include logical operators:

The message of a logpoint determines what gets logged in the output. Expressions let you evaluate and log values of interest. Anything in the message between curly braces, such as `{myObj.myFunc() }` or `{a + b}`, is replaced with the value of that expression in the output. The message `User {name} scored {newScore.score}` in the example above logs output similar to `User user1 scored 99`.


You can use the following language features for expressions.

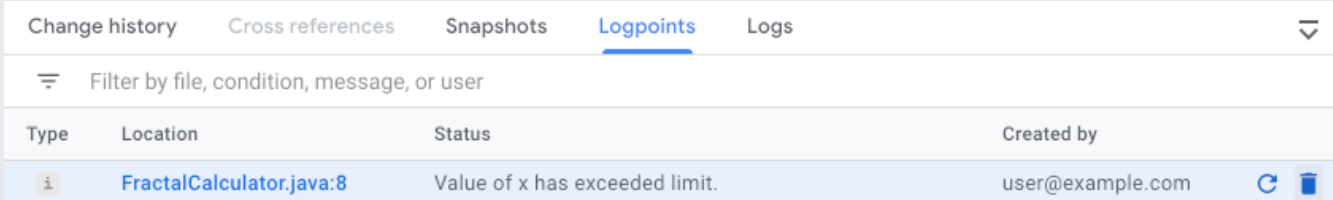





Logpoint output is sent to the appropriate log for the target's environment.

Logpoints become inactive and stop logging messages after 24 hours, and they are automatically deleted after 30 days. You can manually delete logpoints, which both stops the logging and removes the logging from the history for future reference. However, deleting a logpoint doesn't delete the log messages already generated.

To delete a logpoint, follow this process:

In the **Logpoints** tab in the bottom pane of the GCP Console, hold the pointer over the logpoint, and then click **Delete** 



Type	Location	Status	Created by
	FractalCalculator.java:8	Value of x has exceeded limit.	user@example.com  

- [Learn more about Logging \(/logging/docs/\)](#).