Debugging Functions

In this lesson, you will get to know how to debug Lambda functions using Chrome DevTools.

WE'LL COVER THE FOLLOWING

- Environments supported by SAM for debugging
- Google Chrome DevTools

Because Lambda functions run on an auto-scaling architecture and users don't really control how many instances run at any given time, it's not possible to debug them remotely. However, because of the transient nature of Lambda functions, it's relatively easy to debug functions locally. You can capture remote events, for example from logs, as in the previous section and replay them in a local environment. SAM command line tools can also generate test events for popular services, so you do not have to capture them from logs. For more information, check out the section *Generating test events* in Chapter 9.

Environments supported by SAM for debugging

Of course, you could send an event directly to the function handler from an integrated development environment (IDE) or a unit testing tool. That is a great option for quick experiments or even automated regression tests. For more integrated tests, using SAM local invocation becomes incredibly useful. You can pass `-d` followed by a port number, and SAM will set up a debug session on the specified port; you can then use standard remote debugging tools. At the time this was written, in early 2019, SAM supported debugging JavaScript, Python and GO code. If your functions are for a different runtime, you will still be able to invoke them locally, but may not be able to debug them. For up-to-date information on debugging support, check out the page Step-through Debugging Lambda Functions Locally in the AWS SAM developer

guide.

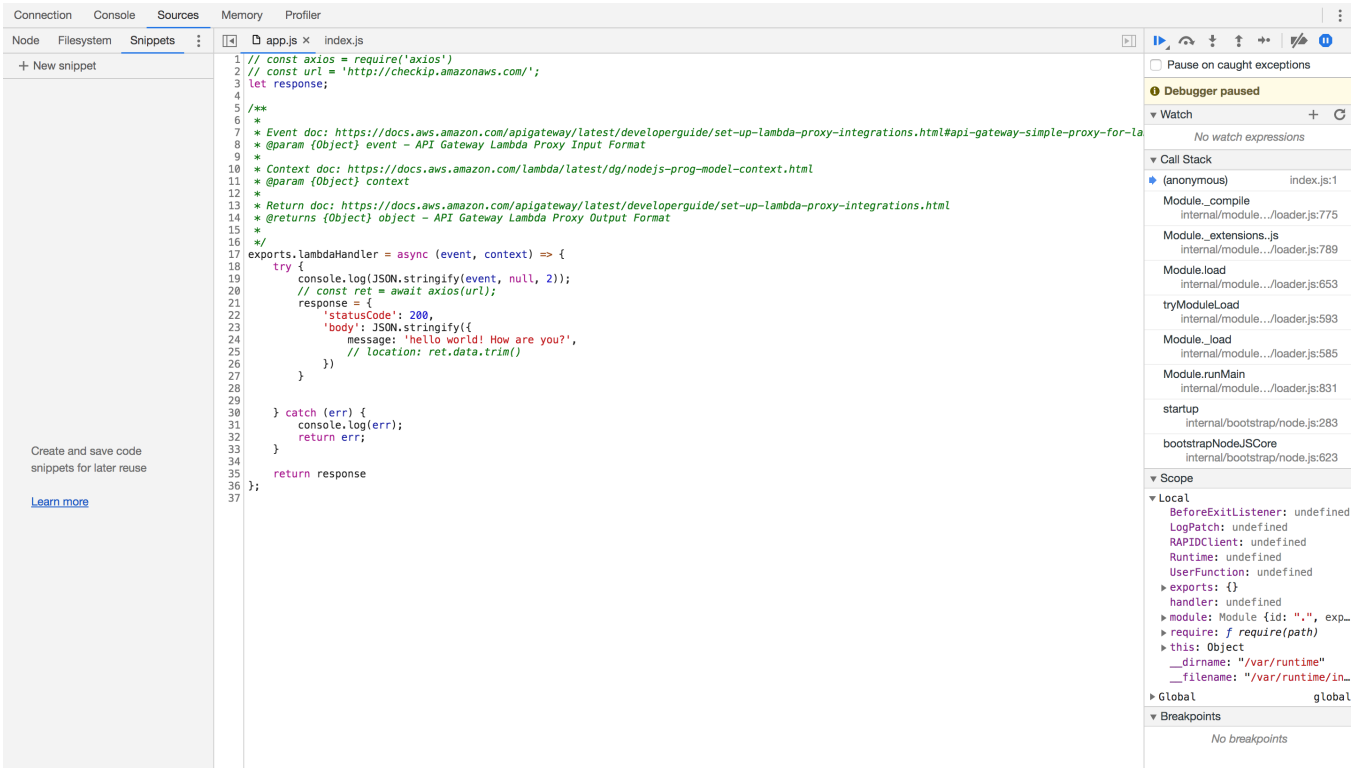
Google Chrome DevTools

For JavaScript runtimes, SAM sets up the debugging session with the Node.js Inspector protocol. Most IDE tools can connect to that debugging session as well as the Google Chrome DevTools. For example, the following command when run on your local machine would launch a debugging session on port 8080 and send an event saved in the previous section:

```
sam local invoke HelloWorldFunction --event event.json -d 8080
```

Of course, you could send an event directly to the function handler from an integrated development environment (IDE) or a unit testing tool. That is a great option for quick experiments or even automated regression tests. For more integrated tests, using SAM local invocation becomes incredibly useful. You can pass `-d` followed by a port number, and SAM will set up a debug session on the specified port; you can then use standard remote debugging tools. At the time this was written, SAM supported debugging JavaScript, Python and GO code. If your functions are for a different runtime, you will still be able to invoke them locally, but may not be able to debug them. For up-to-date information on debugging support, check out the page Step-through Debugging Lambda Functions Locally in the AWS SAM developer guide.

Using `sam local` and Docker makes the development flow significantly faster than deploying everything to AWS. Note, however, that the emulation is not perfect, so you may run into errors locally related to security roles and access to third-party resources for code that works perfectly fine when deployed to Lambda.



Chrome DevTools can connect to the debugging session set up by SAM.

That's it for debugging Lambda Functions. You can move on to the next lesson in which you will learn how to validate templates.