# Installing JavaScript and SAM CLI

This lesson contains a continuation of the setup process which includes installing Node.js and SAM CLI.

WE'LL COVER THE FOLLOWING



## Installing JavaScript tools #

### Node.js #

In this course, you'll be using JavaScript with Node.js for developing Lambda functions. (You do not need Node.js to work with SAM in a different language, but you will need the appropriate tools for that language.) To try examples from the book directly, you'll need Node.js 12 or later installed. To check whether Node.js is installed on your machine, run the following command:

```
node --version
```

Get the Node.js installer for your operating system from https://nodejs.org/ if this command prints an error.

#### Node Version Manager #

The latest Node.js version currently supported by Lambda is 12. You should get Node 12 even if you have a more recent installation. You can manage multiple versions of Node.js on the same system using the Node Version Manager.

Manager.

## Installing the SAM command-line tools #

### Using Homebrew or Linuxbrew #

There are several ways of installing SAM command-line tools. If you use Homebrew or Linuxbrew package management tools, you can install AWS SAM CLI using the following commands:

```
brew tap aws/tap
brew install aws-sam-cli
```

### Using pip #

Alternatively, use the `pip` package manager to download `sam`. Run the following command:

```
pip install aws-sam-cli
```

To check whether the installation worked, run the following command:

```
sam --version
```

If you get a response similar to the following, the software is ready:

```
$ sam --version
SAM CLI, version 0.40.0
```

If you get an error or the command is not found, you may need to restart the shell so that the new environment variables take effect. If that fails as well, check out the prerequisites again and then see the alternative installation instructions in the AWS SAM developer guide online.

> **Running in an isolated environment**
>
> For any Python developers reading this, SAM works perfectly fine in a virtual environment. My preferred way of installing tools such as SAM is to create an environment for each project, so the tools and libraries are completely separate. If you want to use an isolated installation and avoid changing system packages, just set up a new `virtualenv` environment

> before running `pip`.

In the next lesson, you will see how to configure AWS access credentials.