

THE LUA PROGRAMMING LANGUAGE BEGINNER'S GUIDE



In this article, we'll cover the basics of the Lua programming language, including:

- [What it is](#)
- [Main components](#)
- [Key features](#)
- [Pros & cons](#)
- [Use cases](#)
- [Lua vs other languages](#)
- [Installing Lua in a Windows environment](#)
- [Getting it up and running](#)

Let's get started.

What is Lua?



Lua is a robust, lightweight, and embeddable scripting language that supports multiple programming methods, including procedural, object-oriented, functional, and data-driven programming.

As the primary focus on Lua is for scripting, it is rarely used as a standalone programming language.

Instead, it is used as a scripting language that can be integrated (embedded) into other programs written in mainly C and C++. It also supports other programming languages via third-party plugins ([NLua](#)/[KeraLua](#) for .NET/C#).

Popular use cases for Lua include:

- As a popular component in video game and game engine development. For example, Warframe, World of Warcraft, and CRYENGINE all use Lua.
- As a programming language in many network programs, like CISCO Systems, Nmap, and ModSecurity.
- As a programming language in industrial programs such as Adobe Lightroom and MySQL Workbench.
- As a library that [developers](#) can integrate with their programs to enable scripting functionality.

Being a scripting language, Lua does not have its own main application. Instead, it acts exclusively as an embedded part of the host application.

(Explore the most [popular programming languages](#).)

How Lua works

There are two main components of Lua:

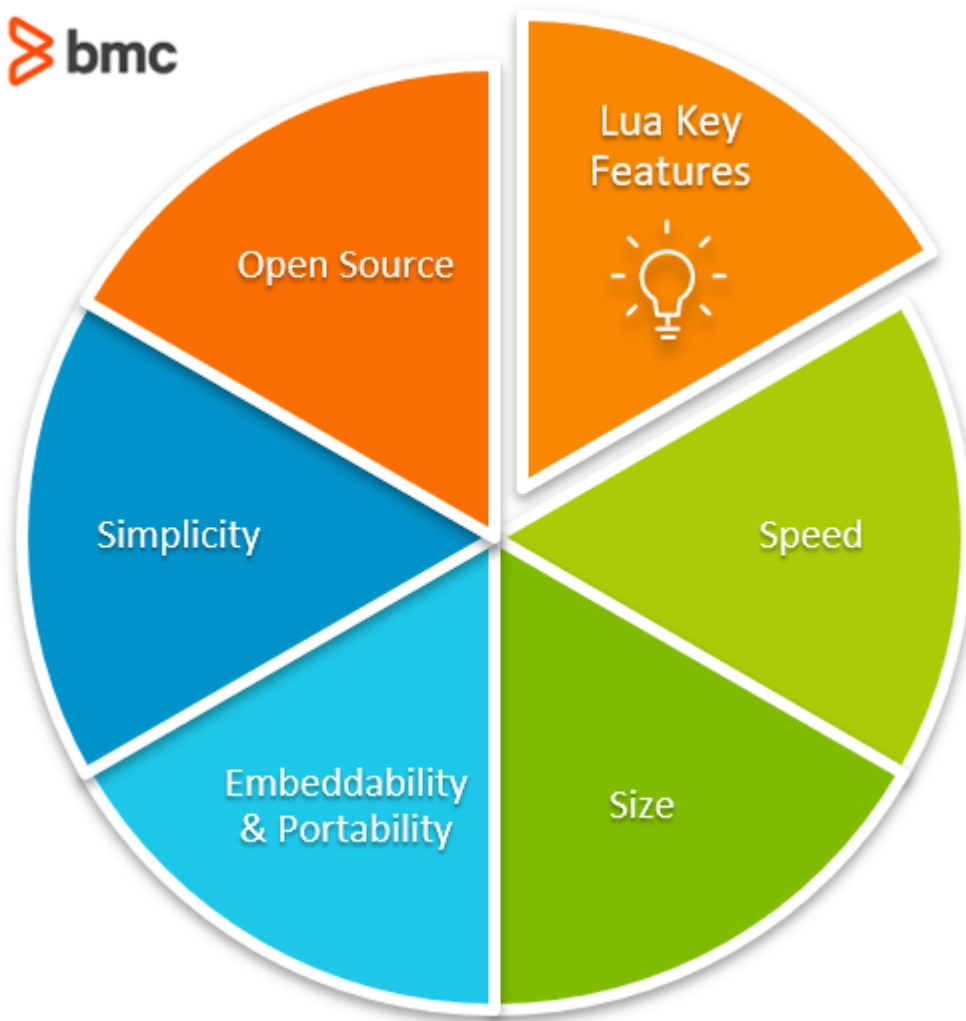
- The Lua interpreter
- The Lua virtual machine (VM)

Lua is not directly interpreted through a Lua file like other languages such as [Python](#). Instead, it uses the Lua interpreter to compile a Lua file to bytecode. The Lua interpreter is written in ANSI C, making it highly portable and capable of running on a multitude of devices.

Usually, the compilation is done at the runtime. However, sometimes it can be done before the runtime to increase load times. Then the Lua virtual machine will run this compiled bytecode. The register-based architecture of the Lua virtual machine closely resembles actual hardware architectures, and it will increase the overall performance of the program.

Key features of Lua

So, what's so great about Lua? These are the defining features:



Speed

Lua is considered one of the fastest programming languages among interpreted scripting languages. In particular, Lua can perform large task orders faster than most other programming languages in both benchmarking and real-world scenarios.

For more speed, an independent implementation of Lua called [LuaJIT](#) uses a just-in-time compiler that makes Lua even faster.

Size

Lua has a considerably smaller footprint than other programming languages, with its complete source code and documentation taking a mere 1.3 MB. The Lua interpreter with all the standard libraries takes 278K, while the complete Lua library takes only 466K.

This small size is ideal when integrating Lua into multiple platforms, from embedded devices to massive game engines, where every byte is valuable.

Portability & embeddability

With its small size, the portability of Lua is nearly unlimited; any platform that supports the standard C compiler can run Lua out of the box. Lua's speed and size become huge advantages when embedding Lua with another programming language. That's because they can help increase the

speed of the program without hindering any existing functionality.

Importantly, Lua does not require complex rewrites to be compatible with other programming languages. Lua can be used with primary programming languages like C, C++, Java, C#, etc.. and other scripting languages like Perl and Ruby, further extending its usability.

Simplicity

Lua is simple in design yet provides powerful functionality. One of the core features of Lua is meta-mechanisms which enable developers to implement features—rather than providing a bunch of features directly in the language itself.

Lua also comes with incremental garbage collection, reducing memory usage and implementation complexity. Its sandboxing feature can be used to isolate functions and resources. This increases the security of the program and provides coroutines for multitasking.

All these features come with a simple syntax and easily understandable format so that anyone can easily pick up Lua and use it in their programs.

License

Lua is free and open-source software distributed under the MIT license. This means anyone can use Lua for any purpose without paying any licensing or royalty fees.

Advantages & drawbacks

Like any language, Lua has its pros and cons.

Advantages of Lua

- **Easy app integration.** Its high performance and small size make it easy to integrate Lua into applications.
- **Simple syntax.** Relatively simple syntax structure with around 20 dedicated keywords, which helps to dive into Lua programming easily.
- **Flexibility.** Without standard libraries, you can customize Lua to meet any need.
- **Cross-platform compatibility and support** for the standard C compiler allows Lua to run virtually anywhere.
- **Dynamic variables** in Lua allow defining variables without defining types, and the type is determined automatically at the runtime.
- **Easy debugging.** Simple and powerful debug library.
- **Plenty of documentation.** Comprehensive documentation to get Lua projects up and running quickly and the active community.

Disadvantages of Lua

- Limited error handling support can lead to longer debug times to identify the exact errors in a Lua script.
- All variables are created as global variables (global scope), which can lead to errors in variable assignments.

- Limited pattern matching support.

When to use Lua

As a scripting language without major limitations, you can use Lua for any scenario, from a simple backend script in a web server to complex game development.

Lua is highly prevalent in video game development as it can be used to create functionality without contaminating the overall performance while also keeping everything separate.

Another area that Lua excels is embedded programming, where size and performance are major concerns. Lua can be used in everyday applications to extend the existing functionality or create new features and functions.

Some popular games, programs, and services that use Lua are Dark Souls, Fable II, Garry's Mod, Wireshark, VLC, Apache, and Nginx Web Servers.

Lua vs other languages

How does Lua stack up against other languages?

Here's a look at the differences between the high-level general programming language Python, the high-level object-oriented Java, and web-focused Javascript—all compared to Lua.

	Lua	Python	Java	JavaScript
Language Type	Scripting language	General purpose programming language	General purpose programming language	Web-focused programming language (Supports client- & server-side)
Usage	Embedded scripting	Multiple (From web development to data analytics)	Multiple (Desktop/mobile apps to Enterprise apps & services)	Web app development (Frontend & backend)
Compiled vs Interpreted	Compiled	Interpreted	Compiled	Interpreted
Package Management	Third-party tools (LuaRocks, LuaDist)	Pip	Maven & Gradle	NPM (Node Package Manager)
Programing Styles	<ul style="list-style-type: none">• Procedural• Object-oriented• Functional• Data-driven	<ul style="list-style-type: none">• Imperative• Functional• Object-oriented• Procedural	<ul style="list-style-type: none">• Object-oriented	<ul style="list-style-type: none">• Procedural• Object-oriented
Ease of use (Syntax)	Simple	Simple	Complex	Simple
Speed	Fastest	Relatively slow	Fast	Faster
Platform Support	Multi-platform	Multi-platform	Multi-platform	Multi-platform (Web oriented)

Installing Lua

Now, let's see how to set up a development environment in Windows. First, we'll install Lua.

Step 1

Navigate to the [Lua.org download page](https://lua.org/download). Here, we will be using a precompiled binary to install Lua in windows. So, click on "get a binary link" as shown in the below screenshot.



The screenshot shows a web browser window with the address bar displaying 'www.lua.org/download.html'. The page has a dark blue header with the 'Lua' logo and the word 'Download'. Below the header is a navigation bar with links: source · binaries · previews · logos · tools · test suites · extras · license · versions · donations · live demo. The main content is divided into two columns. The left column has sections for 'Source', 'Tools', and 'Giving credit'. The right column has sections for 'Building' and 'Supporting Lua'. A red box highlights a paragraph in the 'Building' section that says: 'If you don't have the time or the inclination to compile Lua yourself, [get a binary](#) or try the live demo.' The browser's status bar at the bottom shows the URL 'http://luabinaries.sourceforge.net', a 'Reset' button, and the time '09:46 PM'.

Source

Lua is free software distributed in source code. It may be used for any purpose, including commercial purposes, at absolutely no cost.

All versions are available for download. The current version is [Lua 5.4](#) and its current release is [Lua 5.4.3](#).

[lua-5.4.3.tar.gz](#)
2021-03-15, 350K
md5: ef63ed2ecfb713646a7fcc583cf5f352
sha1: 1dda2ef23a9828492b4595c0197766de6e784bc7

Tools

The main repository of Lua modules is [LuaRocks](#). See also [Awesome Lua](#). Pre-compiled Lua libraries and executables are available at [LuaBinaries](#). The [lua-users](#) wiki lists many user-contributed addons for Lua.

Giving credit

If you use Lua, please give us credit, according to our license. A nice way to give us further credit is to include a [Lua logo](#) and a link to our site in a web page for your product.

Building

Lua is implemented in pure ANSI C and compiles unmodified in all platforms that have an ANSI C compiler. Lua also compiles cleanly as C++.

Lua is very easy to build and install. There are [detailed instructions](#) in the package but here is a simple terminal session that downloads the current release of Lua and builds it in Linux:

```
curl -R -O http://www.lua.org/ftp/lua-5.4.3.tar.gz
tar xzf lua-5.4.3.tar.gz
cd lua-5.4.3
make all test
```

If you have trouble building Lua, [read the FAQ](#).

If you don't have the time or the inclination to compile Lua yourself, [get a binary](#) or try the live demo.

Supporting Lua

You can help to support the Lua project by buying a [book](#) published by Lua.org and by [making a donation](#).

You can also help to spread the word about Lua by buying Lua products at [Zazzle](#).

last update: mon mar 29 11:46:07 utc 2021

Step 2

Click "Download" on the LuaBinaries page, and you will be redirected to a page with a list of precompiled binaries. Select the appropriate version from that list.

We will be using the latest Lua version for Windows 64 bit.

The screenshot shows a web browser window with the address bar displaying `luabinaries.sourceforge.net/download.html`. The page features the Lua logo and the title "LuaBinaries" with the subtitle "Pre-compiled Lua libraries and executables." A left sidebar contains a navigation menu with links: Home, Overview, Motivation, Installation, History, Credits, Contact us, Manual, Configuration, Packaging, Download, and License. The main content area is titled "Download" and explains that binaries, source code, and documentation are available from the SourceForge project files page, providing the URL `https://sourceforge.net/projects/luabinaries/files/`. It then lists shortcuts for popular downloads, categorized by release: "LuaBinaries 5.4.2 - Release 1" and "LuaBinaries 5.3.6 - Release 1". Each release has a table of download links and their descriptions. A mouse cursor is hovering over the link `lua-5.4.2_Win64_bin.zip` in the 5.4.2 release table.

LuaBinaries
Pre-compiled Lua libraries and executables.

Download

All the binaries, source code and documentation are available from the SourceForge project files page:

<https://sourceforge.net/projects/luabinaries/files/>

But here are shortcuts for the most popular downloads:

LuaBinaries 5.4.2 - Release 1

lua-5.4.2_Sources.tar.gz	Source Code and Makefiles
lua-5.4.2_Sources.zip	Source Code and Makefiles
lua-5.4.2_Win32_bin.zip	Windows x86 Executables
lua-5.4.2_Win32_dllw6_lib.zip	Windows x86 DLL and Includes (MingW-w64 6 Built)
lua-5.4.2_Win64_bin.zip	Windows x64 Executables
lua-5.4.2_Win64_dllw6_lib.zip	Windows x64 DLL and Includes (MingW-w64 6 Built)
lua-5.4.2_MacOS1011_bin.tar.gz	MacOS X Intel Executables
lua-5.4.2_MacOS1011_lib.tar.gz	MacOS X Intel Library and Includes
lua-5.4.2_Linux54_64_bin.tar.gz	Linux x64 Executables
lua-5.4.2_Linux54_64_lib.tar.gz	Linux x64 Library and Includes

LuaBinaries 5.3.6 - Release 1

lua-5.3.6_Sources.tar.gz	Source Code and Makefiles
lua-5.3.6_Sources.zip	Source Code and Makefiles
lua-5.3.6_Win32_bin.zip	Windows x86 Executables
lua-5.3.6_Win32_dllw6_lib.zip	Windows x86 DLL and Includes (MingW 6 Built)
lua-5.3.6_Win64_bin.zip	Windows x64 Executables
lua-5.3.6_Win64_dllw6_lib.zip	Windows x64 DLL and Includes (MingW 6 Built)


This will direct the user to a SourceForge page, where the binary will be downloaded.

Download LuaBinaries from

sourceforge.net/projects/luabinaries/files/5.4.2/Tools Exe... 0 bytes 7/0

SOURCEFORGE


Home / Browse / Development / Interpreters / LuaBinaries

 **LuaBinaries**
Brought to you by: [carregal](#), [scuri](#)


Your download will start shortly... 3

Get Updates Share This Problems Downloading?

lua-5.4.2_Win64_bin.zip | Scanned for malware ✓

Mirror Provided by  [Learn more about CFH Cable](#)

Other Useful Business Software




Monitor Your Cisco ASA Like an Expert

See how **Network Insight™ for Cisco® ASA** improves device visibility in **SolarWinds® Network Performance Monitor** and **Network Configuration Manager**

You can get visibility into the health and performance of your Cisco ASA environment in a single dashboard. View VPN tunnel status and get help monitoring firewall high availability, health, and readiness. It's also designed to automatically discover and filter with ACLs, show rule hit counts, and more. [Expand](#)

[Download Free Trial](#)




Even the best code breaks. Fix it faster.

Code breaks. It's what it does. But don't let that stop you. Find & fix problems faster with all your telemetry in one place.

New Relic makes observability simple. Sign up for free today!

[Get started for free](#)



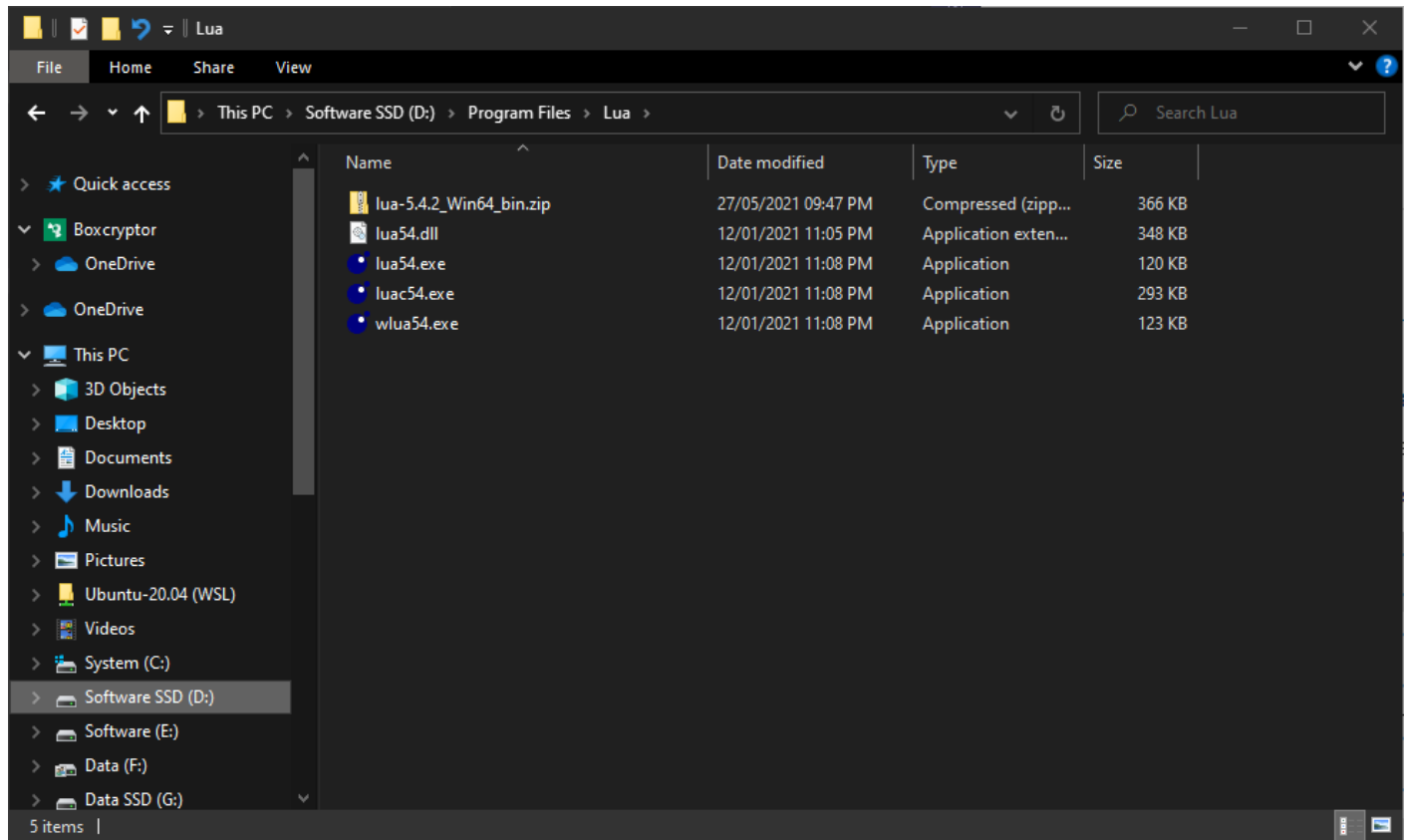
The solution to iOS 14 is more solutions

We have more engineers, more solutions, and more resources on **iOS 14** and attribution than anyone.

Step 3

Move the downloaded Zip file to any location to store the binaries permanently.

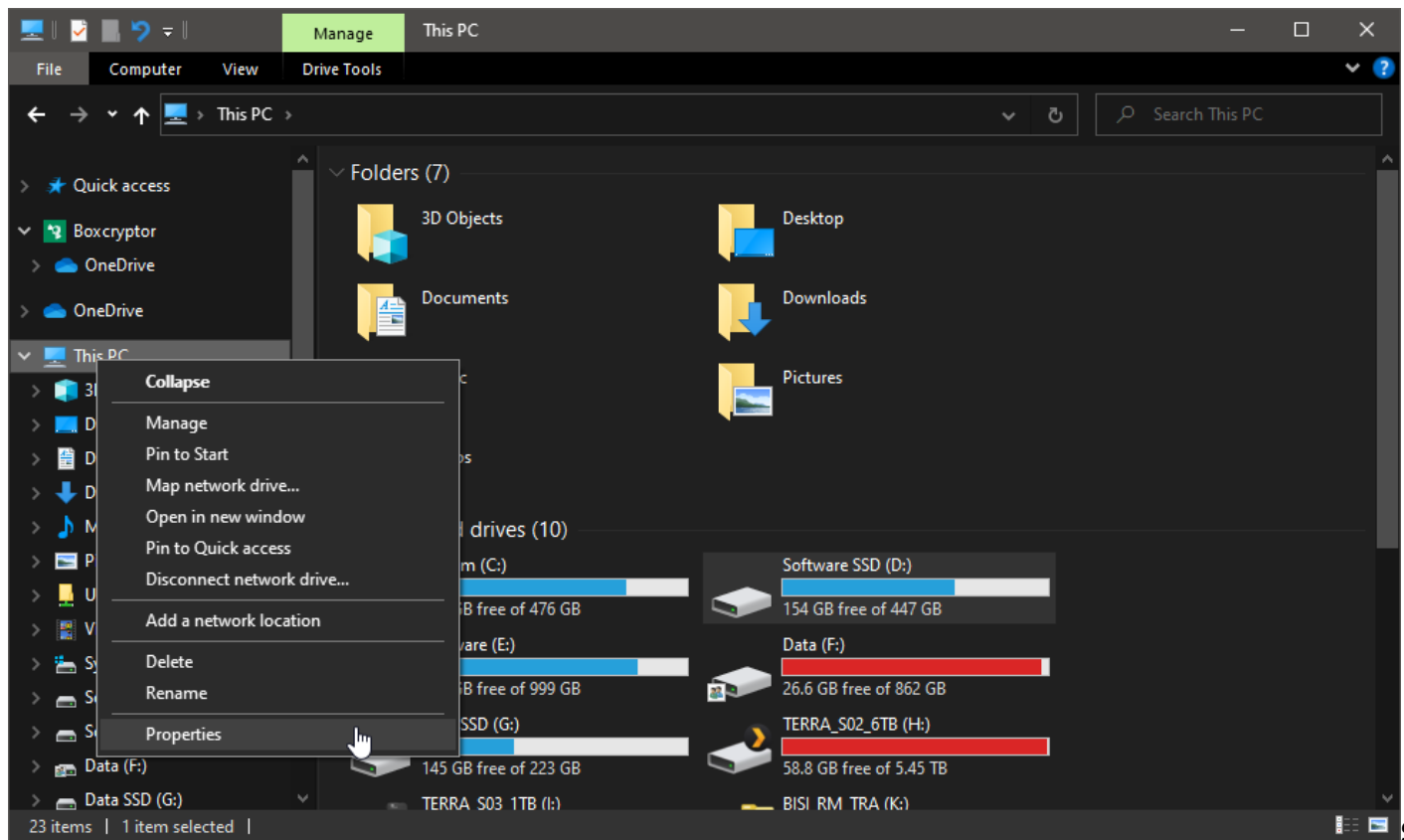
Here, we will be using the "D:\Program Files\Lua" as the location. After moving the Zip file, simply extract its content using any compression utility (Ex: Windows Explorer, 7zip, WinRar).



Step 4

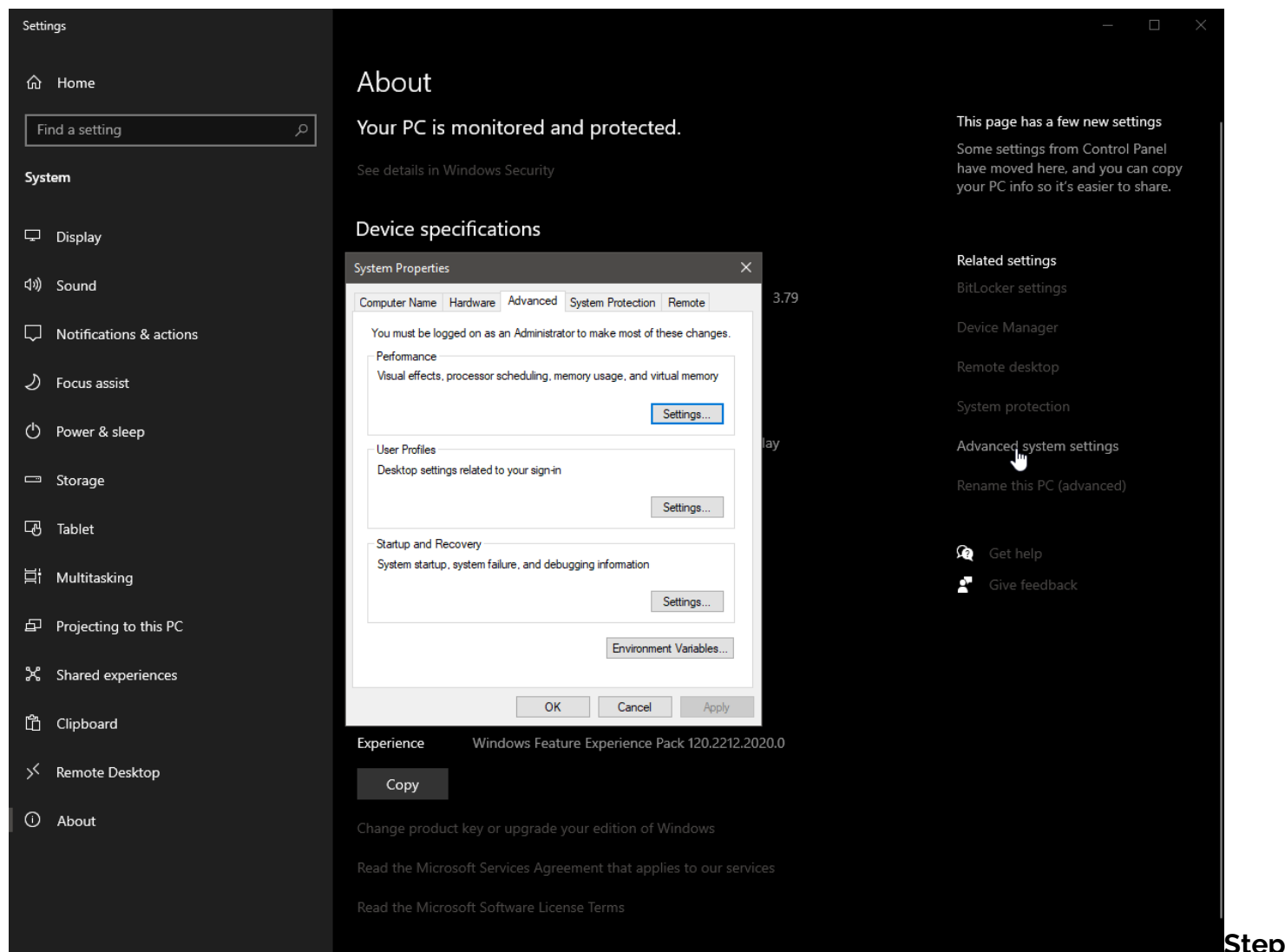
We need to add the location of Lua binaries to the system PATH so that Windows can call Lua from anywhere in the system.

Step 4.1. Navigate to Environment Variables. (Open Windows Explorer, right-click on This PC, and select properties.)



Step

4.2. Click on “Advanced System Settings” in the screen that appears and then click on “Environment Variables”.



Step

4.3. In the system variables section, add the location of the Lua executables as a new entry for the Path variable.

Environment Variables

User variables for bisin

Variable	Value
ANDROID_HOME	F:\Bisina\Downloads\Android\android-sdk
ANDROID_SDK_ROOT	F:\Bisina\Downloads\Android\android-sdk
BESIEGE_GAME_ASSEMBLIES	E:/Games/SteamLibrary/steamapps/common/Besiege/Besiege_Dat...
BESIEGE_UNITY_ASSEMBLIES	E:/Games/SteamLibrary/steamapps/common/Besiege/Besiege_Dat...
ChocolateyLastPathUpdate	132606530855437616
ChocolateyToolsLocation	C:\tools
FileBot	C:\Program Files\FileBot

New...

Edit...

Delete

System variables

Variable	Value
NUMBER_OF_PROCESSORS	12
OS	Windows_NT
Path	C:\Program Files\AdoptOpenJDK\jre-16.0.1.9-hotspot\bin;C:\Progr...
PATHEXT	.COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
POWERSHELL_DISTRIBUTIO...	MSI:Windows 10 Pro
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	AMD64 Family 23 Model 113 Stepping 0. AuthenticAMD

New...

Edit...

Delete

OK

Cancel

Edit environment variable

C:\Windows\System32\OpenSSH\
 C:\Program Files (x86)\NVIDIA Corporation\PhysX\Common
 C:\Program Files\NVIDIA Corporation\NVIDIA NvDLISR
 C:\ProgramData\chocolatey\bin
 C:\Program Files (x86)\Boxcryptor\bin\
 C:\Program Files\AdoptOpenJDK\jdk-11.0.4+11\bin
 C:\Program Files\dotnet\
 C:\Program F
 %M2_HOME%\bin
 C:\ProgramData\chocolatey\lib\maven-snapshot\apache-maven
 C:\Program Files\PowerShell\7\
 C:\tools\BCURRAN3
 C:\Program Files\Microsoft VS Code\bin
 C:\Program Files\WireGuard\
 C:\Program Files\Git\cmd
 C:\Program Files (x86)\dotnet\
 C:\Program Files\Amazon\AWSCLIV2\
 C:\Program Files\Docker\Docker\resources\bin
 C:\ProgramData\DockerDesktop\version-bin
 D:\Program Files\Lua

New

Edit

Browse...

Delete

Move Up

Move Down

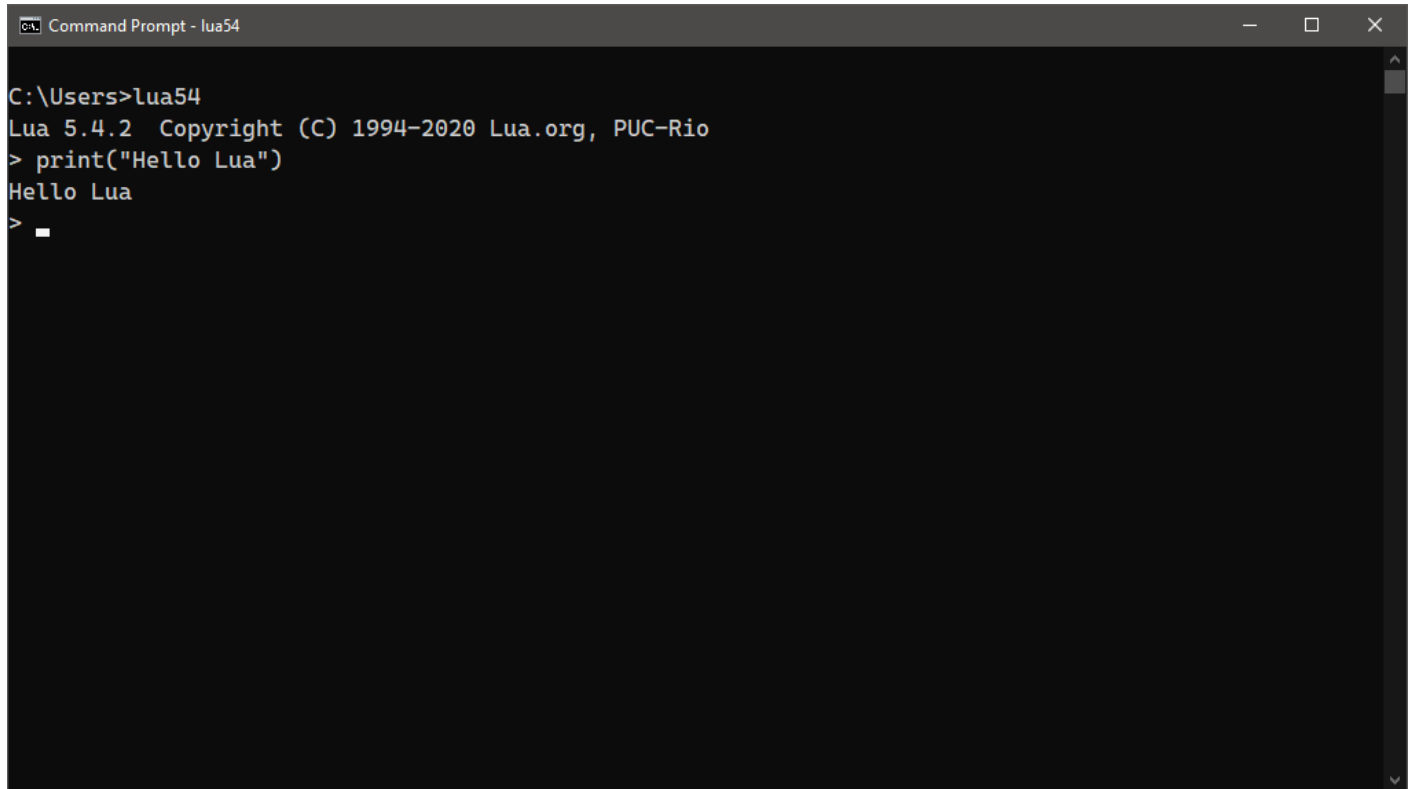
Edit text...

OK

Cancel

Step 5

Check if the system identifies Lua by opening up a command prompt or a PowerShell window and typing the Lua command (Lua with the version - lua54).



```
C:\Users>lua54
Lua 5.4.2 Copyright (C) 1994-2020 Lua.org, PUC-Rio
> print("Hello Lua")
Hello Lua
> _
```

Setting up a Lua development environment

Now that we have installed Lua in the system, we need a development environment to go ahead with coding. For that, we can choose between:

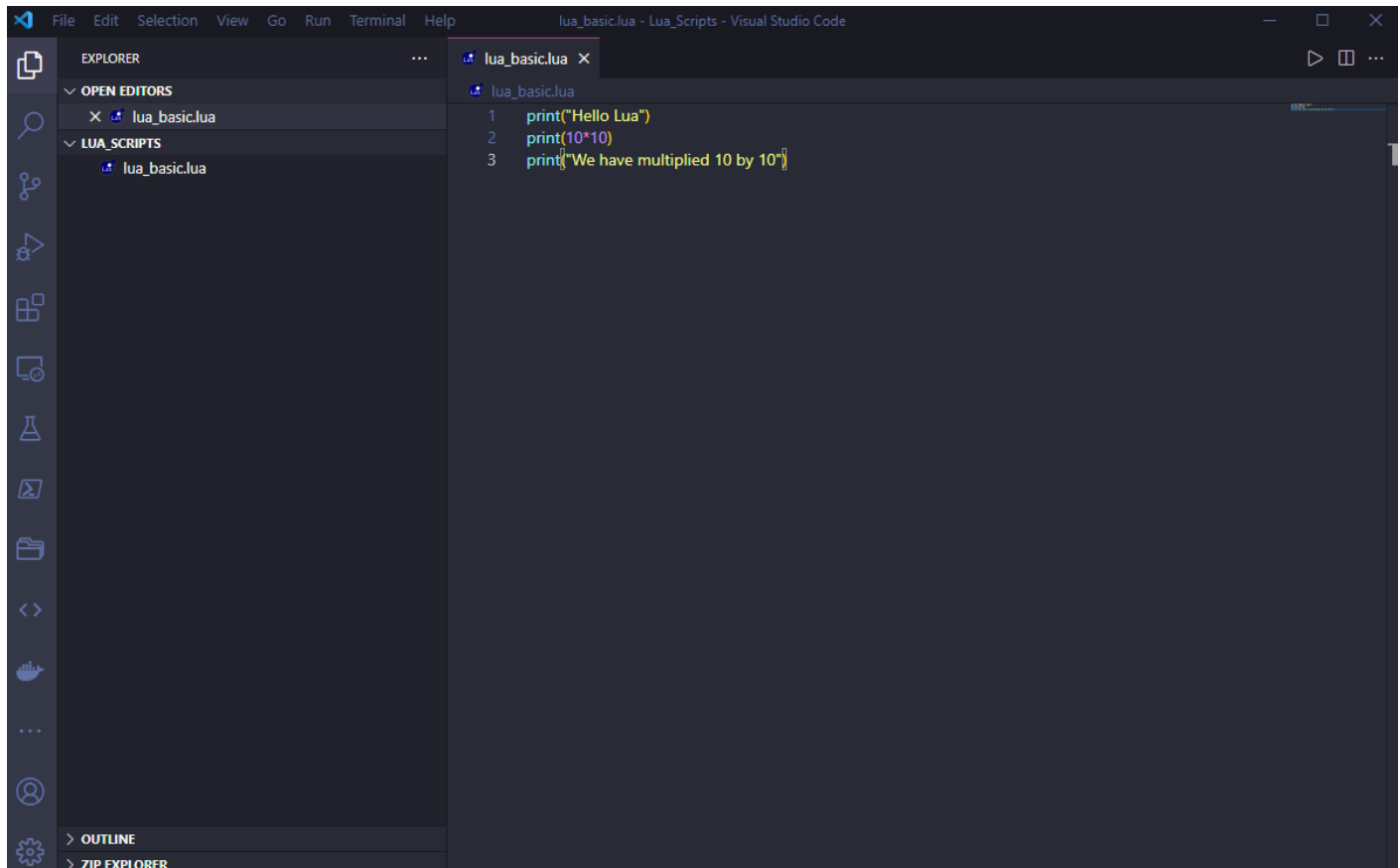
- A dedicated Lua IDE like [ZeroBrane Studio](#)
- A general IDE like [VSCode](#)

We will be using VSCode for this instance.

Step 1

Let's create a file called "lua_basic.lua" in VSCode and save that file in the desired location. Then we will type some print statements there like the following.

```
print("Hello Lua")
print(10*10)
print("We have multiplied 10 by 10")
```

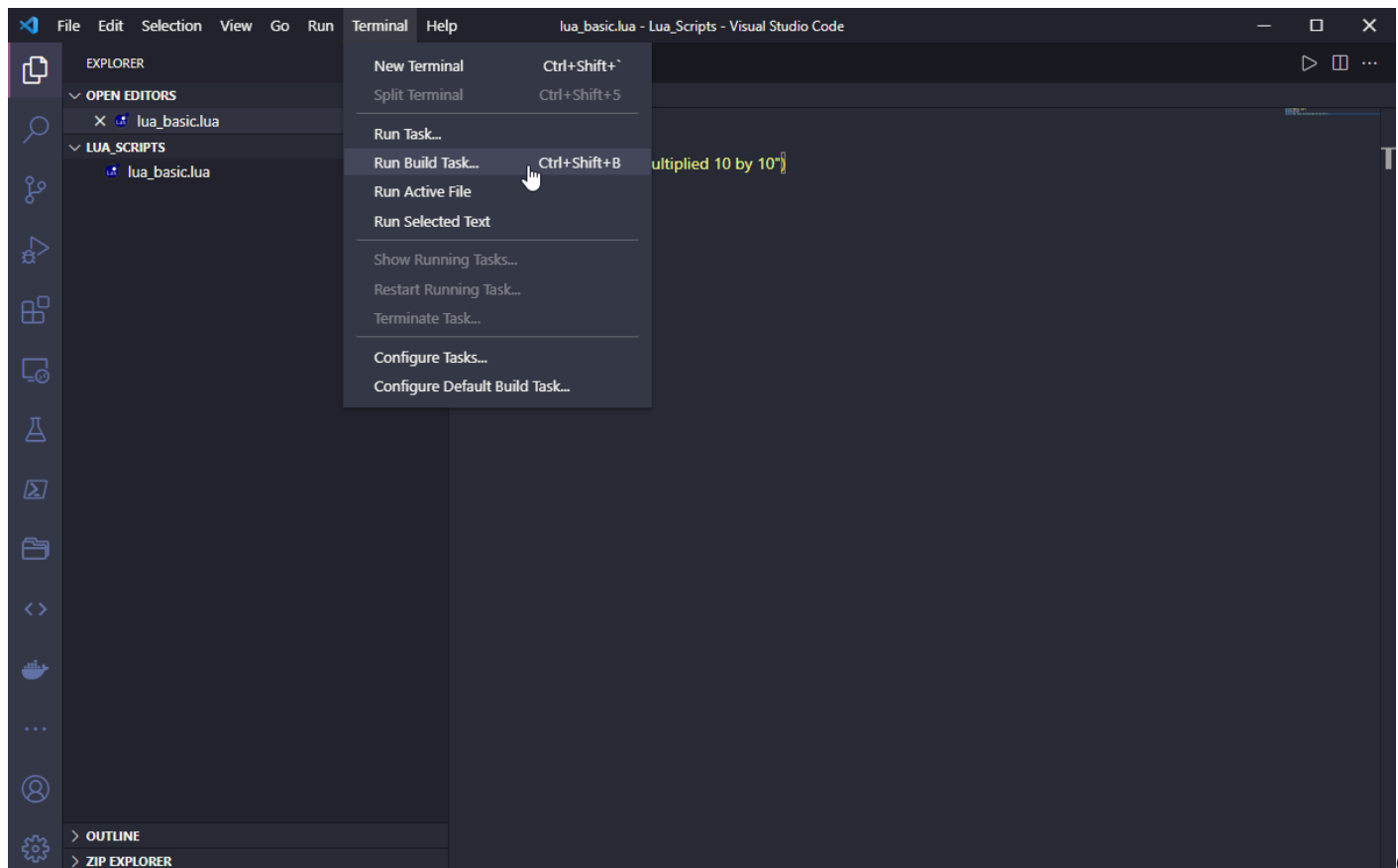



Step 2

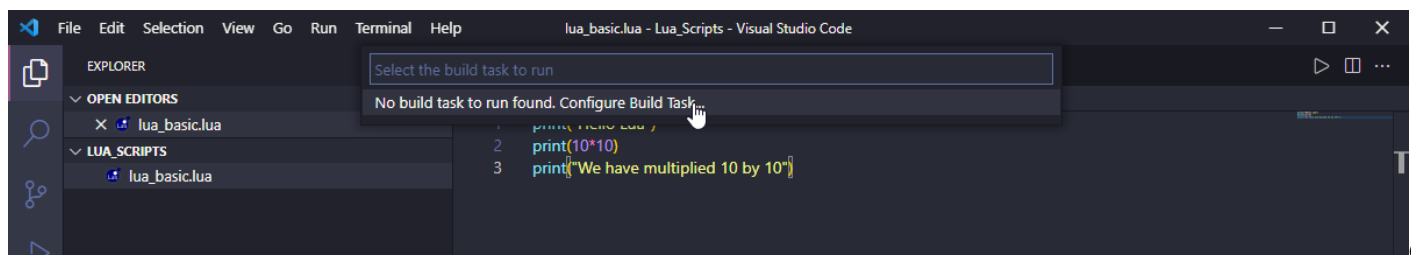
Our Lua program needs to be compiled before running, so we need to create a Build Task. To do that, click on Terminal Menu, then Run Built Task and select the Configure Build Task Option.

(There will be different build task templates depending on the VSCode configuration. Select "Create tasks.json from template" and finally the "Others" option to define a custom build task.)

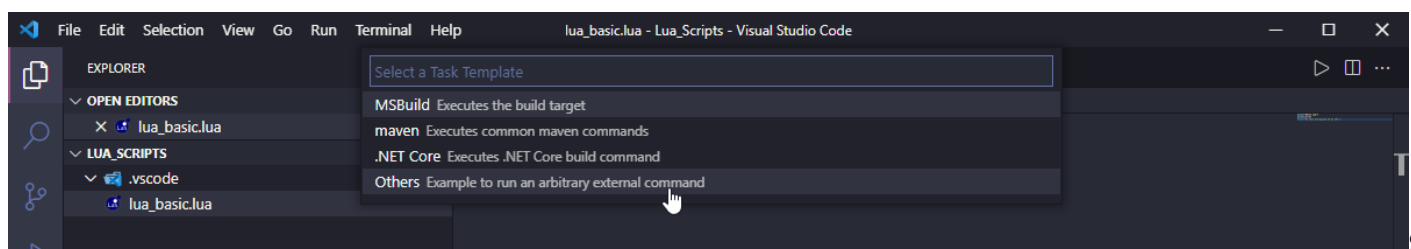
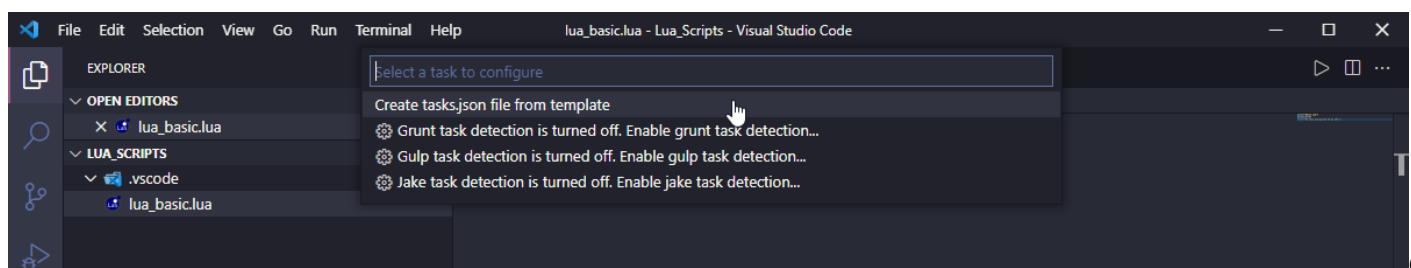
Run Build Task



Configure Build Task



Create tasks.json from template



2.1. We will add the following code block to configure the task. In that code block, we have defined a task called "Run Lua" that will run on the shell with the command "lua54". That command will take the current file as the argument and carry out a build operation.

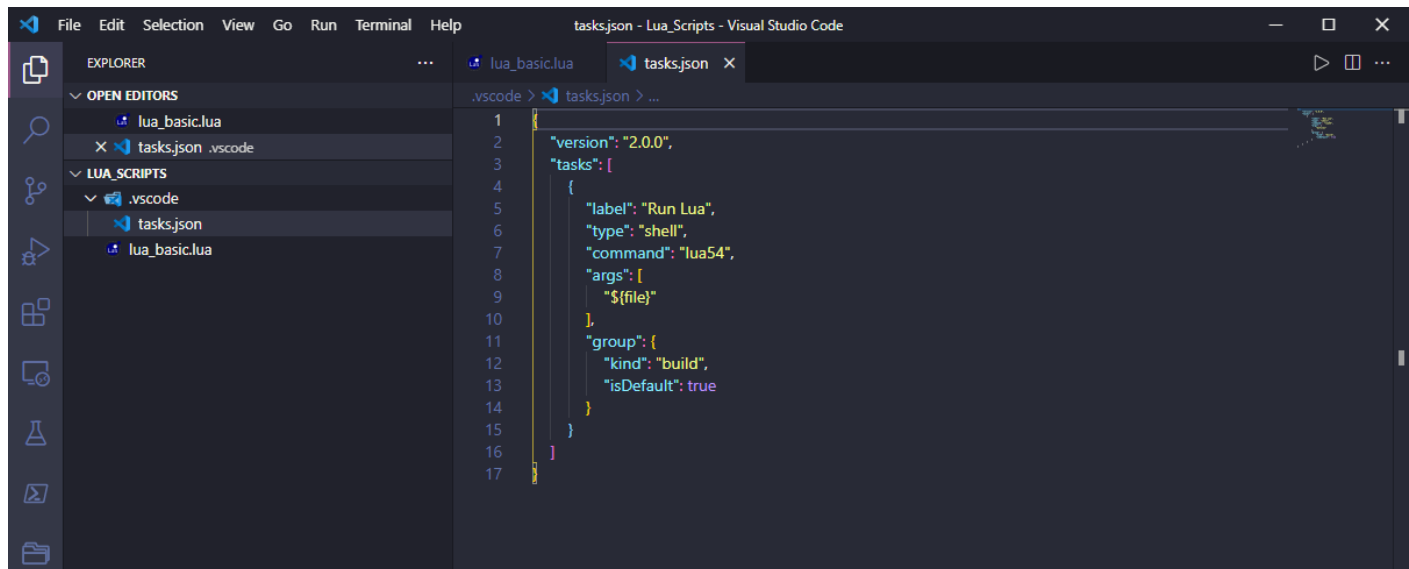
```
{
```

```

"version": "2.0.0",
"tasks": ,
    "group": {
        "kind": "build",
        "isDefault": true
    }
}
]
}

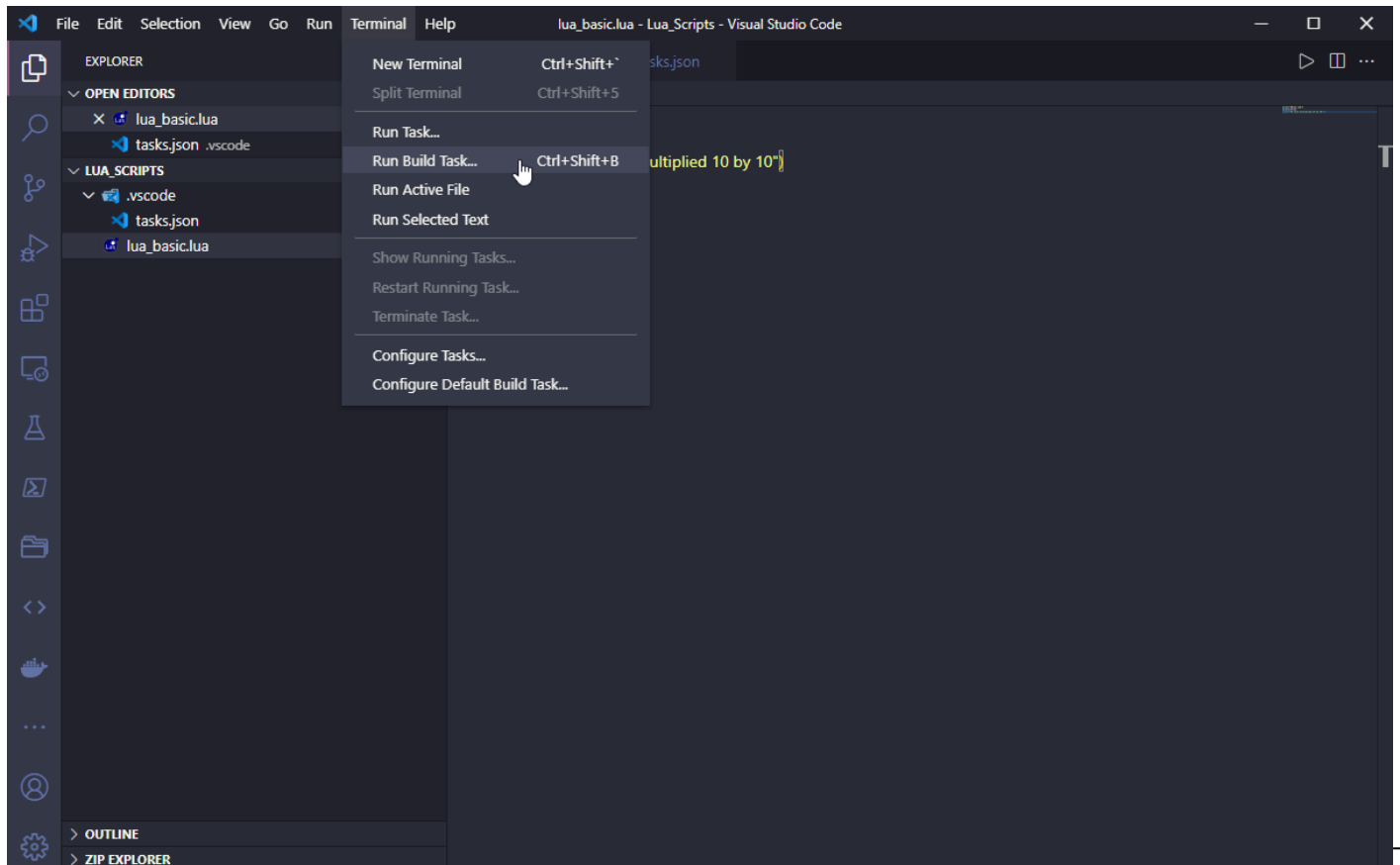
```

task.json



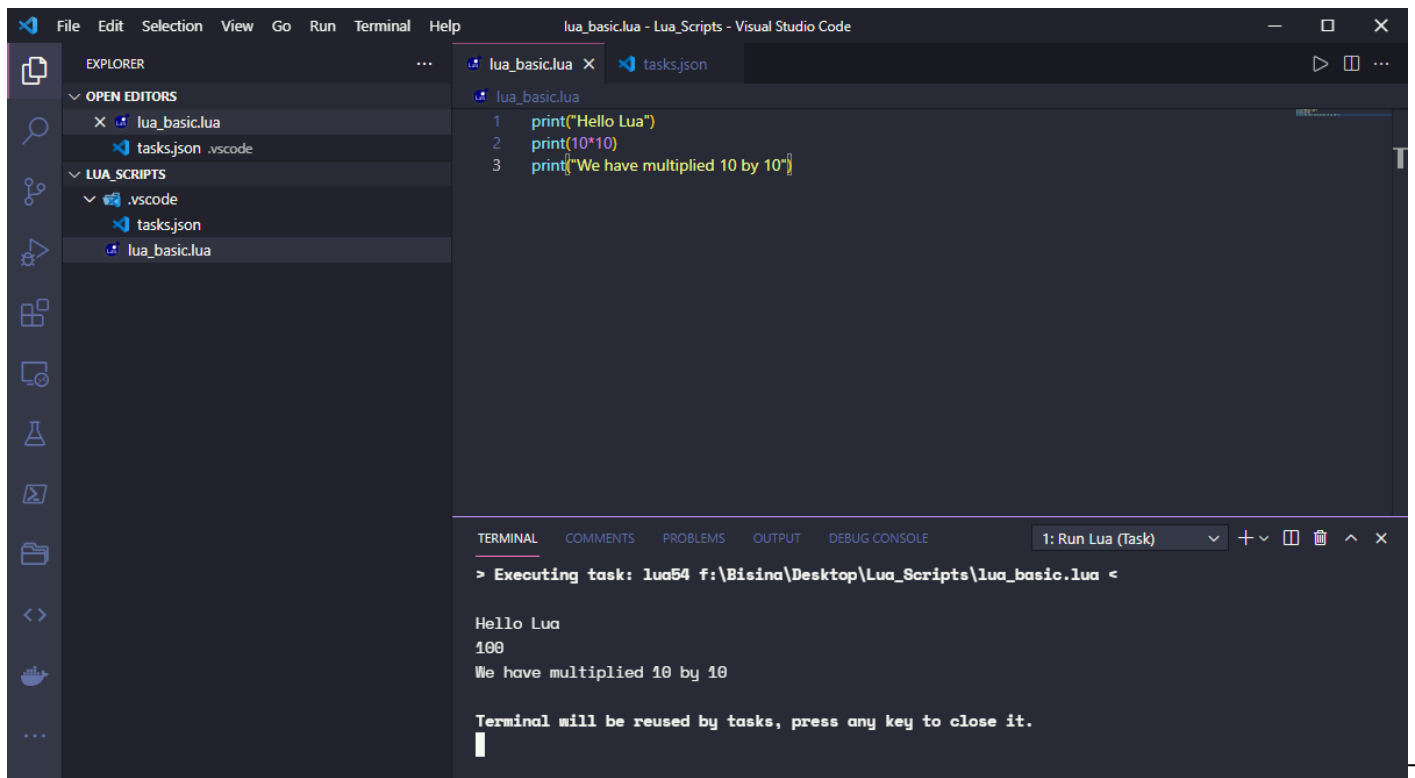
Step 3

Open up the "lua_basic.lua" file again. Then go to the "Terminal" again and click on "Run Build Task" or use the shortcut Ctrl+Shift+B.



This

will compile the file and provide us with the output.



That'

s it! Now we have a working Lua development environment that can be used to create Lua scripts. We can use the official Lua [reference manual](#) to explore the Lua language further.

Lua is powerful

Lua is a powerful scripting language that has limitless potential to add functionality to any program

on a multitude of platforms to suit any use case.

Related reading

- [BMC DevOps Blog](#)
- [Top 5 Best Practices for Software Development](#)
- [Java vs Go: What's The Difference?](#)
- [Python Development Tools: Your Python Starter Kit](#)
- [DevOps Engineer Roles & Responsibilities](#)
- [API/Developer Portals: How To Create Great API Portals](#)