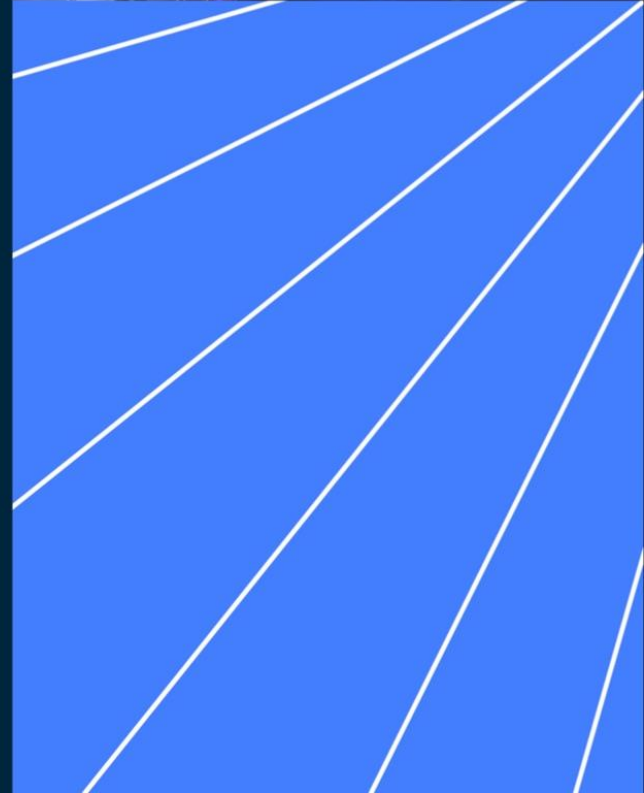


Rust Training

Workstation Setup Guide

DataBank IMX

March 5, 2024



Setting Up Your Workstation for Java Development

Performing Rust development requires a few components to be installed on your workstation. You can, of course, install whatever IDE you prefer, but I will be teaching the class using Visual Studio Code (a popular open-source IDE) and other convenience tools that you may find useful. If you want to set up your development environment similar to mine, the instructions below will walk you through the setup process. If you're following these steps, it is best to do them in the order presented.

Table of Contents

Prerequisites	2
Install RustUp (Rust Language Base Components)	3
Install the C/C++ Build Tools.....	5
Customize Visual Studio Code	7
Create a Test Program	9
Clone the Rust Training Repository	12

Prerequisites

Prior to any of the steps listed below, please complete all tasks in the guide titled:

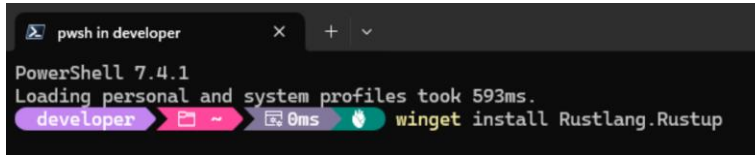
[Setting Up Your Workstation for Development Training.pdf](#)

Install RustUp (Rust Language Base Components)

Before we can write and execute Rust, we need to install the Rust language. This component includes all of the Rust tools including Cargo.

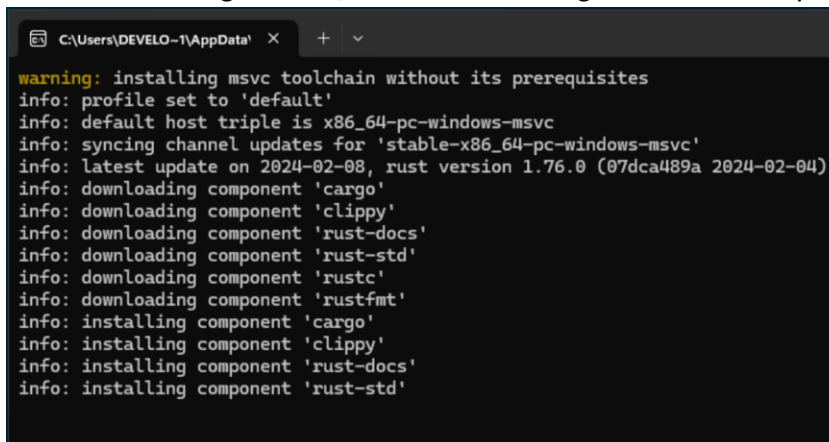
1. In the terminal, enter the following command

```
winget install Rustlang.Rustup
```



```
PowerShell 7.4.1
Loading personal and system profiles took 593ms.
developer ~ 0ms winget install Rustlang.Rustup
```

2. You may see a warning during installation indicating that the prerequisites for the MSVC toolchain are not installed. Ignore this, as we'll be installing those later if they're not already in place.

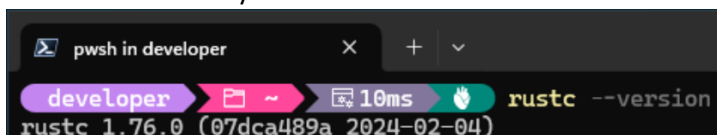


```
C:\Users\DEVELO~1\AppData\Local\Microsoft\WindowsApps\PowerShell\PowerShell.exe pwsh in developer
warning: installing msvc toolchain without its prerequisites
info: profile set to 'default'
info: default host triple is x86_64-pc-windows-msvc
info: syncing channel updates for 'stable-x86_64-pc-windows-msvc'
info: latest update on 2024-02-08, rust version 1.76.0 (07dca489a 2024-02-04)
info: downloading component 'cargo'
info: downloading component 'clippy'
info: downloading component 'rust-docs'
info: downloading component 'rust-std'
info: downloading component 'rustc'
info: downloading component 'rustfmt'
info: installing component 'cargo'
info: installing component 'clippy'
info: installing component 'rust-docs'
info: installing component 'rust-std'
```

3. To verify that you have Rust installed, enter the following in the terminal:

```
rustc --version
```

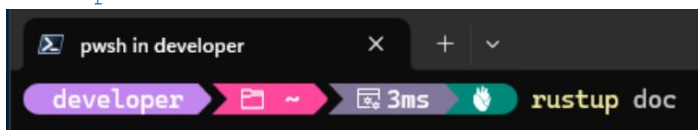
Note: You may need to re-launch the terminal or reboot before this command will work



```
pwsh in developer
developer ~ 10ms rustc --version
rustc 1.76.0 (07dca489a 2024-02-04)
```

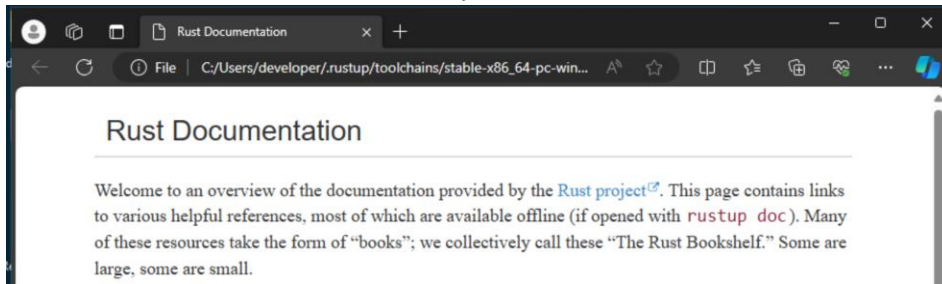
4. Although it's not necessary for the setup, you can also access the Rust documentation using this command:

```
rustup doc
```



```
pwsh in developer
developer ~ 3ms rustup doc
```

5. This will launch the documentation in your default browser:



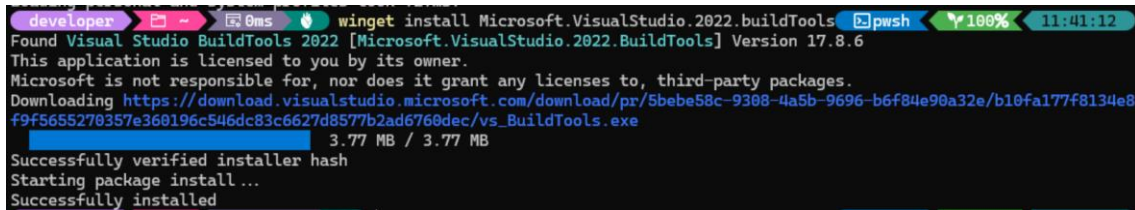
Install the C/C++ Build Tools

Rustup includes the tools necessary to compile Rust, but it does not include a linker to generate the executable. For this we will use the Microsoft C/C++ Build Tools.

Note: If you have a licensed copy of Visual Studio (separate from VS Code) installed on the workstation, you will already have the Build Tools package installed and can skip to step 2

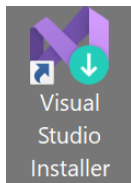
1. To install the Build Tools package, run the following command:

```
winget install Microsoft.VisualStudio.2022.BuildTools
```

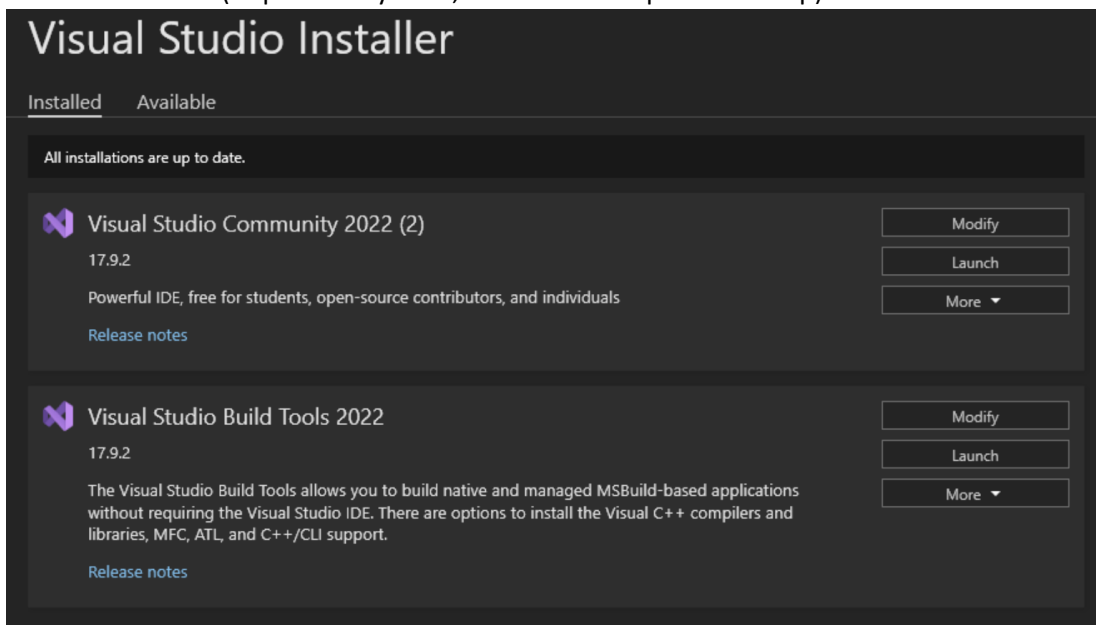


```
developer 0ms winget install Microsoft.VisualStudio.2022.buildTools pwsh 100% 11:41:12
Found Visual Studio BuildTools 2022 [Microsoft.VisualStudio.2022.BuildTools] Version 17.8.6
This application is licensed to you by its owner.
Microsoft is not responsible for, nor does it grant any licenses to, third-party packages.
Downloading https://download.visualstudio.microsoft.com/download/pr/5bebe58c-9308-4a5b-9696-b6f84e90a32e/b10fa177f8134e8
f9f5655270357e360196c546dc83c6627d8577b2ad6760dec/vs_BuildTools.exe
3.77 MB / 3.77 MB
Successfully verified installer hash
Starting package install...
Successfully installed
```

2. Once the build tools are installed, we need to add C/C++ support. Click the start button and run the Visual Studio Installer



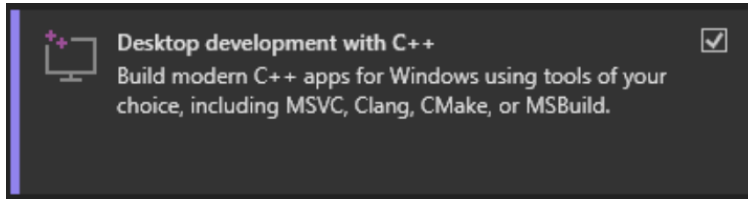
3. After the installer launches, you will see either your licensed Visual Studio version or the Visual Studio Build tools (or potentially both, as on the example PC I set up).



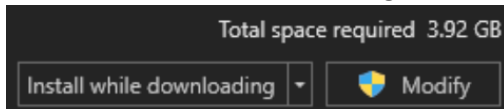
4. On your installed application, click the “Modify” button.



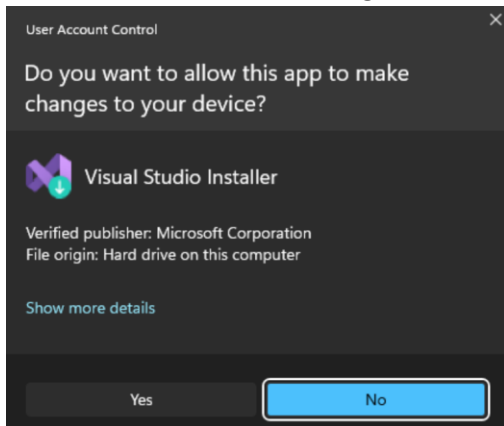
5. Select the toolkit for “Desktop development with C++.”



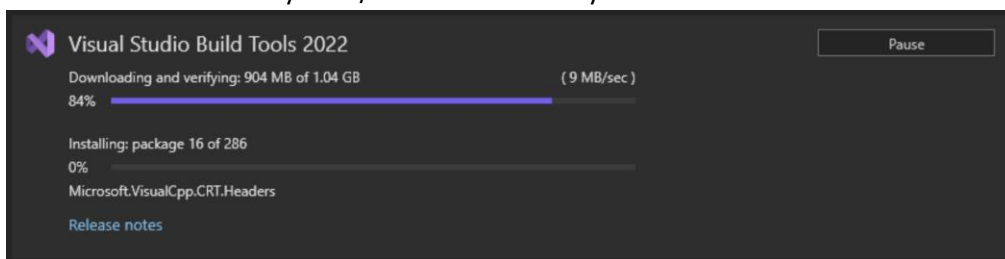
6. Leave “Install while downloading” selected and click the “Modify” button.



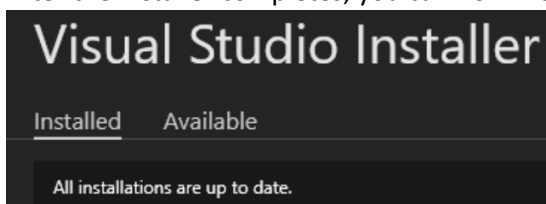
7. Click “Yes” on the UAC warning.



8. The installer will install your C/C++ tools. This may take a while...

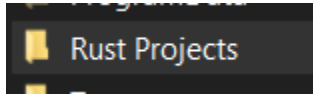


9. After the installer completes, you can now move on to set up and test Rust programming in VS Code.



Customize Visual Studio Code

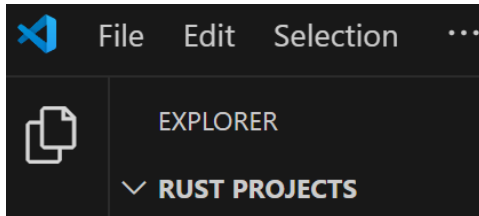
1. Create a folder called “Rust Projects.” I created mine on the C: root.



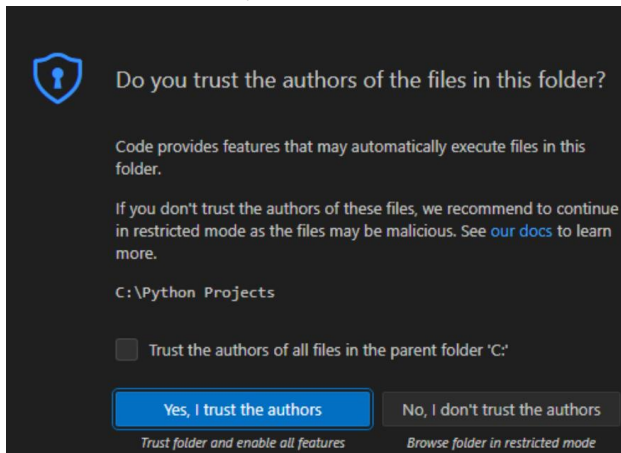
2. In the terminal, navigate to the folder you created and enter the following command
`code .`



3. Visual Studio Code will automatically launch in your project folder



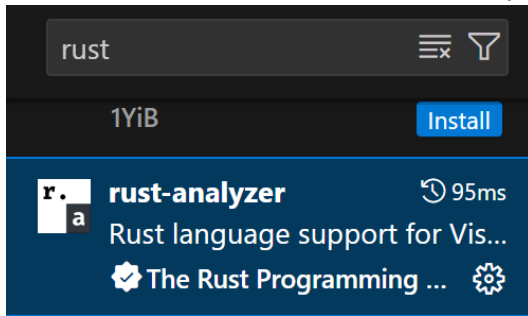
4. You can choose “Yes, I trust the authors” to allow VS Code to trust your project folder.



5. Click on the “extensions” icon on the sidebar



6. Search for “rust” and install the “rust-analyzer” extension



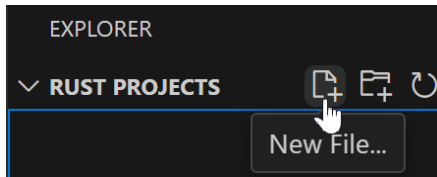
7. You can (optionally) install additional Rust extensions if desired, but this is the only one required for our training.

Create a Test Program

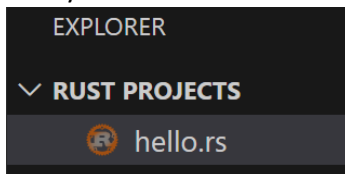
1. Still in VS Code in the “Rust Projects” directory, click on the “explorer” icon on the sidebar



2. Click on the “new file” icon in the explorer



3. Title your file “hello.rs”

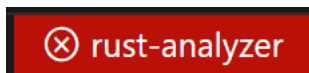


4. Enter the following code

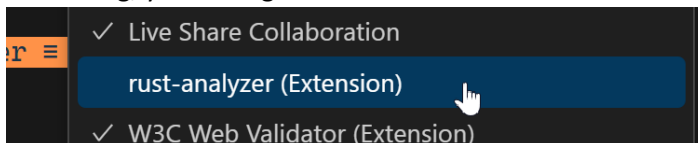
```
fn main() {  
    println!("Hello, World!");  
}
```

```
1  fn main() {  
2      println!("Hello, World!");  
3  }
```

5. At this point, you’ll probably notice that the “rust-analyzer” item in the VS Code taskbar is showing an error.

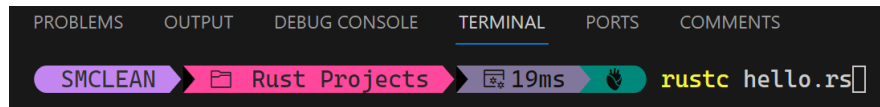


6. That error is because we have not created a complete project, so it is missing metadata necessary to load the folder as a Rust workspace. Rust will still compile and run just fine, so if the red error item is distracting, you can right-click on the VS Code taskbar and uncheck (hide) the rust-analyzer status.



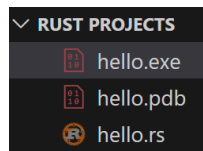
7. In the terminal (sorry... no “play” button for Rust), enter the following command to compile the program.

```
rustc hello.rs
```



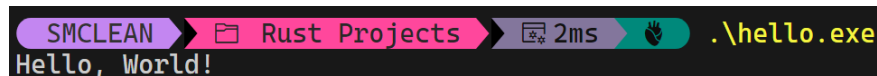
If you receive an error indicating a missing linker, go back and redo the step to [install the build tools](#).

8. You should now see a new file “hello.exe” in the directory.

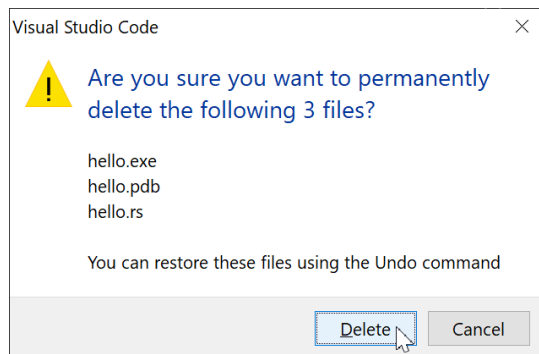


9. In the terminal, run the following command to execute your compiled program.

```
.\hello.exe
```

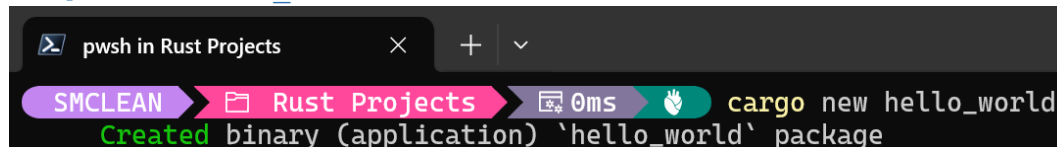


10. Delete the three files that were created and close VS Code



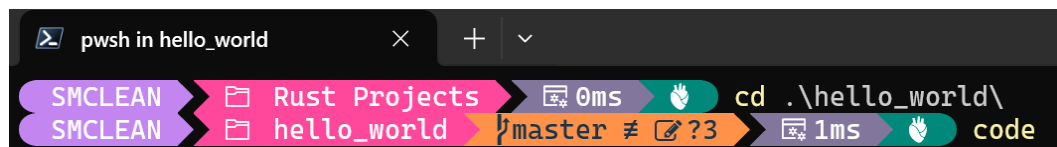
11. In a terminal window, navigate to your “Rust Projects” folder and run the following command to create a new project:

```
cargo new hello_world
```

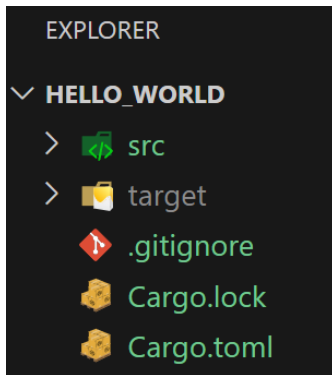


12. Navigate into the new “hello_world” directory and run the following command to launch VS Code:

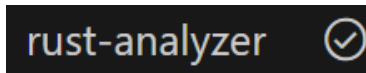
```
code .
```



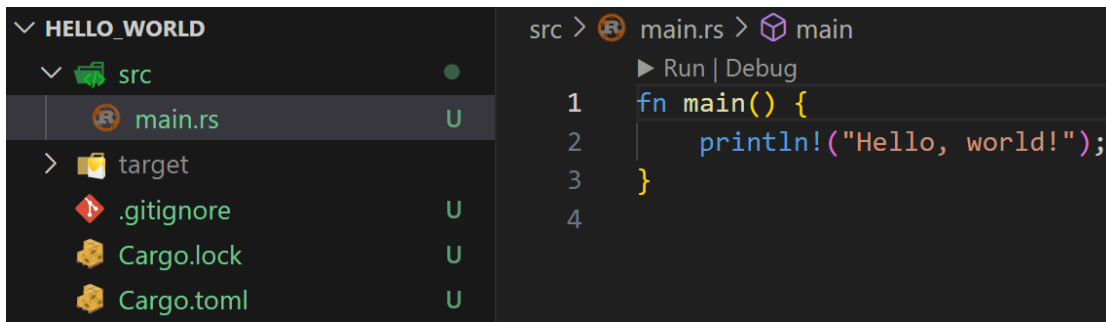
13. You'll notice that a set of files have been created. For now, the content of most of them doesn't matter.



14. If you still have the rust-analyzer status showing, it no longer displays an error, because it was able to load this project as a workspace

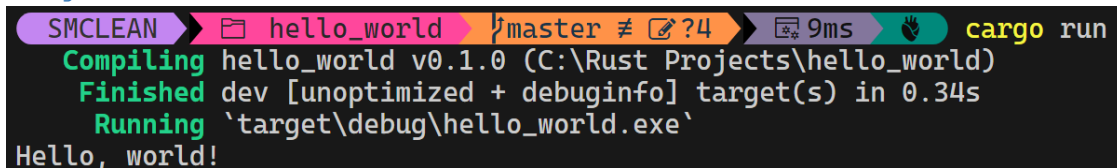


15. In the /src folder, there is a file named main.rs that already contains a "hello world" function. We don't need to edit it for now.



16. In the terminal run the following command to compile and execute the program:

`cargo run`



Note: "hello_world.exe" will be created in the /target/debug folder

17. Congratulations. Rust is successfully installed and tested, and you're ready to code.

Clone the Rust Training Repository

Finally, you'll need to clone a copy of the repository to work with.

I have two different locations where this repository is stored:

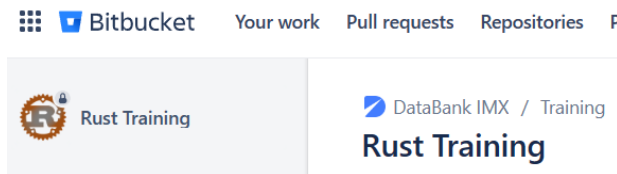
- Bitbucket: <https://bitbucket.org/databankimx/rust-training>
- GitHub: <https://github.com/ZeroKlu/rust-training>

1. For access to either repository, email smclean@databankimx.com to request access.

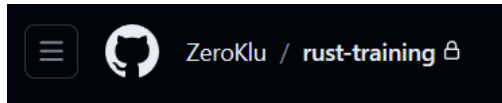
Be sure to indicate whether you need access to Bitbucket or GitHub and provide the username you use on the selected source control system.

2. In a browser, navigate to the repository you selected and make sure you have access:

- a. In Bitbucket, you should see this:



- b. In GitHub, you should see this:



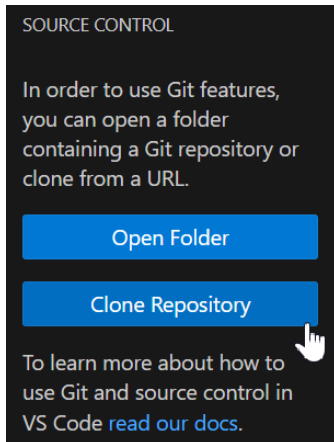
3. Right-click the VS Code icon and select "New Window" to launch an empty instance of VS Code



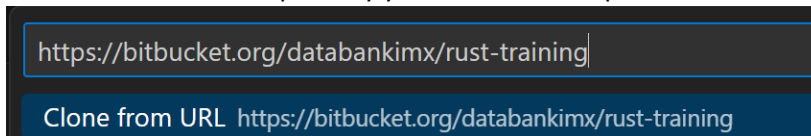
4. Click the Source Control icon on the sidebar



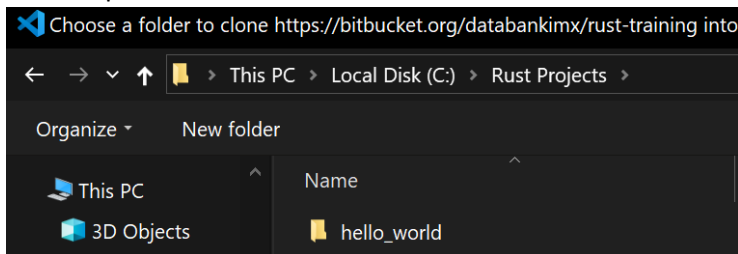
- Click the button labeled “Clone Repository”



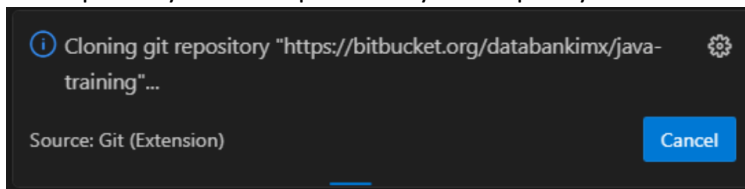
- Enter the URL to the repository you selected in step 1



- Select a path

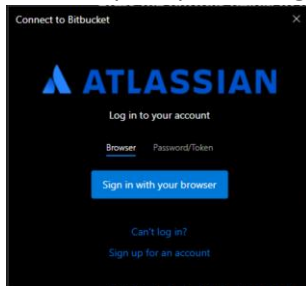


- The repository will be copied locally to the path you selected

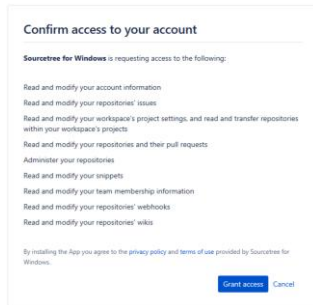


- If you used the GitHub repository, skip to step 14

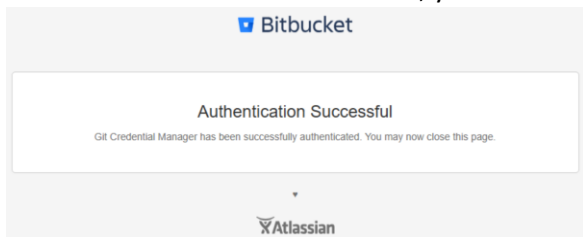
10. You'll be prompted to log in again



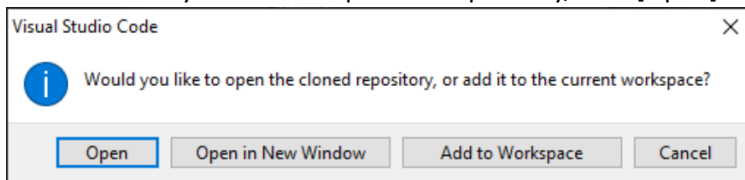
11. Grant access again in the web page that opens



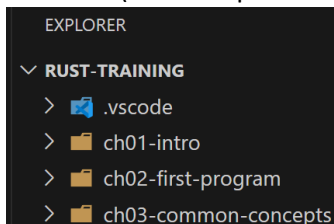
12. You'll see a success alert. After this, you can close the browser



13. When asked if you want to open the repository, click [Open]



14. The repository will open, and you should see a number of folders containing sample code from the textbook (with samples and commentary from me).



Congratulations! Your system is set up for Rust training.

Happy Coding!