Solana Blockchain Developer Bootcamp Setup Instructions

If you need help with and step in the setup of your software, feel free to ask questions in the Chainlink Discord in the #bootcamp channel, or create a post on stackoverflow, tagging the software that you need help installing (ie NPM, Solana-CLI etc)

Software Installation

Please complete the following steps to install the required software for the Solana Blockchain Developer Bootcamp. You can skip any steps for software you already have installed if there is no stated minimum version, otherwise please ensure your version is equal to or greater than the stated minimum version of the software. If your version is less than the stated minimum version, please upgrade to the latest version as per the instructions below.

Windows Users

Solana isn't compatible with Windows. To install the Solana CLI, compile Solana programs and run the Solana local validator, you need to install the Windows Subsystem for Linux, then run all commands from there going forward. If your version of Windows isn't up to date and doesn't recognize the 'wsl' command, you can manually install WSL via the instructions here. Alternatively, you can install it via the Windows Store.

Once you've installed WSL, you can start it by typing 'wsl' in command prompt or powershell, or by opening the installed 'Ubuntu' app if you installed it via the Windows Store. From there, you should follow the steps below to install the required software:

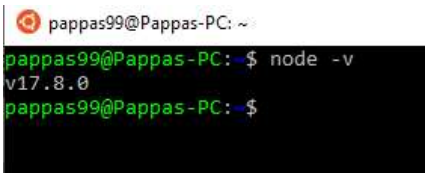
Node.js

Once you're in the Windows Subsystem for Linux shell, type in the following commands to install NVM and NodeJS: You may need to restart your WSL shell before running the 'nvm

```
nvm install node
```

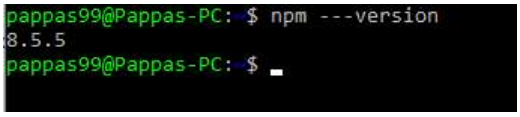
Once you've completed the installation, type the command below to ensure your installation is successful.

```
node -v
```



Once you've verified your Node.js installation, also verify your NPM installation by entering the following command

```
npm -version
```



Rust

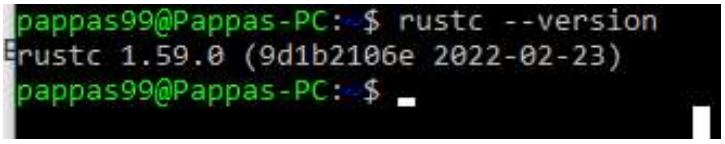
You will need to download and install the Rust programming language SDK. You can do this by entering in the following in your WSL shell:

```
curl --proto '=https' --tlsv1.2 -sSf
https://sh.rustup.rs | sh
```

Follow the instructions to install Rust. If the output tells you to set your $HOME .cargo/env directory,, and suggests a command to execute it, copy and paste the command into the shell and execute it.

When the process has completed, check to see that Rust was installed in your WSL via the following command:

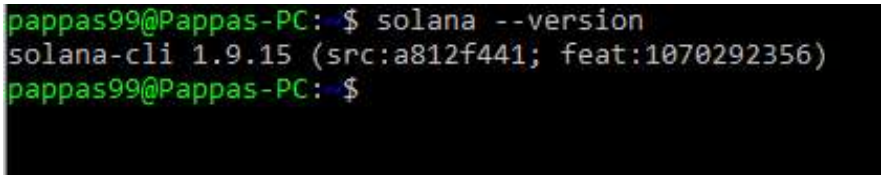
```
rustc --version
```



Solana CLI

You will need to download and install the Solana tool suite to perform tasks such as compiling and deploying your smart contracts. Run the following and follow the instructions to install the Solana CLI.

```
solana --version
```

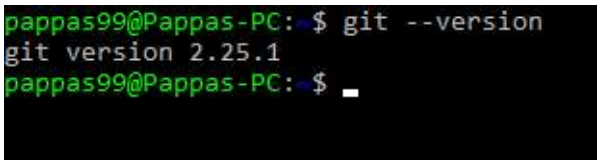


Git

Download and install the distributed revision control system Git. We will need this when installing some packages, and also to check your code into a repository when you're finished. Accept all default values during installation. If you're using macOS and you get prompted to install 'command line developer tools', accept and install them.

If you've installed WSL, you should have a working version of git already. Run the following command in the WSL shell to check to see if it's installed:

```
git --version
```



If git hasn't been installed, you can install it with the following command:

```
sudo apt-get install git
```

Solana Local Validator

During the exercises we will try to use the Solana Local Validator that comes with the Solana CLI install. However the pre-built version in the install is only compatible with CPUs that have AVX2 enabled. In your WSL shell, try to start the local validator. If your CPU is compatible then it will start and you will see normal output.

```
solana config set --url localhost
```

```
solana-test-validator
```

```
warning: `solana-core` (lib) generated 1 warning
    Finished dev [unoptimized + debuginfo] target(s) in 5.67s
     Running `/home/pappas99/solana/target/debug/solana-test-validator`
--faucet-sol argument ignored, ledger already exists
Ledger location: test-ledger
Log: test-ledger/validator.log
⠄ initializing...
Identity: HL66YEQJSWrpqLLUMSeKVHHcJlhUECFCYrJDy5G9D8RP
Genesis Hash: AaE625n3ysBuvHaGLz8AMPqAEPQohjZ9b9dxeRgbUHz6
Version: 1.9.11
Shred Version: 48444
Gossip Address: 127.0.0.1:1024
TPU Address: 127.0.0.1:1027
JSON RPC URL: http://127.0.0.1:8899
⠄ 00:36:17 | Processed Slot: 6687 | Confirmed Slot: 6687 | Finalized Slot: 6655 | Full Snapshot Slot: 6601 | Incremental Snapshot Slot: - | Transactions: 6769 | ⊠49
```

If you get an error, including a ‘core dump’ error, then it means your CPU isn’t compatible, and you need to build the Solana code from source, and run the validator with your compiled source code that’s compatible with your CPU. You can do it with the following commands in your WSL shell. The last command may take some time to complete. More information on building from source can be found on the official Solana repository

```
source $HOME/.cargo/env
rustup component add rustfmt
rustup update
sudo apt-get update
sudo apt-get install libssl-dev libudev-dev pkg-config zlib1g-dev llvm clang make
sudo apt install build-essential
git clone --branch v1.9.12 https://github.com/solana-labs/solana.git
cd solana
cargo build
```

Once this is done, you can attempt to run the validator again directly by running the following command inside your ‘solana’ directory:

```
cd validator
./solana-test-validator
```

```
pappas99@Pappas-PC:~/solana/validator$ ./solana-test-validator
+ exec cargo run --manifest-path=./Cargo.toml --bin solana-test-validator --
warning: an associated function with this name may be added to the standard library in the future
   --> core/src/banking_stage.rs:535:33
    |
535 |             buffered_packet_batches.retain_mut(|buffered_packet_batch_and_offsets| {
    |                                     ^^^^^^^^^^
    |
    = note: `#[warn(unstable_name_collisions)]` on by default
    = warning: once this associated item is added to the standard library, the ambiguity may cause an error or change in behavior!
    = note: for more information, see issue #48919 <https://github.com/rust-lang/rust/issues/48919>
    = help: call with fully qualified syntax `retain_mut(...)` to keep using the current method

warning: `solana-core` (lib) generated 1 warning
    Finished dev [unoptimized + debuginfo] target(s) in 5.67s
     Running `/home/pappas99/solana/target/debug/solana-test-validator`
--faucet-sol argument ignored, ledger already exists
Ledger location: test-ledger
Log: test-ledger/validator.log
⠄ initializing...
Identity: HL66YEQJSWrpqLLUMSeKVHHcJlhUECFCYrJDy5G9D8RP
Genesis Hash: AaE625n3ysBuvHaGLz8AMPqAEPQohjZ9b9dxeRgbUHz6
Version: 1.9.11
Shred Version: 48444
Gossip Address: 127.0.0.1:1024
TPU Address: 127.0.0.1:1027
JSON RPC URL: http://127.0.0.1:8899
⠄ 00:36:17 | Processed Slot: 6687 | Confirmed Slot: 6687 | Finalized Slot: 6655 | Full Snapshot Slot: 6601 | Incremental Snapshot Slot: - | Transactions: 6769 | ⊠499.966795000
```

Visual Studio Code

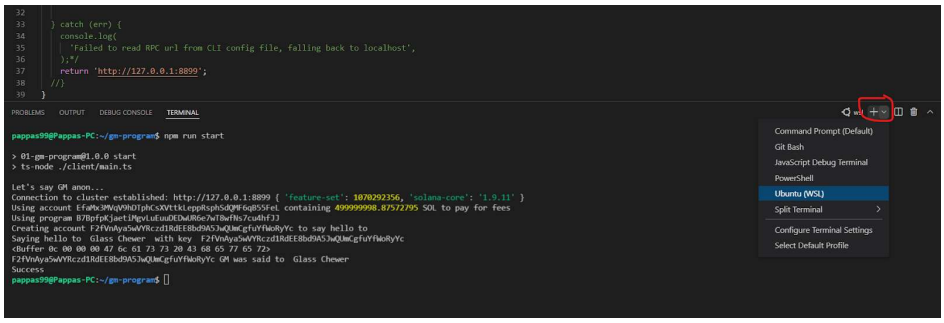
You’ll need a decent text/file editor for doing the exercises. We recommend Visual Studio Code, one of the most popular free, open source editors for smart contract development. Using Visual Studio Code will make it easier when you're

operating system

2. Install the Remote - WSL extension for VSCode if you want to be able to run WSL commands inside the VS Code terminal. Otherwise you can just run them all separately in WSL.
3. Open up a WSL shell window, and start VS Code directly from there by entering the following command:

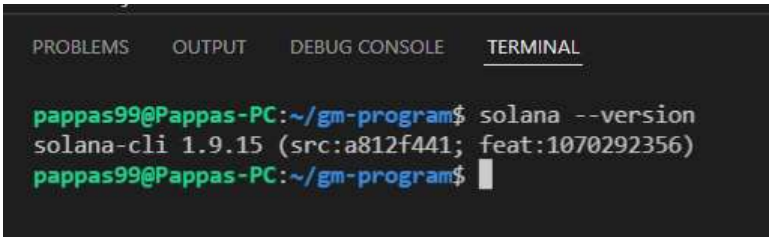
```
code .
```

4. Once VS Code has opened, you can interact with all your installed programs via the WSL Terminal. To open it, choose View -> Terminal from the top menu, or press CTRL + T, and then press the + button on the right of the terminal, and choose the WSL option to open up a new WSL terminal window.



5. You should be able to run all your installed programs from this shell window here, including a local validator.

```
solana -version
```



Linux/MacOs Users

Node.js

Minimum Required Version : 12.2.0
Minimum Windows O/S Version: 10
Minimum macOS version: 10

You can skip this step if you already have a working Node.js 12.2 or greater installation. To check this, you can open a new Terminal/Windows Command Prompt (or another CLI of your choice), type the command below and press enter. For Windows users, Command Prompt can be found by pressing the Windows Start Button and searching for 'Command

section if it meets the minimum requirements. Otherwise if you get an error, it means you don't have Node.js installed, and you should continue these steps to download and install it. If you do have a version of Node.js installed, but it's less than 12.2, follow these instructions to upgrade it.

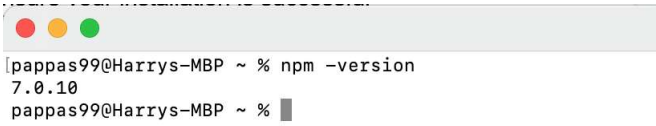
If you don't have Node.js installed, download and install the latest version of the JavaScript runtime environment Node.js and package manager NPM. This step is required for both JavaScript and Python tracks of the bootcamp. Accept all default answers for questions.

Once you've completed the installation, open a new Terminal/Windows Command Prompt (or another CLI of your choice), type the command below to ensure your installation is successful. For windows users, Command Prompt can be found by pressing the Windows Start Button and searching for 'Command Prompt'. For Mac users, the terminal can be found in Applications.

```
node -v
```

Once you've verified your Node.js installation, also verify your NPM installation by entering the following command

```
npm -version
```



Rust

You will need to download and install the Rust programming language SDK. Head to https://rustup.rs/ and follow the instructions to install the latest Rust stable installation.

Note for Windows Users: At some point in the installation, you may receive a message explaining that you'll also need the C++ build tools for Visual Studio 2013 or later. The easiest way to acquire the build tools is to install Build Tools for Visual Studio 2019.

You can verify a successfully installation by running the following command:

```
rustc --version
```

Solana CLI

You will need to download and install the Solana tool suite to perform tasks such as compiling and deploying your smart contracts.

Open a new Terminal/Windows Command Prompt (or another CLI of your choice), and type the command below

```
sh -c "$(curl -sSfL
https://release.solana.com/v1.9.5/install)"
```

Depending on your system you may get a prompt to set the *PATH* environment variable to include the Solana programs. If you get this message, you will need to update your *PATH* variable to include the directory of the installed Solana CLI.

You can verify a successfully installation by running the following command:

```
solana --version
```

Git

Download and install the distributed revision control system Git. We will need this when installing some packages, and also to check your code into a repository when you're finished. Accept all default values during installation. If you're using macOS and you get prompted to install 'command line developer tools', accept and install them.

To test git once it's installed, open a new Terminal/Windows Command Prompt (or another CLI of your choice), and type the command below

```
git --version
```

Visual Studio Code

You'll need a decent text/file editor for doing the exercises. We recommend Visual Studio Code, one of the most popular free, open source editors for smart contract development. Using Visual Studio Code will make it easier when you're following along with the exercise screenshots, and the integrated terminal makes it easier to switch between editing

7. Enable the terminal by selecting View -> Terminal from the top menu (or the keyboard shortcut Ctrl + `). Ensure you can run Node.js commands in your terminal window by running the following command

```
node -v
```

```
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL

pappas99@Pappas ~ % node -v
v12.18.4
pappas99@Pappas ~ %
```

If you're using Windows and you get an error, ensure you're using your CMD Terminal:
- In VS Code, press CTRL + SHIFT + P
- Search for 'Terminal Select Default Profile'
- Select the 'Command Prompt' selection
- Restart VS Code and try running the command again in the terminal

Testing Your Setup

The final step of the setup instructions is to airdrop yourself some SOL tokens on the DevNet network to ensure everything you installed works together, and that you can connect to a public network:

1. Open up your VS Code terminal (View menu -> Terminal), or if you're running WSL in windows, enter the following commands in the WSL shell in VS Terminal

2. Enter the following command into the terminal to set your Solana provider URL to the Devnet cluster

```
solana config set --url
https://api.devnet.solana.com
```

```
pappas99@Pappas ~ % solana config set --url https://api.devnet.solana.com
Config File: /Users/pappas99/.config/solana/cli/config.yml
RPC URL: https://api.devnet.solana.com
WebSocket URL: wss://api.devnet.solana.com/ (computed)
Keypair Path: /Users/pappas99/.config/solana/id.json
Commitment: confirmed
pappas99@Pappas ~ %
```

3. Generate a new keypair account. When promoted for a password, you can enter one, or leave it blank and press enter.

```
solana-keygen new --force
```

```
BIP39 passphrase (empty for none):

Wrote new keypair to /Users/pappas99/.config/solana/id.json
==========================================================================
pubkey: 7krPpUNiGpm7fT9c6D5E4RjCJfBC6iVZXEe3g39iLwAU
==========================================================================
Save this seed phrase and your BIP39 passphrase to recover your new keypair:
spider setup flight until call gold ladder rug verify potato busy timber
==========================================================================
pappas99@Pappas ~ %
```

4. Now that you've created an account, you can use the airdrop program to obtain some SOL tokens. You should see output similar the following below, confirming the transferring of SOL to your newly created account

```
solana airdrop 2
```

```
pappas99@Pappas ~ % solana airdrop 2
Requesting airdrop of 2 SOL

Signature: 5MtA1cUMzc6sEDKZm4zcKNecTZUKvMQRx1oSni92mxRRE6xqK2HGiSWHBGG57amkKjEoUBVSqC5kpASw5k3e42mf

2 SOL
pappas99@Pappas ~ %
```

Congratulations, you're now all set for the Solana Blockchain Developer Bootcamp!