SET UP INSTRUCTIONS

Ubuntu setup for Plutus Playground **1**
- Notes 1
- Install Nix and set up cache 1
  - single-user 1
  - multi-user 3
  - After either Nix installation method 4
- Build the Plutus Playground server and client and start it 5
- Miscellaneous 7
  - Completely uninstalling Nix 7

Plutus Playground Setup on Windows **9**
- Notes 9
- Legend 9
- 1. Install WSL2 10
- 2. Prepare your linux 10
- 3. Install Nix 11
- 4. Get Plutus and Plutus Playground repositories 12
- 5. Build Plutus 12
- 6. Build Plutus Playground 13
- 7. Build dependencies 13
- 8. Start Plutus Playground server 13
- 9. Start the Plutus Playground client 14
- Switching week (Plutus Pioneers Only) 14
- Starting the repl (Optional) 15
- Installing VSCode Editor (Optional) 16
  - Installing Windows Terminal (Optional) 16
- Installing Docker Desktop for Windows (Optional) 17

MacOS Setup for Plutus Pioneer Program **19**
- Notes 19
- Setup 19
- Troubleshooting 23

Ubuntu setup for Plutus Playground

Notes

These instructions were tested on Ubuntu 20.04

Note Part of what Nix will be doing is downloading a lot of build artifacts from IOG to make the compilation processes take a lot less time. It means the difference between 10 minute builds versus 3+ hour builds. The down-side is this data will be quite large on your local system and you should expect another 15G minimum to be used by it. Something to think about before going further, if more disk space is needed than you have on your system.

Install Nix and set up cache

Nix can be installed single-user or multi-user

More detailed info can be found in the Nix Package Manager Guide

single-user

Single-user Nix installation has advantages.

- No daemon and socket are created
- A group of 32 nix users doesn't get created on the system
- Nothing is written into `/etc`

The single-user Nix installer requires curl

```
sudo sh -c 'apt update && apt install curl'
```

Install nix

```
sh <(curl -L https://nixos.org/nix/install) --no-daemon
```

The installer should create the `/nix` directory for you with the proper permissions. When it's done you will see

```
Installation finished!  To ensure that the necessary environment

variables are set, either log in again, or type



  . /home/<youruser>/.nix-profile/etc/profile.d/nix.sh
```

Execute the above command now to set the environment in this shell (also logout/login will achieve this)

Make sure the changes the installer just made to your `~/.profile` make sense.

Next we will add IOG's caches to Nix to speed up our development significantly by using their build artifacts. This is very important and means the difference between 3+ hours and less than 10 minutes!

```
mkdir ~/.config/nix

echo 'substituters = https://hydra.iohk.io https://iohk.cachix.org https://cache.nixos.org/' >>
~/.config/nix/nix.conf

echo 'trusted-public-keys =
hydra.iohk.io:f/Ea+s+dFdN+3Y/G+FDgSq+a5NEWhJGzdjvKNGv0/EQ=
iohk.cachix.org-1:DpRUyj7h7V830dp/i6Nti+NEO2/nhblbov/8MW7Rqoo=
cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=' >>
~/.config/nix/nix.conf
```

multi-user

If you decide you'd like to go with multi-user Nix, read on.

The multi-user Nix installer requires curl and rsync

```
sudo sh -c 'apt update && apt install curl rsync'
```

Install nix

```
sh <(curl -L https://nixos.org/nix/install) --daemon
```

This will run a wizard, prompting you for some things. When it's done we need to set the environment in this shell (also logout/login will achieve this)

```
. /etc/profile.d/nix.sh
```

Next we will add IOG's caches to Nix to speed up our development significantly by using their build artifacts. This is very important and means the difference between 3+ hours and less than 10 minutes!

```
sudo sh -c "echo 'substituters = https://hydra.iohk.io https://iohk.cachix.org
https://cache.nixos.org/' >> /etc/nix/nix.conf"
```

```
sudo sh -c "echo 'trusted-public-keys =
hydra.iohk.io:f/Ea+s+dFdN+3Y/G+FDgSq+a5NEWhJGzdjvKNGv0/EQ=
iohk.cachix.org-1:DpRUyj7h7V830dp/i6Nti+NEO2/nhblbov/8MW7Rqoo=
cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=' >>
/etc/nix/nix.conf"
```

```
sudo systemctl restart nix-daemon.service
```

Optionally add your shell account to the Nix trusted users in order to apply the Nix trusted public keys for your builds, e.g.:

```
sudo sh -c 'echo "trusted-users = $0" >> /etc/nix/nix.conf' `whoami`
```

```
sudo systemctl restart nix-daemon.service
```

After either Nix installation method

If everything worked you should be able to run a nix program

```
nix-env --version
```

and see output like this, version may vary

```
nix-env (Nix) 2.3.14
```

Build the Plutus Playground server and client and start it

We now need to get the plutus repo which contains the libraries for working on Plutus and the Plutus Playground server.

```
git clone https://github.com/input-output-hk/plutus
```

```
cd plutus
```

Note If you will be working on the Plutus Pioneer Project, it will be necessary at this point to `git checkout ...` a specific commit to match the class materials. That commit hash is listed in the exercises week## directory in the `cabal.project` file. For more info on this, see [Working on contracts with and without cabal build]

Now we build the plutus repo

```
nix build -f default.nix plutus.haskell.packages.plutus-core
```

This will take a while the first time. It will only be necessary to do it again if/when you `git pull` changes down from github or switch to a different branch/commit. These probably won't happen often.

Now we will run the Plutus Playground servers. We start these in a `nix-shell` which sets up the environment and has working versions of tools.

Note Optionally, a project exists to control the two Plutus Playground servers as a systemd service, making start/stop/restart much less manual. See [plutus-playground-systemd] for installation and configuration. If you use this, skip past the "terminal window" instructions below and access your new playground server in a browser.

Open two terminal windows

In terminal window 1

```
cd plutus

nix-shell

cd plutus-playground-server

plutus-playground-server
```

If it's successful, you should see `Interpreter ready`

In terminal window 2

```
cd plutus

nix-shell

cd plutus-playground-client

npm run start
```

If it's successful, you should see `[wdm]: Compiled successfully.`

You should now be able to navigate to https://localhost:8009. The browser will complain about it being a risky website, allow it.

Miscellaneous

Completely uninstalling Nix

Nix (unfortunately) installs with a non-distro-specific and not-reversible method and so requires careful unistallation if you don't want it any longer. The Nix documentation says simply removing `/nix` is the way but this leaves a lot of unneeded files on your system. We can clean this up properly.

These instructions work equally well for single- or multi-user Nix.

First, disable and stop the systemd units if they exist. This is harmless to do if they don't exist.

```
sudo systemctl disable --now nix-daemon.service

sudo systemctl disable --now nix-daemon.socket
```

Then delete the many directories and files that are on the system

```
rm -rf $HOME/{.nix-*,.cache/nix,.config/nix}

sudo rm -rf /root/{.nix-channels,.nix-defexpr,.nix-profile,.config/nixpkgs,.cache/nix}

sudo rm -rf /etc/{nix,profile.d/nix.sh*}

sudo rm -rf /nix
```

Now we will remove the 32(!) `nixbld` users that were added to the system for a multi-user installation.

```
sudo sh -c 'for N in $(seq 32); do deluser "nixbld$N"; done'
```

Finally, if you see one, remove the line that was added to your `$HOME/.profile` that sources the nix environment.

Plutus Playground Setup on Windows

Notes

This guide will show you how to set up a Plutus Playground in Debian running on WSL2 in Windows. It also contains specific instructions which are useful for Plutus Pioneers. However it can be used by anyone who wants to run the Plutus Playground on Windows.

This guide will not require you to use an editor outside of debian or even docker. However, as these are popular options I have included optional sections for installation of VSCode editor, WIndows Terminal and Docker.

You can follow this guide for any Linux distribution. This guide will use Debian, and it is the one I recommend if you don't have a specific preference. If you choose another distribution, you will need to substitute some of the Debian Bash commands with the equivalent in your chosen distribution.

Legend

Please pay close attention to which directory each command is called from in the code boxes. Each line will start withh clearly denoting both directory and context before each command. This guide will assume you know how to change directories in Debian and for conciseness has omitted changing of directory and between shells. Make sure you cd into the proper directory in the proper shell by referring to this legend.

Home directory in Debian

`~$`

Home directory inside of nix-shell in Debian

```
[nix-shell ~$]
```

Command prompt in Windows (Elevated)

```
C:\>
```

1. Install WSL2

Follow the steps here to install WSL2 on Windows

https://docs.microsoft.com/en-us/windows/wsl/install-win10

In step 6 you will be asked to install a distribution of linux on your new WSL2. Choose Debian or any other distribution. This guide will be specific to Debian.

When Windows installs your Debian, you will choose a user name and password. You will need these, so choose something fitting.

2. Prepare your linux

You will need some programs installed in Debian to be able to complete this guide. Technically you don't absolutely need VIM if you really dont want it, but I would suggest installing it all the same.

```
~$ sudo apt-get update
```

```
~$ sudo apt-get upgrade
~$ sudo apt-get install wget vim git curl npm
```

3. Install Nix

```
~$ sudo curl -L https://nixos.org/nix/install | sh
```

The following will edit the nix configuration to aid you in building plutus from nix.

```
~$ sudo mkdir /etc/nix

~$ sudo touch /etc/nix/nix.conf

~$ sudo chmod 777 /etc/nix/nix.conf

~$ echo "sandbox = false

use-sqlite-wal = false

system-features = kvm

substituters        = https://hydra.iohk.io https://iohk.cachix.org https://cache.nixos.org/

trusted-public-keys = hydra.iohk.io:f/Ea+s+dFdN+3Y/G+FDgSq+a5NEWhJGzdjvKNGv0/EQ=
iohk.cachix.org-1:DpRUyj7h7V830dp/i6Nti+NEO2/nhblbov/8MW7Rqoo=
cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=" >> /etc/nix/nix.conf

~$ sudo chmod 644 /etc/nix/nix.conf

~$ bash
```

4. Get Plutus and Plutus Playground repositories

These are tho only depositories needed for running Plutus Playground. You do not need the plutus-starter repository.

```
~$ mkdir code
```

```
~/code$ git clone https://github.com/input-output-hk/plutus
```

```
~/code$ git clone https://github.com/input-output-hk/plutus-pioneer-program
```

5. Build Plutus

If you are not a putus pioneer, then skip the next code box (nix build -f default.nix ...).

If you are a plutus pioneer, then first checkout the correct commit. You can read more about why this is necessary in the optinal section for pioneers.

```
~/code/plutus$ cat ~/code/plutus-pioneer-program/code/weekxx/cabal.project | grep -m 1 tag:
```

```
~/code$ git checkout ${commit-hash-tag-from-above-output}
```

Now build Plutus

```
~/code/plutus$ nix build -f default.nix plutus.haskell.packages.plutus-core.components.library
```

6. Build Plutus Playground

```
~/code/plutus$ nix-build -A plutus-playground.client
```

```
  ~/code/plutus$ nix-build -A plutus-playground.server
```

7. Build dependencies

```
~/code/plutus$ nix-build -A plutus-pab
```

```
~/code/plutus$ nix-build -A plutus-playground.generate-purescript
```

```
~/code/plutus$ nix-build -A plutus-playground.start-backend
```

8. Start Plutus Playground server

You are finally ready to fire up Plutus Playground.

```
~/code/plutus$ nix-shell
```

```
[nix-shell ~/code/plutus/plutus-pab$] plutus-pab-generate-purs
```

```
[nix-shell ~/code/plutus/plutus-playground-server$] plutus-playground-generate-purs
```

```
[nix-shell ~/code/plutus/plutus-playground-client$] plutus-playground-server -i 120s
```

9. Start the Plutus Playground client

```
~/code/plutus$ nix-shell
```

```
[nix-shell ~/code/plutus/plutus-playground-client$] npm run start
```

At this point you have everything you need to run plutus playground and the npm will tell you towards the end of its output where you can access it. Default is https://localhost:8009/, so open up your broser and head on over.

In the playground you can use the in browser editor to write code, compile and simulate scenarios. There are also very usefull links in the top right if you need some help getting started. I recommend using these guides for any further information on the specifics of the actual playground.

Switching week (Plutus Pioneers Only)

In the plutus pioneer program, each week you will have to adjust so you are using the correct commit of plutus. You need to match the correct Plutus repository commit hash for the weeks exercises.

Stop the client, server and repl if they are running.

Then update Plutus and Plutus Pioneer Program repositories.

```
~/code/plutus$ git pull origin master
```

```
~/code/plutus-pioneer-program$ git pull origin master
```

Find this weeks correkt plutus commit tag hash.

```
~/code/plutus-pioneer-program/code/weekxx$ cat cabal.project | grep -m 1 tag:
```

```
~/code/plutus$ git checkout ${commit-hash-tag-you-found-in-the-step-above}
```

Rebuild Plutus

```
~/code/plutus$ nix build -f default.nix plutus.haskell.packages.plutus-core
```

Start the server, client and repl again.

- Tip: In some systems, nix-shell can take some time to load up. To save time you can start 3 Bash shells at the same time and load up nix-shell in all three.

Starting the repl (Optional)

With cabal repl you can run GHCi (a sort of cli into the haskell) directly in your terminal. It is very useful for compiling your code if you are editing it in your Debian environment with vim or any other editor, such as a connected VSCode in Windows. It is also necessary to run the repl if you want to follow along in the plutus pioneer lessons.

In the examples below replace the final directory (.. week xx) with the correct week name if you are following the pioneer program or another directory with your code.

```
~/code/plutus$ nix-shell
```

```
[nix-shell ~/code/plutus-pioneer-program/code/weekxx$] cabal update
```

```
[nix-shell ~/code/plutus-pioneer-program/code/weekxx$] cabal build
```

```
[nix-shell ~/code/plutus-pioneer-program/code/weekxx$] cabal repl
```

Installing VSCode Editor (Optional)

If you prefer to run VSCode editor over native linux editors then you can use this guide to install VSCode and then connect it to your linux distribution.

Install VSCode

https://code.visualstudio.com/download

Install "Remote - Development" extension pack [ms-vscode-remote.vscode-remote-extensionpack]

Install "Docker" extension [ms-azuretools.vscode-docker]

Install "Haskell Syntax Highlighting" extension [justusadam.language-haskell]

```
~/code$ code .
```

This will open VSCode with the directory tree in your ~/code directory. Anything inside will be accessible throught the editor. Saving, creating files, deleting files should all work seamlessly between VSCode and Debian at this point.

Installing Windows Terminal (Optional)

If you aren't familiar with Windows Terminal, it is a program which lets you run a tabbed interface for your CLI programs, such as WSL2 instances, Powershell, Azure and Command Prompt. It will also let you configure colors, fonts and set other convenient settings.

Install Windows Terminal

https://docs.microsoft.com/en-us/windows/terminal/get-started

Inside of Windows Terminal, open up settings and add a new profile pointing to your debian executable (located in %programfiles%/WindowsApps/)

If your windows user does not have access to the %programfiles%/WindowsApps/ folder, then you need to give yourself permission. In a command prompt with administrative privileges

```
C:\WINDOWS\system32> takeown /f "%programfiles%\WindowsApps"
```

```
C:\WINDOWS\system32> icacls "%programfiles%/WindowsApps" /grant %username%:RX
```

There are some additional things you could configure in Windows Terminal if you want to create a smooth environment. You could for example make a nix-shell profile that takes the debian executable and executes cd ~/code/plutus && nix-shell or even a Plutus server and Plutus client profile that simply fires up the server and client.

Installing Docker Desktop for Windows (Optional)

You do not need to use Docker if you follow this guide. However, as it is a popular option, I will mention how to do it here.

Follow the instructions located below to install Docker for Windows

https://docs.docker.com/docker-for-windows/install/

Launch Docker and make sure it has WSL2 enabled for your distribution (you will find this in Docker settings).

Before docker load, verify that docker works with.

```
~$ docker run hello-world
```

If you get an error, then try installing docker-load and also adding your user to the docker group

```
~$ sudo apt-get install docker-load
```

```
~$ sudo groupadd docker

~$ sudo usermod -aG docker ${user-name}
```

Now load the Plutus Docker

```
~/plutus$ docker load < $(nix-build default.nix -A devcontainer)
```

The docker will take some time to load. But if all goes well and you don't receive any more error messages, you are ready to go.

MacOS Setup for Plutus Pioneer Program

Credits

Cloned from Reddit

Go give u/RikAlexander karma!

Notes

Should work on Catalina and Big Sur. (Was tested on 2 Macs with Big Sur)

Setup

1 - Install Nix

```
[$] sh <(curl -L https://nixos.org/nix/install) --darwin-use-unencrypted-nix-store-volume
```

2 - Close terminal & reopen (to make sure that all environment variables are set)

3 - Check Nix installation / version with

```
[$] nix --version
```

4 - Edit the /etc/nix/nix.conf file

```
[$] nano /etc/nix/nix.conf
```

5 - Add these lines to the file:

```
substituters       = https://hydra.iohk.io https://iohk.cachix.org https://cache.nixos.org/

trusted-public-keys = hydra.iohk.io:f/Ea+s+dFdN+3Y/G+FDgSq+a5NEWhJGzdjvKNGv0/EQ=
iohk.cachix.org-1:DpRUyj7h7V830dp/i6Nti+NEO2/nhblbov/8MW7Rqoo=
cache.nixos.org-1:6NCHdD59X431o0gWypbMrAURkbJ16ZPMQFGspcDShjY=
```

Note: These lines are there to avoid very long build times

Note 2: if the file /etc/nix/nix.conf doesn't exist: create it. (`[$] mkdir /etc/nix` for the directory and `[$] touch /etc/nix/nix.conf` for the file)

6 - Restart your computer

7 - Now to install, clone the git repo first

```
[$] git clone https://github.com/input-output-hk/plutus.git
```

8 - All the following builds should be executed while in the plutus directory

```
[$] cd plutus
```

9 - Build the Plutus Core (This may take some time :) be patient)

```
[$] nix build -f default.nix plutus.haskell.packages.plutus-core.components.library
```

Note:

On MacOS BigSur some users have reported that the building failed with an error like:

```
error: while setting up the build environment: getting attributes of path '/usr/lib/libSystem.B.dylib': No such file or directory
```

There are two solutions that are reported to solve this problem:

Change the nix build to an unstable (read: newer) build of nixpkgs.

```
[$] sudo nix-channel --add https://nixos.org/channels/nixpkgs-unstable unstable
```

-
- Disabling the`sandbox` and `extra-sandbox-paths` properties in the `/etc/nix/nix.conf` and `~/.config/nix/nix.conf` files.

Note 2:

If anyone gets stuck because of this error:

```
"error: refusing to create Nix store volume ... boot volume is FileVault encrypted"
```

You should heck out these links (Thank you u/call_me_coldass):
https://github.com/digitallyinduced/ihp/issues/93#issuecomment-766332648
https://www.philipp.haussleiter.de/2020/04/fixing-nix-setup-on-macos-catalina/

10 - Build the Plutus Playground Client / Server

```
[$] nix-build -A plutus-playground.client
```

```
[$] nix-build -A plutus-playground.server
```

11 - Build other plutus dependencies

```
[$] nix-build -A plutus-playground.generate-purescript
```

```
[$] nix-build -A plutus-playground.start-backend
```

```
[$] nix-build -A plutus-pab
```

12 - Go into nix-shell

```
[$] nix-shell
```

13 - inside of the nix-shell

```
[$] cd plutus-pab
```

```
[$] plutus-pab-generate-purs
```

```
[$] cd ../plutus-playground-server
```

```
[$] plutus-playground-generate-purs
```

14 - start the playground server

```
[$] plutus-playground-server
```

Great! All set.

15 - Now in a new terminal window:

```
[$] cd plutus
```

```
[$] nix-shell
```

```
[$] cd plutus-playground-client
```

16 - Here we compile / build the frontend of the playground

```
[$] npm run start
```

We're done!

The playground should be up and running.

Open your finest browser and navigate to:

https://localhost:8009/

Cloned from Reddit

Go give u/RikAlexander karma!

Troubleshooting

`nix-shell` exits with segmentation fault

Error message:

```
[$] nix-shell

[1]    296 segmentation fault  nix-shell
```

Solutions:

Comment out the line `withHoogle = false;` in `shell.nix` before running `nix-shell`

```
#withHoogle = false;
```

-
- Better: Switch to master branch, this issue (along with others) should be solved. E.g. the following fixes some more issues with macOS Big Sur
  - Commit: 34aa9c323ed6da68a11f41d41d5aca9f469aaf4b
  - Date: 23.04.2021

`haskell-language-server` fails with segmentation fault / You can't get any Haskell editor-integration working

Problem: running `[$] haskell-language-server` in one of the plutus-pioneer-program repos fails with a segmentation fault

```
[$] cd plutus-pioneer-program/code/week01

[$] haskell-language-server

haskell-language-server version: 0.9.0.0 ...

...

[INFO] Using interface files cache dir: ghcide

[INFO] Making new HscEnv[plutus-pioneer-program-week01-0.1.0.0-inplace]

Segmentation fault: 11
```

-

Solution:

- Upgrade the `plutus` repo to a later release (maybe master branch). Since `haskell-language-server` has version 0.9.0.0 (as you can see in the first line after execution) and this version is not ready for macOS Big Sur. ***Note*** This error probably occurs due to the linker changes introduced in macOS Big Sur, see

`npm run start` for the `plutus-playground-client` fails with modules not found

Problem: Server exits prematurely with compilation errors. Complaining about modules it could not find.

Solution:

- Trigger rebuild of client: [$] cd plutus-playground-client [$] npm clean-install ```
- Try to start client: `[$] npm run start`
- If this is insufficient, try cleaning up the git repo and redo the previous steps
  [$] git clean -xfd ```

Many thanks to the Plutus Community for putting together these instructions. Further information can be found at these links:

https://docs.plutus-community.com/

https://docs.plutus-community.com/docs/setup/buildDocumentation.html

https://nixos.org/manual/nix/stable/#ch-installing-binary

https://www.youtube.com/watch?v=mnfIts6VbU&t=278s&ab_channel=JBarCode

https://www.youtube.com/watch?v=CXHmbOkoVG8&t=1154s&ab_channel=SeamossOnline