
moja global Technical Guide

Nov 26, 2020

Contents

1	FLINT Prerequisites	3
1.1	For Windows Based systems	3
1.2	For Linux Based systems	5
2	FLINT Development Setup	7
2.1	Git and Github guide	8
2.2	Windows Installation	10
2.3	Docker Installation (for Mac and Linux Variants)	13
2.4	FLINT.example	16
3	GCBM Development Setup	33
3.1	GCBM Prerequisites	33
3.2	Windows Installation	34
4	Contributing	39
4.1	Before making a contribution	39
4.2	Ways to contribute to moja global	40
4.3	After making your first contribution	42
4.4	Code Contribution Best Practices	43
4.5	Code Of Conduct	44
5	GitHub Workflow	47
5.1	GitHub Repository maintenance	47
5.2	Bots and Integrations	54
5.3	Automated Checks for pull requests	56
5.4	FLINT Architecture	61
5.5	FLINT Performance	61
5.6	Reviewing a contribution	61
5.7	Manually testing a pull request	62
6	Frequently Asked Questions	63
6.1	Moja Global	63
6.2	FLINT	64
6.3	FLINT Installation Support	66
6.4	GCBM	67
6.5	FLINTpro	67

7	Join the moja global family	69
7.1	moja global Slack	69
7.2	Technical Steering Committee Meetings	69
7.3	Outreach and Student Programs	70
7.4	moja global Outreach	70

moja global provides tools for estimating emissions and removals of greenhouse gases from the land sector.

FLINT: the Full Lands INtegration Tool The Full Lands Integration Tool (FLINT) is the flagship software developed by the moja global community. It is an integrating platform for estimating land-based greenhouse gas emissions and removals. Integrating refers to FLINT's design to combine a wide range of data with models to achieve more accurate estimates of stocks and fluxes of greenhouse gases. FLINT is consistent with the UNFCCC guidelines.

This Documentation is meant for Developers wishing to contribute to moja global repositories. If you would like to get in touch with the maintainers for other reasons, please drop a mail at info@moja-global.com.

Before we take a leap into the process of development, please take a moment to verify if you have the necessary tools setup and skills to get started on this project. You should be familiar with the following :-

1.1 For Windows Based systems

Contents:

1.1.1 Setup Git

If you already have a Git client and Github account, please skip this section. Otherwise keep on reading!

Install and Configure Git

To install Git, please refer to the official git installation instructions [here](#) and the configuration recommendations [here](#).

You can sign up for a Github account [here](#).

After successful installation of Git and GitHub account registration, you can proceed with further instructions on how to fork/clone the moja-global repositories [here](#).

1.1.2 Cmake Installation

CMake is required to build the FLINT from its source code on Windows. CMake is an open-source family of tools designed to build, test and package software.

- Please download and install Cmake from <https://github.com/Kitware/CMake/releases/download/v3.15.2/cmake-3.15.2-win64-x64.msi>

1.1.3 Visual Studio Installation

Building FLINT requires a C++ compiler. On Windows, it is recommended to use the Visual Studio IDE which includes a C++ compiler and several other useful tool. You can either install the Visual Studio 2019 or the Visual Studio 2017 variant.

Please follow these steps for a smooth installation:

For Visual Studio 2019

- Navigate to <https://visualstudio.microsoft.com/downloads/>
- Select the community version download button.
- If you don't have a Visual Studio Subscription, you can create one for free by clicking on "Create a new Microsoft account" on the login page.
- Follow the steps prompted by the installer.

For Visual Studio 2017

- Navigate to <https://visualstudio.microsoft.com/vs/older-downloads/>
- Expand the 2017 version and click on the download button.
- If you don't have a Visual Studio Subscription, you can create one for free by clicking on "Create a new Microsoft account" on the login page.
- Follow the steps prompted by the installer.

1.1.4 Vcpkg Installation

Finally, we need a C++ package manager to acquire and install third-party C++ libraries used by FLINT. Vcpkg maintains a catalog of more than 1,900 libraries that have been tested against Visual Studio 2019 and 2017, and builds these libraries during compilation to ensure compatibility with the FLINT source code. A fork of the original Vcpkg package has been created under moja global for the FLINT required libraries.

To build the libraries please follow the following steps:

- Clone the Vcpkg repository: <https://github.com/moja-global/vcpkg>
- Start a command shell in the Vcpkg repository folder and use the following commands:

```
# bootstrap
bootstrap-vcpkg.bat

# install packages
vcpkg.exe install boost-test:x64-windows boost-program-options:x64-windows boost-
↳ log:x64-windows turtle:x64-windows zipper:x64-windows poco:x64-windows libpq:x64-
↳ windows gdal:x64-windows sqlite3:x64-windows boost-ublas:x64-windows
```

- Once this has completed, start a command shell in your FLINT repository folder. Now use the following commands to create the Visual Studio solution:


```
# Create a build folder under the Source folder
cd Source
mkdir build
cd build

# now create the Visual Studio Solution (2019)
cmake -G "Visual Studio 16 2019" -DCMAKE_INSTALL_PREFIX=C:/Development/Software/
↪moja -DVCPKG_TARGET_TRIPLET=x64-windows -DENABLE_TESTS=OFF -DENABLE_MOJA_MODULES.
↪ZIPPER=OFF -DCMAKE_TOOLCHAIN_FILE=c:\Development\moja-
↪global\vcpkg\scripts\buildsystems\vcpkg.cmake ..

# OR Visual Studio Solution (2017)
cmake -G "Visual Studio 15 2017" -DCMAKE_INSTALL_PREFIX=C:/Development/Software/
↪moja -DVCPKG_TARGET
```

A solution **is** simply a container used by Visual Studio to organize one **or** more ↪
 ↪related projects. When you **open** a solution **in** Visual Studio, it automatically loads ↪
 ↪all the projects that the solution contains.

1.2 For Linux Based systems

Contents:

1.2.1 Setup Docker (for Linux based variants only)

In-order to setup FLINT by using docker containers, please follow the instructions below based on your Linux Distro. If your Linux distro is not listed below, please checkout the [Docker official installation guides](#) for more information:

- **Install on CentOS**

In order to setup the latest version of Docker on CentOS, checkout the official [Docker installation guide for CentOS](#).

- **Install on Fedora**

In order to setup the latest version of Docker on Fedora, checkout the official [Docker installation guide for Fedora](#).

- **Install on Debian**

In order to setup the latest version of Docker on Debian, checkout the official [Docker installation guide for Debian](#).

- **Install on Ubuntu**

In order to setup the latest version of Docker on Ubuntu, checkout the official [Docker installation guide for Ubuntu](#).

Test Docker version

Verify if the docker installation is successful by running the following command. If the following command does not return the version of the docker, your installation has been unsuccessful, please try again.

```
docker --version
```

FLINT Development Setup

This section guides first-time contributors through installing FLINT development environment on Windows and Ubuntu.

The recommended method for installing the FLINT development environment is on Windows using cmake, vcpkg and Visual Studio 2017 or 2019. Inorder to setup FLINT on Linux based systems, Docker containers are preferred. This method creates containers which are a simple way to build FLINT and all required dependencies.

Before Setting Up FLINT

- Please make sure that all the prerequisites required have been installed and configured correctly.
- The repository has been forked and cloned following the Git and Github guide [here](#) .
- It is highly recommended to first setup the FLINT.example repository to get an overview of how FLINT works since the software might seem a bit complex at first sight. The instructions for setting up FLINT.example repository can be found [here](#).

Datasets for FLINT

After setting up FLINT, the next step can be to explore and run FLINT on different datasets. We have a collection of publicly available Opensource datasets for you to choose from and run on FLINT [here](#).

These datasets have been used by many countries and are effective in real-world scenarios. For each dataset information is provided on content and license. Proper permissions have been cited for all the datasets and we urge you to follow the license for every dataset before proceeding to work with it.

If you have another dataset on your mind that would be beneficial for FLINT, please feel free to reach out to us on info@moja.global to share details about this dataset.

Working with Data

Contents:

2.1 Git and Github guide

This guide is to help new contributors setup git, github and navigate their way through making contributions to moja global repositories. It covers the entire process of contributing right from installing git to opening pull requests.

2.1.1 Setup this project using Git

Before setting up this project using Git make sure you have installed and configured git by following the instructions [here](#).

2.1.2 Fork and Clone this project

- In your browser, visit <https://github.com/moja-global/FLINT>. In the upper left corner, there is a **Fork** button. Please click on it to create a fork/copy of the repository on your profile.
- In the terminal screen, clone this repo by running the command where `your-username` represents your Github username.

```
git clone https://github.com/<your-username>/FLINT/
```

- Enter into the newly created project folder by running the command

```
cd FLINT
```

- Configure upstream for the fork so that git can sync work from the upstream if it is updated by running the command

```
git remote add upstream https://github.com/moja-global/FLINT/
```

- Check if upstream is configured by running the command and check if upstream is shown or not.

```
git remote -v
```

- Now, the project is setup using Git. Please carry on with instructions on how to set this up on [Windows](#) or [Linux](#) [here](#). You can revisit this section when you are ready to make a contribution.

2.1.3 Claim an issue

This section will demonstrate how to claim an issue to work on using botmojaglobal.

To work on an issue, claim it by adding a comment with `@botmojaglobal claim` to the issue thread. botmojaglobal is a GitHub workflow bot forked from the [zulipbot](#); it will assign you to the issue and label the issue as `in progress`. Some additional notes:

- You can only claim issues with the `Good for newcomers` or `Help Wanted` labels. botmojaglobal will give you an error if you try to claim an issue without one of those labels.
- Please feel free to ask questions on how to approach the issue or if the tests are failing. The maintainers/reviewers will try to get back to you as soon as possible. You can reach us on the moja-global [slack](#), or through Github.
- If your pull request has some requested changes, after working on it don't forget to leave a comment asking for a review since the reviewers aren't notified when a pull request is updated.

2.1.4 Make a contribution

This section will show you step-by-step how to make a contribution to FLINT using git.

- FLINT stable branch is **develop**. Releases are scheduled periodically when codebase is production-ready and develop is merged into master. develop branch is the latest updated branch and should be used as a base branch for development. All pull requests should be against develop branch only. Make sure you are in the project directory and checkout to develop branch with this command.

```
git checkout develop
```

- Choose an issue to work on. We have issues specifically labelled *Good for newcomers* and *Help Wanted* for new contributors to claim. Before starting to work on any issue, make sure you have claimed it.
- Create a new feature branch from develop branch to work on. The feature branch should have a short name that is relevant to the issue that you will be working on. For example, if you are working on improving documentation in the readme for adding a badge, the branch can be name `add_badge_readme`.

```
git checkout -b <feature-branch-name>
```

- Work on the task. Add tests and documentation for your changes if required. When you are done with your changes, you can check all the files changes using the following command.

```
git status
```

- Add the relevant files and commit the changes. Please make sure that only those files required for this contribution are added. You can later modify your pull request to add other files as per your requirement.

```
git add <file> <file> ...
```

- While committing the changes, make sure your commit message follows our commit-message guidelines mentioned here.

```
git commit -m "relevant commit message"
```

- Make sure your fork is in sync with the latest changes of develop. For this rebase your branch against the latest develop by following the commands below.

```
git checkout develop
git pull origin develop
git checkout <your-branch-name>
git rebase develop
```

- Incase there are any merge conflicts on running the rebase command, follow this guide to resolve them.
- You can now push your changes onto your feature branch using the command below.

```
git push origin <your-branch-name>
```

2.1.5 Create a pull request for your contribution

You can now create a pull request to get your changes merged into the upstream develop branch. Follow this step-by-step guide to create a pull request on Github.

- Navigate to the pull requests tab under FLINT. Click on the **New pull request** button. Compare your feature branch against the **develop** branch to create the pull request. Fill the pull request template by linking the issue number solved.
- In case your pull request is a work in progress, don't forget to add "WIP" in the title of your pull request to let the maintainers know that the pull request is not ready for review yet.
- Please be patient, someone from our team will review your pull request shortly and provide feedback. In case there are changes requested, you can follow the section below on how to update/modify your pull request.
- Also make sure that your pull request is in sync with the latest develop at all times.

NOTE: Don't forget to get credits for your contributions once it gets merged by following this guide [here](#).

2.1.6 Modify your pull request

In case your pull request needs further changes, you can update your pull request by following the steps below.

- Checkout on your feature branch of the pull request.
- Add the changes as required and commit using the amend flag. This will update the last commit thus keeping the commit history clean and within a single commit.

```
git add <file1> <file2>
git commit -amend
```

- Push this onto your feature branch but this time with force flag. This will update the pull request automatically. The reviewer won't be notified about this updation, so leave a comment in your pull request if you want a review.

```
git push origin <your-branch-name> --force
```

2.2 Windows Installation

This section guides first-time contributors through installing FLINT development environment.

Before proceeding further, make sure you have setup the project using Git by following our guide [Git and GitHub Guide](#). Also make sure you have the following prerequisites setup -

2.2.1 Prerequisites

- [Cmake](#)
- [Visual Studio](#)
- [Vcpkg](#)

Now that you have all the necessary prerequisites, you can proceed with the Installation.

2.2.2 Using vcpkg to install required libraries

Start a command shell in the Vcpkg repository folder (that you had cloned earlier) and use the following commands:

```
# bootstrap
bootstrap-vcpkg.bat

# install packages
vcpkg.exe install boost-test:x64-windows boost-program-options:x64-windows boost-
↳ log:x64-windows turtle:x64-windows zipper:x64-windows poco:x64-windows libpq:x64-
↳ windows gdal:x64-windows sqlite3:x64-windows boost-ublas:x64-windows
```

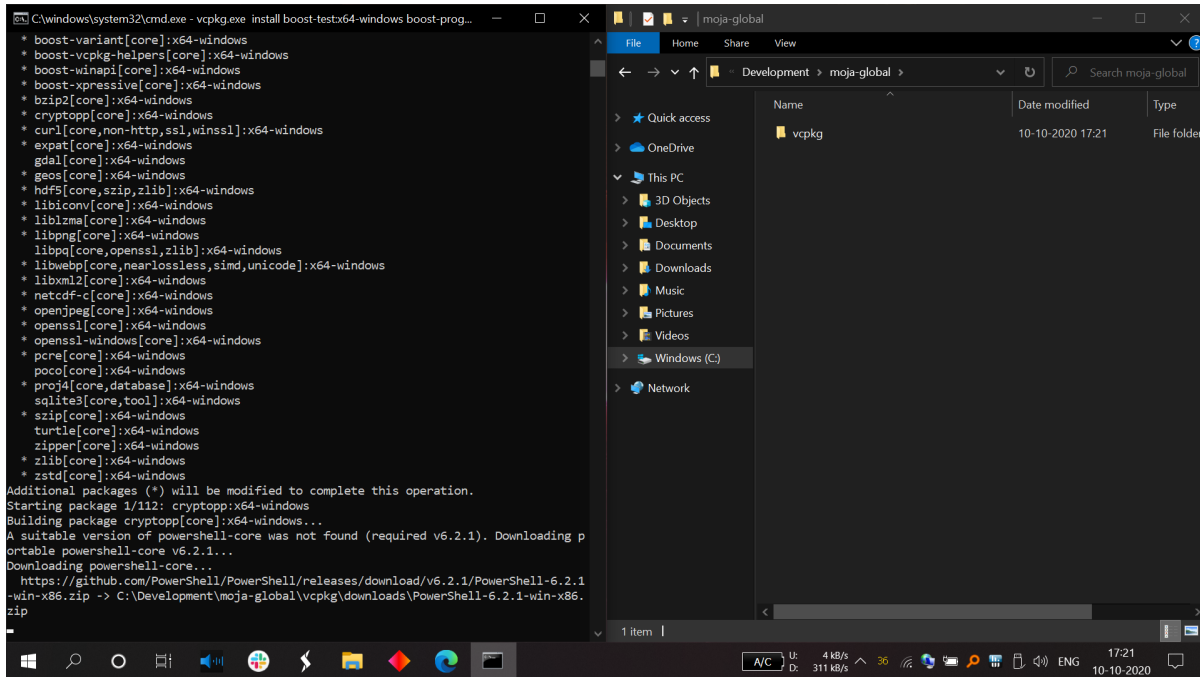


Fig. 1: Installing required packages using vcpkg in Command Prompt

2.2.3 Using cmake to build the project

Note: Please make sure that you have checked out to the `Develop` Branch for the FLINT Installation. You may refer to our [Git and GitHub Guide](#) for instructions on how to switch to develop branch.

Once this has completed, start a command shell in your FLINT repository folder. Now use the following commands to create the Visual Studio solution:

```
# Create a build folder under the Source folder
cd Source
mkdir build
cd build

# from ..\moja\FLINT\source\build
# now create the Visual Studio Solution (2019)
cmake -G "Visual Studio 16 2019" -DCMAKE_INSTALL_PREFIX=..\..\.. -DVCPKG_TARGET_
↳ TRIPLET=x64-windows -DENABLE_TESTS=OFF -DENABLE_MOJA_MODULES_ZIPPER=OFF -DCMAKE_
↳ TOOLCHAIN_FILE=..\..\..\vcpkg\scripts\buildsystems\vcpkg.cmake ..
```

(continues on next page)

(continued from previous page)

```
# OR Visual Studio Solution (2017)
cmake -G "Visual Studio 15 2017" -DCMAKE_INSTALL_PREFIX=..\..\.. -DVCPKG_TARGET_
→TRIPLET=x64-windows -DENABLE_TESTS=OFF -DENABLE_MOJA_MODULES_ZIPPER=OFF -DCMAKE_
→TOOLCHAIN_FILE=..\..\..\vcpkg\scripts\buildsystems\vcpkg.cmake ..
```

Note: All paths used below with `C:\Development\moja-global` will need to be modified to match your system build location of the moja project.

2.2.4 Running the project

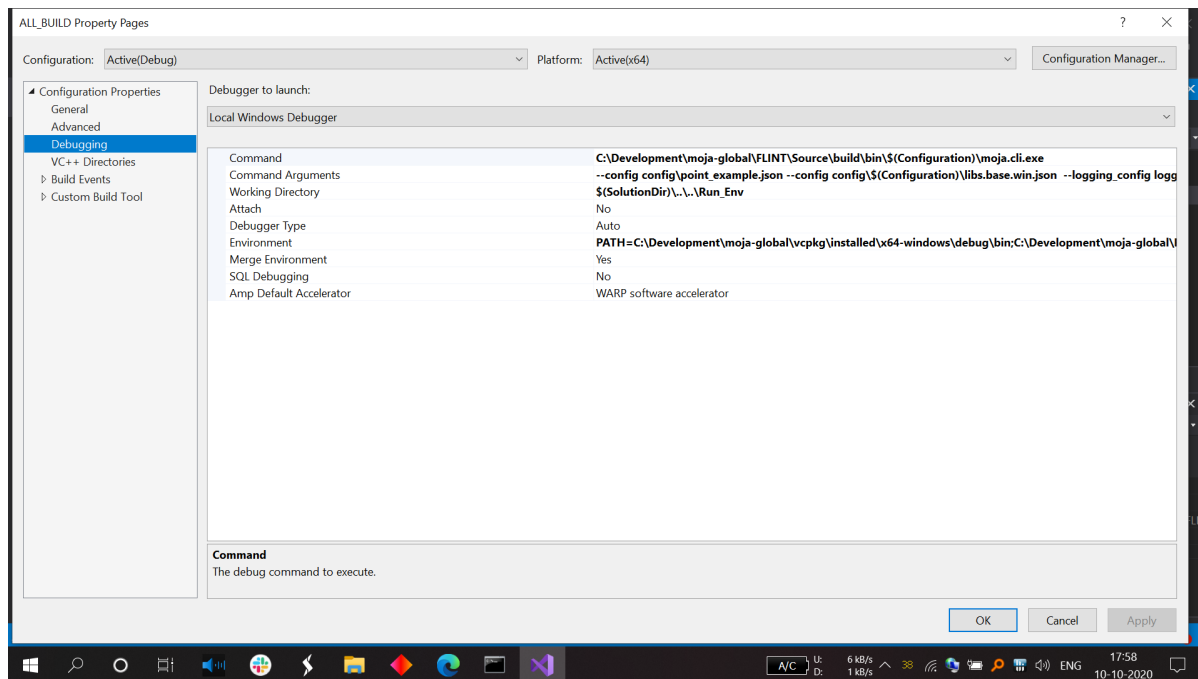


Fig. 2: Running `moja.cli.exe` in Visual Studio Debugging All properties page

We are running the `moja.cli.exe` from the `moja.FLINT` project here. In order to make edits to the Visual Studio Solution we can use the CMake GUI.

2.2.5 Edit solution using CMake GUI

- Launch the CMake GUI
- In the Where to build the binaries field click `Browse Build...` and select the folder you created above (i.e. `C:\Development\moja-global\FLINT\Source\build`). The Where is the source code: field should update, if not, set it correctly.
- You should be able to edit any CMake setting now (i.e. `ENABLE` flags like `ENABLE_TESTS`), then click `Configure` – assuming all libraries and required software has been installed you should have no errors. Now click `Generate` and the Solution with adjustments should be ready to load into Visual Studio.

2.3 Docker Installation (for Mac and Linux Variants)

This section guides first-time contributors through installing FLINT development environment through Docker on Mac and Linux systems.

Before proceeding further, make sure you have setup the project using Git by following our [Git and GitHub Guide](#). Also make sure you have the following prerequisites setup -

2.3.1 Prerequisites

- [Docker](#)

Now that you have all the necessary prerequisites, you can proceed with the Installation.

2.3.2 Setup Docker Container

Containers are a simple way to build FLINT and all required dependencies.

Note: Before setting up, it is recommended to install the [FLINT.example](#) repository first.

2.3.3 Building using prebuilt image

Instead of building the required libraries, pre-built Docker Image is available for FLINT at our [Dockerhub](#) . You can pull and run FLINT using this prebuilt image using the following commands.

```
# pull the image
docker pull mojanglobal/flint
# run a container
docker run --rm -ti mojanglobal/flint:latest bash
# run CLI
moja.cli --help
```

Alternatively, you can build the libraries by Building using the second option below.

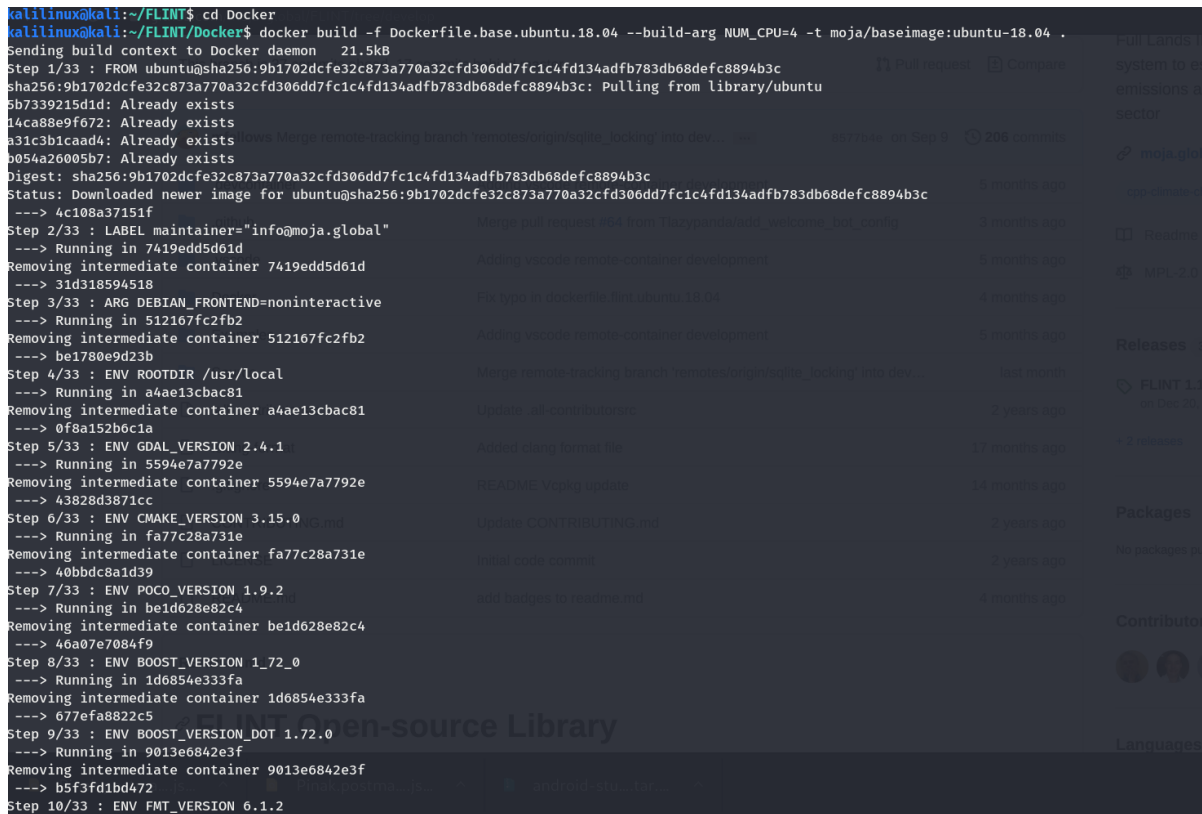
2.3.4 Building the containers

Note: Please make sure that you have checked out to the `Develop` Branch for the FLINT Installation. You may refer to our [Git and GitHub Guide](#) for instructions on how to switch to develop branch.

The build has been split into two Dockerfiles, the first to get and build required libraries. The second to get and build the moja FLINT libraries and CLI program.

```
# working from the Docker folder "flint/tree/develop/Docker"

# build the base
docker build -f Dockerfile.base.ubuntu.18.04 --build-arg NUM_CPU=4 -t moja/
↳baseimage:ubuntu-18.04 .
```



```

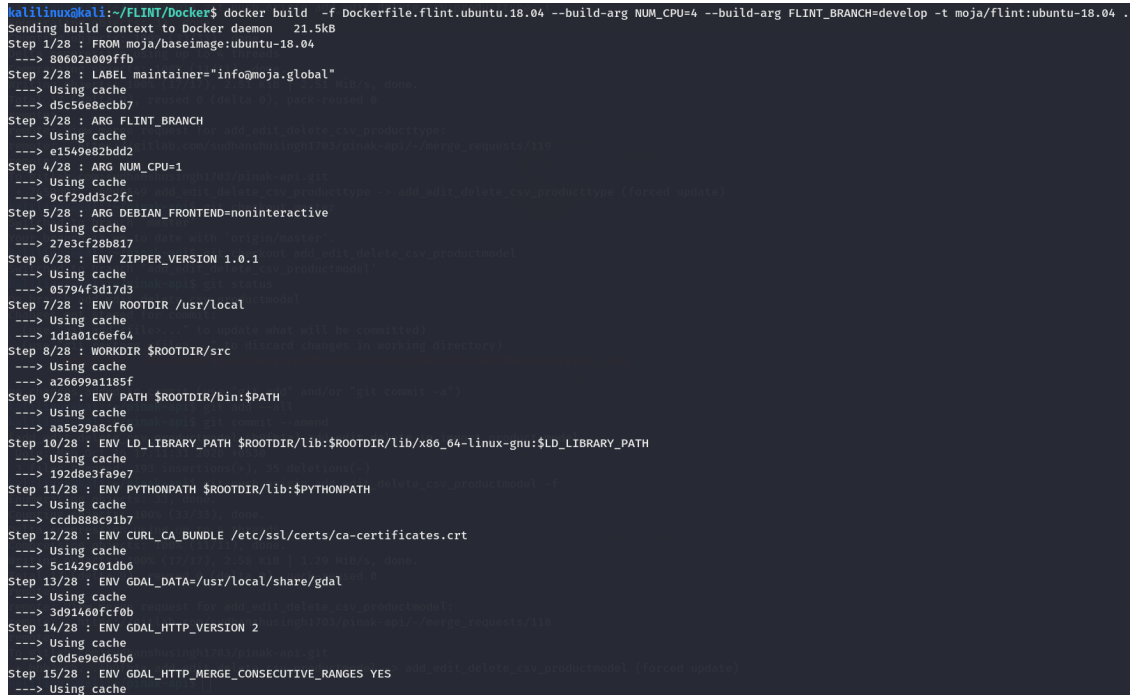
kalilinux@kali:~/FLINT$ cd Docker
kalilinux@kali:~/FLINT/Docker$ docker build -f Dockerfile.base.ubuntu.18.04 --build-arg NUM_CPU=4 -t moja/baseimage:ubuntu-18.04 .
Sending build context to Docker daemon  21.5kB
Step 1/33 : FROM ubuntu@sha256:9b1702dcfe32c873a770a32cfd306dd7fc1c4fd134adfb783db68defc8894b3c
sha256:9b1702dcfe32c873a770a32cfd306dd7fc1c4fd134adfb783db68defc8894b3c: Pulling from library/ubuntu
5b7339215d1d: Already exists
14ca88e9f672: Already exists
a31c3b1caad4: Already exists
0054a26005b7: Already exists
Digest: sha256:9b1702dcfe32c873a770a32cfd306dd7fc1c4fd134adfb783db68defc8894b3c
Status: Downloaded newer image for ubuntu@sha256:9b1702dcfe32c873a770a32cfd306dd7fc1c4fd134adfb783db68defc8894b3c
----> 4c108a37151f
Step 2/33 : LABEL maintainer="info@moja.global"
----> Running in 7419edd5d61d
Removing intermediate container 7419edd5d61d
----> 31d318594518
Step 3/33 : ARG DEBIAN_FRONTEND=noninteractive
----> Running in 512167fc2fb2
Removing intermediate container 512167fc2fb2
----> be1780e9d23b
Step 4/33 : ENV ROOTDIR /usr/local
----> Running in a4ae13cbac81
Removing intermediate container a4ae13cbac81
----> 0f8a152b6c1a
Step 5/33 : ENV GDAL_VERSION 2.4.1
----> Running in 5594e7a7792e
Removing intermediate container 5594e7a7792e
----> 43828d3871cc
Step 6/33 : ENV CMAKE_VERSION 3.15.0
----> Running in fa77c28a731e
Removing intermediate container fa77c28a731e
----> 40bbdc8a1d39
Step 7/33 : ENV POCO_VERSION 1.9.2
----> Running in be1d628e82c4
Removing intermediate container be1d628e82c4
----> 46a07e7084f9
Step 8/33 : ENV BOOST_VERSION 1_72_0
----> Running in 1d6854e333fa
Removing intermediate container 1d6854e333fa
----> 677efa8822c5
Step 9/33 : ENV BOOST_VERSION_DOT 1.72.0
----> Running in 9013e6842e3f
Removing intermediate container 9013e6842e3f
----> b5f3fd1bd472
Step 10/33 : ENV FMT_VERSION 6.1.2

```

Fig. 3: Building the base libraries using Docker

```
# build the flint container
docker build -f Dockerfile.flint.ubuntu.18.04 --build-arg NUM_CPU=4 --build-arg FLINT_BRANCH=develop -t moja/flint:ubuntu-18.04 .

docker build -f Dockerfile.flint.ubuntu.18.04 --build-arg NUM_CPU=4 --build-arg GITHUB_AT=XXXX --build-arg FLINT_BRANCH=develop -t moja/flint:ubuntu-18.04 .
```



```
kallinux@kali:~/FLINT/Docker$ docker build -f Dockerfile.flint.ubuntu.18.04 --build-arg NUM_CPU=4 --build-arg FLINT_BRANCH=develop -t moja/flint:ubuntu-18.04 .
Sending build context to Docker daemon 21.5kB
Step 1/28 : FROM moja/baseimage:ubuntu-18.04
--> 80602a009ffb
Step 2/28 : LABEL maintainer="info@moja.global"
--> Using cache
--> d5c56e9ecbb7
Step 3/28 : ARG FLINT_BRANCH
--> Using cache
--> e1549e82bdd2
Step 4/28 : ARG NUM_CPU=1
--> Using cache
--> 9cf29dd3c2fc
Step 5/28 : ARG DEBIAN_FRONTEND=noninteractive
--> Using cache
--> 27e3cf28b817
Step 6/28 : ENV ZIPPER_VERSION 1.0.1
--> Using cache
--> 05794f3d17d3
Step 7/28 : ENV ROOTDIR /usr/local
--> Using cache
--> 1d1a01c6ef64
Step 8/28 : WORKDIR $ROOTDIR/src
--> Using cache
--> a26699a1185f
Step 9/28 : ENV PATH $ROOTDIR/bin:$PATH
--> Using cache
--> aa5e29a8cf66
Step 10/28 : ENV LD_LIBRARY_PATH $ROOTDIR/lib:$ROOTDIR/lib/x86_64-linux-gnu:$LD_LIBRARY_PATH
--> Using cache
--> 192d8e3fa9e7
Step 11/28 : ENV PYTHONPATH $ROOTDIR/lib:$PYTHONPATH
--> Using cache
--> ccd888e91b7
Step 12/28 : ENV CURL_CA_BUNDLE /etc/ssl/certs/ca-certificates.crt
--> Using cache
--> 5c1429c01db6
Step 13/28 : ENV GDAL_DATA=/usr/local/share/gdal
--> Using cache
--> 3d91460fcf0b
Step 14/28 : ENV GDAL_HTTP_VERSION 2
--> Using cache
--> c0d5e9ed65b6
Step 15/28 : ENV GDAL_HTTP_MERGE_CONSECUTIVE_RANGES YES
--> Using cache
```

Fig. 4: Building the FLINT libraries using Docker

How to use the final container depends on the task. However, the following command will bash into the flint container and allow you to use the CLI program.

```
# run bash on the flint container
docker run --rm -ti moja/flint:ubuntu-18.04 bash
```

Once in, you should be able to run the CLI program `moja.cli`

```
# run CLI
moja.cli --help
```

That should respond with the following options:

2.3.5 Allowed options

General options:

<code>-h [--help]</code>	produce a help message
<code>--help-section arg</code>	produce a help message for a named section
<code>-v [--version]</code>	output the version number

Commandline only options:

(continues on next page)

```

Successfully tagged moja/flint:ubuntu-18.04
kalilinux@kali:~/FLINT/Docker$ docker run --rm -ti moja/flint:ubuntu-18.04 bash
moja@b29c9c3cd080:~$ moja.cli --help
Allowed options:
  --help            produce a help message
  --help-section arg produce a help message for a named section
  --version         output the version number

General options:
  -h [ --help ]          produce a help message
  --help-section arg     produce a help message for a named section
  -v [ --version ]       output the version number

Commandline only options:
  --logging_config arg   path to Moja logging config file
  --config_file arg      path to Moja run config file
  --provider_file arg    path to Moja data provider config file

Configuration file options:
  --config arg           path to Moja project config files
  --config_provider arg  path to Moja project config files for data providers
moja@b29c9c3cd080:~$

```

Fig. 5: Running moja.cli using Docker

(continued from previous page)

```

--logging_config arg   path to Moja logging config file
--config_file arg      path to Moja run config file
--provider_file arg    path to Moja data provider config file

Configuration file options:
  --config arg           path to Moja project config files
  --config_provider arg  path to Moja project config files for data providers

```

2.4 FLINT.example

The FLINT.example gives an example of how to build and run libraries using the FLINT framework. It is recommended to set up FLINT.example repository before setting up FLINT in order to get a better idea of how FLINT works.

The Docker file used here can be found in the Dockerfile file at the root of the repository. This Docker file builds from the image `mojaglobal/flint:bionic` which can be found in docker hub.

There are 3 different environments listed in this document to build and run the examples:

- **Windows - Visual Studio 2019:** develop, run and debug
- **Visual Studio Code:** develop, run and debug
- **Docker:** run only

We currently have four different sample runs:

- **Test Module sample** - Point level
- **RothC sample** - Point level
- **Chapman richards** - Point sample
- **Chapman richards** - Spatial sample

The FLINT.example repository is available [here](#) under the moja global organisation on GitHub. Before proceeding to the instructions for installing FLINT.example, please follow the following steps to clone this repository on your fork:

```
git clone https://github.com/<your-username>/FLINT.example.git
```

For more instructions on our GitHub fork, clone and pull request practices, refer our [Git and GitHub Guide](#).

Contents:

2.4.1 Environment: Visual Studio

In the Visual Studio environment option for setting up FLINT.example, the options to run, develop and debug the repository code is available. Also make sure you have the following prerequisites setup -

Prerequisites

- [Cmake](#)
- [Visual Studio](#)
- [Docker](#)

Now that you have all the necessary prerequisites, you can proceed with the Installation.

Using vcpkg to install required libraries

Start a command shell in the Vcpkg repository folder (that you had cloned earlier) and use the following commands:

```
# bootstrap
bootstrap-vcpkg.bat

# install packages
vcpkg.exe install boost-test:x64-windows boost-program-options:x64-windows boost-
↳ log:x64-windows turtle:x64-windows zipper:x64-windows poco:x64-windows libpq:x64-
↳ windows gdal:x64-windows sqlite3:x64-windows boost-ublas:x64-windows fmt:x64-windows
```

Building the project

Launch the Windows Powershell and run the following commands:

```
# Create a build folder under the Source folder
mkdir -p Source\build
cd Source\build
```

Now depending on which type of simulation you want to execute, you may run one of the following generate commands:

Commands to run cmake for the point simulations:

```
# Point simulations
# Generate the project files
cmake -G "Visual Studio 16 2019" -DCMAKE_INSTALL_PREFIX=C:\Development\Software\moja -
↳ DVCPKG_TARGET_TRIPLET=x64-windows -DOPENSSL_ROOT_DIR=c:\Development\moja-
↳ global\vcpkg\installed\x64-windows -DENABLE_TESTS=OFF -DCMAKE_TOOLCHAIN_
↳ FILE=c:\Development\moja-global\vcpkg\scripts\buildsystems\vcpkg.cmake ..
```

Commands to run cmake for the spatial simulations:

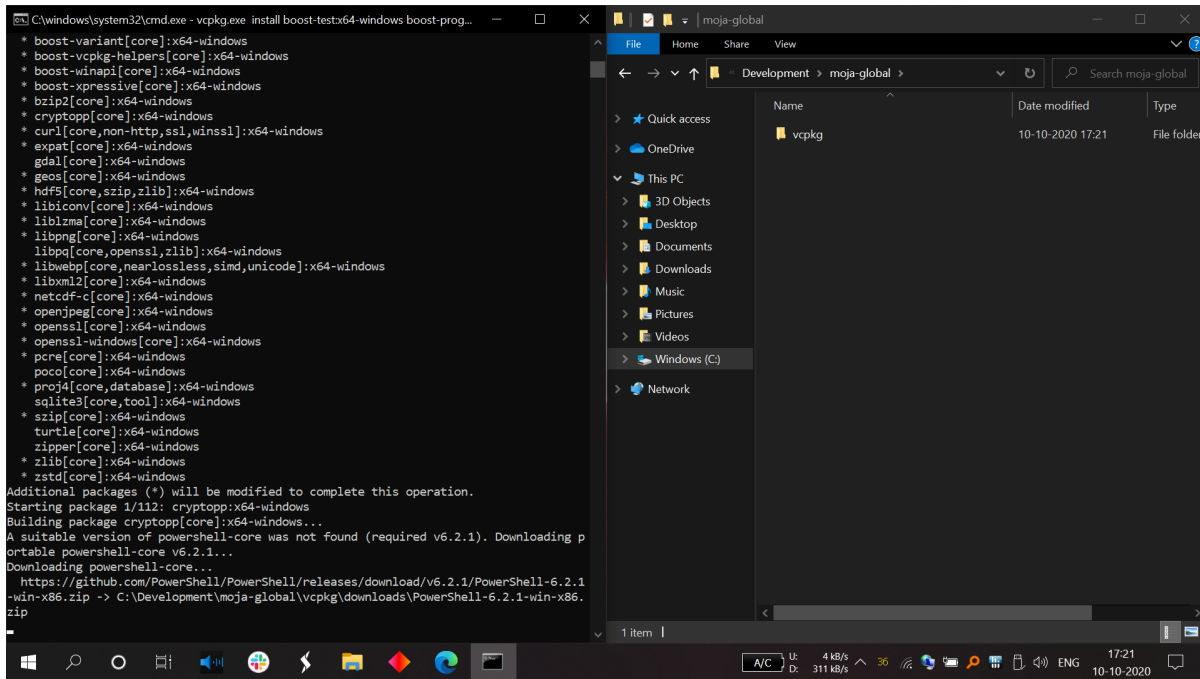


Fig. 6: Installing required packages using vcpkg in Command Prompt

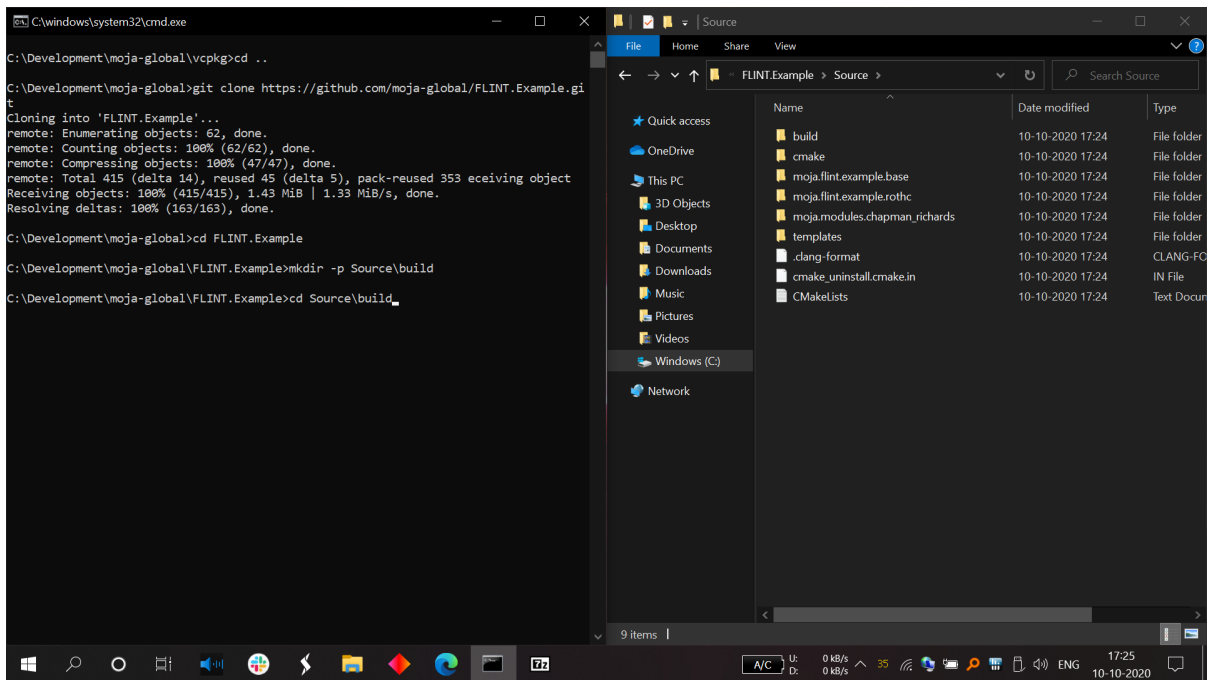


Fig. 7: Creating a build Directory for Cloned FLINT.example repo

```
# Spatial simulations
# if your planning to run spatial chapman richards example you also need to enable_
↪ the gdal module
# Generate the project files
cmake -G "Visual Studio 16 2019" -DCMAKE_INSTALL_PREFIX=C:\Development\Software\moja -
↪ DVCPKG_TARGET_TRIPLET=x64-windows -DOPENSSL_ROOT_DIR=c:\Development\moja-
↪ global\vcpkg\installed\x64-windows -DENABLE_TESTS=OFF -DENABLE_MOJA_MODULES_GDAL=ON_
↪ -DCMAKE_TOOLCHAIN_FILE=c:\Development\moja-global\vcpkg\scripts\buildsystems\vcpkg.
↪ cmake ..
```

Running the project

In order to run and debug the Visual Studio solution -

- Open the visual studio solution that CMake created at C:\Development\moja-global\FLINT\Source\build\moja.sln
- Build the debug configuration ALL_BUILD target by right clicking the ALL_BUILD node and selecting Build.

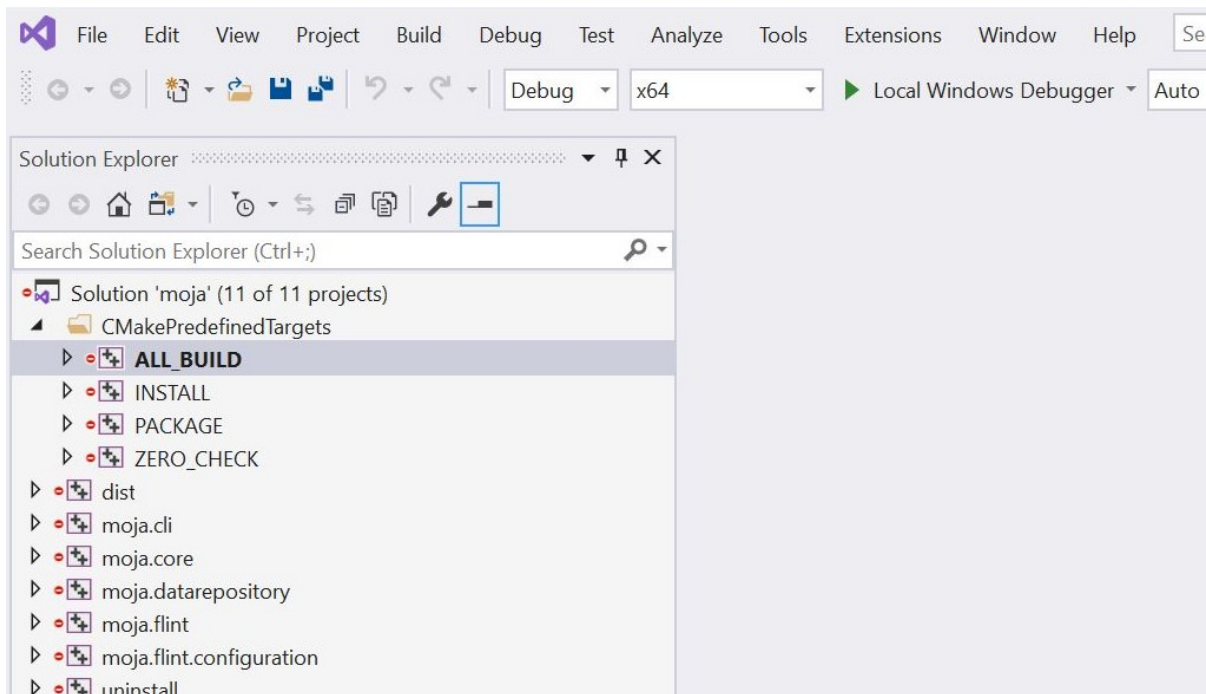


Fig. 8: Navigating to Visual Studio Debugging ALL_BUILD properties page

Running in the IDE and debugging is a little tricky. This could more than likely be resolved with better cmake setups. But for now there is some setup that can make running and debugging work.

The issue is we want to run with the `moja.cli.exe` from the `moja.FLINT` project, but debug in our current IDE (`FLINT.example`).

The solution is to use properties to setup a Debug run in the IDE, making the command run `moja.cli.exe`.

NOTE : All paths used below with `C:\Development\moja-global` will need to be modified to match your system build location of the moja project.

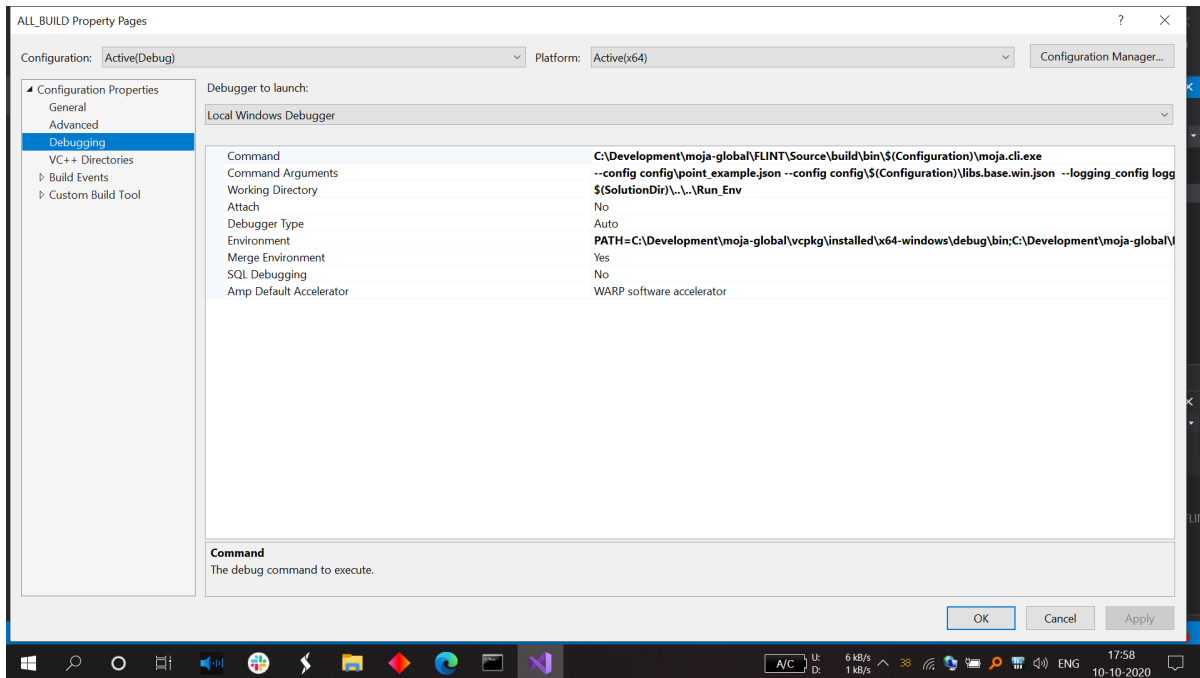


Fig. 9: Running moja.cli.exe in Visual Studio Debugging All properties page

Test Module Example

The settings required in VS2019 are:

```
# Command
C:\Development\moja-global\FLINT\Source\build\bin\$(Configuration)\moja.cli.exe

# Command Args
--config config\point_example.json --config config\$(Configuration)\libs.base.win.
→json --logging_config logging.debug_on.conf

# Working Directory
$(SolutionDir)\..\..\Run_Env

# Environment Debug
PATH=C:\Development\moja-global\vcpkg\installed\x64-windows\debug\bin;
→C:\Development\moja-global\FLINT\Source\build\bin\$(Configuration);%PATH%
LOCAL_LIBS=$(OutDir)
MOJA_LIBS=C:\Development\moja-global\FLINT\Source\build\bin\$(Configuration)

# Environment Release
PATH=C:\Development\moja-global\vcpkg\installed\x64-windows\bin;C:\Development\moja-
→global\FLINT\Source\build\bin\$(Configuration);%PATH%
LOCAL_LIBS=$(OutDir)
MOJA_LIBS=C:\Development\moja-global\FLINT\Source\build\bin\$(Configuration)
```

With Envs: PATH for various libraries built in the Moja stage and LOCAL_LIBS so we can modify the explicit path for our example config to load libraries from this vs build (the default is the same location as the EXE).

To match this, the example point config uses an environment variable in the library path:


```
{
  "Libraries": {
    "moja.flint.example.base": {
      "library": "moja.flint.example.based.dll",
      "path": "%LOCAL_LIBS%",
      "type": "external"
    }
  }
}
```

RothC example

We also have a RothC example for point level simulations. In order to run this example, you may modify the following arguments in the above test settings command arguments. These arguments will point at the right configuration files for RothC. Please follow the following steps to set the correct configuration -

- Build the debug configuration ALL_BUILD target by right clicking the ALL_BUILD node and selecting Build. Then right click the the moja.flint.example.rothc node and select Set as Startup Project then right click again and select properties. Navigate to Configuration Properties/Debugging properties pane and configure the following:

- Command: C:\Development\moja-global\FLINT\Source\build\bin\Debug\moja.cli.exe
- Command Arguments:

```
--config config\point_example.json --config config\debug\libs.base.win.json --
--logging_config logging.debug_on.conf
```

- Working Directory: \$(SolutionDir)..\..\Run_Env
- Environment:

```
PATH=C:\Development\moja-global\FLINT\Source\build\bin\Debug;%PATH% LOCAL_
LIBS=C:\Development\moja-global\FLINT.Example\Source\build\bin\Debug
```

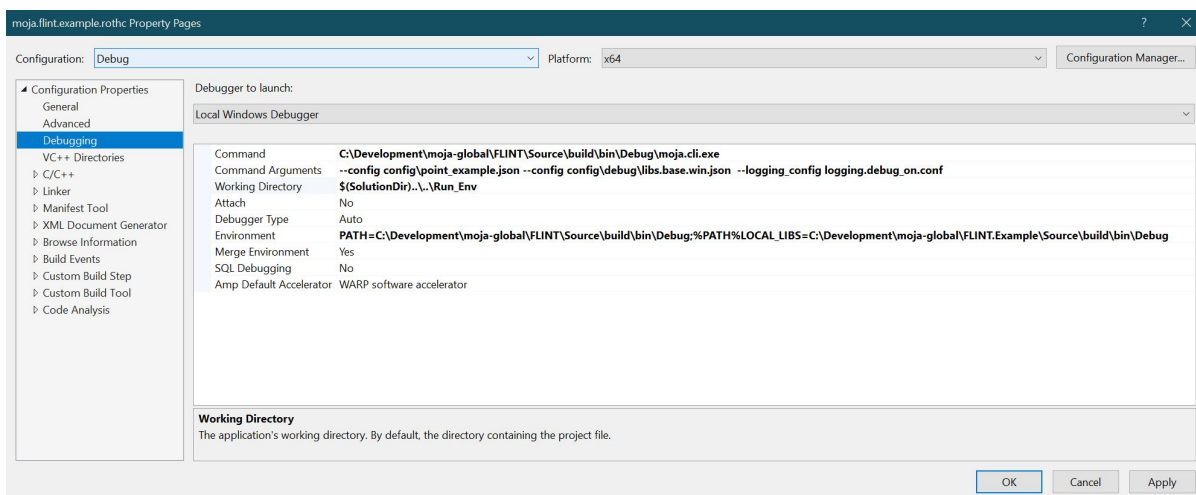


Fig. 10: ALL_BUILD properties page for moja.FLINT.example.rothc

You should now be able to select `Debug->Start Debugging` to start a debug run of the RothC example. You should see something like the following:

Enable moja.modules.GDAL

Before moving on to setting up the Chapman Richards model, we need to enable the `moja.modules.GDAL` flags. We can toggle these flags by clicking on `BROWSE BUILD` and setting it to the build directory where we just built the solution.

- Open the solution that CMake created at `C:\Development\moja-global\FLINT\Source\build\moja.sln`.
- Check the following Flags present:
- `ENABLE_MOJA.MODULES.GDAL`
- `ENABLE_MOJA.MODULES.LIBPQ`
- `ENABLE_MOJA.MODULES.POCO`
- `ENABLE_MOJA.MODULES.ZIPPER`
- Now, Click on `Configure` option twice.
- Click on `Generate` and then you may explore all the enabled modules in Solution Explorer by clicking on `Open Project`.
- Open `CMakePredefinedTargets`, right click on `ALL_BUILD` and click on `Build`

Viola! All libraries have been enabled You may now proceed with the Chapman Richards example!

Chapman Richards example

Based on the moja global repository [Chapman Richards](#) , this sample can be run on both point and spatial versions (over Dominica). Inorder to run this example, you may modify the following arguments in the above test settings command arguments. These arguments will point at the right configuration files for Chapman Richards.

```
# Command Args
# Point
--config config/point_forest_config.json --config config/${Configuration}/libs.gdal.
↪chaprich.win.json
# Spatial
--config config/forest_config.json --config config/${Configuration}/libs.gdal.
↪chaprich.win.json --config_provider config/forest_provider.json
```

2.4.2 Environment: Visual Studio (Remote Containers)

In the Visual Studio environment using Remote Containers option for setting up FLINT.example, the options to run, develop and debug the repository code is available. Here remote containers are available in Visual Studio as an extension and can be installed with the help of the link mentioned in the prerequisites section.

Prerequisites

- [Visual Studio](#)
- [Remote - Container \(VS Extension\)](#)

```

Microsoft Visual Studio Debug Console

<2020-10-19 11:00:24.883380> (info) - Config has files: 2
<2020-10-19 11:00:24.886497> (info) - Using Logging Configuration: logging.debug_on.conf
<2020-10-19 11:00:24.887497> (info) - Using configurations:
<2020-10-19 11:00:24.887497> (info) - config.point_example.json
<2020-10-19 11:00:24.887497> (info) - config.debug.libs.base.win.json
<2020-10-19 11:00:24.888497> (debug) - moja::flint::configuration::JSON2ConfigurationProvider::createLibraries(434) - details (moja.flint.example.base):
path - %LOCAL_LIBS%, filename - moja.flint.example.base.dll
<2020-10-19 11:00:24.899497> (debug) - moja::flint::configuration::Configuration::addLibrary(105) - details (moja.flint.example.base): path - %LOCAL_LIBS%,
filename - moja.flint.example.base.dll
<2020-10-19 11:00:24.910511> (debug) - moja::flint::LibraryManager::LibraryManager(46) - LibraryManager: constructor no args
<2020-10-19 11:00:24.911499> (debug) - moja::flint::LibraryManager::LoadInternalLibrary(427) - LibraryLoadInternalLibraryManager: entered
<2020-10-19 11:00:24.912497> (debug) - moja::flint::LibraryManager::LoadInternalLibrary(431) - LibraryLoadInternalLibraryManager: calling AddLibrary : in
ternal.flint
<2020-10-19 11:00:24.912497> (debug) - moja::flint::LibraryManager::AddLibrary(78) - AddLibrary: entered : internal.flint
<2020-10-19 11:00:24.912497> (debug) - moja::flint::LibraryManager::LoadInternalLibrary(441) - LibraryLoadInternalLibraryManager: calling registrations
<2020-10-19 11:00:24.913497> (debug) - moja::flint::getFlintModuleRegistrations(107) - getFlintModuleRegistrations: entered
<2020-10-19 11:00:24.914507> (debug) - moja::flint::getFlintModuleRegistrations(166) - getFlintModuleRegistrations: exit - 20
<2020-10-19 11:00:24.914507> (debug) - moja::flint::getFlintTransformRegistrations(191) - getFlintTransformRegistrations: entered
<2020-10-19 11:00:24.915497> (debug) - moja::flint::getFlintTransformRegistrations(194) - getFlintTransformRegistrations: exit - 9
<2020-10-19 11:00:24.915497> (debug) - moja::flint::getFlintFlintDataRegistrations(199) - getFlintFlintDataRegistrations: entered
<2020-10-19 11:00:24.915497> (debug) - moja::flint::getFlintFlintDataRegistrations(219) - getFlintFlintDataRegistrations: exit - 8
<2020-10-19 11:00:24.917497> (debug) - moja::flint::getFlintFlintDataFactoryRegistrations(224) - getFlintFlintDataFactoryRegistrations: entered
<2020-10-19 11:00:24.917497> (debug) - moja::flint::getFlintFlintDataFactoryRegistrations(230) - getFlintFlintDataFactoryRegistrations: exit - 1
<2020-10-19 11:00:24.918498> (debug) - moja::flint::getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-19 11:00:24.920658> (debug) - moja::flint::getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-19 11:00:24.922658> (debug) - moja::flint::LibraryManager::RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(416) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(416) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-19 11:00:24.923658> (debug) - moja::flint::getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-19 11:00:24.923658> (debug) - moja::flint::getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(416) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-19 11:00:24.923658> (debug) - moja::flint::LibraryManager::RegisterProviders(416) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-19 11:00:27.121485> (debug) - moja::flint::LocalDomainControllerBase::configure(71) - details (moja.flint.example.base): path - %LOCAL_LIBS%, fi
lename - moja.flint.example.base.dll
<2020-10-19 11:00:27.123484> (debug) - moja::flint::LibraryManager::AddLibrary(78) - AddLibrary: entered : moja.flint.example.base
<2020-10-19 11:00:27.143484> (debug) - moja::flint::LibraryManager::RegisterProviders(407) - RegisterProviders: entered : moja.flint.example.base : count
0
<2020-10-19 11:00:27.144484> (info) - modules loaded:
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: CalendarSequencer
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: OutputFileStream
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: TestModule1
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: TestModule2
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: TestModule3
<2020-10-19 11:00:27.145481> (info) - library: internal.flint, module name: TransactionManagerEndOfStepModule
<2020-10-19 11:00:27.145481> (info) - library: moja.flint.example.base, module name: ErrorScreenWriter
<2020-10-19 11:00:27.145481> (info) - Operation manager: Simple, Version: 1.0, Config: (Kahan[OFF], ZeroTransfers[OFF], AllowNegativeTransfers[ON], WarnN
egativeTransfers[OFF])
OutputFileStream
=====
Started:10/19/20 11:00:27 AUS Eastern Daylight Time
notification,step,stepDate,fracOfStep,stepLenInYears,Pool 1,Pool 2,Pool 3,
onTimingPostInit,0,1919-11-30:23:59:59.999999,1,0,100,100,100,
onOutputStep,1,1920-01-31:23:59:59.999999,1,0,0846994535519126,119,80,101,
onOutputStep,2,1920-02-29:23:59:59.999999,1,0,0792349726775956,114,79,114,70,51,
onOutputStep,3,1920-03-31:23:59:59.999999,1,0,0846994535519126,70,6229,91,877,137,5001,
onOutputStep,4,1920-04-30:23:59:59.999999,1,0,0819672131147541,183,688279,45,87127,70,44045100000001,
onOutputStep,5,1920-05-31:23:59:59.999999,1,0,0846994535519126,49,84978829000001,215,8591277,34,2910840099999,
onOutputStep,6,1920-06-30:23:59:59.999999,1,0,0819672131147541,36,1387766876999,43,1248390729998,306,9860603851,
onOutputStep,7,1920-07-31:23:59:59.999999,1,0,0846994535519126,446,567596367429,68,5428318307697,-215,110428198199,
onOutputStep,8,1920-08-31:23:59:59.999999,1,0,0846994535519126,-454,484816925545,546,266459362273,208,218357563272,
onOutputStep,9,1920-09-30:23:59:59.999999,1,0,0819672131147541,446,590797846939,-863,963491684344,717,372693837406,
onOutputStep,10,1920-10-31:23:59:59.999999,1,0,0846994535519126,934,908074463653,1012,54978304319,-1647,45785750685,
onOutputStep,11,1920-11-30:23:59:59.999999,1,0,0819672131147541,-2735,1846300243,709,105608281153,2326,07900249314,
onOutputStep,12,1920-12-31:23:59:59.999999,1,0,0846994535519126,4286,41313587457,-3910,29282167216,-76,1203142024096,
onOutputStep,13,1921-01-31:23:59:59.999999,1,0,0849315068493151,-1399,34320892396,7527,48348747302,-5828,14027854906,
onOutputStep,14,1921-02-28:23:59:59.999999,1,0,0767123287671233,-8264,12605236091,-5582,88791533766,14147,0139676986,
onOutputStep,15,1921-03-31:23:59:59.999999,1,0,0849315068493151,2358,2886275792,-7951,91991040035,-15306,3887172788,
onOutputStep,16,1921-04-30:23:59:59.999999,1,0,0821917808219178,-29873,9759768702,34601,7351710531,-4427,75919418292,
onOutputStep,17,1921-05-31:23:59:59.999999,1,0,0849315068493151,2364,83159372849,-56137,0363554577,54072,2047617292,
onOutputStep,18,1921-06-30:23:59:59.999999,1,0,0821917808219178,79858,135616858,31142,7992495759,-110700,934866434,
onOutputStep,19,1921-07-31:23:59:59.999999,1,0,0849315068493151,-188901,833636044,88244,1766771275,100957,656958917,
onOutputStep,20,1921-08-31:23:59:59.999999,1,0,0849315068493151,207097,458959599,-289694,472065421,82897,0121058222,
onOutputStep,21,1921-09-30:23:59:59.999999,1,0,0821917808219178,61387,3118397954,144073,932680189,-475161,244519984,
onOutputStep,22,1921-10-31:23:59:59.999999,1,0,0849315068493151,-726406,447886715,-127233,46094836,853939,908835076,
onOutputStep,23,1921-11-30:23:59:59.999999,1,0,0821917808219178,1490292,39853028,-880711,65177855,-609280,746751728,
onOutputStep,24,1921-12-31:23:59:59.999999,1,0,0849315068493151,-1354816,03221816,2377735,94397864,-1022519,91175948,
onOutputStep,25,1922-01-31:23:59:59.999999,1,0,0849315068493151,-1117079,83885587,-2950258,8138742,4067638,6727301,
onOutputStep,26,1922-02-28:23:59:59.999999,1,0,0767123287671233,6395905,58002461,22925,5904244734,-6418531,17044908,
onOutputStep,27,1922-03-31:23:59:59.999999,1,0,0849315068493151,-11482383,1179765,8303214,45881975,3179468,65915676,
onOutputStep,28,1922-04-30:23:59:59.999999,1,0,0821917808219178,8182123,23753653,-19078705,2827793,10896882,0452428,
onOutputStep,29,1922-05-31:23:59:59.999999,1,0,0849315068493151,13781717,2761508,2016112,8501872,-3395730,126338,
onOutputStep,30,1922-06-30:23:59:59.999999,1,0,0821917808219178,-54731235,0710889,7828176,03390251,46903359,0371863,
onOutputStep,31,1922-07-31:23:59:59.999999,1,0,0849315068493151,86305375,4867343,-7506493,6093668,-11240381,8773675,
onOutputStep,32,1922-08-31:23:59:59.999999,1,0,0849315068493151,-42639781,6432979,149729334,937438,-107089253,29414,
onOutputStep,33,1922-09-30:23:59:59.999999,1,0,0821917808219178,-146771052,915279,-130296389,605006,277067736,520286,
onOutputStep,34,1922-10-31:23:59:59.999999,1,0,0849315068493151,45686243,289809,-12654176,98736,-331207766,902449,
onOutputStep,35,1922-11-30:23:59:59.999999,1,0,0821917808219178,-630558244,777592,656748004,770432,-26189459,9928397,
onOutputStep,36,1922-12-31:23:59:59.999999,1,0,0849315068493151,150145178,043947,-1148099720,59609,997954842,552139,
onOutputStep,37,1923-01-31:23:59:59.999999,1,0,0849315068493151,1441909161,9895,769238591,755174,-2211147453,74468,
onOutputStep,38,1923-02-28:23:59:59.999999,1,0,0767123287671233,-3727182454,67642,-1489866514,70877,2237320139,95765,
onOutputStep,39,1923-03-31:23:59:59.999999,1,0,0849315068493151,4451761744,93473,-5590268498,43373,113850703,-479,
onOutputStep,40,1923-04-30:23:59:59.999999,1,0,0821917808219178,360846986,197297,8582424517,65801,-8943271203,85531,
onOutputStep,41,1923-05-31:23:59:59.999999,1,0,0849315068493151,-13433728189,6036,-3822111176,77252,17255839666,3761,
onOutputStep,42,1923-06-30:23:59:59.999999,1,0,0821917808219178,2974119559,7815,-15552791058,0984,-14188528201,6831,
onOutputStep,43,1923-07-31:23:59:59.999999,1,0,0849315068493151,-30063302888,4422,46440110956,7652,-16376807768,3229,

```

Fig. 11: Debug run for rothc example

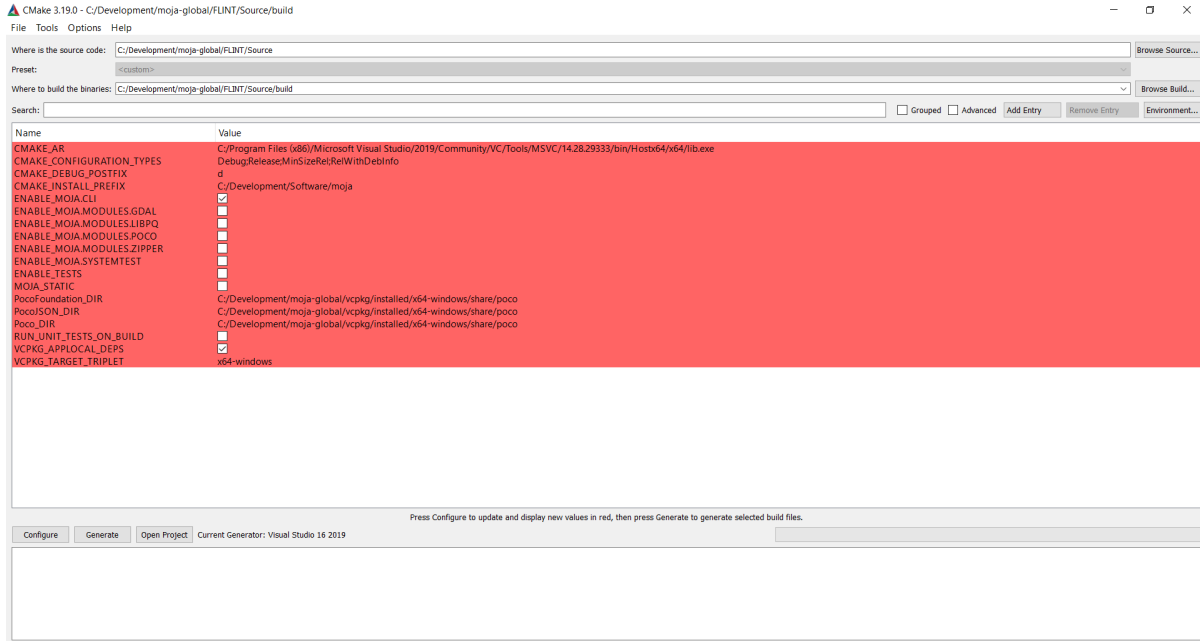


Fig. 12: Navigating to Cmake Configuration Page

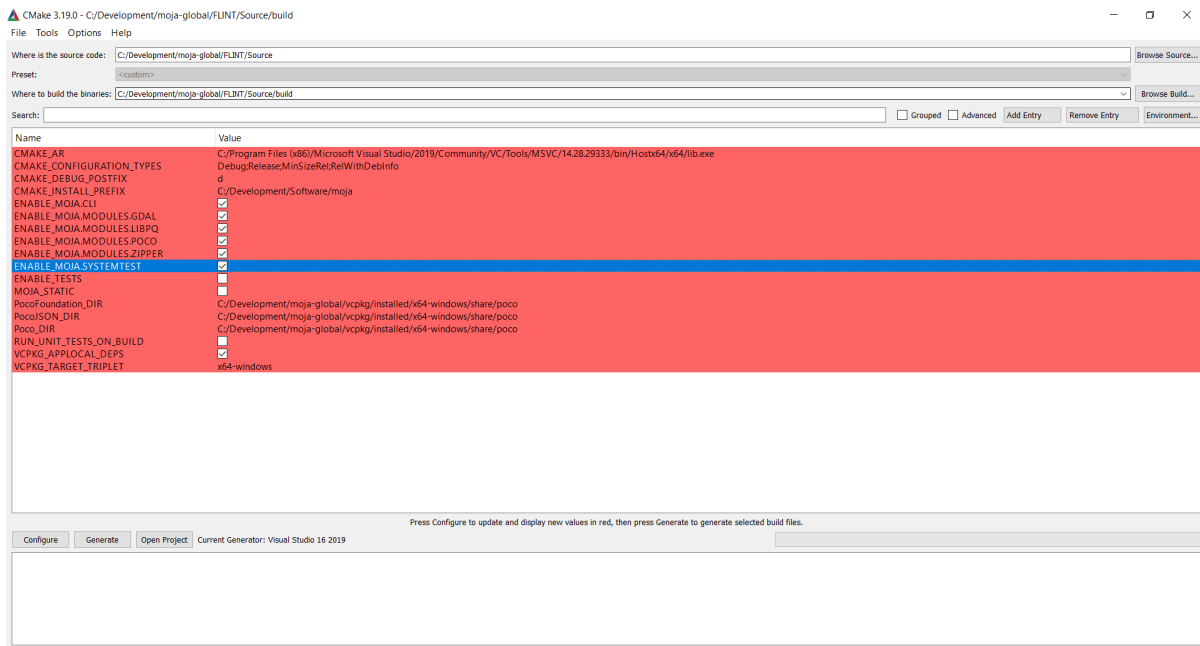


Fig. 13: Checking the flags mentioned

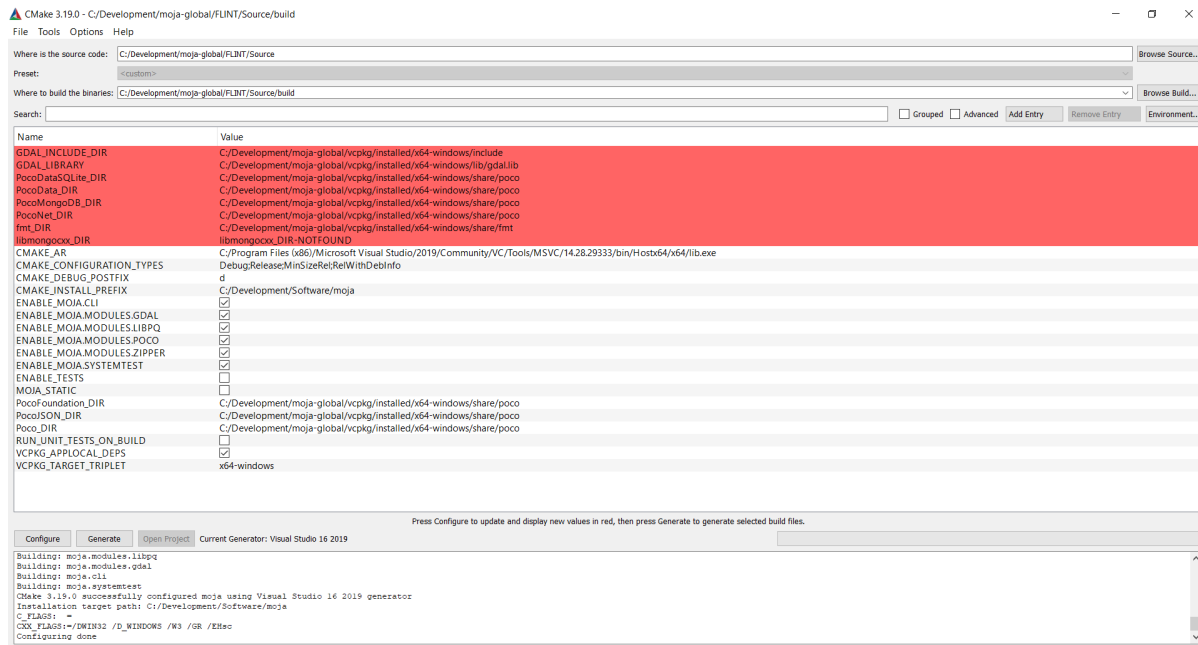


Fig. 14: Configuring the new options highlighted in red

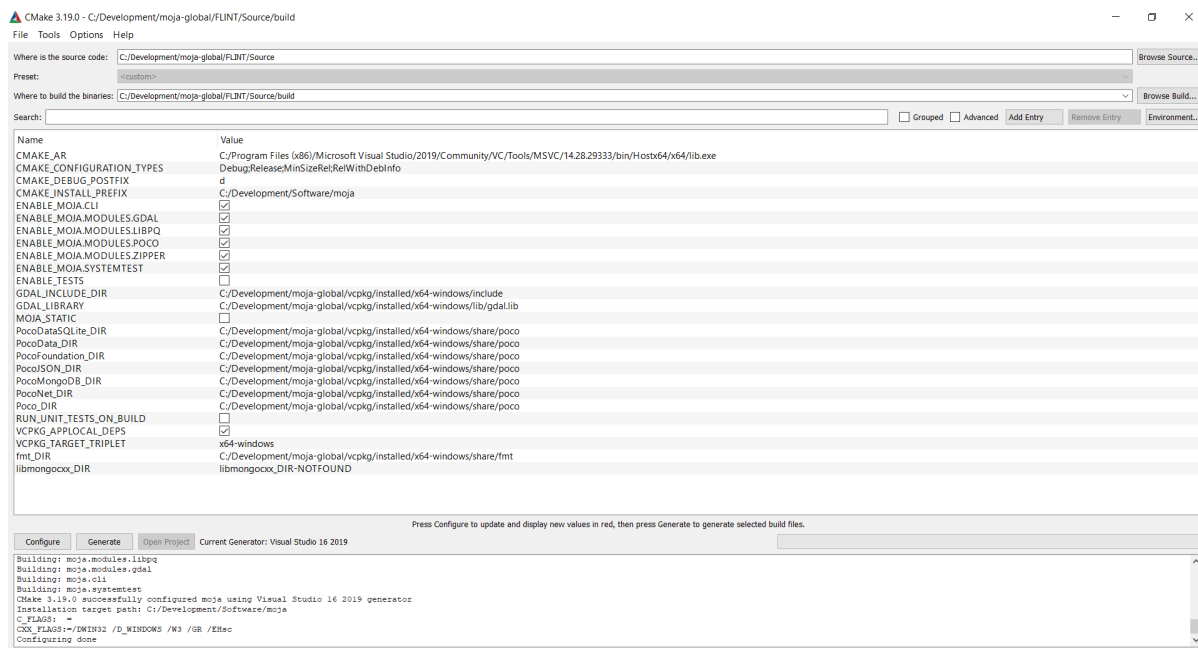
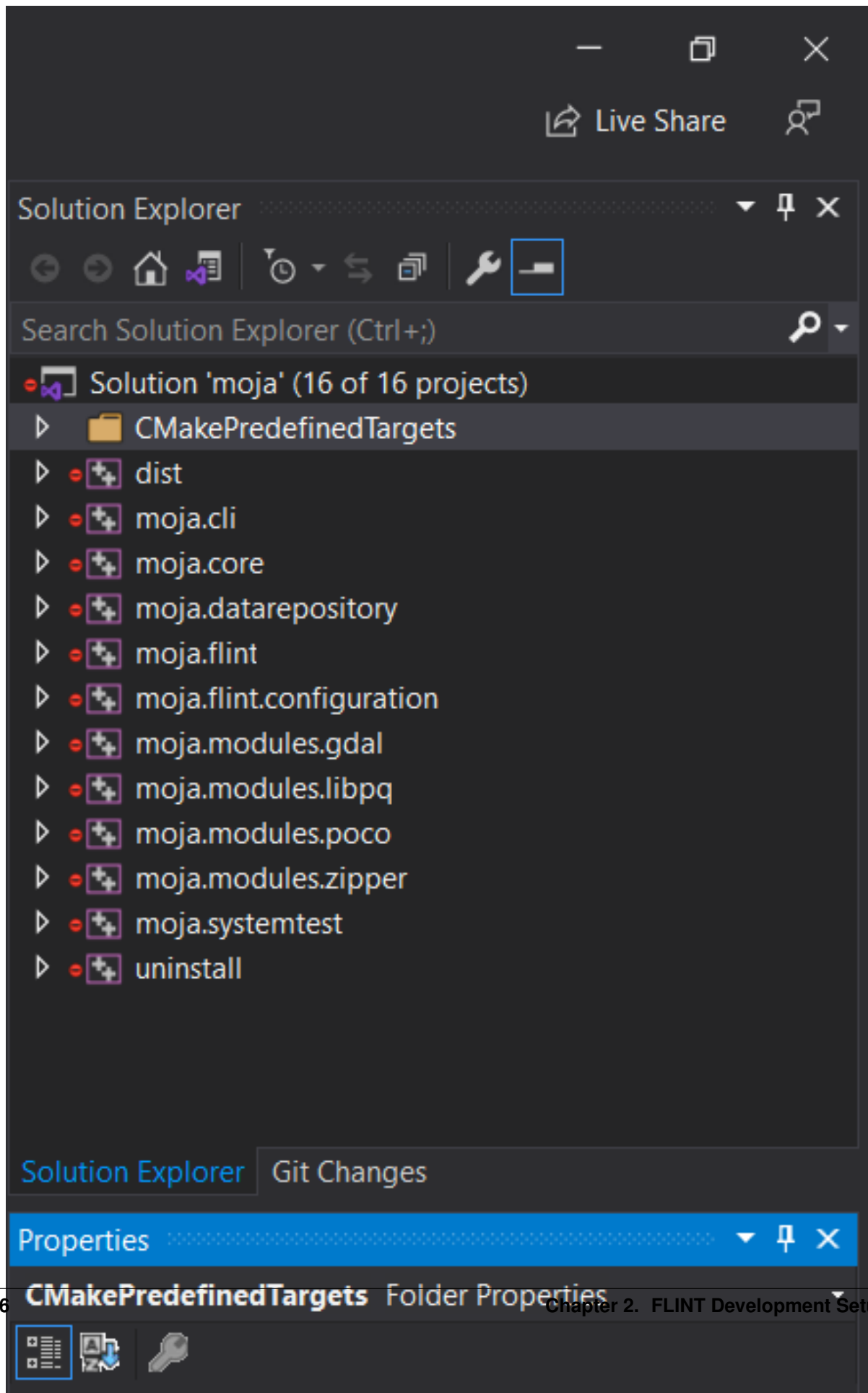


Fig. 15: Generating the new configuration



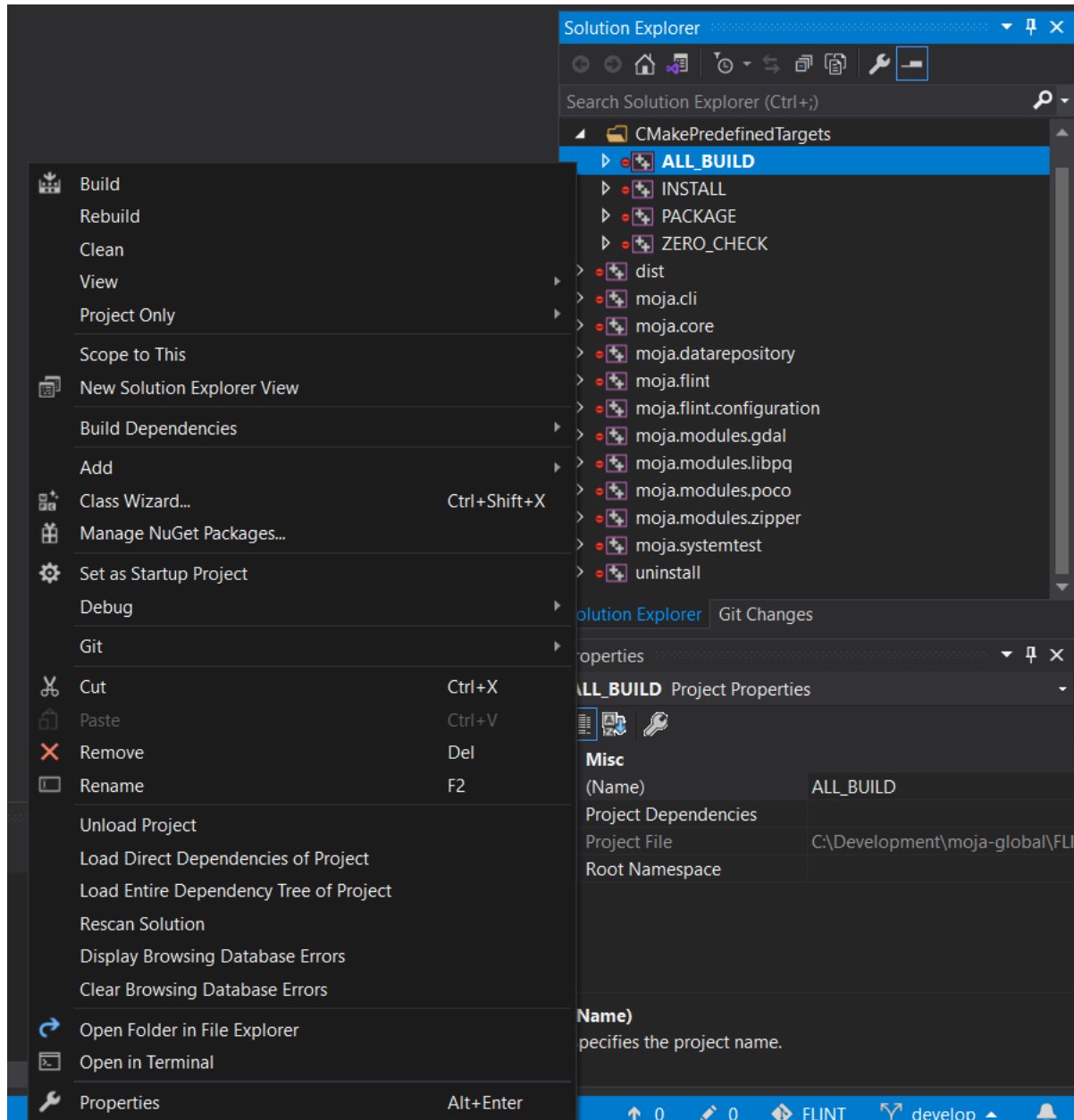


Fig. 17: CMakePredefinedTargets ALL_BUILD Screen

Others extensions may be required, please follow instructions during VS Code startup. Extensions required during development will be installed in the container (listed below).

Building the Project

With these extensions installed, on startup, VS Code should ask if you want to open the project in a Container - OR you can press F1 and select Remote-Containers: Open folder in Container...

The VS Code project has some `launch.json` settings in place (in the `.vscode` folder), these can run both the base and `rothc` samples. It is possible to debug into the `moja.flint` libraries by loading on of the `.cpp/.h` files and setting a breakpoint - OR stepping into a method using the debugger

To build the project the `cmake` and `C++` extensions will be required. These have been specified in the `devcontainer.json` file. To build the library use Cmake Configure, Build and Install.

```
"extensions": [  
    "ms-vscode.cpptools",  
    "austin.code-gnu-global",  
    "twxs.cmake",  
    "ms-vscode.cmake-tools"  
]
```

Once the project opens the folder in the dev container, use the `cmake` commands to configure and build the project. Once this is done you should be ready to run/debug one of the samples.

NOTE : The libraries require a slightly different paths to work inside the dev-container, so there is a new version of the library configs for VS Code. These commands will work from the terminal in the running container after `cmake` has been successful.

```
# start in the correct folder  
cd /workspaces/FLINT.example/Run_Env  
  
# sample  
moja.cli --config config/point_example.json --config config/libs.base.vscode.json --  
↪ logging_config logging.debug_on.conf  
  
# rothc  
moja.cli --config config/point_rothc_example.json --config config/libs.base_rothc.  
↪ vscode.json --logging_config logging.debug_on.conf  
  
# Chapman Richards - forest point  
moja.cli --config config/point_forest_config.json --config config/libs.gdal.chaprich.  
↪ vscode.json  
  
# Chapman Richards - forest spatial  
moja.cli --config config/forest_config.json --config config/libs.gdal.chaprich.vscode.  
↪ json --config_provider config/forest_provider.json
```

2.4.3 Environment: Docker

In the Docker environment option for setting up `FLINT.example`, only the run option is available. In case you want to develop or debug the repository code, please switch to the Visual Studio environments.

Prerequisites

- Docker

Building the docker

```
# from repository root folder
cd Docker
docker build --build-arg NUM_CPU=8 -t moja/flint.example:bionic .
```

- Return to top level folder with `cd ..`

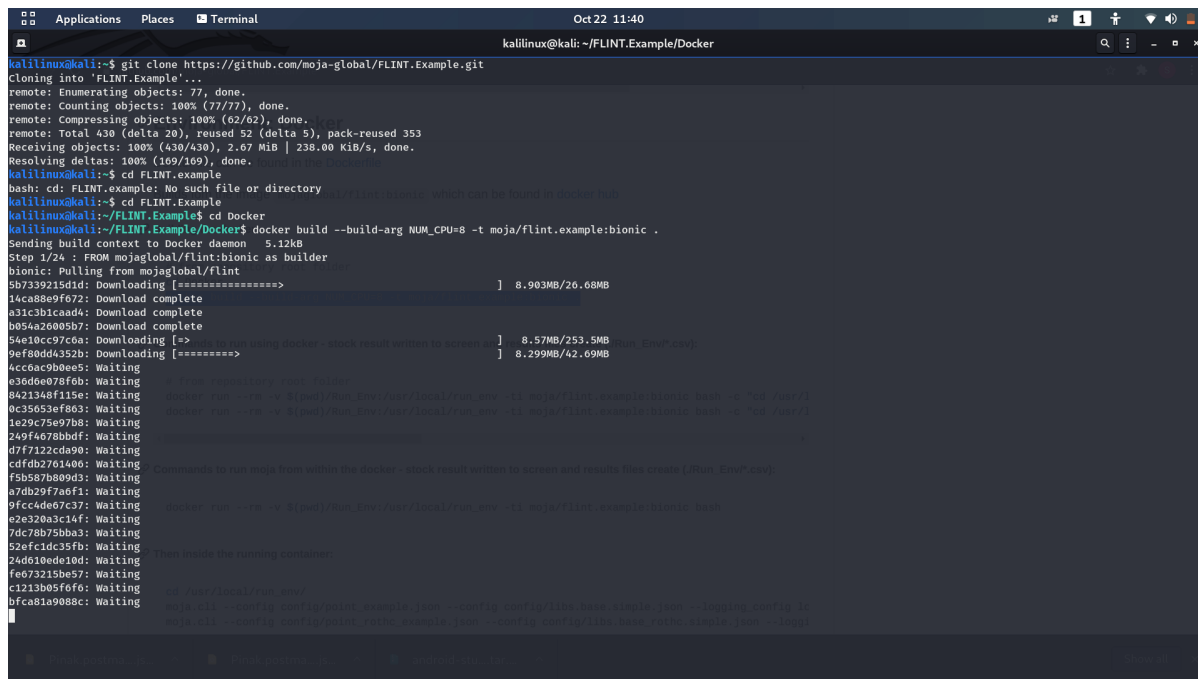


Fig. 18: Building the FLINT.example image using Docker

Commands to run using docker - stock result written to screen and results files create (./Run_Env/*_csv):

```
# from repository root folder

# For Linux
docker run --rm -v $(pwd)/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic_
↪ bash -c "cd /usr/local/run_env/; moja.cli --config config/point_example.json --
↪ config config/libs.base.simple.json --logging_config logging.debug_on.conf"

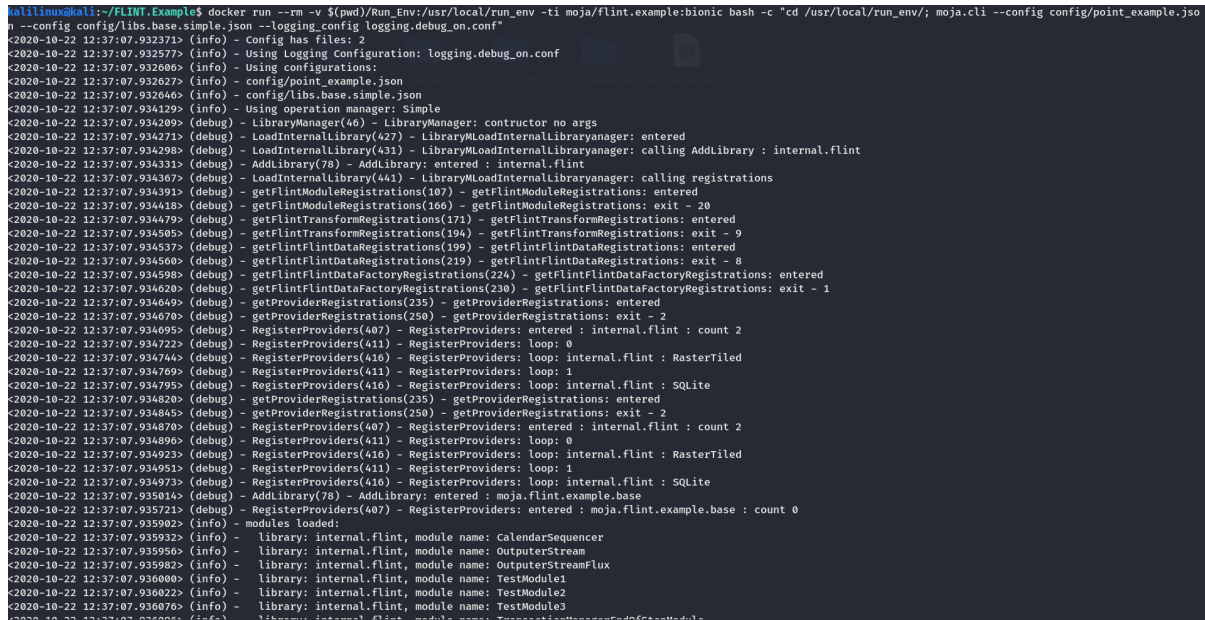
# For Windows
docker run --rm -v %cd%/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic bash_
↪ -c "cd /usr/local/run_env/; moja.cli --config config/point_example.json --config_
↪ config/libs.base.simple.json --logging_config logging.debug_on.conf"
```

For the RothC example, you may run this command:-

```
# For Linux
docker run --rm -v $(pwd)/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic_
↪ bash -c "cd /usr/local/run_env/; moja.cli --config config/point_rothc_example.json --
↪ config config/libs.base_rothc.simple.json --logging_config logging.debug_on.conf"
(continues on next page)
```

(continued from previous page)

```
# For Windows
docker run --rm -v %cd%/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic bash
↪-c "cd /usr/local/run_env/; moja.cli --config config/point_rothc_example.json --
↪config config/libs.base_rothc.simple.json --logging_config logging.debug_on.conf"
```



```
kalilinuxkali:~/FLINT.Example$ docker run --rm -v $(pwd)/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic bash -c "cd /usr/local/run_env/; moja.cli --config config/point_example.json --logging_config logging.debug_on.conf"
<2020-10-22 12:37:07.932371> (info) - Config has files: 2
<2020-10-22 12:37:07.932577> (info) - Using Logging Configuration: logging.debug_on.conf
<2020-10-22 12:37:07.932600> (info) - Using configurations:
<2020-10-22 12:37:07.932622> (info) - config/point_example.json
<2020-10-22 12:37:07.932646> (info) - config/libs.base.simple.json
<2020-10-22 12:37:07.934129> (info) - Using operation manager: Simple
<2020-10-22 12:37:07.934209> (debug) - LibraryManager(46) - LibraryManager: constructor no args
<2020-10-22 12:37:07.934271> (debug) - LoadInternalLibrary(427) - LibraryMLoadInternalLibraryanager: entered
<2020-10-22 12:37:07.934298> (debug) - LoadInternalLibrary(431) - LibraryMLoadInternalLibraryanager: calling AddLibrary : internal.flint
<2020-10-22 12:37:07.934331> (debug) - AddLibrary(78) - AddLibrary: entered : internal.flint
<2020-10-22 12:37:07.934367> (debug) - LoadInternalLibrary(442) - LibraryMLoadInternalLibraryanager: calling registrations
<2020-10-22 12:37:07.934391> (debug) - getFlintModuleRegistrations(107) - getFlintModuleRegistrations: entered
<2020-10-22 12:37:07.934418> (debug) - getFlintModuleRegistrations(166) - getFlintModuleRegistrations: exit - 20
<2020-10-22 12:37:07.934479> (debug) - getFlintTransformRegistrations(171) - getFlintTransformRegistrations: entered
<2020-10-22 12:37:07.934505> (debug) - getFlintTransformRegistrations(194) - getFlintTransformRegistrations: exit - 9
<2020-10-22 12:37:07.934537> (debug) - getFlintFlintDataRegistrations(199) - getFlintFlintDataRegistrations: entered
<2020-10-22 12:37:07.934560> (debug) - getFlintFlintDataRegistrations(219) - getFlintFlintDataRegistrations: exit - 8
<2020-10-22 12:37:07.934620> (debug) - getFlintFlintDataFactoryRegistrations(224) - getFlintFlintDataFactoryRegistrations: entered
<2020-10-22 12:37:07.934649> (debug) - getFlintFlintDataFactoryRegistrations(230) - getFlintFlintDataFactoryRegistrations: exit - 1
<2020-10-22 12:37:07.934670> (debug) - getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-22 12:37:07.934670> (debug) - getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-22 12:37:07.934695> (debug) - RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-22 12:37:07.934722> (debug) - RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-22 12:37:07.934744> (debug) - RegisterProviders(416) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-22 12:37:07.934769> (debug) - RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-22 12:37:07.934795> (debug) - RegisterProviders(416) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-22 12:37:07.934820> (debug) - getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-22 12:37:07.934845> (debug) - getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-22 12:37:07.934870> (debug) - RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-22 12:37:07.934896> (debug) - RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-22 12:37:07.934923> (debug) - RegisterProviders(416) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-22 12:37:07.934951> (debug) - RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-22 12:37:07.934973> (debug) - RegisterProviders(416) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-22 12:37:07.935016> (debug) - AddLibrary(78) - AddLibrary: entered : moja.flint.example.base
<2020-10-22 12:37:07.935721> (debug) - RegisterProviders(407) - RegisterProviders: entered : moja.flint.example.base : count 0
<2020-10-22 12:37:07.935902> (info) - modules loaded:
<2020-10-22 12:37:07.935932> (info) - library: internal.flint, module name: CalendarSequencer
<2020-10-22 12:37:07.935956> (info) - library: internal.flint, module name: OutputStream
<2020-10-22 12:37:07.935982> (info) - library: internal.flint, module name: OutputStreamFlux
<2020-10-22 12:37:07.936006> (info) - library: internal.flint, module name: TestModule1
<2020-10-22 12:37:07.936022> (info) - library: internal.flint, module name: TestModule2
<2020-10-22 12:37:07.936076> (info) - library: internal.flint, module name: TestModule3
<2020-10-22 12:37:07.936092> (info) - library: internal.flint, module name: TestModuleManagerAndDataModule1
```

Fig. 19: Running the examples using Docker

Commands to run moja from within the docker - stock result written to screen and results files create (./Run_Env/*.csv):

```
# For Linux
docker run --rm -v $(pwd)/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic
↪bash

# For Windows
docker run --rm -v %cd%/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic bash
```

Then inside the running container:

```
cd /usr/local/run_env/
moja.cli --config config/point_example.json --config config/libs.base.simple.json --
↪logging_config logging.debug_on.conf
moja.cli --config config/point_rothc_example.json --config config/libs.base_rothc.
↪simple.json --logging_config logging.debug_on.conf
```

Outputs

The runs above will create output files. While Stock values are output to the screen, there will also be some simple CSV files created with both Stock and Flux values for the simulation.

```

kalilinux@kali:~/FLINT.Example$ docker run --rm -v $(pwd)/Run_Env:/usr/local/run_env -ti moja/flint.example:bionic bash
root@c009e2c080ba:/# cd /usr/local/run_env/
root@c009e2c080ba:/usr/local/run_env# moja.cli --config config/point_example.json --config config/libs.base.simple.json --logging_config logging.debug_on.conf
<2020-10-22 12:39:40.383302> (info) - Config has files: 2
<2020-10-22 12:39:40.383595> (info) - Using Logging Configuration: logging.debug_on.conf
<2020-10-22 12:39:40.383632> (info) - Using configurations:
<2020-10-22 12:39:40.383657> (info) - config/point_example.json
<2020-10-22 12:39:40.383677> (info) - config/libs.base.simple.json
<2020-10-22 12:39:40.385225> (info) - Using operation manager: Simple
<2020-10-22 12:39:40.385309> (debug) - LibraryManager(46) - LibraryManager: constructor no args
<2020-10-22 12:39:40.385380> (debug) - LoadInternalLibrary(427) - LibraryLoadInternalLibraryanager: entered
<2020-10-22 12:39:40.385404> (debug) - LoadInternalLibrary(431) - LibraryLoadInternalLibraryanager: calling AddLibrary : internal.flint
<2020-10-22 12:39:40.385430> (debug) - AddLibrary(78) - AddLibrary: entered : internal.flint
<2020-10-22 12:39:40.385460> (debug) - LoadInternalLibrary(441) - LibraryLoadInternalLibraryanager: calling registrations
<2020-10-22 12:39:40.385495> (debug) - getFlintModuleRegistrations(107) - getFlintModuleRegistrations: entered
<2020-10-22 12:39:40.385520> (debug) - getFlintModuleRegistrations(166) - getFlintModuleRegistrations: exit - 20
<2020-10-22 12:39:40.385578> (debug) - getFlintTransformRegistrations(171) - getFlintTransformRegistrations: entered
<2020-10-22 12:39:40.385606> (debug) - getFlintTransformRegistrations(194) - getFlintTransformRegistrations: exit - 9
<2020-10-22 12:39:40.385641> (debug) - getFlintFlintDataRegistrations(199) - getFlintFlintDataRegistrations: entered
<2020-10-22 12:39:40.385665> (debug) - getFlintFlintDataRegistrations(219) - getFlintFlintDataRegistrations: exit - 8
<2020-10-22 12:39:40.385705> (debug) - getFlintFlintDataFactoryRegistrations(224) - getFlintFlintDataFactoryRegistrations: entered
<2020-10-22 12:39:40.385733> (debug) - getFlintFlintDataFactoryRegistrations(230) - getFlintFlintDataFactoryRegistrations: exit - 1
<2020-10-22 12:39:40.385763> (debug) - getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-22 12:39:40.385786> (debug) - getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-22 12:39:40.385811> (debug) - RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-22 12:39:40.385839> (debug) - RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-22 12:39:40.385867> (debug) - RegisterProviders(410) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-22 12:39:40.385890> (debug) - RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-22 12:39:40.385916> (debug) - RegisterProviders(410) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-22 12:39:40.385941> (debug) - getProviderRegistrations(235) - getProviderRegistrations: entered
<2020-10-22 12:39:40.385961> (debug) - getProviderRegistrations(250) - getProviderRegistrations: exit - 2
<2020-10-22 12:39:40.385986> (debug) - RegisterProviders(407) - RegisterProviders: entered : internal.flint : count 2
<2020-10-22 12:39:40.386013> (debug) - RegisterProviders(411) - RegisterProviders: loop: 0
<2020-10-22 12:39:40.386036> (debug) - RegisterProviders(410) - RegisterProviders: loop: internal.flint : RasterTiled
<2020-10-22 12:39:40.386062> (debug) - RegisterProviders(411) - RegisterProviders: loop: 1
<2020-10-22 12:39:40.386088> (debug) - RegisterProviders(410) - RegisterProviders: loop: internal.flint : SQLite
<2020-10-22 12:39:40.386128> (debug) - AddLibrary(78) - AddLibrary: entered : moja.flint.example.base
<2020-10-22 12:39:40.386763> (debug) - RegisterProviders(407) - RegisterProviders: entered : moja.flint.example.base : count 0
<2020-10-22 12:39:40.386926> (info) - modules loaded:
<2020-10-22 12:39:40.386958> (info) - library: internal.flint, module name: CalendarSequencer
<2020-10-22 12:39:40.386985> (info) - library: internal.flint, module name: OutputStream

```

Fig. 20: Running the moja.cli within Docker

```

Example_Point_Flux.csv
Example_Point_Stock.csv
Example_Rothc_Point_Flux.csv
Example_Rothc_Point_Stock.csv

```

The Output files created are visible in the below screenshot :-

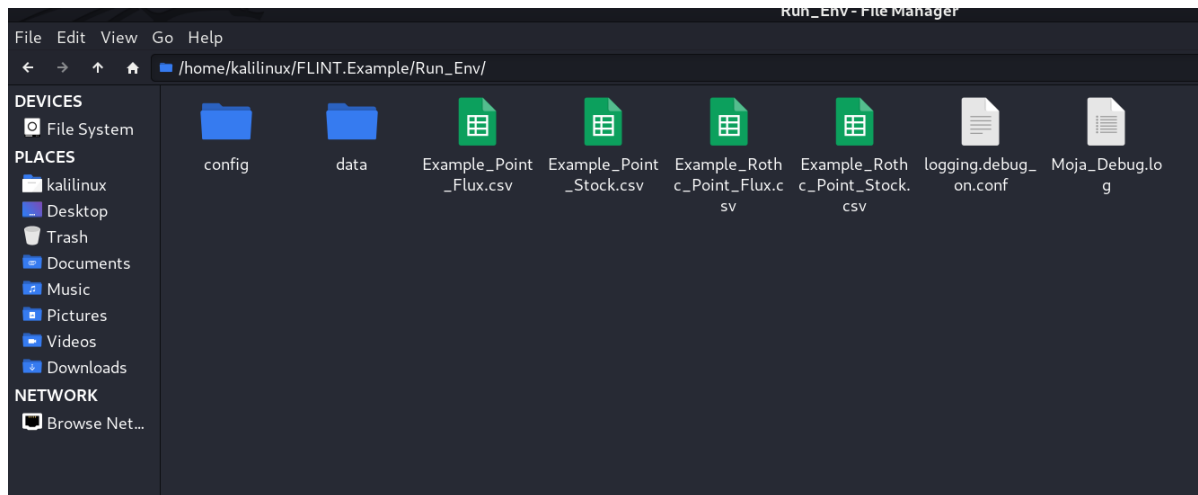


Fig. 21: Output files created from runs

GCBM Development Setup

This section guides first-time contributors through installing GCBM development environment on Windows.

The GCBM files required for setup are not currently publicly available and can be requested for installation by reaching out to us on info@moja.global.

The recommended method for installing the GCBM development environment is on Windows using Python 3.7 environment.

Contents:

3.1 GCBM Prerequisites

Before we take a leap into the process of development, please take a moment to verify if you have the necessary tools setup and skills to get started on this project. You should be familiar with the following:

3.1.1 Python 3.7

This document will guide you through the steps for installing the Python 3.7 environment required for running the pre- and post-processing tools for GCBM.

If you already have Python installed on your computer, follow the instructions in the section *Existing Python Installation*.

If Python is not already installed, follow the instructions in the section *New Python Installation*.

Existing Python Installation

- Locate your existing Python 3.7 installation (where `python.exe` is located). If you have both 32 and 64-bit versions installed (common with ArcGIS), find the path to the 64-bit version.
- From a command prompt in the `python_3_installer` directory, type: `install_modules_only <python path>`

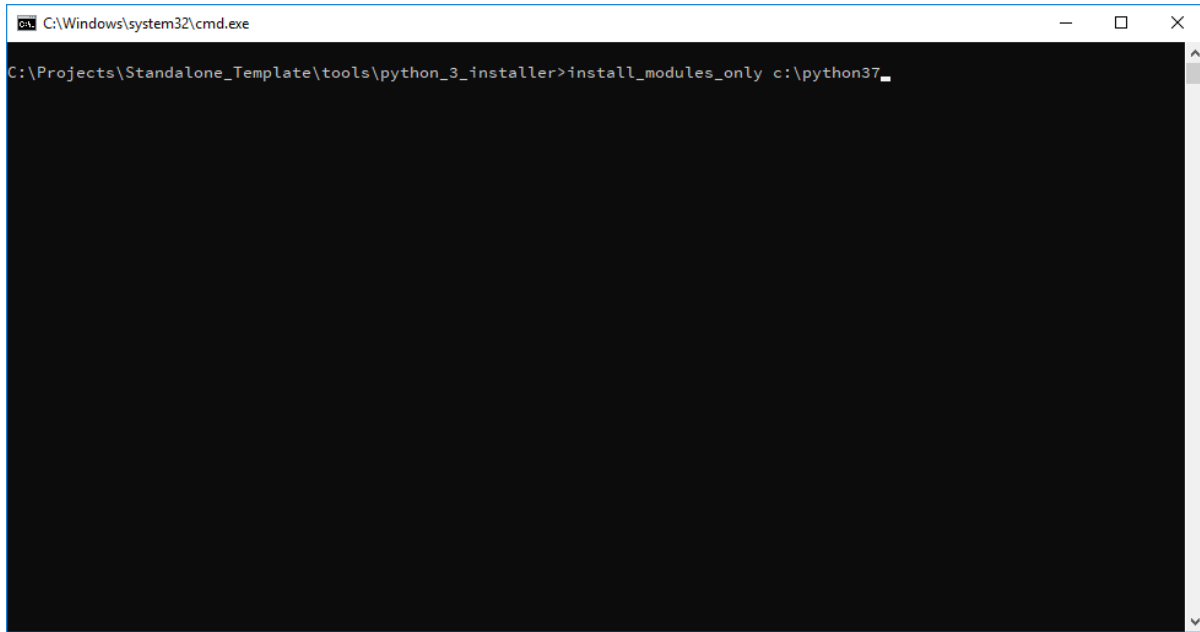


Fig. 1: Setting up modules for Existing Python Installation

New Python Installation

- Verify that you have no existing Python 3.7 installation – it is not usually possible to install the same version in two different locations.
- From a command prompt in the `python_3_installer` directory, type: `install_python [optional install path]`

3.1.2 Microsoft Access Database Driver

You may choose to skip this section if you have Microsoft Access installed.

If you do not have Microsoft Access installed, you will need to install a driver to connect to this type of database.

- Double-click `python_3_installer\installers\AccessDatabaseEngine_x64.exe`

3.1.3 Visual C++ Redistributable Packages

Install the C++ packages required to run GCBM and supporting tools:

- Double-click `tools\VC_redist\install_vc_redist.bat`

3.2 Windows Installation

This section guides first-time contributors through installing GCBM on Windows.

Before proceeding further, make sure you have the following prerequisites setup:

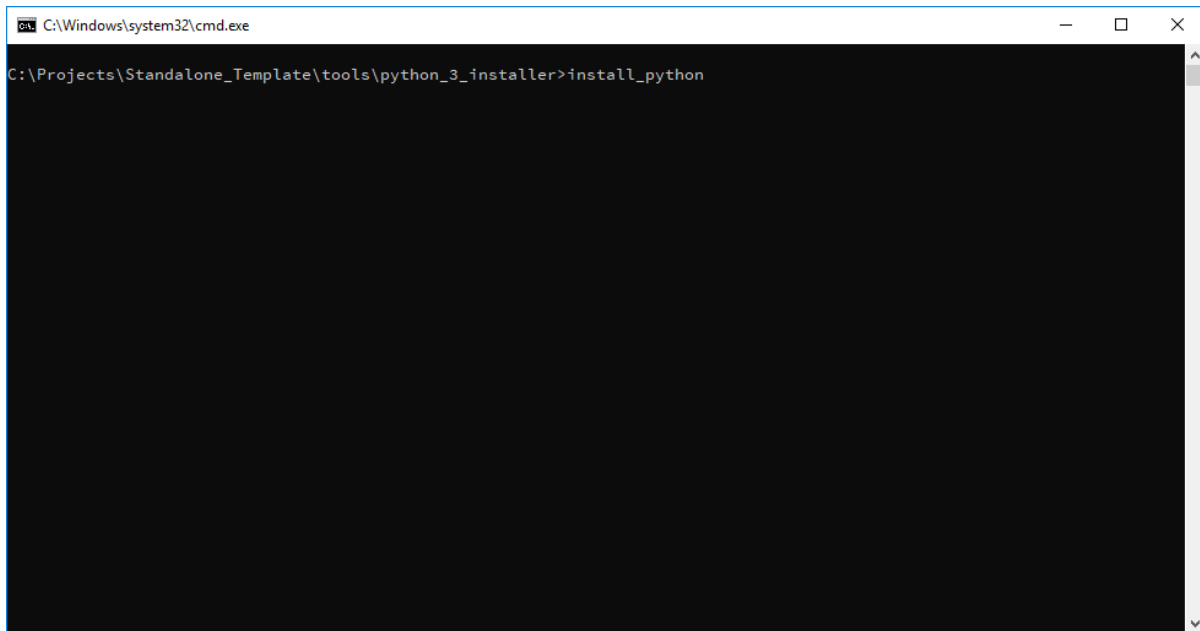


Fig. 2: Setting up Python and modules for New Python Installation

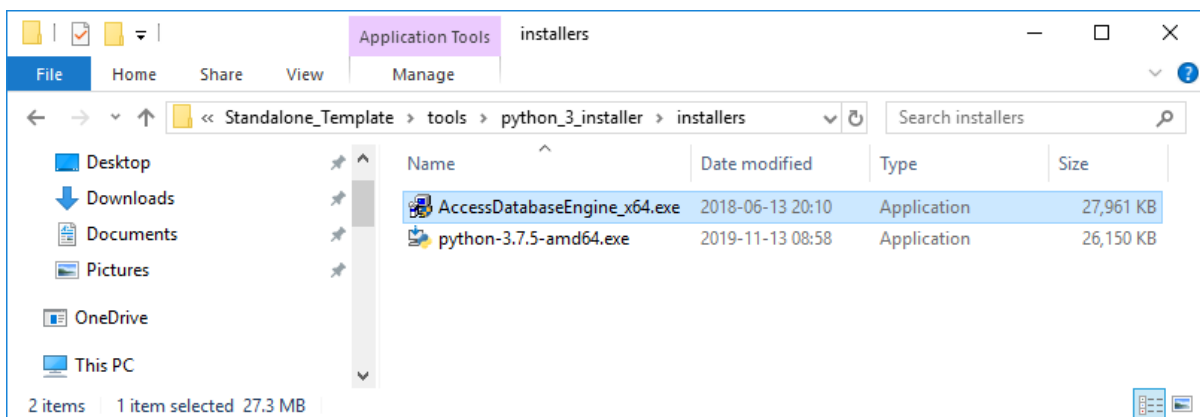


Fig. 3: Installing Microsoft Access Database Driver

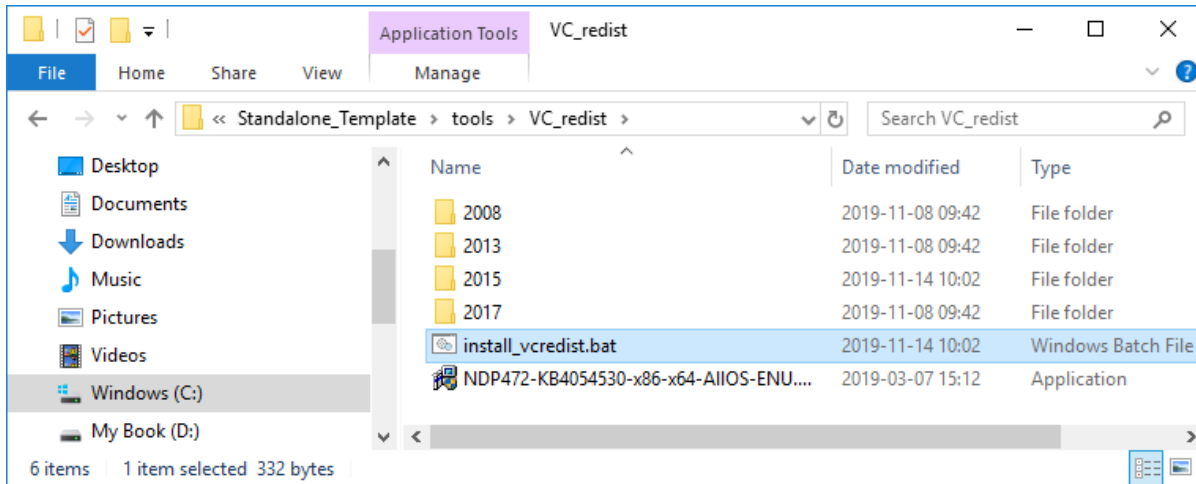


Fig. 4: Installing the C++ packages required to run GCBM and supporting tools

3.2.1 Prerequisites

- Python 3.7
- Microsoft Access Database Driver
- Visual C++ Redistributable Packages

Now that you have all the necessary prerequisites, you can proceed with the Installation.

3.2.2 Update GCBM Run Script

Edit `run_all.bat` and update the Python path to the one used in the Python installation step, and the platform bit-ness to match your version of MS Access if needed:

```

1 @echo off
2
3 REM ***** USER CONFIGURATION *****
4 REM Set simulation start and end years.
5 set SIMULATION_START_YEAR=2010
6 set SIMULATION_END_YEAR=2020
7
8 REM Set Python path - change this to your Python installation directory.
9 set GCBM_PYTHON=C:\Python37
10
11 REM Is your version of MS Access 32 (x86) or 64 (x64) bit?
12 set PLATFORM=x64
13 REM *****

```

Fig. 5: Editing `run_all.bat` file to update Python path and Platform

3.2.3 Test GCBM

Double-click the `run_gcbm.bat` file to run GCBM – if the installation steps were performed correctly, the preprocessing tools, GCBM model, and postprocessing tools should run without any error messages.

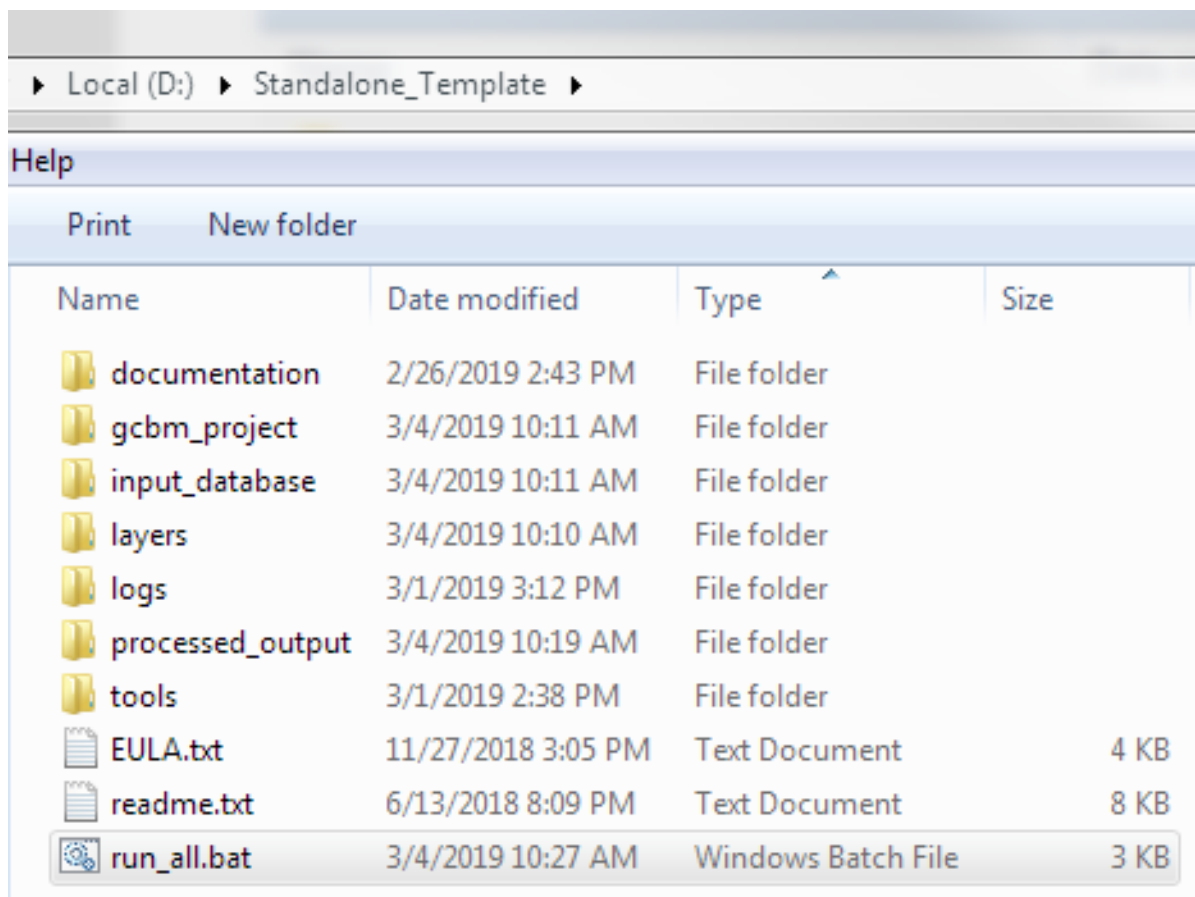


Fig. 6: Running the `run_gcbm.bat` file to execute GCBM

Viola! We are all done.

We're so glad that you're thinking about contributing to moja global. We welcome your contributions!

This guide is to help new contributors to know more about how they can contribute to moja global. It also covers some of the best code contribution practices adopted by moja global to ensure readability and maintainability of code.

Contents:

4.1 Before making a contribution

- Please checkout our [Git and Github guide](#) for detailed instructions on how to setup this project and make a pull request.
- Please follow our [Code of Conduct](#) at all times.
- Make sure your contribution follows the [moja global best contribution practices](#).

4.1.1 Join our communication Channels

If you are new to moja global, joining our communication channels can help you connect with community members and fellow contributors who can help answer your queries. In case you are facing any issues with the project setup or are looking for issues to contribute to, this would be a good place to start. Please check out our [Join us](#) section for more information on how to join our communication channels.

4.1.2 Contributors Website

If you are a first-timer then please checkout our [Contributor Website](#) . Here you can find out beginner friendly issues across all moja global repositories and even get featured as a new contributor! Along with checking out our Contributing workflow, you may also reach out to us through the contact form and share your interesting ideas with us!

4.2 Ways to contribute to moja global

There are plenty of ways you can contribute to moja global repositories. Apart from contributing code, moja global would love any kind of help for non-code related issues as well.

4.2.1 Review & Contribute Science Design

Most of our code is informed by an underlying Science Design. We develop these designs collaboratively and your contributions are most welcome!

Follow the following steps to contribute to or review a science design:

- Find the Science Design
 - Every moja global repository will have a folder called “Science” in the root directory
 - The Science folder will contain 2 types of Science Designs:
 - * PDF files contain completed science designs.
 - * .md files contain a link to science designs under development.
 - Locate the PDF file with the highest version number for review
 - Locate the .md file with the highest version number for a contribution
 - If no .md file is available, proceed with agree on a Science Design
- Review Science Design
 - Open the PDF file you want to review
 - Changes or suggestions are not possible
 - If you detect an inaccuracy or want to propose an improvement, check whether your change has already been made in the .md file with the highest version number
 - If the change has not been made yet, continue with contribute to a science design below.
- Contribute to a Science Design
 - Open the .md file with the highest version number.
 - Follow the link to the Google Doc with the ongoing science discussion
 - Contact the Document Owner to join the discussion or just leave a comment in the text.

4.2.2 Suggest UI/UX Improvements

One of the most important areas of improvement to our flagship software FLINT is the user interface. We really need your help with this!

If you have ideas on how we can improve, please share them with us by creating a [new issue](#). We could then start a new project for your idea!

4.2.3 Contribute Translations

Right now our interfaces do not support translations and we also don’t have a translation strategy in place. But we want to change this. We want our projects to be accessible to non-English speakers. If you have any ideas then please share them with us by creating a [new issue](#).

4.2.4 Coach or Train New Contributors

moja global has defined the following roles to help contributors to achieve their maximum potential while working on moja global repositories.

Coaches are experienced coders or scientists or users who are available to work shoulder to shoulder with new coders, users, or contributors

Ambassadors are experienced coders or scientists or users who are available to provide training to groups of new coders, users, or contributors

For adding yourself as a Coach or Ambassador, reach out to us over mail on info@moja.global. Please ensure you have a track record that proves you know the proposed repository for Mentorship, its purpose and contents well.

Please wait for a few days for us to get back to you. After adding yourself as a Coach or Ambassador, please complete the following steps on our GitHub.

- Ensure you have already [claimed credit](#) for your work in the past
- Add your name to Coaches/Ambassadors in the README document by copying your avatar and name from the `all-contributors` section and pasting it under the **Coaches/Ambassadors** header.
- Submit a Pull-Request with your proposed changes

4.2.5 Answer user Questions

Please help moja global be an active and responsive open source organization! Here is how you may answer user queries.

- Join moja global by emailing on info@moja.global. You will receive a slack workspace invite where you may join us for further discussions on the project of your interest.
- You can review and answer contributor questions on open issues in your repository/team repositories on our [GitHub organization](#).
- You may also review discussions and answer new users queries on moja global's Slack workspace
- If the reply could be useful for others, please add it as an entry under the FAQ section on the README of your repo or in this documentation repo FAQ section for general moja global questions.
- Don't forget to [Claim credit](#) for your contribution

4.2.6 Organize moja global events/meetups

moja global believes events are important to reach out to new users or improve collaboration. If you are planning to organize events to spread the message of moja global and invite contributors, here are the steps to follow:

- Please send an email to info@moja.global with a clear and descriptive subject.
- Please provide background or references to prove your good intentions if you do not have a track record on GitHub. You may also attach files, screenshots and animated GIFs to better illustrate your ideas.

Please allow us a few days to get back to you. Don't forget to let us know how the event was!

4.2.7 Are There Other Ways of Contributing?

Yes, there are a lot of other ways in which you can help us!

You can help us in administration, fundraising, website development, communication/outreach. You can also offer strategy advice. You can even join our strategy board!

If there's some other way, not listed above, in which you'd like to help, then please drop us a line at info@moja.global. We'll get in touch with you!

4.3 After making your first contribution

4.3.1 Get credit for your contribution

We use the All Contributors Bot to recognize contributors.

To get recognized, just add the following line to a comment after making your contribution (like submitting a pull request, replying to a question, resolving an issue, etc.)

```
@all-contributors please add <@username> for <contributions>
```

Replace <@username> with your GitHub username and <contributions> with any word from this [list](#).

If you wish to know more about the `all-contributors` bot usage in moja global, please checkout the [Bots and Integrations](#) section.

4.3.2 Report Bugs, Provide Feedback or Request Features

We welcome all kinds of bug reports, user feedback and feature requests!

If you find an easily reproducible bug and/or are experienced in reporting bugs, feel free to just open an issue on the relevant project on GitHub.

We've created some issue templates to assist you in this. Please use them to create a [new issue](#) in the relevant project's repository.

4.3.3 Pick more complex issues to work on

Now that you have your first contribution merged, you may move on to issues without `Good for newcomers` and `Help Wanted` labels. For specific areas of interest, please filter out the issues with specific labels of your interest.

4.3.4 Help other contributors take their first step

Now that you have made your first contribution, it is time to help other contributors to start their journey. There are a variety of ways in which you can help and guide them to their first contribution!

Reviewing contributions

moja global welcomes all contributors to review each others pull requests and suggest changes. You may also choose to review other contributions by following this guide [here](#).

Answering their queries

moja global urges all contributors and community members to help each other out with queries on Slack/GitHub. This can help new contributors in setting up moja global repositories and will drive the path to their first contribution. For more details on how to answer queries, please follow this [guide](#).

Mentor new contributors

If you know the contents of the repository well and would like to help contributors reach milestones in the development of the repository, then you may take up the role of a **Coach** or **Ambassador** to guide and work with new contributors. We believe this kind of healthy collaboration can nurture young coders and provide a platform to grow for both the parties. If you are interested, please check out this [guide](#) to know more about these roles.

Create beginner-friendly Issues

You may also create beginner friendly issues for new contributors to claim. Please ensure that the difficulty level of these issues are easy and if possible can be implemented without the need for setting up the Project completely. Since these issues are targeted at beginners, please share all necessary details required to solve the issue (eg. File-names that require change) in the issue description and label the issues with one or more of the following labels `good-first-issue`, `Good for newcomers` or `Help Wanted`.

4.4 Code Contribution Best Practices

This guide is to inform new contributors about the best practices followed by moja global. Before making any contribution please go through these guidelines to ensure your contribution can be merged with minimal changes.

4.4.1 Commit Guidelines

- Make sure your commit is passing all the tests. If your commit is failing tests then please try to fix the same commit instead of making a new commit by using `git --amend`
- Every commit should be directed to solve a single problem instead of trying to solve multiple issues at once. This encourages simplicity and also makes it easier for the reviewers to review.
- If the code introduced in the commit decreases the coverage/requires tests, don't forget to add them too.

Although it's encouraged to make commits following these guidelines but incase you missed, you can always fix your history using `git rebase -i`.

4.4.2 Commit Message Guidelines

- Keep the commit message short but concise explaining your changes and the problem you are trying to solve.
- Try to write the commit messages in an imperative tone for example:- 'Fix', 'Update', 'Add' instead of 'Fixed', 'Updated', 'Added'.
- Reference the issue solved in the commit message, for example:- `Fixes #8293: Add login api unit tests`. This will automatically close the issue referenced when your pull request gets merged.

4.4.3 Developer Certificate of Origin

moja global follows [Developer Certificate of Origin\(DCO\)](#) as a method to certify that the contribution you have submitted was created in whole or in part by you and you have the right to submit it under the open source license to moja global.

To apply this, please sign off all your commit messages with a line like this:

```
Signed-off-by: Random J Developer <random@developer.example.org>
```

Alternatively, you may also add the `--signoff` flag to the `git commit` command that will automatically add this line to your commit message.

4.4.4 Code Style & Conventions

This section will focus on the style guidelines and conventions used by moja global across its repositories.

Depending on the language of the repository you are working on, we have an array of tools and checks to ensure that the Code Style is not violated to promote further maintainability of code.

We encourage you to add the mentioned tools as plugins in your editor.

- **C++ Repositories:**

moja global follows the C++ style guide developed by Google for their open-source projects. The google style guide is aimed at enable coders to utilise the power of C++ while at the same time managing the potential complexity that can arise when coding in C++.

The style guide can be found at: <https://google.github.io/styleguide/cppguide.html>

It is possible that exceptions to the google style guide may be specified, in which case they will be listed here. Currently there are no exceptions.

- Coding enforcement: We know that the style guides are long and detailed and not always easy to adhere to. As such, the intention is to use [Clang-Tidy](#) as a tool to check and correct code formatting as determined by the Google C++ style guide. This will be implemented as an automated check through the Continuous Integration system.

- **Python Repositories:**

moja global follows the Python Style guide PEP8 that provides coding conventions for Python code. It is fairly common for Python code to follow this style guide. It's a great place to start since it's already well-defined.

This guide can be found at : <https://www.python.org/dev/peps/pep-0008/>

- Coding enforcement: In order to enforce the PEP8 conventions along with error detection, [Pylint](#) as a tool can be used and can be integrated with your editor as well. This will be implemented as an automated check through the Continuous Integration system.

4.5 Code Of Conduct

moja global governs its participants according to the Contributor Covenant Code of Conduct. As a contributor, you agree to uphold this code. Please report unacceptable behavior to info@moja.global. If you want your report to be handled confidentially, please report to guy@moja.global.

4.5.1 Contributor Covenant Code of Conduct

4.5.2 Our Pledge

In the interest of fostering an open and welcoming environment, we as contributors and maintainers pledge to making participation in our project and our community a harassment-free experience for everyone, regardless of age, body size, disability, ethnicity, sex characteristics, gender identity and expression, level of experience, education, socio-economic status, nationality, personal appearance, race, religion, or sexual identity and orientation.

4.5.3 Our Standards

Examples of behavior that contributes to creating a positive environment include:

- Using welcoming and inclusive language
- Being respectful of differing viewpoints and experiences
- Gracefully accepting constructive criticism
- Focusing on what is best for the community
- Showing empathy towards other community members

Examples of unacceptable behavior by participants include:

- The use of sexualized language or imagery and unwelcome sexual attention or advances
- Trolling, insulting/derogatory comments, and personal or political attacks
- Public or private harassment
- Publishing others' private information, such as a physical or electronic address, without explicit permission
- Other conduct which could reasonably be considered inappropriate in a professional setting

4.5.4 Our Responsibilities

Project maintainers are responsible for clarifying the standards of acceptable behavior and are expected to take appropriate and fair corrective action in response to any instances of unacceptable behavior.

Project maintainers have the right and responsibility to remove, edit, or reject comments, commits, code, wiki edits, issues, and other contributions that are not aligned to this Code of Conduct, or to ban temporarily or permanently any contributor for other behaviors that they deem inappropriate, threatening, offensive, or harmful.

4.5.5 Scope

This Code of Conduct applies both within project spaces and in public spaces when an individual is representing the project or its community. Examples of representing a project or community include using an official project e-mail address, posting via an official social media account, or acting as an appointed representative at an online or offline event. Representation of a project may be further defined and clarified by project maintainers.

4.5.6 Enforcement

Instances of abusive, harassing, or otherwise unacceptable behavior may be reported by contacting the project team at guy@moja.global. All complaints will be reviewed and investigated and will result in a response that is deemed

necessary and appropriate to the circumstances. The project team is obligated to maintain confidentiality with regard to the reporter of an incident. Further details of specific enforcement policies may be posted separately.

Project maintainers who do not follow or enforce the Code of Conduct in good faith may face temporary or permanent repercussions as determined by other members of the project's leadership.

4.5.7 Attribution

This Code of Conduct is adapted from the Contributor Covenant, version 1.4, available at <https://www.contributor-covenant.org/version/1/4/code-of-conduct.html>

For answers to common questions about this code of conduct, see <https://www.contributor-covenant.org/faq>

moja global as an Open source organisation wants to provide their users the best possible collaborative experience while meeting their security requirements and limiting their maintenance effort. Working on a collaborative platform like GitHub at times can be overwhelming, especially as the number of users and repositories grow within an organization.

This had led to the adoption of some GitHub workflow practices to ensure the smooth working of moja global. These practices include repository maintenance practices, automated checks for pull requests as well more advanced testing methodologies and bots to ease the workflow.

Contents:

5.1 GitHub Repository maintenance

This section guides contributors and maintainers on the guidelines to follow while setting up a new repository and maintaining it under moja global.

5.1.1 Repository Creation

New repository under moja global is generated from the template repository [Import-me](#). This ensures that the start-up files for a standard moja global repo is already included in the new repository and the commit history also remains clean.

Follow these steps to generate your new repository from Template repository:

- Navigate to <https://github.com/moja-global>.
- Select the **New** button for new repository creation.
- On the create repository page, select the template `Import-me` from the dropdown titled `Repository Template`.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Repository template

Start your repository with a template repository's contents.

No template ▾

✓ No template

moja-global/Import-Me

moja-global ▾ /

Great repository names are short and memorable. Need inspiration? How about [literate-octo-winner](#)?

Description (optional)

- Upon selecting the template, please make sure the checkbox for `include all branches` is unmarked.

Repository template

Start your repository with a template repository's contents.

moja-global/Import-Me ▾

☐ Include all branches

Copy all branches from moja-global/Import-Me and not just the default branch.

Owner *

moja-global ▾



Repository name *

new-repository ✓

Great repository names are short and memorable. Need inspiration? How about [literate-octo-winner](#)?

Description (optional)

new repository generated from template example

- ☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Create repository

- Now go ahead and add details of your repository name,description and visibility setting (Public or Private).
- Click on the `Create repository` button as the final step!

5.1.2 Setting up Labels for your repository

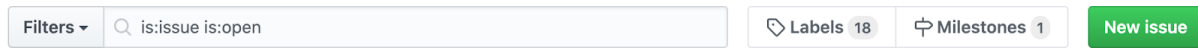
Labels are visual tools for the project. They make it easier to filter issues and prioritize tasks. Additionally, they also help new contributors identify areas of interest for your project.

They can help improve awareness of different types of contribution methods in the project. (e.g. science, communication and documentation tasks).

- Navigate to `https://github.com/moja-global/<repository_name>/labels` where `<repository_name>` is the name of the new repository created. You can find all the labels setup for your repository here. In order to create a new label, click on the New Label button.



- Configure each issues' labels in a way that makes sense for your project. The labels should classify the issues/pull requests in an appropriate manner so they can be easily applied for filtering later. Every issue and pull request labels can be found under the Issue tab with the labels button.



moja global has the following labels available on an organisational level. You can use the labels depending on the issue, and make new labels specific to the repository, if needed as explained above.

It is recommended to provide every issue with 4 types of labels: 1 from each type of label below.

Category:

- Cat = Blocked = Progress on the issue is Blocked, either due to waiting for another code change, or not in control.
- Cat = Bug = Something isn't working
- Cat = Comms = Propose a way to better communicate a feature
- Cat = Doc = Adding or updating documentation
- Cat = Good for newcomers = These issues require minimal context and are well-suited for new contributors
- Cat = Feedback = Describe how we can improve your experience
- Cat = Help Wanted = Anybody out there, can you give me a hand?
- Cat = Need Info
- Cat = New Feature = Suggest an idea for this project
- Cat = Science = Suggest how the science can be improved

Priority:

- Priority = High = High Priority issues/pull request that require immediate attention
- Priority = Low = Low Priority issues/pull request that require attention only after Mid Priority issues are resolved.
- Priority = Medium = Mid Priority issues/pull request that require attention after High Priority issues are resolved.

Time:

- T = 1 Hour = Resolving this issue will take about 1 hour
- T = 2 Hours = Resolving this issue will take about 2 hours

- `T = 4 Hours` = Resolving this issue will take about 1/2 day
- `T = 8 Hours` = Resolving this issue will take about 1 day
- `T = Break me up` = This issue takes more than 1 day and needs to be broken up into smaller tasks

Difficulty:

- `X = Easy` = This is a good issue for new contributors
- `X = Intermediate` = Solving this issue requires some experience
- `X = VeryDifficult` = Solving this issue requires advanced expertise

5.1.3 Creating and maintaining Project Boards

Project boards on GitHub help you organize and prioritize your work by creating them for specific feature work, comprehensive roadmaps, or even release checklists.

There are 2 types of project boards available:

- **Repository:** Boards for use in a single repository.
- **Organization:** Boards for use in a GitHub organization across multiple repositories (but private to organization members)

Moja global team uses boards for development and documentation at the repository level. It means repository-specific boards for focused work in each repository.

Creating your first board

- Project boards can be found under the `Projects` tab in the same row as `Issues` and `Pull requests` on a specific repository.
- If you have enough permissions on the repository or as an organisation member, then you'll be able to create a new project by clicking on the green button labeled `Create Project`.



Organize your issues with project boards

[Learn More](#)[Create a project](#)

Did you know you can manage projects in the same place you keep your code? Set up a project board on GitHub to streamline and automate your workflow.

- Configure the name and description for the project board. You can also choose templates to set up basic columns and sorting for your board. Currently, moja global team selects `Basic kanban` for Kanban-style boards.

Create a new project
Coordinate, track, and update your work in one place, so projects stay transparent and on schedule.

Project board name
sample board

Description (optional)
A sample project board

Project template
Save yourself time with a pre-configured project board template.
Template: Basic kanban

Create project

- After creating the project board, you may make adjustments to it as needed. You can create new columns, set up automation and add pre-existing GitHub issues and pull requests to the project board.

However, it is recommended that the contributors use the existing board in each repo rather than creating a new board unless necessary.

Adding issues/pull requests to your Project Board

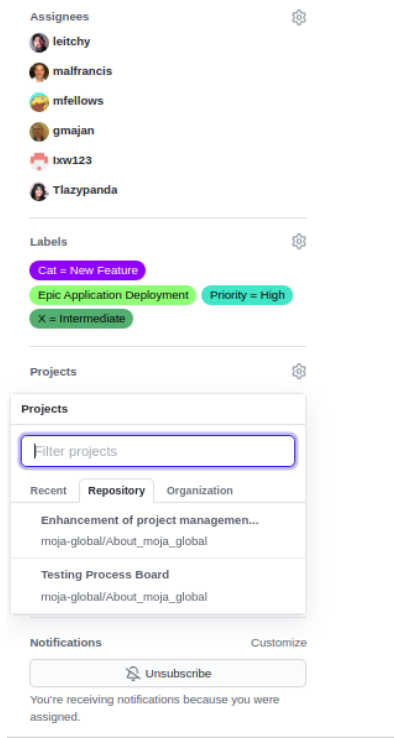
After you set up a project board, you need to populate it with issues and pull requests and keep updating the board on a regular basis.

The `Basic kanban` template offers the following columns for every issue/pull request. You may classify your issues/pull requests into one of according to the criteria below.

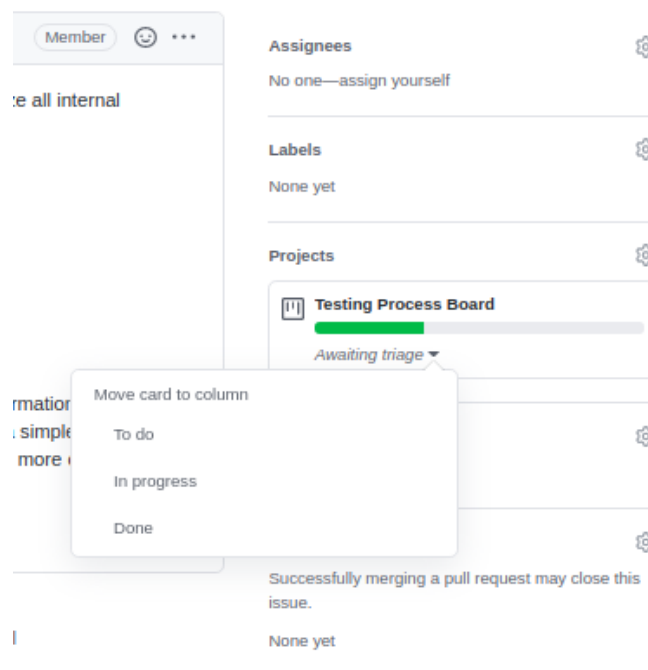
- **Todo:** Open issues/pull requests, Reopened issues/pull requests
- **In progress:** Issues that have been assigned, pull requests that are open and reviewed
- **Done:** Closed issues/pull requests, Merged pull requests

In order to add a issue/pull request to a project board, follow these steps:

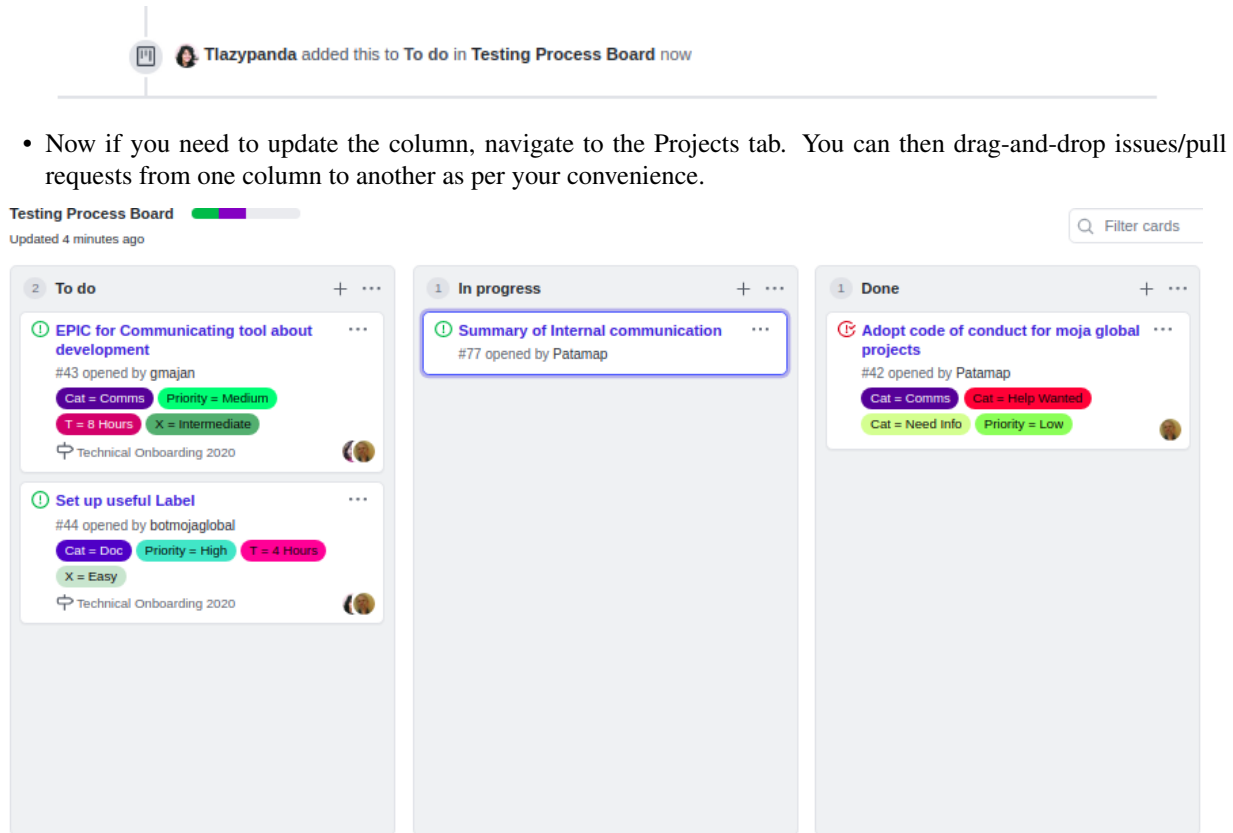
- Navigate to the specific issue/pull request.
- Under the right sidebar, check out the `Projects` tab. Click on the settings icon next to the `Projects` tab and select the relevant project board. If the settings icon is not visible to you, then you don't have enough permissions to add an issue/pull request to a project board.



- After selection of the board, you may classify it into one of the three columns by clicking on the **Awaiting Triage** dropdown and selecting any one of the above options (Todo, In progress, Done).



- Once you have classified your issue/pull request into the correct column, you can see an update for the same shown in your issue.

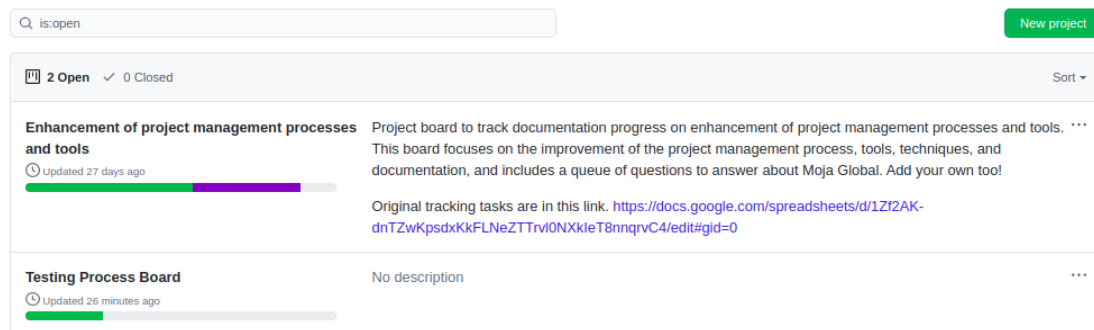


Build project boards into your workflow

After you set up a project board and populate it with issues and pull requests, you need to integrate it into your workflow. Project boards are effective only when actively used.

The moja global team uses the project boards as a way to track our progress as a team, update external stakeholders on development, and estimate team bandwidth for reaching our milestones.

The following image shows how we can track progress with GitHub project boards.



As moja global is an open-source project and community, consider using the project boards to update other team members, and encourage participation inside of GitHub issues and pull requests.

In the meanwhile, we also consider using the project boards for development. It also helps remind us and other core contributors to spend 5 minutes each day updating progress as needed.

5.2 Bots and Integrations

moja global makes use of an array of bots and integrations with GitHub in order to provide a smooth experience to new contributors and to also make the process of contributing as effortless as possible.

This section focuses on the bots used within the organization and how to maintain them.

5.2.1 All-contributors Bot

All-contributors bot is a [GitHub bot](#) to automate acknowledging contributors to your open source projects. This is achieved by setting up a contributors table in the readme which is edited according to the commands given to our bot. This bot is created and maintained by the *all-contributors* organisation [here](#).

Usage

- **Add a contributor** : Comment on Issue or Pull Request, asking @all-contributors to add a contributor:

```
@all-contributors please add <username> for <contributions>

* <contribution> : See the `Emoji Key (Contribution Types Reference)` <https://
↳allcontributors.org/docs/en/emoji-key>`_ for a list of valid contribution types.
```

- **Intent-Based Bot** : Your request to the bot doesn't need to be perfect. The bot will use basic Natural Language Parsing to determine your intent. For example, this will work too:

```
Jane you are crushing it in documentation and your infrastructure work has been great.
↳too. Let's
add jane.doe23 for her contributions. cc @all-contributors``
```

- The bot will then create a Pull Request to add the contributor, then reply with the pull request details.

Contributing

If you wish to configure or modify the bot settings according to the needs of the repository, you may update the `.all-contributorsrc` JSON file. The data used to generate the contributors list will be stored in there, and you can configure how you want @all-contributors to generate the list. Here are more details on the [configuration options](#) available.

5.2.2 Botmojaglobal

Botmojaglobal uses @zulipbot, a GitHub workflow bot from the zulip organisation, to handle issues and pull requests in our repositories in order to create a better workflow for contributors. This bot is created and maintained by the **zulip** organisation [here](#).

Usage

- **Claim an issue** : Comment `@botmojaglobal claim` on the issue you want to claim; **@botmojaglobal** will assign you to the issue and label the issue as `in progress`.
 - If you're a new contributor, **@botmojaglobal** will give you read-only collaborator access to the repository and leave a welcome message on the issue you claimed.
 - You can also claim an issue that you've opened by including `@botmojaglobal claim` in the body of your issue.
 - If you accidentally claim an issue you didn't want to claim, comment `@botmojaglobal abandon` to abandon an issue.
- **Label your issues** : Add appropriate labels to issues that you opened by including **@botmojaglobal add** in an issue comment or the body of your issue followed by the desired labels enclosed within double quotes ("").
 - For example, to add the bug and help wanted labels to your issue, comment or include `@botmojaglobal add "bug" "help wanted"` in the issue body.
 - You'll receive an error message if you try to add any labels to your issue that don't exist in your repository.
 - If you accidentally added the wrong labels, you can remove them by commenting `@botmojaglobal remove` followed by the desired labels enclosed with double quotes ("").
- **Find unclaimed issues** : Use the [GitHub search feature](#) to find unclaimed issues by adding one of the following filters to your search:
 - `-label: in progress` (excludes issues labeled with the `in progress` label)
 - `no:assignee` (shows issues without assignees)

Issues labeled with the `in progress` label and/or assigned to other users have already been claimed.

- **Track inactive claimed issues** : If a claimed issue has not been updated for a week, **@botmojaglobal** will post a comment on the inactive issue to ask the assignee(s) if they are still working on the issue.
 - If you see this comment on an issue you claimed, you should post a comment on the issue to notify **@botmojaglobal** that you're still working on it.
 - If **@botmojaglobal** does not receive a response from the assignee within 3 days of an inactive issue prompt, **@botmojaglobal** will automatically remove the issue's current assignee(s) and the `in progress` label to allow others to work on an inactive issue.

Contributing

If you wish to help develop and contribute to **@botmojaglobal**, check out the [mojaglobal/zulipbot](#) repository fork on GitHub and read the project's contributing guidelines for more information.

5.2.3 Welcome Bot

Welcome Bot is a [github app](#) that welcomes new users based off maintainer defined comments that should be located in a `.github/config.yml`. This app is configured for moja global in order to provide a richer experience to all new contributors. This bot is created and maintained by the *probot* organisation [here](#).

Usage

- **Welcome Bot is activated on all repositories** : Since the bot config file is already present in `.github` folder of `import-me` template repository, creating a new repository from this template ensures that the welcome bot is activated on the new repository.
- **Features** : The bot provides messages for 3 different scenarios and the messages are fetched from `.github/config.yml` file in each repository:
 - `newIssueWelcomeComment` : This message is displayed whenever a new contributor opens their first issue in the repository.
 - `newPRWelcomeComment` : This message is displayed whenever a new contributor open their first pull request in the repository.
 - `firstPRMergeComment` : This message is displayed whenever a new contributor's first pull request gets merged in the repository.

You can opt out of having the bot comment on first time pull requests, pull request merges, or new issues by not filling in a value for each of the above respective fields.

Contributing

If you wish to modify the messages displayed by Welcome Bot, you may modify the `.github/config.yml` file in the repository.

5.3 Automated Checks for pull requests

In order to maintain the code quality and coverage of our repositories, moja global deploys a series of tools. These tools include our Continuous Integration Setup that runs a complete test suite, Automated Code quality checks as well as Coverage tracking tools.

This section focuses on how these automated tests are setup and how may they be configured on our GitHub repositories. Currently only the following checks are available in our [FLINT.data_processing](#) repository.

5.3.1 Continuous Integration

moja global uses [GitHub Actions](#) for Continuous Integration. GitHub Actions creates an environment based on the Operating System of your choice (Linux in our case) and runs our test suite. This Continuous Integration script is triggered by every pull request and only passes when all the tests run successfully. This script also uploads our coverage report to [Codecov](#) for tracking the coverage compared to our base coverage percentage.

- In order to view the pull request build, please click on the **Details** link of the `Python application / build (push)` tab.
- In the case where tests fail, we can debug the problem from going through the console output as displayed here.
- After analysis of the test failure, you may then try to debug the test locally as well by running the command:-

```
python -m unittest discover tests -v
```

- Sometimes tests may also fail if your pull request is not rebased to the latest master. So it is recommended to take a rebase before creating the pull request.
- If you are still facing issues with the test failure, please reach out to the maintainers of the repository.

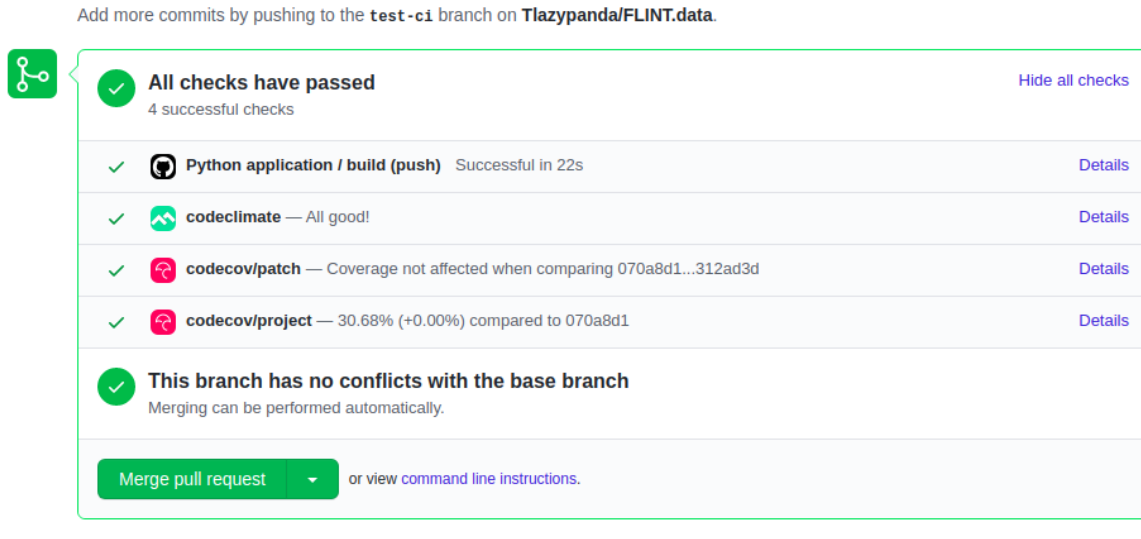


Fig. 1: GitHub pull request Checks

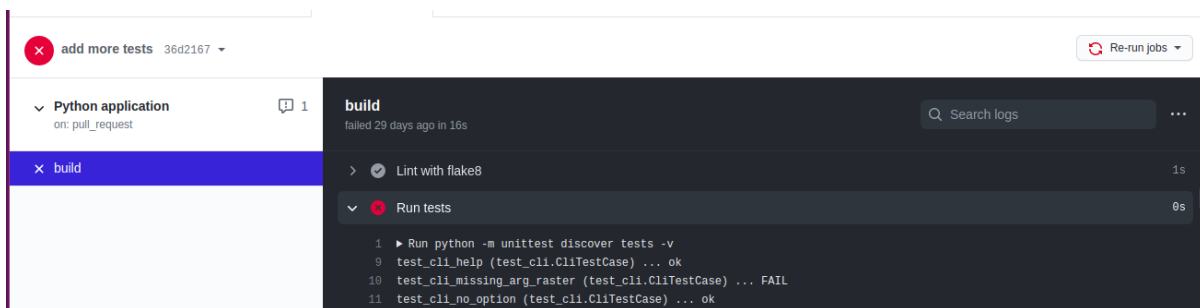


Fig. 2: GitHub actions Pull Request Build Console

5.3.2 Code Coverage Check

moja global uses [Codecov](#) as a tool for tracking coverage of our application. As mentioned above, the Continuous Integration script uploads the coverage report to Codecov. Codecov then compares the coverage percentage to that of our base pull request and asserts if the Code coverage has increased/decreased. After evaluation, the CodeCov bot comments on the pull request with the details of our pull request coverage.

If the coverage percentage remains same or higher than before, the check passes. Else if the percentage becomes lower, the check fails.

- In order to debug and understand the failure of this check you may click on the **Details** tab of the `codecov/` project check under Checks tab.

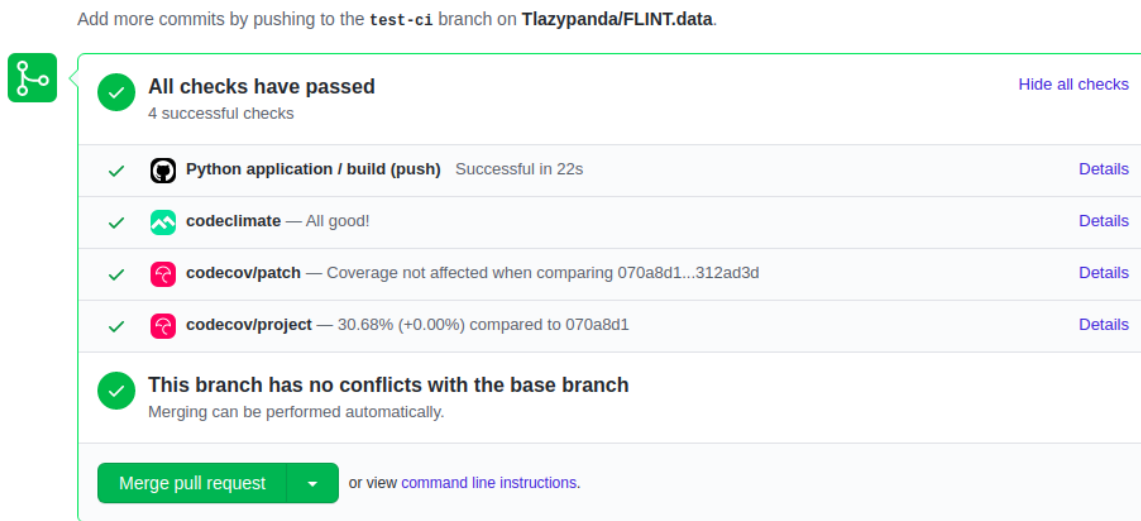


Fig. 3: GitHub pull request Checks

- This will navigate you to the Codecov dashboard where you can see the exact files and their Coverage.



Fig. 4: Codecov Dashboard

- Open the file(s) which has led to the decrease of the coverage and find the exact lines which require tests. The lines marked red here lack tests while the ones marked green are good to go!

```

33
34 ❶ def set_logger(level: str, catch_warnings: bool = False) -> logging.Logger:
35     """Initialize loggers"""
36     level = level.upper()
37
38     package_logger = logging.getLogger('terracotta')
39     package_logger.setLevel(level)
40
41     # stream handler
42     ch = logging.StreamHandler()
43     ch_fmt: logging.Formatter
44
45     ❶ if use_colors:
46         fmt = ' {log_color!s}[{levelshortname!s}]{reset!s} {message!s}'
47
48         class ColoredPrefixFormatter(colorlog.ColoredFormatter):
49             def format(self, record: Any, *args: Any) -> Any:
50                 record.levelshortname = LEVEL_PREFIX[record.levelname]
51                 return super().format(record, *args)
52
53         ch_fmt = ColoredPrefixFormatter(fmt, log_colors=LOG_COLORS, style='{')
54     else:
55         ❶ fmt = ' [{levelshortname!s}] {message!s}'
56
57         class PrefixFormatter(logging.Formatter):
58             ❶ def format(self, record: Any) -> Any:
59                 record.levelshortname = LEVEL_PREFIX[record.levelname]
60                 return super().format(record)
61
62     ❶ ch_fmt = PrefixFormatter(fmt, style='{')

```

Fig. 5: Codecov File coverage

- Add tests for the lines mentioned in the above step. This should resolve your coverage failure.
- Check your coverage locally by running:-

```
coverage run --source flintdata -m unittest discover
```


- Now that your coverage is all set! Modify the pull request to retrigger the Codecov check.


5.3.3 Code Quality Check

moja global currently uses [Codeclimate](#) as our Code Quality tool. Codeclimate is a third-party tool that provides automated code reviews on every pull request for better code maintainability. If any lines of code can be improved according to the programming convention, then the check fails providing detailed information on whichever segments of the code that need to be modified.



- In order to find the Codeclimate check, please click on the **Details** link of the `codeclimate` tab.
- This will redirect you to the detailed report on the issues that needs to be fixed on the codeclimate dashboard. You may also view other files in your code here to check the code quality. The check below depicts the scenario where no issues are found in your code and the code is ready to be merged!
- After debugging issues (if any) in the previous step and modifying the pull request, Codeclimate will automatically check again against the latest changes.



Add more commits by pushing to the `test-ci` branch on **Tlazypanda/FLINT.data**.







 **All checks have passed** [Hide all checks](#)


4 successful checks

  **Python application / build (push)** Successful in 22s [Details](#)

  **codeclimate** — All good! [Details](#)

  **codecov/patch** — Coverage not affected when comparing 070a8d1...312ad3d [Details](#)


  **codecov/project** — 30.68% (+0.00%) compared to 070a8d1 [Details](#)


 **This branch has no conflicts with the base branch**
Merging can be performed automatically.


Merge pull request

 or view [command line instructions](#).

Fig. 6: GitHub pull request Checks

 **Test if code climate is getting triggered #4** [Compare on GitHub](#)

 Tlazypanda wants to merge `test-ci` into `master`

 **All good!** No new issues were found.

[Issues](#) [Ratings](#)

No new, fixed, or changed issues.

Fig. 7: Codeclimate Dashboard

5.4 FLINT Architecture

5.5 FLINT Performance

5.6 Reviewing a contribution

moja global welcomes all contributors to review each others pull requests and suggest changes. If you have been contributing to moja global repositories, we highly encourage you to review pull requests as well. Here's a guide on how to get started!

Quoting GitHub documentation: "Reviews allow for discussion of proposed changes and help ensure that the changes meet the repository's contributing guidelines and other quality standards."

As an open source organisation, reviewing pull requests can help to build a deeper understanding of the codebase and also incorporate good code practices.

5.6.1 General Guidelines for reviewing

- While suggesting changes in the pull requests, the key is to direct your suggestions at the code and not at the **author**. This promotes a healthy discussion instead of making someone feel that their code wasn't upto the mark.
- Make sure the pull request is rebased and is in sync with the latest target branch. For providing the author instructions on how to do that, please direct them to our [pull request section of Git and GitHub guide](#).
- Make sure the pull request is directed to the correct target branch.
- Make sure the feature branch, the commit message and the pull request title is concise and appropriate. For providing the author instructions on how to do that, please direct them to our [commit message section of Code Contribution Best Practices](#).
- Make sure the commits added in the pull request are clean and few in number. This will help us in keeping the commit history clean.
- If the number of files changed in a pull request is quite high, it might be a good idea to ask the author to split the pull request into smaller ones if possible.
- Check whether all PR checks are passing or not. If not, you may also help the author debug these checks and help them contribute.

5.6.2 Things to look out for in a pull request

Performance

If the code changes in the pull request can be optimised in terms of Time/Memory Complexity, feel free to suggest these changes in the pull request. You may use benchmarking tools to find out the difference in Execution Time of the PR code vs the proposed changes.

Code Style and conventions

If the code is not properly formatted or doesn't follow the [style guide here](#), please make sure to suggest changes in the pull request for the same. Following code style and conventions promotes readability and maintainability of code in the longer run.

Documentation

If the code introduces new Features or improves upon existing features that might require documentation to support the change, then don't forget to suggest changes in the pull request to add the same. Proper documentation promotes clarity and makes it easier for future contributors to work on the same feature. For a more detailed guide on how documentation is to be added for any change, please checkout [documentation section here](#).

Tests

If the code requires additional tests to support its validity, please make sure that proper tests have been added. Also make sure the tests added cover edge cases and test various scenarios instead of the most commonly used ones.

Design

If the code changes introduce design changes in terms of UI/UX, please ask for screenshots/gif supporting this change. Feel free to ask for screenshots on devices with different screen sizes such as mobile/tablet view to get a better idea if the design promotes responsiveness. Responsive design helps the application to be more accessible to people thus reaching a wider audience.

5.6.3 What to do if you are not sure

As a reviewer, it might be difficult sometimes to figure out whether the pull request changes will work or not. So in times of doubt, the best way to review is to checkout the contributors branch and manually test the pull request. Here is detailed guide on [manual testing of a pr](#).

5.7 Manually testing a pull request

This guide is aimed at both authors and reviewers of pull requests to properly test the pull request before merging.

- Checkout to the Pull Request branch by running this command where <author> is the GitHub username of the PR author

```
git clone https://github.com/<author>/<repository-name>/  
git checkout <branch-name>
```

- Run the application locally and try to verify if the issue raised has been solved completely by the Pull Request changes.
- If the changes are design specific, try to test the PR in different screen sizes to check if the design is responsive.
- Always keep the console/debugger window open to catch any warnings/errors that might otherwise go unnoticed.
- If the application facilitates multiple user sign up, check with different users to catch any permission/security related issues.

Frequently Asked Questions

6.1 Moja Global

6.1.1 Why is moja global using open source?

moja global is open source for quality and sustainability

The quality of the software is the most important reason for companies to use open source software when it is available. The quality of the open source software is mainly a result from the pooling of resources from various organisations, governments and companies who would otherwise be competing as well as the diversity of the contributors. Open source code management systems have overtaken private systems as open source code has far [fewer bugs](#) at the time the code is accepted.

Open source is a guarantee for sustainability. Once a tool is released as open source, it will remain open source. The tool has been built by various people who keep the ownership over their contribution. They only give others the right to use their contribution under the same licence (in moja global's case mostly MPL2.0). This allows everybody to use the contribution from everybody else. Undoing this decision is not possible as the licence is irrevocable. Practically, it would also be impossible because all the small contributions from every person would have to be dealt with separately.

More info is provided in the White Paper - [Governments, open source, and moja global](#).

6.1.2 Are moja global tools free?

moja global's tools can be used for free

moja global tools are open source and therefore free to use.

The only obligation users have is to share all improvements they make to the software with all other users in line with the [Licence](#).

moja global does not charge any fees nor seeks payments from users. However, for reasons of sustainability, contributions - in kind or cash - are necessary. These are always voluntary.

More information about contributions can be found in the document [Who Pays?](#)

(The misperception is that moja.global is not for free or has hidden costs)

6.1.3 Is moja global controlled by one country?

moja global is a truly global, neutral platform owned by its users

Users own and control moja global. More [info on governance can be found here](#).

All users have the right to take a seat on the board. The strategy board decides on strategy and budget. The board supervises 2 co-directors. The chair of the technical steering committee is a Canadian national.

At the time FLINT started only two countries in the world had a spatially explicit system using integrating software to estimate land-sector GHG fluxes. Those were Canada and Australia. The strong points of each of these systems were taken on in the FLINT design process. As a result FLINT has a number of key ingredients comparable to the software being used in those two countries.

(Misconception: moja global is Australian/Canadian dominated)

6.1.4 How to contract moja global?

Financial support to moja global is possible as grants to the Linux Foundation

Direct financial support is possible through a grant to the Linux Foundation. The money is released once moja global's board approves the expenditure.

However, moja global prefers contributions in kind. Donors can contribute by providing funds directly to contributors, user groups or countries to enable code development as well as documentation and science support.

moja global does not provide implementation support to countries. This role is provided by some parties in moja global's ecosystem (companies, country departments, international organisations, etc.) To remain a credible facilitator, moja global should not compete with its collaborators.

More information about contributions can be found in the document [Who Pays?](#)

6.1.5 What licence is moja global using?

moja global uses mostly MPL2.0

Most moja global software has been released under the open source licence MPL2.0. This means that anybody can download, use, change and redistribute the open source tools on the condition that you share the improvements you make with all other users. This way everybody wins.

The licence is soft-copy-left: If you develop a module without using existing moja global parts, you are under NO obligation to share this module. If you do release it under MPL2.0, that would be great for others of course and very much appreciated but there is no obligation.

This allows companies to develop specific modules or services using the software for commercial purposes (e.g. [FLINTpro SaaS](#)) They are free to do so as long as they share the improvements to the open source software so everybody can profit from those improvements.

6.2 FLINT

6.2.1 What is FLINT?

FLINT is a software platform to estimate emissions and sinks of greenhouse gases from land use

FLINT (Full Lands Integration Tool) is moja global's flagship software platform mainly used for estimating emissions and sinks from land use (but in the future also for economics, biodiversity, etc).

It is a second generation software that is basically a big calculator that can manage data, keep (Carbon or other) pool values, and shift (Carbon or other) amounts between pools based on models / modules.

FLINT is building on the first generation tools developed in Canada (CBM-CFS3) and Australia (FullCAM). Those who designed these first generation tools have joint forces to design and build the second generation tool FLINT.

FLINT is a platform combined with a range of modules that are specific for a particular situation: forest modules, soil modules, litter, wood products, etc. Developing modules can be done externally and they can be connected to the FLINT platform as needed. This design option was chosen to create the flexibility to adapt to local circumstances and to make the development of modules simpler.

Since every user picks and chooses its own particular modules, every user ends up with a(n almost) unique configuration. Several countries have given their national configuration a name: Kenya = SLEEK, Colombia = SEPAC, Indonesia = INCAS, etc.

6.2.2 What is the difference between moja global and FLINT?

moja global is an organisation; FLINT is a software tool

moja global provides rules and infrastructure to help users to collaborate. Its sole aim is to promote the widest possible collaboration on and use of its tools.

FLINT (Full Lands Integration Tool) is one of moja global's open source software tools.

6.2.3 Can beginners use FLINT?

Anybody can use FLINT

The absolute brilliant thing about FLINT is that it is a sophisticated system for entry level users.

Designing a MRV system for the land sector is among the most complex things in the world. Globally there are only a handful of people who have achieved that level of expertise. FLINT is the product of these brains. In essence FLINT is a sophisticated system, designed in such a way it can be used by people at entry level. Additional skills, capacity and depth of understanding are built over time by using the software and through training.

Some level of technical understanding is necessary to run the FLINT. But there is a big difference between the level of technical understanding needed to design your own national MRV system and the technical understanding needed to run the FLINT software and understand the calculations. Compare it to a car: to design a car one needs sophisticated skills. To maintain the car you need practical technical skills. Any lay person with a license can drive the car.

Even to run the system, a user can call on the support of (or hire) other users. Users can **only use their own resources** , use support, or rely on software-as-a-service.

6.2.4 Can we continue to use our old system when switching to FLINT?

FLINT works with whatever is already in place

FLINT is the name for the open source MRV platform offered by moja global. FLINT is combined with science modules to develop country specific configurations. Country specific implementations make each national system unique. In Kenya, FLINT is known as SLEEK. In Canada as Generic Carbon Budget Model (GCBM). In Colombia it is called SEPAC. etc.

FLINT based systems build on the work that has already been done and data that a country has available including land cover maps, forest inventories, emissions factors, etc.

(misconception: FLINT forces users to start from scratch and competes with existing national systems)

6.2.5 Can one see how FLINT calculates emissions?

FLINT is fully transparent

The brilliant advantage of open source is that the tool is always available for review. Everybody is invited to review the code so they can see exactly what the software is doing.

All documentation about the software is accessible.

In addition most modules that can be plugged into the software are open source and have their own detailed documentation.

The aim is to improve the documentation until even those not well versed in IPCC rules can go to the tool, read through the documentation and get a fairly good sense of how the software works in a matter of days. Where documentation is not clear, feedback is used to further improve the documentation or even better, those who have questions are encouraged to suggest improvements to documentation and code.

(Misconception: FLINT is a black box)

6.3 FLINT Installation Support

6.3.1 I am trying to setup FLINT from the master branch but am running into errors. What am I doing wrong?

FLINT's stable development branch is **develop**. **develop** branch is the latest updated branch and should be used as a base branch for development. Therefore, we recommend you to checkout to this branch and target your pull requests against **develop** branch. For more instructions on how to do this, please refer to our [Git and GitHub Guide](#).

6.3.2 I use the Mac operating system. Is it possible to install FLINT?

Yes Absolutely! You can install FLINT on Mac using [our docker installation](#).

6.3.3 What is the difference between FLINT and FLINT.example repositories on moja global GitHub?

FLINT is our framework for estimating emissions and sinks from land use (but in the future also for economics, biodiversity, etc) where the user has to provide the config files or data. Whereas, FLINT.example provides the user with some sample example files that the user can run to get a look and feel of FLINT's output. Hence we recommend you to first install FLINT.example prior to FLINT.

6.3.4 How to configure Visual Studio for FLINT?

To smoothly work with C++ Development on Visual Studio, we recommend you to add Desktop Development with C++ workload while undergoing [Visual Studio installation process](#) mentioned in our prerequisites section.

6.3.5 I am trying to setup the Docker installation for FLINT but am running into errors. What am I doing wrong?

In case of the Docker installation for FLINT, it might be possible that the Docker hardware requirements are not met. Please ensure that atleast 4 CPU cores & 4 GB of RAM has been allotted to the Docker machine.

6.3.6 I have tried the above but my errors persist. What should I do?

We recommend you to join our [Slack workspace](#) and post your queries in the #installation-support channel. We will try to get back to you as soon as possible!

6.4 GCBM

6.4.1 What is GCBM?

GCBM is runs CBM science models on the FLINT platform

GCBM (Generic Carbon Budget Model) is a combination of the FLINT platform with the science modules developed by the Canadian Forest Service.

These are the same science modules used in the first generation tool (CBM-CFS3). Since the science and processes behind both tools are very similar, it is relatively easy to transition from CBM-CFS3 to GCBM.

The CBM-CFS3 is widely used throughout Canada and globally and its use is supported by the Canadian Forest Service of Natural Resources Canada.

The next generation GCBM is currently used by the CFS with a number of partner organizations to advance the science of forest carbon estimation and to support policy analyses such as the assessment of mitigation options in the forest sector.

6.5 FLINTpro

6.5.1 What is FLINTpro?

FLINTpro is a commercial software as a service version using the FLINT platform

[FLINTpro](#) is a cloud-based version of the FLINT platform. It has been developed by the Mullion Group. Using FLINT for commercial purposes is allowed under the MPL2.0 licence. The Mullion Group is sharing all the improvements to the FLINT platform with the open source community.

CHAPTER 7

Join the moja global family

Welcome to the moja global family! Thank you for taking the first step in connecting with us. We at moja global believe in healthy collaboration and invite you to join hands.

7.1 moja global Slack

moja global Slack provides a platform for contributors to communicate, discuss ideas and ask questions. It acts as the primary communication tool used under moja global. The moja global slack is also bound by our [Code Of Conduct](#). Make sure you follow the guidelines before posting. We encourage you to be active participants to make moja global a responsive open source organisation.

In order to receive an invite for the moja global Slack workspace, please email info@moja.global with a description of the project areas you are interested in. After evaluation from the mentors, you will receive a slack workspace invite. Please be patient since it may take time for maintainers to respond.

After joining the Slack Workspace, we encourage you to join project specific channels of your interest. You may also [answer other contributors queries](#) or guide contributors towards resources in case they are stuck.

7.2 Technical Steering Committee Meetings

The moja global team hosts a Technical Steering Committee Meeting on every 2nd Wednesday of the month. These meetings are a way for all the community members to connect, discuss progress and developments from a Technical viewpoint and engage in collaborative discussions for new/existing projects.

If you are new to the community and want to know more about moja global and meet the team, we highly encourage you to attend these meetings. Since the discussions here are not catered to a specific topic, this would be a good place to start and familiarize yourself with our community and work. Interestingly, if you have a new idea for a project or improvements to an existing one, please do join these meetings and share your thoughts with us!

We also recommend all interested GSoC and other student program aspirants to join these meetings to discuss or ask their queries for any project idea that they might have. The moja global team will be sure to hear you out and provide early feedback to steer you towards your goal!

The details for all the meetings are sent out on our [Slack](#). Looking forward to seeing you in the next meeting!

7.3 Outreach and Student Programs

moja global participates in outreach and student programs regularly to welcome and provide students and under-represented communities an opportunity to work with moja global.

In the past, we've participated in programs like [Google Summer Of Code](#) (under DIAL), [Google Season Of Docs](#), [Linux Mentorship Programs](#) and other funding and outreach programs.

Since we receive many applications through these programs, moja global encourages applicants to checkout the following guidelines to maximise chances of getting selected -

- **Start early** - It is encouraged for applicants to start early in the process to get a better understanding of the codebase and suggest new ideas. Many of our GSOC alums had started contributing before the application period and had active engagement in the community.
- **Get in touch with us** - We would love to hear your project ideas and see how they align with our current codebase. Just drop an email at info@moja.global and we will get back to you.
- **Make contributions** - It is highly encouraged to make contributions to the existing repositories. Often the applicants selected are the ones who have made multiple contributions throughout the application period and have stayed consistent.
- **Review peer contributions** - Along with making contributions, it is also encouraged to help out other community members and applicants by providing peer reviews. Code reviews also help to get a deeper understanding of the codebase.
- **Stay active on [Slack](#)** - Being active and friendly on slack by answering queries asked by other new contributors is another quality that moja global values and can make your application stand out.

We wish you all the best in your journey!

Most of our outreach program Alums go on to act as maintainers for their projects and mentor other new contributors in their journey.

Due to limited slots while we can't ensure selection for these programs, if you are passionate to work on the project we will definitely try to find an alternative source of funding.

7.4 moja global Outreach

moja global as an organisation is making efforts to manage Climate Change and act as a Change Maker in the environmental space. We would appreciate if you can spread the word for our initiative and invite more contributors on board to help us build more open source tools available for the world.

Here is how you may help our mission:

- **Star/Fork our projects on GitHub:** Our [GitHub Organization](#) houses many Open source tools that are under active development. We encourage you to star/fork these repositories. You may also help us in their development.
- **Follow us on LinkedIn:** Our [LinkedIn page](#) provides the latest updates about the use of our Software around the world. You may also connect with volunteers at our organisation to discuss and engage in collaborative spirit.
- **Follow us on Twitter:** Our [Twitter page](#) also provides concise latest developments of the organisation.

If you have been contributing to moja global for sometime now, you may also:

- **Write a Blog:** Blog about your experience working with moja global and the impact it has had on your skillset. This will be a great way to let other contributors know about the atmosphere here at moja global.
- **Organise meetups:** [Organise meetups](#) to share your experience in open source by working with moja global and moja global's mission.

We appreciate every little step to promote moja global and are extremely grateful for your efforts!