

Build Guide

MOJA FLINT IMPLEMENTATION

Version 1.0 ● 30 March 2020



www.moja.global

Contents

Introduction	5
Requirements	6
Hardware Requirements	6
Software Requirements	7
Environment Preparation	9
Hardware Preparation	9
Operating System Preparation	10
Tools Installation	13
Libraries Installation	17
Instructions	19
1. Local Build Instructions	19
2. Docker Build Instructions	22
Annex A : Hardware Audits Reference	30
Annex B : HW Configurations Reference	33
Annex C : OS Audits Reference	34
Annex D : OS Configurations Reference	37
Appendix 1 : Basic Dependencies	40
Appendix 2 : Core Dependencies	44
Appendix 3: Environmental Variables	46
Appendix 4: Keys For Accessing BIOS settings	47
Abbreviations	48
References	49

Introduction

A FLINT 'implementation' is a unique configuration of modules attached to the FLINT, with additional data management and output processes.

In practice, FLINT implementations can be run locally or distributed as Docker images that extend Moja's FLINT Library Image. Through this extension, they obtain all the FLINT-level functionality required to coordinate the interaction of data, modules and outputs.

This document provides a step by step guide on the preparation and building of a Moja FLINT Implementation Image, starting from Moja Global's FLINT Image.

Requirements

This chapter highlights the hardware and software requirements for building Moja FLINT Implementations:

Hardware Requirements

a) Recommended Hardware Specifications

01.	Processor	Intel Core i7, Minimum, with Virtualization Support
02.	RAM	16GB, Minimum
03.	Hard Drive	1TB, Minimum

Refer to the following guides to audit how your hardware stacks against these recommendations:

- [Annex A1 : Check Processor Capacity](#)
- [Annex A2 : Check RAM Capacity](#)
- [Annex A3 : Check Hard Drive Capacity](#)
- [Annex A4 : Check Support for Virtualization](#)

Software Requirements

a) Recommended Operating Systems

01.	Windows 10 Pro	Latest Version
02.	Windows 10 Enterprise	Latest Version

Refer to the following guides to audit how your Operating System stacks against these recommendations:

- [Annex C1 : Check Windows Version Edition](#)
- [Annex C2 : Check Windows Version Build Number](#)

Please note that the information on the latest build of Windows 10 can be obtained from this page:

- <https://support.microsoft.com/en-us/help/4464619/windows-10-update-history>

b) Required Tools

01.	CMake	Latest Version
02.	Docker	Latest Version
03.	GIT	Latest Version
04.	Notepad++	Latest Version
05.	Visual Studio	Community 2019

c) Required Libraries

01.	Boost	Version 1.63.0
02.	Eigen	Version 3.3.3
03.	Moja	Latest Version
04.	OpenSSL	Version 1.1.0
05.	POCO	Version 1.7.7
06.	SQLite Amalgamation	Version 3170000
07.	Turtle	Version 1.3.0

Environment Preparation

This chapter highlights how to prepare the environment for building Moja FLINT Implementations.

Hardware Preparation

a) Firmware Virtualization

01. Check if the Firmware Virtualization is enabled

02. Enable it if not

The following guides have been added as references for carrying out the above task:

- [Annex A5 : Check Firmware Virtualization Enablement Status](#)
- [Annex B1 : Enable Firmware Virtualization](#)

Operating System Preparation

a) Windows Version

01. Check if the Windows version is the latest

02. Update it if not

The following guides have been added as references for carrying out the above task:

- [Annex C1 : Check Windows Version Edition](#)
- [Annex C2 : Check Windows Version Build Number](#)
- [Annex D1 : Update to the latest version of Windows 10](#)

Please note that the information on the latest build of Windows 10 can be obtained from this page:

- <https://support.microsoft.com/en-us/help/4464619/windows-10-update-history>

b) Administrative Rights

01. Check if the logged in user account has administrative rights

02. Switch to an account that has administrative rights if not

The following guide has been added as a reference for carrying out the above task:

- [Annex C4 : Check whether a user account has administrative privileges](#)

c) Account Password

01. Check if the logged in user account has a password

02. Set it if not

The following guides have been added as a references for carrying out the above task:

- [Annex C5 : Check whether a user account has a password](#)
- [Annex D2: Add a password to a user account](#)

d) Windows Hyper-V Features

01. Check whether Windows Hyper-V Features have been turned on

02. Turn them on if not

The following guides have been added as a references for carrying out the above task:

- [Annex C6 : Check whether Windows Hyper-V Features have been turned on](#)
- [Annex D3 : Turn on Windows Hyper-V Features](#)

e) Port 445

01. Check whether Port 445 is open for TCP connections

02. Open it if not

The following guides have been added as a references for carrying out the above task:

- [Annex C7 : Check if port 445 is open for TCP connections](#)
- [Annex D4 : Open port 445 for TCP connections](#)

Tools Installation

CMake

a) Pre Installation

01. Go to <https://cmake.org/download/>
 02. Download the latest CMake Binary Distribution for Windows
-

b) Steps

 The following installation steps were written with reference to CMake 3.17.0

01. Right click the CMake installer and select **Install**
 02. Click **Next** to confirm that you want to proceed with the installation
 03. Acknowledge the license terms and Click **Next**
 04. Optionally select the second option to the add **CMake to the system PATH for all users**
 05. Optionally check the last option to create a **CMake Desktop Icon**
 06. Click **Next** to proceed
 07. Leave the install path unchanged to install CMake in the default location
 08. Click **Next** to proceed
 09. Click **Install** to begin the installation
 10. Click **Finish** to exit the installation
-

Docker

a) Pre Installation

01. Complete the [hardware preparation instructions](#) as described earlier
 02. Complete the [operating system preparation instructions](#) as described earlier
 03. Go to <https://www.docker.com/products/docker-desktop>
 04. Download the Docker Desktop installer for Windows ¹
-

b) Steps

 The following installation steps were written with reference to Docker Desktop 2.2.0.3

01. Right click the Docker Desktop installer and select **Run as administrator**
 02. Wait for Docker Desktop to download the required packages
 03. Leave the first checkbox checked to add a Docker Desktop shortcut to your desktop
 04. Click **OK** to proceed
 05. Wait for Docker Desktop to unpack its files and install
 06. Click **Close** to exit the installation
-

¹ This step will ask you to sign-in into your **docker hub** account. If you don't have one, please sign-up up for free here: <https://hub.docker.com/signup>

c) Configuration

 The following configuration steps were written with reference to Docker Desktop 2.2.0.3

01. Double click the **Docker Desktop Shortcut** to start it
 02. Wait for Docker Desktop to notify you that its up and running and then proceed to the next step
 03. Go to the System Tray ² and click the Docker Desktop icon ³
 04. Select **Settings** on the pop-up menu
 05. Select the **Resources** menu on the Settings Window
 06. Click the **Advanced** Resource subcategory if not currently selected
 07. Increase the **Memory** available to Docker to at least 4GB
 08. Click the **File Sharing** Resource subcategory
 09. Select drive **C:** as the local drive you want to be available to your containers
 10. Click **Apply & Restart** to save the changes
 11. Close the settings window after Docker Desktop successfully restarts
-

² The System Tray is another name given to the Notification Area found at the right-side of the Windows Taskbar.

³ If you cannot see the Docker Desktop icon at first glance, try looking for it in the pop-up drawer of the System Tray

Git

a) Pre Installation

01. Go to <https://git-scm.com/downloads>
 02. Download the latest Git binary release for Windows
-

b) Steps

 The following installation steps were written with reference to Git 2.26.0

01. Right click the Docker Desktop installer and select **Run as administrator**
 02. Click **Install** to acknowledge the license terms and carry out the installation
-

c) Configuration ⁴

 The following configuration steps were written with reference to Git 2.26.0

01. Open the Windows 10 search tool
 02. Search for **Windows PowerShell**, open it and use it to execute the commands that follow
 - (i) `git config --global user.name "<your_user_name>"` ⁵ (set your global username)
 - (ii) `git config --global user.email "<your_email@some-service.com>"` ⁶ (set your global email)
-

⁴ These steps assume that you have a Github Account. If you don't, please sign up here: <https://github.com/join>

⁵ Replace the content in the angular bracket, including the bracket itself, with your github username

⁶ Replace the content in the angular bracket, including the bracket itself, with your github email

Notepad++

a) Pre Installation

01. Go to <https://notepad-plus-plus.org/downloads/>
 02. Download the latest Notepad++ release for Windows
-

b) Steps

 The following installation steps were written with reference to Notepad++ 7.8.5

01. Right click the Notepad++ installer and select **Run as administrator**
 02. Leave **English** as the selected language and click **OK**
 03. Click **Next** to confirm that you want to proceed with the installation
 04. Acknowledge the license terms and Click **Next**
 05. Leave the install path unchanged to install Notepad++ in the default location; click **Next**
 06. Leave the selected features unchanged to install the default components; click **Next**
 07. Click **Install** to carry out the installation
 08. Click **Finish** to complete the installation
-

Visual Studio

(a) Pre Installation

02. Go to <https://my.visualstudio.com/Downloads> ⁷
 03. Type **Visual Studio Community 2019** in the **search downloads** field
 04. Press **Enter** and wait for Microsoft to retrieve the desired Visual Studio versions
 05. **Download** the installer of the latest version of Visual Studio Community 2019
-

(b) Steps

 The following installation steps were written with reference to Visual Studio Community 2019 (V 16.5)

01. Right click the Visual Studio installer and select **Run as administrator**
 02. Click **Continue** and wait for the installer to prepare for the installation
 03. Check the **Desktop development with C++** option under the **Workloads** tab
 04. Check the **GitHub extension for Visual Studio** option under the **Individual components** tab ⁸
 05. Click **Install** in the bottom right corner of the screen
 06. Wait for the installation to complete then restart the computer
-

⁷ This will require you to sign in. You can sign in with any of your Microsoft accounts credentials e.g Skype credentials. You can alternatively sign in with your Github credentials

⁸ The **GitHub extension for Visual Studio** option is located under the **Code tools** category

Libraries Installation

Adding the libraries that the Moja FLINT implementations depend upon into the local environment is vital if you plan on building the implementations locally. There are two ways to do this: i) Build the libraries manually into the local environment or ii) build the libraries into the local environment via the vcpkg tool.

When installing third party libraries we recommend using the vcpkg tool to get it right the first time round. When installing the Moja FLINT Library, we recommend that you build it manually.



Important

- See **Moja Base Libraries Build Guide** for instructions on how to build the third party libraries
- See **Moja FLINT Libraries Build Guide** for instructions on how to build the Moja libraries

Instructions

1. Local Build Instructions

The specific steps for building Moja FLINT Implementations cannot be itemized as they will vary from implementation to implementation. We recommend that you refer to the implementation's documentation for proper guidance.

2. Docker Build Instructions

A FLINT Implementation can be conveniently packaged and distributed as a single Docker image. Officially, such images are referred to by the name of the implementation e.g. Moja SLEEK Image for the SLEEK Implementation of the FLINT.

This section provides a step by step guide on the preparation and building of Moja Implementation Images.

2.1. Starting Off

2.1.1. Specify the Parent Image from which the image should be built:

```
FROM moja/flint:bionic
```

- The FLINT Docker Image should by design extend the Moja FLINT Library Image. This is because the Moja FLINT Library Image supplies all the FLINT-level functionality required to coordinate the interaction of data, modules and outputs. Please see the **Moja FLINT Library Image Build Guide** for more information about this.

2.1.2. Add a little metadata to describe the image:

```
LABEL project="=FLINT Examples"\
       image="<flint-implementation> Docker Image"\
       version="1.0"\
       maintainer="Moja Global <info@moja.global>"
```

- It's considered good practice to add a little description about our images so that users can learn more about them should they choose to run the “docker inspect” command. This is typically done through the use of Docker label commands.

2.1.3. Set the image's frontend to noninteractive:

```
ARG DEBIAN_FRONTEND=noninteractive
```

- Ubuntu has several interfaces that can be swapped at will. One of these interfaces: The noninteractive frontend, is considered an anti-frontend. It never interacts with its users at all. Instead, it chooses default answers for all of the questions asked. This makes it an ideal candidate for automatic installs.

2.1.4. Specify the number of CPUs that can be comfortably used during the build process:

```
ARG NUM_CPU=1
```

- This specification will come in handy when controlling the number of jobs that can be run concurrently via “make commands”.

2.1.5. Declare the FLINT Implementation repository variables:

```
ARG TOKENIZED_FLINT_IMP_REPOSITORY_URL  
ARG FLINT_IMP_REPOSITORY_BRANCH
```

- These should be supplied during build time.
- If we were making an image for the SLEEK Implementation of the FLINT for example, the tokenized repository URL would be a URL with the following structure:
https://<Personal_Access_Token>@github.com/SLEEK-TOOLS/moja.sleek.git

2.2. Adding the FLINT Implementation

2.2.1. Clone the FLINT Implementation source code from the specified branch to a local directory:

```
RUN cd $ROOTDIR/src \  
  && git clone --recursive --depth 1 -b ${FLINT_IMP_REPOSITORY_BRANCH} \  
    ${TOKENIZED_FLINT_IMP_REPOSITORY_URL} moja.<flint-implementation>
```

2.2.2. Create a build directory under the repository’s src folder and make it the working directory:

```
RUN mkdir -p moja.<flint-implementation>/Source/build \  
  && cd moja.<flint-implementation>/Source/build
```

2.2.3. Build and install the FLINT Implementation, then clean up the source files:

```
RUN cmake \  
  -DCMAKE_BUILD_TYPE=RELEASE \  
  -DCMAKE_INSTALL_PREFIX=$ROOTDIR .. \  
  -DENABLE_TESTS:BOOL=OFF \  
  -DBoost_USE_STATIC_LIBS=OFF \  
  -DBUILD_SHARED_LIBS=ON \  
  -DMOJA_INCLUDE_DIR=$ROOTDIR/include .. \  
&& make --quiet -j $NUM_CPU \  
&& make --quiet install \  
&& make clean \  
&& cd $ROOTDIR \  
&& rm -Rf /usr/local/src/*
```

2.3. Adding a User

2.3.1. Add a user called moja:

```
RUN useradd -ms /bin/bash moja
```

2.3.2. Set moja as the user to use for subsequent commands:

```
USER moja
```

2.3.3. Change the working directory to Moja's home directory:

```
WORKDIR /home/moja
```

2.4. Saving the Image

Save the image as “Dockerfile.<flint-implementation>.bionic”.

- Please don't include the quotes in the image name.

2.5. Building the image

2.5.1. Change the working directory to the directory with your Docker file.

- If you've been following the instructions in the previous chapter: preparing image, then this is the directory with the file named "Dockerfile.<flint-implementation>.bionic" where "<flint-implementation>" is the name of an actual implementation of the FLINT e.g. SLEEK.

2.5.2. Run the command below to build the image:

```
docker build -f Dockerfile.<flint-implementation>.bionic \  
    --build-arg TOKENIZED_FLINT_IMP_REPOSITORY_URL=[URL] \  
    --build-arg FLINT_IMP_REPOSITORY_BRANCH=[BRANCH] \  
    -t moja/<flint-implementation>:bionic .
```

- The -f option specifies the name of the docker file to be built - in this case, "Dockerfile.<flint-implementation>.bionic".
- The -t option specifies the name that the built image should be tagged with - in this case "moja/<flint-implementation>:bionic".
- The TOKENIZED_FLINT_IMP_REPOSITORY_URL argument specifies the URL of the FLINT Implementation GIT Repository with the user's Personal Access Token pre-appended. If we were making an image for the SLEEK Implementation of the FLINT for example, the tokenized repository URL would be a URL with the following structure:
https://<Personal_Access_Token>@github.com/SLEEK-TOOLS/moja.sleek.git
- The FLINT_IMP_REPOSITORY_BRANCH argument specifies the branch of the FLINT Implementation repository whose source code should be checked out for the build.
- The period, ".", at the end of the command specifies the location of the Docker file to be built - in this case the current directory.
- Please replace all occurrences of "<flint-implementation>" with the name of an actual implementation of the FLINT e.g. SLEEK.

You can optionally update all other variables declared using the ARG directive at the build phase through the use of the --build-arg option. For example:

```
docker build -f Dockerfile.<flint-implementation>.bionic \  
    --build-arg NUM_CPU=4 \  
    --build-arg TOKENIZED_FLINT_IMP_REPOSITORY_URL=[URL] \  
    --build-arg FLINT_IMP_REPOSITORY_BRANCH=[BRANCH] \  
    -t moja/<flint-implementation>:bionic .
```

2.6. Conclusion

This guide illustrated how to prepare and build the Moja FLINT Library Image.

All the code associated with it is available at : <https://github.com/moja-global/flint-examples.git>.

This is a Docker-based project, so it should be easy to import and use as is

Annex A : Hardware Audits Reference

A1 : Check Processor Capacity

- | | |
|-----|---|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the System Information tool and open it |
| 03. | Select System Summary menu on the System Information window |
| 04. | Look for the Processor specification on the right pane |

A2 : Check RAM Capacity

- | | |
|-----|---|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the System Information tool and open it |
| 03. | Select the System Summary menu on the System Information window |
| 04. | Look for the Physical Memory specifications on the right pane |

A3 : Check Hard Drive Capacity

01. Open the Windows 10 search tool
02. Search for the **System Information** tool and open it
02. Expand the **Components** category in the **System Information** window
03. Expand the **Storage** subcategory under the **Components** category
04. Click the Disks subcategory under the **Storage** subcategory
05. Look for the **Size** specifications under the disk descriptions ⁹

A4 : Check Support for Virtualization

01. Open the Windows 10 search tool
02. Search for the **Task Manager** tool and open it
03. Open the **Performance** tab on the opened window ¹⁰
04. Look for a line that says “**Virtualization: (En/Dis)abled**” on the bottom-right side of the opened tab

⁹ Watch out for multiple disk descriptions with different sizes when multiple Hard Drives are present

¹⁰ You might have to click on **More details** to see this tab the very first time you open Task Manager

A5 : Check Firmware Virtualization Enablement Status

- | | |
|-----|---|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the Task Manager tool and open it |
| 03. | Open the Performance tab on the opened window ¹¹ |
| 04. | Look for a line that says “ Virtualization: Enabled ” on the bottom-right side of the opened tab |

¹¹ You might have to click on **More details** to see this tab the very first time you open Task Manager

Annex B : HW Configurations Reference

B1 : Enable Firmware Virtualization

- | | |
|-----|--|
| 01. | Restart the PC |
| 02. | Press the key required to enter BIOS (See Appendix 4: Keys For Accessing BIOS settings) |
| 03. | Navigate to either the Advanced , Security or the Systems Configurations tab |
| 04. | Select Virtualization or Virtualization Technology and then press the Enter key ¹² |
| 05. | Select Enabled and then press the Enter key |
| 06. | Press the F10 key then select Yes and press the Enter key to save the changes and Reboot ¹³ |

¹² On some Lenovo PCs, the Virtualization option will be found buried one level deeper under a **CPU Setup** option

¹³ On some Sony PCs, you will need to navigate to a dedicated **Exit** tab to save changes and exit

Annex C : OS Audits Reference

C1 : Check the Windows Version Edition

- | | |
|-----|---|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the System Information tool and open it |
| 03. | Select the System Summary menu on the System Information window |
| 04. | Look for the OS Name specification on the right pane |

C2 : Check the Windows Version Build Number

- | | |
|-----|---|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the System Information tool and open it |
| 03. | Select the System Summary menu on the System Information window |
| 04. | Look for the Version specification on the right pane |

C3 : Check for the latest Windows Operating System

- | | |
|-----|--|
| 01. | Open https://en.wikipedia.org/wiki/List_of_Microsoft_Windows_versions |
| 02. | Look for the latest Windows Version , Edition and Build Number |

C4 : Check whether a user account has administrative privileges

01. Open the Windows 10 search tool
 02. Search for the **Manage your account** tool and open it
 03. Look for the word "**Administrator**" underneath the **account** name
-

C5 : Check whether a user account has a password

01. Open the Windows 10 search tool
 02. Search for the **Manage your account** tool and open it
 03. Click the **Sign-in options** on the left pane of the opened window
 04. Scroll down to the **Password** section on the right pane of the opened window
 05. Look for a statement that says "**Sign in with your account's password**" underneath it
-

C6 : Check whether Windows Hyper-V features are turned on

01. Open the Windows 10 search tool
 02. Search for the **Turn Windows features on or off** tool and open it
 03. Locate the Hyper-V section and find out if it's checked
-

C7 : Check if port 445 is open for TCP connections

- | | |
|-----|--|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the Windows Defender Firewall tool and open it |
| 03. | Click Advanced settings on the left pane of the Windows Defender Firewall window |
| 04. | Click the Inbound Rules category on the left pane of the newly popped up window |
| 05. | Locate the Local Port column on the newly opened Inbound Rules table |
| 06. | Scroll down this Local Port column and see whether there's a TCP entry for port 445 |
| 07. | Click the Outbound Rules category on the left pane of the newly popped up window |
| 08. | Locate the Remote Port column on the newly opened Outbound Rules table |
| 09. | Scroll down this Remote Port column and see whether there's a TCP entry for port 445 |

Annex D : OS Configurations Reference

D1 : Update to the latest version of Windows 10

01. Go to <https://www.microsoft.com/en-us/software-download/windows10>
02. Click the **Update now** button to download the **Windows 10 Update Assistant**
03. Right click the downloaded **Windows 10 Update Assistant** and select **Run as administrator**
04. Click **Update Now** on the newly opened window
05. Click **Next** after the PC is ascertained as being compatible with the update
06. Click **Minimise** to optionally have the update run in the background
07. Click **Restart now** to restart your PC when the update is complete

D2 : Add a password to a user account

01. Open the Windows 10 search tool
02. Search for the **Manage your account** tool and open it
03. Click the **Sign-in options** on the left pane of the opened window
04. Scroll down to the **Password** section on the right pane of the opened window
05. Click the **Add** button underneath it
06. Enter the password and password hint details and click **Next**
07. Click **Finish**

D3 : Turn on Windows Hyper-V features

- | | |
|-----|--|
| 01. | Open the Windows 10 search tool |
| 02. | Search for the Turn Windows features on or off tool and open it |
| 03. | Locate the Hyper-V section |
| 04. | Check it and click OK |
| 05. | Click Restart now to finish installing the requested changes |

D4 : Open port 445 for TCP connections

01. Open the Windows 10 search tool
02. Search for the **Windows Defender Firewall** tool and open it
03. Click **Advanced settings** on the left pane of the **Windows Defender Firewall** window
04. Click the **Inbound Rules** category on the leftmost pane of the newly popped up window
05. Click the **New Rule** option on the rightmost pane of the newly popped up window
06. Select **Port** as the type of rule to be created
07. Click **Next**
08. Select **TCP** as the protocol of the rule to created
09. Select **Specific local ports** and enter **445** as the port that the rule should be apply to
10. Click **Next**
11. Select **Allow the connection** as the action to take when a connection matches the conditions
12. Check **Domain, Private and Public** to have the rule apply to each of these profiles
13. Click **Next**
14. Enter **Docker** as the name of the rule and click **Finish**
15. Repeat Steps 04 to 14 for **Outbound Rules**

Appendix 1 : Basic Dependencies

Dependency	About
bash-completion	<p>Programmable completion for the bash shell.</p> <p>This package extends bash's standard completion behavior to achieve complex command lines with just a few keystrokes. It was conceived to produce programmable completion routines for the most common Linux/UNIX commands, reducing the amount of typing sysadmins and programmers need to do on a daily basis.</p>
build-essential	<p>Informational list of build-essential packages.</p> <p>This package contains an informational list of packages which are considered essential for building Debian packages. It also depends on the packages on that list, to make it easy to have the build-essential packages installed.</p>
doxygen	<p>Documentation generation tool.</p> <p>Doxygen is a documentation system for C, C++, Java, Objective-C, Python, IDL and to some extent PHP, C#, and D. It can generate an on-line class browser (in HTML) and/or an off-line reference manual (in LaTeX) from a set of documented source files.</p>
doxygen-latex	<p>Doxygen dependency package.</p> <p>Adds dependencies for all LaTeX packages required to build documents using the default stylesheet.</p>
git	<p>Fast, scalable, distributed version control system.</p> <p>This package provides the git main components with minimal dependencies.</p>
gdb	<p>GNU Debugger.</p> <p>GDB is a source-level debugger, capable of breaking programs at any specific line, displaying variable values, and determining where errors occurred. Currently, gdb supports C, C++, D, Objective-C, Fortran, Java, OpenCL C, Pascal, assembly, Modula-2, Go, and Ada. A must-have for any serious programmer.</p>
graphviz	<p>Open source graph visualization software.</p> <p>Graph visualization is a way of representing structural information as diagrams of abstract graphs and networks. This package contains graph visualization command-line tools.</p>

libcurl4-gnutls-dev	<p>Development files and documentation for libcurl (GnuTLS flavour).</p> <p>libcurl is an easy-to-use client-side URL transfer library, supporting DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP. libcurl supports SSL certificates, HTTP POST, HTTP PUT, FTP uploading, HTTP form based upload, proxies, cookies, user+password authentication (Basic, Digest, NTLM, Negotiate, Kerberos), file transfer resume, http proxy tunneling and more.</p>
libeigen3-dev	<p>Lightweight C++ template library for linear algebra.</p> <p>Eigen 3 is a lightweight C++ template library for vector and matrix math, a.k.a. linear algebra. Unlike most other linear algebra libraries, Eigen 3 focuses on the simple mathematical needs of applications.</p>
libgeos-dev	<p>Geometry engine for GIS.</p> <p>GEOS provides a spatial object model and fundamental geometric functions. It implements the geometry model defined in the OpenGIS Consortium Simple Features Specification for SQL.</p>
libhdf4-alt-dev	<p>Hierarchical Data Format development files (without NetCDF).</p> <p>HDF is a multi-object file format for storing and transferring graphical and numerical data mainly used in scientific computing. HDF supports several different data models, including multidimensional arrays, raster images, and tables. Each defines a specific aggregate data type and provides an API for reading, writing, and organizing the data and metadata.</p>
libhdf5-serial-dev	<p>Packages providing libhdf5-serial-dev</p> <p>This is a virtual package.</p>
libnetcdf-dev	<p>Creation, access, and sharing of scientific data.</p> <p>NetCDF (network Common Data Form) is a set of interfaces for array-oriented data access and a freely distributed collection of data access libraries for C, Fortran, C++, Java, and other languages. The netCDF libraries support a machine-independent format for representing scientific data. Together, the interfaces, libraries, and format support the creation, access, and sharing of scientific data.</p>
libpoppler-dev	<p>PDF rendering library.</p> <p>Poppler is a PDF rendering library based on Xpdf PDF viewer.</p>
libpq-dev	<p>Header files for libpq5 (PostgreSQL library).</p> <p>Header files and static library for compiling C programs to link with the libpq library in order to communicate with a PostgreSQL database backend.</p>
libproj-dev	<p>Cartographic projection library.</p>

	<p>Proj and invproj perform respective forward and inverse transformation of cartographic data to or from Cartesian data with a wide range of selectable projection functions (over 100 projections).</p>
libspatialite-dev	<p>Geospatial extension for SQLite.</p> <p>The Spatialite extension enables SQLite to support spatial (geometry) data in a way conformant to OpenGis specifications, with both WKT and WKB formats.</p>
libssl-dev	<p>Secure Sockets Layer toolkit.</p> <p>This package is part of the OpenSSL project's implementation of the SSL and TLS cryptographic protocols for secure communication over the Internet. It contains development libraries, header files, and manpages for libssl and libcrypto.</p>
libxml2-dev	<p>Development files for the GNOME XML library.</p> <p>XML is a metalanguage to let you design your own markup language. A regular markup language defines a way to describe information in a certain class of documents (eg HTML). XML lets you define your own customized markup languages for many classes of documents. It can do this because it's written in SGML, the international standard metalanguage for markup languages.</p>
nasm	<p>General-purpose x86 assembler.</p> <p>Netwide Assembler: NASM will currently output flat-form binary files, a.out, COFF and ELF Unix object files, and Microsoft 16-bit DOS and Win32 object files.</p>
openssl	<p>Secure Sockets Layer toolkit - cryptographic utility.</p> <p>This package is part of the OpenSSL project's implementation of the SSL and TLS cryptographic protocols for secure communication over the Internet. It contains the general-purpose command line binary /usr/bin/openssl, useful for cryptographic operations.</p>
postgis	<p>Geographic objects support for PostgreSQL.</p> <p>PostGIS adds support for geographic objects to the PostgreSQL object-relational database. In effect, PostGIS "spatially enables" the PostgreSQL server, allowing it to be used as a backend spatial database for geographic information systems (GIS).</p>
postgresql-client-10	<p>Front-end programs for PostgreSQL</p> <p>This metapackage always depends on the currently supported database client package for PostgreSQL.</p>
python3-dev	<p>Header files and a static library for Python (default).</p>

	Header files, a static library and development tools for building Python modules, extending the Python interpreter or embedding Python in applications.
python3-numpy	Fast array facility to the Python 3 language. Numpy contains a powerful N-dimensional array object, sophisticated (broadcasting) functions, tools for integrating C/C++ and Fortran code, and useful linear algebra, Fourier transform, and random number capabilities.
python3-pip	Python package installer. pip is the Python package installer. It integrates with virtualenv, doesn't do partial installs, can save package state for replaying, can install from non-egg sources, and can install from version control repositories.
software-properties-common	Manages the repositories that you install software from (common). This software provides an abstraction of the used apt repositories. It allows you to easily manage your distribution and independent software vendor software sources.
sqlite3	Command line interface for SQLite 3. SQLite is a C library that implements an SQL database engine. Programs that link with the SQLite library can have SQL database access without running a separate RDBMS process.
wget	Retrieves files from the web. Wget is a network utility to retrieve files from the web using HTTP(S) and FTP, the two most widely used internet protocols. It works non-interactively, so it will work in the background, after having logged off. The program supports recursive retrieval of web-authoring pages as well as FTP sites.

Appendix 2 : Core Dependencies

Dependency	About
boost	<p>Free, peer-reviewed, portable C++ source libraries.</p> <p>boost libraries are a collection of C++ libraries that provide support for standard tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing..</p>
cmake	<p>Build process manager.</p> <p>CMake is an open-source, cross-platform family of tools designed to build, test and package software.</p>
fmt	<p>A modern formatting library.</p> <p>{fmt} is an open-source formatting library for C++ that can be used as a safe and fast alternative to (s)printf and iostreams</p>
gdal	<p>Raster and Vector translation library.</p> <p>GDAL is a translator library for raster and vector geospatial data formats that is released under an X/MIT style Open Source License by the Open Source Geospatial Foundation.</p>
poco	<p>C++ libraries for building network- and internet-based applications.</p> <p>The PORTable COMponents (POCO) C++ Libraries are a set of cross-platform C++ libraries for developing computer network-centric, portable applications in C++.</p>
rabbitmq-c	<p>This is a C-language AMQP client library for use with v2.0+ of the RabbitMQ broker.</p> <p>RabbitMQ is an open-source message-broker software that originally implemented the Advanced Message Queuing Protocol (AMQP) and has since been extended with a plug-in architecture to support Streaming Text Oriented Messaging Protocol (STOMP), Message Queuing Telemetry Transport (MQTT), and other protocols.</p>
SimpleAmqpClient	<p>C++ wrapper around the rabbitmq-c C library</p> <p>SimpleAmqpClient is an easy-to-use C++ wrapper around the rabbitmq-c C library</p>
sqlite	<p>Database engine.</p>

	<p>SQLite is a C-language library that implements a small, fast, self-contained, high-reliability, full-featured, SQL database engine.</p>
turtle	<p>Mock object library.</p> <p>Turtle is a C++ mock object library based on Boost with a focus on usability, simplicity and flexibility.</p>
zipper	<p>C++ wrapper around minizip compression library.</p> <p>Zipper is a reliable, simple and flexible compression library that supports all kinds of inputs and outputs. Moreover it allows the compression of files into memory instead of being restricted to file compression only, and using data from memory instead of just files as well.</p>

Appendix 3: Environmental Variables

Variable	About
CURL_CA_BUNDLE	CURL_CA_BUNDLE is an environmental variable used to specify a custom Certificate Authority (CA) certificate path.
GDAL_DATA	GDAL_DATA is an environment variable used to specify location of supporting files used by GDAL libraries as well as GDAL and OGR utilities.
GDAL_HTTP_MERGE_CONSECUTIVE_RANGES	GDAL_HTTP_MERGE_CONSECUTIVE_RANGES is an environment variable used to specify if ranges of a single ReadMultiRange request that are consecutive should be merged into a single request
GDAL_HTTP_MULTIPLEX	GDAL_HTTP_MULTIPLEX is an environment variable used to specify if multiplexing can be used to download multiple ranges in parallel, during ReadMultiRange requests that can be emitted by the GeoTIFF driver
GDAL_HTTP_VERSION	GDAL_HTTP_VERSION is an environment variable used to specify which HTTP version to use
LANG	LANG is an environment variable used to specify a locale
LC_ALL	LC_ALL is an environment variable used to override all LC_xxx environmental variables. LC_xxx environment variables e.g. LC_CTYPE, LC_NUMERIC, LC_TIME, LC_COLLATE, LC_MONETARY, LC_MESSAGES, and so on, are the environment variables meant to override LANG and affect a single locale category only
LD_LIBRARY_PATH	LD_LIBRARY_PATH is an environment variable used to specify a list of directories in which to search for Executable and Linkable Format (ELF) libraries at execution time
PATH	PATH is an environment variable used to specify to the shell which directories to search for executable files (i.e., ready-to-run programs) in response to commands issued by a user
PYTHONPATH	PYTHONPATH is an environment variable used to specify additional directories where python will look for modules and packages

Appendix 4: Keys For Accessing BIOS settings

Manufacturer	F1	F2	F3	F6	F10	F11	F12	ESC	INS	DEL
Acer	A	C								C
Asus		C							A	A
DELL	A	C	A				A			A
HP	A	A		A	C	A	A	C		
Lenovo	C	C								
Sony	A	C	C							
Toshiba	A	C						A		

Where C = Most Common and A = Alternative

Abbreviations

Abbreviation	Meaning
CEIP	Customer Experience Improvement Program
CPU	Central Processing unit
FLINT	Full Lands Integration Tool
HW	Hardware
OS	Operating System
RAM	Random Access Memory
SLEEK	System for Lands Based Emissions Estimation in Kenya
TCP	Transmission Control Protocol
URL	Uniform Resource Locator

References

Basic Dependencies:

Ubuntu Packages	https://packages.ubuntu.com/
-----------------	---

Core Dependencies:

Boost C++ Libraries	https://www.boost.org/
CMake	https://cmake.org/
fmtlib/fmt	https://github.com/fmtlib/fmt
GDAL	https://gdal.org/
POCO	https://pocoproject.org/
RabbitMQ C	https://github.com/alanxz/rabbitmq-c
SimpleAmqpClient	https://github.com/alanxz/SimpleAmqpClient
SQLite	https://www.sqlite.org/index.html
Turtle	http://turtle.sourceforge.net/
Zipper	https://github.com/sebastiandev/zipper