

FRONTIER S  
I >

# GINAN INSTALLATION AND INTEGRATION GUIDE

GEOSCIENCE AUSTRALIA

GINAN PPP OPEN-SOURCE TOOLKIT

## DOCUMENT RELEASE

This document version written by FrontierSI on behalf of Geoscience Australia.

Ginan Installation and Integration Guide, Version 0.5 dated 3 July 2023.

Geoscience Australia makes this document publicly available to assist Ginan users with:

- Understanding what Ginan may be used for, and
- How Ginan can be built as a system to best perform certain tasks.

All referenced versions of software were correct at the time of publication. Users are advised to be aware of the latest versions of software that are available, including Ginan, and modify any commands or instructions appropriately.

Geoscience Australia (GA) provides the Ginan open-source software, Ginan position correction products and this document free of charge but on an “as is” and “with all faults” basis without any warranty whatsoever. GA does not warrant that Ginan artefacts of any kind shall meet any requirements or expectations or be fit for any intended purposes.

GA assumes no responsibility for errors or omissions in the contents of Ginan and related services and reserves the right to make additions, deletions, or modifications to Ginan at any time without prior notice.

GA does not guarantee the accuracy, relevance, timeliness, or completeness of any information or data available through [ginan.au](http://ginan.au) or on linked external websites.

All trademarks are the property of their respective owners, and Geoscience Australia makes no claim of ownership by the mention of products that contain these marks.

# TABLE OF CONTENTS

<b>1</b>	<b>PROJECT BACKGROUND.....</b>	<b>4</b>
1.1	The Ginan project .....	4
1.2	This document.....	4
<b>2</b>	<b>CONFIGURATION IDENTIFICATION .....</b>	<b>5</b>
2.1	Use cases .....	5
2.2	Types of configuration .....	5
2.3	Pathways .....	7
2.3.1	Windows .....	7
2.3.2	Linux.....	7
2.3.3	Embedded devices .....	7
2.3.4	Larger scale systems .....	7
2.3.5	Processing flexibility .....	8
<b>3</b>	<b>PC CONFIGURATION - FOUNDATION WORK .....</b>	<b>9</b>
3.1	Supported use cases .....	9
3.2	Operating systems .....	9
3.3	Windows Subsystem for Linux .....	9
<b>4</b>	<b>GINAN AND DOCKER .....</b>	<b>17</b>
4.1	Context .....	17
4.2	Installing Python .....	17
4.3	Installing Docker .....	18
4.4	Setting Ginan to work.....	23
<b>5</b>	<b>THE NATIVE GINAN COMPILATION.....</b>	<b>30</b>
5.1	Understanding the native Ginan compilation components.....	30
5.1.1	GitHub .....	30
5.1.2	Compilers .....	30
5.1.3	make .....	30
5.1.4	Utilities.....	30
5.1.5	Libraries .....	31
5.2	GitHub .....	31
5.3	Installing the compilers.....	34
5.4	Installing make and the utilities.....	36
5.5	Installing libraries.....	41
5.6	Mongo .....	52
5.7	Ginan source .....	56
<b>6</b>	<b>EMBEDDED CONFIGURATION.....</b>	<b>63</b>
6.1	Supported use cases .....	63

6.2	Hardware Software Integration .....	63
7	SERVER BASED CONFIGURATION .....	66
7.1	Supported use cases .....	66
7.2	Hardware Software Integration .....	66

# CONTENTS

## 1 Project background

### 1.1 The Ginan project

Precise positioning based on constellations of navigation satellites brings significant economic and social benefits to Australia. Precise positioning reduces fertiliser and chemical spray waste in agriculture. It improves the efficiency of operations in large mine sites. Precise positioning improves safety in aircraft operations and can even give added freedom of movement to vision impaired people.

The purpose of the Ginan project is to develop the software and data to allow everybody in Australia to enjoy the benefits of precise positioning through the creation of new services and products, and in doing so drive economic growth enhancing Australia's prosperity. Ginan is part of the Positioning Australia program, financed by the Australian Government and managed by Geoscience Australia.

For more information on the Ginan project see the [Geoscience Australia website](#). Ginan is being rolled out in a phased approach and will offer products in four distinct categories:

1. The software itself.
2. Standard precise point positioning (PPP) product files.
3. Precise point positioning correction messages.
4. New PPP products and applications yet to be defined.

For more details of the products produced by the Ginan project see the [product descriptions](#).

### 1.2 This document

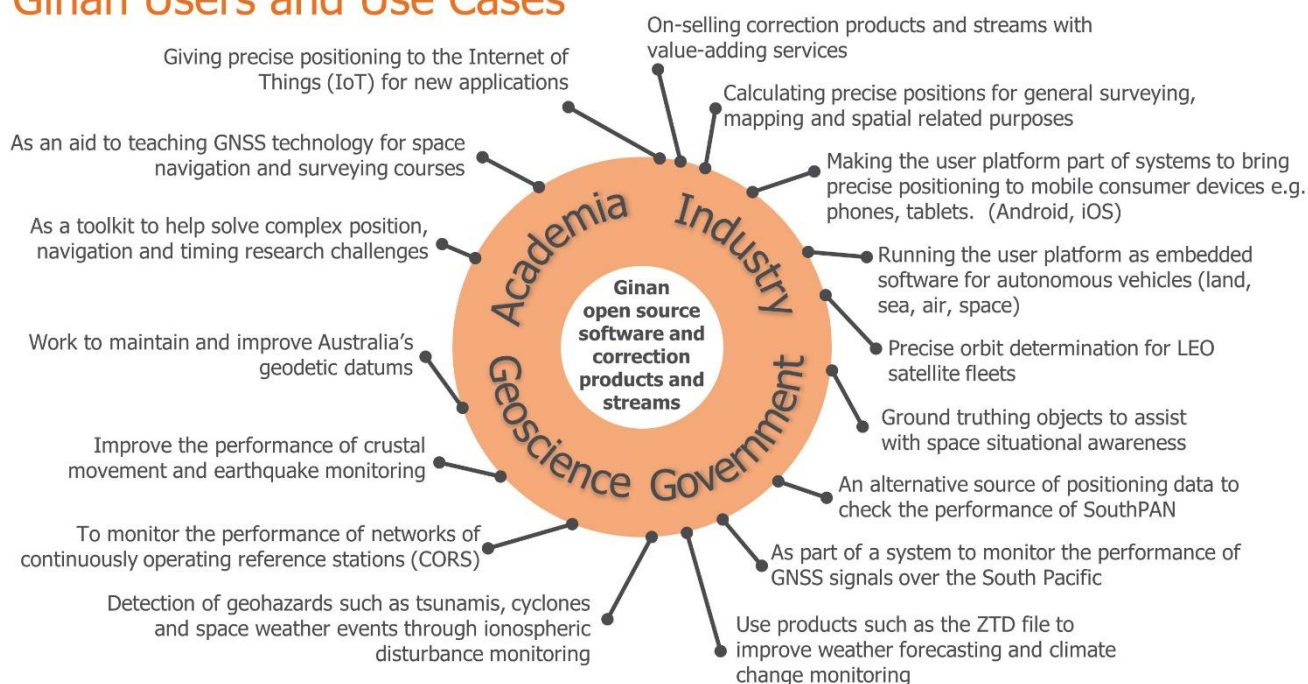
This document describes representative hardware and software configurations, creating systems that both support the Ginan use cases identified to date. It also contains detail on installation requirements and processes.

## 2 Configuration identification

### 2.1 Use cases

The principal Ginan users and use cases identified to date are shown in figure 1 below.

### Ginan Users and Use Cases



**Figure 1** – Ginan Users and Use Cases

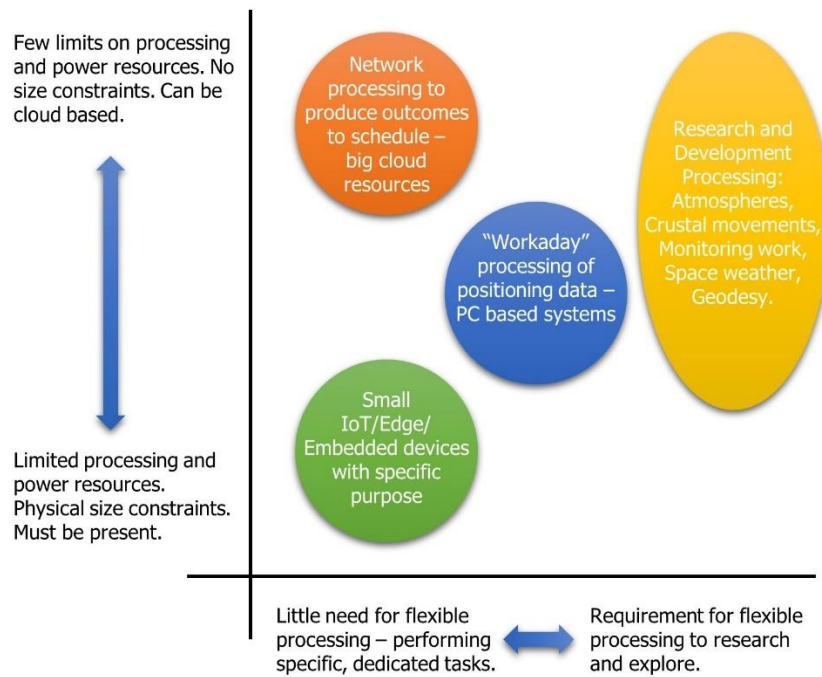
Configurations of Ginan software and hardware can be built to satisfy the requirements of Ginan's use cases.

### 2.2 Types of configuration

Figure 1 identifies sixteen use cases. Although all based on precise positioning, they are quite different in some respects. This is possible because Ginan is a very flexible PPP toolkit. To help the user better understand how the options provided by Ginan can be used to achieve certain outcomes, hardware and software configurations can be further characterised by:

- The need for flexibility in the type of processing that users expect from Ginan,
- Processing power, electrical power and physical size constraints.

This is summarised in figure 2 below:



**Figure 2** – Ginan configuration characteristics

Choices must be made with regards to the software and hardware combinations used in a configuration. These choices are summarised in figure 3 below:



**Figure 3** – Hardware and software combinations

The Geoscience Australia (GA) team develop Ginan on Intel x86 servers running the Ubuntu Linux distribution. This is the “base” hardware and software combination for Ginan. Native compilations of Ginan into this environment are straightforward. In addition to the content in this document, the Ginan GitHub repository contains installation scripts for several Linux distributions.

Docker is the GA team’s preferred way of running Ginan in a container or virtual environment. Docker is a software application that creates a container which provides all the resources that the Ginan Docker image needs to run. Containers increase the range of systems on which Ginan can be used. In simple terms, if a system supports Docker it can run Ginan. Docker Desktop can be downloaded for free and installed on Linux, Windows and MacOS computers.

## 2.3 Pathways

### 2.3.1 Windows

For Windows computers – laptops and PCs – there are two options for running Ginan:

1. Create a Linux environment on the Windows computer using the Windows Subsystem for Linux (WSL) and then natively compile Ginan into that Linux environment following the installation instructions on the GitHub website and in this document.
2. Download the Docker Desktop for Windows and install the Ginan Docker image into Docker. The Ginan Docker image is made publicly available on "Docker Hub" which is an online docker image repository - <https://hub.docker.com/r/gnssanalysis/ginan>. This document and the GitHub website provides instructions for pulling the image from docker hub.

At the current time there is no native Windows installation.

### 2.3.2 Linux

For Linux computers the options are very similar:

1. Ginan source code can be natively compiled into the Linux environment.
2. Download the Docker Engine (as opposed to Docker Desktop) for Linux and install the Ginan Docker image into Docker. The Docker Engine is preferable in this case because it can run natively on Linux. Docker Desktop will install a Linux virtual machine into Linux which is a bit unnecessary.

The same is possible for MacOS given that MacOS is based on a version of Linux and shares many of the same tools. MacOS is a little bit more complicated because depending on their age, Apple PCs use either the Intel x86 processor architecture or the newer ARM based M architecture.

### 2.3.3 Embedded devices

With embedded devices the answer is very much “it depends”.

If the embedded device has the power to run a Linux distribution like Ubuntu, Debian or Fedora, then the pathway is the same as any other Linux device – a native compile or an image run in Docker. However, if the embedded device is based on ARM processor architecture – like the Raspberry Pi – then the code must be compiled for that architecture (as opposed to Intel x86). This requires native compilers for ARM or the Docker buildx tool to create an ARM compatible image.

At the current time there is no simple pathway for devices running Android, iOS (Apple phones and tablets) or Arduino.

### 2.3.4 Larger scale systems

The GA team has demonstrated that Ginan can scale up to undertake network processing tasks.

The structure includes the Ginan software running within the context of a Docker container. Python scripts are used to undertake specific discrete activities required for the processing tasks. The running of the Python scripts is scheduled by an application such as Argo which is designed to manage workflows. The Docker containers are managed by an application like Kubernetes in a cloud service provided by an organisation like Amazon Web Services (AWS).



### 2.3.5 Processing flexibility

Ginan's ability to complete different processing tasks is due to the way its yaml files can be configured. For an application where the processing tasks and the desired outputs are well defined and understood, a yaml file can be developed and tested, and then left to execute within the Ginan instance. Examples include:

1. Ginan in an "edge computing" environment where its role is to calculate its receiver's position on a regular basis and report any instance where deviations exceed a certain threshold.
2. Ginan in a "product production" environment where its task is to take data from a CORS network and produce specific precise point positioning (PPP) products such as SP3 files.

For other applications, the yaml files can deliver flexibility. For example:

1. A researcher has installed Ginan on their laptop. Their research is to examine which combination of GNSS constellations and correction streams gives the most consistently accurate results. The yaml file can be edited to include and exclude combinations of data inputs to meet test case requirements.
2. Another scientist is conducting research into whether CORS can be used to detect earthquakes. They have many CORS connected into their network and one special one which sits on an "earthquake simulator". Due to the heavy data processing load, their instance of Ginan is running in the cloud. The yaml file can be used to connect to all the CORS in the test network, and to test what level of process noise gives the best result when the "earthquake simulator" is shaking away at different displacement magnitudes.

The example yaml files are contained within the exampleConfigs directory of the Ginan GitHub repository:

<https://github.com/GeoscienceAustralia/ginan/tree/main/exampleConfigs>

Documentation on the examples can be found here:

<https://geoscienceaustralia.github.io/ginan/page.html?c=on&p=ginanConfiguration.md#complete-ginan-examples>

### 3 PC configuration - foundation work

#### 3.1 Supported use cases

This configuration supports the following use cases:

- UC 1 As an aid to teaching GNSS technology for space-based navigation, and geodesy and surveying courses,
- UC 2 As a toolkit to help solve complex position, navigation and timing research challenges,
- UC 9 Calculating precise positions for general surveying, mapping and spatial related purposes.

#### 3.2 Operating systems

From the beginning, the Ginan software has been developed on a Linux system, specifically Ubuntu. Users with PCs that run Microsoft Windows have several options:

- Install Windows Subsystem for Linux (WSL 2) and create a Linux environment within their PC, then
- Install Docker and run the Ginan Docker package, or
- Compile Ginan into their new Linux environment and run it from there.

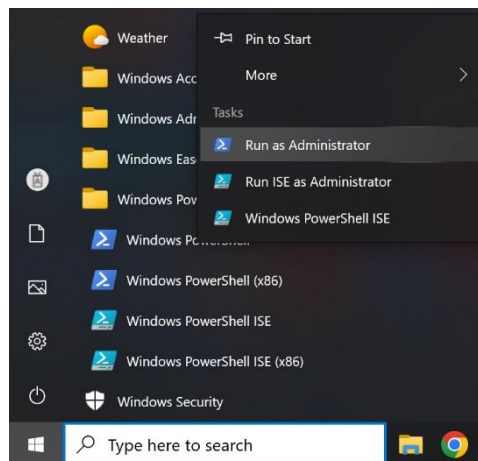
Note that WSL 2 must be installed to use Docker on a Windows PC. WSL2 is the latest version of WSL and it includes the Linux kernel. Any references to WSL in this document mean “WSL 2”.

The following sections describe how to take a Windows laptop or PC and install Linux – in this case the Ubuntu distribution – using WSL.

#### 3.3 Windows Subsystem for Linux

WSL lets users install a Linux distribution (such as Ubuntu) and use Linux applications, utilities, and bash command-line tools directly on Windows, unmodified, without the overhead of a traditional virtual machine or dual-boot setup.

To install WSL a user must be logged into their PC as a user that has administration rights. Administration rights are important – the process will not work without them. The user then must open PowerShell as an Administrator.



**Figure 4** – Opening Windows PowerShell as Administrator

To use WSL, two things must happen. The first is that WSL itself is installed, followed by Ubuntu. Installing WSL means installing a virtual machine within which a Linux kernel can run. The use of a Linux kernel improves input/output (IO) performance and provides full system call compatibility.

The install process allows a user to install several different Linux distributions. The best way to see this is to start with the following command highlighted in yellow:



#### Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Loading personal and system profiles took 761ms.
PS C:\Users\Admin> wsl --list --online
The following is a list of valid distributions that can be installed.
Install using 'wsl --install -d <Distro>'.

NAME                                FRIENDLY NAME
Ubuntu                              Ubuntu
Debian                              Debian GNU/Linux
kali-linux                          Kali Linux Rolling
Ubuntu-18.04                        Ubuntu 18.04 LTS
Ubuntu-20.04                        Ubuntu 20.04 LTS
Ubuntu-22.04                        Ubuntu 22.04 LTS
OracleLinux_7_9                     Oracle Linux 7.9
OracleLinux_8_7                     Oracle Linux 8.7
OracleLinux_9_1                     Oracle Linux 9.1
SUSE-Linux-Enterprise-Server-15-SP4 SUSE Linux Enterprise Server 15 SP4
openSUSE-Leap-15.4                  openSUSE Leap 15.4
openSUSE-Tumbleweed                 openSUSE Tumbleweed
PS C:\Users\Admin>
```

**Figure 5** – Linux distributions available through WSL

At the time of writing the best distribution for Ginan is Ubuntu 22.04 LTS. A user can start the installation in two ways:



#### Windows PowerShell

```
PS C:\Users\Admin> wsl --install
```

**Figure 6** – Simple WSL – Linux install command

This will install WSL and Ubuntu 22.04 LTS because Ubuntu 22.04 LTS is the default Linux distribution. Alternatively, a user can execute the following more specific command:



#### Windows PowerShell

```
PS C:\Users\Admin> wsl -install -d Ubuntu-22.04
```

**Figure 7** – Specific WSL – Ubuntu Linux install command

If WSL is already installed, the command will bring up the WSL help text showing all the various arguments and options.

If the user does not have administrator privileges, the system will respond with the rather dramatic 'Catastrophic error' message. There is nothing catastrophic about it all – it just means the user must acquire those administrator privileges.

All being well, WSL will start to be installed:

#### Windows PowerShell

```
PS C:\Users\Admin> wsl -install
Installing: Windows Subsystem for Linux
[=====                20.0%                ]
```

**Figure 8** – The installation of WSL is underway

Followed by Ubuntu:

#### Windows PowerShell

```
PS C:\Users\Admin> wsl -install
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
[=====                20.0%                ]
```

**Figure 9** – The installation of Ubuntu in WSL is underway

Then, when that is complete the user will get a success message:

#### Windows PowerShell

```
PS C:\Users\Admin> wsl -install
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the system is
rebooted.
PS C:\Users\Admin>
```

**Figure 10** – WSL and Ubuntu installed successfully

The user must now shut down all applications and reboot the computer.

Another step may be needed here. If the user has administrator permissions, then they should be able to find the Ubuntu icon and get started. However, if the user does not have administrator permissions on their computer – often this is the case with work computers – the user will need to install WSL and Ubuntu again under their own user name. This is because each user must have their own instance of Ubuntu. If our user was called George (rather than Admin) the resulting screen would look very similar but with George replacing Admin.



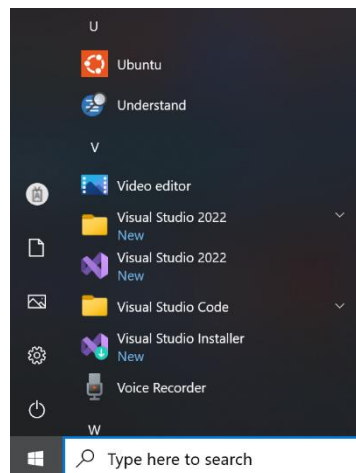
## Windows PowerShell

```
PS C:\Users\george> wsl -install
Installing: Windows Subsystem for Linux
Windows Subsystem for Linux has been installed.
Installing: Ubuntu
Ubuntu has been installed.
The requested operation is successful. Changes will not be effective until the system is
rebooted.
PS C:\Users\george>
```

**Figure 11** – WSL and Ubuntu installed for George

This second installation is much quicker. Again the user must shut down all applications and reboot the computer.

Ubuntu should now appear as a Windows application from the start menu:



**Figure 12** – The Ubuntu icon appears in the Windows start menu for George

Ubuntu can be started by clicking on the icon in the Windows Start menu. Another command line terminal appears:

```
george@ubuntu: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match your  
Windows username.  
For more information visit: https://aka.ms/wslusers  
Enter new UNIX username:
```

**Figure 13** – Setting up Ubuntu

The user provides a username and then is prompted for a password. When typing the password nothing is shown on the screen.

```
george@ubuntu: ~  
Installing, this may take a few minutes...  
Please create a default UNIX user account. The username does not need to match your  
Windows username.  
For more information visit: https://aka.ms/wslusers  
Enter new UNIX username: George  
New password:  
Retype new password:  
passwd: password updated successfully  
Installation successful!  
Windows Subsystem for Linux is now available in the Microsoft Store!  
You can upgrade by running 'wsl.exe -update' or by visiting https://aka.ms/wslstorepage  
Installing WSL from the Microsoft Store will give you the latest WSL updates, faster.  
For more information please visit https://aka.ms/wslstoreinfo  
  
To run a command as administrator (user "root"), use "sudo <command>".  
See "man sudo_root" for details.  
  
Welcome to Ubuntu 22.04.2 LTS (GNU/Linux 4.4.0-19041-Microsoft x86_64)  
  
* Documentation:  https://help.ubuntu.com  
* Management:    https://landscape.canonical.com  
* Support:        https://ubuntu.com/advantage  
  
This message is shown once a day. To disable it please create the /home/George/.hushlogin  
file.  
george@ubuntu:~$
```

**Figure 14** – Setting up a user name and password in WSL Ubuntu

The user name and password is specific to each separate Linux distribution that is installed and has no bearing on the user's Windows user name.

Once a user name and password is created, the account will be the default user for the distribution and automatically sign-in on launch.

This account will be considered the Linux administrator, with the ability to run sudo (Super User Do) administrative commands.

Each Linux distribution running on WSL has its own Linux user accounts and passwords. A Linux user account will have to be configured every time a distribution is added, reinstalled, or reset.

It is obviously very important to remember the user name and password. Information on how to reset a password and a host of other functions can be found at <https://aka.ms/wslusers>

Ubuntu and just about all Linux distributions are constantly being developed with new releases to system components – referred to as packages - released on a regular basis. It is important for users to keep their software up to date, if for no other reason than to try and prevent cyber attacks by closing off vulnerabilities.

The way to update Ubuntu is to use two simple commands: `apt-get update` and `apt-get upgrade`. `apt-get update` downloads the package lists from the repositories and updates them to get information on the newest versions of packages and their dependencies. `apt-get upgrade` will fetch new versions of packages based on those updated package lists.

`apt` is a command-line utility for installing, updating, removing, and otherwise managing deb (Debian) packages on Ubuntu, Debian, and related Linux distributions.

Running the command `apt-get update` will produce an outcome of the form below. The actual information displayed will depend on the user's version of Ubuntu.

```
george@ubuntu: ~
george@ubuntu:~$apt-get update
george@ubuntu:~$[sudo] password for george:
Hit:1 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]
Get:5 http://archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [457 kB]
Get:7 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [123 kB]

Some lines removed here.

Get:37 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 Packages [23.4 kB]
Get:38 http://archive.ubuntu.com/ubuntu jammy-backports/universe Translation-en [15.0 kB]
Get:39 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [548 B]
Get:40 http://archive.ubuntu.com/ubuntu jammy-backports/multiverse amd64 c-n-f Metadata [116 B]
Fetched 25.2 MB in 1min 7s (374 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
46 packages can be upgraded. Run 'apt list --upgradable' to see them.
george@ubuntu:~$
```

**Figure 15** – Running the `apt-get update` command.

Running the command `apt-get upgrade` will produce an outcome of the form below. Again the actual information displayed will depend on the user's version of Ubuntu.

```
george@ubuntu: ~  
george@ubuntu:~$apt-get upgrade  
Reading package lists... Done  
Building dependency tree... Done  
Reading state information... Done  
Calculating upgrade... Done  
The following packages will be upgraded:  
  appport binutils binutils-common binutils-x86-64-linux-gnu ca-certificates distro-info-  
data dpkg iptables libbinutils  
  libctf-nobfd0 libctf0 libgssapi-krb5-2 libip4tc2 libip6tc2 libk5crypto3 libkrb5-3  
libkrb5support0 libncurses6  
  libncursesw6 libperl5.34 libpython3.10 libpython3.10-minimal libpython3.10-stdlib  
libssh-4 libssl3 libtinfo6  
  libxtables12 ncurses-base ncurses-bin openssl perl perl-base perl-modules-5.34 python3-  
appport python3-problem-report  
  python3-software-properties python3.10 python3.10-minimal snapd software-properties-  
common tzdata vim vim-common  
  vim-runtime vim-tiny xxd  
46 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.  
30 standard LTS security updates  
Need to get 60.8 MB of archives.  
After this operation, 13.3 kB disk space will be freed.  
Do you want to continue? [Y/n]
```

**Figure 16** – Running the `apt-get upgrade` command.

To continue with the upgrade, the user types a Y at the prompt:

```
george@ubuntu: ~  
Do you want to continue? [Y/n] y  
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 dpkg amd64 1.21.1ubuntu2.2  
[1239 kB]  
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-bin amd64 6.3-  
2ubuntu0.1 [184 kB]  
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libperl5.34 amd64 5.34.0-  
3ubuntu1.2 [4818 kB]  
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 perl amd64 5.34.0-  
3ubuntu1.2 [232 kB]  
  
Many lines removed here.  
  
Get:43 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libssh-4 amd64 0.9.6-  
2ubuntu0.22.04.1 [185 kB]  
Get:44 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 software-properties-  
common all 0.99.22.7 [14.1 kB]  
Get:45 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 python3-software-  
properties all 0.99.22.7 [28.8 kB]  
Get:46 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 snapd amd64 2.58+22.04.1  
[23.8 MB]
```



```
Fetched 60.8 MB in 2min 32s (401 kB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
(Reading database ... 24137 files and directories currently installed.)
Preparing to unpack .../dpkg_1.21.1ubuntu2.2_amd64.deb ...
Unpacking dpkg (1.21.1ubuntu2.2) over (1.21.1ubuntu2.1) ...
Setting up dpkg (1.21.1ubuntu2.2) ...
(Reading database ... 24137 files and directories currently installed.)
Preparing to unpack .../ncurses-bin_6.3-2ubuntu0.1_amd64.deb ...
Unpacking ncurses-bin (6.3-2ubuntu0.1) over (6.3-2) ...
Setting up ncurses-bin (6.3-2ubuntu0.1) ...
(Reading database ... 24137 files and directories currently installed.)

Many lines removed here.

Preparing to unpack .../24-python3-software-properties_0.99.22.7_all.deb ...
Unpacking python3-software-properties (0.99.22.7) over (0.99.22.6) ...
Preparing to unpack .../25-snapd_2.58+22.04.1_amd64.deb ...
Unpacking snapd (2.58+22.04.1) over (2.58+22.04) ...
Setting up libip4tc2:amd64 (1.8.7-1ubuntu5.1) ...
Setting up python3-problem-report (2.20.11-0ubuntu82.5) ...
Setting up libip6tc2:amd64 (1.8.7-1ubuntu5.1) ...
Setting up distro-info-data (0.52ubuntu0.4) ...
Setting up binutils-common:amd64 (2.38-4ubuntu2.2) ...
Setting up libctf-nobfd0:amd64 (2.38-4ubuntu2.2) ...
Setting up perl-modules-5.34 (5.34.0-3ubuntu1.2) ...
Setting up xxd (2:8.2.3995-1ubuntu2.8) ...
Setting up python3-apport (2.20.11-0ubuntu82.5) ...
Setting up tzdata (2023c-0ubuntu0.22.04.2) ...

Current default time zone: 'Australia/Sydney'
Local time is now:      Tue Jun 13 17:05:52 AEST 2023.
Universal Time is now:  Tue Jun 13 07:05:52 UTC 2023.
Run 'dpkg-reconfigure tzdata' if you wish to change it.

Setting up vim-common (2:8.2.3995-1ubuntu2.8) ...
Setting up python3-software-properties (0.99.22.7) ...

Many lines removed here.

Setting up binutils (2.38-4ubuntu2.2) ...
Setting up software-properties-common (0.99.22.7) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for dbus (1.12.20-2ubuntu4.1) ...
Processing triggers for ca-certificates (20230311ubuntu0.22.04.1) ...
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
george@ubuntu:~$
```

**Figure 17** – running the `apt-get upgrade` command.

The user is now at the point where they have a fully up-to-date version of Ubuntu running on their Windows PC.

## 4 Ginan and Docker

### 4.1 Context

The following sections describe the installation of Docker Desktop and the running of Ginan on a Windows PC that has WSL and Ubuntu installed.

### 4.2 Installing Python

The Ginan team use the computer language Python to write scripts that help users achieve outcomes with Ginan. The Python interpreter must be installed to use those scripts.

On many systems Python comes pre-installed. A user can enter the `python` command to start the Python interpreter to check and see if it is already installed. On windows the `py` command is more likely to work. If Python is installed the response will include the version number, for example:



Windows PowerShell

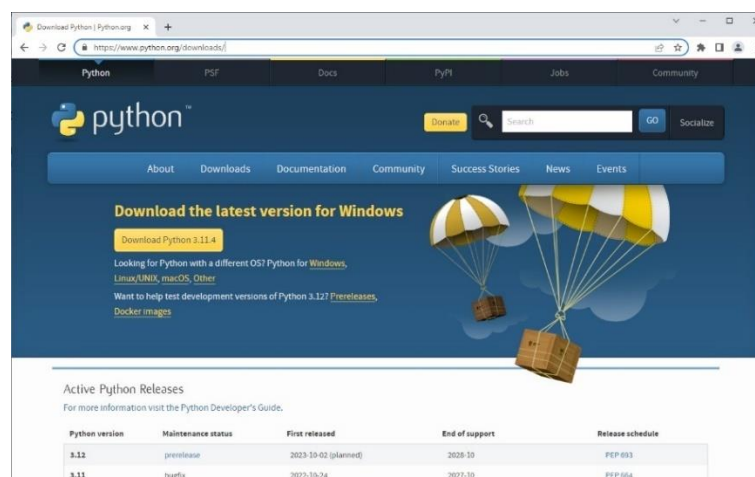
```
PS C:\Users\george> py
Python 3.9.13 (tags/v3.9.13:6de2ca5, May 17 2022, 16:36:42) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>>exit()
PS C:\Users\george>
```

**Figure 18** – Python 3.9.13 is installed on this computer

The latest version of Python for Windows (and other systems) can be downloaded from <https://www.python.org/downloads/>

The installation of Python should be a straightforward operation by downloading and installing the Windows package. More information on installing Python for Windows can be found at python.org here:

<https://docs.python.org/3/using/windows.html>



**Figure 19** – Python download site

### 4.3 Installing Docker

The following assumes that the user has installed WSL 2 on their PC.

For Windows, the Docker version to install is called Docker Desktop (Linux machines are better off installing Docker Engine). Docker Desktop is free for individual developers, education, and open source communities. A user should download the installer and then run it. The installer is available from <https://www.docker.com/products/docker-desktop/>

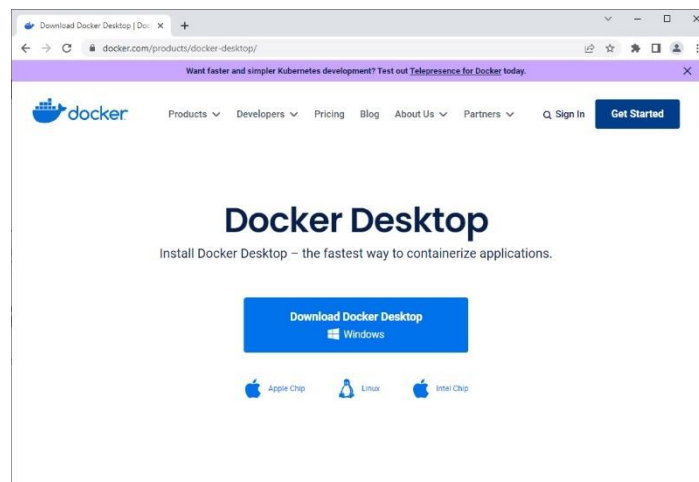


Figure 20 – Docker download site

The user's account must have administration rights to be able to complete the installation.

During the installation, when prompted, ensure the **Use WSL 2 instead of Hyper-V** option on the Configuration page is selected.

Follow the instructions on the installation wizard to authorize the installer and proceed with the install.

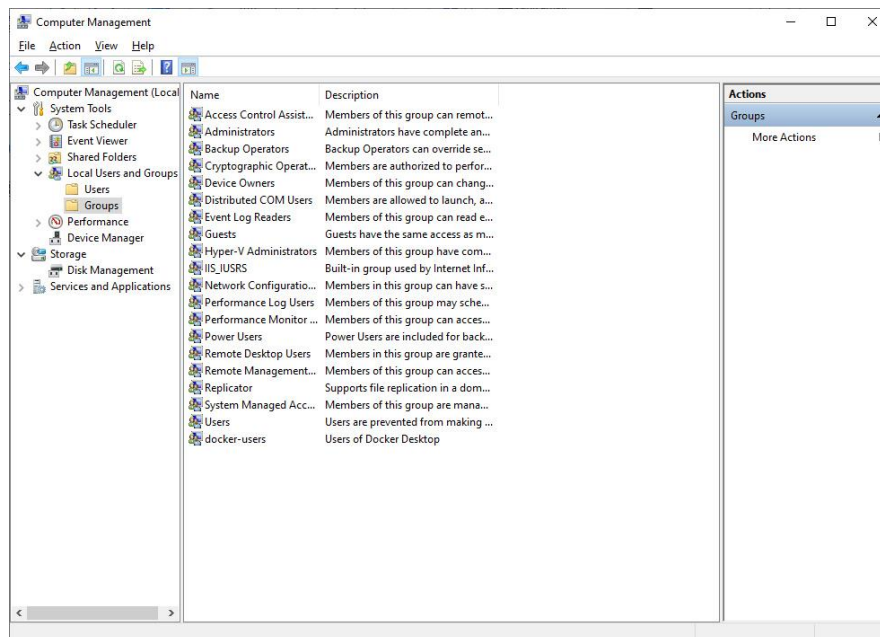
When the installation is successful, click Close to complete the installation process.

If the user tries to start Docker now the following error message will appear:



Figure 21 – Docker denied

If the user's administration account is different to the user's account, the user must be added to the **docker-users** group. To do this the user must run Computer Management as an administrator and navigate to **Local Users and Groups > Groups > docker-users** and add the user to the group. Log out and log back in for the changes to take effect.



**Figure 22** – Docker docker-users group (at the bottom of the list) in the Computer Management window

Before starting, Docker may also warn that WSL needs updating – even if it has just been installed. The user just needs to run the update command:

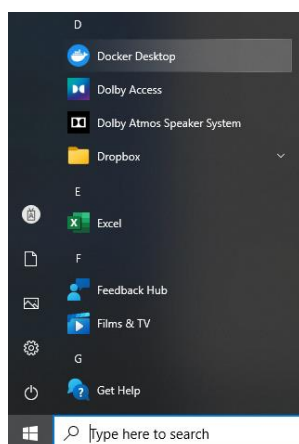
## Windows PowerShell

```
PS C:\Users\george> wsl -update
```

**Figure 23** – WSL update

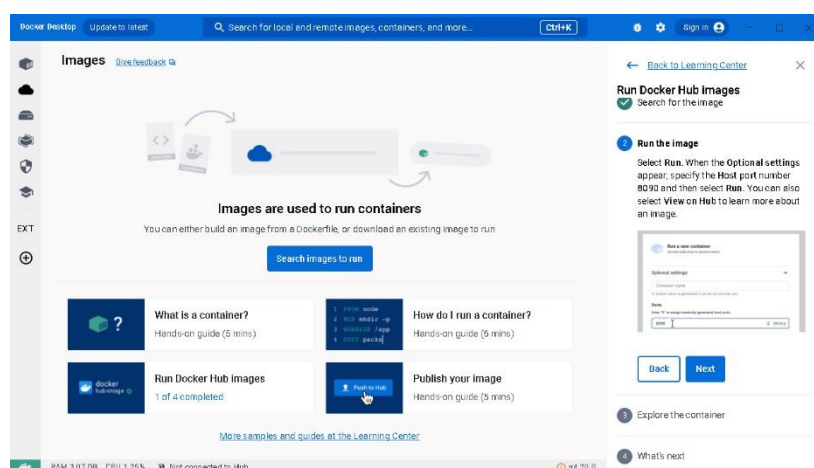
With WSL already installed, the update process is quite quick.

It should be possible now to start Docker. Select Docker Desktop from the Windows start menu:



**Figure 24** – Starting Docker from the Windows Start menu

All being well the Docker Windows GUI will appear:



**Figure 25** – The Docker Windows GUI

Given that this is the first time Docker has run there will be no containers or images visible in the GUI. This is the moment to get the Ginan Docker image. Open a Windows PowerShell terminal and type the command below:



Windows PowerShell

Windows PowerShell

Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell <https://aka.ms/pscore6>

Loading personal and system profiles took 815ms.

```
PS C:\Users\george>docker run -it -v /data:/data gnssanalysis/ginan:v2.0.1-beta bash
```

**Figure 26** – Getting the Ginan Docker image

This command does three things:

- It gets the image itself,
- Mounts the '/data' directory of the host to '/data' directory in the Ginan container
- Runs the Ginan container with an interactive bash shell.

The download can take quite a while - an hour or so depending on the user's internet connection speed. During the process items will transition between **Waiting > Downloading > Extracting > Download complete > Pull complete**:



#### Windows PowerShell

```
PS C:\Users\george>docker run -it -v /data:/data gnssanalysis/ginan:v2.0.1-beta bash
Unable to find image 'gnssanalysis/ginan:v2.0.1-beta' locally
v2.0.1-beta: Pulling from gnssanalysis/ginan
b549f31133a9: Downloading [=====>]
5.921MB/27.5MB
ae3ffc632859: Downloading [>]
4.815MB/475.1MB
e57b14f6fb01: Download complete
0e643926f267: Download complete
6efc22e6754c: Downloading [>]
5.356MB/334.4MB
561508933e07: Waiting
67a4bf1f2d7a: Waiting
5988e8ff1278: Waiting
b8e5661a3761: Waiting
```

**Figure 27** – Getting the Ginan Docker image – in progress



#### Windows PowerShell

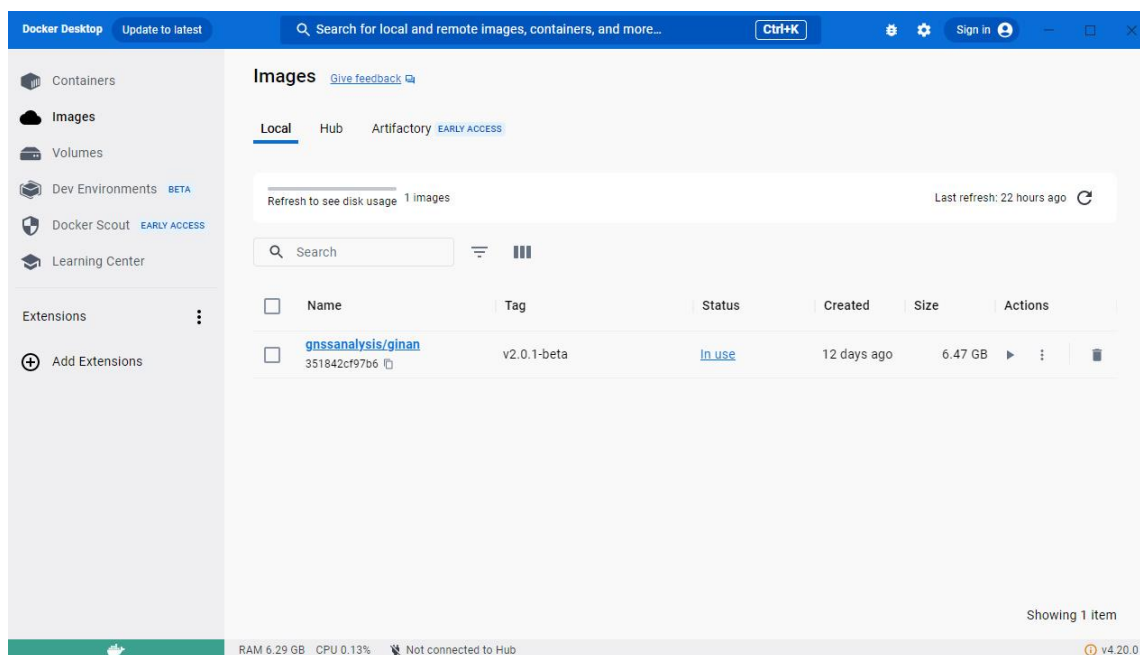
```
PS C:\Users\george>docker run -it -v /data:/data gnssanalysis/ginan:v2.0.1-beta bash
Unable to find image 'gnssanalysis/ginan:v2.0.1-beta' locally
v2.0.1-beta: Pulling from gnssanalysis/ginan
b549f31133a9: Pull complete
ae3ffc632859: Pull complete
e57b14f6fb01: Pull complete
0e643926f267: Pull complete
6efc22e6754c: Pull complete
561508933e07: Pull complete
67a4bf1f2d7a: Pull complete
5988e8ff1278: Pull complete
b8e5661a3761: Pull complete
7b0d0d9fdcb5: Pull complete
01476b575e5b: Pull complete
3b7f27627a92: Pull complete
a055647ab68e: Pull complete
9d3d411ad0c6: Pull complete
9b0126c1dfba: Pull complete
ed42e067422f: Pull complete
03eabd50d5d0: Pull complete
ca995359248a: Pull complete
ad229c548f93: Pull complete
3db5e21b5d54: Pull complete
```

```
Digest: sha256:205148edc58f3289deebb99f8061cbd0ad9240ff1207949d253888b983a37ca5
Status: Downloaded newer image for gnssanalysis/ginan:v2.0.1-beta
root@ccae85695e07:/ginan#
```

**Figure 28** – Getting the Ginan Docker image – complete

Note that at the end the cursor has changed to **root@ccae85695e07:/ginan#**

The Docker image has been downloaded and is running in a Docker container. The Docker Windows GUI shows the image:



**Figure 29** – The Ginan Docker image running in Docker

To confirm that Ginan is installed the user can enter a simple `pea --help` command. Pea is the main component of Ginan.



Windows PowerShell

```
root@ccae85695e07:/ginan# pea --help
```

```
PEA starting... (HEAD release-v2.0.1-beta-e62cecc-dirty from Fri Jun 9 04:16:42 2023)
```

Options:

```
-h [ --help ]           Help
-q [ --quiet ]          Less output
-v [ --verbose ]        More output
-V [ --very-verbose ]   Much more output
-Y [ --yaml-defaults ] arg Print set of parsed parameters and their
                        default values according to their priority
                        level (1-3), and generate configurator.html
                        for visual editing of yaml files
-d [ --config_description ] arg Configuration description
-l [ --level ] arg      Trace level
```

```

-L [ --fatal_message_level ] arg Fatal error level
-e [ --elevation_mask ] arg      Elevation Mask
-n [ --max_epochs ] arg          Maximum Epochs
-i [ --epoch_interval ] arg      Epoch Interval
-u [ --user ] arg                Username for RTCM streams
-p [ --pass ] arg                Password for RTCM streams
-y [ --config ] arg              Configuration file
--atx_files arg                  ANTEX files
--nav_files arg                  Navigation files

Lines removed here

--jpl_files arg                  JPL planetary and lunar ephemerides file
--start_epoch arg                Start date/time
--end_epoch arg                  Stop date/time
--dump-config-only                Dump the configuration and exit
--walkthrough                    Run demonstration code interactively with
                                commentary

PEA finishedroot@ccae85695e07:/ginan#

```

**Figure 30** – The output confirms that Ginan is running

That completes the installation of Docker and the Ginan Docker image.

## 4.4 Setting Ginan to work

The best way to start learning what Ginan can do is to use the examples provided by the Ginan team. The examples consist of two things:

- yaml files which set up the configuration of Ginan to perform certain tasks, and
- the data and metadata files needed to complete those tasks.

The examples can be downloaded from the Ginan GitHub site – exampleConfigs directory. Before that, the user should be aware that running Ginan on Windows using Docker involves having to work with two filesystems – the regular Windows file system, and the file system contained in the Docker container. Both of these systems can be accessed using Windows Explorer. This is necessary for the user to be able to move data around – to get result files and edit yaml files to achieve certain tasks.

To download all the examples a Python script can be used – see below. This is using Python installed previously. This process can be a little unnerving – during the download the text doesn't change at all – there is no indication as to how the process is going, and the download can take quite a while.



Windows PowerShell

```

root@a87fd4549557:/ginan# python3 scripts/download_example_input_data.py
INFO:root:default input path relative to script location selected: /ginan/inputData
INFO:root:default output path relative to script location selected: .
INFO:root:('products', 'data') selected
INFO:root:reading tags from /ginan/docker/tags
INFO:root:requesting checksum for "https://peanpod.s3.ap-southeast-
2.amazonaws.com/aux/products.tar.bz2"
INFO:root:server says OK

```



```

INFO:root:Got "GHcDb6sEdIzbJnbXADo9sg=="
INFO:root:products.tar.bz2 not found on disk ['GHcDb6sEdIzbJnbXADo9sg=='].
INFO:root:requesting "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/products.tar.bz2"
INFO:root:downloading from https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/products.tar.bz2 to /ginan/inputData/products.tar.bz2
INFO:root:Extracting /ginan/inputData/products.tar.bz2 to /ginan/inputData
INFO:root:requesting checksum for "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/data.tar.bz2"
INFO:root:server says OK
INFO:root:Got "Hojhe4sTUBke2g+zTfDEkw=="
INFO:root:data.tar.bz2 not found on disk ['Hojhe4sTUBke2g+zTfDEkw=='].
INFO:root:requesting "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/data.tar.bz2"
INFO:root:downloading from https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/data.tar.bz2 to /ginan/inputData/data.tar.bz2
INFO:root:Extracting /ginan/inputData/data.tar.bz2 to /ginan/inputData
root@a87fd4549557:/ginan#

```

**Figure 31** – Downloading the Ginan examples using the Python script

Once complete, the user can look for the example yaml files using `cd` and `ls` commands:



#### Windows PowerShell

```

root@a87fd4549557:/ginan# ls
CHANGELOG.md  ISSUES.md  README.md  aws  bitbucket-pipelines.yml  docker
inputData  pipelineTests  src
Docs  LICENSE.md  archived  bin  deleteExamples  exampleConfigs  lib
scripts
root@a87fd4549557:/ginan# cd exampleConfigs
root@a87fd4549557:/ginan/exampleConfigs# ls
README.md  ex202.yaml  example_inputs.yaml  products
brdc2sp3.yaml  ex203.yaml  large_post_network.yaml  record_streams.yaml
data  ex204.yaml  medium_post_network.yaml  static_user.yaml
dynamic_user_ppp.yaml  ex205.yaml  mongo_outputs.yaml  tiny_post_network.yaml
dynamic_user_rtk.yaml  ex206.yaml  network_clocks_and_biases.yaml
tiny_realtime_network.yaml
ex02_fit_sp3_pseudoobs.yaml  ex42_gin2_pp_user_3freq.yaml
network_orbits_clocks_and_biases.yaml
ex201.yaml  ex48_gin2_pp_network_orbits_uduc_120s.yaml  pea

```

**Figure 32** – The example yaml files revealed in the exampleConfig directory

At this point there is an opportunity to start customising the Ginan set-up to suit the user. There is a document available on the meaning of yaml file parameters from the Ginan support website. [\[Can add a reference when it there.\]](#)

In this example the user will alter a couple lines in the yaml file. The user wishes to run the example `ex201.yaml` – to achieve *Post Processed Static Receiver positioning using dual frequency GPS and GAL observations, using IGS REPRO3 final orbits clocks and biases*. This contains:

```

ginan/exampleConfigs/ex201.yaml
inputs:

```

```
include_yamls:
  - tiny_post_network.yaml
  - static_user.yaml
  - mongo_outputs.yaml
  - example_inputs.yaml

outputs:
  metadata:
    config_description: "ex201"
```

The user can copy the text of the example from the Ginan GitHub site and then edit it using their editor on Windows. In this instance the user doesn't want to use a Mongo database, but rather have outputs in the form of files. Also they would like to have a GPX format. The new yaml file looks like this:

```
ginan/exampleConfigs/ex201george.yaml
inputs:
  include_yamls:
    - tiny_post_network.yaml
    - static_user.yaml
    #    - mongo_outputs.yaml
    - example_inputs.yaml

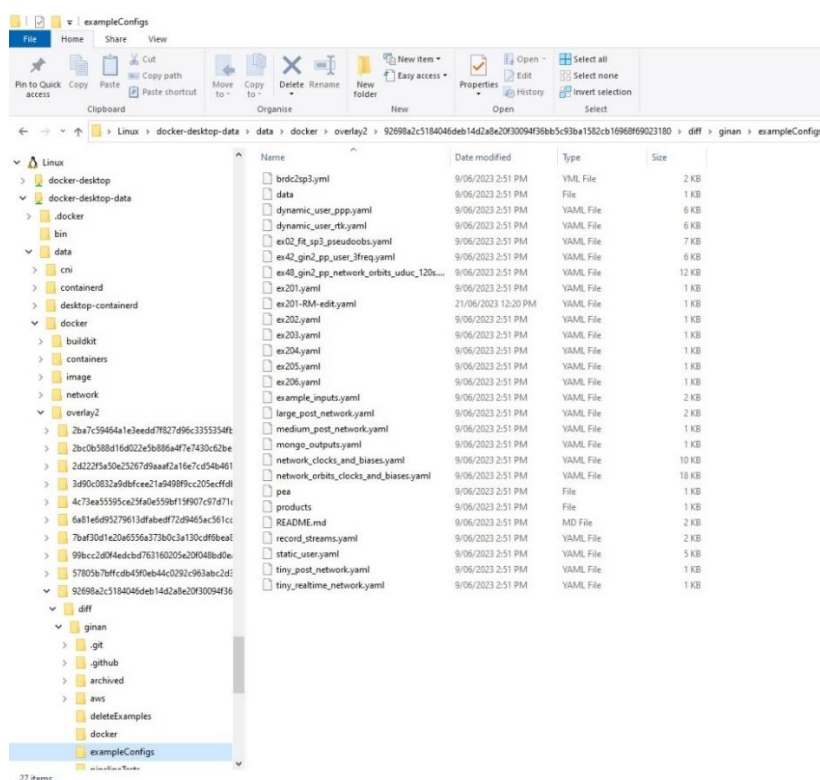
outputs:
  metadata:
    config_description: "ex201"
  root_directory:      /data/outputs/<CONFIG>/
  gpx:
    output: true
```

The mongo output has been commented out using a #. New lines have been added to set the gpx output to true. Also the user has specified that output data is to be put in a directory /data/outputs/<CONFIG>/ where config in this case is simply the yaml name ex201.

The user can then put that new yaml file into the exampleConfigs directory using Windows explorer. The easiest way to do this is to use the Windows search function. All the data associated with the Ginan Docker image will be somewhere in:

Linux > docker-desktop-data > data > docker ...

A search for exampleConfigs against Linux should locate the directory:



**Figure 33** – Using Windows Explorer to find the exampleConfigs directory.

The user can put the new ex201george.yaml into the exampleConfigs. The user is just about ready to run their file. This time though they want the data output to a more convenient Windows directory. That is specified on the command line enclosed in double quotes ""



### Windows PowerShell

```
PS C:\Users\george> docker run -it -v "C:\Users\george\Documents\Ginan\Data:/data"
gnssanalysis/ginan:v2.0.1-beta bash
```

**Figure 34** – Starting the Ginan Docker image and specifying a different output directory

And then the user issues the command to execute the yaml file and off Ginan goes:



### Windows PowerShell

```
root@a87fd4549557:/ginan/exampleConfigs# pea -y ex201george.yaml

PEA starting... (HEAD release-v2.0.1-beta-e62cecc-dirty from Fri Jun 9 04:16:42 2023)

Checking configuration file ex201-RM-edit.yaml
Loading configuration from file tiny_post_network.yaml
Loading configuration from file static_user.yaml
Loading configuration from file example_inputs.yaml
Loading configuration from file ex201-RM-edit.yaml
```

```
=====
```

```
Configuration...
```

```
=====
```

```
Inputs:
```

```
    nav_files:    products/brdm1990.19p
    snx_files:     products/igs19P2062.snx
products/tables/igs_satellite_metadata_2203_plus.snx
products/IGS1R03SNX_20191950000_07D_07D_CRD.SNX
    atx_files:    products/igs20.atx
    clk_files:    products/IGS2R03FIN_20191990000_01D_30S_CLK.CLK
    bsx_files:    products/IGS2R03FIN_20191990000_01D_01D_OSB.BIA
    blq_files:    products/OLOAD_GO.BLQ
```

[Lines removed here](#)

```
Threading with 8 threads
```

```
Logging with trace level:2
```

```
Mongo actions requested but mongo is not available - check it is enabled and connected correctly
```

```
Loading ATX file products/igs20.atx
Loading ERP file products/igs19P2062.erp
Loading SP3 file products/IGS2R03FIN_20191980000_01D_05M_ORB.SP3
Loading SP3 file products/IGS2R03FIN_20191990000_01D_05M_ORB.SP3
Loading SP3 file products/IGS2R03FIN_20192000000_01D_05M_ORB.SP3
Loading NAV file products/brdm1990.19p
```

[Lines removed here](#)

```
Starting to process epochs...
```

```
Starting epoch #1
```

```
Synced 4 stations...
```

```
Warning: SVN not found for E06
```

```
Warning: Block type not found for E06, attitude modelling etc may be affected, check sinex file
```

```
Warning: Looking for precise position, but no precise ephemerides found for E06
```

```
Warning: SSR Corrections not found for E06
```

```
Warning: No sat pos found for E06.
```

```
Warning: SVN not found for E10
```

[Lines removed here](#)

```
-----DOING PPPPP KALMAN FILTER -----
```

```
Processed epoch #1 - GPS time: 2062 345600 - 2019-Jul-18 00:00:00 (took 00:00:06.97)
```

```
Starting epoch #2
```

```
Synced 4 stations...
```

```
Warning: Cannot chunk filter with current configuration - disabling.
```

```
-----DOING PPPPP KALMAN FILTER -----
```

```
Processed epoch #2 - GPS time: 2062 345630 - 2019-Jul-18 00:00:30 (took 00:00:00.45)
```

```
Starting epoch #3
Synced 4 stations...
-----DOING PPPPP KALMAN FILTER -----

Processed epoch #3 - GPS time: 2062 345660 - 2019-Jul-18 00:01:00 (took 00:00:00.42)

Starting epoch #4
Synced 4 stations...

Lines removed here

Starting epoch #106
Synced 4 stations...
-----DOING PPPPP KALMAN FILTER -----

Processed epoch #106 - GPS time: 2062 348750 - 2019-Jul-18 00:52:30 (took 00:00:00.56)

RTS lag: 86220Found FILTER_MINUS
Found TRANSITION_MATRIX
Found TRANSITION_MATRIX
Found MEASUREMENT
Found FILTER_PLUS

RTS lag: 86250Found FILTER_MINUS
Found TRANSITION_MATRIX
Found TRANSITION_MATRIX
Found MEASUREMENT
Found FILTER_PLUS

Lines removed here

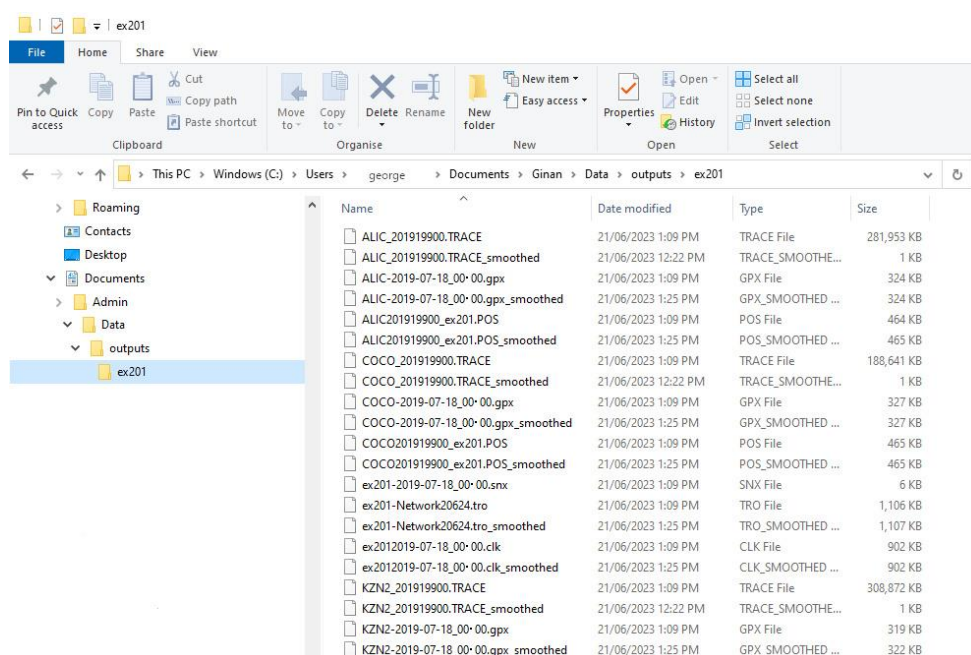
Outputting RTS products...
Removing RTS file: /data/outputs/ex201/./Filter-ex201-Network.rts_forward
Removing RTS file: /data/outputs/ex201/./Filter-ex201-Network.rts_backward

PEA started processing at : 2023-06-21 02:23:04.45
and finished processing at : 2023-06-21 03:25:43.31
Total processing duration : 01:02:38.86

PEA finishedroot@a87fd4549557:/ginan/exampleConfigs#
```

**Figure 35** –Ginan Docker image messages while running

The outputs can be found in the specified directory:



**Figure 36** –Ginan Docker image processing output files

Ginan is a very flexible toolkit that can be used to achieve a wide array of outcomes. The best way to get started is to use the examples. A user will quickly realise and understand though that much of Ginan can be customised to suit their needs. This includes input and output file directories, source data, source data types, output formats and processing options.

## 5 The native Ginan compilation

### 5.1 Understanding the native Ginan compilation components

Docker creates an environment in which the Ginan Docker image can run. The Ginan Docker image is effectively Ginan compiled into executable code along with all its dependencies – all the other software libraries that Ginan needs to run.

With a native compilation the user is effectively building a Ginan image but directly into Ubuntu – there is no container as such. This makes the process a little more involved because the Ginan source code, and the source code for all the dependencies, must be downloaded and then built in a particular order.

#### 5.1.1 GitHub

Many open source projects are housed in GitHub. GitHub provides services including:

- A safe repository to store open source project code,
- Access for several developers to work on code simultaneously,
- The management of changes to code – keeping track of who does what and allowing developers to have their own versions of code,
- Merging different code versions together to create new releases,
- Allowing users to download specific version of code.

Ginan has its own GitHub repository which is where the user will get the Ginan source code. GitHub also provides a software application that simplifies the process of getting code out of GitHub. This is one of the applications that will be installed as part of this process.

#### 5.1.2 Compilers

Compilers are needed to turn the source code into executable code. Ginan uses the GNU Compiler Collection <https://gcc.gnu.org/>. This includes the C compiler `gcc`, the C++ compiler `g++` and the FORTRAN compiler `gfortran`.

#### 5.1.3 make

To help with the compiling and building process, Ginan uses make files. `make` and `cmake` are tools which control the generation of executables and other non-source files of a program from the program's source files. `make` gets its knowledge of how to build Ginan from a file called the makefile, which lists each of the non-source files and how to compute it from other files.

#### 5.1.4 Utilities

`curl` is used in command lines or scripts to transfer data.

`wget` is a free software package for retrieving files using HTTP, HTTPS, FTP and FTPS, the most widely used Internet protocols. It is a non-interactive commandline tool.

`net-tools` is a collection of programs that form the base set of the NET-3 networking distribution for the Linux operating system.

`openssl` software is a robust, commercial-grade, full-featured toolkit for general-purpose cryptography and secure communication.

`openssh-server` is part of a collection of tools for the remote control of, and transfer of data between, networked computers. The server (sshd) listens continuously for client connections from any of the client tools. When a connection request occurs, sshd sets up the correct connection depending on the type of client tool connecting.

**Libssl3:** libssl is the portion of OpenSSL which supports TLS ( SSL and TLS Protocols ), and depends on libcrypto.

### 5.1.5 Libraries

OpenBLAS is an optimized Basic Linear Algebra Subprograms (BLAS) library including support for vector and matrix calculations.

yaml is a human-readable data-serialization language. Files following the yaml syntax are used to control the operation of Ginan.

Boost is a large collection of C++ libraries.

Eigen3 is used for performing matrix calculations.

netCDF4 (Network Common Data Form, version 4) is a set of software libraries and self-describing, machine-independent data formats for array-oriented scientific data.

## 5.2 GitHub

For a Windows PC the prerequisite is that the user has installed WSL and Ubuntu. Before installing `git` the user should complete an `update` and `upgrade`.

```

george@ubuntu: ~
george@ubuntu:~$ sudo apt update
george@ubuntu:~$[sudo] password for george:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Hit:2 http://archive.ubuntu.com/ubuntu jammy InRelease
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-backports InRelease [108 kB]

Some lines removed here.

Get:21 http://archive.ubuntu.com/ubuntu jammy-backports/universe amd64 c-n-f Metadata [552 B]
Get:22 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [131 kB]
Get:23 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [15.6 kB]
Fetched 4928 kB in 19s (256 kB/s)
Reading package lists... Done
Get:36 http://archive.ubuntu.com/ubuntu jammy-backports/restricted amd64 c-n-f Metadata [15.6 kB]
Fetched 4928 kB in 19s (256 kB/s)
Reading package lists... Done
george@ubuntu:~$ sudo apt upgrade
george@ubuntu:~$[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  bind9-dnsutils bind9-host bind9-libs libcap2 libcap2-bin libglib2.0-0 libglib2.0-bin
  libglib2.0-data libpam-cap libx11-6
  libx11-data
11 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 3836 kB of archives.
After this operation, 2048 B of additional disk space will be used.
Do you want to continue? [Y/n] y

```



```
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libcap2 amd64 1:2.44-1ubuntu0.22.04.1 [18.3 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libpam-cap amd64 1:2.44-1ubuntu0.22.04.1 [7928 B]
```

Some lines removed here.

```
Setting up libglib2.0-bin (2.72.4-0ubuntu2.2) ...
Setting up bind9-dnsutils (1:9.18.12-0ubuntu0.22.04.2) ...
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
george@ubuntu:~$
```

**Figure 37** – An update and upgrade prior to installing `git`

Once the update and upgrade are completed the user may type the install command for git: `sudo apt install git-all`. The form of this command – using apt – indicates that git is available as a deb (Debian) package. These packages make installation quite straightforward with the one command.

```
george@ubuntu: ~
george@ubuntu:~$ sudo apt install git-all
[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils bzip2 cvs cvsps fontconfig-config fonts-
dejavu-core git-cvs git-doc
  git-email git-gui git-mediawiki git-svn gitk gitweb libalgorithm-c3-perl libapr1
libaprutil1 libaprutil1-dbd-sqlite3
  libaprutil1-ldap libauthen-sasl-perl libb-hooks-endofscope-perl libb-hooks-op-check-perl
libcgi-fast-perl libcgi-pm-perl
  libclass-c3-perl libclass-c3-xs-perl libclass-data-inheritable-perl libclass-inspector-
perl

Some lines removed here.

Suggested packages:
  apache2-doc apache2-suexec-pristine | apache2-suexec-custom www-browser bzip2-doc mksh
rcs git-daemon-run
  | git-daemon-sysvinit aspell meld mediawiki subversion libgssapi-perl libltdb-perl
libnet-daemon-perl
  libsql-statement-perl libcrypt-ssleay-perl libscalar-number-perl lm-sensors libtest-
fatal-perl libbusiness-isbn-perl
  libauthen-ntlm-perl libyaml-shell-perl tcl-tclreadline mesa-utils xfonts-cyrillic
The following NEW packages will be installed:
  apache2 apache2-bin apache2-data apache2-utils bzip2 cvs cvsps fontconfig-config fonts-
dejavu-core git-all git-cvs
  git-doc git-email git-gui git-mediawiki git-svn gitk gitweb libalgorithm-c3-perl libapr1
libaprutil1
  libaprutil1-dbd-sqlite3 libaprutil1-ldap libauthen-sasl-perl libb-hooks-endofscope-perl
libb-hooks-op-check-perl
```

Many lines removed here.

```
0 upgraded, 192 newly installed, 0 to remove and 0 not upgraded.
Need to get 60.2 MB of archives.
After this operation, 269 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libapr1 amd64 1.7.0-8ubuntu0.22.04.1 [108 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1 amd64 1.6.1-5ubuntu4.22.04.1 [92.6 kB]
Get:3 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-dbd-sqlite3 amd64 1.6.1-5ubuntu4.22.04.1 [11.3 kB]
Get:4 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libaprutil1-ldap amd64 1.6.1-5ubuntu4.22.04.1 [9168 B]
```

Many lines removed here.

```
Get:190 http://archive.ubuntu.com/ubuntu jammy/main amd64 xbitmaps all 1.1.1-2.1ubuntu1 [23.4 kB]
Get:191 http://archive.ubuntu.com/ubuntu jammy/universe amd64 xterm amd64 372-1ubuntu1 [857 kB]
Get:192 http://archive.ubuntu.com/ubuntu jammy/main amd64 libauthen-sasl-perl all 2.1600-1.1 [43.1 kB]
Fetched 60.2 MB in 3min 5s (326 kB/s)
Extracting templates from packages: 100%
Preconfiguring packages ...
Selecting previously unselected package libapr1:amd64.
(Reading database ... 24147 files and directories currently installed.)
Preparing to unpack .../000-libapr1_1.7.0-8ubuntu0.22.04.1_amd64.deb ...
Unpacking libapr1:amd64 (1.7.0-8ubuntu0.22.04.1) ...
Selecting previously unselected package libaprutil1:amd64.
Preparing to unpack .../001-libaprutil1_1.6.1-5ubuntu4.22.04.1_amd64.deb ...
Unpacking libaprutil1:amd64 (1.6.1-5ubuntu4.22.04.1) ...
```

Many lines removed here.

```
Selecting previously unselected package libauthen-sasl-perl.
Preparing to unpack .../191-libauthen-sasl-perl_2.1600-1.1_all.deb ...
Unpacking libauthen-sasl-perl (2.1600-1.1) ...
Setting up libxcb-dri3-0:amd64 (1.14-3ubuntu3) ...
Setting up libx11-xcb1:amd64 (2:1.7.5-1ubuntu0.2) ...
Setting up libpciaccess0:amd64 (0.16-3) ...
Setting up libdrm-nouveau2:amd64 (2.4.113-2~ubuntu0.22.04.1) ...
```

Some lines removed here.

```
Setting up liblwp-protocol-https-perl (6.10-1) ...
Setting up libwww-perl (6.61-1) ...
Setting up libmediawiki-api-perl (0.52-1) ...
Setting up git-mediawiki (1:2.34.1-1ubuntu1.9) ...
Setting up git-all (1:2.34.1-1ubuntu1.9) ...
Processing triggers for install-info (6.8-4build1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
Processing triggers for ufw (0.36.1-4build1) ...
ERROR: Couldn't determine iptables version
Processing ufw triggers failed. Ignoring.
```

```
Processing triggers for man-db (2.10.2-1) ...
george@ubuntu:~$ git version
git version 2.34.1
george@ubuntu:~$
```

**Figure 38** –A successful installation of `git`

That final command and the response indicates that `git` has been successfully installed.

### 5.3 Installing the compilers

To install the compilers (a package with apt) the user will type: `sudo apt install gcc g++ gfortran`

```
george@ubuntu: ~
george@ubuntu:~$ sudo apt install gcc g++ gfortran
george@ubuntu:~$[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cpp cpp-11 g++-11 gcc-11 gcc-11-base gfortran-11 libasan6 libatomic1 libc-dev-bin libc-
devtools libc6-dev libcc1-0
  libcrypt-dev libdeflate0 libgcc-11-dev libgd3 libgfortran-11-dev libgfortran5 libgomp1
libisl23 libitm1 libjbig0
  libjpeg-turbo8 libjpeg8 liblsan0 libmpc3 libns1-dev libquadmath0 libstdc++-11-dev
libtiff5 libtirpc-dev libtsan0
  libubsan1 libwebp7 linux-libc-dev manpages-dev rpcsvc-proto
Suggested packages:
  cpp-doc gcc-11-locales g++-multilib g++-11-multilib gcc-11-doc gcc-multilib make
autoconf automake libtool flex bison gdb
  gcc-doc gcc-11-multilib gfortran-multilib gfortran-doc gfortran-11-multilib gfortran-11-
doc libcoarrays-dev glibc-doc
  libgd-tools libstdc++-11-doc
The following NEW packages will be installed:
  cpp cpp-11 g++ g++-11 gcc gcc-11 gcc-11-base gfortran gfortran-11 libasan6 libatomic1
libc-dev-bin libc-devtools
  libc6-dev libcc1-0 libcrypt-dev libdeflate0 libgcc-11-dev libgd3 libgfortran-11-dev
libgfortran5 libgomp1 libisl23
  libitm1 libjbig0 libjpeg-turbo8 libjpeg8 liblsan0 libmpc3 libns1-dev libquadmath0
libstdc++-11-dev libtiff5 libtirpc-dev
  libtsan0 libubsan1 libwebp7 linux-libc-dev manpages-dev rpcsvc-proto
0 upgraded, 40 newly installed, 0 to remove and 0 not upgraded.
Need to get 73.7 MB of archives.
After this operation, 235 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 gcc-11-base amd64 11.3.0-
1ubuntu1~22.04.1 [20.9 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libisl23 amd64 0.24-2build1 [727
kB]
```

```
Get:3 http://archive.ubuntu.com/ubuntu jammy/main amd64 libmpc3 amd64 1.2.1-2build1 [46.9 kB]
```

Some lines removed here.

```
Get:39 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 libc-devtools amd64 2.35-0ubuntu3.1 [28.9 kB]
```

```
Get:40 http://archive.ubuntu.com/ubuntu jammy/main amd64 manpages-dev all 5.10-1ubuntu1 [2309 kB]
```

```
Fetched 73.7 MB in 3min 17s (374 kB/s)
```

```
Extracting templates from packages: 100%
```

```
Selecting previously unselected package gcc-11-base:amd64.
```

```
(Reading database ... 31430 files and directories currently installed.)
```

```
Preparing to unpack .../00-gcc-11-base_11.3.0-1ubuntu1~22.04.1_amd64.deb ...
```

```
Unpacking gcc-11-base:amd64 (11.3.0-1ubuntu1~22.04.1) ...
```

Some lines removed here.

```
Selecting previously unselected package manpages-dev.
```

```
Preparing to unpack .../39-manpages-dev_5.10-1ubuntu1_all.deb ...
```

```
Unpacking manpages-dev (5.10-1ubuntu1) ...
```

```
Setting up gcc-11-base:amd64 (11.3.0-1ubuntu1~22.04.1) ...
```

```
Setting up manpages-dev (5.10-1ubuntu1) ...
```

```
Setting up libdeflate0:amd64 (1.10-2) ...
```

```
Setting up linux-libc-dev:amd64 (5.15.0-75.82) ...
```

Some lines removed here.

```
Setting up g++-11 (11.3.0-1ubuntu1~22.04.1) ...
```

```
Setting up gfortran (4:11.2.0-1ubuntu1) ...
```

```
update-alternatives: using /usr/bin/gfortran to provide /usr/bin/f95 (f95) in auto mode
```

```
update-alternatives: using /usr/bin/gfortran to provide /usr/bin/f77 (f77) in auto mode
```

```
Setting up g++ (4:11.2.0-1ubuntu1) ...
```

```
update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
```

```
Processing triggers for man-db (2.10.2-1) ...
```

```
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
```

```
george@ubuntu:~$ gcc --version
```

```
gcc (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
```

```
Copyright (C) 2021 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
george@ubuntu:~$ g++ --version
```

```
g++ (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
```

```
Copyright (C) 2021 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
george@ubuntu:~$ gfortran --version
```

```
GNU Fortran (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
```

```
Copyright (C) 2021 Free Software Foundation, Inc.
```

```
This is free software; see the source for copying conditions. There is NO  
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

```
george@ubuntu:~$
```

**Figure 39** – a successful install of `gcc`, `g++` and `gfortran`

The `-version` command shows that the compilers have been installed successfully.

## 5.4 Installing make and the utilities

Ginan needs the following utilities:

- `make`,
- `cmake`,
- `curl`,
- `wget`,
- `net-tools`,
- `openssl`,
- `openssh-server`,
- `libssl3`,

The users version of Ubuntu may have some of those packages already installed, and thanks to apt update – apt upgrade they will be at the latest standard. To check if a package is installed the user can issue the command below:

```

george@ubuntu: ~
george@ubuntu:~$ dpkg -s make cmake curl wget net-tools openssl openssh-server libssl3
dpkg-query: package 'make' is not installed and no information is available

dpkg-query: package 'cmake' is not installed and no information is available

Package: curl
Status: install ok installed
Priority: optional
Section: web
Installed-Size: 443
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Multi-Arch: foreign
Version: 7.81.0-1ubuntu1.10
Depends: libc6 (>= 2.34), libcurl4 (= 7.81.0-1ubuntu1.10), zlib1g (>= 1:1.1.4)
Description: command line tool for transferring data with URL syntax
 curl is a command line tool for transferring data with URL syntax, supporting
 DICT, FILE, FTP, FTPS, GOPHER, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3,
 POP3S, RTMP, RTSP, SCP, SFTP, SMTP, SMTPS, TELNET and TFTP.
 .
 curl 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, proxy tunneling and a
 busload of other useful tricks.
Homepage: https://curl.haxx.se
Original-Maintainer: Alessandro Ghedini <ghedo@debian.org>

Package: wget
Status: install ok installed
Priority: standard
Section: web
Installed-Size: 984

```

```
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Multi-Arch: foreign
Version: 1.21.2-2ubuntu1
Depends: libc6 (>= 2.34), libidn2-0 (>= 0.6), libpcre2-8-0 (>= 10.22), libpsl5 (>=
0.16.0), libssl3 (>= 3.0.0~alpha1), libuuid1 (>= 2.16), zlib1g (>= 1:1.1.4)
Recommends: ca-certificates
Conflicts: wget-ssl
Conffiles:
 /etc/wgetrc c43064699caf6109f4b3da0405c06ebb
Description: 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 -- you can use Wget to make mirrors of archives and
home pages or to travel the web like a WWW robot.
.
Wget works particularly well with slow or unstable connections
by continuing to retrieve a document until the document is fully
downloaded. Re-getting files from where it left off works on
servers (both HTTP and FTP) that support it. Both HTTP and FTP
retrievals can be time stamped, so Wget can see if the remote
file has changed since the last retrieval and automatically
retrieve the new version if it has.
.
Wget supports proxy servers; this can lighten the network load,
speed up retrieval, and provide access behind firewalls.
Homepage: https://www.gnu.org/software/wget/
Original-Maintainer: Noël Köthe <noel@debian.org>
```

dpkg-query: package 'net-tools' is not installed and no information is available

```
Package: openssl
Status: install ok installed
Priority: optional
Section: utils
Installed-Size: 2053
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Multi-Arch: foreign
Version: 3.0.2-0ubuntu1.10
Depends: libc6 (>= 2.34), libssl3 (>= 3.0.2-0ubuntu1.2)
Suggests: ca-certificates
Conffiles:
 /etc/ssl/openssl.cnf 6b4a72a4ce84bab35d884e536295e14f
Description: Secure Sockets Layer toolkit - cryptographic utility
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 such as:
 * creating RSA, DH, and DSA key parameters;
 * creating X.509 certificates, CSRs, and CRLs;
```

```

* calculating message digests;
* encrypting and decrypting with ciphers;
* testing SSL/TLS clients and servers;
* handling S/MIME signed or encrypted mail.
Homepage: https://www.openssl.org/
Original-Maintainer: Debian OpenSSL Team <pkg-openssl-devel@alioth-lists.debian.net>

dpkg-query: package 'openssh-server' is not installed and no information is available

Package: libssl3
Status: install ok installed
Priority: optional
Section: libs
Installed-Size: 5824
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Architecture: amd64
Multi-Arch: same
Source: openssl
Version: 3.0.2-0ubuntu1.10
Depends: libc6 (>= 2.34), debconf (>= 0.5) | debconf-2.0
Description: Secure Sockets Layer toolkit - shared libraries
 This package is part of the OpenSSL project's implementation of the SSL
 and TLS cryptographic protocols for secure communication over the
 Internet.
.
 It provides the libssl and libcrypto shared libraries.
Homepage: https://www.openssl.org/
Original-Maintainer: Debian OpenSSL Team <pkg-openssl-devel@alioth-lists.debian.net>

```

**Figure 40** – checking installed utilities

That command tells the user that:


- curl,
- wget,
- openssl,
- libssl3,

are already installed, but that:

- make,
- cmake,
- net-tools,
- openssh-server,

are not installed at this time, so they are needed. The user achieves this with the following command:

```

 george@ubuntu: ~
george@ubuntu:~$ sudo apt install make cmake net-tools openssh-server
[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done

```

```

Reading state information... Done
The following additional packages will be installed:
  cmake-data dh-elpa-helper emacsen-common libarchive13 libjsoncpp25 librhash0 libwrap0
ncurses-term
  openssh-sftp-server ssh-import-id
Suggested packages:
  cmake-doc ninja-build cmake-format lrzip make-doc molly-guard monkeysphere ssh-askpass
The following NEW packages will be installed:
  cmake cmake-data dh-elpa-helper emacsen-common libarchive13 libjsoncpp25 librhash0
libwrap0 make ncurses-term
  net-tools openssh-server openssh-sftp-server ssh-import-id
0 upgraded, 14 newly installed, 0 to remove and 0 not upgraded.
Need to get 8704 kB of archives.
After this operation, 40.2 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 openssh-sftp-server amd64
1:8.9p1-3ubuntu0.1 [38.7 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/main amd64 libwrap0 amd64 7.6.q-31build2
[47.9 kB]

```

Some lines removed here.

```

Get:11 http://archive.ubuntu.com/ubuntu jammy/main amd64 make amd64 4.3-4.1build1 [180 kB]
Get:12 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 ncurses-term all 6.3-
2ubuntu0.1 [267 kB]
Get:13 http://archive.ubuntu.com/ubuntu jammy/main amd64 net-tools amd64
1.60+git20181103.0eebece-1ubuntu5 [204 kB]
Get:14 http://archive.ubuntu.com/ubuntu jammy/main amd64 ssh-import-id all 5.11-0ubuntu1
[10.1 kB]
Get:10 http://archive.ubuntu.com/ubuntu jammy-updates/main amd64 cmake amd64 3.22.1-
1ubuntu1.22.04.1 [5013 kB]
Fetched 8704 kB in 1min 12s (120 kB/s)
Preconfiguring packages ...
Selecting previously unselected package openssh-sftp-server.
(Reading database ... 36517 files and directories currently installed.)
Preparing to unpack .../00-openssh-sftp-server_1%3a8.9p1-3ubuntu0.1_amd64.deb ...
Unpacking openssh-sftp-server (1:8.9p1-3ubuntu0.1) ...
Selecting previously unselected package libwrap0:amd64.
Preparing to unpack .../01-libwrap0_7.6.q-31build2_amd64.deb ...

```

Some lines removed here.

```

Selecting previously unselected package make.
Preparing to unpack .../10-make_4.3-4.1build1_amd64.deb ...
Unpacking make (4.3-4.1build1) ...
Selecting previously unselected package ncurses-term.
Preparing to unpack .../11-ncurses-term_6.3-2ubuntu0.1_all.deb ...
Unpacking ncurses-term (6.3-2ubuntu0.1) ...
Selecting previously unselected package net-tools.
Preparing to unpack .../12-net-tools_1.60+git20181103.0eebece-1ubuntu5_amd64.deb ...
Unpacking net-tools (1.60+git20181103.0eebece-1ubuntu5) ...
Selecting previously unselected package ssh-import-id.
Preparing to unpack .../13-ssh-import-id_5.11-0ubuntu1_all.deb ...
Unpacking ssh-import-id (5.11-0ubuntu1) ...
Setting up openssh-sftp-server (1:8.9p1-3ubuntu0.1) ...
Setting up net-tools (1.60+git20181103.0eebece-1ubuntu5) ...

```



```
Setting up libarchive13:amd64 (3.6.0-1ubuntu1) ...
Setting up ssh-import-id (5.11-0ubuntu1) ...
Setting up emacs-common (3.0.4) ...
Setting up libwrap0:amd64 (7.6.q-31build2) ...
Setting up make (4.3-4.1build1) ...
Setting up dh-elpa-helper (2.0.9ubuntu1) ...
Setting up libjsoncpp25:amd64 (1.9.5-3) ...
Setting up librhash0:amd64 (1.4.2-1ubuntu1) ...
Setting up cmake-data (3.22.1-1ubuntu1.22.04.1) ...
Setting up ncurses-term (6.3-2ubuntu0.1) ...
Setting up openssh-server (1:8.9p1-3ubuntu0.1) ...

Creating config file /etc/ssh/sshd_config with new version
Creating SSH2 RSA key; this may take some time ...
3072 SHA256:Z0Hk7H01SfH3hL6SoFDu4GA7nKGAcIbGnhZ9QG/Avk8 root@FSI-LT-12 (RSA)
Creating SSH2 ECDSA key; this may take some time ...
256 SHA256:YawybyYP5XM08X/VCREt3UDhZw1iQ49v3fosKou2D9Jo root@FSI-LT-12 (ECDSA)
Creating SSH2 ED25519 key; this may take some time ...
256 SHA256:ky6Gdaeo72TCq4I7Bi1jQhaC+gsEGwZNhe/ZOCKUhrG root@FSI-LT-12 (ED25519)
invoke-rc.d: could not determine current runlevel
Created symlink /etc/systemd/system/ssh.service → /lib/systemd/system/ssh.service.
Created symlink /etc/systemd/system/multi-user.target.wants/ssh.service →
/lib/systemd/system/ssh.service.
Setting up cmake (3.22.1-1ubuntu1.22.04.1) ...
Processing triggers for ufw (0.36.1-4build1) ...
ERROR: Couldn't determine iptables version
Processing ufw triggers failed. Ignoring.
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
george@ubuntu:~$
```

**Figure 41** – Installing utilities

Another dpkg command will now show all those packages have been installed.

If the user tries to install a package that no longer has an installation candidate (deb package) they will get a message along the lines of that below:

```
george@ubuntu: ~
george@ubuntu:~$ sudo apt install libssl1.0-dev
[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Package libssl1.0-dev is not available, but is referred to by another package.
This may mean that the package is missing, has been obsoleted, or
is only available from another source

E: Package 'libssl1.0-dev' has no installation candidate
```

**Figure 42** – An installation candidate cannot be found

This usually means that the particular version of the package is no longer supported by this particular version of Linux – and that there is likely to be a newer version available. This can be fixed by specifying that newer version. However, if a particular version is required it may become a little more complicated. It might mean finding and downloading the source for a version and then compiling.

## 5.5 Installing libraries

Ginan needs the following libraries:

- OpenBLAS,
- yaml,
- Boost,
- Eigen3,
- netCDF4,

OpenBLAS has a deb package and so can be installed in the usual way:

```

george@ubuntu: ~
george@ubuntu:~$ sudo apt install libopenblas-dev
[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  libopenblas-pthread-dev libopenblas0 libopenblas0-pthread
The following NEW packages will be installed:
  libopenblas-dev libopenblas-pthread-dev libopenblas0 libopenblas0-pthread
0 upgraded, 4 newly installed, 0 to remove and 0 not upgraded.
Need to get 11.5 MB of archives.
After this operation, 107 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libopenblas0-pthread amd64
0.3.20+ds-1 [6803 kB]

Some lines removed here.

Setting up libopenblas0:amd64 (0.3.20+ds-1) ...
Setting up libopenblas-pthread-dev:amd64 (0.3.20+ds-1) ...
update-alternatives: using /usr/lib/x86_64-linux-gnu/openblas-pthread/libblas.so to
provide /usr/lib/x86_64-linux-gnu/libblas.so (libblas.so-x86_64-linux-gnu) in auto mode
update-alternatives: using /usr/lib/x86_64-linux-gnu/openblas-pthread/liblapack.so to
provide /usr/lib/x86_64-linux-gnu/liblapack.so (liblapack.so-x86_64-linux-gnu) in auto
mode
update-alternatives: using /usr/lib/x86_64-linux-gnu/openblas-pthread/libopenblas.so to
provide /usr/lib/x86_64-linux-gnu/libopenblas.so (libopenblas.so-x86_64-linux-gnu) in auto
mode
Setting up libopenblas-dev:amd64 (0.3.20+ds-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
george@ubuntu:~$

```

**Figure 43** – an installation OpenBLAS

Yaml does not have a deb package but it is straightforward to download and compile. The user will use git – installed earlier – to get the software from the GitHub repository. This repository is maintained by jbeder and is a C++ implementation of the yaml 1.2 specification.

The user creates a temporary directory tmp, moves into that directory then uses the `git clone` command to get the yaml software.

```

george@ubuntu: ~
george@ubuntu:~$ mkdir tmp
george@ubuntu:~$ ls
tmp
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ git clone https://github.com/jbeder/yaml-cpp.git
Cloning into 'yaml-cpp'...
remote: Enumerating objects: 8598, done.
remote: Counting objects: 100% (3/3), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 8598 (delta 0), reused 1 (delta 0), pack-reused 8595
Receiving objects: 100% (8598/8598), 4.31 MiB | 619.00 KiB/s, done.
Resolving deltas: 100% (5610/5610), done.
george@ubuntu:~/tmp$ ls
yaml-cpp
george@ubuntu:~/tmp$ cd yaml-cpp
george@ubuntu:~/tmp/yaml-cpp$ ls
BUILD.bazel      LICENSE      cmake_uninstall.cmake.in  install.txt  util
CMakeLists.txt  README.md   docs                    src          yaml-cpp-
config.cmake.in
CONTRIBUTING.md  WORKSPACE  include                test         yaml-cpp.pc.in
george@ubuntu:~/tmp/yaml-cpp$ cd ..
george@ubuntu:~/tmp$

```

**Figure 44** – Cloning yaml from GitHub

With the following commands the user is creating a new directory called cmake-build, and then is using the `cmake` tool to create makefiles. The items after the `cmake` command are parameters passed into the `cmake` process. The first tells `cmake` where the software is to be installed.

```

george@ubuntu: ~
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ cd yaml-cpp
george@ubuntu:~/tmp/yaml-cpp$ mkdir cmake-build
george@ubuntu:~/tmp/yaml-cpp$ cd cmake-build
george@ubuntu:~/tmp/yaml-cpp/cmake-build$ cmake .. -DCMAKE_INSTALL_PREFIX=/usr/local/ -
DYAML_CPP_BUILD_TESTS=OFF
-- The CXX compiler identification is GNU 11.3.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped

```

```
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/rupert/tmp/yaml-cpp/cmake-build
george@ubuntu:~/tmp/yaml-cpp/cmake-build$
```

**Figure 45** – Creating the yaml makefiles using cmake

Then the `make` command uses the makefiles to compile the source code and install it. The `-j2` option specifies that the process can use two threads – two parallel compilations – which can speed things up. This number can be increased but each thread needs 2 GB of memory.

```
george@ubuntu: ~
george@ubuntu:~/tmp/yaml-cpp/cmake-build$ sudo make install yaml-cpp -j2
[sudo] password for george:
[ 2%] Building CXX object CMakeFiles/yaml-cpp.dir/src/contrib/graphbuilder.cpp.o
[ 5%] Building CXX object CMakeFiles/yaml-cpp.dir/src/contrib/graphbuilderadapter.cpp.o
[ 7%] Building CXX object CMakeFiles/yaml-cpp.dir/src/binary.cpp.o
[10%] Building CXX object CMakeFiles/yaml-cpp.dir/src/convert.cpp.o
[13%] Building CXX object CMakeFiles/yaml-cpp.dir/src/depthguard.cpp.o

Some lines removed here.

[ 81%] Building CXX object CMakeFiles/yaml-cpp.dir/src/tag.cpp.o
[ 84%] Linking CXX static library libyaml-cpp.a
[ 84%] Built target yaml-cpp
[ 86%] Building CXX object util/CMakeFiles/yaml-cpp-sandbox.dir/sandbox.cpp.o
[ 89%] Building CXX object util/CMakeFiles/yaml-cpp-parse.dir/parse.cpp.o
[ 92%] Linking CXX executable parse
[ 94%] Linking CXX executable sandbox
[ 94%] Built target yaml-cpp-sandbox
[ 94%] Built target yaml-cpp-parse
[ 97%] Building CXX object util/CMakeFiles/yaml-cpp-read.dir/read.cpp.o
[100%] Linking CXX executable read
[100%] Built target yaml-cpp-read
Install the project...
-- Install configuration: ""
-- Installing: /usr/local/lib/libyaml-cpp.a
-- Up-to-date: /usr/local/include
-- Installing: /usr/local/include/yaml-cpp
-- Installing: /usr/local/include/yaml-cpp/anchor.h

Some lines removed here.

-- Installing: /usr/local/include/yaml-cpp/parser.h
-- Installing: /usr/local/include/yaml-cpp/stlemitter.h
-- Installing: /usr/local/include/yaml-cpp/traits.h
-- Installing: /usr/local/include/yaml-cpp/yaml.h
-- Installing: /usr/local/lib/cmake/yaml-cpp/yaml-cpp-targets.cmake
-- Installing: /usr/local/lib/cmake/yaml-cpp/yaml-cpp-targets-noconfig.cmake
-- Installing: /usr/local/lib/cmake/yaml-cpp/yaml-cpp-config.cmake
```

```
-- Installing: /usr/local/lib/cmake/yaml-cpp/yaml-cpp-config-version.cmake
-- Installing: /usr/local/lib/pkgconfig/yaml-cpp.pc
Consolidate compiler generated dependencies of target yaml-cpp
[100%] Built target yaml-cpp
george@ubuntu:~/tmp/yaml-cpp/cmake-build$
```

**Figure 46** – Building and installing yaml using the makefiles created previously

Now that the software is built and installed the user has the option of getting rid of the source code if there is no pressing reason to keep it. The `rm -rf` command will remove files and folders, even those that are write protected. The user should make sure they are in the right directory.

```
george@ubuntu: ~
george@ubuntu:~/tmp/yaml-cpp/cmake-build$ cd ..
george@ubuntu:~/tmp/yaml-cpp$ cd ..
george@ubuntu:~/tmp$ pwd
/home/George/tmp
george@ubuntu:~/tmp$ ls
yaml-cpp
george@ubuntu:~/tmp$ rm -rf yaml-cpp
george@ubuntu:~/tmp$
```

**Figure 47** – Removing the yaml source code files

GINAN relies on a number of the utilities provided by Boost, such as time and logging libraries. Boost is installed in a slightly different way to yaml. The Boost team offer a file containing all the source code in a highly compressed format. So rather than use git, the user can use `wget` to get the file and then `tar` to unpack it. With the `tar` command the cursor will just sit there and blink – there are no messages. At the time of writing Boost 1.82.0 is the latest version.

```
george@ubuntu: ~
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ wget -c
https://boostorg.jfrog.io/artifactory/main/release/1.82.0/source/boost_1_82_0.tar.gz
--2023-06-30 10:30:03--
https://boostorg.jfrog.io/artifactory/main/release/1.82.0/source/boost_1_82_0.tar.gz
Resolving boostorg.jfrog.io (boostorg.jfrog.io)... 54.186.43.133, 44.236.191.39,
44.240.220.66, ...
Connecting to boostorg.jfrog.io (boostorg.jfrog.io)|54.186.43.133|:443... connected.
HTTP request sent, awaiting response... 302
Location: https://jfrog-prod-usw2-shared-oregon-main.s3.amazonaws.com/aol-
boostorg/filestore/0e/0e0d4bdac2628ebff2c2a63b87514217832839d?X-Artifactory-
username=anonymous&X-Artifactory-repoType=local&X-Artifactory-repositoryKey=main&X-
Artifactory-packageType=generic&X-Artifactory-
artifactPath=release%2F1.82.0%2Fsource%2Fboost_1_82_0.tar.gz&X-Artifactory-
projectKey=default&x-jf-traceId=3e7fa21cae1a2f30&response-content-
disposition=attachment%3Bfilename%3D%22boost_1_82_0.tar.gz%22&response-content-
type=application%2Fgzip&X-Amz-Security-
```

Token=IQoJb3JpZ2luX2VjEDAAcXVzLXdlc3QtMiJHMEUCIA2Tbo8P3K9j0Ihf%2B2cwpUfNqJ8WqXSFXW5SI5mjJRgZAIEAmX5M9LjX0S5fFSX5vt8UBxQ1RUPJuDLdyI7RMl9wRV0sqjqwUImf%2F%2F%2F%2F%2F%2F%2F%2F%2F

Some lines removed here.

```

DZoI4zB%2FML6x%2BKQGOpBTPjMBv8RK4H4%2F6hgFVERxgi0PqPZQsZSkrAQbd6iATAlCRrnyo7XqMU80gRp4mh1
UYVUKFPQnFlW7%2B7oGvJNAI1XlBSq1Yd1ciYq%2FXC%2BzCeMr3o9ZiY7oC%2FFFudf90TSgmWMpkB%2BRxqQvvuQ
T5CHKm9zw40xe2Tyw1%2FUBJQ4Cm7HI5Z78TtIFHp70yibFYuWRgKS41ViG1PwSg%3D%3D&X-Amz-
Algorithm=AWS4-HMAC-SHA256&X-Amz-Date=20230630T003004Z&X-Amz-SignedHeaders=host&X-Amz-
Expires=30&X-Amz-Credential=ASIASG3IHPL6UWLOBDE0%2F20230630%2Fus-west-
2%2Fs3%2Faws4_request&X-Amz-
Signature=3b97f78dea27d1e45d698cca626baeb96bffd0a5bafd94aa90d893558f5c37a
Resolving jfrog-prod-usw2-shared-oregon-main.s3.amazonaws.com (jfrog-prod-usw2-shared-
oregon-main.s3.amazonaws.com)... 52.92.149.153, 52.92.144.89, 52.218.244.234, ...
Connecting to jfrog-prod-usw2-shared-oregon-main.s3.amazonaws.com (jfrog-prod-usw2-shared-
oregon-main.s3.amazonaws.com)|52.92.149.153|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 142580547 (136M) [application/x-gzip]
Saving to: 'boost_1_82_0.tar.gz'

```

```
boost_1_82_0.tar.gz
100%[=====>] 135.97M  615KB/s  in 4m
12s
```

```
2023-06-30 10:34:18 (552 KB/s) - 'boost_1_82_0.tar.gz' saved [142580547/142580547]
```

```
george@ubuntu:~/tmp$ tar -xf boost_1_82_0.tar.gz
george@ubuntu:~/tmp$ ls
boost_1_82_0  boost_1_82_0.tar.gz
george@ubuntu:~/tmp$
```

**Figure 48** – Getting and extracting the Boost files

The user moves into the `boost_1_82_0` directory and types in the `bootstrap.sh` command. That command that will start the bootstrap script present inside the Boost extracted folder. It will start building a b2 engine. b2 is an executable written in C which runs the Boost build system. The install command finishes the job. Be aware there are many, many files – over 18,000 – that get compiled. The process goes on for many minutes.

```

george@ubuntu: ~
george@ubuntu:~/tmp$ cd boost_1_82_0
george@ubuntu:~/tmp/boost_1_82_0$ ./bootstrap.sh
Building B2 engine..

###
###
### Using 'gcc' toolset.
###
###

g++ (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
Copyright (C) 2021 Free Software Foundation, Inc.

```

This is free software; see the source for copying conditions. There is NO warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

###  
###

```
> g++ -x c++ -std=c++11 -O2 -s -DNDEBUG builtins.cpp class.cpp command.cpp compile.cpp
constants.cpp cwd.cpp debug.cpp debugger.cpp execcmd.cpp execnt.cpp execunix.cpp
filesystem.cpp filent.cpp fileunix.cpp frames.cpp function.cpp glob.cpp hash.cpp hcache.cpp
hdrmacro.cpp headers.cpp jam_strings.cpp jam.cpp jamgram.cpp lists.cpp make.cpp make1.cpp
md5.cpp mem.cpp modules.cpp native.cpp object.cpp option.cpp output.cpp parse.cpp
pathnt.cpp pathsys.cpp pathunix.cpp regexp.cpp rules.cpp scan.cpp search.cpp startup.cpp
subst.cpp sysinfo.cpp timestamp.cpp variable.cpp w32_getreg.cpp modules/order.cpp
modules/path.cpp modules/property-set.cpp modules/regex.cpp modules/sequence.cpp
modules/set.cpp -o b2
tools/build/src/engine/b2
Unicode/ICU support for Boost.Regex?... not found.
Generating B2 configuration in project-config.jam for gcc...
```

Bootstrapping is done. To build, run:

```
./b2
```

To generate header files, run:

```
./b2 headers
```

The configuration generated uses gcc to build by default. If that is unintended either use the `--with-toolset` option or adjust configuration, by editing 'project-config.jam'.

Further information:

- Command line help:  
./b2 --help
- Getting started guide:  
[http://www.boost.org/more/getting\\_started/unix-variants.html](http://www.boost.org/more/getting_started/unix-variants.html)
- B2 documentation:  
<http://www.boost.org/build/>

```
george@ubuntu:~/tmp/boost_1_82_0$ sudo ./b2 -j2 install
```

Thousands of lines removed here.

```
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/clocale
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/climits
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/cfloat
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/cerrno
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/cctype
common.copy /usr/local/include/boost/compatibility/cpp_c_headers/cassert
boost-install.generate-cmake-config- bin.v2/libs/headers/build/install/boost_headers-
config.cmake
```



```

boost-install.generate-cmake-config-version-
bin.v2/libs/headers/build/install/boost_headers-config-version.cmake
common.copy /usr/local/lib/cmake/boost_headers-1.82.0/boost_headers-config.cmake
common.copy /usr/local/lib/cmake/boost_headers-1.82.0/boost_headers-config-version.cmake
boost-install.generate-cmake-config-version-
bin.v2/tools/boost_install/BoostConfigVersion.cmake
common.copy /usr/local/lib/libboost_chrono.so.1.82.0
ln-UNIX /usr/local/lib/libboost_chrono.so
common.copy /usr/local/lib/cmake/Boost-1.82.0/BoostConfigVersion.cmake
boost-install.generate-cmake-variant- bin.v2/libs/chrono/build/gcc-11/release/threading-
multi/visibility-hidden/libboost_chrono-variant-shared.cmake
boost-install.generate-cmake-config- bin.v2/libs/chrono/build/install/boost_chrono-
config.cmake
boost-install.generate-cmake-config-version-
bin.v2/libs/chrono/build/install/boost_chrono-config-version.cmake
...on 18600th target...
common.copy /usr/local/lib/cmake/boost_chrono-1.82.0/boost_chrono-config.cmake
common.copy /usr/local/lib/cmake/boost_chrono-1.82.0/boost_chrono-config-version.cmake
common.copy /usr/local/lib/cmake/boost_chrono-1.82.0/libboost_chrono-variant-shared.cmake
...updated 18602 targets...
george@ubuntu:~/tmp/boost_1_82_0$

```

**Figure 49** – Compiling the Boost files

As before, if the user has no need for the source code or the tar file they can simply be removed.

```

george@ubuntu: ~
george@ubuntu:~/tmp/boost_1_82_0$ cd ..
george@ubuntu:~/tmp$ pwd
/home/George/tmp
george@ubuntu:~/tmp$ ls
boost_1_82_0  boost_1_82_0.tar.gz
george@ubuntu:~/tmp$ rm -rf boost_1_82_0 boost_1_82_0.tar.gz
george@ubuntu:~/tmp$

```

**Figure 50** – Removing the Boost source code files and tar file

Eigen is installed in much the same way as yaml. While yaml is hosted on GitHub, Eigen is hosted on GitLab. They are different organisations but both are based on git. Eigen requires a clone and download, the creation of makefiles, and then a build and install.

```

george@ubuntu: ~
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ git clone https://gitlab.com/libeigen/eigen.git
Cloning into 'eigen'...
remote: Enumerating objects: 119648, done.
remote: Counting objects: 100% (1327/1327), done.
remote: Compressing objects: 100% (439/439), done.
remote: Total 119648 (delta 943), reused 1260 (delta 887), pack-reused 118321

```



```
Receiving objects: 100% (119648/119648), 103.70 MiB | 620.00 KiB/s, done.
Resolving deltas: 100% (98778/98778), done.
Updating files: 100% (1856/1856), done.
```

```
george@ubuntu:~/tmp$ ls
```

```
eigen
```

```
george@ubuntu:~/tmp$ cd eigen
```

```
george@ubuntu:~/tmp/eigen$ git checkout 3.4.0
```

```
Note: switching to '3.4.0'.
```

You are in 'detached HEAD' state. You can look around, make experimental changes and commit them, and you can discard any commits you make in this state without impacting any branches by switching back to a branch.

If you want to create a new branch to retain commits you create, you may do so (now or later) by using -c with the switch command. Example:

```
git switch -c <new-branch-name>
```

Or undo this operation with:

```
git switch -
```

Turn off this advice by setting config variable advice.detachedHead to false

```
HEAD is now at 3147391d9 Change version to 3.4.0.
```

```
george@ubuntu:~/tmp/eigen$ mkdir cmake-build
```

```
george@ubuntu:~/tmp/eigen$ cd cmake-build
```

```
george@ubuntu:~/tmp/eigen/cmake-build$ cmake ..
```

```
-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Performing Test EIGEN_COMPILER_SUPPORT_CPP11
```

Many lines removed here.

```
-- Performing Test COMPILER_SUPPORT_ansi - Success
-- Performing Test COMPILER_SUPPORT_OPENMP
-- Performing Test COMPILER_SUPPORT_OPENMP - Success
-- Found unsuitable Qt version "" from NOTFOUND
-- Looking for a Fortran compiler
-- Looking for a Fortran compiler - /usr/bin/f95
-- The Fortran compiler identification is GNU 11.3.0
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Check for working Fortran compiler: /usr/bin/f95 - skipped
-- Could NOT find CHOLMOD (missing: CHOLMOD_INCLUDES CHOLMOD_LIBRARIES)
```

Many lines removed here.

```
-- Could NOT find UMFPACK (missing: UMFPACK_INCLUDES UMFPACK_LIBRARIES)
-- Could NOT find KLU (missing: KLU_INCLUDES KLU_LIBRARIES)
-- Could NOT find SuperLU (missing: SUPERLU_INCLUDES SUPERLU_LIBRARIES SUPERLU_VERSION_OK)
(Required is at least version "4.0")
-- A version of Pastix has been found but pastix_nomp.h does not exist in the include
directory. Because Eigen tests require a version without MPI, we disable the Pastix
backend.
-- *****
-- ***      Eigen's unit tests configuration summary      ***
-- *****
--
-- Build type:           Release
-- Build site:           FSI-LT-12
-- Build string:         linux-4.4.0-19041-microsoft--11.3.0-sse2-64bit
-- Enabled backends:     Boost.Multiprecision,
-- Disabled backends:    CHOLMOD, UMFPACK, KLU, SuperLU, PaStiX, METIS, SPQR, Qt4
support, GoogleHash, Adolc, MPFR C++, fftw, OpenGL,
-- Default order:       Column-major
-- Maximal matrix/vector size: 320
-- SSE2:                 Using architecture defaults
-- SSE3:                 Using architecture defaults
-- SSSE3:                Using architecture defaults
-- SSE4.1:               Using architecture defaults
-- SSE4.2:               Using architecture defaults
-- AVX:                  Using architecture defaults
-- AVX2:                 Using architecture defaults
-- FMA:                  Using architecture defaults
-- AVX512:               Using architecture defaults
-- AVX512DQ:             Using architecture defaults
-- AltiVec:              Using architecture defaults
-- VSX:                  Using architecture defaults
-- MIPS MSA:             Using architecture defaults
-- ARM NEON:             Using architecture defaults
-- ARMv8 NEON:           Using architecture defaults
-- S390X ZVECTOR:        Using architecture defaults
-- C++11:                OFF
-- SYCL:                 OFF
-- CUDA:                 OFF
-- HIP:                  OFF
--
CXX:                     /usr/bin/c++
CXX_VERSION:             c++ (Ubuntu 11.3.0-1ubuntu1~22.04.1) 11.3.0
CXX_FLAGS:               -std=c++03 -pedantic -Wall -Wextra -Wundef -Wcast-align -Wchar-
subscripts -Wnon-virtual-dtor -Wunused-local-typedefs -Wpointer-arith -Wwrite-strings -
Wformat-security -Wlogical-op -Wenum-conversion -Wdouble-promotion -Wshadow -Wno-psabi -
Wno-variadic-macros -Wno-long-long -fno-check-new -fno-common -fstrict-aliasing -ansi
Sparse lib flags:
-- *****
--
-- Configured Eigen 3.4.0
--
-- Available targets (use: make TARGET):
-- +-----
```

```

-- Target | Description
-----+-----
-- install | Install Eigen. Headers will be installed to:
--           | <CMAKE_INSTALL_PREFIX>/<INCLUDE_INSTALL_DIR>
--           | Using the following values:
--           |   CMAKE_INSTALL_PREFIX: /usr/local
--           |   INCLUDE_INSTALL_DIR: include/eigen3
--           | Change the install location of Eigen headers using:
--           |   cmake . -DCMAKE_INSTALL_PREFIX=yourprefix
--           | Or:
--           |   cmake . -DINCLUDE_INSTALL_DIR=yourdir
-- doc      | Generate the API documentation, requires Doxygen & LaTeX
-- check    | Build and run the unit-tests. Read this page:
--           | http://eigen.tuxfamily.org/index.php?title=Tests
-- blas     | Build BLAS library (not the same thing as Eigen)
-- uninstall| Remove files installed by the install target
-- -----+-----
--
-- Configuring done
-- Generating done
-- Build files have been written to: /home/rupert/tmp/eigen/cmake-build
george@ubuntu:~/tmp/eigen/cmake-build$ sudo make -j2 install
[sudo] password for george:
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/include/eigen3/signature_of_eigen3_matrix_library
-- Installing: /usr/local/share/pkgconfig/eigen3.pc
-- Installing: /usr/local/include/eigen3/Eigen
-- Installing: /usr/local/include/eigen3/Eigen/Cholesky

Many lines removed here.

-- Installing: /usr/local/include/eigen3/unsupported/Eigen/CXX11/src/util/CXX11Meta.h
-- Installing:
/usr/local/include/eigen3/unsupported/Eigen/CXX11/src/util/CXX11Workarounds.h
-- Installing: /usr/local/include/eigen3/unsupported/Eigen/CXX11/src/util/EmulateArray.h
-- Installing: /usr/local/include/eigen3/unsupported/Eigen/CXX11/src/util/MaxSizeVector.h
george@ubuntu:~/tmp/eigen/cmake-build$

```

**Figure 51** – cloning and installing Eigen from GitLab

NetCDF4 has a deb package and so can be installed in the usual way:

```

george@ubuntu: ~
george@ubuntu:~$ sudo apt install libnetcdf-dev libnetcdf-c++4-dev
[sudo] password for george:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  hdf5-helpers libaec-dev libaec0 libcurl4-openssl-dev libdpkg-perl libfile-fcntllock-perl
  libhdf5-103-1 libhdf5-cpp-103-1

```

```

libhdf5-dev libhdf5-fortran-102 libhdf5-hl-100 libhdf5-hl-cpp-100 libhdf5-hl-fortran-100
libjpeg-dev libjpeg-turbo8-dev
libjpeg8-dev libnetcdf-c++4-1 libnetcdf19 libssl-dev libsz2 pkg-config zlib1g-dev
Suggested packages:
libcurl4-doc libidn11-dev libkrb5-dev libldap2-dev librtmp-dev libssh2-1-dev debian-
keyring bzip2 libhdf5-doc netcdf-bin
netcdf-doc libssl-doc dpkg-dev
The following NEW packages will be installed:
hdf5-helpers libaec-dev libaec0 libcurl4-openssl-dev libdpkg-perl libfile-fcntllock-perl
libhdf5-103-1 libhdf5-cpp-103-1
libhdf5-dev libhdf5-fortran-102 libhdf5-hl-100 libhdf5-hl-cpp-100 libhdf5-hl-fortran-100
libjpeg-dev libjpeg-turbo8-dev
libjpeg8-dev libnetcdf-c++4-1 libnetcdf-c++4-dev libnetcdf-dev libnetcdf19 libssl-dev
libsz2 pkg-config zlib1g-dev
0 upgraded, 24 newly installed, 0 to remove and 0 not upgraded.
Need to get 8555 kB of archives.
After this operation, 39.3 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://archive.ubuntu.com/ubuntu jammy/universe amd64 hdf5-helpers amd64
1.10.7+repack-4ubuntu2 [14.2 kB]
Get:2 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libaec0 amd64 1.0.6-1 [20.1
kB]

```

Many lines removed here.

```

Get:23 http://archive.ubuntu.com/ubuntu jammy/universe amd64 libnetcdf-c++4-dev amd64
4.3.1-3build1 [110 kB]
Get:24 http://archive.ubuntu.com/ubuntu jammy/main amd64 pkg-config amd64 0.29.2-1ubuntu3
[48.2 kB]
Fetched 8555 kB in 21s (404 kB/s)
Selecting previously unselected package hdf5-helpers.
(Reading database ... 42637 files and directories currently installed.)
Preparing to unpack .../00-hdf5-helpers_1.10.7+repack-4ubuntu2_amd64.deb ...
Unpacking hdf5-helpers (1.10.7+repack-4ubuntu2) ...

```

Many lines removed here.

```

Selecting previously unselected package libnetcdf-c++4-dev.
Preparing to unpack .../22-libnetcdf-c++4-dev_4.3.1-3build1_amd64.deb ...
Unpacking libnetcdf-c++4-dev (4.3.1-3build1) ...
Selecting previously unselected package pkg-config.
Preparing to unpack .../23-pkg-config_0.29.2-1ubuntu3_amd64.deb ...
Unpacking pkg-config (0.29.2-1ubuntu3) ...
Setting up libjpeg-turbo8-dev:amd64 (2.1.2-0ubuntu1) ...
Setting up hdf5-helpers (1.10.7+repack-4ubuntu2) ...
Setting up libfile-fcntllock-perl (0.22-3build7) ...

```

Many lines removed here.

```

Setting up libhdf5-hl-fortran-100:amd64 (1.10.7+repack-4ubuntu2) ...
Setting up libnetcdf-c++4-1 (4.3.1-3build1) ...
Setting up libhdf5-dev (1.10.7+repack-4ubuntu2) ...
update-alternatives: using /usr/lib/x86_64-linux-gnu/pkgconfig/hdf5-serial.pc to provide
/usr/lib/x86_64-linux-gnu/pkgconfig/hdf5.pc (hdf5.pc) in auto mode
Setting up libnetcdf-dev (1:4.8.1-1) ...
Setting up libnetcdf-c++4-dev (4.3.1-3build1) ...

```

```
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.1) ...
george@ubuntu:~$
```

Figure 52 – Installing netCDF4

## 5.6 Mongo

Ginan can output data in a variety of industry standard file types and its own TRACE files. For some users that may be enough. But for users who want to capture large quantities of data and analyse that data – say for research purposes – Ginan comes with the ability to write data to a Mongo database. The Ginan team have also released the Exploratory Data Analysis (EDA) tool which works with Mongo and can be used to visualise data produced by Ginan.

To use Mongo, Ginan needs access to the MongoDB C and C++ drivers. Even if the user doesn't feel the need to use Mongo, the drivers are still required – they are used for json formatting and other self-descriptive markup. The C and C++ drivers are downloaded as compressed files and built in a similar way to Boost.

```
george@ubuntu: ~
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ wget https://github.com/mongodb/mongo-c-
driver/releases/download/1.24.1/mongo-c-driver-1.24.1.tar.gz
--2023-07-03 14:40:39-- https://github.com/mongodb/mongo-c-
driver/releases/download/1.24.1/mongo-c-driver-1.24.1.tar.gz
Resolving github.com (github.com)... 20.248.137.48
Connecting to github.com (github.com)|20.248.137.48|:443... connected.
HTTP request sent, awaiting response... 302 Found

Some lines removed here.

Connecting to objects.githubusercontent.com
(objects.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8256773 (7.9M) [application/octet-stream]
Saving to: 'mongo-c-driver-1.24.1.tar.gz'

mongo-c-driver-1.24.1.tar.gz 100%[=====>]
7.87M 653KB/s in 13s

2023-07-03 14:40:54 (635 KB/s) - 'mongo-c-driver-1.24.1.tar.gz' saved [8256773/8256773]

george@ubuntu:~/tmp$ tar -xf mongo-c-driver-1.24.1.tar.gz
george@ubuntu:~/tmp$ cd mongo-c-driver-1.24.1
george@ubuntu:~/tmp/mongo-c-driver-1.24.1$ mkdir cmake-build
george@ubuntu:~/tmp/mongo-c-driver-1.24.1$ cd cmake-build
george@ubuntu:~/tmp/mongo-c-driver-1.24.1/cmake-build$ cmake -
DENABLE_AUTOMATIC_INIT_AND_CLEANUP=OFF -DENABLE_EXAMPLES=OFF ../
-- The C compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
```

Some lines removed here.

```
-- Performing Test MONGOC_HAVE_SS_FAMILY
-- Performing Test MONGOC_HAVE_SS_FAMILY - Success
-- Compiling against OpenSSL
-- SASL disabled
-- Adding -fPIC to compilation of mongoc_static components
-- Building with MONGODB-AWS auth support
-- Build files generated for:
--   build system: Unix Makefiles
-- Configuring done
-- Generating done
-- Build files have been written to: /home/rupe/tmp/mongo-c-driver-1.24.1/cmake-build
george@ubuntu:~/tmp/mongo-c-driver-1.24.1/cmake-build$ cmake --build . -- -j 2
[ 0%] Building C object src/libbson/CMakeFiles/bson_shared.dir/src/bson/bcon.c.o
[ 0%] Building C object src/libbson/CMakeFiles/bson_static.dir/src/bson/bcon.c.o
[ 0%] Building C object src/libbson/CMakeFiles/bson_static.dir/src/bson/bson.c.o
[ 0%] Building C object src/libbson/CMakeFiles/bson_shared.dir/src/bson/bson.c.o
[ 1%] Building C object src/libbson/CMakeFiles/bson_static.dir/src/bson/bson-atomic.c.o
[ 1%] Building C object src/libbson/CMakeFiles/bson_shared.dir/src/bson/bson-atomic.c.o
[ 1%] Building C object src/libbson/CMakeFiles/bson_static.dir/src/bson/bson-clock.c.o
[ 1%] Building C object src/libbson/CMakeFiles/bson_shared.dir/src/bson/bson-clock.c.o
[ 1%] Building C object src/libbson/CMakeFiles/bson_static.dir/src/bson/bson-context.c.o
```

Many lines removed here.

```
[ 98%] Built target test-libmongoc-lib
[ 98%] Building C object src/libmongoc/CMakeFiles/test-libmongoc.dir/tests/test-libmongoc-
main.c.o
[ 99%] Building C object src/libmongoc/CMakeFiles/test-atlas-executor.dir/tests/test-
atlas-executor.c.o
[100%] Linking C executable test-libmongoc
[100%] Linking C executable test-atlas-executor
[100%] Built target test-atlas-executor
[100%] Built target test-libmongoc
george@ubuntu:~/tmp/mongo-c-driver-1.24.1/cmake-build$ sudo cmake --build . --target
install -- -j 2
[sudo] password for george:
Consolidate compiler generated dependencies of target bson_shared
Consolidate compiler generated dependencies of target bson_static
[ 8%] Built target bson_shared
[ 11%] Built target bson_static
Consolidate compiler generated dependencies of target zlib_obj
[ 15%] Built target zlib_obj
```

Many lines removed here.

```
-- Installing: /usr/local/lib/cmake/libmongoc-static-1.0/libmongoc-static-1.0-config.cmake
-- Installing: /usr/local/lib/cmake/libmongoc-static-1.0/libmongoc-static-1.0-config-
version.cmake
-- Installing: /usr/local/share/mongo-c-driver/COPYING
-- Installing: /usr/local/share/mongo-c-driver/NEWS
-- Installing: /usr/local/share/mongo-c-driver/README.rst
-- Installing: /usr/local/share/mongo-c-driver/THIRD_PARTY_NOTICES
-- Installing: /usr/local/share/mongo-c-driver/uninstall.sh
george@ubuntu:~/tmp/mongo-c-driver-1.24.1/cmake-build$
```

**Figure 53** – installing MongoDB C drivers

The process is very similar for the C++ drivers but `curl` is used rather than `wget`.

```

george@ubuntu: ~
george@ubuntu:~$ cd tmp
george@ubuntu:~/tmp$ curl -OL https://github.com/mongodb/mongo-cxx-
driver/releases/download/r3.8.0/mongo-cxx-driver-r3.8.0.tar.gz
.8.0.tar.gz
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left   Speed
  0     0    0     0    0     0      0      0  --:--:-- --:--:-- --:--:--    0
100 1776k 100 1776k    0     0 362k      0  0:00:04 0:00:04 --:--:-- 588k
george@ubuntu:~/tmp$ tar -xf mongo-cxx-driver-r3.8.0.tar.gz
george@ubuntu:~/tmp$ cd mongo-cxx-driver-r3.8.0/build
george@ubuntu:~/tmp/mongo-cxx-driver-r3.8.0/build$ cmake -DCMAKE_BUILD_TYPE=Release -
DCMAKE_INSTALL_PREFIX=/usr/local -DENABLE_EXAMPLES=OFF ../
-DENABLE_EXAMPLES=OFF ../
-- The CXX compiler identification is GNU 11.3.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done

Some lines removed here.

-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Build files generated for:
--   build system: Unix Makefiles
-- Configuring done
-- Generating done
CMake Warning:
  Manually-specified variables were not used by the project:

    ENABLE_EXAMPLES

-- Build files have been written to: /tmp/mongo-cxx-driver-r3.8.0/build
george@ubuntu:~/tmp/mongo-cxx-driver-r3.8.0/build$ sudo cmake --build . --target
EP_mnmlstc_core -- -j 2
[sudo] password for george:
[ 0%] Creating directories for 'EP_mnmlstc_core'
[ 0%] Performing download step (git clone) for 'EP_mnmlstc_core'
-- EP_mnmlstc_core download command succeeded. See also /tmp/mongo-cxx-driver-
r3.8.0/build/src/bsoncxx/third_party/EP_mnmlstc_core-prefix/src/EP_mnmlstc_core-
stamp/EP_mnmlstc_core-download-*.log
[ 0%] No update step for 'EP_mnmlstc_core'
[ 0%] No patch step for 'EP_mnmlstc_core'
[ 0%] Performing configure step for 'EP_mnmlstc_core'

```



```

-- EP_mnmlstc_core configure command succeeded. See also /tmp/mongo-cxx-driver-
r3.8.0/build/src/bsoncxx/third_party/EP_mnmlstc_core-prefix/src/EP_mnmlstc_core-
stamp/EP_mnmlstc_core-configure-*.log
[ 0%] Performing build step for 'EP_mnmlstc_core'
-- EP_mnmlstc_core build command succeeded. See also /tmp/mongo-cxx-driver-
r3.8.0/build/src/bsoncxx/third_party/EP_mnmlstc_core-prefix/src/EP_mnmlstc_core-
stamp/EP_mnmlstc_core-build-*.log
[ 0%] Performing install step for 'EP_mnmlstc_core'
-- EP_mnmlstc_core install command succeeded. See also /tmp/mongo-cxx-driver-
r3.8.0/build/src/bsoncxx/third_party/EP_mnmlstc_core-prefix/src/EP_mnmlstc_core-
stamp/EP_mnmlstc_core-install-*.log
[100%] Performing fix-includes step for 'EP_mnmlstc_core'
[100%] Completed 'EP_mnmlstc_core'
[100%] Built target EP_mnmlstc_core
george@ubuntu:~/tmp/mongo-cxx-driver-r3.8.0/build$ cmake --build . -- -j 2
[ 1%] Built target EP_mnmlstc_core
[ 1%] Building CXX object src/bsoncxx/CMakeFiles/bsoncxx_shared.dir/array/element.cpp.o
[ 1%] Building CXX object src/bsoncxx/CMakeFiles/bsoncxx_testing.dir/array/element.cpp.o
[ 2%] Building CXX object src/bsoncxx/CMakeFiles/bsoncxx_shared.dir/array/value.cpp.o

Many lines removed here.
Process takes quite a while.

[ 98%] Built target test_unified_format_spec
[ 98%] Building CXX object
src/mongocxx/test/CMakeFiles/test_versioned_api.dir/spec/unified_tests/assert.cpp.o
[100%] Building CXX object
src/mongocxx/test/CMakeFiles/test_versioned_api.dir/spec/monitoring.cpp.o
[100%] Building CXX object
src/mongocxx/test/CMakeFiles/test_versioned_api.dir/spec/util.cpp.o
[100%] Linking CXX executable test_versioned_api
[100%] Built target test_versioned_api
george@ubuntu:~/tmp/mongo-cxx-driver-r3.8.0/build$ sudo cmake --build . --target install
[sudo] password for rupert:
[ 1%] Built target EP_mnmlstc_core
Consolidate compiler generated dependencies of target bsoncxx_shared
[ 4%] Built target bsoncxx_shared
Consolidate compiler generated dependencies of target bsoncxx_testing
[ 9%] Built target bsoncxx_testing
Consolidate compiler generated dependencies of target test_bson

Some lines removed here.

[ 97%] Built target test_unified_format_spec
Consolidate compiler generated dependencies of target test_versioned_api
[100%] Built target test_versioned_api
Install the project...
-- Install configuration: "Release"
-- Installing: /usr/local/share/mongo-cxx-driver/LICENSE
-- Installing: /usr/local/share/mongo-cxx-driver/README.md
-- Installing: /usr/local/share/mongo-cxx-driver/THIRD-PARTY-NOTICES

Some lines removed here.

-- Installing: /usr/local/include/mongocxx/v_noabi/mongocxx/config/config.hpp
-- Installing: /usr/local/include/mongocxx/v_noabi/mongocxx/config/version.hpp

```



```
-- Installing: /usr/local/lib/pkgconfig/libmongocxx.pc
-- Installing: /usr/local/share/mongo-cxx-driver/uninstall.sh
george@ubuntu:~/tmp/mongo-cxx-driver-r3.8.0/build$ cd $dir/tmp
george@ubuntu:~/tmp$
```

**Figure 54** – Installing MongoDB C++ drivers

As before, if the user has no need for the source code or the tar file they can simply be removed.

```
george@ubuntu: ~
george@ubuntu:~/tmp$ pwd
/home/George/tmp
george@ubuntu:~/tmp$ sudo rm -rf mongo-c-driver-1.24.1 mongo-c-driver-1.24.1.tar.gz
mongo-cxx-driver-r3.8.0 mongo-cxx-driver-r3.8.0.tar.gz
george@ubuntu:~/tmp$
```

**Figure 55** – Removing the Mongo drivers' source code files and tar file

The instructions for installing and using MongoDB are here: <https://github.com/GeoscienceAustralia/ginan#mongodb-pea-optional>

## 5.7 Ginan source

All that activity gets the user to the point where they can download the Ginan source from GitHub and then use a Python script to download example data and yaml files – in a similar way as was done before for the Docker image. pip3 should be installed as part of Ubuntu. If it turns out it isn't, the user can type the command `sudo apt install python3-pip`

```
george@ubuntu: ~
george@ubuntu:~$ pwd
/home/George
george@ubuntu:~/tmp$ git clone https://github.com/GeoscienceAustralia/ginan.git
Cloning into 'ginan'...
remote: Enumerating objects: 11450, done.
remote: Counting objects: 100% (5920/5920), done.
remote: Compressing objects: 100% (1817/1817), done.
remote: Total 11450 (delta 4353), reused 4821 (delta 4081), pack-reused 5530
Receiving objects: 100% (11450/11450), 73.10 MiB | 589.00 KiB/s, done.
Resolving deltas: 100% (5850/5850), done.
george@ubuntu:~$
george@ubuntu:~$ ls
ginan
george@ubuntu:~$ cd ginan
george@ubuntu:~/ginan$ ls
CHANGELOG.md Docs ISSUES.md LICENSE.md README.md archived deleteExamples docker
exampleConfigs scripts src
george@ubuntu:~/ginan$ sudo pip3 install gnssanalysis
Collecting gnssanalysis
  Downloading gnssanalysis-0.0.36.tar.gz (112 kB)
```

```
Preparing metadata (setup.py) ... done
Collecting boto3
  Downloading boto3-1.26.165-py3-none-any.whl (135 kB)
    _____ 135.9/135.9 KB 557.3 kB/s eta 0:00:00

Some lines removed here.

Collecting urllib3<1.27,>=1.25.4
  Downloading urllib3-1.26.16-py2.py3-none-any.whl (143 kB)
    _____ 143.1/143.1 KB 633.7 kB/s eta 0:00:00

Collecting pycparser
  Downloading pycparser-2.21-py2.py3-none-any.whl (118 kB)
    _____ 118.7/118.7 KB 658.2 kB/s eta 0:00:00

Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.7->matplotlib>gnssanalysis) (1.16.0)
Building wheels for collected packages: gnssanalysis, unlzw
  Building wheel for gnssanalysis (setup.py)... done
  Created wheel for gnssanalysis: filename=gnssanalysis-0.0.36-py3-none-any.whl size=106243 sha256=38905b07d11ce4bb82202a94ea88b6f143f72c39819677515bd33189cd72f654
  Stored in directory:
/root/.cache/pip/wheels/b1/1b/ea/26a181f4e03db377364c1bffe1094acff202ebaed2303cdede
  Building wheel for unlzw (setup.py) ... done
  Created wheel for unlzw: filename=unlzw-0.1.1-cp310-cp310-linux_x86_64.whl size=19791 sha256=8d640321405ebd3ea4478857206bee2d138bfcf86de1f239e0c2d2cc663c999e
  Stored in directory:
/root/.cache/pip/wheels/6c/13/fe/99e9d25dd1320c89c68470437ffbfa1b093758f851b3c616e92
Successfully built gnssanalysis unlzw
Installing collected packages: pytz, urllib3, tzdata, tqdm, tomli, tenacity, python-dateutil, pycparser, pluggy, plotext, pillow, packaging, numpy, kiwisolver, jmespath, iniconfig, fonttools, exceptiongroup, dnspython, cyclical, click, scipy, pytest, pymongo, plotly, pandas, contourpy, cffi, botocore, unlzw, s3transfer, matplotlib, boto3, gnssanalysis
Successfully installed boto3-1.26.165 botocore-1.29.165 cffi-1.15.1 click-8.1.3 contourpy-1.1.0 cyclical-0.11.0 dnspython-2.3.0 exceptiongroup-1.1.1 fonttools-4.40.0 gnssanalysis-0.0.36 iniconfig-2.0.0 jmespath-1.0.1 kiwisolver-1.4.4 matplotlib-3.7.1 numpy-1.25.0 packaging-23.1 pandas-2.0.3 pillow-10.0.0 plotext-4.2.0 plotly-5.15.0 pluggy-1.2.0 pycparser-2.21 pymongo-4.4.0 pytest-7.4.0 python-dateutil-2.8.2 pytz-2023.3 s3transfer-0.6.1 scipy-1.11.1 tenacity-8.2.2 tomli-2.0.1 tqdm-4.65.0 tzdata-2023.3 unlzw-0.1.1 urllib3-1.26.16
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: https://pip.pypa.io/warnings/venv
george@ubuntu:~/ginan$ python3 scripts/download_example_input_data.py
INFO:root:default input path relative to script location selected:
/home/rupert/ginan/inputData
INFO:root:default output path relative to script location selected: .
INFO:root:(('products', 'data')) selected
INFO:root:reading tags from /home/rupert/ginan/docker/tags
INFO:root:requesting checksum for "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/products.tar.bz2"
INFO:root:server says OK
INFO:root:Got "KvpDGu9PATZ5nr02uqzvzvw=="
INFO:root:products.tar.bz2 not found on disk ['KvpDGu9PATZ5nr02uqzvzvw=='].
INFO:root:requesting "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/products.tar.bz2"
```

```

INFO:root:downloading from https://peanpod.s3.ap-southeast-
2.amazonaws.com/aux/products.tar.bz2 to /home/rupert/ginan/inputData/products.tar.bz2
INFO:root:Extracting /home/rupert/ginan/inputData/products.tar.bz2 to
/home/rupert/ginan/inputData
INFO:root:requesting checksum for "https://peanpod.s3.ap-southeast-
2.amazonaws.com/aux/data.tar.bz2"
INFO:root:server says OK
INFO:root:Got "Hojhe4sTUBke2g+zTfDEkw=="
INFO:root:data.tar.bz2 not found on disk ['Hojhe4sTUBke2g+zTfDEkw=='].
INFO:root:requesting "https://peanpod.s3.ap-southeast-2.amazonaws.com/aux/data.tar.bz2"
INFO:root:downloading from https://peanpod.s3.ap-southeast-
2.amazonaws.com/aux/data.tar.bz2 to /home/rupert/ginan/inputData/data.tar.bz2
INFO:root:Extracting /home/rupert/ginan/inputData/data.tar.bz2 to
/home/rupert/ginan/inputData
george@ubuntu:~/ginan$

```

**Figure 56** – Getting the Ginan source code and downloading example resources using one of the Python scripts.

Now everything should be in place to build the Ginan source. The user may see warning messages during the build process. So long as they are warning messages – and not errors – things are fine.

```

george@ubuntu: ~
george@ubuntu:~$ pwd
/home/George
george@ubuntu:~$ ls
ginan  tmp
george@ubuntu:~$ cd ginan
george@ubuntu:~/ginan$ mkdir src/build
george@ubuntu:~/ginan$ cd src
george@ubuntu:~/ginan/src$ ls
Architecture  CMakeLists.txt  build  cmake  cpp  doc_templates  fortran
george@ubuntu:~/ginan/src$ cd build
george@ubuntu:~/ginan/src/build$ cmake -DOPTIMISATION=FALSE ../
-- The C compiler identification is GNU 11.3.0
-- The CXX compiler identification is GNU 11.3.0
-- Detecting C compiler ABI info
-- Detecting C compiler ABI info - done
-- Check for working C compiler: /usr/bin/cc - skipped
-- Detecting C compile features
-- Detecting C compile features - done
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: /usr/bin/c++ - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- The Fortran compiler identification is GNU 11.3.0
-- Detecting Fortran compiler ABI info
-- Detecting Fortran compiler ABI info - done
-- Check for working Fortran compiler: /usr/bin/gfortran - skipped
-- Setting optimisation on
-- Setting parallelisation on
-- Found OpenMP_C: -fopenmp (found version "4.5")

```

```

-- Found OpenMP_CXX: -fopenmp (found version "4.5")
-- Found OpenMP_Fortran: -fopenmp (found version "4.5")
-- Found OpenMP: TRUE (found version "4.5")
-- Setting docs on
-- Could NOT find Doxygen (missing: DOXYGEN_EXECUTABLE)
Doxygen need to be installed to generate the doxygen documentation, disabling
-- Looking for pthread.h
-- Looking for pthread.h - found
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Found OpenSSL: /usr/lib/x86_64-linux-gnu/libcrypto.so (found version "3.0.2")
-- Found Boost: /usr/local/lib/cmake/Boost-1.82.0/BoostConfig.cmake (found suitable
version "1.82.0", minimum required is "1.73.0") found components: log log_setup date_time
filesystem system thread program_options serialization timer
-- Looking for Fortran sgemm
-- Looking for Fortran sgemm - found
-- Found BLAS: /usr/lib/x86_64-linux-gnu/libopenblas.so
-- Found BLAS library: OpenBLAS
-- Found YAML library: /usr/local/lib/libyaml-cpp.a
-- Found Eigen version: 3.4.0
-- Found Boost version: 1.82.0
-- Found Mongocxx version: 3.8.0
-- Found C++ compiler: GNU 11.3.0
-- Found Git: /usr/bin/git (found version "2.34.1")
-- Git branch tag: untagged-488959f
-- Git branch: main
-- Found NetCDF: /usr/lib/x86_64-linux-gnu/libnetcdf_c++4.so;/usr/lib/x86_64-linux-
gnu/libnetcdf.so
-- CMAKE_CURRENT_BINARY_DIR: /home/ruPERT/ginan/src/build
-- CMAKE_CURRENT_SOURCE_DIR: /home/ruPERT/ginan/src
-- CMAKE_CXX_FLAGS: -std=c++2a -ggdb3 -fpermissive -Wall -Wno-write-strings -Wno-
deprecated-declarations -Wno-format-overflow -Wno-narrowing -Wno-unused-but-set-variable -
Wno-sign-compare -Wno-stringop-truncation -Wno-unused-variable -Wno-switch -Wno-dangling-
else -Wno-misleading-indentation -Wno-format-truncation -Wno-extern-c-compat -Wno-format-
zero-length -save-temps -O3 -fopenmp -pthread
-- CMAKE_Fortran_FLAGS: -g -frecursive -fall-intrinsics -Wall -fcheck=all -fbacktrace
-Wno-unused-dummy-argument -Wno-conversion -Wno-unused-variable -Wno-tabs -Wno-surprising
-Wno-unused-label -Wno-maybe-uninitialized -O3 -fopenmp -pthread
-- Configuring done
-- Generating done
CMake Warning:
  Manually-specified variables were not used by the project:

    OPTIMISATION

-- Build files have been written to: /home/ruPERT/ginan/src/build
george@ubuntu:~/ginan/src/build$ make -j2
[ 0%] Building CXX object cpp/CMakeFiles/otl.dir/loading/loading.cpp.o
[ 0%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/a2af.c.o
[ 0%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/a2tf.c.o
[ 1%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/ab.c.o
[ 1%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/ae2hd.c.o
[ 1%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/af2a.c.o
[ 1%] Building C object cpp/3rdparty/sofa/CMakeFiles/sofa_lib.dir/src/anp.c.o

```

Many many lines removed here.

```
[ 99%] Built target ginan_core
[ 99%] Building CXX object cpp/CMakeFiles/ginan.dir/gui/ginan.cpp.o
[ 99%] Building CXX object cpp/CMakeFiles/pea.dir/pea.cpp.o
[ 99%] Linking CXX executable /home/rupe/rupe/ginan/bin/pea
[ 99%] Building CXX object cpp/CMakeFiles/ginan.dir/gui/anode.cpp.o
[ 99%] Built target pea
[ 99%] Building CXX object cpp/CMakeFiles/ginan.dir/gui/beast.cpp.o
[100%] Linking CXX executable /home/rupe/rupe/ginan/bin/ginan
[100%] Built target ginan
george@ubuntu:~/ginan/src/build$
```

Figure 57 – Building Ginan source

That is Ginan built with all its dependencies. In the example below the yaml file used in the Docker section is used again – ex201george.yaml. This can be moved into the **exampleConfig** directory using Windows Explorer. The crucial command to set Ginan to work is `sudo ../bin/pea -y ex201george.yaml`.

```
george@ubuntu: ~
george@ubuntu:~/ginan/src/build$ cd ..
george@ubuntu:~/ginan/src$ cd ..
george@ubuntu:~/ginan$ cd exampleConfigs
george@ubuntu:~/ginan/exampleConfigs$ ls
README.md                ex203.yaml                mongo_outputs.yaml
brdc2sp3.yaml            ex204.yaml                network_clocks_and_biases.yaml
data                     ex205.yaml                network_orbits_clocks_and_biases.yaml
dynamic_user_ppp.yaml    ex206.yaml                pea
dynamic_user_rtk.yaml    ex42_gin2_pp_user_3freq.yaml  products
ex02_fit_sp3_pseudoobs.yaml ex48_gin2_pp_network_orbits_uduc_120s.yaml
record_streams.yaml
ex201.yaml               example_inputs.yaml        static_user.yaml
ex201george.yaml         large_post_network.yaml    tiny_post_network.yaml
ex202.yaml               medium_post_network.yaml   tiny_realtime_network.yaml
george@ubuntu:~/ginan/exampleConfigs$ sudo ../bin/pea -y ex201george.yaml

PEA starting... (HEAD release-v2.0.1-beta-e62cecc-dirty from Fri Jun 9 04:16:42 2023)

Checking configuration file ex201-RM-edit.yaml
Loading configuration from file tiny_post_network.yaml
Loading configuration from file static_user.yaml
Loading configuration from file example_inputs.yaml
Loading configuration from file ex201-RM-edit.yaml

=====
Configuration...
=====
Inputs:
    nav_files:  products/brdm1990.19p
```

```
snx_files: products/igs19P2062.snx
products/tables/igs_satellite_metadata_2203_plus.snx
products/IGS1R03SNX_20191950000_07D_07D_CRD.SNX
atx_files: products/igs20.atx
clk_files: products/IGS2R03FIN_20191990000_01D_30S_CLK.CLK
bsx_files: products/IGS2R03FIN_20191990000_01D_01D_OSB.BIA
blq_files: products/LOAD_G0.BLQ

Many many lines removed here.

RTS lag: 86370Found FILTER_MINUS
Found TRANSITION_MATRIX
Found TRANSITION_MATRIX
Found METADATA
Found METADATA

Outputting RTS products...
Removing RTS file: /data/outputs/ex201/./Filter-ex201-Network.rts_forward
Removing RTS file: /data/outputs/ex201/./Filter-ex201-Network.rts_backward

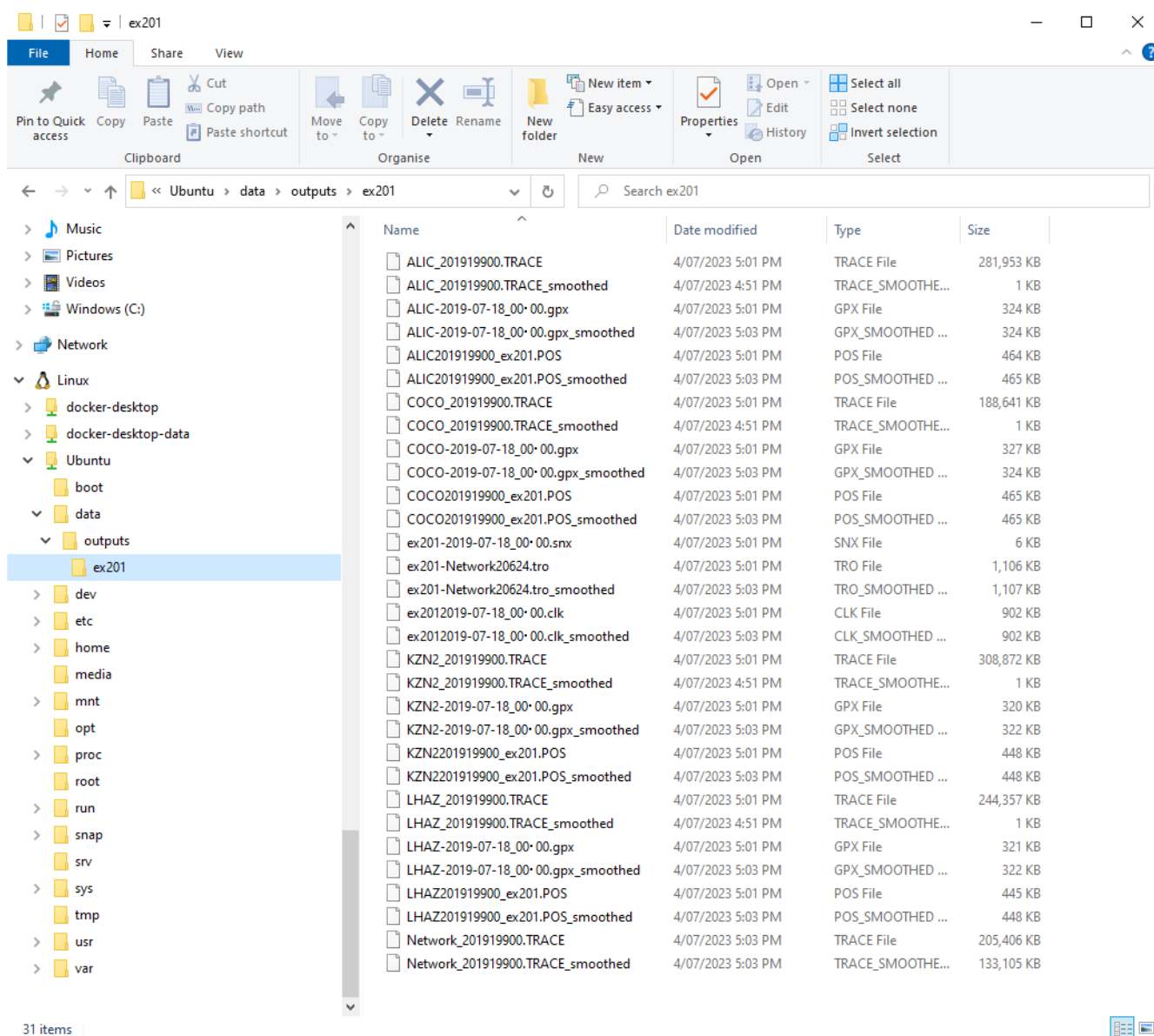
PEA started processing at : 2023-07-04 06:51:13.27
and finished processing at : 2023-07-04 07:03:52.85
Total processing duration : 00:12:39.58

PEA finished

george@ubuntu:~/ginan/exampleConfigs$
```

**Figure 58** – Ginan running and processing data in accordance with ex201george.yaml

Finally the products created by Ginan's processing can be found in **data/outputs/ex201**



**Figure 59** – After processing data in accordance with ex201george.yaml, Ginan produces the output files



## 6 Embedded configuration

### 6.1 Supported use cases

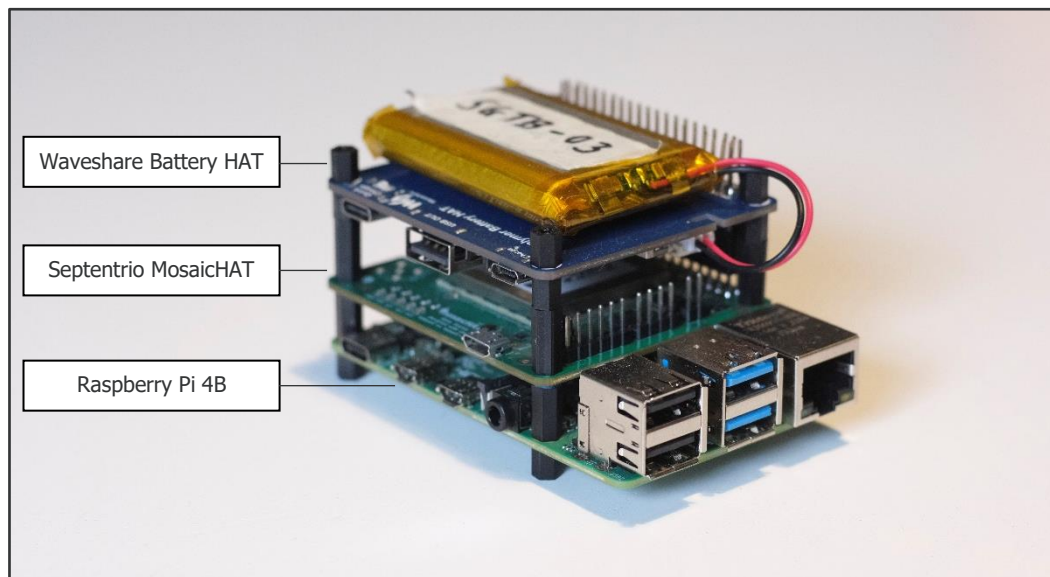
This configuration supports the following use cases:

- UC 4 Improve the performance of crustal movement and earthquake monitoring,
- UC 7 Giving precise positioning to the Internet of Things (IoT) for new applications and edge-computing,
- UC 11 Making the user platform part of systems to bring precise positioning to mobile consumer devices e.g. phones, tablets. (Android, iOS),
- UC 12 Running the user platform as embedded software for autonomous vehicles (land, sea, air, space),

### 6.2 Hardware Software Integration

The embedded reference configuration of Ginan is hosted on a Raspberry Pi 4B single board computer running Raspberry Pi OS Lite (Debian 11 Bullseye). A GNSS receiver hardware attached on top (HAT) that supports Radio Technical Commission for Maritime Services 3 (RTCM3) data output is required to provide Ginan with GNSS observations, and a Li-Po battery HAT supplies the Raspberry Pi and the GNSS receiver with power.

Figure 37 below shows the specific hardware used for the embedded Ginan configuration.



**Figure 60** - The core embedded Ginan hardware.

The HAT components are connected to the Raspberry Pi's General Purpose Input/Output (GPIO) header, providing the GNSS HAT serial communication via Universal Asynchronous Receiver/Transmitter (UART) and power from the battery HAT.

The Septentrio MosaicHAT is an Open Source GNSS HAT that integrates Septentrio's mosaic-X5 GNSS receiver module. The module is configured to output RTCM3 GNSS observations to the Raspberry Pi's UART serial port. The following RTCM3 messages are enabled:

- 1006 Stationary RTK Reference Station ARP & Antenna Height
- 1013 System Parameters, time offsets, lists of messages sent
- 1019 GPS Broadcast Ephemeris
- 1020 GLONASS Broadcast Ephemeris
- 1033 Receiver and Antenna Descriptors
- 1042 BeiDou Broadcast Ephemeris
- 1044 QZSS Broadcast Ephemeris



- 1045 Galileo F/NAV Ephemeris
- 1046 Galileo I/NAV Ephemeris
- 1077 GPS MSM7 (Pseudorange, PhaseRange, Doppler, CNR, with high resolution)
- 1087 GLONASS MSM7 (Pseudorange, PhaseRange, Doppler, CNR, with high resolution)
- 1097 Galileo MSM7 (Pseudorange, PhaseRange, Doppler, CNR, with high resolution)
- 1117 QZSS MSM7 (Pseudorange, PhaseRange, Doppler, CNR, with high resolution)
- 1127 BeiDou MSM7 (Pseudorange, PhaseRange, Doppler, CNR, with high resolution)
- 1230 GLONASS L1 and L2 Code-Phase Biases

The Raspberry Pi 4B hardware UART serial port needs to be properly configured to receive data from the MosaicHAT. By default, the hardware UART is used by a getty login shell and the Bluetooth subsystem. These need to be disabled to allow the MosaicHAT to use the high-performance hardware UART. This can be configured with the following modifications to the Raspberry Pi operating system.

1. Disable console getty programs:
  - a. Edit `/boot/cmdline.txt` (as superuser)
  - b. Remove the line with `console=serial0,115200`
2. Disable the console service (run commands as superuser):
  - a. `systemctl stop serial-getty@ttyS0.service`
  - b. `systemctl disable serial-getty@ttyS0.service`
3. Enable UART serial port:
  - a. Edit `/boot/config.txt` (as superuser)
  - b. Append the following lines to the end of file:
 

```
enable_uart=1
dtoverlay=miniuart-bt
```

Ginan is built with docker on a computer (x86-64 architecture) running Ubuntu 22.04. The docker buildx tool is used to build the Ginan environment image (dependencies) and the Ginan image for ARM64v8-A architecture. The docker buildx plugin is installed when following the Docker Engine Installation Instructions (See: <https://docs.docker.com/engine/install/ubuntu/>), additionally it requires the `qemu-user-static` and `binfmt-support` which can be installed with the `apt` package management system.

To reduce the image size and speed up build time, it is recommended to edit the Ginan environment image (`ginan/aws/docker-env/Dockerfile`) and remove dependencies that are not required. These are:

1. Doxygen
2. Graphviz
3. Texlive-latex-base
4. Texlive-latex-extra
5. Python3-pip
6. Python3-dev
7. MongoDB

If these features are desired, MongoDB and Python3 (for the Exploratory Data Analysis package) can be installed on the host or in separate docker containers. The Ginan environment image can be built like so (from the Ginan root folder):

```
docker buildx build -t ginan-env-arm64 -f ./aws/docker-env/Dockerfile --platform linux/arm64 .
```

Once complete, edit the Ginan docker file (`ginan/docker/Dockerfile`) and set the base image to `ginan-env-arm64`. Edit the `make` instruction to only install the PEA package. If python was not installed in the environment image then remove the line with `RUN pip3 install gnssanalysis==x.x.xx`. Build the Ginan image like so (from the Ginan root folder):

```
docker buildx build -t ginan-arm64 -f ./docker/Dockerfile --platform linux/arm64 .
```

Once complete, the Ginan image should be exported as a tar archive and copied to the Raspberry Pi's filesystem.

```
docker save ginan-arm64 | gzip > ginan-arm64.tar.gz
```

Then load the Ginan image to the Raspberry Pi's docker engine like so:

```
docker load < ginan-arm64.tar.gz
```

The PEA is configured with a YAML file that specifies key-value pairs for predefined parameters. Ginan can be configured to ingest the RTCM3 observations from the serial port with the following YAML snippet:

```
inputs:
  gnss_observations:
    rtcm_inputs:
      - serial:///dev/ttyAMA0
```

The Ginan docker image can be run interactively like so:

```
docker run -it -v <workspace>:/ginan/data --device=/dev/ttyAMA0 ginan-arm64 bash
```

This also mounts a specified *workspace* directory (containing the YAML and other input files) as a volume and forwards the MosaichAT's serial port to the container.

Prior to running the PEA, prepare the serial port with *stty* (set teletype). The following command sets the baudrate to 115200, set 0 characters minimum for a completed read, and disable several settings (these settings may differ between GNSS receivers):

```
stty -F /dev/ttyAMA0 115200 min 0 -icrnl -ixon -opost -onlcr -isig -icanon -iexten -echo -echoe -  
echok -echoctl -echoke
```

Finally, in the Ginan container, run PEA and point to the configuration file in the Ginan docker container:

```
pea --config data/config.yml
```

## 7 Server based configuration

### 7.1 Supported use cases

This configuration supports the following use cases:

- UC 3 Work to maintain and improve Australia's geodetic datums,
- UC 5 To monitor the performance of networks of continuously operating reference stations (CORS).
- UC 6 Detection of geohazards such as tsunamis, cyclones and space weather events through ionospheric disturbance monitoring.
- UC 8 On-selling correction products and streams with value-adding services,
- UC 12 Precise orbit determination including LEO satellite fleets,
- UC 13 Ground truthing objects to assist with space situational awareness.
- UC 14 An alternative source of positioning data to check the performance of SouthPAN,
- UC 15 As part of a system to monitor the performance of GNSS signals over the South Pacific,
- UC 16 Use products such as the ZTD file to improve weather forecasting and climate change monitoring.

### 7.2 Hardware Software Integration

GA maintains an operational instance of Ginan with three primary purposes:

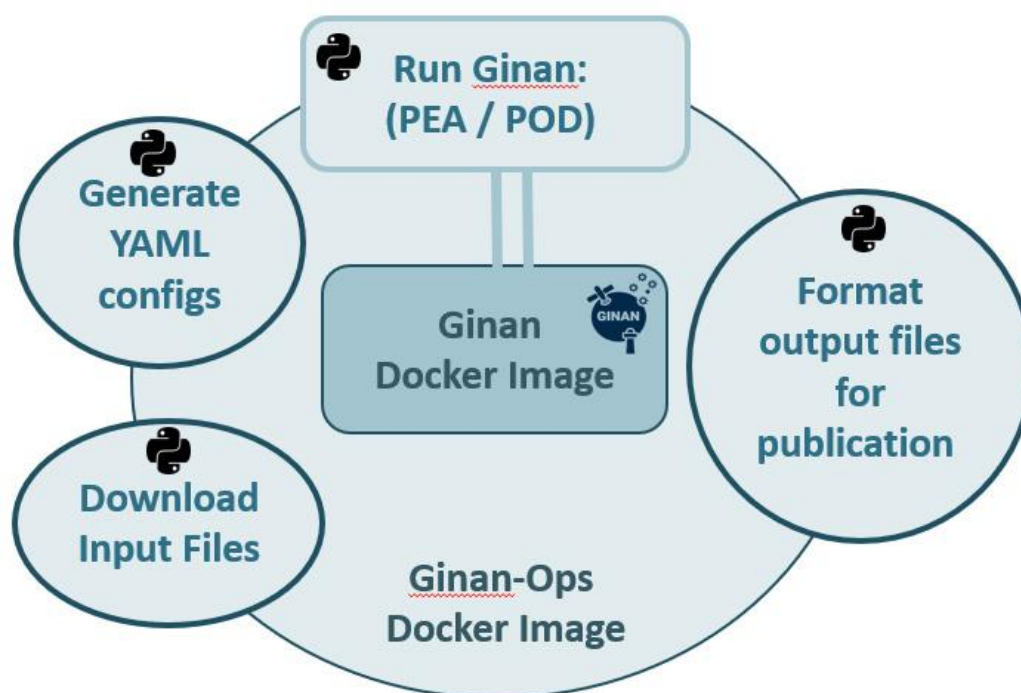
- Building IGS Analysis Centre style products (files): Orbits, Clocks, CORS positions,
- Provide a positioning service to the public in a best effort / experimental state: GPS Broadcast Correction streams,
- To gain experience in running and operating Ginan to inform the development team.

GA operates three types of operational instance:

- A development system where new versions of Ginan can be tested along with any changes needed to the operational frameworks that support Ginan,
- A quality assurance system, where a new version of Ginan can be load and stress tested and assessed for reliability,
- A production environment that is used to produce PPP file products and streams.

At the heart of the GA system is the Ginan docker image. Around this is built the Ginan-Ops docker image. The Ginan-Ops image has python scripts to:

- Download input files,
- Generate YAML configs,
- Run Ginan,
- Format output files for publication.



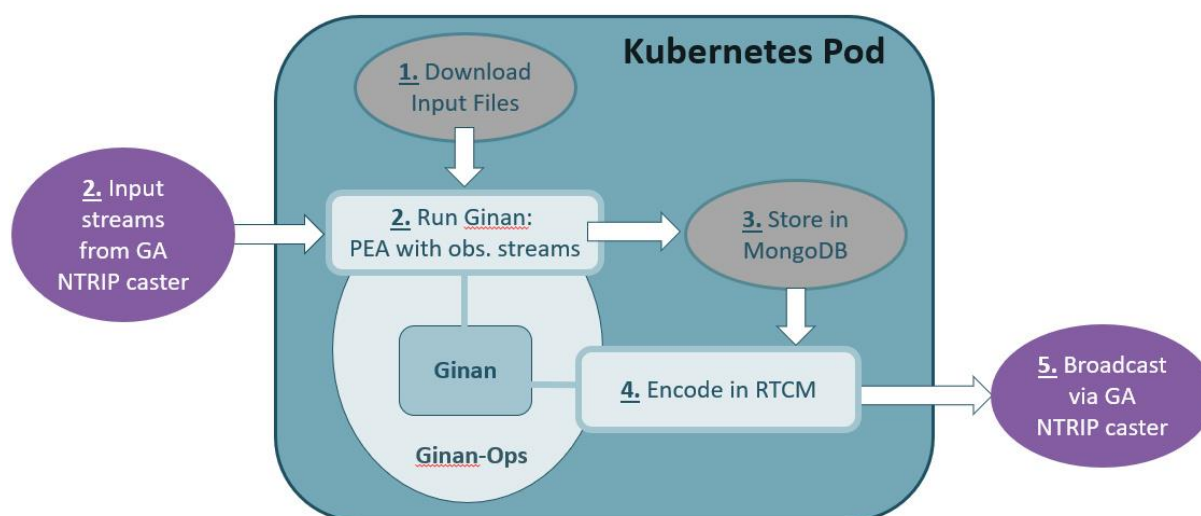
**Figure 61** - The Ginan-Ops Docker image.

In the context of cloud computing, **Pods** are the smallest deployable units of computing that you can create and manage in Kubernetes. A pod is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A pod models an application-specific "logical host": it contains one or more application containers, such as a Docker container, which are relatively tightly coupled.

The file-based products are produced on pods instantiated only for the time needed to create the files (destroyed afterwards).

The real-time tasks are run continuously on a given pod:

- Real-time orbits and clocks correction stream,
- Real-time PPP results of a sample of Australian CORS,
- Monitoring + Comparison exists around the main outputs of Ginan-Ops.



**Figure 62** – Real-time streams pod – as an example.

In summary, the GA Ginan stack consists of:

- Bitbucket as a repository for all development code. Bitbucket is closely integrated with Jira and used for work task identification and scheduling. Bitbucket pipelines (automated tests) are used for each commit (simple test) and merges into the main branch (extensive test).
- Amazon Web Services provides the on-demand cloud computing platform.
- Kubernetes (K8s) is an open-source container orchestration system for automating software deployment, scaling, and management in cloud environments.
- Terraform is an open-source infrastructure-as-code software tool created by HashiCorp. Users define and provide data centre infrastructure using a declarative configuration language known as HashiCorp Configuration Language (HCL) or JSON.
- Argo Workflows is an open source container-native workflow engine for orchestrating parallel jobs on Kubernetes. Argo Workflows is implemented as a Kubernetes CRD (Custom Resource Definition).
- Python: manages steps in running Ginan jobs (Downloads, POD, PEA runs, etc.).
- Ginan: the precise point positioning toolkit from Geoscience Australia.

The operation of Ginan is monitored for:

- **Stability:**
  - Streams are online,
  - File-based jobs complete successfully,
  - Resource-use is within limits.
- **Quality:**
  - Compare GA's correction streams with other IGS Analysis Centres.

Stability is monitored using Argo and Grafana:

- Argo workflow diagrams track the progress of file-based jobs,
- Grafana dashboards monitor resource use (CPU, Memory) and stream messages,
- Alerts are posted to the team instant messaging platform (from Argo, Grafana).

Quality is monitored using Kibana:

- SP3, CLK and SNX file are products compared against IGS Rapid/Final products,

- Ginan correction streams compared against IGS correction stream,
- Kibana is used to display the results of Comparisons.