CS-462

Lab Project
Network Slicing in 5g Drones

Vaibhav Kumar Jonwal
202251150

Contents

NS-3 Setup

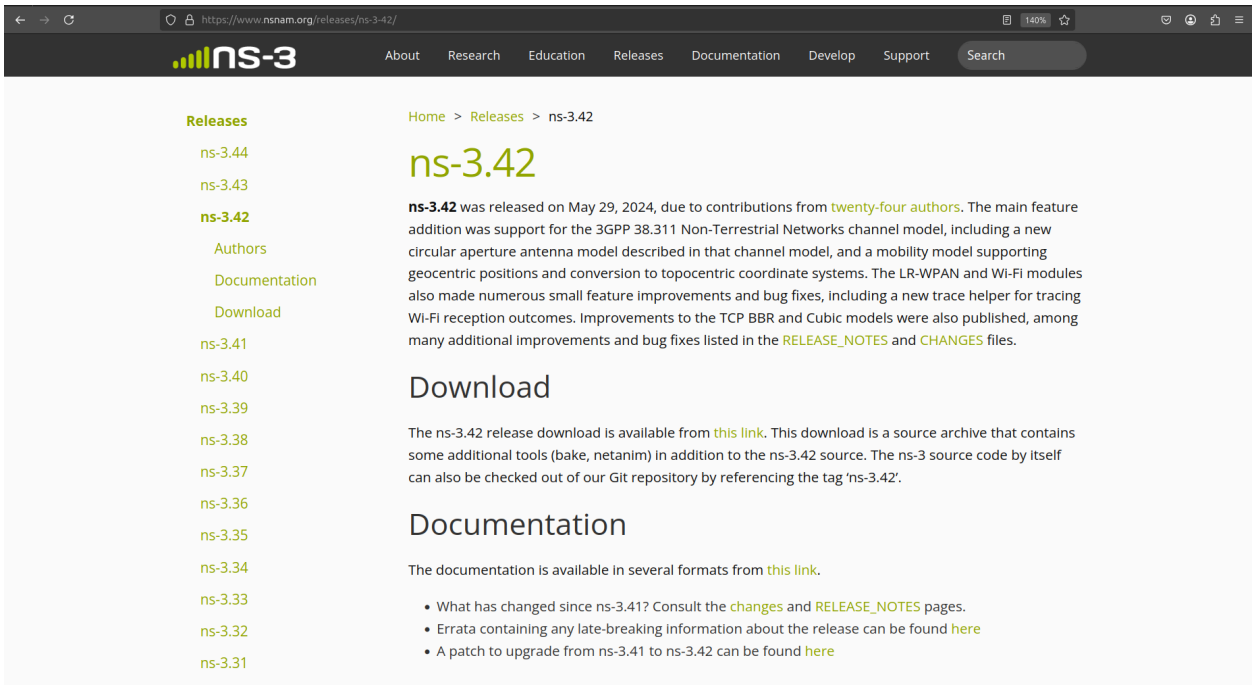
Installation

1. Downloading NS-3

(We will use ns-3.42 for compatibility with 5g-lena)

Step1: Go to Link: https://www.nsnam.org/releases/ns-3-42/

Step2: Click on this link or from website as shown in the below image to download the zipped file from the ns3 website.



Step3: Make your project directory

(Consider we are at /username directory)

Go to Terminal:

cd Desktop

mkdir project

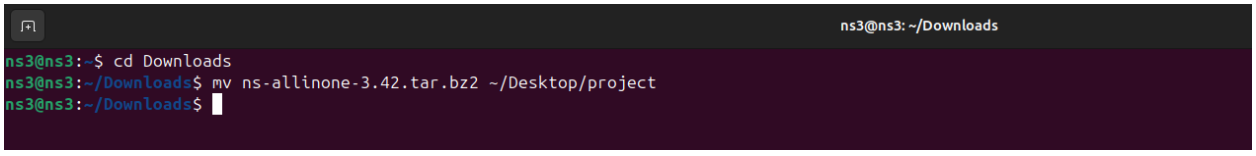
Step4: Move the file to your project directory

(Consider we are at /username directory)

Go to Terminal:

cd Downloads

mv ns-allinone-3.42.tar.bz2 ~/Desktop/project



Step5: Go to project Directory and extract the file

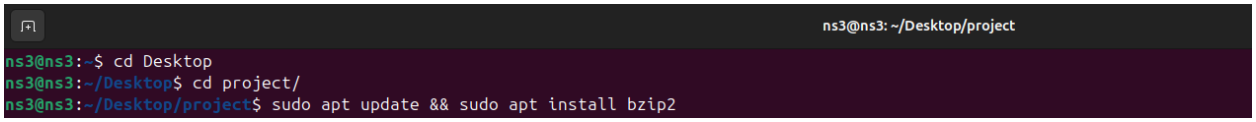
(Consider we are at /username directory)

Go to Terminal:

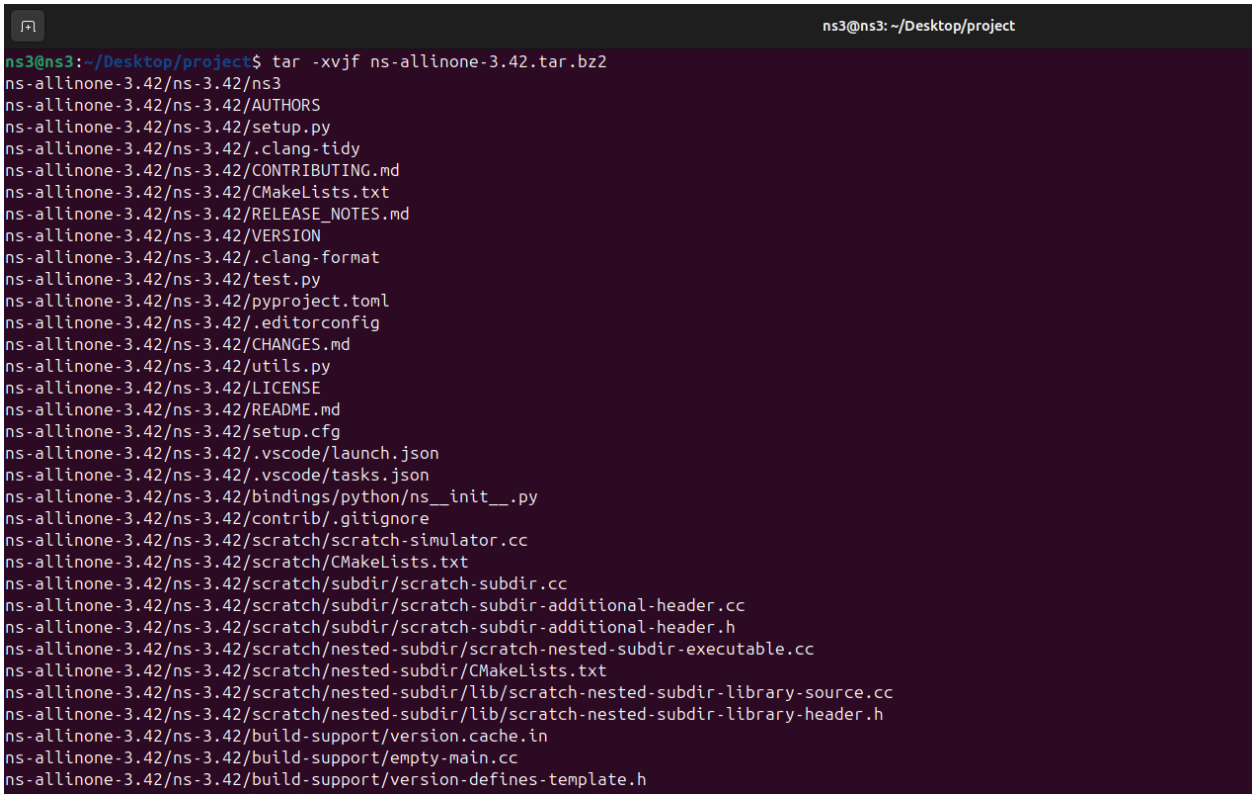
cd Desktop

cd project

sudo apt update && sudo apt install bzip2



tar -xvjf ns-allinone-3.42.tar.bz2



2. Install Dependencies

Install all the dependencies required to build

(Consider we are at /username directory)

Go to Terminal:

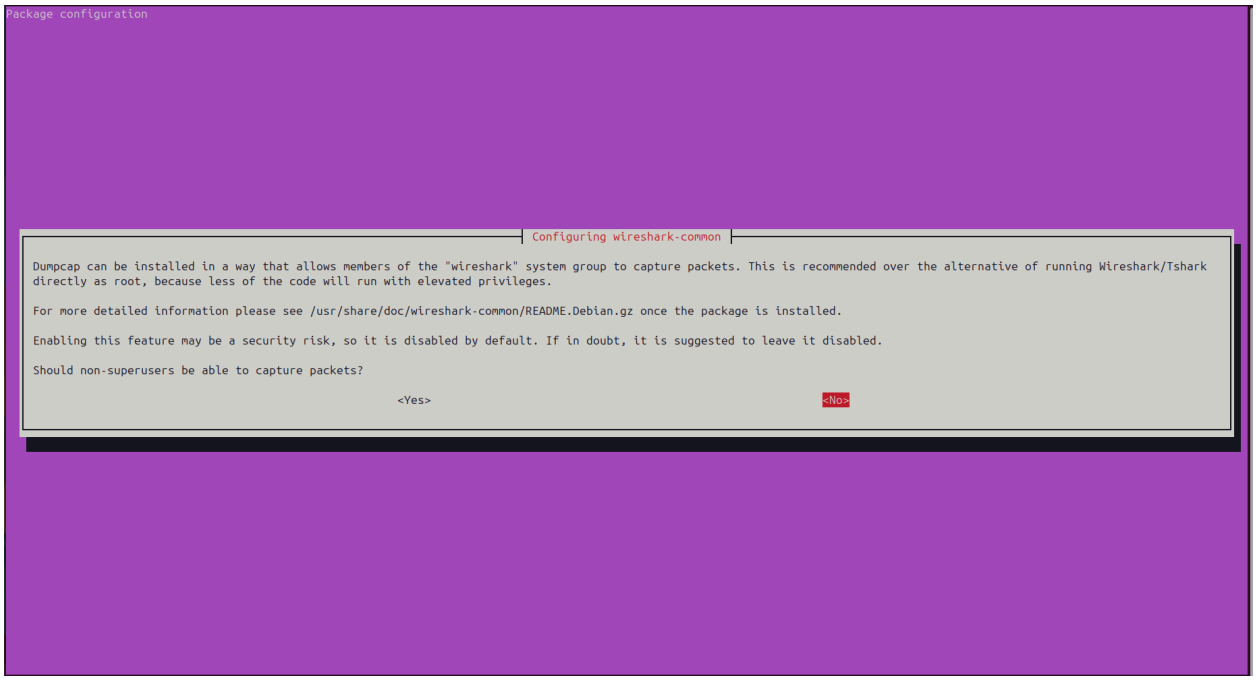
sudo apt install g++ python3 cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark sqlite3 libsqlite3-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools openmpi-bin openmpi-common openmpi-doc libopenmpi-dev doxygen graphviz imagemagick python3-sphinx dia texlive dvipng latexmk texlive-extra-utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgsl-dev libgslcblas0 libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-utilities ebtables bridge-utils libboost-all-dev ccache

```
ns3@ns3:~$ sudo apt install g++ python3 cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark sqlite3 libs
qlite3-dev qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools openmpi-bin openmpi-common openmpi-doc libopenmpi-dev doxygen graphviz imagemagick python3-sphinx dia texlive dvipng latexm
k texlive-extra-utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgsl-dev libgslcblas0 libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-utilities e
btables bridge-utils libboost-all-dev ccache
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
python3 is already the newest version (3.12.3-0ubuntu2).
python3 set to manually installed.
python3-gi is already the newest version (3.48.2-1).
python3-gi set to manually installed.
tcpdump is already the newest version (4.99.4-3ubuntu4).
tcpdump set to manually installed.
```

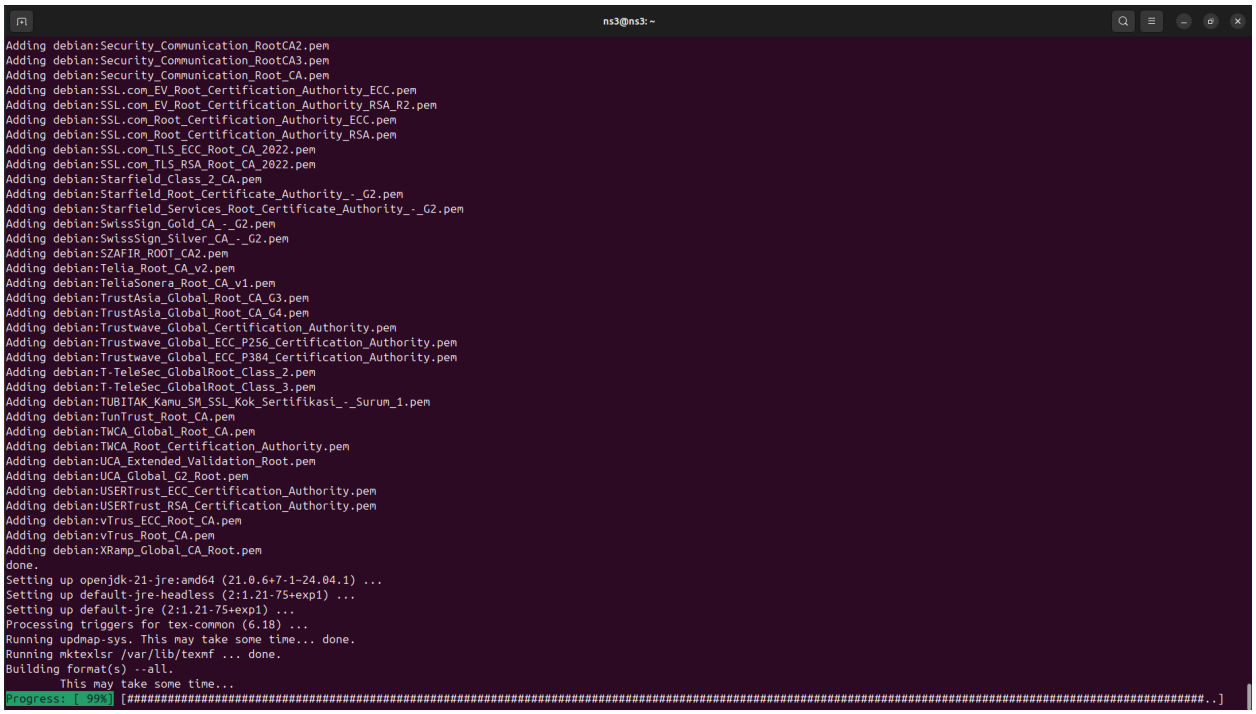
(Type Y and continue to Download)

```
The following packages will be upgraded:
  gir1.2-gtk-3.0 gtk-update-icon-cache libexpat1 libgtk-3-0t64 libgtk-3-bin libgtk-3-common liblzma5 libpython3.12-minimal lib
  python3.12-minimal xz-utils
14 upgraded, 691 newly installed, 0 to remove and 171 not upgraded.
Need to get 798 MB of archives.
After this operation, 2,974 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

(Opt any option as per your preference)



(Wait until the progress hits 100%)



3. Building NS-3

Step1: Go to ns-allinone-3.42 directory

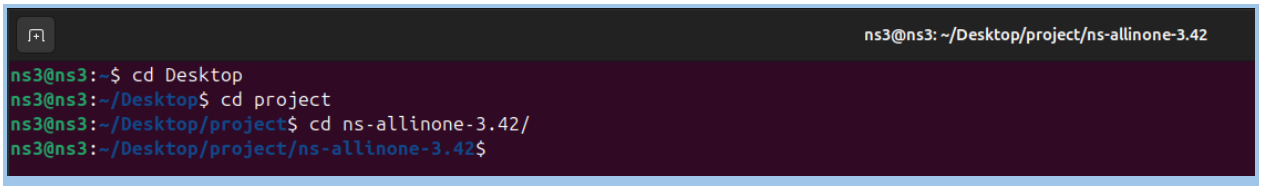
(Consider we are at /username directory)

Go to Terminal:

cd Desktop

cd project

cd ns-allinone-3.42

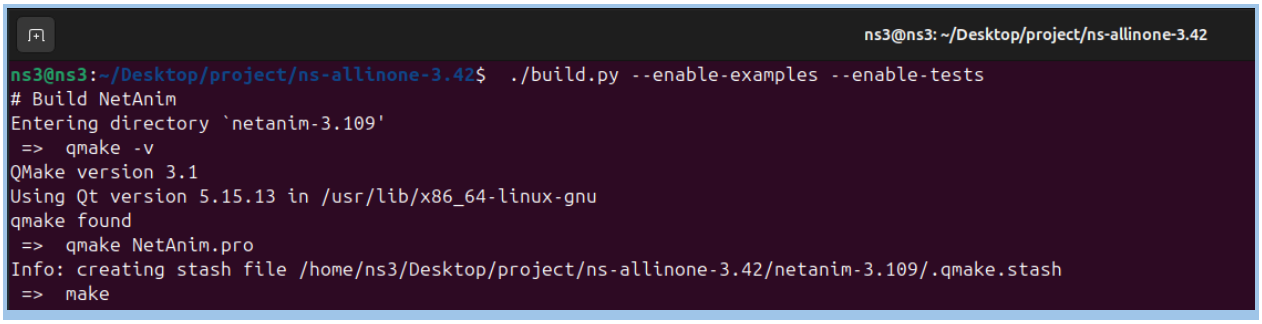


Step2: Build NS-3

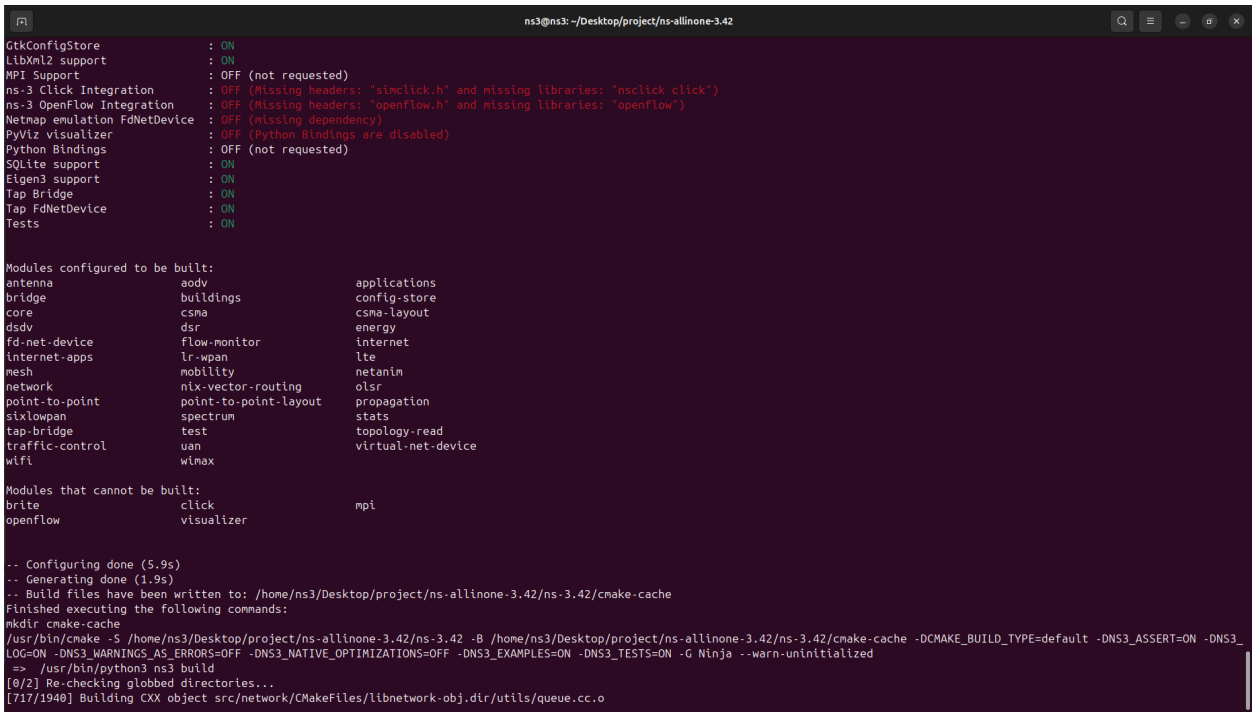
(Consider we are at /ns-allinone-3.42 directory)

Go to Terminal:

./build.py --enable-examples --enable-tests



(Wait until it completes 1940/1940)



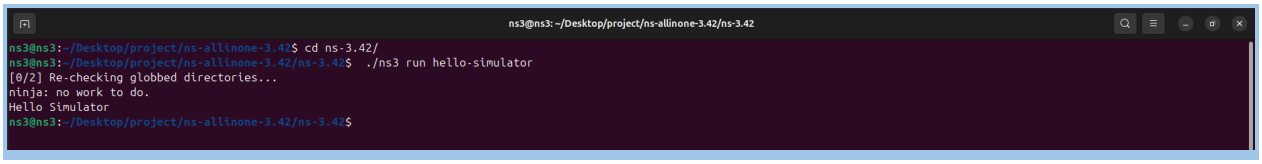
Step3: Check the Built

(Consider we are at /ns-allinone-3.42 directory)

Go to Terminal:

cd ns-3.42

./ns3 run hello-simulator



(It is showing Hello Simulator)

NetSimulyzer Setup

Installation

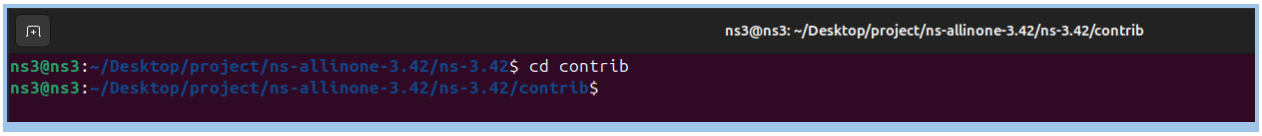
1. **Downloading NetSimulyzer NS-3 module**

Step1: Go to /contrib directory

(Consider we are at /ns-3.42 directory)

Go to Terminal:

cd contrib



Step2: Go to /contrib directory

(Consider we are at /ns-3.42 directory)

Go to Terminal:

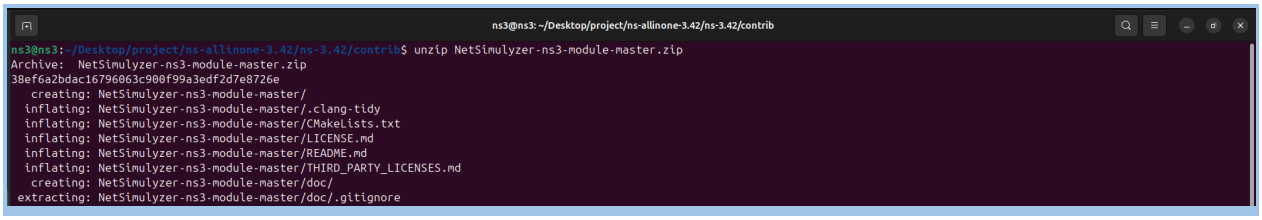
wget https://github.com/usnistgov/NetSimulyzer-ns3-module/archive/master.zip -O NetSimulyzer-ns3-module-master.zip

Step3: Unzip downloaded file

(Consider we are at /contrib directory)

Go to Terminal:

unzip NetSimulyzer-ns3-module-master.zip



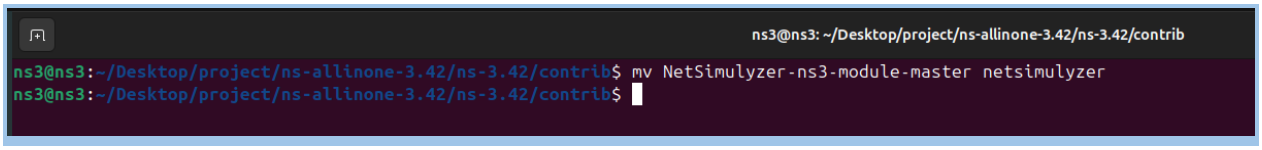
Step3: Rename the resulting directory to netsimulyze

(As ns-3 will not accept a module named differently than its directory.)

(Consider we are at /contrib directory)

Go to Terminal:

mv NetSimulyzer-ns3-module-master netsimulyzer



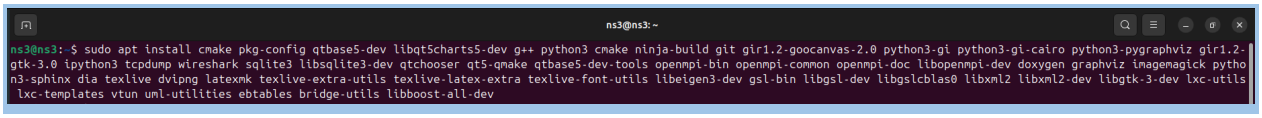
2. Install Dependencies

Install all dependencies for NetSimulyzer

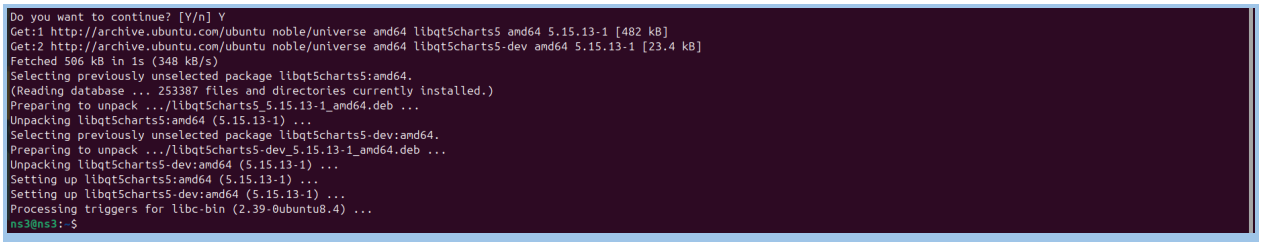
(Consider we are at /username directory)

Go to Terminal:

sudo apt install cmake pkg-config qtbase5-dev libqt5charts5-dev g++ python3 cmake ninja-build git gir1.2-goocanvas-2.0 python3-gi python3-gi-cairo python3-pygraphviz gir1.2-gtk-3.0 ipython3 tcpdump wireshark sqlite3 libsqlite3-dev qtchooser qt5-qmake qtbase5-dev-tools openmpi-bin openmpi-common openmpi-doc libopenmpi-dev doxygen graphviz imagemagick python3-sphinx dia texlive dvipng latexmk texlive-extra-utils texlive-latex-extra texlive-font-utils libeigen3-dev gsl-bin libgsl-dev libgslcblas0 libxml2 libxml2-dev libgtk-3-dev lxc-utils lxc-templates vtun uml-utilities ebtables bridge-utils libboost-all-dev



(Press Y to continue)



3. Building NS-3 for NetSimulyzer

Step1: Go to ns-allinone-3.42 directory

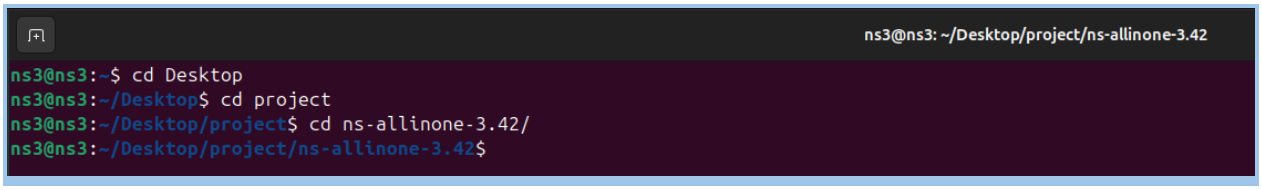
(Consider we are at /username directory)

Go to Terminal:

cd Desktop

cd project

cd ns-allinone-3.42

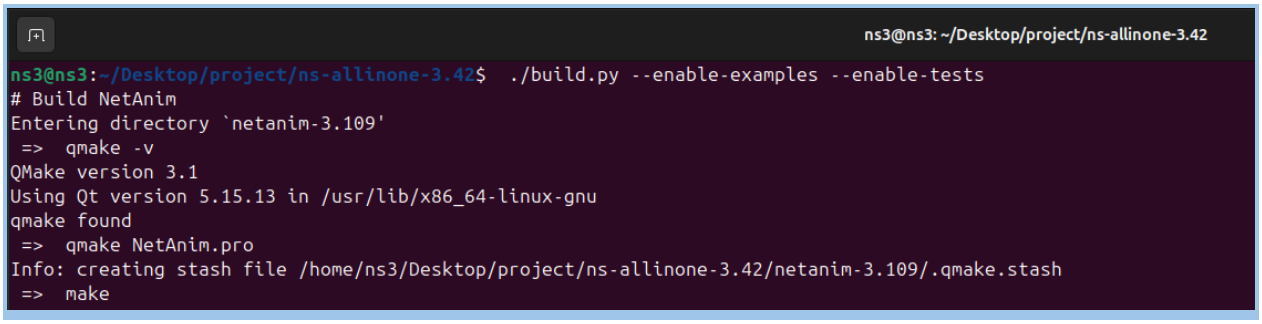


Step2: Build NS-3

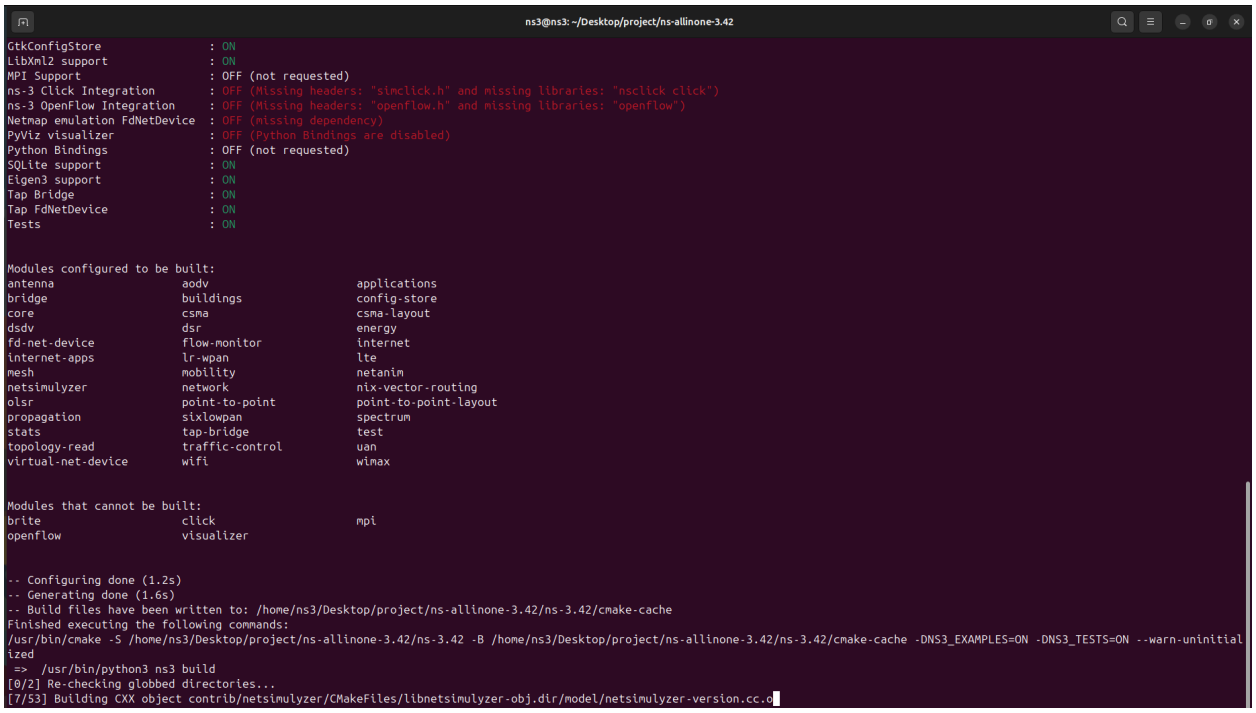
(Consider we are at /ns-allinone-3.42 directory)

Go to Terminal:

./build.py --enable-examples --enable-tests



(Wait until it completes 53/53)



Step3: Check the Built

(Consider we are at /ns-allinone-3.42 directory)

Go to Terminal:

cd ns-3.42

./ns3 run contrib/netsimulyzer/examples
mobility-buildings-example.cc

4. Installing NetSimulyzer Software

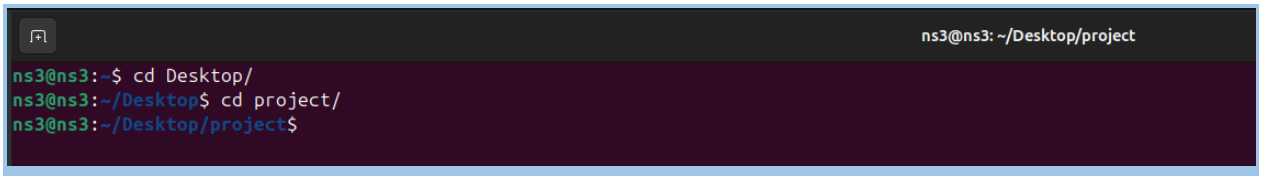
Step1: Go to project directory

(Consider we are at /username directory)

Go to Terminal:

cd Desktop

cd project

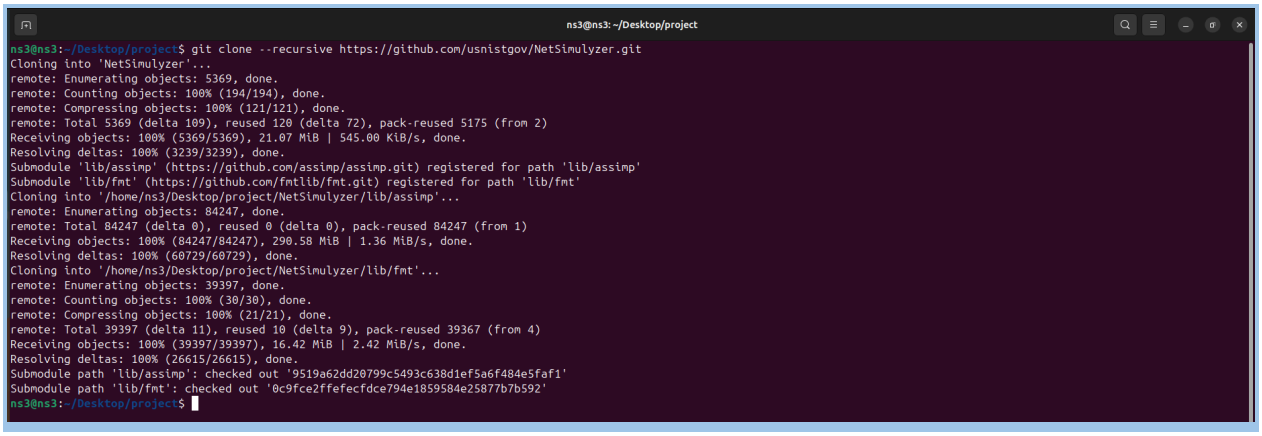


Step2: Clone the NetSimulyzer Repo

(Consider we are at /project directory)

Go to Terminal:

clone --recursive https://github.com/usnistgov/NetSimulyzer.git



Step3: Make /build directory

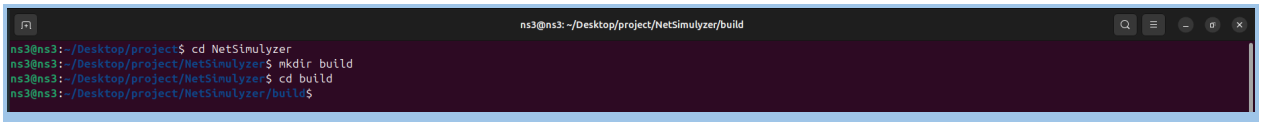
(Consider we are at /project directory)

Go to Terminal:

cd NetSimulyzer

mkdir build

cd build

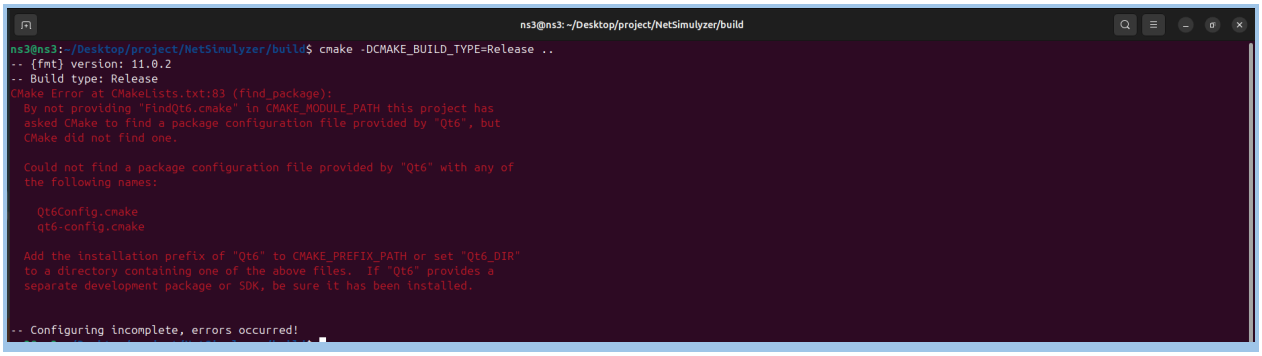


Step4: Build Software

(Consider we are at /build directory)

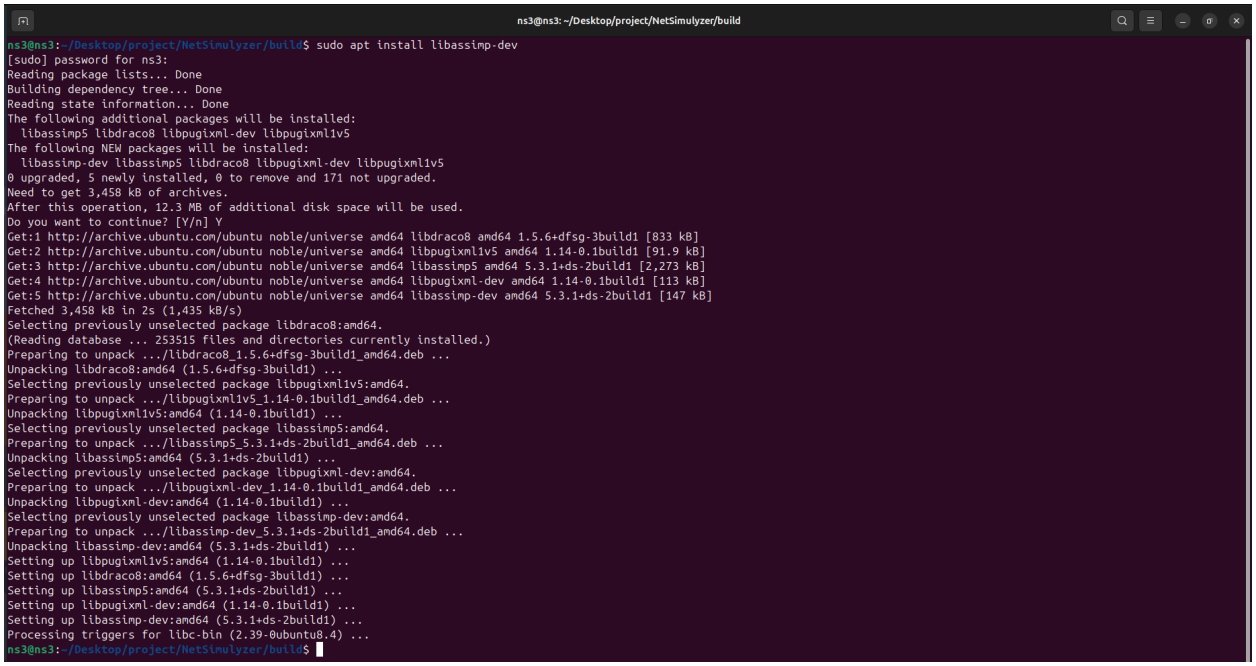
Go to Terminal:

cmake -DCMAKE_BUILD_TYPE=Release ..

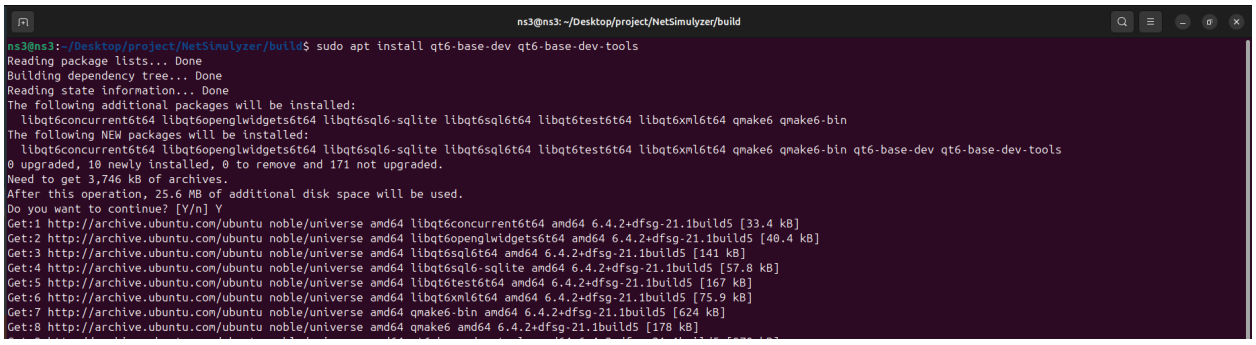


(Error Occurred, we will solve the error by installing required dependencies)

sudo apt install libassimp-dev



sudo apt install qt6-base-dev qt6-base-dev-tools



(Now Again)

cmake -DCMAKE_BUILD_TYPE=Release ..

```
ns3@ns3: ~/Desktop/project/NetSimulyzer/build
ns3@ns3:~/Desktop/project/NetSimulyzer/build$ cmake -DCMAKE_BUILD_TYPE=Release ..
-- {fmt} version: 11.0.2
-- Build type: Release
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD
-- Performing Test CMAKE_HAVE_LIBC_PTHREAD - Success
-- Found Threads: TRUE
-- Performing Test HAVE_STDATOMIC
-- Performing Test HAVE_STDATOMIC - Success
-- Found WrapAtomic: TRUE
-- Found OpenGL: /usr/lib/x86_64-linux-gnu/libOpenGL.so
-- Found WrapOpenGL: TRUE
-- Found XKB: /usr/lib/x86_64-linux-gnu/libxkbcommon.so (found suitable version "1.6.0", minimum required is "0.5.0")
-- Found WrapVulkanHeaders: /usr/include
-- Configuring done (0.7s)
-- Generating done (0.0s)
-- Build files have been written to: /home/ns3/Desktop/project/NetSimulyzer/build
ns3@ns3:~/Desktop/project/NetSimulyzer/build$
```

cmake --build .

```
ns3@ns3: ~/Desktop/project/NetSimulyzer/build
ns3@ns3:~/Desktop/project/NetSimulyzer/build$ cmake --build . --parallel
[  0%] Built target QCustomPlot_autogen_timestamp_deps
[  1%] Building CXX object lib/fmt/CMakeFiles/fmt.dir/src/format.cc.o
[  3%] Building CXX object lib/fmt/CMakeFiles/fmt.dir/src/os.cc.o
[  4%] Automatic MOC and UIC for target QCustomPlot
[  4%] Built target QCustomPlot_autogen
[  6%] Building CXX object lib/QCustomPlot/CMakeFiles/QCustomPlot.dir/QCustomPlot_autogen/mocs_compilation.cpp.o
[  8%] Building CXX object lib/QCustomPlot/CMakeFiles/QCustomPlot.dir/qcustomplot.cpp.o
[  9%] Linking CXX static library libfmt.a
[  9%] Built target fmt
[ 13%] Building CXX object parser/CMakeFiles/parser.dir/file-parser.cpp.o
[ 13%] Building CXX object parser/CMakeFiles/parser.dir/handler/JsonHandler.cpp.o
```

(Wait until the build)

```
ns3@ns3: ~/Desktop/project/NetSimulyzer/build
[ 18%] Built target netsimulyzer_autogen_timestamp_deps
[ 19%] Built target netsimulyzer_autogen
[ 21%] Building CXX object CMakeFiles/netsimulyzer.dir/netsimulyzer_autogen/mocs_compilation.cpp.o
[ 22%] Building CXX object CMakeFiles/netsimulyzer.dir/main.cpp.o
[ 24%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/area/Area.cpp.o
[ 26%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/building/Building.cpp.o
[ 27%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/decoration/Decoration.cpp.o
[ 29%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/link/LogicalLink.cpp.o
[ 31%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/link/WiredLink.cpp.o
[ 32%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/node/Node.cpp.o
[ 34%] Building CXX object CMakeFiles/netsimulyzer.dir/src/group/node/TrailBuffer.cpp.o
[ 36%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/camera/ArcCamera.cpp.o
[ 37%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/camera/Camera.cpp.o
[ 39%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/font/FontManager.cpp.o
[ 40%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/framebuffer/PickingFramebuffer.cpp.o
[ 42%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/helper/Floor.cpp.o
[ 44%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/mesh/Mesh.cpp.o
[ 45%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/model/Model.cpp.o
[ 47%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/model/ModelCache.cpp.o
[ 49%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/renderer/Renderer.cpp.o
[ 50%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/shader/Shader.cpp.o
[ 52%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/helper/SkyBox.cpp.o
[ 54%] Building CXX object CMakeFiles/netsimulyzer.dir/src/render/texture/TextureCache.cpp.o
[ 55%] Building CXX object CMakeFiles/netsimulyzer.dir/src/settings/SettingsManager.cpp.o
[ 57%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/about/AboutDialog.cpp.o
[ 59%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/LoadWorker.cpp.o
[ 60%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/MainWindow.cpp.o
[ 62%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/scene/SceneWidget.cpp.o
[ 63%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/settings/SettingsDialog.cpp.o
[ 65%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/util/file-operations.cpp.o
[ 67%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/chart/ChartManager.cpp.o
[ 68%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/chart/ChartWidget.cpp.o
[ 70%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/chart/ControlsChartView.cpp.o
[ 72%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/controls/SingleKeySequenceEdit/SingleKeySequenceEdit.cpp.o
[ 73%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/log/ScenarioLogWidget.cpp.o
[ 75%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/node/NodeWidget.cpp.o
[ 77%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/detail/DetailManager.cpp.o
[ 78%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/detail/DetailWidget.cpp.o
[ 80%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/playback/PlaybackJumpDialog.cpp.o
[ 81%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/playback/PlaybackTimeStepDialog.cpp.o
[ 83%] Building CXX object CMakeFiles/netsimulyzer.dir/src/window/playback/PlaybackWidget.cpp.o
[ 85%] Building CXX object CMakeFiles/netsimulyzer.dir/src/conversion.cpp.o
[ 86%] Linking CXX executable netsimulyzer
[100%] Built target netsimulyzer
ns3@ns3:~/Desktop/project/NetSimulyzer/build$
```

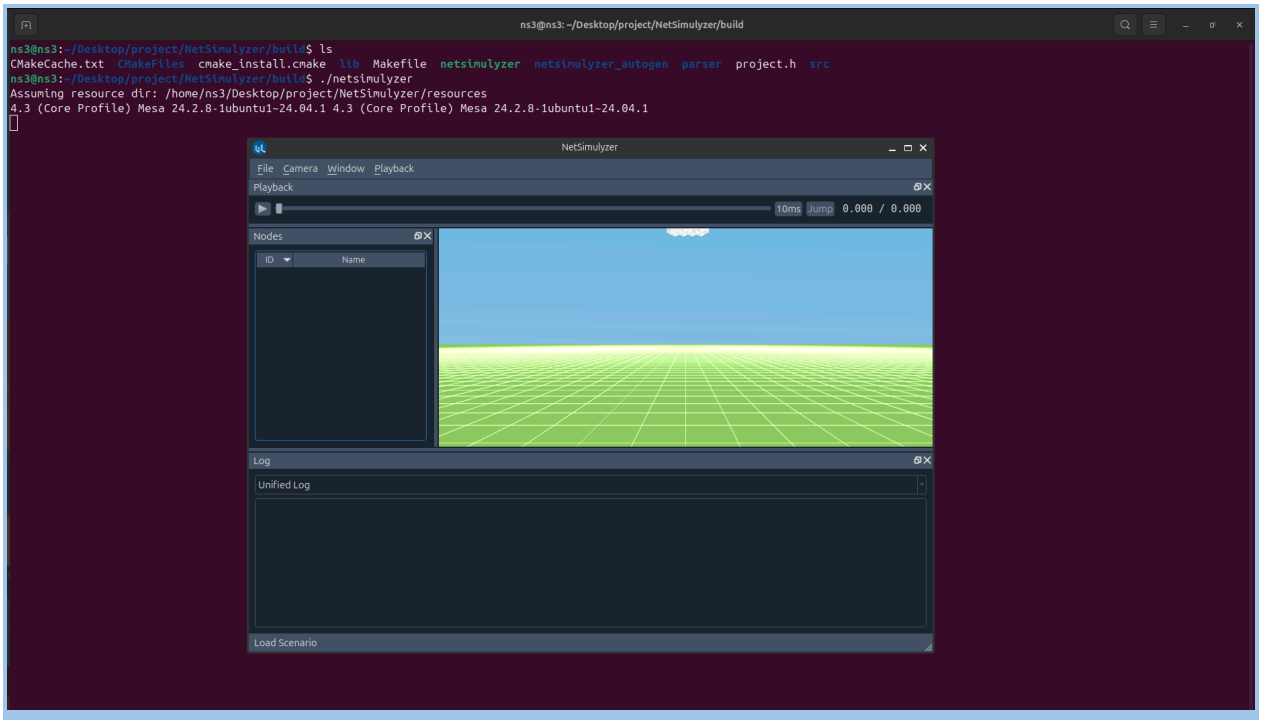
Step5: Check NetSimulyzer

(Consider we are at /build directory)

Go to Terminal:

cd NetSimulyzer

./netsimulyzer



Drone Simulation

Setup Code & Simulation

1. **Write Code**

Step1: Go to /scratch directory

(Consider we are at /username directory)

Go to Terminal:

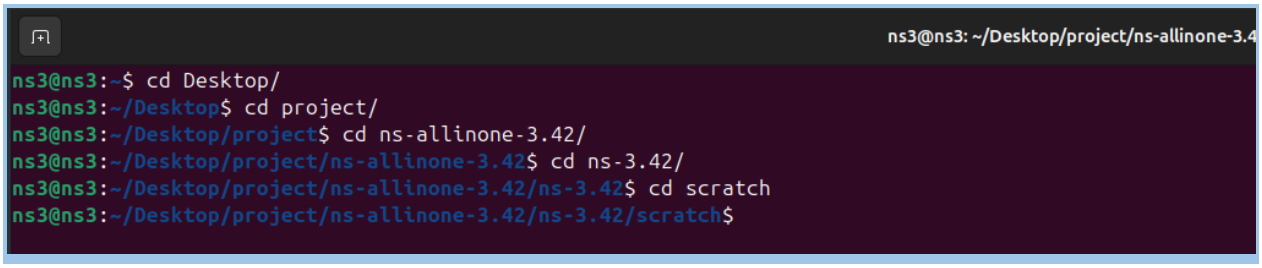
cd Desktop

cd project

cd ns-allinone-3.42

cd ns-3.42

cd scratch



Code

```
#include "ns3/aodv-module.h"
#include "ns3/core-module.h"
#include "ns3/internet-module.h"
#include "ns3/mobility-module.h"
#include "ns3/network-module.h"
#include "ns3/ping-helper.h"
#include "ns3/point-to-point-module.h"
#include "ns3/yans-wifi-helper.h"
#include <ns3/netsimulyzer-module.h>
#include <cmath>
#include <iostream>

using namespace ns3;

/**
 * @defgroup aodv-examples AODV Examples
 * @ingroup aodv
```

```
 * @ingroup examples
 */


/**
 * @ingroup aodv-examples
 * @ingroup examples
 * @brief Test script.
 *
 * This script creates 1-dimensional grid topology and then ping last node from the first one:
 *
 * [10.0.0.1] <-- step --> [10.0.0.2] <-- step --> [10.0.0.3] <-- step --> [10.0.0.4]
 *
 * ping 10.0.0.4
 *
 * When 1/3 of simulation time has elapsed, one of the nodes is moved out of
 * range, thereby breaking the topology.  By default, this will result in
 * stopping ping replies reception after sequence number 33. If the step size is reduced
```

```
 * to cover the gap, then also the following pings can be received.
 */
class AodvExample
{
  public:
        AodvExample();
        /**
        * @brief Configure script parameters
        * @param argc is the command line argument count
        * @param argv is the command line arguments
        * @return true on successful configuration
        */
        bool Configure(int argc, char** argv);
        /// Run simulation
        void Run();
        /**
        * Report results
        * @param os the output stream
        */
```

```
        void Report(std::ostream& os);

private:
        // parameters
        /// Number of nodes
        uint32_t size;
        /// Distance between nodes, meters
        double step;
        /// Simulation time, seconds
        double totalTime;
        /// Write per-device PCAP traces if true
        bool pcap;
        /// Print routes if true
        bool printRoutes;

        // network
        /// nodes used in the example
        NodeContainer nodes;
        /// devices used in the example
```

```
        NetDeviceContainer devices;
        /// interfaces used in the example
        Ipv4InterfaceContainer interfaces;

  private:
        /// Create the nodes
        void CreateNodes();
        /// Create the devices
        void CreateDevices();
        /// Create the network
        void InstallInternetStack();
        /// Create the simulation applications
        void InstallApplications();
};

int
main(int argc, char** argv)
{
        AodvExample test;
```

```
        if (!test.Configure(argc, argv))
        {
        NS_FATAL_ERROR("Configuration failed. Aborted.");
        }

        test.Run();
        test.Report(std::cout);
        return 0;
}

//-----------------------------------------------------------------------------
AodvExample::AodvExample()
        : size(10),
        step(5),
        totalTime(100),
        pcap(true),
        printRoutes(true)
{
}
```

```
bool
AodvExample::Configure(int argc, char** argv)
{
    // Enable AODV logs by default. Comment this if too noisy
    // LogComponentEnable("AodvRoutingProtocol", LOG_LEVEL_ALL);

    SeedManager::SetSeed(12345);
    CommandLine cmd(__FILE__);

    cmd.AddValue("pcap", "Write PCAP traces.", pcap);
    cmd.AddValue("printRoutes", "Print routing table dumps.", printRoutes);
    cmd.AddValue("size", "Number of nodes.", size);
    cmd.AddValue("time", "Simulation time, s.", totalTime);
    cmd.AddValue("step", "Grid step, m", step);

    cmd.Parse(argc, argv);
    return true;
```

```
}

void
AodvExample::Run()
{
	//  Config::SetDefault
("ns3::WifiRemoteStationManager::RtsCtsThreshold", UintegerValue (1));
//
	//  enable rts cts all the time.
	CreateNodes();
	CreateDevices();
	InstallInternetStack();
	InstallApplications();

	std::cout << "Starting simulation for " << totalTime << " s ...\n";
  //  AnimationInterface anim("testing.xml");
	Simulator::Stop(Seconds(totalTime));
	Simulator::Run();
	Simulator::Destroy();
}
```

```
void
AodvExample::Report(std::ostream&)
{
}

void
AodvExample::CreateNodes()
{
auto orchestrator = CreateObject<netsimulyzer::Orchestrator>
("testing.json");
netsimulyzer::NodeConfigurationHelper nodeHelper{orchestrator};
nodeHelper.Set ("Model",
netsimulyzer::models::QUADCOPTER_UAV_VALUE);

// Shows every Node in the scenario
for (auto node = NodeList::Begin (); node != NodeList::End (); node++)
        nodeHelper.Install (*node);
        std::cout << "Creating " << (unsigned)size << " nodes " << step << "
m apart.\n";
```

```
        nodes.Create(size);
        // Name nodes
        for (uint32_t i = 0; i < size; ++i)
        {
        std::ostringstream os;
        os << "node-" << i;
        Names::Add(os.str(), nodes.Get(i));
        }
        nodeHelper.Install (nodes);
        // Create static grid
        MobilityHelper mobility;
        mobility.SetPositionAllocator("ns3::GridPositionAllocator",
                                      "MinX",
                                      DoubleValue(0.0),
                                      "MinY",
                                      DoubleValue(0.0),
                                      "Z",
                                      DoubleValue(15.0),
                                      "DeltaX",
```

```
                                DoubleValue(step),
                                "DeltaY",
                                DoubleValue(0),
                                "GridWidth",
                                UintegerValue(size),
                                "LayoutType",
                                StringValue("RowFirst"));

        mobility.SetMobilityModel("ns3::GaussMarkovMobilityModel"),
        "Bounds", BoxValue(Box(-5, 5, -5, 5, 0, 15)),
        "TimeStep", TimeValue(Seconds(0.05)),
        "Alpha", DoubleValue(0.85),
        "MeanVelocity",
StringValue("ns3::UniformRandomVariable[Min=5|Max=20]"),
        "MeanDirection",
StringValue("ns3::UniformRandomVariable[Min=0|Max=6.283185307]"),
        "MeanPitch",
StringValue("ns3::UniformRandomVariable[Min=0.05|Max=0.05]"),
        "NormalVelocity",
StringValue("ns3::NormalRandomVariable[Mean=0.0|Variance=0.0|Boun
d=0.0]"),
```

```
        "NormalDirection",
StringValue("ns3::NormalRandomVariable[Mean=0.0|Variance=0.2|Boun
d=0.4]"),
        "NormalPitch",
StringValue("ns3::NormalRandomVariable[Mean=0.0|Variance=0.02|Bou
nd=0.04]");



        mobility.Install(nodes);
}

void
AodvExample::CreateDevices()
{
        WifiMacHelper wifiMac;
        wifiMac.SetType("ns3::AdhocWifiMac");
        YansWifiPhyHelper wifiPhy;
        YansWifiChannelHelper wifiChannel =
YansWifiChannelHelper::Default();
        wifiPhy.SetChannel(wifiChannel.Create());
        WifiHelper wifi;
```

```
        wifi.SetRemoteStationManager("ns3::ConstantRateWifiManager",
                                     "DataMode",
                                     StringValue("OfdmRate6Mbps"),
                                     "RtsCtsThreshold",
                                     UintegerValue(0));
        devices = wifi.Install(wifiPhy, wifiMac, nodes);

        if (pcap)
        {
        wifiPhy.EnablePcapAll(std::string("aodv"));
        }
}

void
AodvExample::InstallInternetStack()
{
        AodvHelper aodv;
        // you can configure AODV attributes here using aodv.Set(name,
value)
        InternetStackHelper stack;
```

```
        stack.SetRoutingHelper(aodv); // has effect on the next Install ()
        stack.Install(nodes);
        Ipv4AddressHelper address;
        address.SetBase("10.0.0.0", "255.0.0.0");
        interfaces = address.Assign(devices);

        if (printRoutes)
        {
        Ptr<OutputStreamWrapper> routingStream =
        Create<OutputStreamWrapper>("aodv.routes", std::ios::out);
        Ipv4RoutingHelper::PrintRoutingTableAllAt(Seconds(8),
routingStream);
        }
}

void
AodvExample::InstallApplications()
{
        PingHelper ping(interfaces.GetAddress(size - 1));
```

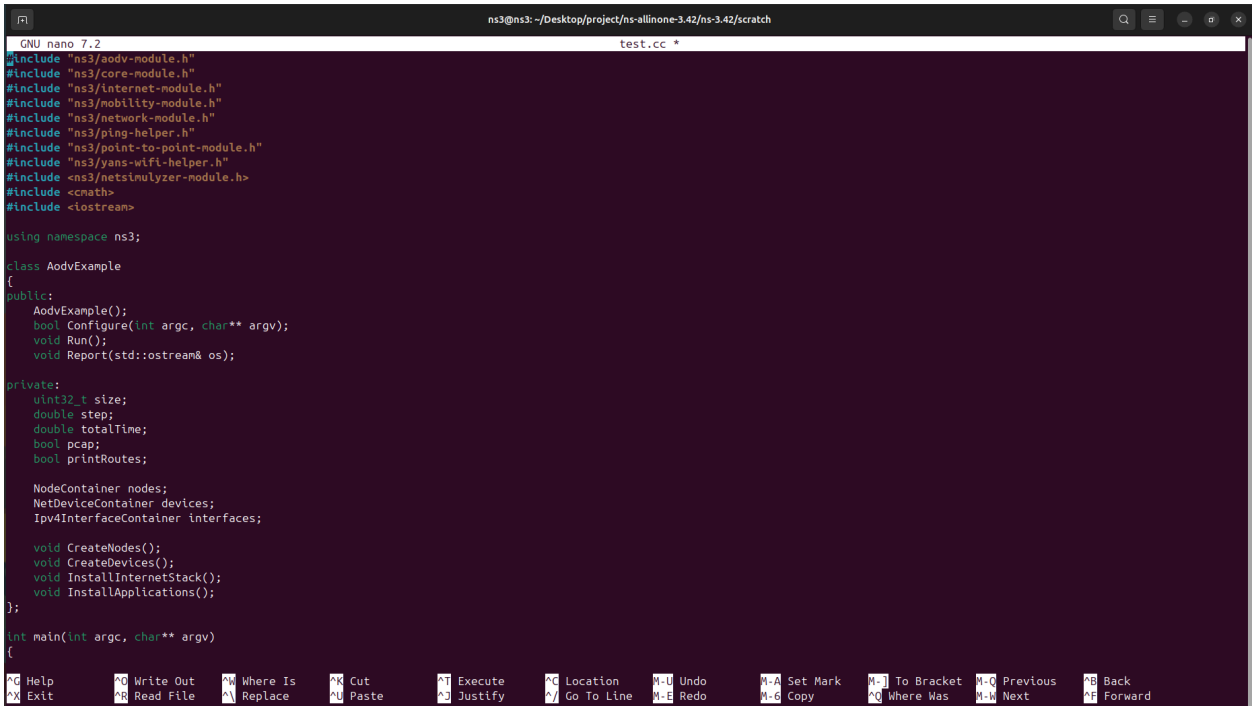
```
        ping.SetAttribute("VerboseMode",
EnumValue(Ping::VerboseMode::VERBOSE));

        ApplicationContainer p = ping.Install(nodes.Get(0));
        p.Start(Seconds(0));
        p.Stop(Seconds(totalTime) - Seconds(0.001));

        // move node away
        Ptr<Node> node = nodes.Get(size / 2);
        Ptr<MobilityModel> mob = node->GetObject<MobilityModel>();
        Simulator::Schedule(Seconds(totalTime / 3),
                            &MobilityModel::SetPosition,
                            mob,
                            Vector(1e5, 1e5, 1e5));
}
```

Step2: Write code

(Consider we are at /scratch directory)

Go to Terminal:

nano test.cc

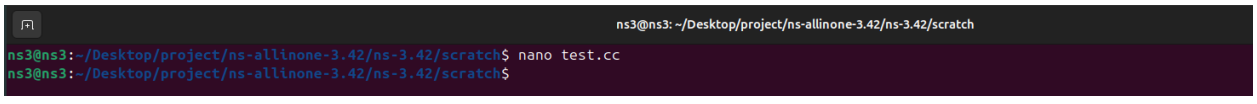


(Paste the code provided below)

Press: Ctrl+x

Write: Y

Press: Enter



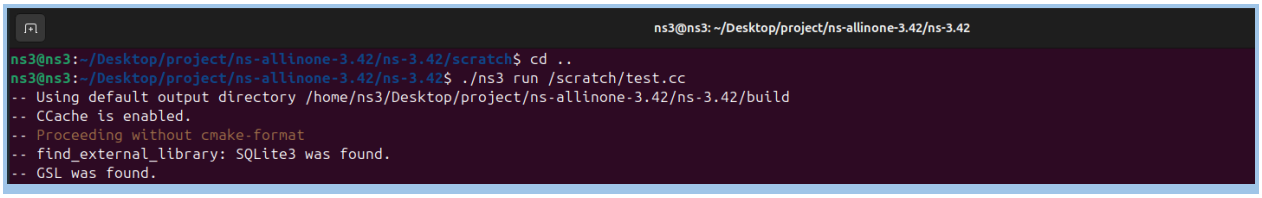
Step2: Run Code

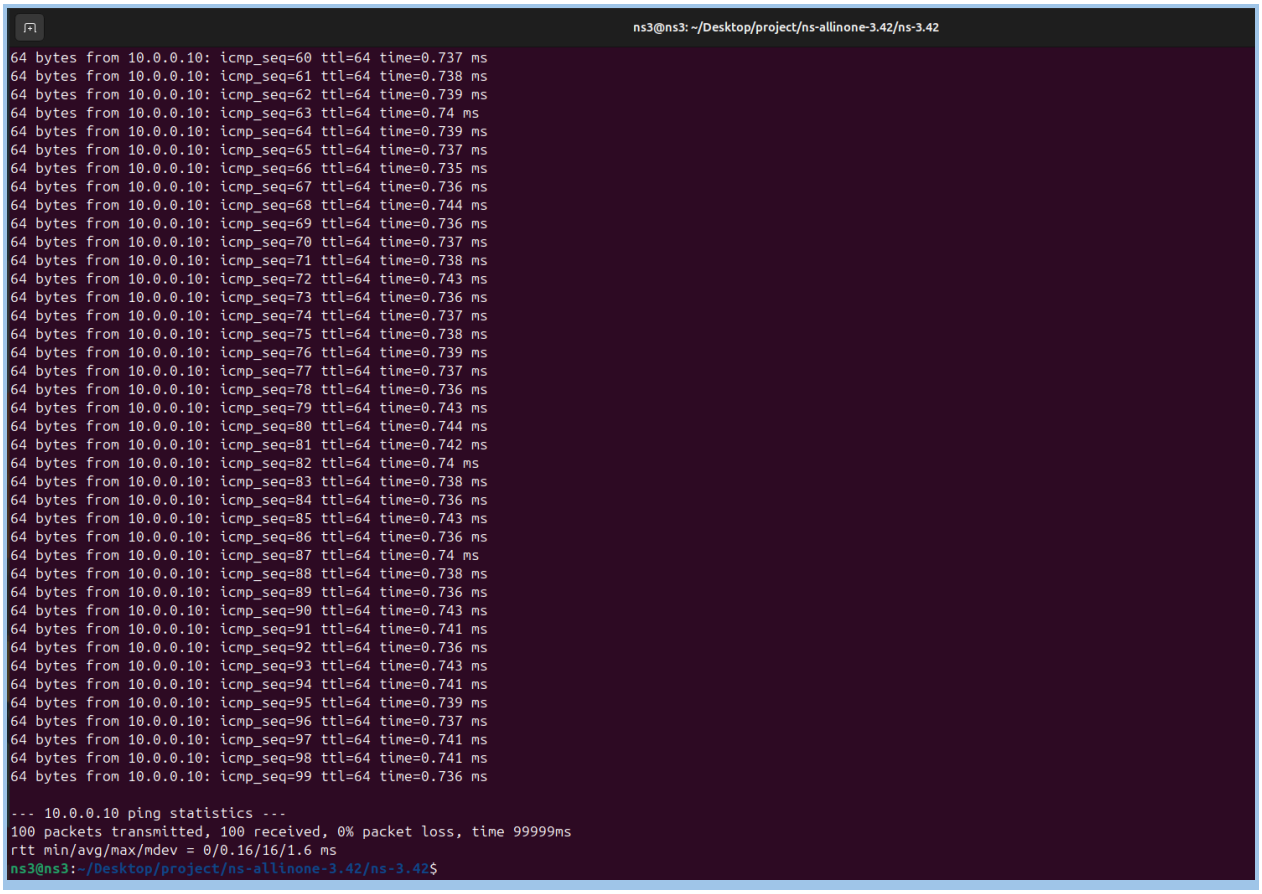
(Consider we are at /scratch directory)

Go to Terminal:

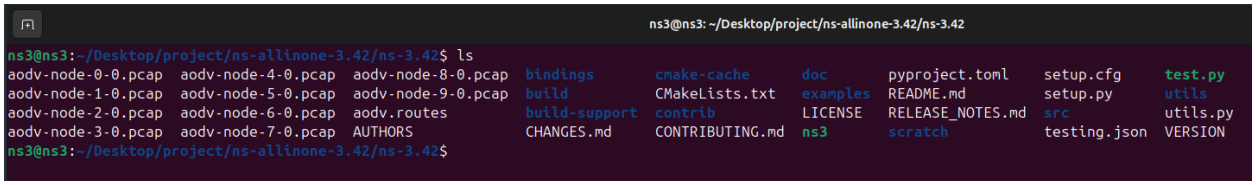
cd ..

./ns3 run scratch/test.cc





(Now we can find there is a testing.json file in the /ns-3.42 directory)



2. Running Simulation

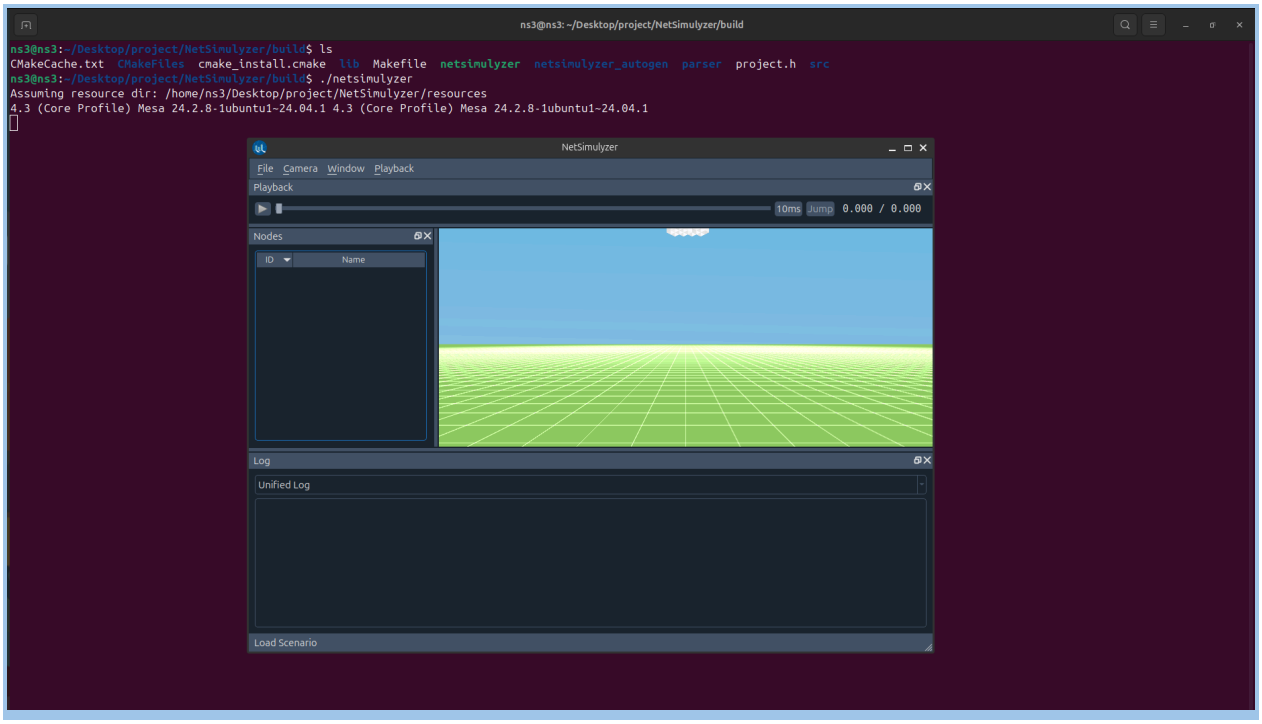
Step1: Run NetSimulyzer

(Consider we are at /username directory)

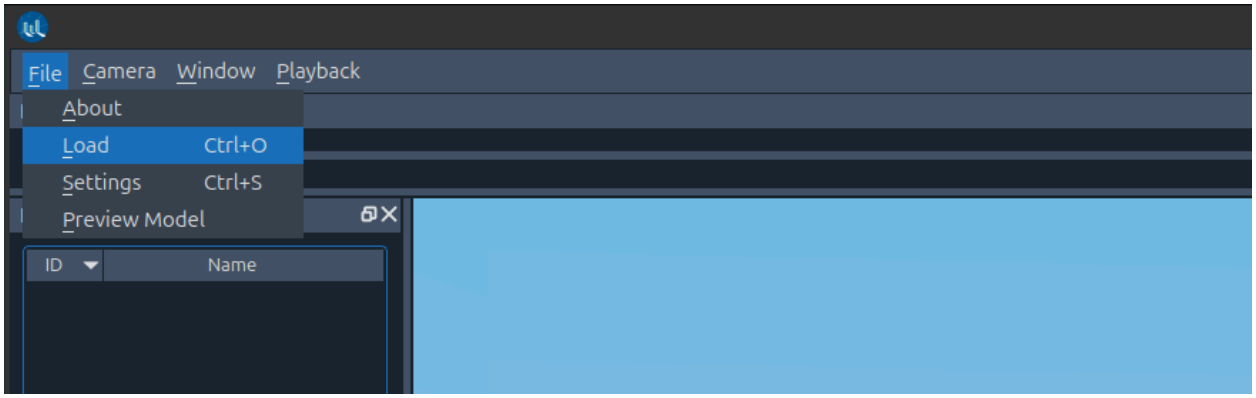
Go to Terminal:

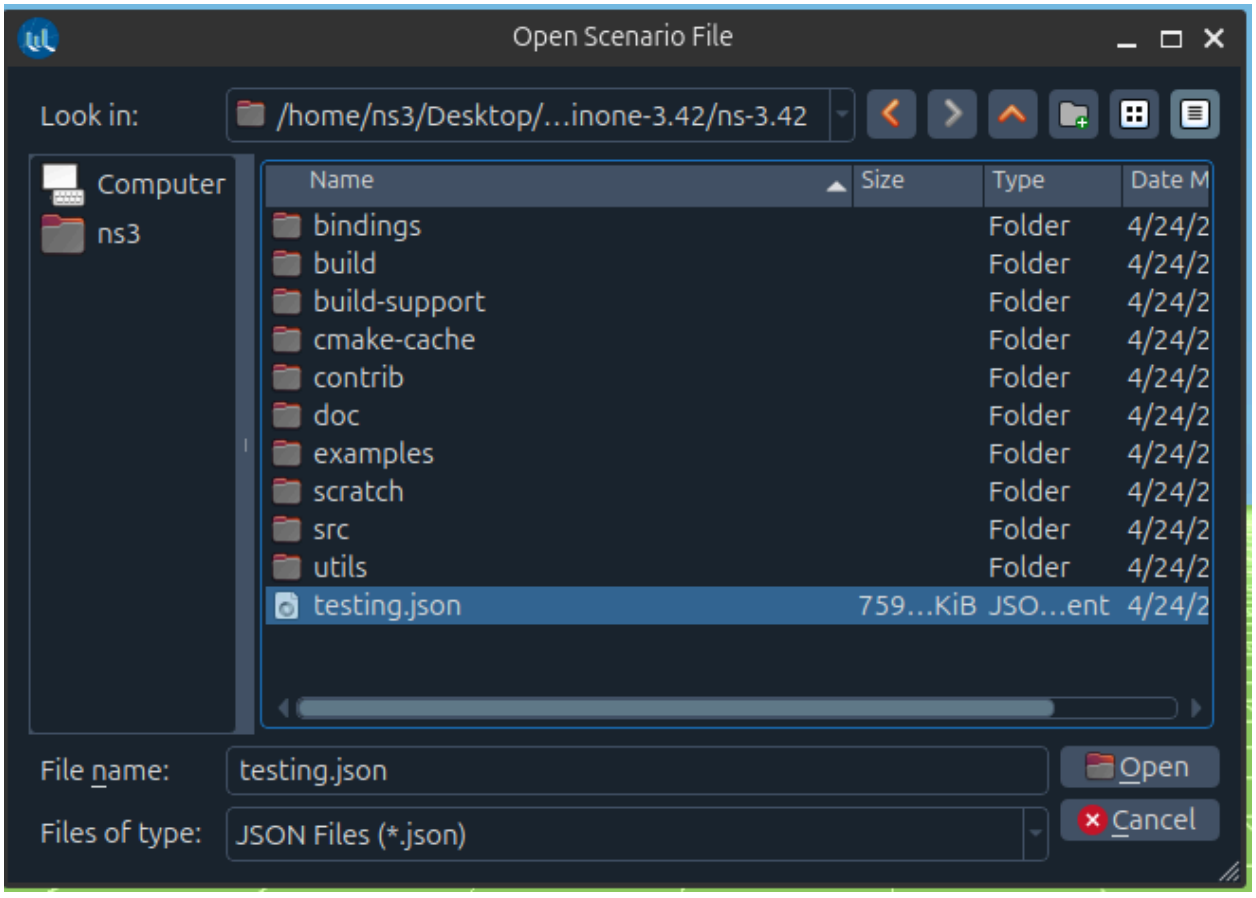
cd Desktop/project/NetSimulyzer/build

./netsimulyzer



Step2: Load The testing.json File





(Open)

Step3: Play the simulation