Nelson Chung

nelson2\_chung@compalbn.com

Abstract

The document show the technique about xxx for project – GW7557

Technique IntrO

Software Technique Document

History

|  |  |  |
| --- | --- | --- |
| Version | Author | Release |
| 0.1 | Yintao\_ling | First release |

內容

[Background 3](#_Toc42175364)

[Technique 3](#_Toc42175365)

[Reference 3](#_Toc42175366)

[Abbreviation 3](#_Toc42175367)

# Background

開源的路由器平台，作為5G的基礎之一。

# Technique



OpenWrt是適合於嵌入式裝置的一個Linux系統，提供了大量現成軟體套件且可讀寫的檔案系統。使用者能藉此選擇自己的應用程式與設定，客製化套用在路由器上。對開發者也不需要建構整個韌體來取得特定的應用程式，目前Opkg提供超過3500個套件，LuCI互動介面。

開發歷史



2002年，Linksys公司推出了一款無線路由器：WRT54G。這款路由器從諸多跡象表明，WRT54G中使用了Linux作業系統，不同於當年微處理器加WiFi晶片的標準格式。

Linux是基於GPL的協議的系統。而GPL基於 GNU 的架構下的，簡單的說，GNU就是『自由軟體』的意思，放置於網路上供人自由下載使用的軟體，提供原始程式碼與編譯完成的執行檔，並鼓勵使用者修改程式使符合使用者本身的服務需求。

在眾人的要求及法律的程序下，Linksys公布了該路由器WRT54G的開源碼。自此，許多人開始對其程式碼修改、維護，出現了許多相關管理社群：DD-Wrt、Tomato、OpenWrt。

OpenWrt的Wrt三個字，最早就是來源於WRT54G這款產品，意思是開源的路由。OpenWrt並不直接繼承Linksys的開源碼，而是把Linksys的程式作為參考，從頭到尾創作出來的一個獨立路由器系統，也成功的擴展到其它品牌的路由器上，進化成了一個通用的開源路由器系統。

後續推出了許多改版，2016年OpenWrt核心成員決定放棄OpenWrt計畫，重新啟動名為LEDE計畫。一部分原因是缺乏維護、支援型號漸漸脫節，另一部分更有傳言Qualcomm一直藉由OpenWrt作為標準開發公版，其掌握度與利益考量讓成員不適。目前名稱為OpenWrt，但程式碼源來自LEDE。

系統特點

OpenWRT是一個高度模組化、高度自動化的嵌入式Linux系統，擁有強大的網路元件和擴展性，常常被用於工業設備、電話、小型機器人、智慧家居、路由器以及VOIP設備中。同時，它還提供了100多個已編譯好的軟體，而且數量還在不斷增加，而OpenWrt SDK 更簡化了開發軟體的工序。

一般嵌入式Linux系統的開發過程，無論是PowerPC, ARM或MIPS的處理器，都必須要經過下列過程：

1. 創建Linux的交叉編譯環境
2. 建立Bootloader
3. 移植Linux Kernel
4. 建立Rootfs根文件系統
5. 安裝驅動程式
6. 安裝軟體

因為OpenWrt不同於其他的開發社群是從零的架構，它囊括了工具鏈(toolchain)、Kernal、package與rootfs。可以快速且簡單訂製出完整的韌體。

但由於OpenWrt並非路由器官方的韌體，硬體的要求與相容性可能會造成部分功能無法正常運作。

建立韌體

首先，必須要有硬體套件，除了要有能支援的SoC之外，還要有至少4MB的Flash與32MB的RAM。目前好取得的硬體為樹梅派(RaspiberryPi)，或是虛擬機VirtualBox。

首先我們需要在另一個Linux系統上做編譯，alphago server因為gcc/g++版本過時而無法操作，所以在其他伺服器上操作。



當然可以選擇其他branch，詳細的release差異可以到網頁上看。



抓下來的檔案如上，這邊簡單講一下各個功能：

|  |  |
| --- | --- |
| **BSDmakefile** | BSD系統的make檔。 |
| **config** | 放了幾個全域的編譯設定檔，在執行make menuconfig時就會用到。 |
| **Config.in** | 專案的主設定檔，同樣是make menuconfig時被用到。它又引用了許多設定檔，包括config資料夾的設定檔。 |
| **docs** | TeX/LaTeX格式的説明文檔，內容少且舊。 |
| **feeds.conf.default** | 這個檔裡列出了OpenWrt軟體倉庫的位址。  可以用scripts/feeds命令來讀取這些軟體倉庫。 |
| **.git** | git維護本地代碼的倉庫。 |
| **.gitattributes** | git的本地設定檔。 |
| **.gitignore** | git的黑白名單。 |
| **include** | 主要放mk檔。大部分的make腳本都放在這裡面。 |
| **LICENCE** | GPLv2協議。所有基於OpenWrt開發的產品，都應該按照GPLv2開放源碼。 |
| **Makefile** | 主要編譯檔。 |
| **package** | OpenWrt系統是由很多小零件拼成的，比如linux做內核，libc提供程式的運行庫，busybox提供大部分命令列工具，luci提供了網頁控制介面，dnsmasq提供了DNS、DHCP服務，dropbear提供了ssh服務……所有這些零件都被稱為package。  前面提到OpenWrt支援的軟體倉庫（我們稱它為feeds），裡面放的其實也是一個個的package。OpenWrt從必要性上來區分feeds和package。比如dnsmasq，它提供DNS、DHCP服務，是一個路由系統運行所必須的零件，所以把它放在package裡。有一些零件則是可選的，比如samba伺服器，它對路由來說沒那麼重要，就放在協力廠商的軟體倉庫裡。 |
| **READEM** | 想看就看。 |
| **rules.mk** | 這也是一個通用的make腳本，被廣泛引用到其它Makefile裡。 |
| **scripts** | 這個目錄放置了編譯OpenWrt會用到的各種輔助腳本。很快我們就會用到這裡面的feeds腳本，來完成luci的安裝。 |
| **target** | 這個目錄非常重要，它存放的是OpenWrt支援的所有平台的資訊。我們在開發和移植OpenWrt的過程中會圍繞這個目錄做很多工作。 |
| **toolchain** | 工具鏈，定義了交叉工具的編譯方法。  需要針對路由器的GCC、LD、AR等工具。 |
| **tools** | 一些工具的源碼，用途跟scripts目錄比較類似。 |

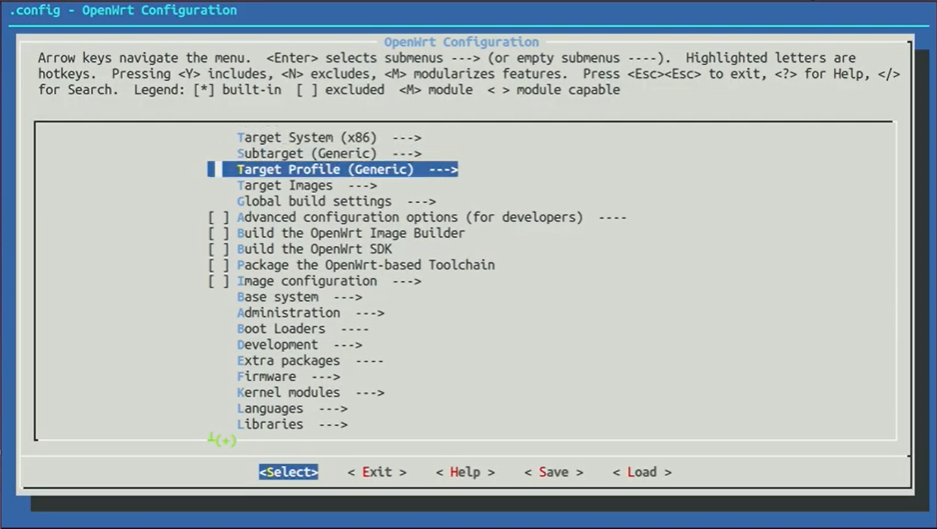
更新現有的feeds，亦可稱作package。



接下來進行編譯，OpenWrt內建了強大的編譯機制，幾乎所有的步驟都是自動化了，所以編譯過程非常簡單。



第一步會出現一個選單，用於定制你要編譯的軟體。最基本的選項是前三項，分別是指定晶片廠商(Target System)、晶片系列(Subtarget)、產品型號(Target Profile)。一般的，選擇好前三項就可以得到一個最基本的可用的路由器軟體。其它的選項OpenWrt都設有一個預設值，大家可以慢慢摸索修改。



選擇完成之後保存退出，它會在openwrt根目錄下生成一個 .config文件。這個檔包含了本次設定的全部配置。

第二步make開始編譯。OpenWrt是由許許多多的零件組合成的。每個零件的編譯過程都類似下面這樣：先從網上下載原始程式碼放到dl目錄下，然後把原始程式碼解壓到build\_dir裡，如果有patch的話，再給原代碼打上補丁，接下來就是編譯代碼，編譯成功之後，結果檔被放到bin目錄下（部分結果檔也會被放到staging目錄下）。

|  |  |
| --- | --- |
| make options | |
| **make oldconfig** | |
|  | 以前一次config檔為基礎做編譯，會顯示期間變化的部分。 |
| **make kernel\_menuconfig** | |
|  | Linux kernel 設置選單，一般而言不須更改。 |
| **make -j4** | |
|  | 平行編譯，4為數量。可以提升編譯速度。若不清楚數字的話可以抓CPUx2的值。 |
| **make V=s** | |
|  | 顯示詳細編譯經過。 |
| **make target/linux/compile** | |
|  | 只編譯kernel層的內容。 |
| **make target/linux/{clean,prepare,compile}** | |
|  | 會清除kernel層編譯目錄、重新抓取dl壓縮包、再編譯kernel層。 |
| **make package/package\_name/{clean,prepare,compile}** | |
|  | 針對單一package重載並編譯。 |
| **make clean** | |
|  | 清除build\_dir相關物件。.config、tmp、staging等會保留。 |
| **make distclean** | |
|  | 清除所有編譯物件，.config、toolchain等都會移除。 |
| **make dirclean** | |
|  | 少用到，會刪除build\_dir、tmp、staging。 |

通常第一次編譯過程偏長，一是OpenWrt編譯的時候會從網上下載各種各樣的套裝軟體（包括Linux kernel）。二是OpenWrt要產生交叉編譯工具鏈（Cross Compile Toolchain）。一堆編譯工具，像編譯器（GCC)、連結器（LD）等等，這些程式是放在其它平臺上（非正在執行的機器）上運行的。交叉編譯工具鏈編譯完成之後，會放在staging目錄下。

編譯後會產生下列相關檔案：

|  |  |
| --- | --- |
| **dl** | 下載目錄。OpenWrt自身所帶的代碼非常少，絕大部分代碼都要在編譯過程中去網上下載。 |
| **.config** | 這是make menuconfig生成的專案配置。 |
| **tmp** | 暫存檔案。OpenWrt在執行任何配置、編譯過程前，會先掃描所有的toolchain、package、target路徑，生成索引、緩存之類的暫存檔案，起加速作用。如果你改動了專案結構或配置選項，但在menuconfig裡卻看不到你的改動，可以刪除這個目錄。 |
| **build\_dir** | OpenWrt的編譯都發生成build\_dir下面。包括主機工具，交叉編譯工具鏈（toolchain），Linux內核，各種套裝軟體等等。這個目錄也可以隨時刪除。 |
| **staging** | staging也是一個臨時目錄，是一些編譯生成的中間結果，像是編譯後的Toolchain與package，之後編譯就可直接引用此項目，不必花費而外時間重新編譯。 |
| **bin** | 編譯的最終結果，包括路由器軟體，以及可安裝的套裝軟體（.ipk文件）。 |
| **feeds** | 使用過scripts/feeds命令產生。用於配置OpenWrt的協力廠商套裝軟體，類似CentOS裡的yum、Ubuntu裡的apt、Python裡的pip。 |

編譯完成後，在/bin/xxxx目錄下會產生一些文件：



目錄下可能會有數十樣bin檔，事實上這些檔案有99%是相同的，只有dts上面的差異。dts是廠商對版子的描述文件，紀錄內存大小、支援I/O、起始位置或中斷訊號等等。利用這些參數交給kernel處理，讓linux能保持相容性。

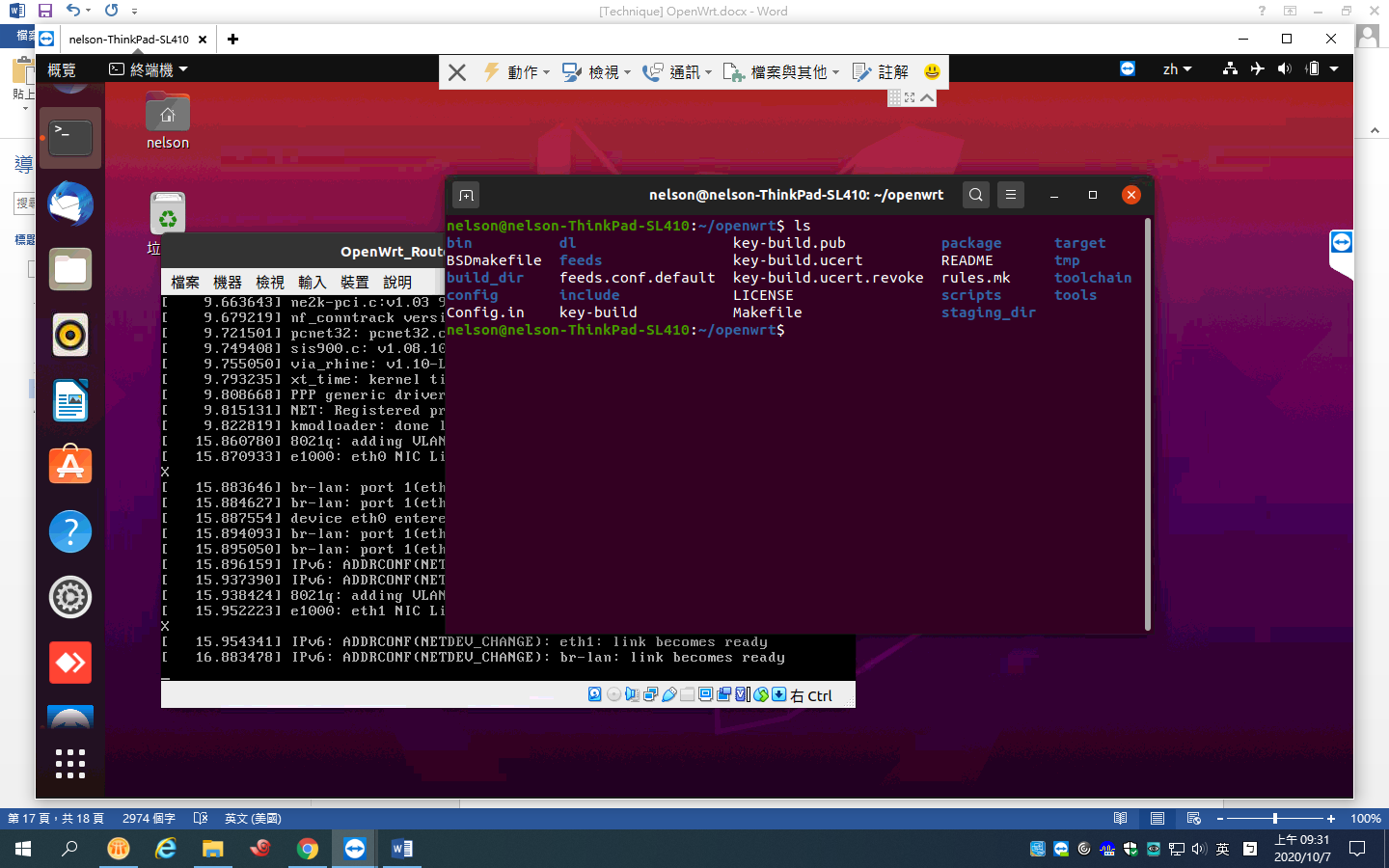
名稱取名也有它得含意：

|  |  |
| --- | --- |
| **ramips** | 這格會標示CPU晶片系列與它的IS形式。 |
| **mt7620** | 該晶片的型號。 |
| **ex2700** | 路由器廠商與路由器產品型號。 |
| **squashfs** | Rootfs的形式，squashfs只能讀且壓縮過的文件系統。 |
| **sysupgrade/**  **factory** | 皆為openwrt支援形式。若路由器本身已裝過openwrt可用sysupgrade。但有些路由器會做軟體偵測，factory版本可以躲過這些問題，如果要安裝成openwrt的可用factory版本。 |

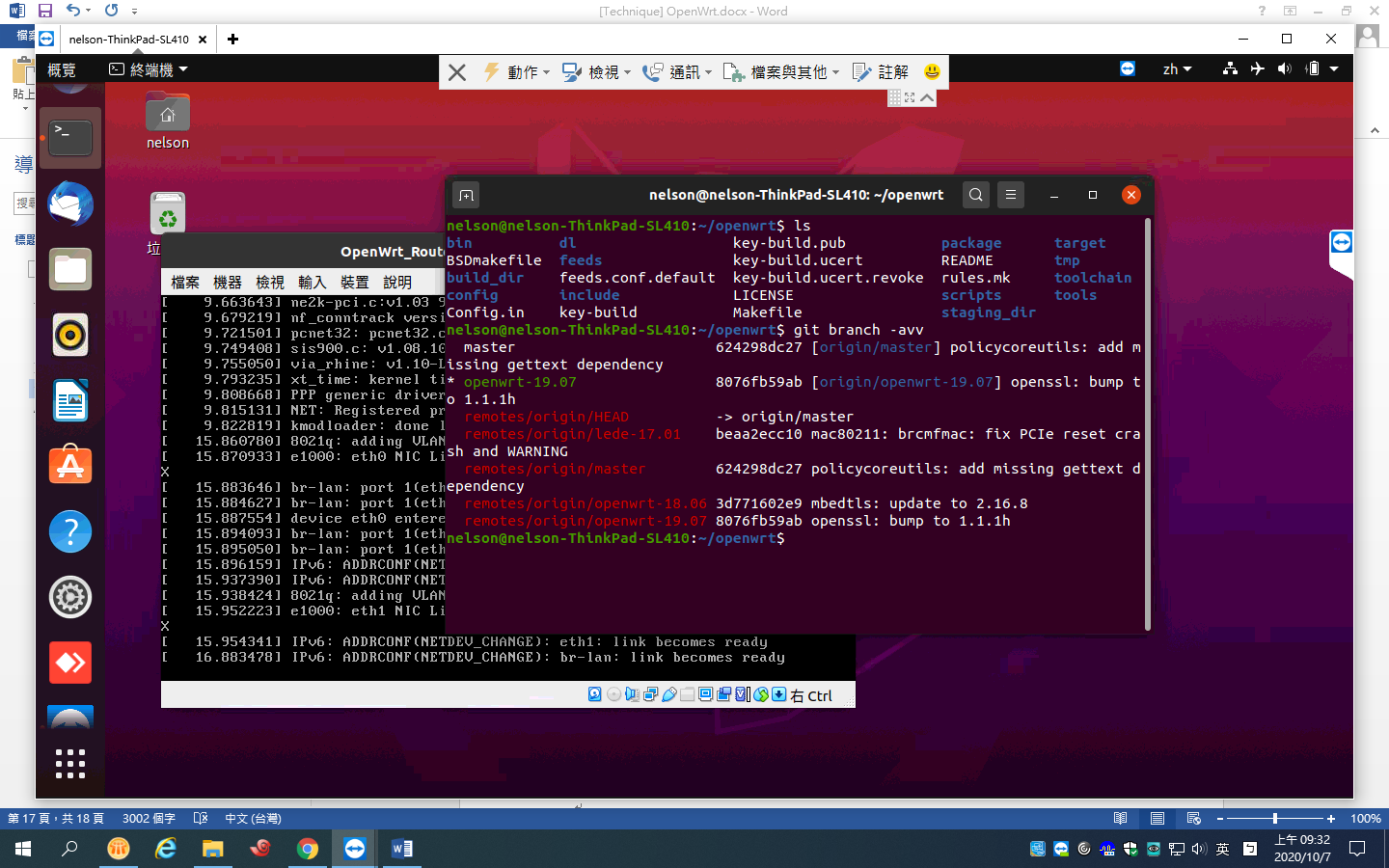
實際操作 – 架設虛擬機

這邊使用VirtualBox作為實際操作的例子。

首先將openwrt相關的文件git下來：



可以更換不同的release版本，2020/10/06時最新stable版為19.07。



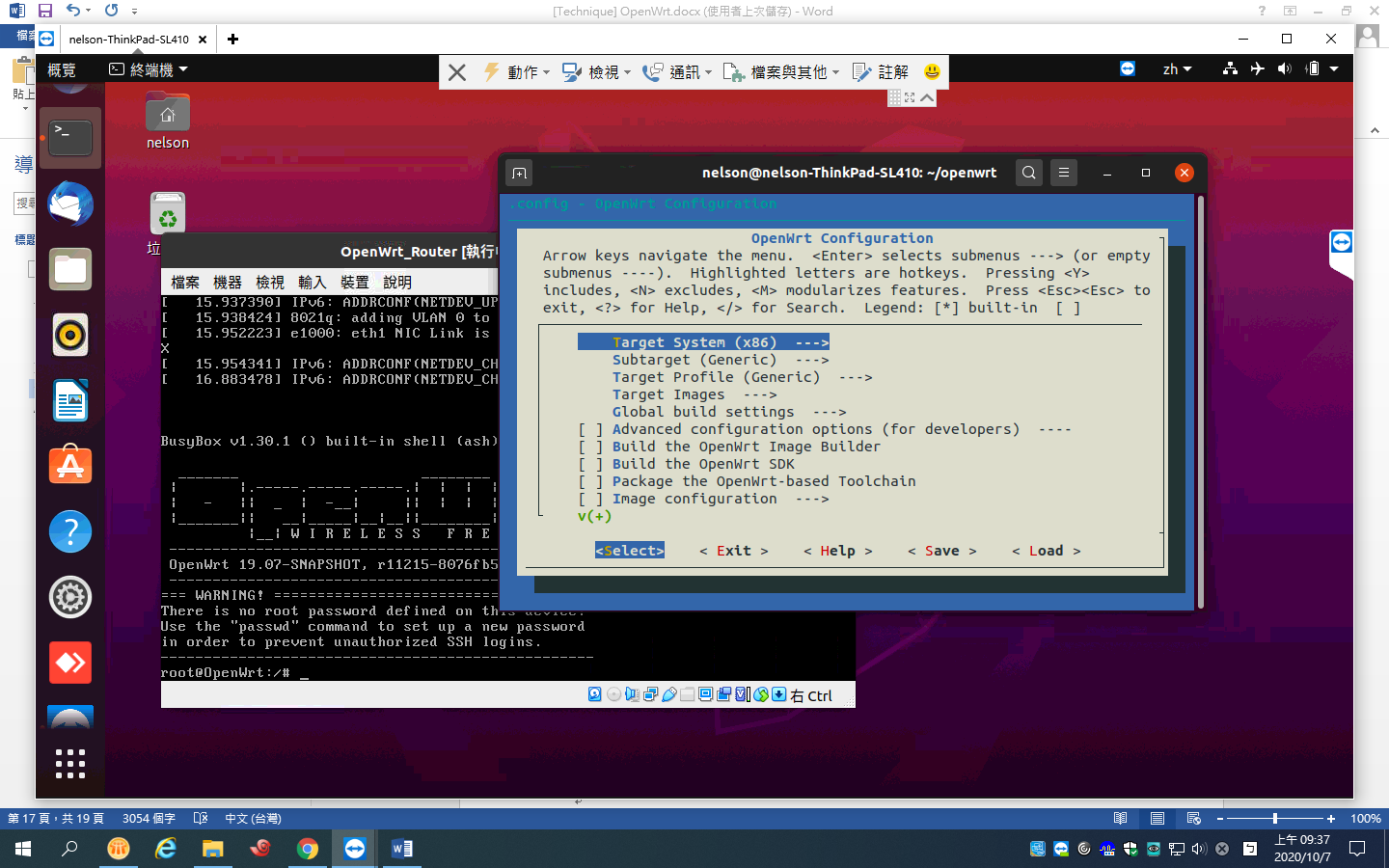
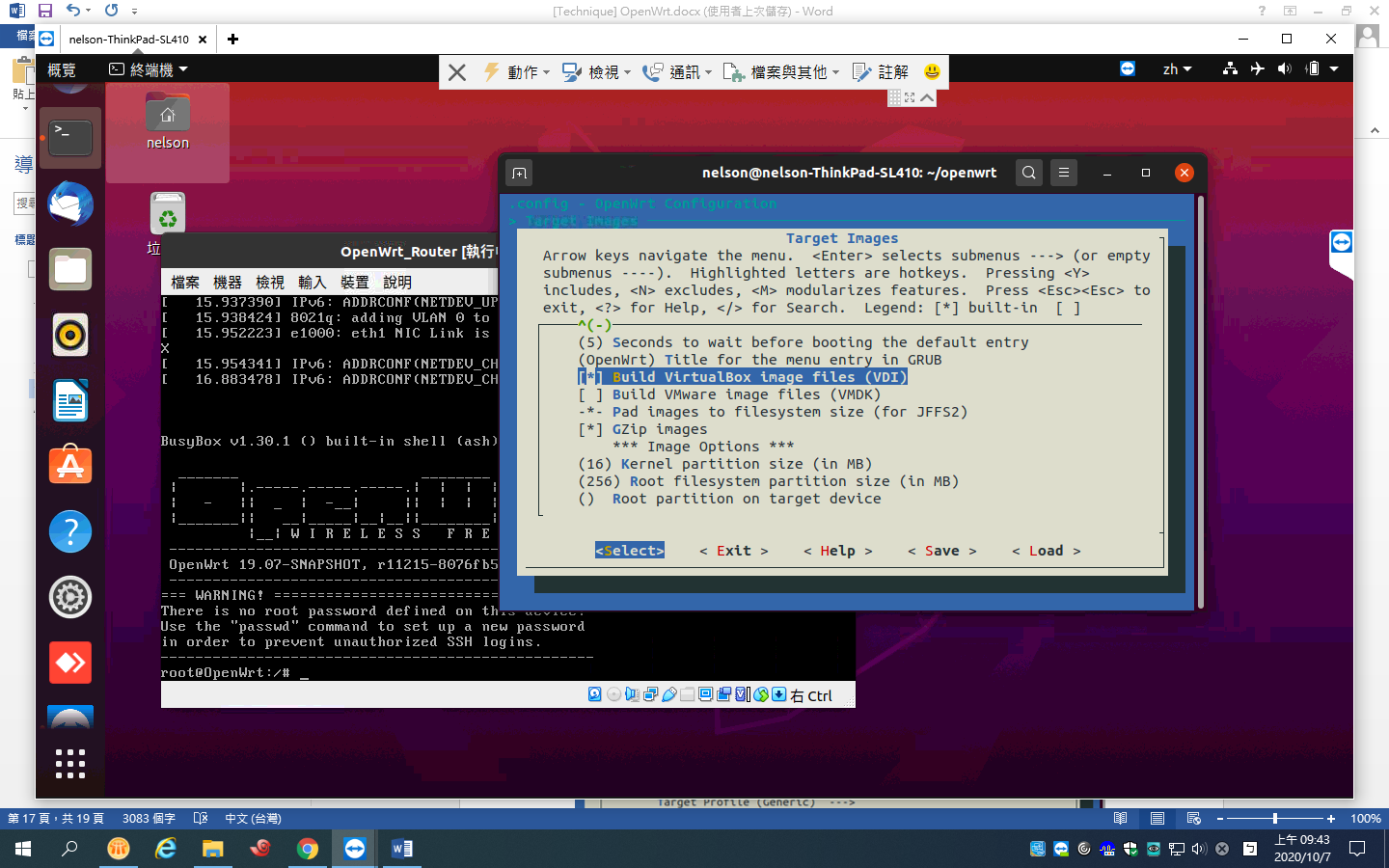
執行./script/feeds來更新與安裝軟體庫。



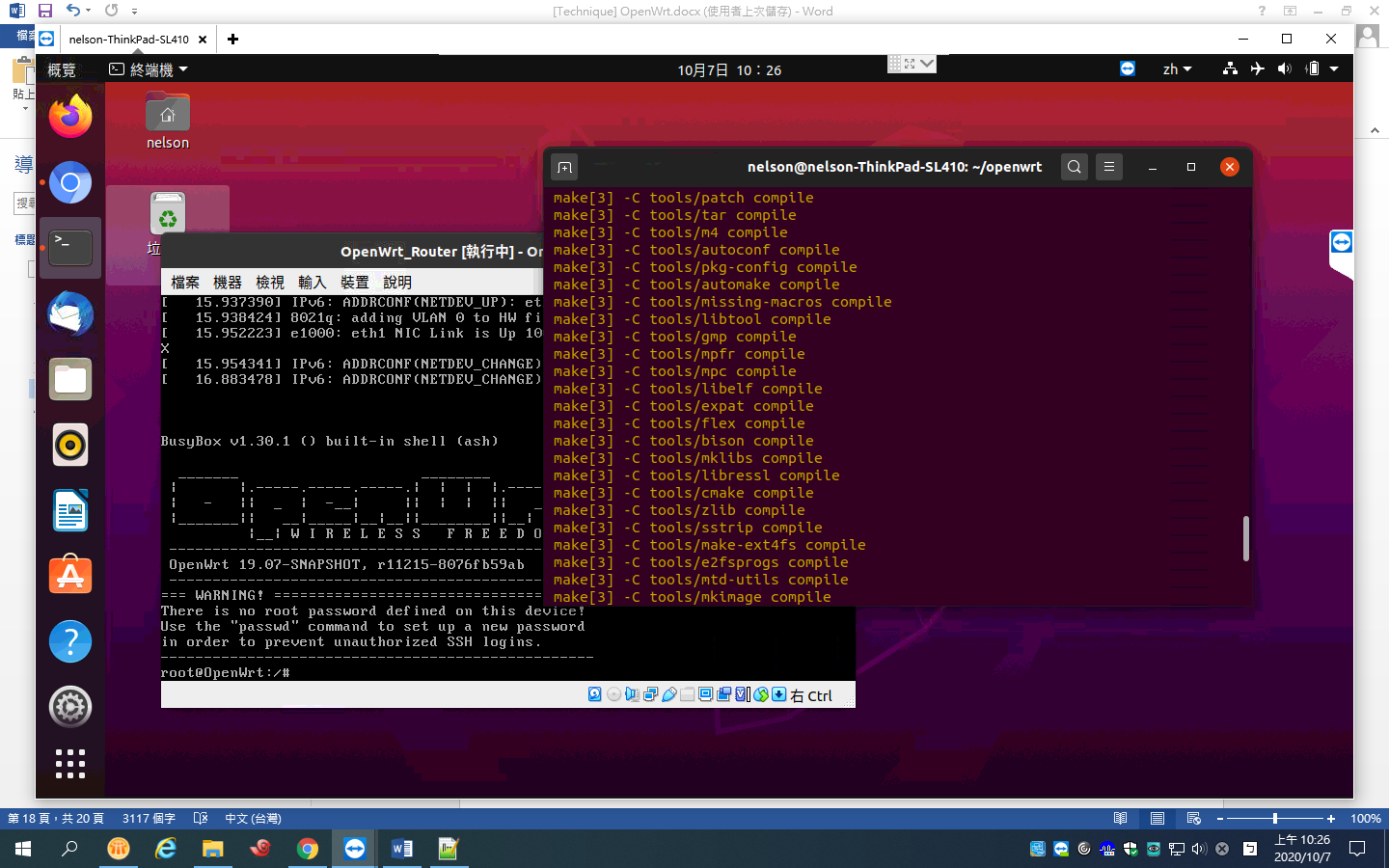
完成後可以開始設定編譯環境，輸入make menuconfig。

根據Virtualbox環境設定Target與Profile為 **x86-Generic-Generic**。

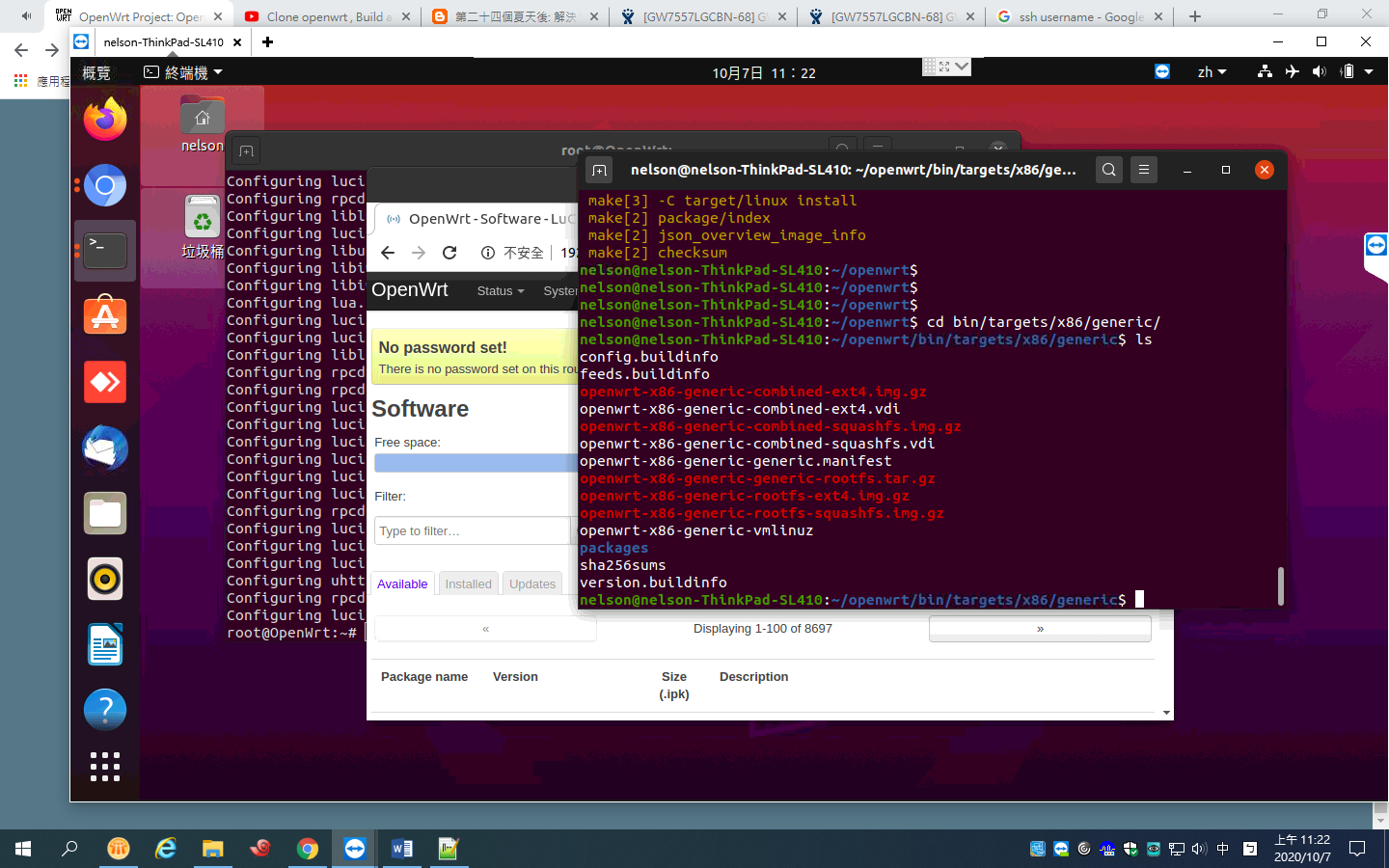
進到 Target Image選單，確定選項Build VirtualBox image files有勾選。

完成後可以開始建立鏡像檔，執行make。第一次執行可能要花1-2個小時。



完成編譯後，在 //bin/targets/x86/generic下會有下列檔案：

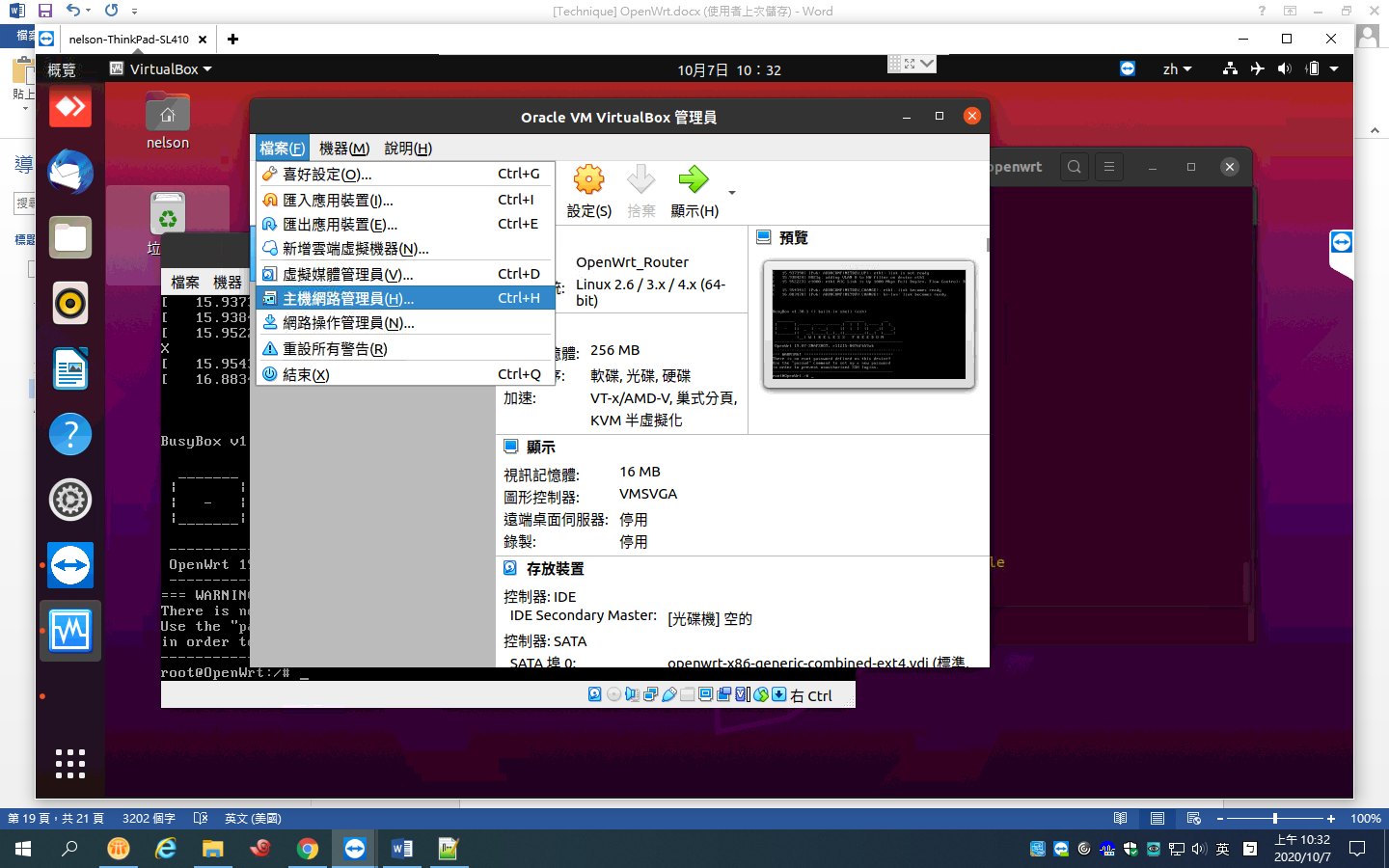


我們需要用的是結尾 .vdi的檔案，畫面中兩檔案差別在於檔案系統類型。

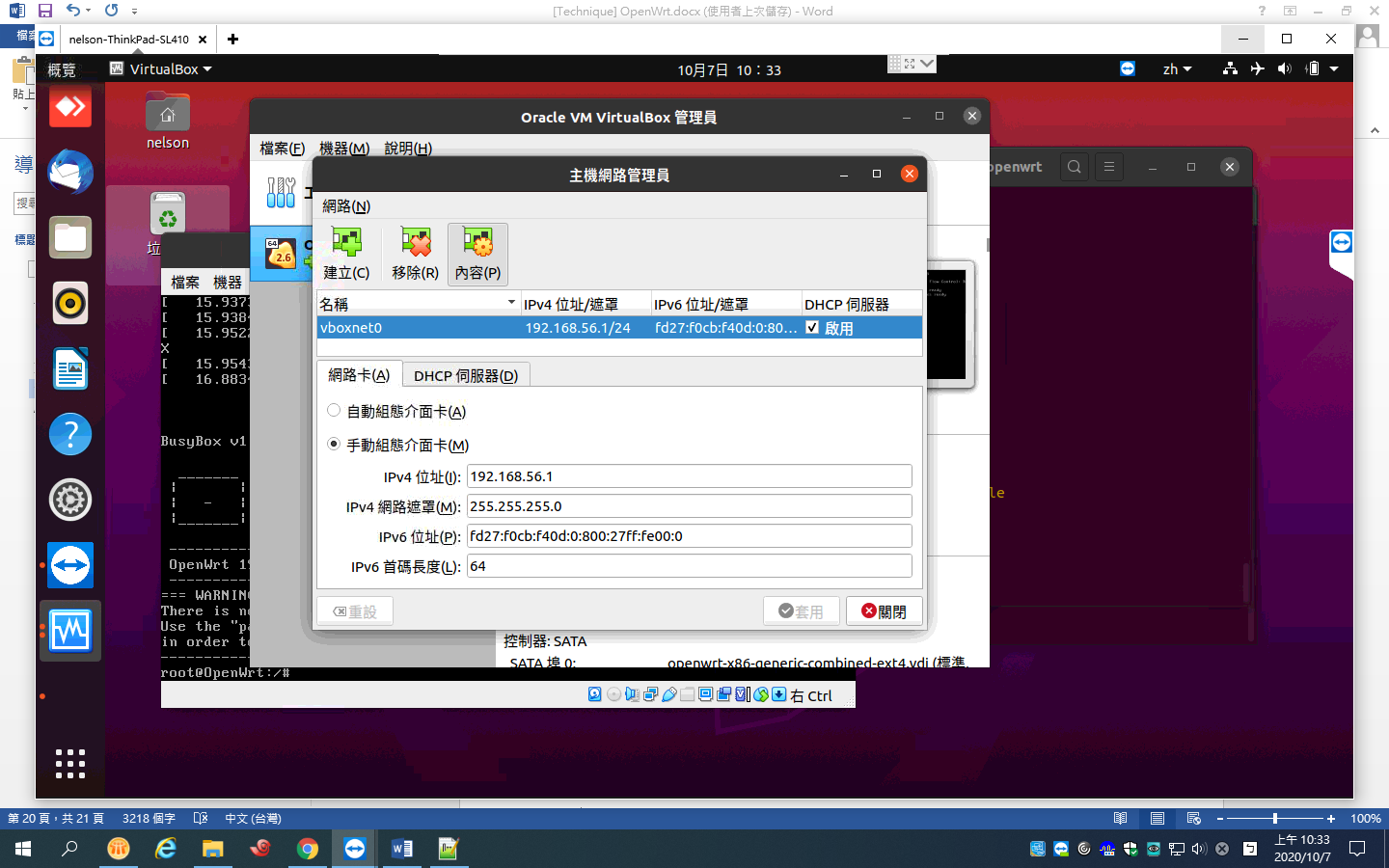
接下來設定VirtualBox環境設定 (路由器接線)。

在VirtualBox VM管理員下點擊 檔案>主機網路管理員。

這主要是管理VM與主電腦間的連接，目的是方便主電腦對VM直接管理。



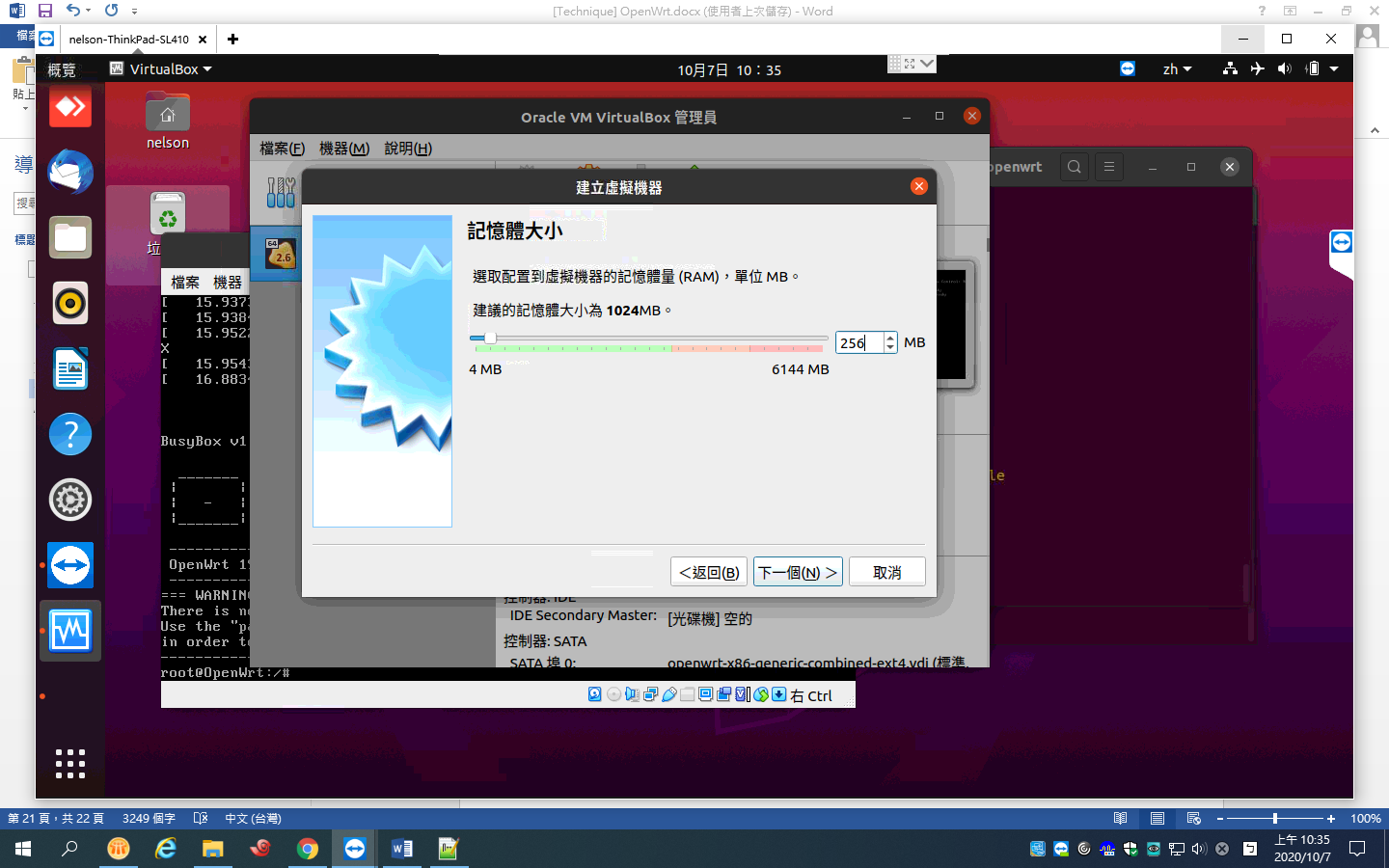
新增新的網路，基本上預設就行了，主機IP位置為192.168.56.1。



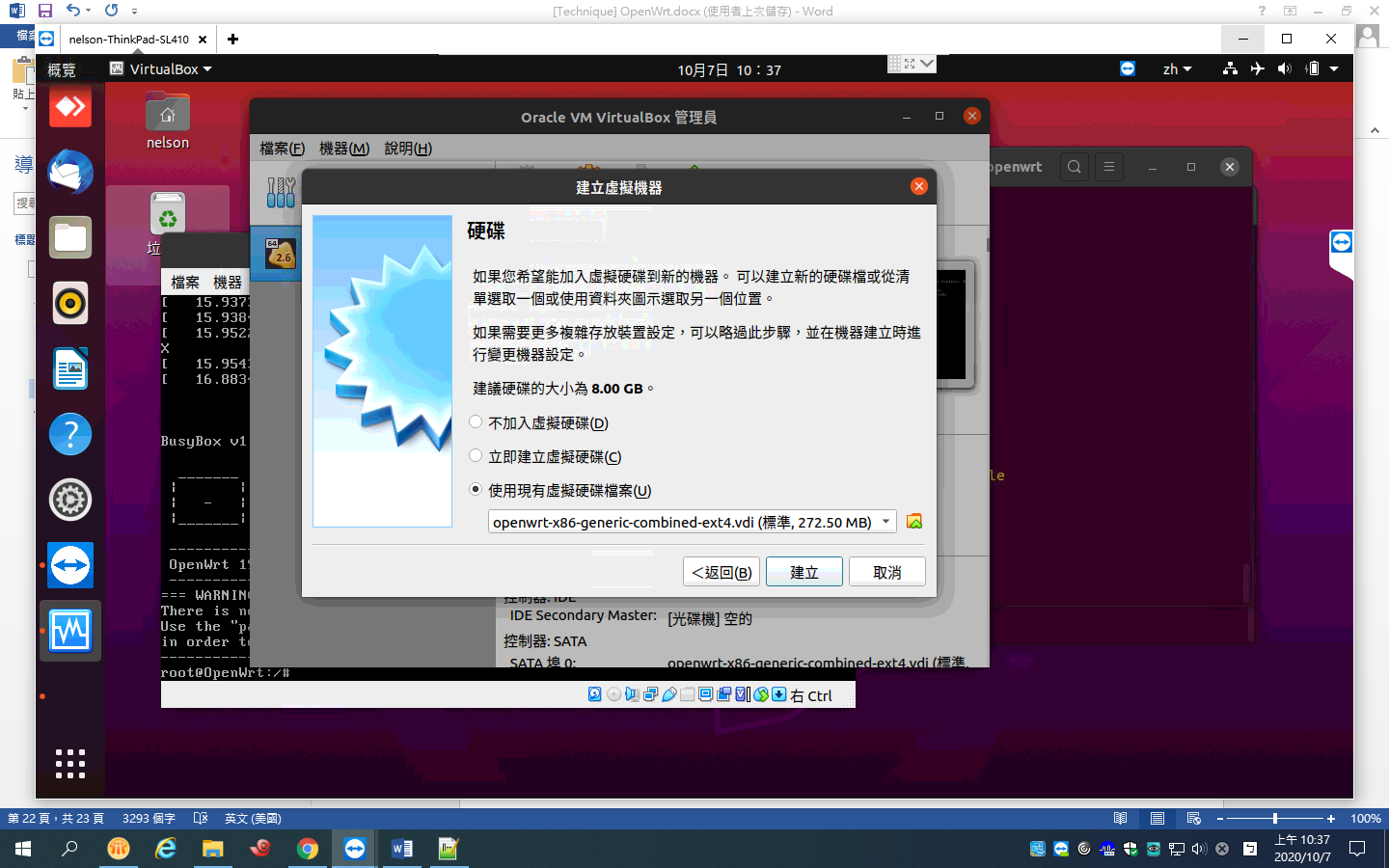
接下來新增機器：在主頁面點擊新增，輸入對應名稱與系統類型版本。



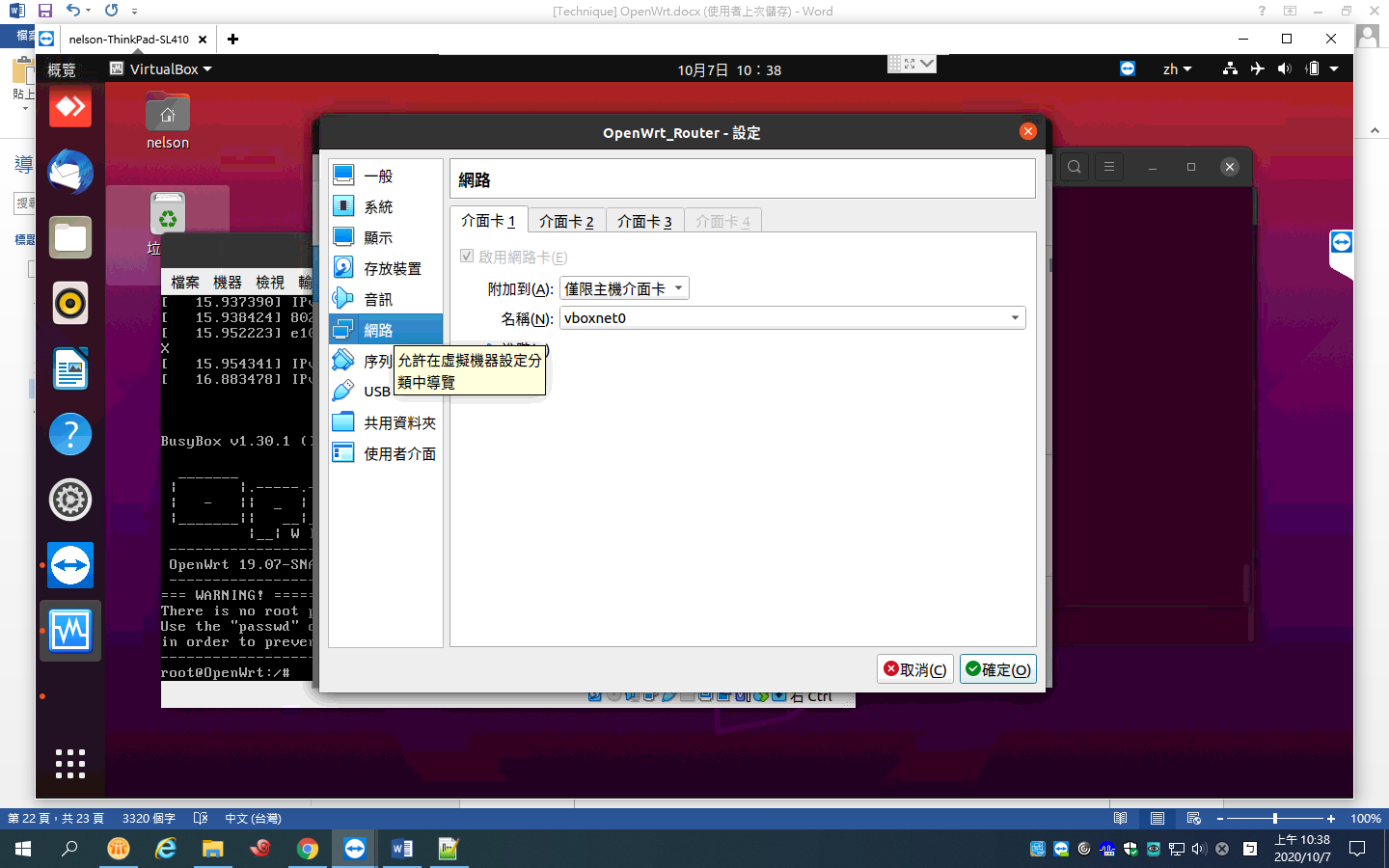
接下來設定記憶體大小，官方推薦至少128MB以上。



硬碟使用現有虛擬硬碟檔案，選取剛剛建立的vdi檔。



完成後會建立新的機器，接下來要對它進行設定，點擊設定>網路。

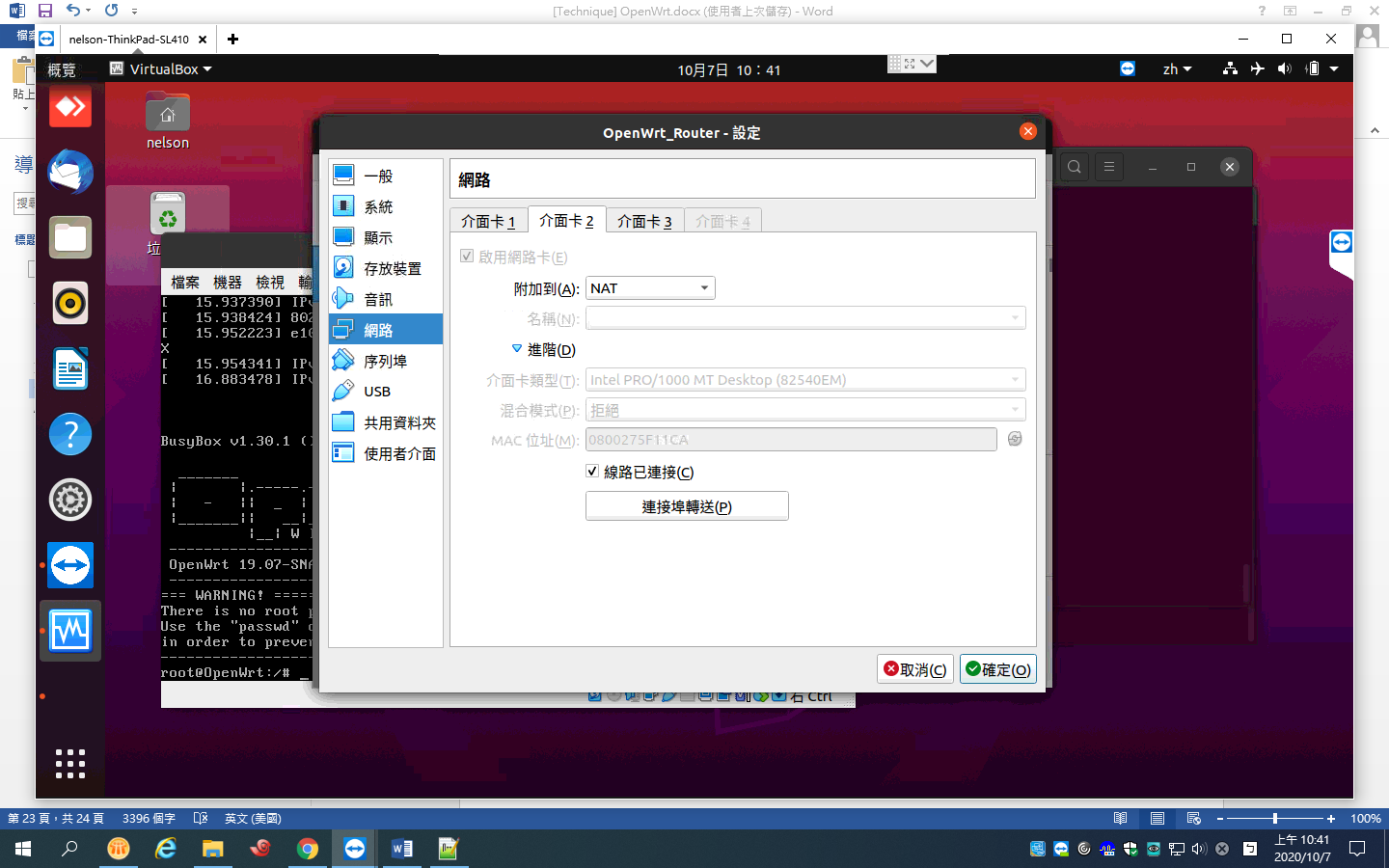


這邊要給它三張網卡：

第一張為**僅限主機介面卡**，負責虛擬路由器與主機間的連接。

第二張為**NAT**，為虛擬路由器的WAN端。

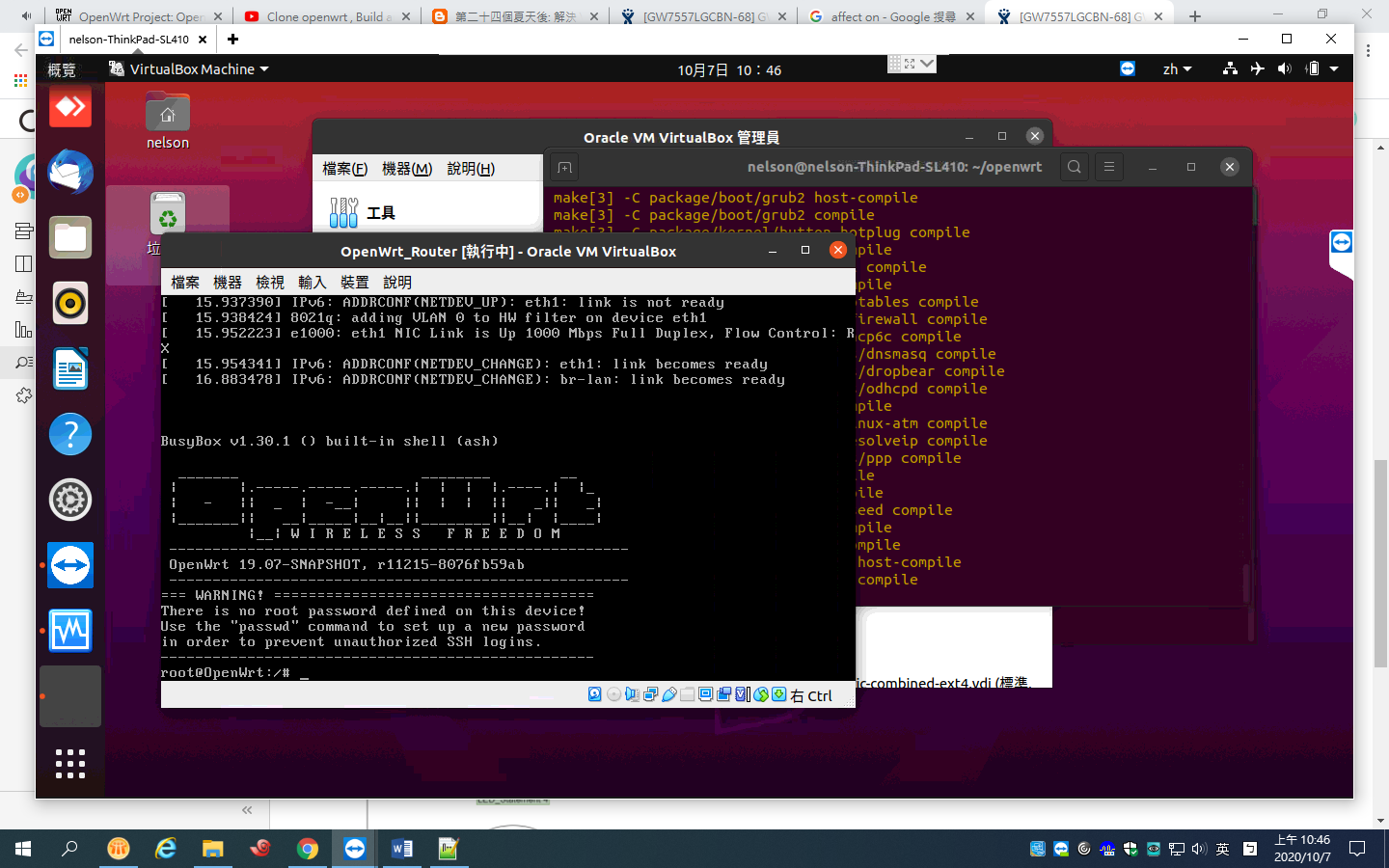
第三張為**內部網路卡**，為虛擬路由器的LAN端。  
底下選項基本上預設就行了，除非你有其他虛擬機要連它。

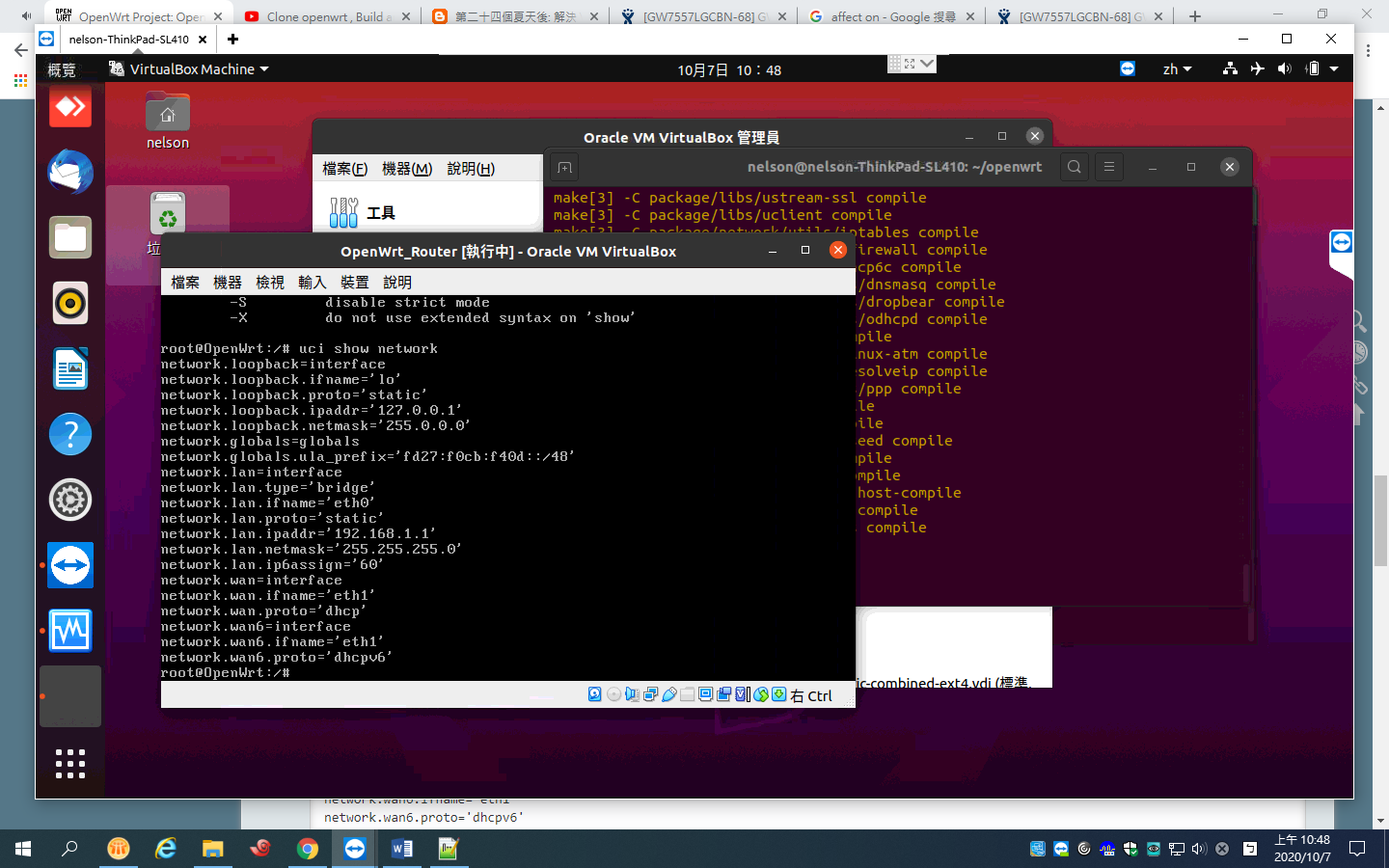


完成後就成功架設虛擬路由器了！

實際操作 – OpenWrt 設定

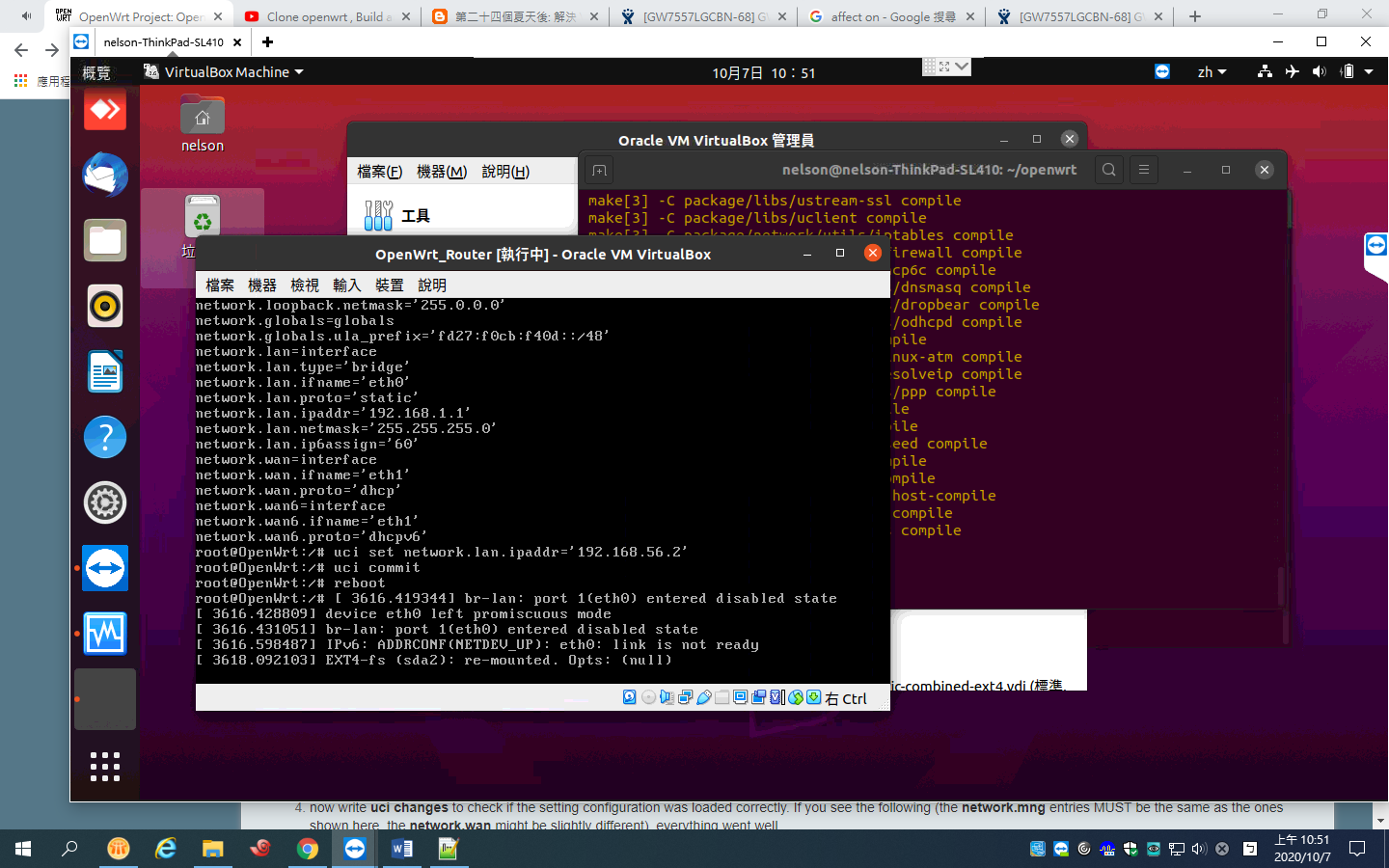
開機後按Enter，若出現下列介面代表成功了。



輸入 uci show network 來確認目前網路設定。

============== 虛擬機 僅限主機介面卡設定 =============

這邊為了虛擬機與主電腦ssh連接，將預設LAN的位置改為主機端網段。



這樣就不用在機器的介面下處理設定了。

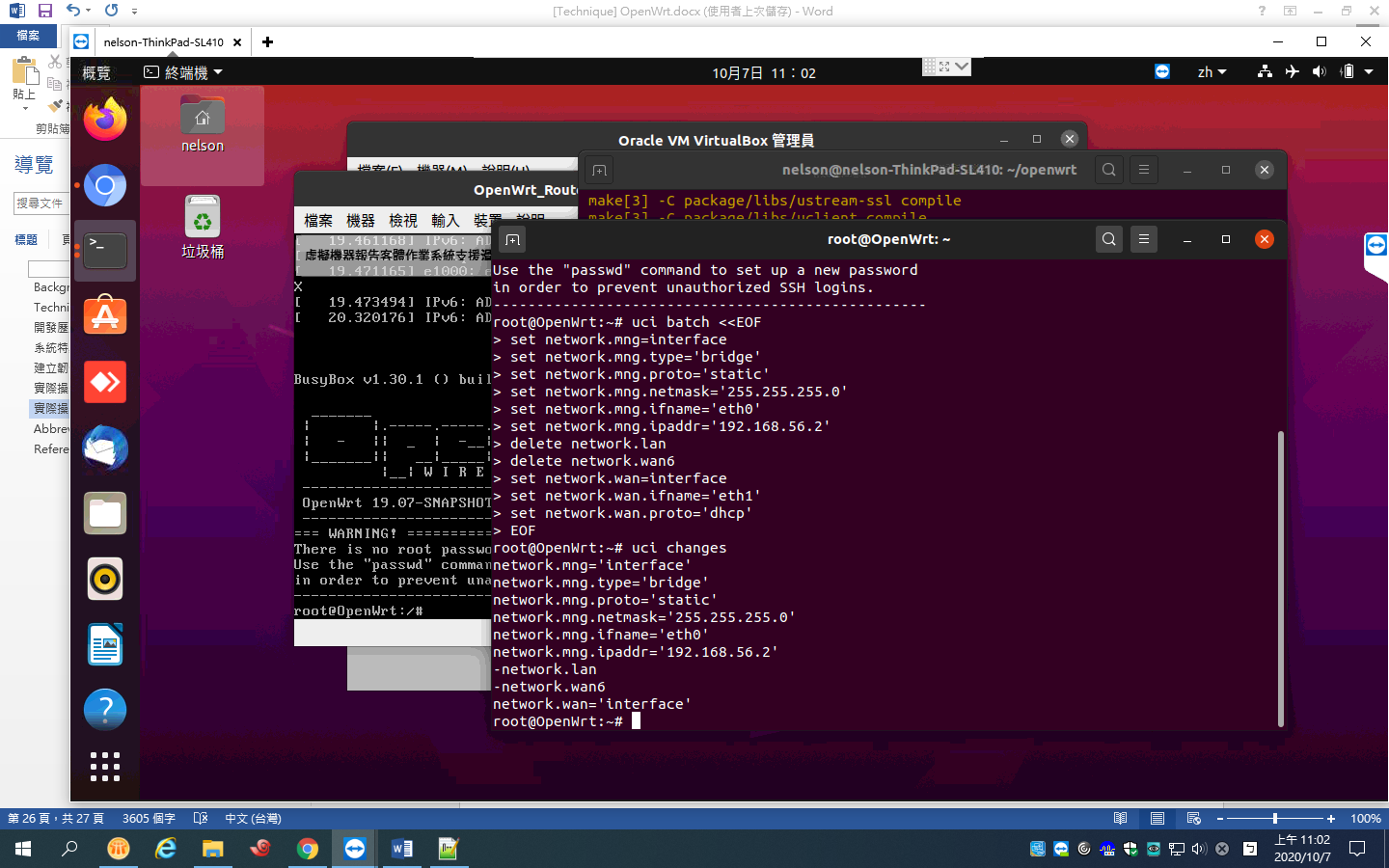


接下來要新增Management Port並把LAN port 還給路由器使用：

複製貼上下列指令，新增mng給192.168.56.2使用：

|  |
| --- |
| uci batch <<EOF  set network.mng=interface  set network.mng.type='bridge'  set network.mng.proto='static'  set network.mng.netmask='255.255.255.0'  set network.mng.ifname='eth0'  set network.mng.ipaddr='192.168.56.2'  delete network.lan  delete network.wan6  set network.wan=interface  set network.wan.ifname='eth1'  set network.wan.proto='dhcp'  EOF |

指令 uci changes 可以查看更改的設定。



確認好就 uci commit ，再重新開機。

新增LAN Port 給路由器，複製貼上下列指令，好了重開機。

也可以用vim修改 /etc/config/network檔。

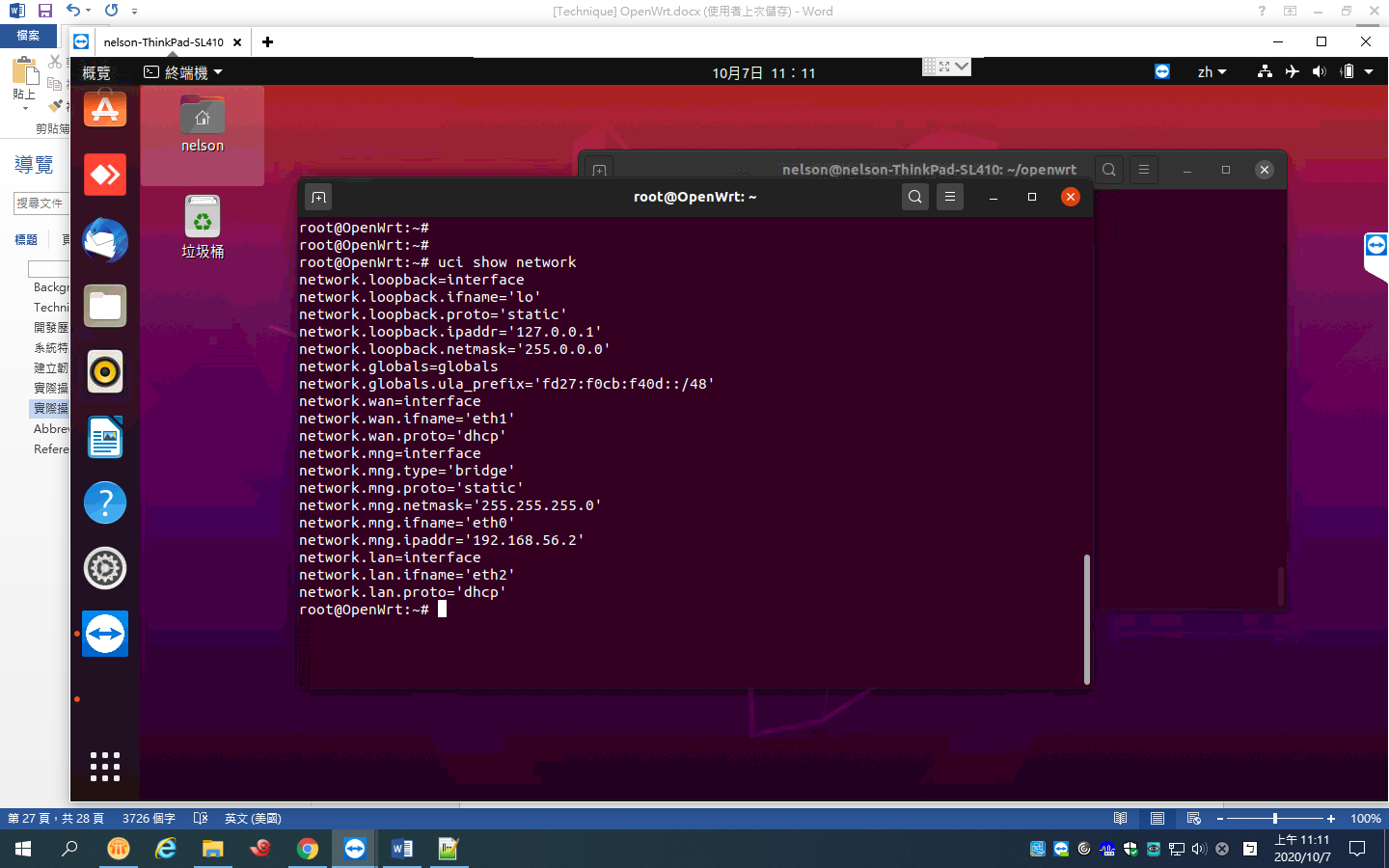
|  |
| --- |
| uci batch <<EOF  set network.lan=interface  set network.lan.ifname='eth2'  set network.lan.proto='static'  set network.mng.ipaddr='192.168.1.1'  set network.mng.netmask='255.255.255.0'  EOF  uci commit |

這樣就設定完了，可以ping看看wan端有沒有通。

現在機器上 network.mng 為僅限主機介面卡。

network.wan 為NAT卡，能連到外面網路。

network.lan 為內部網路卡，能發配位置給連到這台機器的設備。

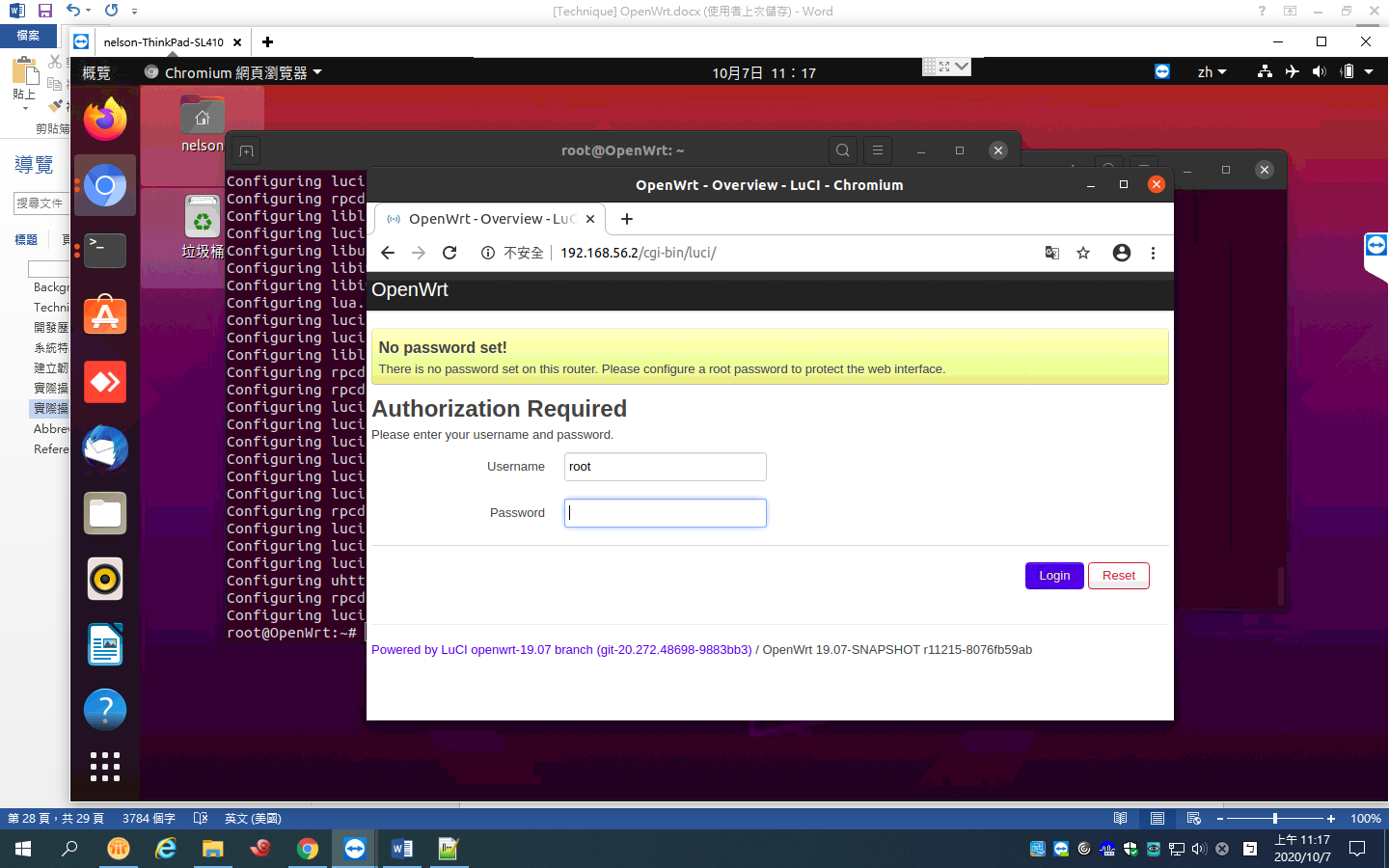


=============================================

接下來更新opkg 資料庫及下載luci網頁圖形介面。

|  |
| --- |
| opkg update  opkg install luci |

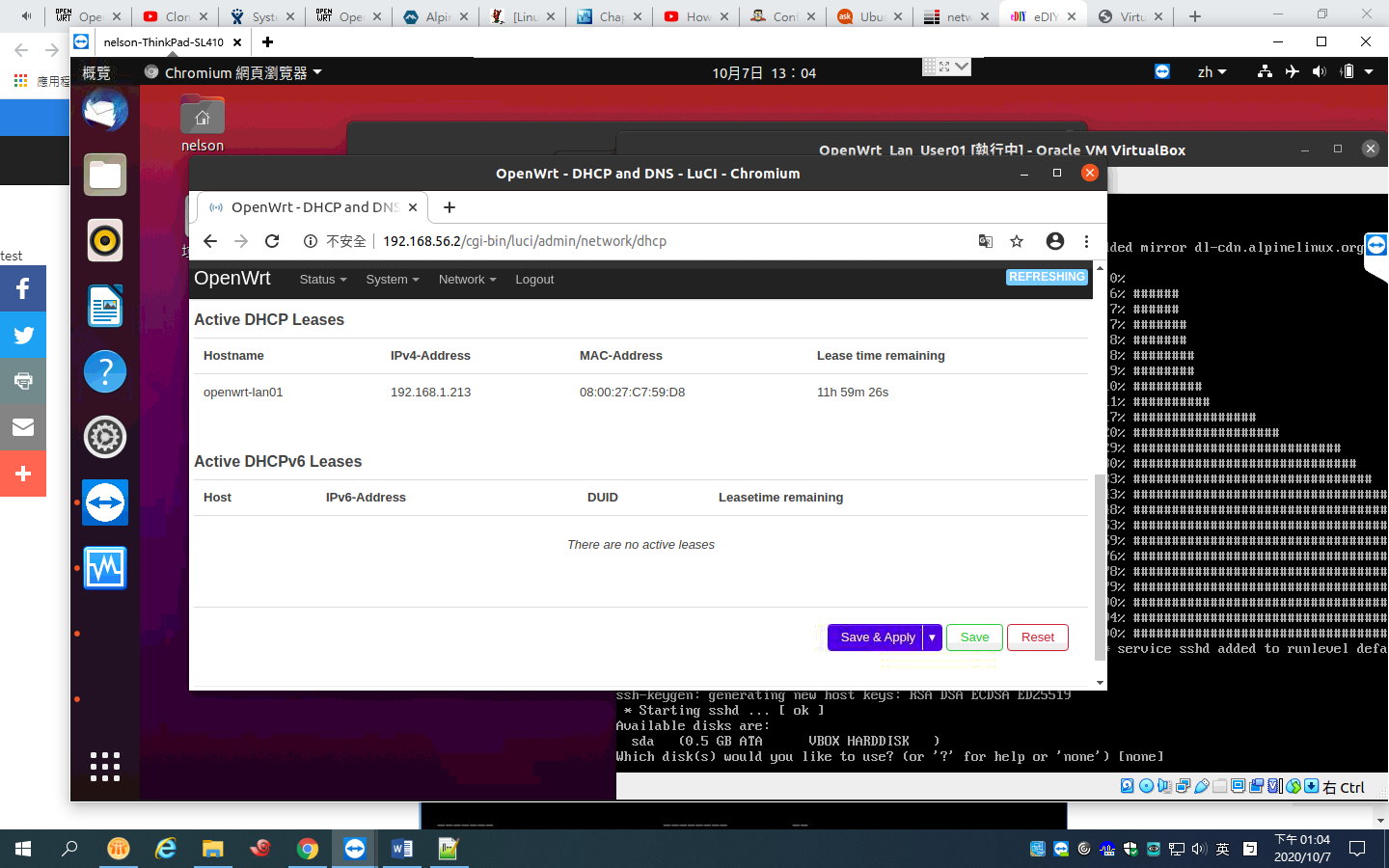
安裝完後，再主機的網頁上連接192.168.56.2路由器。



這樣就有網頁式設定介面了，剩下的從網頁互動就行了喔。  
其他更多指令可以在OpenWrt官網查詢：

<https://openwrt.org/zh-tw/docs/guide-user/base-system/uci>

也可以設定其他Lan User來檢查路由器的機能是否正常。



# Abbreviation

|  |  |
| --- | --- |
| Name | Full information |
|  |  |
|  |  |
|  |  |
|  |  |

# Reference

[Build sysyem Setup Linux](https://openwrt.org/docs/guide-developer/build-system/install-buildsystem)  
[Build system Usage](https://openwrt.org/docs/guide-developer/build-system/use-buildsystem)  
[Building your own firmware](https://openwrt.org/docs/guide-user/additional-software/beginners-build-guide)

[DeviWiki](https://deviwiki.com/)  
[Run in Virtual box](https://www.youtube.com/watch?v=kjzQg_sB8rY&ab_channel=ShruthiVenugopal)  
[Adding new SoC](https://www.youtube.com/watch?v=R0EaknpHScU&ab_channel=prplFoundation)  
<http://nossiac.com/blog/openwrt-tutorial-1.html>

<https://openwrt.org/docs/guide-user/virtualization/virtualbox-vm>