How to install the Yocto Project SDK in STM32CubeIDE

Contents

Stable: 23.11.2021 - 10:23 / Revision: 23.11.2021 - 10:16

Stable: 23.11.2021 - 10:20 | Revision: 23.11.2021 - 10:10

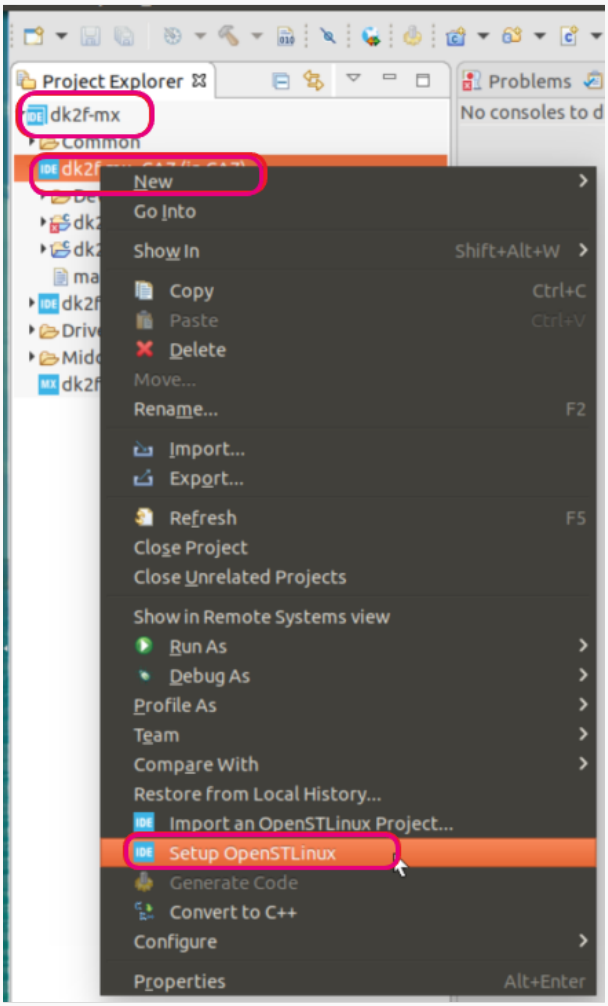
A quality version of this page, approved on *23 November 2021*, was based off this revision.

This article explains the way STM32CubeIDE is managing the Yocto Project® SDK provided by OpenSTLinux.

1 Overview

From STM32CubeIDE release 1.4.0 on **Linux® host ONLY**, STM32CubeIDE supports OpenSTLinux projects and its associated Yocto Project® SDK. Inside STM32CubeIDE, this support means two new Eclipse® plugins (SDK & Sources) to be installed, directly from the embedded CA7 project menu context:

- *Setup OpenSTLinux*
- *Import an OpenSTLinux Project...*

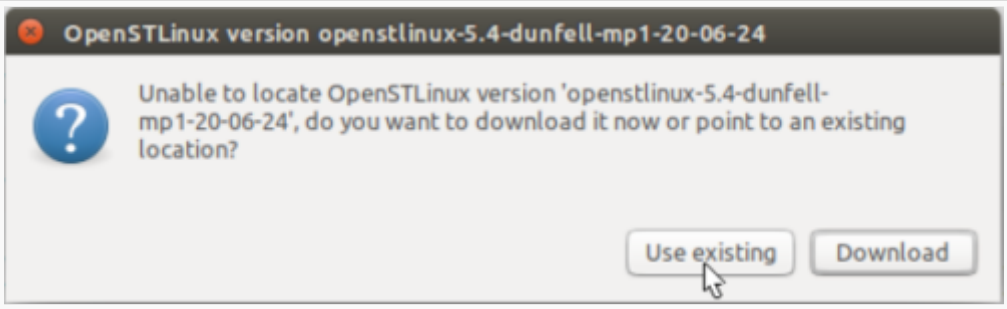


OpenSTLinux Contextual Menu

Two flavors are proposed for installing the Yocto Project® SDK:

- Yocto SDK is already installed on the host workstation, typically after a download of the STM32MP1 OpenSTLinux Developer Package. In that case, only a setup is needed for STM32CubeIDE to use it.
- Yocto SDK is not present on the host workstation. It can be installed via STM32CubeIDE.

The choice is proposed after the ***Setup OpenSTLinux*** menu, selecting *Use existing*.

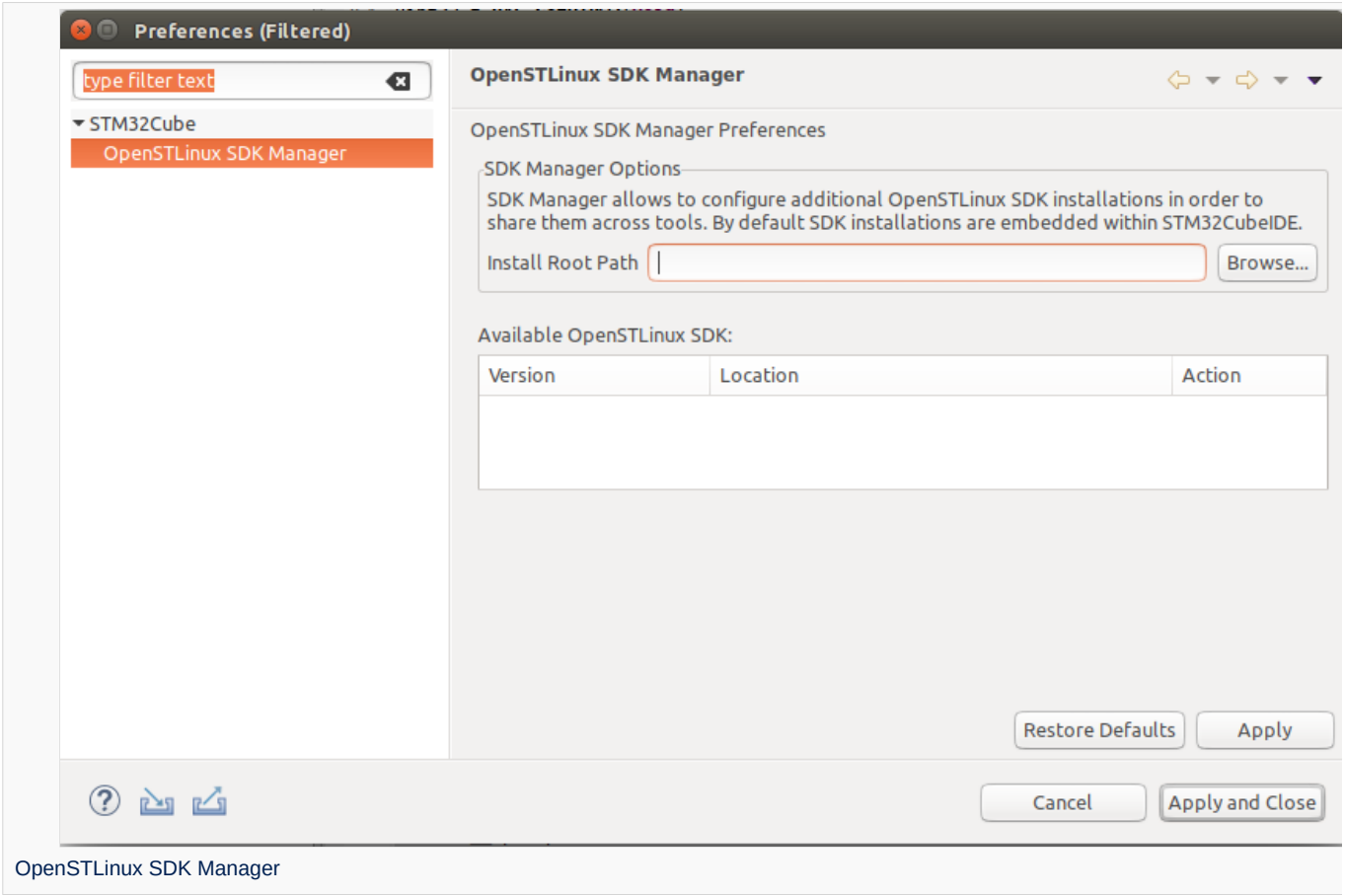


OpenSTLinux SDK Installation Menu

Note that setup OpenSTLinux phase includes also download and installation of **OpenSTLinux Sources** plugin.

2 Using already installed Yocto Project® SDK

You must provide the Yocto Project® SDK *Install Root Path* in the STM32Cube Preferences. Typically if using the default Developper pakage directory tree under : <working root directory >/Developer-Package/SDK.



OpenSTLinux SDK Manager

3 Installing Yocto Project® SDK via STM32CubeIDE

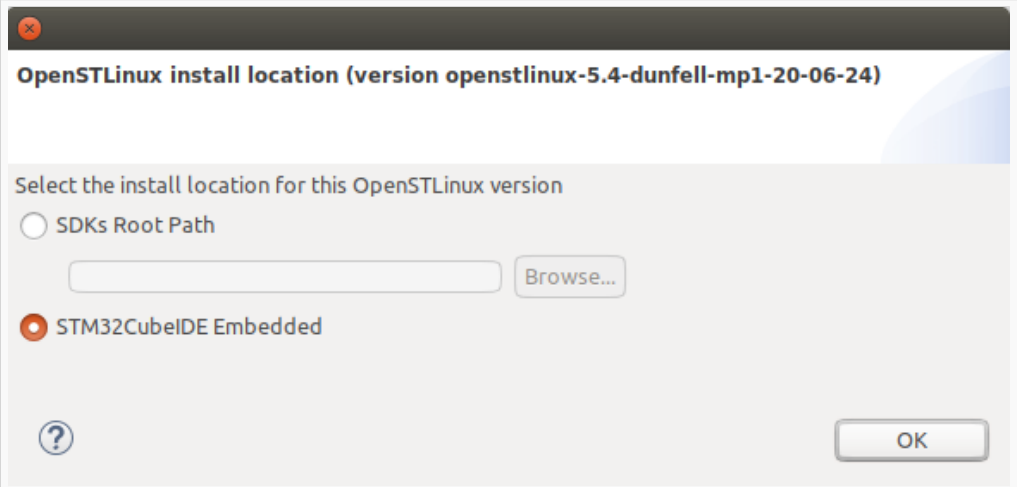
This corresponds to the *Download* choice where **OpenSTLinux SDK** plugin is installed. Note that missing OpenSTLinux required packages leads to unpredictable Yocto Project® SDK usage.

Warning

OpenSTLinux development requires specific packages on the host workstation. See PC_prerequisites.

It is then possible to install the Yocto Project® SDK as:

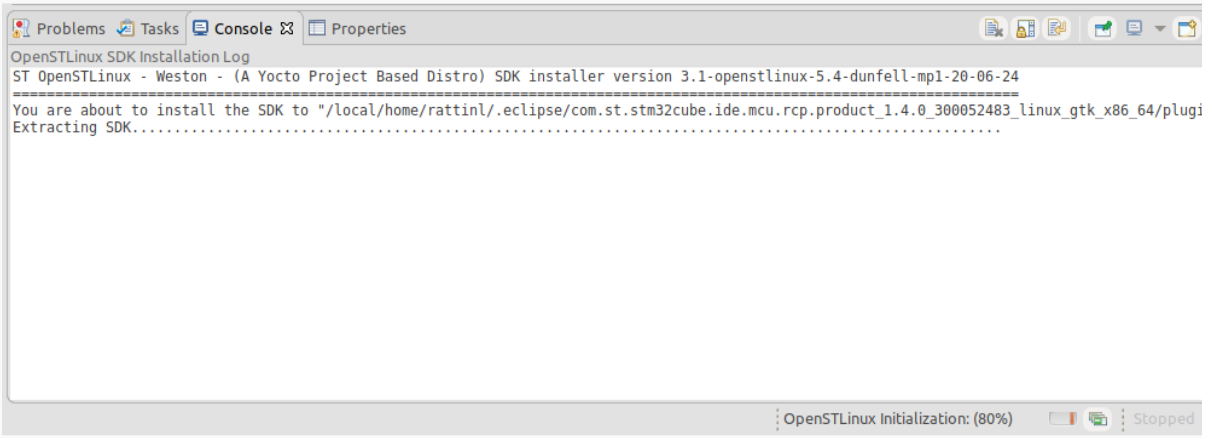
- ***external***, on the host workstation disk, outside STM32CubeIDE scope; Yocto SDK removal is under final user responsibility
- ***embedded*** inside the STM32CubeIDE; Yocto SDK removal is managed by STM32CubeIDE via plugin **OpenSTLinux SDK**



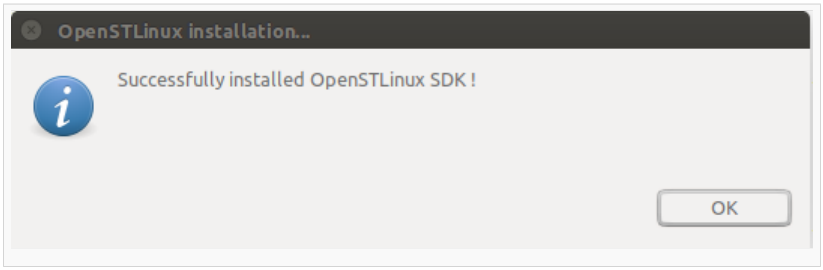
OpenSTLinux SDK Install Location

After accepting unsigned content installation warning, the Yocto Project® SDK installation script is launched and appears in an STM32CubeIDE console.

Installation Security Warning



OSTL SDK Install Script



.

Stable: 17.11.2021 - 16:09 / Revision: 17.11.2021 - 09:00

A quality version of this page, approved on *17 November 2021*, was based off this revision.

Information

Recommended setup: Native Linux® PC

Contents

1 Purpose

This article explains and describes the host PC hardware and software configuration required to activate and run the STM32 MPU platforms.

2 Recommended PC configurations

The PC requirements depend on the Package you want to use.

The table below guides through the selection and configuration of the host PC environment according the targeted Package:

Host environment	Starter Package	Developer Package	Distribution Package
Windows® (64 bits) Tested with Windows 7 and Windows 10 Preferred version is Windows 10	native	Virtual machine	Virtual Machine
Linux (64 bits) Tested with Ubuntu®**20.04** and 18.04	Native	Native + additional packages (see Linux PC chapter)	native + additional packages (see Linux PC chapter)

There are no absolute minimal requirements regarding the PC hardware configuration. However ST recommends to meet or exceed the following hardware requirements when using the *Developer Package* or *Distribution Package*.

The table below corresponds to the minimal validated configuration:

Hardware item	Minimal validated configuration	Comments / Recommendations
CPU	core i5-2540M @ 2.6 GHz 2 cores (4 threads) 3-Mbyte cache	**64-bit instruction set is mandatory** **8 cores/threads or more** is a good configuration for *Developer Package* and *Distribution Package*.
RAM	8 Gbytes	**16 Gbytes or more** are recommended especially for *Virtual machine* setup, *Developer Package* and *Distribution Package*.
Hard drive	320 Gbytes	**1 Tbytes** is the best suited configuration for *Distribution Package*

3 Linux® PC

A Linux PC with recent **Ubuntu LTS** (20.04 or 18.04) is the recommended setup. Other Ubuntu revisions could also supported (refer to Yocto reference manual[1]) .

Information

ST solutions are tested and validated on a Linux PC running Ubuntu 20.04 LTS (and Ubuntu 18.04 LTS).

3.1 Checking Internet access

- Internet access through http and https protocols

Required at least for *Developer Package* and *Distribution Package*.

The command below enables checking internet access through http/https protocols:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

If an 'OK' message is returned, the network is well configured. In such case, skip the rest of this section and jump to Install extra packages.

Any other situation likely indicates that a proxy is required for http/https protocols. The best solution to set a proxy for http/https protocols is via the shell variables http_proxy and https_proxy:

```
PC $> export http_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyProx
yPort>
PC $> export https_proxy=http://<MyProxyLogin>:<MyProxyPassword>@<MyProxyServerUrl>:<MyPro
xyPort>
```

Since your password (<MyProxyPassword>) might contain "special characters", translate it into ASCII hexacode. To do this, use the command below:

```
PC $> echo -n " <MyProxyPassword>" | od -A n -t x1 -w128 | head -1 | tr " " "%"
```

Check again the Internet access by using the following command:

```
PC $> wget -q www.google.com && echo "Internet access over HTTP/HTTPS is OK !" || echo
"No internet access over HTTP/HTTPS ! You might need to set up a proxy."
```

- Internet access for *sudo* commands

Required for *Distribution Package*.

By default, *sudo* commands are executed in the *root* user environment; no Internet proxy settings are applied for *root* users. *Root* users must be able to browse Internet after having created an alias passing the proxy settings on *sudo* commands:

```
PC $> alias sudo='sudo http_proxy=$http_proxy'
```

Check that the *sudo* commands are successful (requires Internet access):

```
PC $> sudo apt-get update
```

- **Internet over git://, ssh:// and others specifics protocols**

Required for *Distribution Package*.

In addition to http/https protocols (used in 90% of the Internet traffic), other protocols such as git:// or ssh:// might be required.

For example, in the context of the *Distribution Package*, some "git fetch" commands might require "git:// protocols".
To support these protocols through a proxy, directly setup the proxy in the $HOME/.gitconfig file (core.gitproxy) and use a tool such as *cockscrew*[2] to tunnel the git:// flow into the http flow:

```
PC $> sudo apt-get update
PC $> sudo apt-get install corkscrew

PC $> git config --replace-all --global core.gitproxy "$HOME/bin/git-proxy.sh"
PC $> git config --add --global core.gitproxy "none for <MyPrivateNetworkDomain>"
(optional, for example .st.com or localhost)
PC $> echo 'exec corkscrew <MyProxyServerUrl> <MyProxyPort> $* $HOME/.git-proxy.auth' >
$HOME/bin/git-proxy.sh
PC $> chmod 700 $HOME/bin/git-proxy.sh
PC $> echo '<MyProxyLogin>:<MyProxyPassword>' > $HOME/.git-proxy.auth
PC $> chmod 600 $HOME/.git-proxy.auth
```

Use the command below to test the proxy settings:

```
PC $> git ls-remote git://git.openembedded.org/openembedded-core > /dev/null && echo OK
|| echo KO
```

The command returns 'OK' if the proxy settings are correct.

3.2 Installing extra packages

Required for *Developer Package* and *Distribution Package*.

Additional Ubuntu packages must be installed to perform basic development tasks, basic cross-compilation (via *Developer Package*) or more complex cross-compilation such as OpenEmbedded does (via *Distribution Package*):

- Packages required by OpenEmbedded/Yocto (details here):

```
PC $> sudo apt-get update
PC $> sudo apt-get install gawk wget git-core diffstat unzip texinfo gcc-multilib build-
essential chrpath socat cpio python3 python3-pip python3-pexpect xz-utils debianutils
iputils-ping python3-git python3-jinja2 libegl1-mesa libsdl1.2-dev pylint3 pylint xterm
PC $> sudo apt-get install make xsltproc docbook-utils fop dblatex xmlto
PC $> sudo apt-get install libmpc-dev libgmp-dev
```

- Packages needed for some "Developer Package" use cases:

```
PC $> sudo apt-get install libncurses5 libncurses5-dev libncursesw5-dev libssl-dev linux-
headers-generic u-boot-tools device-tree-compiler bison flex g++ libyaml-dev libmpc-dev
libgmp-dev
```

- Package for repo (used to download the "Distribution Package" source code):

```
Please follow the installation instructions described in repo.
```

For Ubuntu 20.04, first sets python3 as default: ***PC $>*** *sudo apt install python-is-python3*

- Useful tools:

```
PC $> sudo apt-get install coreutils bsdmainutils sed curl bc lrzsz corkscrew cvs
subversion mercurial nfs-common nfs-kernel-server libarchive-zip-perl dos2unix texi2html
diffstat libxml2-utils
```

3.2.1 Installing extra packages for Android™

Below information is related to the Android™ distribution

Before downloading and building the STM32MPU distribution for Android™, make sure your system meets the following requirements:

- A 64-bit Linux® environment.
- At least 150 Gbytes of free disk space. If you conduct multiple builds, even more space is required.
- At least 8 Gbytes of RAM/swap. If you are running Linux on a virtual machine, at least 16 Gbytes of RAM /swap are required.

For more details, refer to the requirements page of the AOSP website.

Use the following commands to install the packages required to build an environment for Android (Distribution Package), after having installed the required packages listed on Android website:

```
sudo apt-get update
sudo apt-get install chrpath curl libxml2-utils gdisk pv python-
pycryptopp python-crypto autotools-dev automake libusb-1.0-0-dev
```

To ensure USB communication (with ADB) between the host and the device, see ADB § Device Connection.

To run Android-provided tests (CTS and VTS), see Android Platform Testing.

At this stage: Your environment is ready for Android build, debug and test.

3.3 Additional configurations

- Allowing up to 16 partitions per MMC

By default a maximum of 8 MMC partitions are allowed on Linux systems. All Packages (such as Starter Package) need more than 10 partitons for the storage device. To extend the number of partitions per device to 16, the following options must be added to modprobe:

```
PC $> echo 'options mmc_block perdev_minors=16' > /tmp/mmc_block.conf
PC $> sudo mv /tmp/mmc_block.conf /etc/modprobe.d/mmc_block.conf
```

- Checking for *locale* setup

Required for *Distribution Package*.

The *locale* setting is used by some applications/commands (including by *Distribution Package* applications/commands).
Verify that the *locale* settings are as follows:

```
PC $> locale
LANG=en-US.UTF-8
```

If the *locale* command returns a different configuration than the one shown above, reconfigure it as follows:

```
PC $> sudo update-locale LANG=en_US.UTF-8
```

- Adding users in basics groups

The *user* login must belong to the basic Linux groups such as **disk**, **tty**, **dialout** or **plugdev**
Use the *groups* command to list groups for the current user:

```
PC $> groups
```

If needed, add *user* to the missing *<groups>*:

```
PC $> sudo adduser $USER <group>
```

Then **reboot** the PC.

3.4 Seting up *Git* user information

Required for *Developer Package* and *Distribution Package*.

The user Information are needed by git[3] if *commit* and/or *push* commands are used:

```
PC $> git config --global user.name "Your Name"
PC $> git config --global user.email "you@example.com"
```

4 Windows PC

Starter Package **may run on Windows.**
Developer Package **and** ***Distribution Package*** **require a Linux environment.**

Warning

ST solutions, while reportedly functional when running on a Linux Virtual machine, are only validated for Linux native setups.

There are several ways to run Linux system on top of a Windows host PC. ST recommends to use a Virtual Machine System:

1. Install a virtual machine such as VMWare [4]
2. Setup a **64-bit** Ubuntu image compatible with your virtual machine

ST, in an experimental way, has also run *Developer Package* and *Distribution Package* on WSL2 (Windows Subsystem for Linux 2). Refer to WSL2 chapter.

4.1 Virtual machine system

4.1.1 Installing the virtual machine

ST has selected VMWare as Linux virtual machine solution.

VMWare is a commercial company specialized in virtualization solutions. The available solutions to support a virtual Linux machine on a Windows PC are:

- VMWare Workstation Player (paid solution) for commercial use (download here [5])
- VMWare Workstation Player (free solution) for home use (download here [6])

Please proceed with the installation of the virtual machine.

Before running the virtual machine, make sure the virtualization is activated in the BIOS (it should be activated by default for any retail PC).

4.1.2 Downloading the Ubuntu image for the virtual machine

The "osboxes.org" [7] website provides virtual machine images compatible with VMWare(*.vmdk).

ecosystem release v3.0.0 : **Setup have been validated and tested on Ubuntu 18.04 (64bits and Ubuntu 20.04 (64bits).**
ecosystem release v3.1.0 : **Setup have been validated and tested on Ubuntu 20.04 (64bits and Ubuntu 18.04 (64bits).**

Download the 64-bit Ubuntu image available at [8], then:

1. Unzip the downloaded file
2. In VMware, create a virtual machine using the Ubuntu virtual disk downloaded from osboxes.org.

The recommended usage is to dedicate, at least, half of the host machine to the virtual machine:

```
- CPU: 2 cores at least,
- RAM: 6 Gbytes or more is a good choice (the more RAM allocated to Virtual Machine the
better - the RAM allocated to Virtual Machine must be at least 4 Gbytes),
- Network: NAT is a good easy way to benefit from a network connection within the virtual
machine.
```

The virtual size of the virtual disk downloaded from osboxes.org is about 500 Gbytes. Even if, at beginning, the real size of the file of the virtual disk is less, the size can grow up to 500 Gbytes over distribution package compiling or package development.

Information

For VMware, first create a default virtual machine, then add the previously downloaded *.vmdk* file.

Please refer to the VMwarePlayer screenshot tutorial.

4.1.3 Launching Ubuntu image

Warning

For "AZERTY" keyboard users:

The default keyboard configuration is "QWERTY".
To configure the keyboard to "AZERTY", start by opening a session (take care that the keyboard layout is QWERTY).
TIP: the password for the default user "*osboxes.org*" is "*osboxes.org*".
TIP: the '.' character is obtained by clicking ':' on an AZERTY keyboard configured in QWERTY.
Once the session is open, click the 'En' icon on top/right of the screen, select the French ('Fr') keyboard layout and move it to the first position in the list.
Optionally the 'En' keyboard can be completely removed. If the 'Fr' option is not present, it can be added with the 'Text entry setting' menu.

Default 'Ubuntu *Credentials'* **are set to "osboxes.org" for both login and password.**

Warning

Adjusting screen resolution:

The (default) resolution used by the virtual machine is 800x600 (smallest available). It is not automatically adjusted to the display resolution. To adjust the resolution, click the "settings" icon ('toothed wheel' on top/right of the screen), then "system settings ..." > "display" and select the appropriate resolution for the display (do not to forget to click the "Apply" button on bottom/left of the "Screen Resolution Setting" window).

For a better experience with the VMware virtual machine, install "vmware-tools" in order to be able to use the clipboard to drag-and-drop and copy/paste files between VMware and Windows. A step-by-step installation procedure of **vmware-tools** is available in PreRequisite-Vmware-tools.pdf.

The virtual machine is now up and running!

The Ubuntu setup must be finalized according recommendations provided in Linux PC chapter

Warning

USB connection speed:

A USB connection is required to access STLink (debugger and serial port) and by STM32CubeProgrammer. The speed of the USB connection between Linux running in the virtual machine and the external USB devices can be severely impacted by the following factors:

- the virtual machine USB setup
- the USB controller in the host PC
- the USB device connected to host PC
- any USB hub between the USB host and the USB device.

If the speed of your USB connection is too low:

- try different USB configurations of the virtual machine;
- connect the USB device directly to the host USB port (without any USB hub);
- try connecting the USB device to another USB port of the host (some PC have different USB controller on different USB port).

4.2 WSL2 (experimental)

Even if STMicroelectronics strongly recommends to use a Linux® environment, the *Developer Package* and *Distribution Package* work on WSL2 (Windows Sub-system Linux 2) environment. WSL is a feature provided by **Windows 10®**.
WSL2 is a new version of the architecture that powers the Windows subsystem for Linux to run ELF64 Linux binaries on Windows (more details on aka.ms/wsl2). It is available on Windows 10 since build 18917. The *Developer Package* successfully runs on WSL2 but unsuccessfully on WSL.

- Installing WSL2 :
  - To install WSL2, read https://docs.microsoft.com/fr-fr/windows/wsl/wsl2-install
  - Once WSL2 is installed, jump to #Linux_PC to make your WSL2 ready to run *Developer Package* and/or *Distribution Package*.
- WSL2 limitations :
  - Up to now (March 2021), WSL2 does not support hardware such as USB or serial devices (more details).
    This means that STM32CubeProgrammer must be used through native Windows.
  - WSL2 files are not browsable from Windows native file explorer.
    To share files between WSL2 and Windows, it is recommend to use the mount point */mnt/c* from WSL2 and perform copies.
- Tips for using WSL2 :
  - Launching graphical application : on *wiki.ubuntu.com*, go to WSL page and read the chapter Running Graphical Applications.

4.3 References

- Supported Linux Distributionsl
- https://en.wikipedia.org/wiki/Corkscrew_(program)
- Git
- http://vmware.com
- https://my.vmware.com/en/web/vmware/free#desktop_end_user_computing/vmware_workstation_player/15_0
- https://www.vmware.com/products/workstation-player/workstation-player-evaluation.html
- http://osboxes.org
- https://www.osboxes.org/ubuntu/#ubuntu-20-04-3-vmware