

UEFI & EDK II Training

PLATFORM BUILD LAB - OVMF

tianocore.org

PLATFORM BUILD LABS

- ✱ Build a EDK II Platform using OVMF package
- ✱ Run Ovmf using Qemu 

BUILD OVMFPKG

Setup OvmfPkg to build and run w/ QEMU

PRE-REQUISITES UBUNTU 16.04

Instructions from: [tianocore wiki Ubuntu_1610](https://www.tianocore.org/wiki/Ubuntu_1610)

Example Ubuntu 16.04

The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo apt-get install build-essential uuid-dev iasl git gcc-5 nasm python3-distutils
```

build-essential - Informational list of build-essential packages

uuid-dev - Universally Unique ID library (headers and static libraries)

iasl - Intel ASL compiler/decompiler (also provided by acpica-tools)

git - support for git revision control system

gcc-5 - GNU C compiler (v5.4.0 as of Ubuntu 16.04 LTS)

nasm - General-purpose x86 assembler

python3 - distutils - distutils module from the Python standard library

```
bash$ sudo apt-get install qemu
```

Qemu – Emulation with Intel architecture with UEFI Shell



Pre-requisites Clear Linux* Project

Example Using Clear Linux* Project

The following need to be accessible for building Edk2, From the terminal prompt (Cnt-Alt-T):

```
bash$ sudo swupd bundle-add devpkg-util-linux
```

Devpkg-util-linux – includes bundles for developer tools for writing “C” Applications included: gcc, nasm, uuid, etc.

```
bash$ sudo swupd bundle-add kvm-host
```

Qemu – Emulation with Intel architecture with UEFI Shell



Create QEMU Run Script

1. Create a run-ovmf directory under the home directory

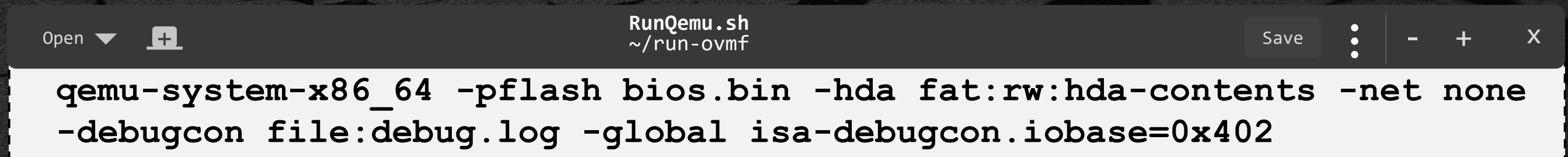
```
bash$ cd ~  
bash$ mkdir ~run-ovmf  
bash$ cd run-ovmf
```

2. Create a directory to use as a hard disk image

```
bash$ mkdir hda-contents
```

3. Create a Linux shell script to run the QEMU from the run-ovmf directory

```
bash$ gedit RunQemu.sh
```



The screenshot shows a text editor window titled "RunQemu.sh" with the path "~/run-ovmf". The editor contains the following command:

```
qemu-system-x86_64 -pflash bios.bin -hda fat:rw:hda-contents -net none  
-debugcon file:debug.log -global isa-debugcon.iobase=0x402
```

4. Save and Exit

DOWN LOAD THE EDK II SOURCE

- OPTIONAL

OPTIONAL - Open a terminal prompt and create a source working directory

```
bash$ mkdir ~/src
bash$ cd ~/src
bash$ mkdir edk2-ws
```

OPTIONAL - Internet Proxies – (company Firewall used for example)

```
bash$ export http_proxy=http://proxy-us.company.com:911
bash$ export ftp_proxy=$http_proxy
```

OPTIONAL - Download edk2 source tree using Git

```
bash$ git clone --recursive https://github.com/tianocore/edk2
bash$ git clone https://github.com/tianocore/edk2-libc.git
```

OPTIONAL - Build the tools

```
bash$ make -C edk2/BaseTools
```

NOTE: Lab Material will have a different “edk2”

SETUP LAB MATERIAL

Lab_Material_FW.zip

DOWN LOAD LAB MATERIAL

Download the Lab_Material_FW.zip from :  github.com
[Lab_Material_FW.zip](#)

OR

Use git clone to download the Lab_Material_FW

```
bash$ cd $HOME  
bash$ git clone https://github.com/tianocore-training/Lab_Material_FW.git
```

Directory Lab_Material_FW will be created

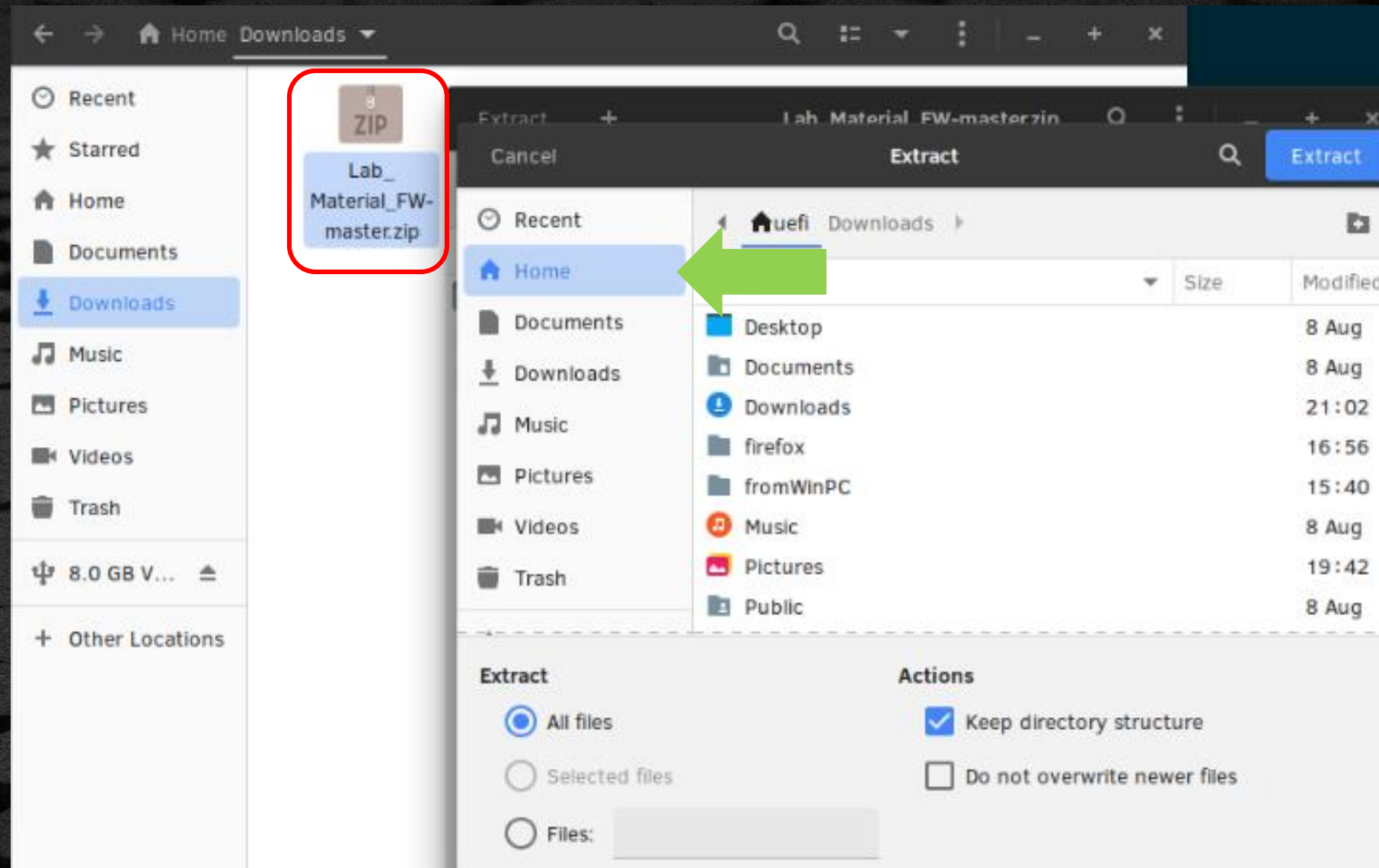
FW

- Documentation
- DriverWizard
- edk2-ws
- edk2Linux
- LabSampleCode

BUILD EDK II OVMF

-Extract the Source

1. Extract the Downloaded `Lab_Material_FW.zip` to `Home` (this will create a directory `FW`)



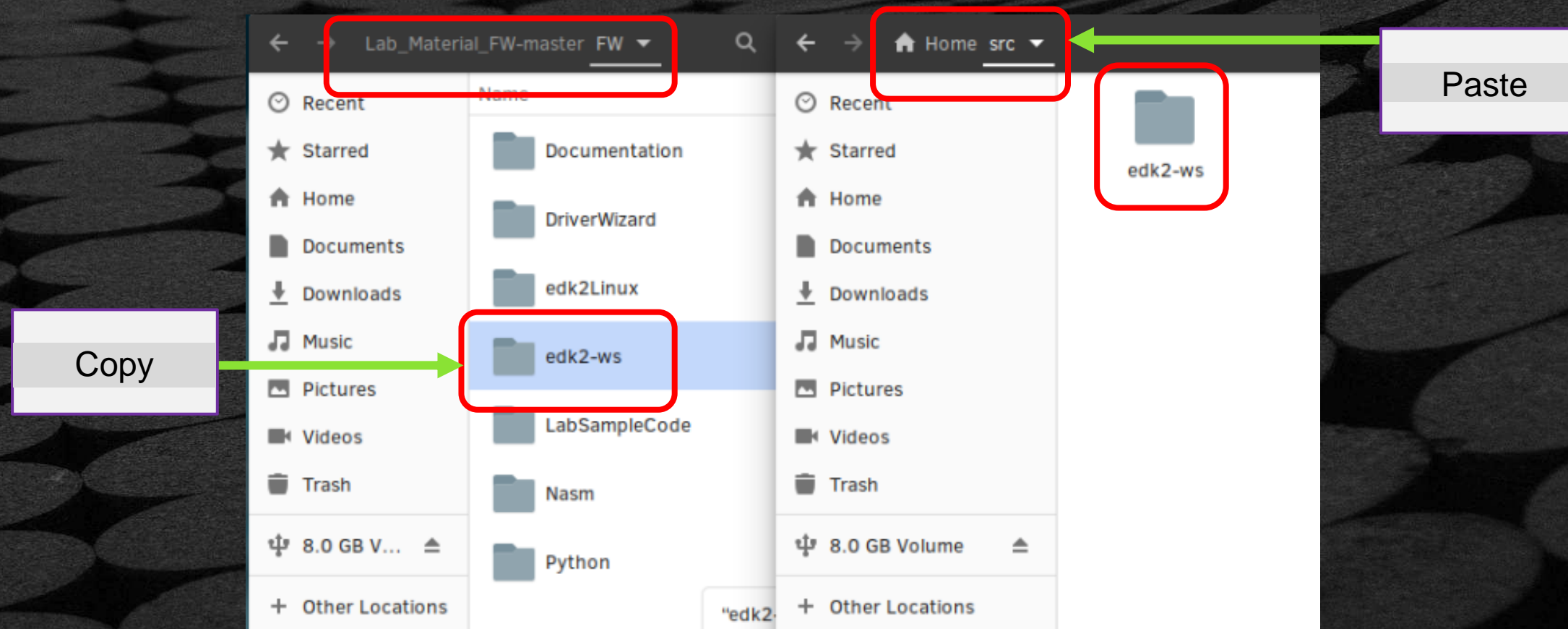
BUILD EDK II OVMF

- Copy the Source

2. Open a terminal prompt (Alt-Cnt-T)
3. Create a working space source directory under the home directory

```
bash$ mkdir ~src
```

4. From the FW folder, copy and paste folder “~.../FW/edk2-ws” to ~src



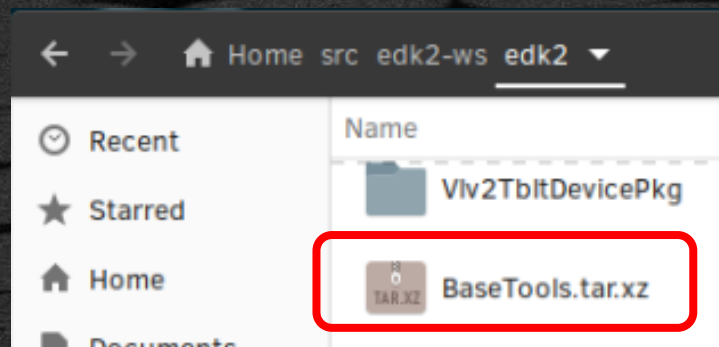
BUILD EDK II OVMF

-Getting BaseTools

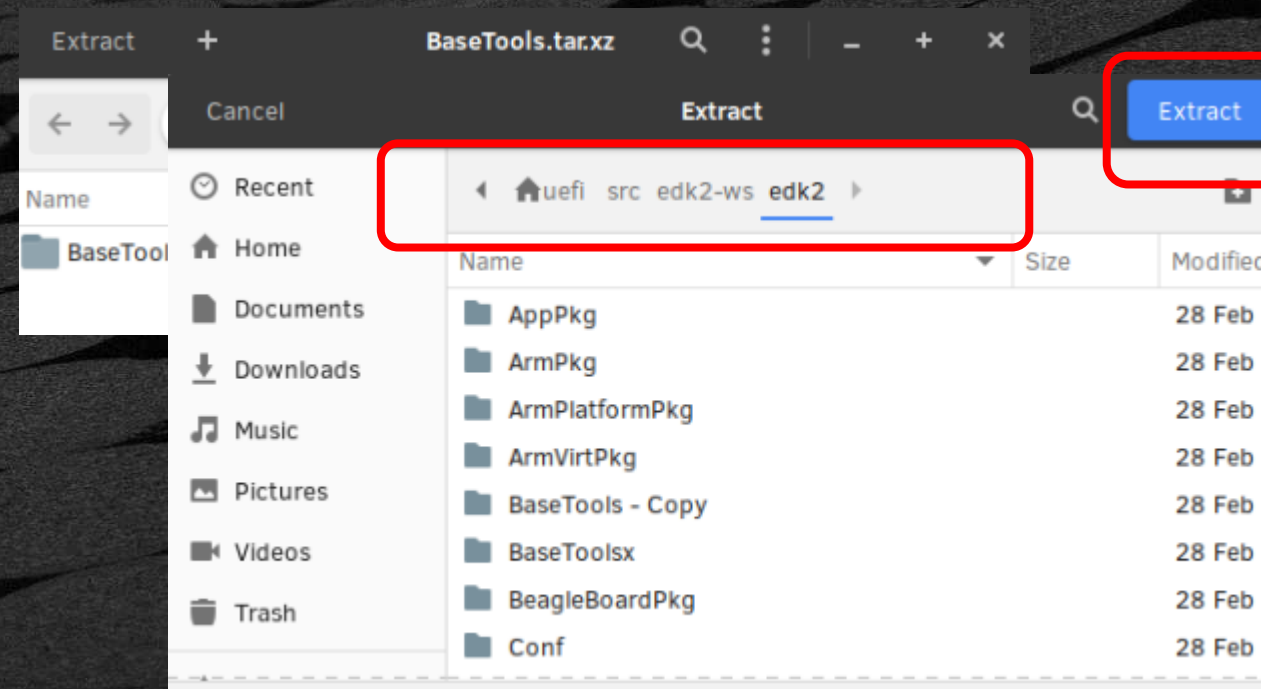
5. Rename or `mv` the directory “~src/edk2-ws/edk2/BaseTools”

```
bash$ cd ~src/edk2-ws/edk2
bash$ mv BaseTools BaseToolsX
bash$ tar -xf BaseTools.tar.xz
```

6. Extract the file `BaseTools.tar.xz` to ~src/edk2-ws/edk2



Double click



Select Extract directory
\$HOME/src/edk2-ws/edk2

BUILD EDK II OVMF

- Building BaseTools

7. Export work space & platform path

```
bash$ cd ~src/edk2-ws
```

```
bash$ export WORKSPACE=$PWD
```


```
bash$ export PACKAGES_PATH=$WORKSPACE/edk2:$WORKSPACE/edk2-libc
```

8. Run Make

```
bash$ cd edk2
```

```
bash$ make -C BaseTools/
```

Make sure the tests pass OK



```
uefi@clr-0: ~/src/edk2-ws/edk2
test_Workspace___init__ (CheckPythonSyntax.Tests) ... ok
test_build_BuildReport (CheckPythonSyntax.Tests) ... ok
test_build___init__ (CheckPythonSyntax.Tests) ... ok
test_build_build (CheckPythonSyntax.Tests) ... ok
test_sitecustomize (CheckPythonSyntax.Tests) ... ok
test32bitUnicodeCharInUtf8Comment (CheckUnicodeSourceFiles.Tests) ... ok
test32bitUnicodeCharInUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testSupplementaryPlaneUnicodeCharInUtf16File (CheckUnicodeSourceFiles.Tests) ...
ok
testSurrogatePairUnicodeCharInUtf16File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testSurrogatePairUnicodeCharInUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ..
. ok
testUtf16InUniFile (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8File (CheckUnicodeSourceFiles.Tests) ... ok
testValidUtf8FileWithBom (CheckUnicodeSourceFiles.Tests) ... ok

-----
Ran 270 tests in 4.121s

OK
make[1]: Leaving directory '/home/uefi/src/edk2-ws/edk2/BaseTools/Tests'
make: Leaving directory '/home/uefi/src/edk2-ws/edk2/BaseTools'
uefi@clr-0:~/src/edk2-ws/edk2$
```


BUILD OVMF PLATFORM

BUILD EDK II OVMF

-Update Target.txt

What is OVMF?

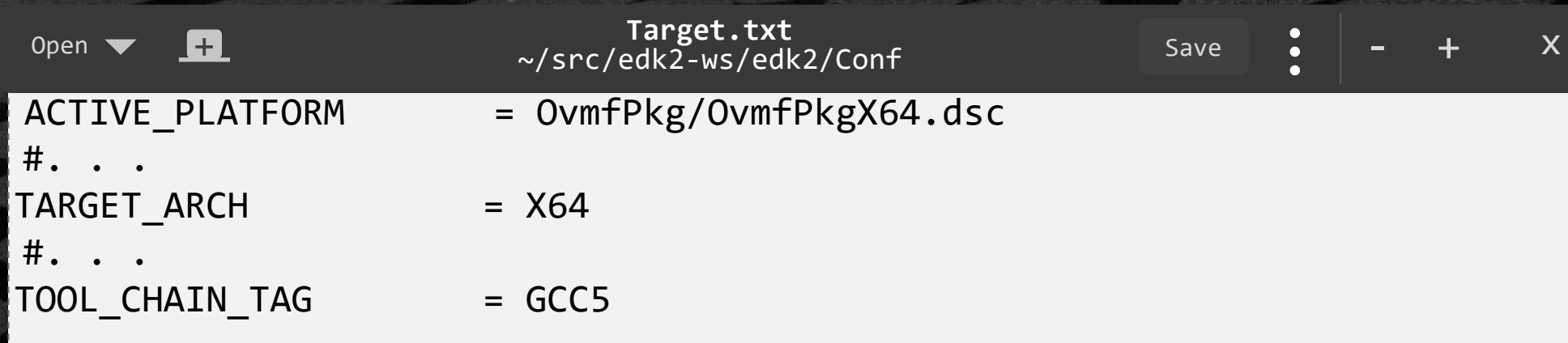
Open Virtual Machine Firmware - Build with edk2

```
bash$ cd ~/src/edk-ws/edk2
bash$ . edksetup.sh
```

```
uefi@clr-0~/src/edk2-ws/edk2 $ . edksetup.sh
Loading previous configuration from /home/uefi/src/edk2-ws/edk2/Conf/Build
WORKSPACE: /home/uefi/src/edk2-ws
EDK_TOOLS_PATH: /home/uefi/src/edk2-ws/edk2/BaseTools
CONF_PATH: /home/uefi/src/edk2-ws/edk2/Conf
uefi@clr-0~/src/edk2-ws/edk2 $
```

Edit the file Conf/target.txt

```
bash$ gedit Conf/target.txt
```



```
Open ▼ + Target.txt
~/src/edk2-ws/edk2/Conf Save ⋮ - + X
1. ACTIVE_PLATFORM = OvmfPkg/OvmfPkgX64.dsc
   #. . .
2. TARGET_ARCH = X64
   #. . .
3. TOOL_CHAIN_TAG = GCC5
```

Save and build

```
bash$ build -D ADD_SHELL_STRING
```

More info: [tianocore - wiki/OVMF](https://www.tianocore.org/wiki/OVMF)

BUILD EDK II OVMF

-Inside Terminal

```

uefi@clr-0: ~/src/edk2-ws/edk2
┌───┴───┐
│ k2-ws/Build/OvmfX64/DEBUG_GCC5/X64/ │
│ ib/OUTPUT/./XenHypercall.obj -I/home │
│ percallLib/X64 -I/home/uefi/src/edk │
│ ib/XenHypercallLib/DEBUG -I/home/ue │
└───┴───┘

uefi@clr-0:~/src/edk2-ws/edk2 $ . edk2
Loading previous configuration from /home/uefi/src/edk2-ws/edk2/OvmfPkg -
WORKSPACE: /home/uefi/src/edk2-ws
EDK_TOOLS_PATH: /home/uefi/src/edk2-ws
CONF_PATH: /home/uefi/src/edk2-ws
PYTHON_COMMAND: /usr/bin/python3.7

uefi@clr-0:~/src/edk2-ws/edk2 $ build y-bounds -ffunction-sections -fdata
Build environment: Linux-5.2.7-816.na
Build start time: 21:58:04, Aug.15 2019

Workspace: /home/uefi/src/edk2-ws
Packages path: /home/uefi/src/edk2-ws/Build/OvmfX64/DEBUG_GCC5/X64
EDK_TOOLS_PATH: /home/uefi/src/edk2-ws
CONF_PATH: /home/uefi/src/edk2-ws
PYTHON_COMMAND: /usr/bin/python3.7

Architecture(s) = X64
Build target = DEBUG
Toolchain = GCC5

Active Platform = /home/uefi/src/edk2-ws/edk2/OvmfPkg/OvmfPkgX64.dsc
Flash Image Definition = /home/uefi/src/edk2-ws/edk2/OvmfPkg/OvmfPkgX64.fdf
Processing meta-data ..

Region Name = None
Generate Region at Offset 0x20000
Region Size = 0xE0000
Region Name = FV
Generate Region at Offset 0x100000
Region Size = 0xB00000
Region Name = FV
GUID cross reference file can be found at /home/uefi/src/edk2-ws/Build/OvmfX64/DEBUG_GCC5/FV/Guid.xref

FV Space Information
SECFV [10%Full] 212992 total, 21808 used, 191184 free
PEIFV [19%Full] 917504 total, 183016 used, 734488 free
DXEFV [35%Full] 11534336 total, 4069440 used, 7464896 free
FVMAIN_COMPACT [34%Full] 3440640 total, 1191176 used, 2249464 free

- Done -
Build end time: 22:03:31, Aug.15 2019
Build total time: 00:05:27

uefi@clr-0:~/src/edk2-ws/edk2 $

```

Finished build

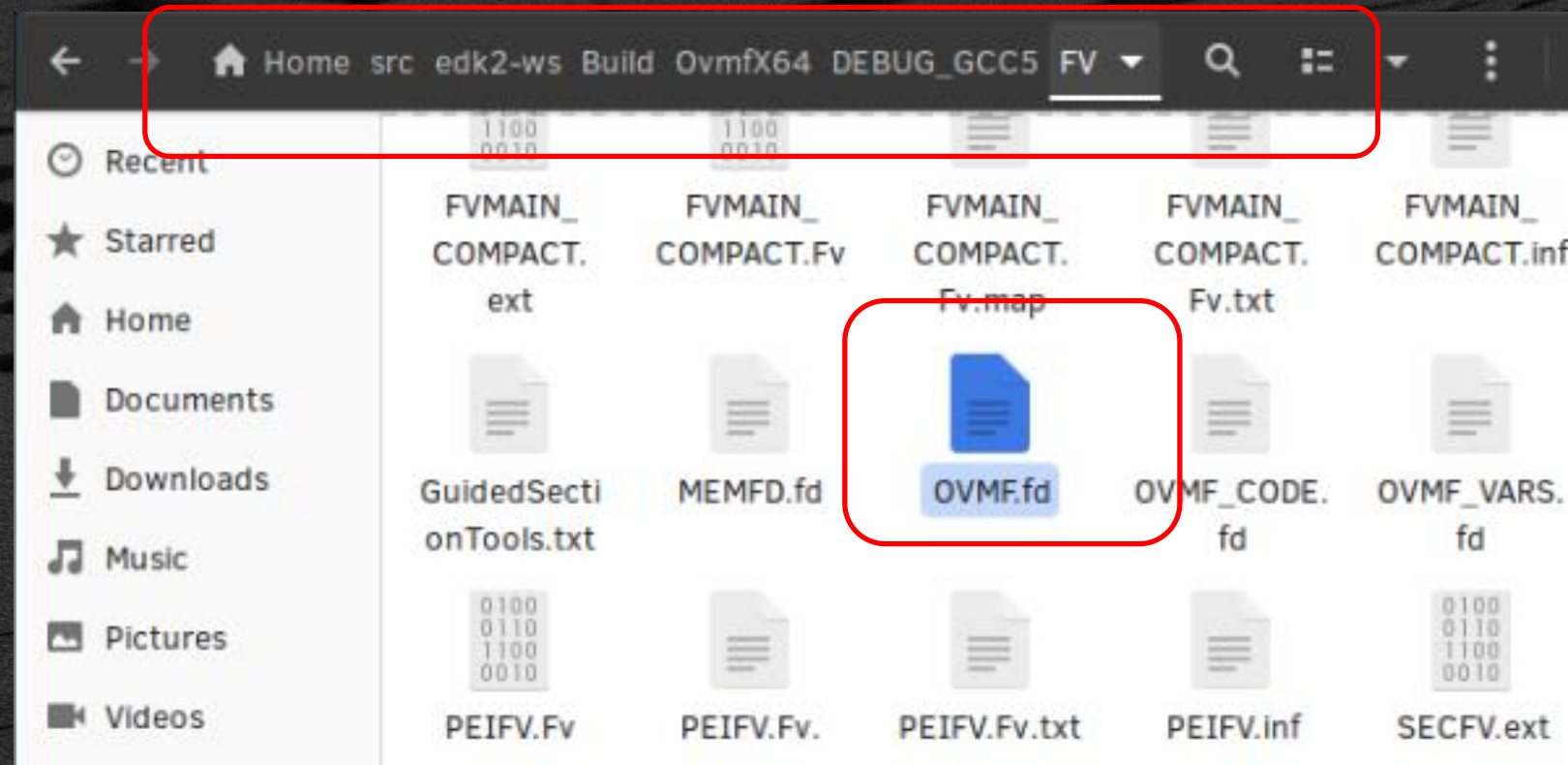
BUILD EDK II OVMF

-Verify Build Succeeded

OVMF.fd should be in the Build directory

- For GCC5 with X64, it should be located at

```
~/src/edk2-ws/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd
```





Change to run-ovmf directory under the home directory

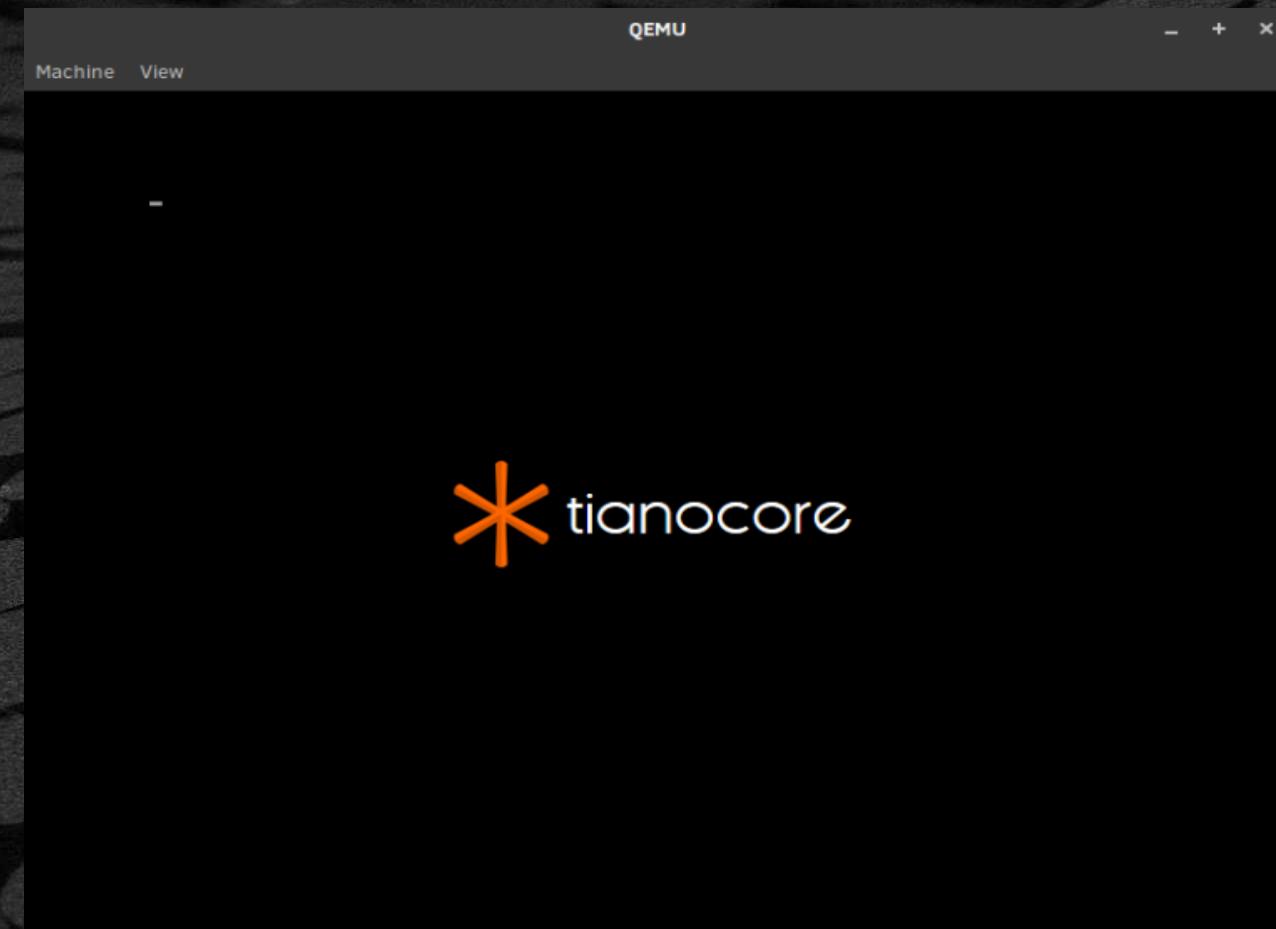
```
bash$ cd $HOME/run-ovmf
```

Copy the OVMF.fd BIOS image created from the build to the run-ovmf directory naming it bios.bin

```
bash$ cp ~/src/edk2-  
ws/Build/OvmfX64/DEBUG_GCC5/FV/OVMF.fd  
bios.bin
```

Run the RunQemu.sh Linux shell script

```
bash$ . RunQemu.sh
```



SUMMARY

- ✱ Build a EDK II Platform using OVMF package
- ✱ Run Ovmf using Qemu 

Questions?

Return to Main Training Page



Return to Training Table of contents for next presentation [link](#)

