**Prep**

* 1. Install Windows Pro 1903
  2. Install Windows ADK 1903
  3. Install ADK WinPE Addon

**Prep for lab machines being SL2s**

* 1. Download SL2 Driver MSI
  2. Expand SL2 Driver MSI using:  
       
     msiexec /a <<SL2 Driver MSI name>> /targetdir=c:\SLDrivers
  3. Download SL2 Recovery Image
  4. Expand SL2 Recovery Image into c:\SLRecovery

**Prep for WinPE**

* 1. Open the Deployment and Imaging Tools Environment as administrator
  2. Execute:  
       
     copype amd64 C:\WinPE\_amd64
  3. Change to the C:\WinPE\_amd64 directory
  4. Add the SL2 drivers to the WinPE image:  
       
     md offline  
     Dism /Mount-Image /ImageFile:C:\WinPE\_amd64\media\sources\boot.wim /MountDir:C:\WinPE\_amd64\offline /Index:1  
     Dism /Image:C:\WinPE\_amd64\offline /Add-Driver /Driver:c:\SLDrivers /Recurse  
     Dism /Unmount-Image /MountDir:C:\WinPE\_amd64\offline /Commit
  5. Place a USB drive of at least 128Gb, 256Gb preferable, into the machine and create a 4GB FAT32 primary partition:  
       
     diskpart  
     lis dis  
     sel dis X (where X is the disk number for the USB)  
     cle  
     cre par pri size=4096  
     Sel par 1  
     act  
     format fs=FAT32 quick  
     ass  
     exit
  6. Make the WinPE drive:  
       
     MakeWinPEMedia /UFD C:\WinPE\_amd64 <DriverLetter>: (the letter that the drive has assigned to it e.g. e)
  7. Use Disk Manager to create a second volume on the USB with the remaining space.
  8. Copy the required files from the recovery image into that new partition (F: in my case)  
       
     robocopy C:\SLRecovery\sources\ f:\SWMFiles \*.swm /v
  9. Copy the partition creation txt file to the WinPE drive. If you are not using SL2 devices then you should modify this file to reflect the structure that you want. A default version of this can be found at <https://docs.microsoft.com/en-us/windows-hardware/manufacture/desktop/boot-to-vhd--native-boot--add-a-virtual-hard-disk-to-the-boot-menu> - Scroll down to the "clean and prepare section" and use the steps in the appropriate BIOS/UFEI entry:  
       
     copy c:\SLRecovery\sources\CreatePartitions-UEFI.txt e:\
  10. Eject the USB Drive

**Install the host machine and create the new deployment files**

* 1. Boot using the WinPE drive on the Lab machine
  2. Using Notepad from the WinPE environment, modify the partition creation text file to select the correct disk (for SL2s that is 0) and clean it. Add these commands to the start of the file:  
       
     sel dis 0  
     cle
  3. Save the file and execute:  
       
     diskpart /s CreatePartitions-UEFI.txt
  4. If the disk numbers are correct then you will have created the appropriate partitions for deploying the host OS in preparation for creating the final host image. If there were errors then you will need to go back and confirm that the disk numbers and script are correct.
  5. Deploy the Host OS using the recovery SWM files:  
       
     Dism /Apply-Image /ImageFile:E:\SWMFiles\install.swm /SWMFile:E:\SWMFiles\install\*.swm /Index:1 /ApplyDir:W:\
  6. Configure the new deployment to boot and give it a reasonable name (I use Host and the drive letters created with the partition creation text file):  
       
     w:  
     cd w:\windows\system32  
     bcdboot w:\windows /s S: /f UEFI  
     bcdedit /set {default} description Host
  7. Exit the WinPE environment and boot into Windows on the Host
  8. Run through the Windows OOBE and create an Offline account to log in. I use HOL with no password.
  9. Using the upgrade option on the "Download Windows 10" page to force an upgrade to 1903. After the upgrade run Windows Update to get all the updates. After the updates, open the Microsoft Store and ensure that all apps are updated.
  10. Boot from WinPE again
  11. Capture the image using:  
        
      Dism /Capture-Image /ImageFile:"e:\HOLImage.wim" /CaptureDir:C:\ /Name:HOLLab
  12. Switch to the WinPE boot drive (D in my case) and use notepad to create a DeployHost.cmd with the following entries:   
        
      diskpart /s CreatePartitions-UEFI.txt   
      dism /Apply-Image /ImageFile:e:\HOLImage.wim /Index:1 /ApplyDir:w:\   
      w:   
      cd \Windows\System32   
      bcdboot w:\Windows /s S: /f UEFI   
      bcdedit /set {default} description Host
  13. Exit from the WinPE shell and confirm that the system boots correctly.

**Install the VHD**

* 1. Boot into WinPE again
  2. Execute the following commands:  
       
     c:  
     md HOLFiles  
     diskpart  
     create vdisk file=c:\HOLFiles\HOL.VHD maximum 256000 type=expandable  
     attach vdisk  
     cre par pri  
     format quick label=HOL-VHD  
     assign letter=V  
     lis vol
  3. Before you exit diskpart find the volume that corresponds to the System volume and then execute:  
       
     sel vol <<Volume Number>>  
     assign letter=s  
     exit
  4. Apply the same image again - Note the change in drive letter:  
       
     dism /Apply-Image /ImageFile:e:\HOLImage.wim /Index:1 /ApplyDir:v:\
  5. Execute the following commands:  
       
     v:  
     cd \Windows\System32   
     bcdboot v:\Windows /s S: /f UEFI   
     bcdedit /set {default} description "HOL Lab"
  6. Exit WinPE and confirm that there are 2 boot entries and you can boot into "HOL Lab"

**Install the required tooling and source for the HOL**

* 1. Rename the system HOLLab
  2. Download <https://www.nxp.com/webapp/sps/download/license.jsp?colCode=IMX_CST_TOOL> to a USB drive
  3. Create the C:\HOLFirmware directory on the VHD
  4. Copy the CST file into C:\HOLFirmware
  5. Turn on the Windows Subsystem for Linux in the Windows Features applet and reboot as required.
  6. Open the Microsoft Store app and install Ubuntu LTS 18.04
  7. Create an account called demo with a password of demo
  8. Turn on "Use Ctrl+Shift+C/V as Copy/Paste" for the Ubuntu shell.
  9. Open the Ubuntu shell with "Run as Administrator"
  10. Execute the following commands:

sudo apt-get update

sudo apt-get upgrade

sudo apt-get install build-essential python python-dev python-crypto python-wand device-tree-compiler bison flex swig iasl uuid-dev wget git bc libssl-dev python3-setuptools python3 attr python3-pip

sudo pip3 install pyelftools

pushd ~

wget <https://releases.linaro.org/components/toolchain/binaries/6.4-2017.11/arm-linux-gnueabihf/gcc-linaro-6.4.1-2017.11-x86_64_arm-linux-gnueabihf.tar.xz>

tar xf gcc-linaro-6.4.1-2017.11-x86\_64\_arm-linux-gnueabihf.tar.xz

rm gcc-linaro-6.4.1-2017.11-x86\_64\_arm-linux-gnueabihf.tar.xz

popd

cd /mnt/c/HOLFirmware

mkdir u-boot

setfattr -n system.wsl\_case\_sensitive -v 1 u-boot

* 1. Modify the ~/.bashrc file to include the line:  
       
     export CROSS\_COMPILE=/home/demo/gcc-linaro-6.4.1-2017.11-x86\_64\_arm-linux-gnueabihf/bin/arm-linux-gnueabihf-
  2. Install the CST  
       
     tar xf cst-3.1.0.tgz  
     mv release cst  
     rm cst-3.1.0.tgz
  3. Exit the Ubuntu Shell.
  4. Install Visual Studio 2017. Select UWP and Desktop C++ workloads. Esnure these additional options are selected:

VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM)

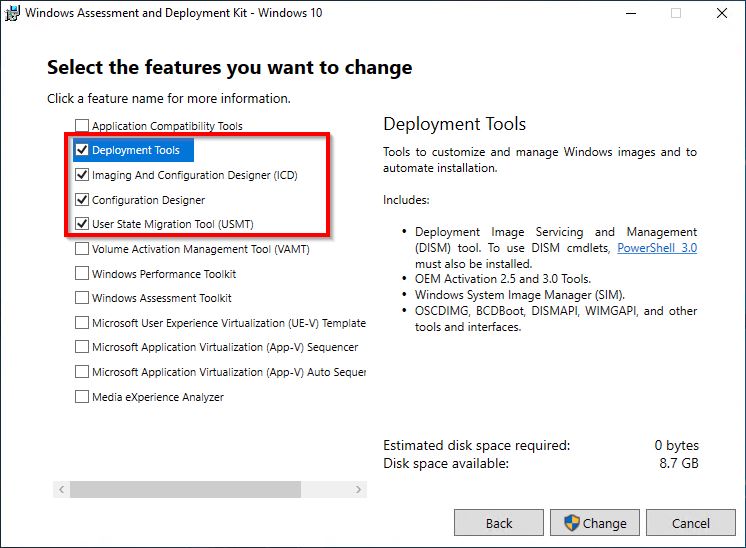
VC++ 2017 version 15.9 v14.16 Libs for Spectre (ARM64)

VC++ 2017 version 15.9 v14.16 Libs for Spectre (X86 and x64)

Visual C++ compilers and libraries for ARM

Visual C++ compilers and libraries for ARM64  
Git for Windows

* 1. Install the Windows ADK 1809 (All these tools can be found from this page <https://docs.microsoft.com/en-us/windows-hardware/manufacture/iot/set-up-your-pc-to-customize-iot-core>)



* 1. Set Visual Studio to run as Administrator
  2. Install the WinPE Add-on for 1809
  3. Install the WDK 1809 including the VSIX.
  4. Install the Windows 10 IoT Core Packages (install all architectures as this removes a warning from the IoTPShell)
  5. Install the IoT Core ADK Add-ons (use a developer command prompt and do the clone in C:\ to setup the directory in the Root folder)
  6. Install the Windows 10 IoT Core Dashboard
  7. Ensuring that the Ubuntu shell is running as Administrator, execute:  
       
     cd /mnt/c/HOLFirmware  
     git clone --recursive <https://github.com/ms-iot/imx-iotcore.git>

git clone --recursive -b u-boot-imx <https://github.com/ms-iot/u-boot.git>

git clone -b ms-iot <https://github.com/ms-iot/optee_os.git>

git clone --recursive -b tcps-feature <https://github.com/Microsoft/RIoT.git>

git clone -b imx <https://github.com/ms-iot/imx-edk2-platforms.git>

git clone --recursive <https://github.com/ms-iot/edk2>

* 1. Exit the Ubuntu Shell
  2. Open an Administrator Command Prompt and change directory to C:\HOLFirmware\imx-iotcore\build\tools
  3. Run StartBuildEnv.bat and in the newly opened shell run C:\HOLFirmware\imx-iotcore\build\tools\SetupCertificate.bat
  4. Exit all shells
  5. Download and extract DD for Windows (<http://www.chrysocome.net/downloads/dd-0.6beta3.zip>) into the C:\HOLFirmware directory
  6. Turn "Developer Mode" on.

**Confirm that the system is installed correctly**

* 1. Remove any removable media (SD Cards, USB Drives etc)
  2. Open Visual Studio
  3. Open the Output Window from the View menu
  4. Open C:\HOLFirmware\imx-iotcore\build\solution\iMXPlatform\iMXPlatform.sln
  5. Ensure the solution is clean by right clinking on it and selecting Clean Solution
  6. Right click on the GenerateBSP Project and build the BSP - This should complete without error
     1. Note: There seems to be some sensitivity with the network connection and SignTool. If you do get errors trying cleaning and building again before looking into the error source.
  7. Build the solution - This should generate an FFU without error
     1. Note: There seems to be some sensitivity with the network connection and SignTool. If you do get errors trying cleaning and building again before looking into the error source.
  8. Close Visual Studio - Do not clean the solution before exiting as Lab 1 depends on the BSP being built.
  9. Open the Ubuntu Shell
  10. BUG: Modify the HummingBoardEdge Makefile to include the flags to turn off fTPM  
        
      nano /mnt/c/HOLFirmware/imx-iotcore/build/firmware/HummingBoardEdge\_iMX6Q\_2GB/Makefile  
        
      Add flag "EDK2\_FLAGS=-D CONFIG\_NOT\_SECURE\_UEFI=1" directly below the other EDK2 settings.
  11. Execute the following:  
        
      cd /mnt/c/HOLFirmware/imx-iotcore/build/firmware/HummingBoardEdge\_iMX6Q\_2GB/

make

* 1. If this builds without issues then the tooling and main source for the Lab VHD is correctly setup.

**Final configuration steps**

* 1. Place a link to the instruction site on the desktop - This is a public site and only provides the instructions and related links  
       
     <https://github.com/elev8-consulting/IOTCoreHOL>
  2. STEPS REDACTED
  3. Finally setup the Start menu to include only Visual Studio, Developer Command Prompt, Ubuntu, Deployment and Imaging Tools and the Windows IoT Core Dashboard.

**Prepare for VHD Deployment**

* 1. Clean the VHD drive by right clicking on it, selecting Properties, Disk Cleanup, Clean System Files and then selecting everything to be cleaned.
  2. Shutdown Windows and boot into WinPE
  3. Copy the VHD back to the USB key with:  
       
     robocopy c:\HOLFiles e:\HOLFiles /v
  4. Using Notepad create a DeployVHD.CMD with the following lines:

robocopy e:\HOLFiles w:\HOLFiles /v

diskpart /s AttachVHD.txt

c:

cd \Windows\System32

bcdboot c:\Windows /s S: /f UEFI

bcdedit /set {default} description "HOL Lab"

* 1. Using Notepad create a AttachVHD.txt file with these commands:  
       
     select vdisk file=w:\HOLFiles\HOL.vhd

attach vdisk

Exit

* 1. Using Notepad create a UpdateVHD.cmd file with this command:  
       
     robocopy e:\Holfiles c:\HOLFiles /v
  2. The USB drive should now be complete. Boot from USB and test the drive by deploying the host, deploying the VHD and then updating the VHD.