

WORKSHOP

# Data in use Protection Compass

Keep the cape in the Cloud and on the Edge

N

## Confidential Computing hands-on lab

Hands-on lab step-by-step

Version 1.0 (Alpha) - June 2020

<https://aka.ms/DataInUseProtectionWS>



Information in this document, including URL and other Internet Web site references, is subject to change without notice. Unless otherwise noted, the example companies, organizations, products, domain names, e-mail addresses, logos, people, places, and events depicted herein are fictitious, and no association with any real company, organization, product, domain name, e-mail address, logo, person, place or event is intended or should be inferred. Complying with all applicable copyright laws is the responsibility of the user.



Attribution 4.0 International (CC BY 4.0)

Microsoft and any contributors grant you a license to this document under the [Creative Commons Attribution 4.0 International Public License](#), see the [LICENSE](#) file, and grant you a license to any code in the repository under the MIT License, see the [LICENSE-CODE](#) file.

Microsoft, Windows, Microsoft Azure and/or other Microsoft products and services referenced in the document may be either trademarks or registered trademarks of Microsoft in the United States and/or other countries. The license for this document does not grant you rights to use any Microsoft names, logos, or trademarks. Microsoft's general trademark guidelines can be found at <http://go.microsoft.com/fwlink/?LinkID=254653>.

Privacy information can be found at <https://privacy.microsoft.com/en-us/>

Microsoft and any contributors reserve all other rights, whether under their respective copyrights, patents, or trademarks, whether by implication, estoppel or otherwise.

# Table of contents

<b>ABSTRACT AND LEARNING OBJECTIVES.....</b>	<b>1</b>
Overview.....	1
Hands-on lab requirements.....	1
Help references.....	2
<b>BEFORE THE HANDS-ON LAB.....</b>	<b>3</b>
Task 1: Access your lab environment.....	3
Task 2: Start a VM in the lab.....	3
Task 3: Connect to the Windows-based development VM in the lab.....	3
Task 3bis: Connect to a Linux-based testing VM in the lab.....	4
<b>EXERCISE 0: INSTALL A (CROSS-PLATFORM) DEVELOPMENT ENVIRONMENT .....</b>	<b>6</b>
Task 1: Connect to the development VM.....	6
Task 2: Update Visual Studio.....	6
Task 3: Install the Intel SGX SDK for Windows.....	7
Task 4: Install Visual Studio Code and the Open Enclave extension.....	7
Task 5: Connect to the Linux-based testing VM.....	8
Task 6: Install the Intel SGX SDK for Linux.....	8
Task 7: Install the Open Enclave SDK.....	9
<b>EXERCISE 1: MASTER THE BASICS.....</b>	<b>11</b>
Task 1: Run your first SGX enclave (simulated) on the Windows-based development VM.....	11
Task 2: Use Intel SGX SDK samples on the Linux-based Linux.....	12
Task 3: Use Open Enclave SDK samples on the Linux-based Machine.....	12
<b>EXERCISE 2: LIFT AND SHIFT A COMPLETE APP .....</b>	<b>13</b>
Task 1: Install and use SGX-LKL Library OS.....	13
Task 2: Test-drive the Fortanix Confidential Computing Enclave Manager (optional).....	14
Task 3: Understand the Anjuna Enterprise Enclaves for Azure Confidential Computing (optional).....	16
<b>EXERCISE 3: ATTEST AN ENCLAVE.....</b>	<b>18</b>
Task 1: Locally (intra-platform) attest an enclave.....	18
Task 2: Remotely (inter-platform) attest an enclave.....	18

Task 3: Test-drive the local vs. remote attestation mechanisms..... 19

**AFTER THE HANDS-ON LAB.....20**

Task 1: Stop a VM ..... 20

**APPENDIX .....21**

Linux VMs private key ..... 21

# Abstract and learning objectives

In this hands-on lab, you will further learn about to make your applications more secure by leveraging [Azure Confidential Computing](#).

[Intel SGX](#)-enabled [DCsv2-series virtual machines](#) (VM) generally available as of this writing allow to:

- Safeguard data from malicious and insider threats while it's in use.
- Maintain control of data throughout its lifetime.
- Protect and validate the integrity of code in the cloud.
- Ensure that data and code remain outside the view of the cloud platform provider.

## Overview

The **Confidential Computing hands-on lab** is a series of exercises that will walkthrough you through the development of new vs. refactored applications that leverage Azure Confidential Computing as well as the "Lift & Shift" model for existing applications.

The hands-on lab can be implemented on your own. Please note that this hands-on lab does not contain a complete set of step-by-step instructions, but rather, to achieve this, refers to:

- Existing documents, and more especially the series of developer guides "[Embracing as a developer the new perspectives of Confidential Computing](#)".
- Specific GitHub repos and theirs (Mark Down) readme and quick starts.
- Specific Web pages.

## Hands-on lab requirements

- A [Microsoft account](#).
- An Azure subscription. If you don't have an Azure subscription, create a [free account](#) before you begin.
- A DevTest lab environment provisioned with a set of two virtual machines (VMs) (**completed the day before the lab!**):
  - One of these VMs is a Windows-based development environment configured with:
    - [Visual Studio Community 2017](#) installed as code editor (along with the Intel SGX SDK pre-installed).
    - A terminal console which allows you to both:
      - Execute Git commands, such as [Git for Windows](#) (2.10 or later).
      - Remotely connect to a VM in SSH, such as [OpenSSH](#), [PuTTY](#).
  - The other VM is an Intel SGX-enabled/capable Linux-based testing VMs.

You will have to create and prepare two VMs, namely:

Name of the VM	OS of the VM	Type of the VM
CC-HOL-WDEV-01	Windows (Windows Server 2016)	<a href="#">Standard D2 v3</a>
CC-HOL-LTEST-01	Linux (Ubuntu 18.04 LTS)	<a href="#">Standard DC2s v2</a>

## Help references

Description	Links
Building and executing Trusted Execution Environment (TEEs) based applications on Azure – A starter guide for developers	<a href="https://download.microsoft.com/download/e/0/3/e03ff018-35ac-4bac-9638-3e45787bc23c/Building-and-Executing-TEE-based-applications-on-Azure-(April-2020).pdf">https://download.microsoft.com/download/e/0/3/e03ff018-35ac-4bac-9638-3e45787bc23c/Building-and-Executing-TEE-based-applications-on-Azure-(April-2020).pdf</a>
Leveraging Attestations with Trusted Execution Environment (TEE) based applications on Azure - A starter guide for developers	<a href="https://download.microsoft.com/download/e/0/3/e03ff018-35ac-4bac-9638-3e45787bc23c/Leveraging-Attestations-with-TEE-based-applications-on-Azure-(April-2020).pdf">https://download.microsoft.com/download/e/0/3/e03ff018-35ac-4bac-9638-3e45787bc23c/Leveraging-Attestations-with-TEE-based-applications-on-Azure-(April-2020).pdf</a>

# Before the hands-on lab

**Duration:** 5-10 minutes

In this pre-exercise, you will set up your environment, i.e. reclaim the virtual machines (VM) for use in the rest of the hands-on lab. You should follow all steps provided *before* attending the hands-on lab.

**IMPORTANT:** Most Azure resources require unique names. Throughout this lab you will see the word "SUFFIX" as part of resource names. You should replace this with the value defined as part of the provisioning of your DevTest Lab environment to ensure the resource is uniquely named.

## Task 1: Access your lab environment

1. Launch a browser and navigate to <https://portal.azure.com>. Once prompted, login with your Microsoft Azure credentials. If prompted, choose whether your account is an organizational account or just a Microsoft Account. Enter the related credentials to sign in.
2. In the [Azure Portal](#), open the resource group named **COMPASS-HOL-RG**.
3. Click **DevTest Lab** to access the hands-on lab environment that has been prior provisioned for you.

## Task 2: Start a VM in the lab

1. To start a listed Windows virtual machine (VM), right-click the intended VM row under **Claimable virtual machines** and select **Claim machine**. This will start the VM.
2. When the VM is started, it will be displayed in the **My Virtual Machines** pane. After one minute, the status will be **Running**.

You can now connect to the VM. To do so:

- With the **CC-HOL-WDEV-01** Windows lab VM, follow the instructions as per **Task 3: Connect to the Windows-based development VM in the lab**.
- With the **CC-HOL-LTEST-01** Linux lab VM, follow instead the instructions as per **Task 3bis: Connect to a Linux-based testing VM in the lab**.

## Task 3: Connect to the Windows-based development VM in the lab

1. Under **My Virtual Machines**, select the **CC-HOL-WDEV-01** lab VM row.
2. At the end of line click ..., and then select **Connect**.

RDP SSH BASTION

**Connect with RDP**

To connect to your virtual machine via RDP, select an IP address, optionally change the port number, and download the RDP file.

IP address \*

Public IP address (52.166.88.167) ▼

Port number \*

3389

**Download RDP File**

Can't connect?

[Test your connection](#)

[Troubleshoot RDP connectivity issues](#)

3. Select **Download RDP file**, then open the downloaded RDP file.
4. Click **Connect** on the Remote Desktop Connection dialog.
5. When invited, specify the username and password as detailed in each exercise in the rest of this document. (*Do not use your organizational account or your Microsoft Account one.*)

Below is the list of username and password of administrator account on the Windows VM:

Name of VM	Username	Password
CC-HOL-WDEV-01	azuredemo	Password.1!!

6. Click **Yes** to connect, if prompted that the identity of the remote computer cannot be verified.

At the stage, you should be connected on the Windows desktop of the **CC-HOL-WDEV-01** lab VM.

## Task 3bis: Connect to a Linux-based testing VM in the lab

1. First, save the provided private key to a new text file, for example `keyfile`.

**Note** Typical SSH private key files don't have any file extension. This however doesn't really matter and if you can't save the file without an extension, any non-binary extension will do. A widely used convention on Windows-based systems consist of using **.pk** for private key and **.pubk** for public keys.

2. Under **My Virtual Machines**, select the intended VM row, i.e. the one for the **CC-HOL-LTEST-01** lab VM.
3. At the end of line click **...**, and then select **Connect**. A SSH connection string will be displayed.

[Connect](#)
[Start](#)
[Restart](#)
[Stop](#)
[Artifacts](#)
[Claim machine](#)
[Unclaim](#)
[Delete](#)

**Running**

Resource group : [acclab22611888606000](#)

Virtual network/subnet : [DtlACCLab2/DtlACCLab2Subnet](#)

IP address or FQDN : [acclab22611888606000.uksouth.cloudapp.azure.com](#)

NAT protocol / Port to co... : SSH [58131](#)



- Now open on your local machine, a prompt command line and enter the provided SSH connection string, using the path to `keyfile` as the `<private key path>` argument. For example:

```
C:\> ssh -i ~/keys/keyfile azureadmin@<FQDN> -p <PORT>
```

- When prompted for the authenticity check, ensure the fingerprint is either  
SHA256: yW4AyDfXb66iVz64rhGU/KyUMQFPatKsHmJ2m2Hby1U Or  
MD5: 71:88:bb:ed:8a:96:2c:f4:64:21:8b:ec:05:7a:b7:61, and then type "yes".

**(REPLACE these values with your own ones.)**

At the stage, you should be connected on the Linux lab VM with a Bash shell.

**You should follow all steps provided *before* attending the Hands-on lab.**

**You are now ready to start Exercise 0! And setup your development environment.**

# Exercise 0: Install a (cross-platform) development environment

**Duration:** 40 minutes

As the title indicates, this exercise aims at installing a cross-platform development environment that encompasses both the **CC-HOL-WDEV-01** lab VM and the **CC-HOL-LDEV-01** one with:

- The relevant integrated development environment(s) (IDE), along with the suitable extension/pug-ins if any.
- The key Software Development Kit (SDK) in the context, namely the [Intel SGX SDK](#) (for both Windows and Linux) and the [Open Enclave SDK](#) (for Linux).

**Important note** A Windows package does exist for the Open Enclave SDK. However, a system with support for SGX with Flexible Launch Control (FLC) is required which is NOT the case of the **CC-HOL-WDEV-01** lab VM. So, it will not be a feature in this hands-on lab.

## Objectives:

- Identify and understand the requirements that pertain to the development of Confidential Computing applications in C/C++.
- Setup and configure all the key SDKs, with the required driver and related tooling to configure the underlying hardware platform or to integrate with the targeted IDEs.
- Prepare a suitable environment for the rest of the exercises suggested in this hands-on lab.

**Difficulty:** 1/5

## Task 1: Connect to the development VM

In this task, you will first connect to the **CC-HOL-WDEV-01** lab VM. Proceed as per **Task 3: Connect to the Windows-based development VM in the lab**.

## Task 2: Update Visual Studio

In this task, you will add update to the Visual Studio Community 2017 installation on your **CC-HOL-WDEV-01** lab VM.

1. On your **CC-HOL-WDEV-01** lab VM, launch **Visual Studio Installer**: press **WIN** key and type "**Visual Studio Installer**".
2. Select **More** and **Modify** for the Visual Studio Community 2017 installation.

**Note** If updates are available, select the Update button first, and install the updates before moving on to installing the additional components. Once the updates are installed, the Update button will be replaced with the **Modify** button.

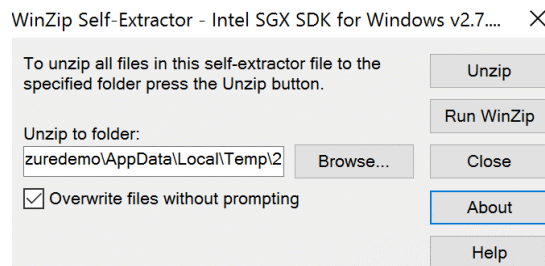
3. Open the Windows Explorer and navigate to the **Downloads** folder.

## Task 3: Install the Intel SGX SDK for Windows

See [Intel® Software Guard Extensions SDK](#), [Get Started with the SDK](#), [Getting Started with Intel® Software Guard Extensions SDK for Microsoft® Windows® OS](#), and [Intel® SGX SDK Installation Guide for Windows](#).

In this task, you will in addition download and install the Intel SGX SDK on your **CC-HOL-WDEV-01** lab VM.

1. On your Lab VM, from the IE browser session, navigate to <https://software.intel.com/sgx>.
2. Click **Download the SDK**.
3. On the Get started with the SDK page, click **Download Windows SDK**.
4. Sign-in or created an account to get access to the SDK.
5. Click **Agree** for the End-User license Agreement.
6. Select **Intel Software Guard Extensions SDK for Windows**.
7. Click **Intel SGX SDK for Windows v2.7.101.2.exe**. The download starts.



8. Once the SDK has been downloaded, launch the auto extractor, and choose a path to unzip the folder, then click **Unzip**.
9. Go to the folder and double-click the installer file `Intel SGX SDK for Windows v2.7.101.2.exe` to begin the installation process.
10. Ensure that **Intel SGX SDK**, **Intel SGX Visual Studio Debugger** and **Intel SGX Visual Studio 2017 Integration** are ticked and proceed with the installation.

As noticed above, the **CC-HOL-WDEV-01** lab VM does NOT provide support for SGX with Flexible Launch Control (FLC), **so the emulation/simulation mode will be later used**.

## Task 4: Install Visual Studio Code and the Open Enclave extension

In this task, you will in addition download and install Visual Studio Code on your **CC-HOL-WDEV-01** lab VM.

1. Open Visual Studio Code
2. Now install the Open Enclave extension for Visual Studio Code. \*
3. From Visual Studio Code, press **"CTRL" + "SHIFT" + X**
4. Search for "Microsoft Open Enclave" and click **Install**

## Task 5: Connect to the Linux-based testing VM

In this task, you will first connect to the **CC-HOL-LDEV-01** lab VM. Proceed as per **Task 3: Connect to the Windows-based development VM in the lab**.

## Task 6: Install the Intel SGX SDK for Linux

See [Intel® Software Guard Extensions SDK](#), [Get Started with the SDK](#), and [Intel® Software Guard Extensions \(Intel® SGX\) SDK for Linux\\* OS Installation Guide](#).

In this task, now that you're connected to the **CC-HOL-LTEST-01** lab VM, you will install the [Intel SGX SDK for Linux](#).

Intel has open sourced its Software Guard Extensions (SGX) technology for Linux. The SGX Software Development Kit (SDK) for linux is one of the contributed projects to the [Confidential Computing Consortium](#), see [Confidential Computing Consortium Defining and Enabling Confidential Computing](#).

As such, the Linux SGX implementation includes the SGX Platform Software, the SGX SDK and the SGX driver. The two formers are hosted on the [linux-sgx](#) project and the latter on the [linux-sgx-driver](#) one on GitHub.

As already touched with **Task 3**, the Intel SGX SDK is basically a collection of APIs, documentation, sample source code, tools, and libraries. Using them, a developer can create and play with SGX-enabled programs written in C/C++. Current release supports of number of Linux distros like the Ubuntu 18.04 LTS 64-bit version of the **CC-HOL-LTEST-01** lab VM.

Download and install the Intel SGX SDK for Linux as per instructions of the [Intel® Software Guard Extensions \(Intel® SGX\) SDK for Linux\\* OS Installation Guide](#).

You might have to run the following commands before:

```
sudo apt update
sudo apt -y install dkms
```

One should note that the above Intel documentation does not indicate from where to download the files `sgx_linux_x64_driver.bin` et `sgx_linux__x64_sdk_.bin`. These files can be downloaded from <https://01.org/intel-software-guard-extensions/downloads>.

To install the latest version of the SGX driver, run the following commands:

```
wget https://download.01.org/intel-sgx/sgx-linux/2.9.1/distro/ubuntu18.04-server/sgx_linux_x64_driver_1.33.bin -O sgx_linux_x64_driver.bin
chmod +x sgx_linux_x64_driver.bin
sudo ./sgx_linux_x64_driver.bin
```

To install the latest version of the SGX driver, run the following commands:

```
wget https://download.01.org/intel-sgx/sgx-linux/2.9.1/distro/ubuntu18.04-server/sgx_linux_x64_sdk_2.9.101.2.bin -O sgx_linux_x64_sdk.bin
chmod +x sgx_linux_x64_sdk.bin
sudo ./sgx_linux_x64_sdk.bin
```

When prompted, "Do you want to install in current directory? [yes/no] :", type: **no**

When prompted "source /opt/intel/sgxsdk/environment": type: **/opt/intel**

```
source ~/sgxsdk/environment
```

To install the SGX Platform Software, run the following commands:

```
echo 'deb [arch=amd64] https://download.01.org/intel-sgx/sgx_repo/ubuntu bionic main' | sudo tee
/etc/apt/sources.list.d/intel-sgx.list
wget -qO - https://download.01.org/intel-sgx/sgx_repo/ubuntu/intel-sgx-deb.key | sudo apt-key add -

echo "deb http://apt.llvm.org/bionic/ llvm-toolchain-bionic-7 main" | sudo tee
/etc/apt/sources.list.d/llvm-toolchain-bionic-7.list
wget -qO - https://apt.llvm.org/llvm-snapshot.gpg.key | sudo apt-key add -

echo "deb [arch=amd64] https://packages.microsoft.com/ubuntu/18.04/prod bionic main" | sudo tee
/etc/apt/sources.list.d/msprod.list
wget -qO - https://packages.microsoft.com/keys/microsoft.asc | sudo apt-key add -

sudo apt update
```

```
sudo apt-get install libsgx-launch libsgx-urts
```

When prompted, type "Y"

```
sudo apt-get install libsgx-epid libsgx-urts
```

When prompted, type "Y"

```
sudo apt-get install libsgx-quote-ex libsgx-urts
```

## Task 7: Install the Open Enclave SDK

See [Install the Open Enclave SDK \(Ubuntu 18.04\)](#), [Install the Open Enclave SDK \(OE SDK\)](#) in [Quickstart: Deploy an Azure confidential computing VM in the Azure portal](#).

While still connected to the **CC-HOL-LTEST-01** lab VM, in this task, you will now install the [Open Enclave SDK](#).

The Open Enclave (OE) SDK is a hardware-agnostic open source library for developing applications that utilize hardware-based trusted execution environments (TEEs), a.k.a. enclaves.

As such, this OE SDK aims to generalize the development of enclave applications across TEEs from different hardware vendors. The current implementation provides support for Intel SGX as well as preview support for OP-TEE OS on ARM TrustZone.

**Important note** The Open Enclave SDK do NOT require a prior installation of the Intel SGX SDK. There is NO dependency between the two.

As an [open source project](#) also (hosted in GitHub), this OE SDK also strives to provide a transparent solution that is agnostic to specific vendors, service providers and choice of operating systems. The OE SDK is yet another of the contributed projects to the above [Confidential Computing Consortium](#).

Follow the steps indicated on the [Open Enclave website](#) for Ubuntu Server 18.04. Specific instructions for the latest version (0.9 SDK, 1.32 driver as April 2020) are located [here](#).

**Note** You should also install **pkg-config** as it is required by any project using SGX and **isn't** listed in the Intel tutorial

**You should now move to the next exercise.**

# Exercise 1: Master the basics

**Duration:** 40 minutes

Based on the prior configured cross-platform development environment, this exercise now aims at covering the basics of the development of trusted secure application on top of an Intel SGX capable platform. Such a development is suitable for brand new applications in C/C++ as well as for refactoring existing C/C++ applications with a trusted portion to run inside a hardware-based trusted execution environment (TEE), a.k.a. enclave.

## Objectives:

From both the Linux and the Windows operation systems (OS):

- Understand (and dig a little bit into) fundamentals.
- Create a trusted portion of an existing C/C++ application to run inside a hardware-based TEE/enclave with "the building blocks", thanks to the mains SDKs, i.e. the [Intel SGX SDK](#) and the [Open Enclave SDK](#).

**Difficulty:** 2/5

## Task 1: Run your first SGX enclave (simulated) on the Windows-based development VM

See [Intel® SGX Developer Guide for Windows](#) and [Intel® SGX SDK Developer Reference for Windows](#).

In this task, you will first connect to the **CC-HOL-WDEV-01** lab VM, build your first SGX enclave project on Windows and execute it in the debug mode.

1. To connect the **CC-HOL-WDEV-01** lab VM, proceed as per **Task 3: Connect to the Windows-based development VM in the lab**.
2. Launch Visual Studio 2017.
3. Click on **Not Now, Maybe later**.
4. Click on **Start Visual Studio**. You might be invited to sign in to unlock the IDE, and potentially start using your Azure credits if any, publish code to a (private) Git repository sync your settings, etc. you can use a Microsoft account to do so. You can freely create an account [here](#).
5. Open the **SampleEnclave** project under `C:\Program Files`  
`(x86)\Intel\IntelSGXSDK\src\SampleEnclave`.

**The SGX enclave should be run in a simulated mode as the underlying hardware isn't SGX-capable.**

You may refer to the following documentation for further explanations : [Getting Started with Intel® Software Guard Extensions SDK for Microsoft\\* Windows\\* OS](#) / [An introduction to creating a sample enclave using Intel® Software Guard Extensions](#)

6. Set some breakpoints and run the **SampleEnclave** project in debug mode to observe the calls (ECALL and OCALL) between the host application and the enclave.

## Task 2: Use Intel SGX SDK samples on the Linux-based Linux

See [Intel® SGX Developer Guide for Linux](#) and [Intel® Software Guard Extensions \(Intel® SGX\) SDK for Linux\\* OS Developer Reference](#).

In this task, you will now connect to the **CC-HOL-LTEST-01** lab VM, and test one of the samples that comes along with the Intel SGX SDK.

1. To connect the **CC-HOL-LTEST-01** lab VM, proceed as per **Task 3bis: Connect to a Linux-based testing VM in the lab**.
2. Build and run the **SampleEnclave** sample project located under the `SampleCode` folder as per instructions of the `readme.txt` file (<https://github.com/intel/linux-sgx/tree/master/SampleCode/SampleEnclave>). The above folder is located under `/opt/intel/sgxsdk/SampleCode/SampleEnclave`.
3. Type the following commands:

```
cd /opt/intel/sgxsdk/SampleCode/SampleEnclave
sudo make
./app
```

4. You will observe the following result:

```
Checksum(0x0x7ffdade1b370, 100) = 0xfffd4143
Info: executing thread synchronization, please wait...
Info: SampleEnclave successfully returned.
Enter a character before exit ...
```

## Task 3: Use Open Enclave SDK samples on the Linux-based Machine

In this task, you will continue to work on the **CC-HOL-LTEST-01** lab VM.

1. Prepare your Linux environment as per section **Using the Open Enclave SDK** of the [Building and executing Trusted Execution Environment \(TEEs\) based applications on Azure – A starter guide for developers](#). This should be the case at this stage of the hands-on lab. (Optional)
2. Follow the guide [Building Open Enclave SDK Samples on Linux](#) and build one sample.

**You should now move to the next exercise.**



## Exercise 2: lift and shift a complete app

**Duration:** 45 minutes

Beside the development of new/refactored applications for which you should now have a glimpse of the related considerations, this exercise aims at illustrating an alternate/complementary approach, i.e. the "Lift & Shift" model where an existing applications can just works inside hardware-based TEEs/enclaves.

Such a model typically lays on top of Library OS strategy with secure containers and/or leverages ISVs.

**Objectives:**

- Understand model fundamentals and key principles.
- Illustrate the Library OS strategy through the lens of the [SGX-LKL](#) project on GitHub, one of the key open source project to consider in this space.
- Feature an existing ISV solution available in the Microsoft Azure Marketplace for Azure Confidential Computing.

**Difficulty:** 2/5

### Task 1: Install and use SGX-LKL Library OS

In this task, you will install the SGX-LKL Library OS on the **CC-HOL-LTEST-01** lab VM for running a sample Linux application inside of Intel SGX enclaves.

As introduced above, SGX-LKL is a library OS designed to run unmodified Linux binaries inside intel SGX enclaves. More specifically, the goal of this open source project is to provide the necessary system support for complex applications (e.g., TensorFlow, PyTorch, and OpenVINO) and programming language runtimes (e.g., Python, the DotNet CLR and the JVM). SGX-LKL can run these applications in SGX enclaves without modifications or reliance on the untrusted host OS.

As described on the repo itself, the SGX-LKL project includes several components, mainly a:"

- A launcher and host interface modelled after a lightweight VM interface.
- A port of Linux to run in this environment, using the [Linux Kernel Library](#) (LKL) to provide mature system support for complex applications within the enclave.
- A port of the [musl](#) standard C library to run on top of this version of Linux.

SGX-LKL has support for in-enclave user-level threading, signal handling, and paging. System calls are handled within the enclave by LKL when possible, and asynchronous system call support is provided for the subset of system calls that require direct access to external resources and are therefore processed by the host OS.

SGX-LKL can be run in hardware mode, when it requires an Intel SGX compatible CPU, and also in software simulation mode, when it runs on any Intel CPU without hardware security guarantees."

The `master` branch of the project features the Intel SGX driver while the `oe_port` branch contains an experimental port of SGX-LKL to use the Open Enclave SDK as an enclave abstraction layer. This Open Enclave Edition, namely SGX-LKL-OE is an ongoing research project.

1. First connect the **CC-HOL-LTEST-01** lab VM, proceed as per **Task 3bis: Connect to a Linux-based testing VM in the lab**.
  2. Run SGX-LKL in an Intel SGX based TEE/enclave:
    - With the Intel SGX driver, follow all the instructions as per `README.md` file at <https://github.com/lsds/sgx-lkl> to build and install SGX-LKL, and run the **HelloWorld** application, i.e. a simple Java sample application available in the `apps/jvm/helloworld-java` directory of the project.
- or-
- With the Open Enclave Edition, follow instead all the instructions as per `README.md` file at [https://github.com/lsds/sgx-lkl/tree/oe\\_port](https://github.com/lsds/sgx-lkl/tree/oe_port) to build and install SGX-LKL-OE, and run one of the sample applications available in the `samples` directory of the project.

The next two tasks represent an opportunity to test or at least better understand “Lift & Shift” solutions available from ISVs. Pick one of them to discover yet another experience regarding this model.

## Task 2: Test-drive the Fortanix Confidential Computing Enclave Manager (optional)

See [Fortanix Enclave Manager Quickstart Guide](#), [Fortanix Enclave Manager SaaS User Guide](#) and .

In this task, you will further see how to make your existing applications secure and trusted, i.e. confidential without changing a line of code with this ISV solution available in the Azure Marketplace [here](#).

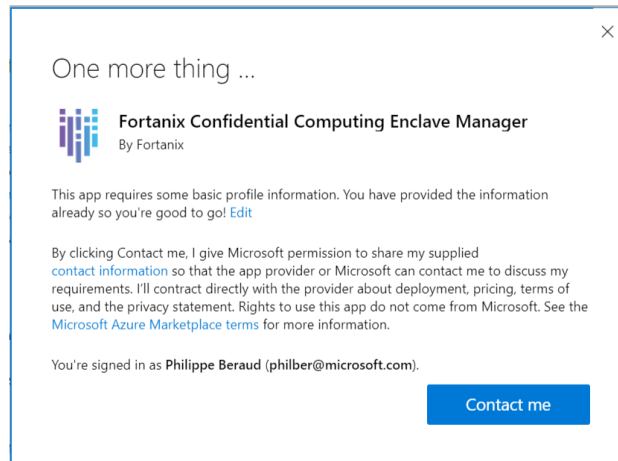
The Fortanix Confidential Computing Enclave Manager enables containerized applications to run in confidential computing environments, verifies the integrity of those environments, and manages the enclave application lifecycle. Fortanix provides the flexibility to run and manage the broadest set of containerized applications, including existing applications, new enclave-native applications, and pre-packaged applications.

All the containerized applications in Fortanix Enclave Manager run on a compute node inside a cluster. Before any compute node can be used, e.g. a [DCsv2 series](#) virtual machine (VM) in Azure Confidential Computing (ACC), it must be enrolled in Enclave Manager. This is referred as to the node enrollment process where a node agent software is deployed onto the target compute node to manage it along with the applications’ images running in enclaves. The node agent is compatible with Enclave Manager and enables running containerized applications in the Intel SGX enclaves.

As of this writing, you can start a free trial of the solution at <https://em.fortanix.com>:

- Sign up for an account.
- Enroll an Intel SGX-enabled compute nodes to Enclave Manager in your account.
- Start running confidential compute workloads after converting the related container image.

Or simply click on **CONTACT ME**.

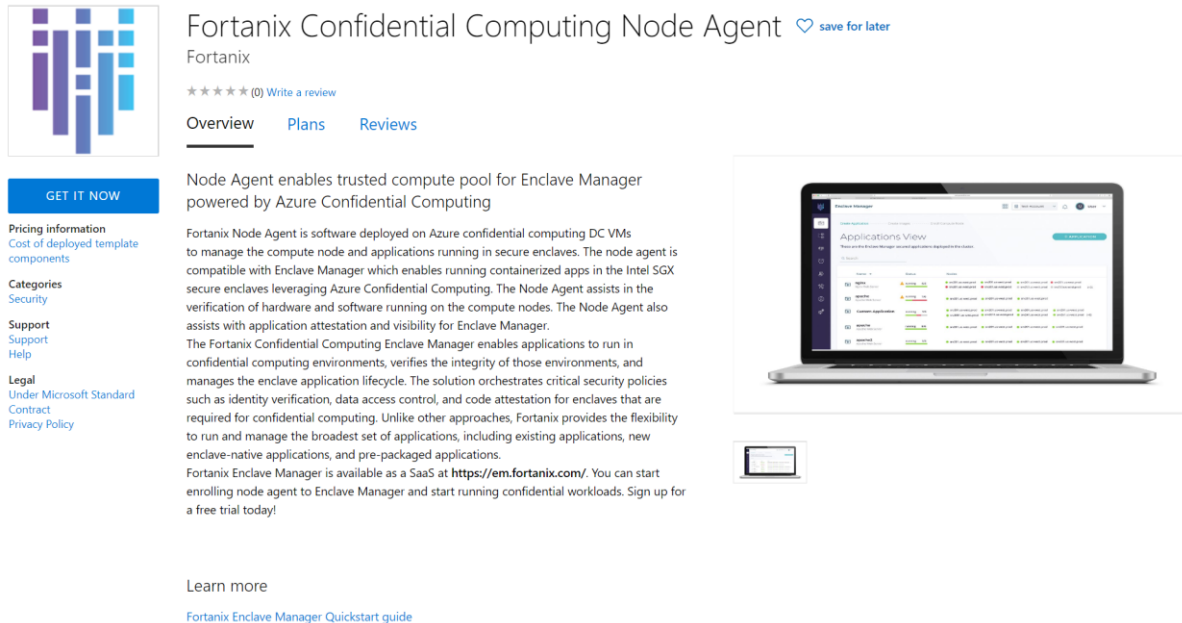


Signing up for an account is an asynchronous process that requires a working day to complete.

**This task assumes that you already signed up in advance for a free trial of the solution.**

Follow all the instructions as per [Fortanix Enclave Manager Quickstart Guide](#). This will guide you through the use of the Fortanix Enclave Manager SaaS portal:

- Create and select an account, i.e. a functional grouping structure for your applications and compute nodes.
- Add a containerized application, here a Python Flask Server application. the [Fortanix public Docker Hub repository](#) will be used for the source containerized image: `fortanix/python-flask`. For the output image, you can use your private [Docker Hub](#) repository or instead your own registry instance of [Azure Container Registry](#) (ACR), see [Docker Hub Quickstart](#) resp. [Quickstart: Create a private container registry using the Azure portal](#).
- Create a Fortanix Enclave Manager image, i.e. a particular software release or a version of an application associated with one enclave hash (MRENCLAVE).
- Enroll a compute node agent:
  - Either visit [Fortanix Confidential Computing Node Agent](#) on the Microsoft Azure Marketplace to create the node agent to register the compute node



**Fortanix Confidential Computing Node Agent** [save for later](#)

Fortanix

★★★★★ (0) [Write a review](#)

[Overview](#) [Plans](#) [Reviews](#)

**GET IT NOW**

**Pricing information**  
Cost of deployed template components

**Categories**  
Security

**Support**  
[Support](#)  
[Help](#)

**Legal**  
[Under Microsoft Standard Contract](#)  
[Privacy Policy](#)

Node Agent enables trusted compute pool for Enclave Manager powered by Azure Confidential Computing

Fortanix Node Agent is software deployed on Azure confidential computing DC VMs to manage the compute node and applications running in secure enclaves. The node agent is compatible with Enclave Manager which enables running containerized apps in the Intel SGX secure enclaves leveraging Azure Confidential Computing. The Node Agent assists in the verification of hardware and software running on the compute nodes. The Node Agent also assists with application attestation and visibility for Enclave Manager.

The Fortanix Confidential Computing Enclave Manager enables applications to run in confidential computing environments, verifies the integrity of those environments, and manages the enclave application lifecycle. The solution orchestrates critical security policies such as identity verification, data access control, and code attestation for enclaves that are required for confidential computing. Unlike other approaches, Fortanix provides the flexibility to run and manage the broadest set of applications, including existing applications, new enclave-native applications, and pre-packaged applications.

Fortanix Enclave Manager is available as a SaaS at <https://em.fortanix.com/>. You can start enrolling node agent to Enclave Manager and start running confidential workloads. Sign up for a free trial today!

[Learn more](#)  
[Fortanix Enclave Manager Quickstart guide](#)

-or-

- Manually download and deploy the [latest node agent](#) on an existing Intel SGX-enabled confidential VM.
- Eventually run the application image on the declared compute node.

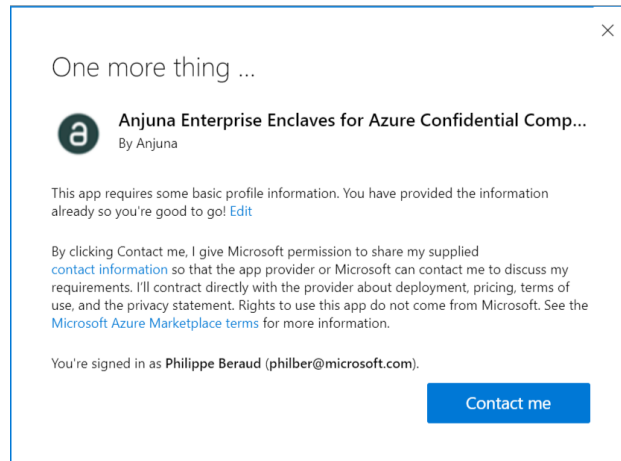
## Task 3: Understand the Anjuna Enterprise Enclaves for Azure Confidential Computing (optional)

See [Anjuna solution brief](#) and [Preventing Insider Threats With Anjuna Enterprise Enclaves](#).

In this task, you will further see how to make your existing applications secure and trusted, i.e. confidential without changing a line of code with this other ISV solution available in the Azure Marketplace [here](#).

Azure Confidential Computing (ACC) instances offer the opportunity to quickly protect any application from insider threats by leveraging Intel SGX-enabled VMs and Anjuna Enterprise Enclaves software. With a single command, Anjuna automatically creates a secure enclave that isolates and encrypts all application resources in runtime, at rest, and on the network, to achieve the strongest end-to-end data protection available. No changes to the application code or SDKs are required.

Prior any use of the solution, you will need also to click on **CONTACT ME** in the related page of the Azure Marketplace to request a demo of the solution and its use case.



For the sake of this task, and as of this writing, simply start by reading the paper [Preventing Insider Threats With Anjuna Enterprise Enclaves](#) to get a better understanding of the solution and the perspectives it opens up.

Continue with the blog post [Compiler-based Techniques for Enhancing Performance and Privacy in Enclaves](#), and the related paper [CoSMIX: A Compiler-based System for Secure Memory Instrumentation and Execution in Enclaves](#) along with presentation videos [1](#) & [2](#) if time permits.

**You should now move to the next exercise.**

## Exercise 3: Attest an enclave

**Duration:** 50 minutes

This exercise aims at illustrating how to attest an enclave and hence providing an attestation.

In the Intel SGX architecture, attestation refers to the process of demonstrating that a specific enclave was established on an Intel SGX-enabled platform. As such, the Intel SGX architecture provides two attestation mechanisms: local (intra-platform) attestation vs. remote (inter-platform) attestation.

The Intel SGX SDK provides APIs used by applications to implement these attestation processes. Same is true for the Open Enclave (OE) SDK.

**Objectives:**

- Understand fundamentals of local vs. remote attestation.
- Test local vs. remote attestation through a sample application.

**Difficulty:** 3/5

### Task 1: Locally (intra-platform) attest an enclave

See:

- Section **Local (Intra-Platform) Attestation** in the [Intel® SGX Developer Guide for Linux](#).
- Section **Local Attestation** in the [Intel® Software Guard Extensions \(Intel® SGX\) SDK for Linux\\* OS Developer Reference](#).

In this task, you will see how to locally (intra-platform) attest an enclave and thus how to exchange encrypted data between local enclaves running on the same Intel SGX-enabled platform thanks to this mechanism.

This first attestation mechanism creates an authenticated assertion between two enclaves running on the same platform.

You will use the **CC-HOL-LTEST-01** lab VM with the Open Enclave SDK to do so.

Start by reading the complete scenario as depicted in section **Scenario 1: Exchanging encrypted data between local enclaves** of the [Leveraging Attestations with Trusted Execution Environment \(TEE\) based applications on Azure - A starter guide for developers](#).

### Task 2: Remotely (inter-platform) attest an enclave

See:

- Section **Remote (Inter-Platform) Attestation** in the [Intel® SGX Developer Guide for Linux](#).
- Section **Remote Attestation** in the [Intel® Software Guard Extensions \(Intel® SGX\) SDK for Linux\\* OS Developer Reference](#).

In this task, you will now consider how to exchange encrypted data between remote enclaves potentially running on distinct Intel SGX-enabled platform thanks to the remote attestation mechanism. This second mechanism extends local attestation as seen in the previous exercise to provide assertions to 3<sup>rd</sup> parties outside the platform referred to as remote attestation. The remote attestation process leverages a quoting service.

Like the previous scenario, you will continue using the **CC-HOL-LTEST-01** lab VM with the Open Enclave SDK to do so.

Read this time the complete scenario as depicted in section **Scenario 2: Exchanging encrypted data between remote enclaves** of the [Leveraging Attestations with Trusted Execution Environment \(TEE\) based applications on Azure - A starter guide for developers](#).

## Task 3: Test-drive the local vs. remote attestation mechanisms

1. To connect the **CC-HOL-LTEST-01** lab VM, proceed as per **Task 3bis: Connect to a Linux-based testing VM in the lab**.
2. Clone and build the samples' code repo as per eponym subsections in section **Building and running the sample codes for the scenarios** in the above starter guide for developers.
3. Follow all the instructions.

**You should now move to the next exercise.**

# After the hands-on lab

**Duration:** 5 minutes

In this last “exercise”, you will stop the lab VMs that were created in support of the lab.

## Task 1: Stop a VM

In this task, you will need to stop the **CC-HOL-WDEV-01**, **CC-HOL-LTEST-01**, and **CC-HOL-LTEST-02** lab VMs.

1. Under **My Virtual Machines**, select the intended VM row, at the end of line click ..., and then select **Unclaim**.

**You should follow all steps provided *after* attending the hands-on lab.**

**If so, this concludes this hands-on lab.**



# Appendix

## Linux VMs private key

**(REPLACE the below private key with your own value.)**

```
-----BEGIN RSA PRIVATE KEY-----
MIIEpQIBAAKCAQEA2Rybr0lBYI9BjiiWSCThMIjSt8DkRpMYSIfdWLLh/Qmb5vgN
eWZVvr1wM7Do0YWJ/gUeBFZFuRGk9/d1taxLqvJ07MkoI3SjSc69oUbJskL30Iag
Tm4+/MpGMQ9b2189ZlDcQqyeuKwFRlwmeFQS8Hs+bYtjpcdgmyloUtN6bFJEtT0T
qDD0aB0Xgl5qBrD+QR+GQ6rVzNjbt9RPbXXtQtLqQfPrdPr88cR2mpddVYW7L9ey
xUD5vCZG10qp1gV/WhWZ6+OpPIAhJTWudbEUr9FI7AUQDc9VdPoLhqSrJFBt0TY2
CXYMnjV74xI6lXyYn0hnYVcw1oKYZ2v4KHmxxQIDAQABAoIBAQC08Yb/9jwGXFNb
wLl4eCI3NzDpRhMcC9hIkFfigbcpGsZR8PSeXAReE/Zwe051RMWW4odjCvTNSGLXL
MueIsALNifH6cFqQBHIpu+15svC+Yzem570UdFtVMD1AbdgnS0u+80sCivr7bvH
00sr2M5oDS795Qpsd0Hqu2Tv1ppb8JVskpWE49pxFU3IcJjBsQH0tsVXkWDsBLVw
dPyQtWFOuCuVpW+MdQxoctzrvKa5C59R0gWYOw6oIgle2G07IFA2+RQHf56uedlR
Au7+ZSq+jGFFQbe2xGMGFzhrRitJrIyfbxxraIwo4qIy/Xg3P55W5NceJWM1IRJuZ
bnMSjleBAoGBAPMOaZfE2lpvh2pBb61NXHU9yAthHQ8M800E7u0KfNYbxP/559IY
x/erBaRZYXcQzARQoUQIX9qLQj88r+250hHoJcXZnlsD05YEGlNu1obbxyCannpV
own4302dMgo7/2RWSk5Yk+2k/vdNnI9ZWKFx00KXKEhyONTsuzAankr1AoGBA0Ss
fli00p5sgjPvYj+i9Ss6RA7Hl+OWQirU/8YafvSinICy39zZzJ7K41G1Ab2J2lEZ
YvcMB6pL62AKGU0eCRKon0duTWgizliLYcJn4U9jGzYE5HbN7w2XB61bgmGWLZJJ
tODz+Y6fZyX9bixdAyzNqIepGGkSVCfL/aOB3/1hAoGAX9YR+p7K/he1zEbK8wlq
6GuY4ivDmnifwoGsd9jHymCet9Pg/W16A4Tyr4/yp1D/MMBeJgYrTW/VijuynsjD
NP5VToL1Nqu6pfcuLjGo8vnbTVZiIJh9tePkoKTX40Mu+3Au0l+IzI5fXkHC9p7j
HyMwoPe7EX3AP0yvDl0gkKUCgYEAxHtuCDsVJQCJE4TRf2pOjDkBN03KNXPrIJBp
wNcNVLfQD0kizsmZZqtFjNohR7GGE37jq0/+0hYHhTrIKJnxI8YdLawZ+KtHb487
kvuifarj05QSlf42NBAC0ZlS0vVl7LdGIq+fMyvF49VWb+nvi3SeJQpm/gkQpC1D
n1U9l6ECgYEAt/EfW7xDa1F608tWXdWmEuk5WwiTIq0QJW3w8B06P0Bd9j68vKa
UwnkjHwhbEYv+HXA47fJcemJ1+GBPXMCHxI19XiJaIEG4xHfX17RJ7LbT5mzeQuG
hpjJfbygMXZLEYdmdxFeJ+8DFPMcZhwIGfZq6ti7KVYqvJXnofFcba8=
-----END RSA PRIVATE KEY-----
```