

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 setup

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

| | |
|--|-----------|
| ABSTRACT AND LEARNING OBJECTIVES | 5 |
| Overview | 5 |
| Hands-on lab requirements | 5 |
| DEPLOY THE LAB ENVIRONMENT PRE-BUILD | 7 |
| Alternative 1: Deploy the lab with the Azure Portal | 7 |
| Task 1: Provision a resource group | 7 |
| Task 2: Create the Windows-based development VM | 7 |
| Task 3: Generate an SSH key pair with OpenSSH | 8 |
| Task 4: Create two Linux-based testing VMs | 10 |
| Alternative 2: Deploy the lab using ARM templates | 11 |
| PREPARE THE WINDOWS-BASED DEVELOPMENT VM | 13 |
| Task 1: Connect to your Windows-based development VM | 13 |
| Task 2: Install Visual Studio | 14 |
| Task 3: Install the Chocolatey package manager | 14 |
| Task 4: Install Visual Studio Code | 14 |
| Task 5: Install Git and enable long paths | 14 |
| Task 6: Install Docker EE | 15 |
| Task 7: Shrink the main partition (optional) | 16 |
| PREPARE THE LINUX-BASED TESTING VMS (OPTIONAL) | 18 |
| Task 1: Shrinking the main partition size (optional, strongly not recommended) | 18 |
| EXPORT THE VMS' DISKS | 20 |
| Task 1: Create a storage account | 20 |
| Task 2: Prepare the Virtual Machines for the export | 20 |
| Preparing a Linux-based Virtual Machine for generalization | 20 |
| Preparing a Windows-based Virtual Machine for generalization | 20 |
| Generalize the Virtual Machines | 21 |
| Task 3: Export the disk images on the storage account | 21 |
| SHRINK THE RESULTING VHD IMAGES | 23 |
| DESCRIBING THE LAB THROUGH JSON | 24 |

Task 1: Creating a JSON file for each machine..... 24

Task 2: Upload the files to the blob containers 25

SHARE THE VHD IMAGES..... 26

Task 1: Get a link to an exported image 26

UNDERSTANDING THE LAB WORKFLOW 27

From model VMs to Storage Account [This guide] 27

From Storage Account to reusable Image Definitions 27

From Image Definitions to user VMs 27

DELETE THE LAB ENVIRONMENT PRE-BUILD..... 29

Task 1: Delete the Resource group in which you placed your Azure resources..... 29

APPENDIX A: PARTIAL AUTOMATION WITH DOCKER 30

APPENDIX B. JSON FILES FOR THE VIRTUAL MACHINES..... 32

CC-HOL-WDEV-01.json..... 32

CC-HOL-LTEST-01.json..... 32

CC-HOL-LTEST-02.json..... 32

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 local machine running local machine running Windows 10 1809 and above.
- A Windows-based development VM configured with (**complete the day before the lab!**):
 - [Visual Studio Community 2017](#) installed as code editor along with the Intel SGX SDK.
 - [Visual Studio Code](#) installed as (yet) another code editor with the [Open Enclave extension](#).
 - 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](#).
- Two Intel SGX-capable Linux-based testing VMs. (The second machine is intended for future use.)

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

| Name of the VM | OS of the VM | Type of the VM |
|-----------------|-------------------------------|----------------------------------|
| CC-HOL-WDEV-01 | Windows (Windows Server 2019) | Standard D2 v3 |
| CC-HOL-LTEST-01 | Linux (Ubuntu 18.04 LTS) | Standard DC2s v2 |
| CC-HOL-LTEST-02 | Linux (Ubuntu 18.04 LTS) (*) | Standard DC2s v2 |

(*) For future use

Deploy the lab environment pre-build

This section will guide you throughout the setup of the environment for use in the hands-on lab. This environment comprises a set of three VMs as stated before. You should follow all steps provided *before* exporting the disk image of these VMs.

IMPORTANT: Most Azure resources require unique names. Throughout this lab you will see the word “**CC-HOL-**” as a PREFIX of all the resource names.

Alternative 1: Deploy the lab with the Azure Portal

Task 1: Provision a resource group

In this task, you will create an Azure resource group for the resources used throughout this lab.

1. In the [Azure Portal](#), select **Resource groups**, select **+Add**, then enter the following in the **Create a resource group** blade:

Basics Tags Review + create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

Project details

Subscription * ⓘ

Resource group * ⓘ

Resource details

Region * ⓘ

- a. **Subscription:** Select the subscription you are using for this hands-on lab.
 - b. **Resource group:** Enter **CC-HOL-RG**
 - a. **Region:** Select a nearby location. Remember this location for other resources in this hands-on lab.
2. Select **Review + create**.
 3. Once the validation passed, click **Create**.
 4. When invited, click **Go to resource group**.


Task 2: Create the Windows-based development VM

In this task, you will provision an Azure virtual machine (VM) on a Windows Server 2019 (x64) image to later install and use the Visual Studio Community 2017 and the Visual Studio Code. This will be used as the Windows-based development machine throughout the lab.

Important note It is recommended you use a Standard DS2 or D2 instance size for this VM. See [Sizes for Windows virtual machines in Azure](#).

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 organization account or just a Microsoft Account.
2. In the [Azure Portal](#), select **Virtual machines**, click + **Add**, and then **Virtual machine**.
3. On the blade **Create a virtual machine** that comes up, set the following configuration on the **Basics** tab.
 - a. **Subscription**: Select the subscription you are using for this hands-on lab
 - b. **Resource Group**: Choose Use existing and select the **CC-HOL-RG** resource group
 - c. **Region**: Select the location you are using for this hands-on-lab. For example, West Europe
 - d. **Virtual machine name**: Enter **CC-HOL-WDEV-01**
 - e. **Availability options**: Leave No infrastructure redundancy required selected
 - f. **Image**: Click **Browse all public and private images**. In the **Select an image** dialog, search for "Windows Server 2019 Datacenter with containers", and then select **Windows Server 2019 Datacenter with Containers – Gen2. Windows Server 2019 Datacenter with Containers** should now be displayed.
 - g. **Azure Spot instance**: Leave No selected
 - h. **Size**: leave **Standard D2 v3** selected
 - i. **Username**: Enter *demouser*
 - j. **Password**: Enter *Password.1!!*
 - k. **Public inbound ports**: Leave Allow selected ports selected
 - l. **Select inbound ports**: Select RDP (3389)
 - m. **Already have a Windows license**: Leave **No** selected
 - n. In the **Networking** tab, ensure that a new Virtual Network is created, this network will be then used by all the other machines in this lab.
4. Select **Review + create** to move to the last step of the configuration.
5. Once validated, Click **Create** provision the virtual machine.
6. It may take 10+ minutes for the virtual machine to complete provisioning.

✓ Your deployment is complete



Deployment name: CreateVm-MicrosoftVisualStudio.VisualStudio-...

Subscription: [Subscription NTO-France Microsoft Azure](#)

Resource group: [RGLabVM](#)

Start time: 4/22/2020, 10:10:38 AM

Correlation ID: 471e0f9a-60ae-4ba8-900a-aaaa25fd8636

▼ Deployment details [\(Download\)](#)

^ Next steps

[Setup auto-shutdown](#) Recommended

[Monitor VM health, performance and network dependencies](#) Recommended

[Run a script inside the virtual machine](#) Recommended

[Go to resource](#)

Create another VM

7. Click **Go to resource**.

Task 3: Generate an SSH key pair with OpenSSH

If you already have an SSH key pair you can skip this task.

In this task, you will start by installing OpenSSH on your local machine – We assume here that your local machine is running Windows 10 1809 and above -. The OpenSSH Client and OpenSSH Server are separately installable components in Windows 10 1809 and above.

Note For information about the OpenSSH availability on Windows 10, see [here](#).

OpenSSH includes different tools and more specifically the `ssh-keygen` command for generating secure SSH key pairs, that can be in turn used for key authentication with SSH. SSH key pairs refer to the public and private key files that are used by certain authentication protocols.

1. On your local machine, open an elevated PowerShell console.
2. Run the following command:

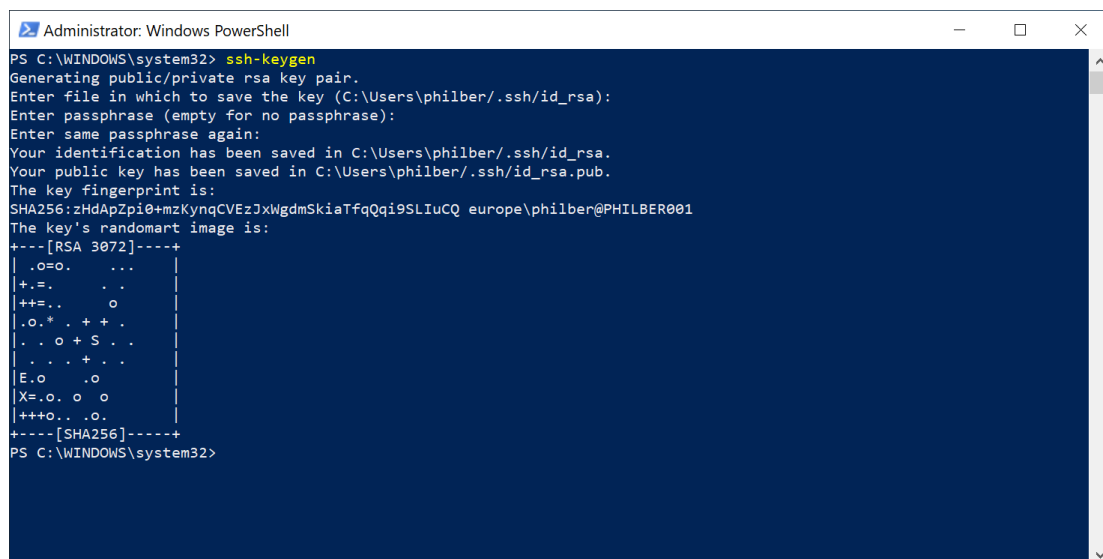
```
PS C:\> Add-WindowsCapability -Online -Name OpenSSH.Client~~~~0.0.1.0
```

Once the installation completes, you can use the OpenSSH client from PowerShell or the Windows 10 command shell.

3. Generate your SSH key pair for the lab:

```
PS C:\> ssh-keygen
```

You can just hit ENTER to generate them, but you can also specify your own filename if you want. At this point, you'll be prompted to use a passphrase to encrypt your private key files. The passphrase works with the key file to provide 2-factor authentication. For this example, we are leaving the passphrase empty.



```
Administrator: Windows PowerShell
PS C:\WINDOWS\system32> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (C:\Users\philber/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in C:\Users\philber/.ssh/id_rsa.
Your public key has been saved in C:\Users\philber/.ssh/id_rsa.pub.
The key fingerprint is:
SHA256:zHdApZpi0+mzKynqCvEzJxWgdmSkiaTfqQqi9SLIuCQ europe\philber@PHILBER001
The key's randomart image is:
+---[RSA 3072]---+
|.o=.   .   |
|+.   .   |
|++=..  o   |
|.o.*. + + . |
|. . o + S . |
|. . + . . |
|E.o  .o   |
|X=.o. o o   |
|++o.. .o.   |
+---[SHA256]-----+
PS C:\WINDOWS\system32>
```

Note SSH public-key authentication uses asymmetric cryptographic algorithms to generate two key files – one "private" and the other "public". The **private key** file is the equivalent of a password and should be protected under all circumstances. If someone acquires your private key, they can log in as you to any SSH server you have access to. The **public key** is what is placed on the SSH server and may be shared without compromising the private key.

When using key authentication with an SSH server, the SSH server and client compare the public key for username provided against the private key. If the public key cannot be validated against the client-side private key, authentication fails.

By default, the files are saved in the following folder `%USERPROFILE%\ssh`:

| File | Description |
|---|------------------------------|
| <code>%USERPROFILE%\ssh\id_rsa</code> | Contains the RSA private key |
| <code>%USERPROFILE%\ssh\id_rsa.pub</code> | Contains the RSA public key. |

Task 4: Create two Linux-based testing VMs

In this task, you will provision two Azure Confidential Computing VM using the newly introduced [DCv2-series](#) family of VM. These two VMS, namely the **CC-HOL-LTEST-01** and **CC-HOL-LTEST-02** lab VMs, will be used as the Linux-based testing machines throughout the lab.

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 organization account or just a Microsoft Account.
2. In the [Azure Portal](#), select **+Create a resource**, type "Confidential Comput" into the Search the Marketplace box, and select **Azure Confidential Compute (Virtual Machine)** from the results.
3. On the blade that comes up, click **Create**.
4. Set the following configuration on the **Basics** tab.
 - a. **Subscription:** Select the subscription you are using for this hands-on lab
 - b. **Resource Group:** Choose Use existing and select the **CC-HOL-RG** resource group
 - c. **Region:** Select the location where this series of VM is available. For example, UK South
 - d. **Image:** Select **Ubuntu Server 18.04 (Gen 2)**
 - e. **Virtual machine name:** Enter **CC-HOL-LTEST-01**
 - f. **Username:** Enter **azureadmin**
 - g. **Authentication type:** Select **SSH public key** for stronger authentication
 - h. **SSH public key:** Specify your SSH public key.
 - i. On a Windows command prompt, run the following command to retrieve the content of your SSH public key and copy it to the clipboard.

```
C:\> type %USERPROFILE%\ssh\id_rsa.pub | clip
```

```

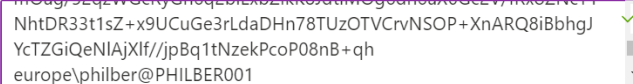
Command Prompt
Microsoft Windows [Version 10.0.19592.1001]
(c) 2020 Microsoft Corporation. All rights reserved.

C:\Users\philber>type %USERPROFILE%\ssh\id_rsa.pub | clip

C:\Users\philber>
  
```

- ii. Paste the content in this field.

SSH public key * ⓘ



```
m0ag/2E4qW0Cny0noq6B1K0Zmk0Zdm0ge0no0K0CEV/m00ZNCY1
NhtDR33t1sZ+x9UCuGe3rLdaDHn78TUzOTVCrvNSOP+XnARQ8iBbhgJ
YcTZGiQeNIajXlf//jpBq1tNzekPcoP08nB+qh
europe\philber@PHILBER001
```

5. Click on **Next: Virtual Machine Settings >** to continue.
6. Set the following configuration on the **Virtual Machine Settings** tab.
 - a. **Virtual Machine Size:** select **DC2s_v2**
 - b. **OS Disk Type:** Leave **Premium SSD** selected
 - c. **Virtual network:** Choose the network created in Task 1.
 - d. **Subnet:** Default subnet
 - e. **Select public inbound port:** Select **SSH (Linux)/RDP (Windows)** as you're going to use SSH
 - f. **Boot diagnostic:** Leave **Disabled** selected
7. Select **Review + create** to move to the last step of the configuration. Validation of your configuration settings will occur on the **Summary** tab.
8. If validation has passed, review your configuration.
9. Click **Create** provision the virtual machine.
10. It may take 10+ minutes for the VM to complete provisioning.

Repeat the above step to create the second Linux-based testing VM, i.e. the **CC-HOL-LTEST-02** lab VM.

You are now ready to further prepare the above created VMs.

Alternative 2: Deploy the lab using ARM templates

This method is intended for more advanced user, already familiar with the [Alternative 1](#) and wishing to automate the lab deployment.

Azure Resource Management (ARM) uses the practice of infrastructure as code to provide a reliable and unified way to deploy any infrastructure on demand.

Throughout every step in Alternative 1, once the machines are properly configured, an ARM template can be generated, along with all the parameters corresponding to the all configuration that has been done so far.

As an example, to download the ARM template corresponding to the above [Task 4](#), follow the instructions until **Step 8**. Next, click on **Download a template for automation**.

The **Template** tab contain the machine general description. This file contains all the necessary components needed to deploy *any* instance of an Ubuntu Server 18.04 Server image on a DCsv2 virtual machines. This is reusable for any other deployment involving this type of machine.

The **Parameters** tab contains all the details of a specific deployment, such as the Linux username and password (SSH key) in this case.

To deploy the two Linux machine needed for Task 4, the same template can be used, as both machines are Ubuntu Server 18.04 deployed on DCsv2 virtual machines. All differences between the two machines can be expressed in two specific parameters files.

Once these two files have been saved to the disk, this ARM template can be deployed using the [Azure CLI](#) using a single command.

```
az deployment group create `
  --resource-group $resource-group-name `
  --template-file "$template-file-path" `
  --parameters "$parameters-file-path "
```

Deploying the entire hands-on lab with its resource group should now be possible with only 4 commands, one command to create the resource group, and one command to create each machine using the arm template.

```
# Create the resource group
az group create `
  --name $resource-group-name `
  --location "$resource-group-location" `
# Create the Windows machine (Task 1)
az deployment group create `
  --resource-group $resource-group-name `
  --template-file "$windows-template-file-path" `
  --parameters "$windows-parameters-file-path "
# Create the first Linux machine
az deployment group create `
  --resource-group $resource-group-name `
  --template-file "$linux-template-file-path" `
  --parameters "$linux-machine-1-parameters-file-path "
# Create the second Linux machine
az deployment group create `
  --resource-group $resource-group-name `
  --template-file "$linux-template-file-path" `
  --parameters "$linux-machine-2-parameters-file-path "
```

As stated before, this method requires a good understanding of the deployment architecture. Thus, it is highly recommended to use the first alternative at least once before attempting this one.

A more advanced example, using this method to deploy the lab semi-automatically, is presented in **Appendix A: Partial automation with Docker** above.

Prepare the Windows-based development VM

You will now prepare the Windows-based development VM, i.e. the **CC-HOL-WDEV-01** lab VM.

Task 1: Connect to your Windows-based development VM

In this task, you will open an RDP connection to your **CC-HOL-WDEV-01** lab VM and disable Internet Explorer Enhanced Security Configuration.

1. From the left-hand menu in the Azure portal, select Resource groups, then enter your resource group name into the filter box, and select it from the list.
2. Next, select your lab VM from the list.
3. On your Lab VM blade, click **Connect** from the top menu, and then select **RDP**.

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](#)

4. Select **Download RDP file**, then open the downloaded RDP file.
5. Click **Connect** on the Remote Desktop Connection dialog.
6. Enter the following credentials when prompted.
 - a. User name: CC-HOL-WDEV-01\demouser
 - b. Password: Password.1!!
7. Click **Yes** to connect, if prompted that the identity of the remote computer cannot be verified.
8. Once logged in, launch the Server Manager. This should start automatically, but you can access it via the Start menu if it does not start.
9. Select Local Server, then select **Off** next to IE Enhanced Security Configuration.
10. In the Internet Explorer Enhanced Security Configuration dialog, select Off under Administrators and Users.
11. Select **OK**.
12. Close the Server Manager.

Task 2: Install Visual Studio

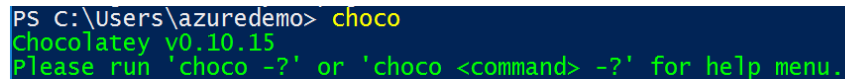
In this task, you will install Visual Studio Community 2017 on your Lab VM.

1. On your **CC-HOL-WDEV-01** lab VM, use a browser to navigate to: <https://docs.microsoft.com/fr-fr/visualstudio/productinfo/vs2017-system-requirements-vs>
2. Download **Visual Studio Community 2017** (2017 is the required version, 2019 won't work with the SGX SDK)
3. Follow the installation process, making sure all **Windows 10 SDK** are selected. Checking the Windows C++ development package is also recommended.
4. Wait for the installation process to complete.

Task 3: Install the Chocolatey package manager

In this task, you will add the Chocolatey package manager on your **CC-HOL-WDEV-01** lab VM for convenience purpose during the lab.

1. Navigate to <https://chocolatey.org/install>
2. Follow the installation method, opening a PowerShell prompt with **administrative privileges**
3. Execute **choco** to verify the installation. Output should look like the image below



```
PS C:\Users\azuredemo> choco
chocolatey v0.10.15
Please run 'choco -?' or 'choco <command> -?' for help menu.
```

Task 4: Install Visual Studio Code

In this task, you will install **Visual Studio Code** to your system using the Chocolatey package manager.

1. Open a PowerShell console with elevated privileges.
2. Run the following command:

```
choco install vscode -y
```

Visual Studio Code is now installed!

Task 5: Install Git and enable long paths

In this task, you will install **Visual Studio Code** to your system using the Chocolatey package manager.

1. Open a PowerShell console.
2. Run the following command:

```
choco install git -y
```

3. Enable long git paths (required):

```
git config --global core.longpaths true
```

Task 6: Install Docker EE

In this task, you will install Docker onto Windows Server 2019 1809. To run both Linux container on Windows Server, you will need nested virtualization, that is enabled by default on Ds2_v3 Azure virtual machines. The Linux containers feature has then just to be enabled.

1. Open a PowerShell console
2. Run the following command:

```
Install-WindowsFeature -Name Hyper-V -IncludeManagementTools -Restart
```

3. Restart the VM and reopen a PowerShell console
4. Run the following command:

```
Install-Module "DockerMsftProvider" -Force
```

5. Restart the VM and reopen a PowerShell console
6. Run the following commands:

```
Update-Module "DockerMsftProvider"  
Install-Package Docker -ProviderName "DockerMsftProvider" -Update -Force  
Install-WindowsFeature Containers
```

7. Restart the VM and reopen a PowerShell console
8. Run the following command:

```
Set-Content -Value "`{"experimental`":true`}" -Path C:\ProgramData\docker\config\daemon.json
```

9. Restart Docker service:

```
Restart-Service docker
```

10. Create a directory for Linux containers:

```
mkdir "C:\Program Files\Linux Containers"
```

11. Navigate into this directory:

```
cd "C:\Program Files\Linux Containers"
```

12. Get LCOW archive:

```
curl -OutFile release.zip https://github.com/linuxkit/lcow/releases/download/v4.14.35-v0.3.9/release.zip
```

13. Extract LCOW:

```
Expand-Archive -DestinationPath . .\release.zip
```

14. Clean up the archive:

```
rm release.zip
```

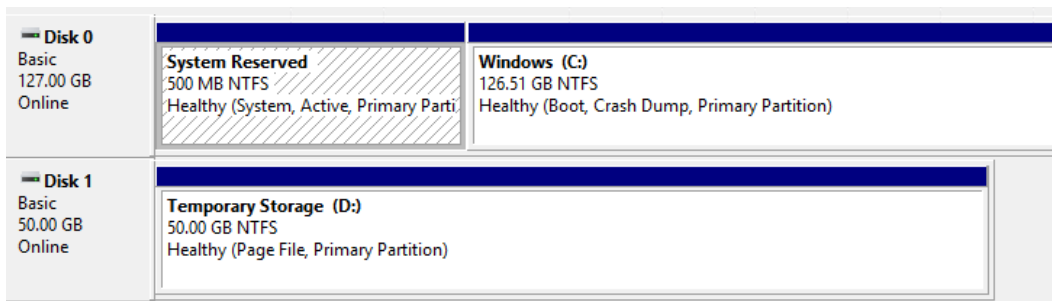
15. Test a Linux container:

```
docker run -it ubuntu /bin/bash
```

Task 7: Shrink the main partition (optional)

To reduce the final .vhd image size, the main Windows partition must be reduced accordingly. This step isn't mandatory and only allows the resulting disk snapshot to be smaller.

1. Press the Windows + R, this should open a "Run command" prompt
2. Type *diskmgmt.msc* into the prompt and press Enter



3. Two disks should be available. Only Disk 0 will be exported into the .vhd file. Right click on Windows (C:) and select **Shrink Volume**.
4. On the next screen, select the amount of space to shrink. Remember to let at least a small amount of remaining space to use during the lab.
5. Click on **Shrink**.

Having followed these directions, you will then be able to reduce the size of the VHD file down to the same size at the main partition later in section **Shrink the resulting VHD images** below.

Prepare the Linux-based testing VMs (optional)

Task 1: Shrinking the main partition size (optional, strongly not recommended)

CAUTION: Shrinking a live EXT4 partition is a difficult and dangerous process. These manipulations may result in the VM being unusable. Thus, following these steps is strongly not recommended.

The default OS disk size for an Ubuntu-Server image is 30GB. However, the actual Disk usage sits around 2GB. By reducing the Ubuntu main partition size, it should be possible to reduce amount of space required to store the Disk image.

However, reducing the size of a live partition is not a well-supported process, and would typically be done by using a live CD containing a partitioning utility like Gparted. This can't obviously be done with premade VMs.

The workaround used consists of pivoting the root onto a ramdisk while the modification to the root partition are done. Once again, this process is very likely to make the VM unusable.

The first step is to stop all running services allowed to write onto the root partition (**sda1**).

```
sudo service <service_name> stop
```

By default, on the chosen VM, no such services should be running. However, your mileage may vary.

Next, unmount all unused filesystems, and check if the root partition has been unmounted.

```
sudo umount -a  
sudo mount # check if /dev/sda1 has been unmounted
```

If the root partition is still mounted, it means that a running process is holding onto it. This can be asserted using this command

```
sudo fuser -vm /dev/sda1
```

Stop any service still using the partition before proceeding. Next, the root will be *pivoted* onto a temporary file system.

```
mkdir /tmp/tmproot  
mount -t tmpfs none /tmp/tmproot
```

```
mkdir /tmp/tmproot/{proc,sys,dev,run,usr,var,tmp,oldroot}
cp -ax /{bin,etc,mnt,sbin,lib } /tmp/tmproot/
cp -ax /usr/{bin,sbin,lib } /tmp/tmproot/usr/
cp -ax /var/{account,empty,lib,local,lock,nis,opt,preserve,run,spool,tmp,yp} /tmp/tmproot/var/
mount --make-rprivate /
pivot_root /tmp/tmproot /tmp/tmproot/oldroot
```

Ensure you can still connect via SSH in another terminal. This most likely won't be the case. You will then need to copy the `authorized_hosts` file in the old root `.ssh` directory into the new temp root. Once you can connect again via SSH, kill every process still user the old root using

```
sudo fuser -vm /oldroot
```

Finally, close the first terminal to release the final resource holding onto the old root. In the second terminal, you should be able to unmount the old root.

```
sudo umount /oldroot
```

Now that the old root is unmounted, it can be resized at will. For example:

```
resize2fs /dev/sda1 3G
```

Let's now revert all the changes back, pivot the root back and remove the temporary root created.

```
mount /dev/sda1 /oldroot
mount --make-rprivate /
pivot_root /oldroot /oldroot/tmp/tmproot
for i in dev proc sys run; do mount --move /tmp/tmproot/$i /$i; done
umount /tmp/tmproot
rmdir /tmp/tmproot
mount -a
mount --make-rshared /
systemctl isolate default.target
```

Log out and back in again, the resized root should be up and running.

Having followed these directions, you will then be able to reduce the size of the VHD file down to the same size at the main partition later in section **Shrink the resulting VHD images** below.

Export the VMs' disks

You will export the disk of the VMs you created and prepared before, i.e. the **CC-HOL-WDEV-01**, **CC-HOL-LTEST-01**, and **CC-HOL-LTEST-02** lab VMs. See **Appendix A: Partial automation with Docker** below for semi-automated example using Docker.

Task 1: Create a storage account

Using the Azure CLI, you can create a new storage account easily. Within this storage account, a storage container (a storage account folder) is then created to store the VMs disk images.

```
$storageAccountName="<your-storage-account-name>"
$storageAccountRG="<your-storage-account-resource-group>"
$storageContainerName="<your-storage-container-name>"
#Create the storage account
az storage account create --name $storageAccountName --resource-group $storageAccountRG
#Retrieves its credentials
$storageCredentials = az storage account keys list -n $storageAccountName --resource-group
$storageAccountRG | ConvertFrom-Json
#Creates a container within the storage account
az storage container create --account-name $storageAccountName --account-key
$storageCredentials[0].value --name $storageContainerName
```

To create an azure storage account using the Azure Portal, please refer to section **Prepare your environment for the hands-on labs** of the document entitled **Available hands-on labs: Hands-on lab deployment instructions**.

Task 2: Prepare the Virtual Machines for the export

In this task, you will separate the managed disk from the created VMs to download an image of it. First, you must stop the Azure agent in both Linux and Windows machines.

Preparing a Linux-based Virtual Machine for generalization

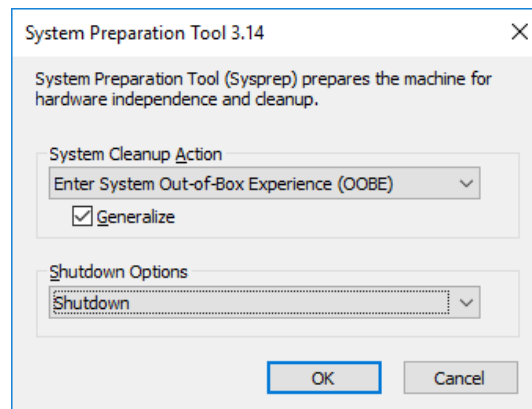
The Az Agent must be stopped to properly generalize the VM. Be careful, this process will render the VM:

```
# Connect to the Linux machines and execute this command
sudo waagent -deprovision+user
# For Windows VMs : Follow these instructions
# https://docs.microsoft.com/fr-fr/azure/virtual-machines/windows/capture-image-resource
```

Preparing a Windows-based Virtual Machine for generalization

1. Press **Windows + R**
2. Type "%windir%\system32\sysprep\sysprep.exe"

3. In the following GUI, make sure the "**Generalize**" box is ticked and select Shutdown in "**Shutdown Options**".



4. Click **OK**.

Generalize the Virtual Machines

Both Windows and Linux machines require to be deallocated and generalized before copying their virtual disk,

```
az vm deallocate --resource-group HE-HOL-RG --name <machineName>
az vm generalize --resource-group HE-HOL-RG --name <machineName>
```

Task 3: Export the disk images on the storage account

This step purpose will be to copy over the VMs disk images to the storage account. This step requires both [Az-Copy 10](#) and [Azure PowerShell](#) to be installed.

Then change the variable to fit your needs and execute the script below. To create a storage account, a container and get the storage access key, please refer to section **Prepare your environment for the hands-on labs** of the document entitled **Available hands-on labs: Hands-on lab deployment instructions**.

In our illustration, the account storage name is **compasswshols** whereas the blob container name is **hol-vhds**.

The Url is as follows: <https://compasswshols.blob.core.windows.net/hol-vhds>

```
#Provide the subscription Id of the subscription where managed disk is created
$subscriptionId = "<your-subscription-id>"

#Provide the name of your resource group where managed is created
$resourceGroupName = "CC-HOL-RG"

#Machines within the resource group you want to save an image of, the three created machines here
$machineNames = @("CC-HOL-LTEST-01", "CC-HOL-LTEST-02", "CC-HOL-WDEV-01" )

#Provide storage account name where you want to copy the underlying VHD of the managed disk.
$storageAccountName = "<your-storage-account-name>"
```

```

#Name of the storage container where the downloaded VHD will be stored
$storageContainerName = "<your-storage-container-name>"

#Provide the key of the storage account where you want to copy the VHD of the managed disk.
$storageAccountKey = '<your-storage-container-key>'

#Provide Shared Access Signature (SAS) expiry duration in seconds e.g. 3600.
#An Azure to Azure transfer should take way less than 1 hour, only consider changing this value
#on Azure to premise tranfer
$sasExpiryDuration = "3600"

function getDiskName($machineName){
    return az disk list `
        --resource-group $resourceGroupName `
        -o tsv `
        --query " [?contains(name,'$machineName')].{Name:name} "
}

function copyDiskImage($machineName){

    $diskName = getDiskName $machineName
    # Set the context to the subscription Id where managed disk is created
    Select-AzSubscription -SubscriptionId $SubscriptionId

    #Generate the SAS for the managed disk
    $sas = Grant-AzDiskAccess -ResourceGroupName $ResourceGroupName -DiskName $diskName -
DurationInSeconds $sasExpiryDuration -Access Read




    #Create the context of the storage account where the underlying VHD of the managed disk will be
    copied
    $destinationContext = New-AzStorageContext -StorageAccountName $storageAccountName -
StorageAccountKey $storageAccountKey

    $containerSASURI = New-AzStorageContainerSASToken -Context $destinationContext -ExpiryTime(get-
date).AddSeconds($sasExpiryDuration) -FullUri -Name $storageContainerName -Permission rw
    $containername, $sastokenkey = $containerSASURI -split "\?"
    $containerSASURI = "$containername/$machineName`?$sastokenkey"
    azcopy copy $sas.AccessSAS $containerSASURI --s2s-preserve-access-tier=false
}

foreach ($machineName in $machineNames) {
    copyDiskImage $machineName
}

```

The extracted image should now be visible in the storage account.

| NAME | ACCESS TIER | ACCESS TIER LAST MODIFIED | LAST MODIFIED | BLOB TYPE | CONTENT TYPE | SIZE | STATUS | RI |
|---|-------------|---------------------------|-----------------------|-----------|--------------------------|----------|--------|----|
|  CC-HOL-LTEST-01 | | | 30/04/2020 à 19:06:25 | Page Blob | application/octet-stream | 30.0 GB | Active | |
|  CC-HOL-LTEST-02 | | | 30/04/2020 à 19:09:47 | Page Blob | application/octet-stream | 30.0 GB | Active | |
|  CC-HOL-WDEV-01 | | | 30/04/2020 à 19:13:15 | Page Blob | application/octet-stream | 127.0 GB | Active | |

Shrink the resulting VHD images

Important note This step requires to have completed section **Task 7: Shrink the main partition (optional)** above for the Windows-based development VM, conversely section **Prepare the Linux-based testing VMs (optional)** above for the Linux-based testing VMs.

By default, a VHD file has the same size as the whole disk. However, unpartitioned space can be omitted. Thus, by reducing the main partition size in the earlier steps, the size of the VHDs images can be reduced.

1. Download the [disk resizer utility](#) (Windows only) and extract it
2. Go to the storage account page on the Azure portal. For example, **compasswshols** in our illustration.
3. Click on the blob container containing the VHDs images. For example, **hol-vhds** in our illustration:
<https://compasswshols.blob.core.windows.net/hol-vhds>
4. Right click on the VHD image to be resized
5. In the context menu, select **Get Shared Access Signature** (SAS)
6. In the **Permissions** Box, tick Read, List, Write and Create
7. Click on **Create** to generate the SAS
8. In a command prompt, execute the **WindowsAzureDiskResizer** utility with 2 arguments. The first being the new size of the VHD in Gigabytes (must be an integer), the second being the generated SAS.

```
.\WindowsAzureDiskResizer.exe <new_size_in_GB> "<windows_SAS>"
```

Describing the lab through JSON

Task 1: Creating a JSON file for each machine

Each disk image must be accompanied by a JSON description file. This description file will be used to rebuild the machine from the disk.

```
{
  "timestamp": "2020-05-13T18:27:35",
  "description": "CC - Lab",
  "publisher": "Microsoft NTO",
  "osType": "Windows",
  "imageName": "CC-HOL-WDEV-01",
  "vhdFileName": "CC-HOL-WDEV-01",
  "size": "Standard_D2_v3",
  "storageType": "Standard",
  "OsState": "Generalized",
  "HyperVGeneration": "V2",
  "dnsServer": ""
}
```

Each attribute is a straightforward way to represent an aspect of the machine:

Timestamp: JSON creation timestamp

Description: Image template description

[**Publisher:** Image template owner, Defaults to "Custom"]

osType: Either "Windows" or "Linux". The OS is stored on the disk itself, and this attribute is used to open either SSH or RDP.

imageName: Name of the machine to create

vhdFileName: VHD file to used to build the machine

size: VM size to deploy the Disk image on. All available sizes are available of the [Microsoft Website](#).

storageType: SSD type to use. Either "Premium" or "Standard". Comparison is available on the [Microsoft website](#).

[**OsState:** Either "Generalized" or "Specialized". A Generalized image is meant to be use as a model to create other VMs, whereas a Specialized image can be regarded as a snapshot of a specific VM. More details can be found [here](#). All VMs used in this lab are Generalized. Defaults to "Specialized"]

[**HyperVGeneration:** Either "V1" or "V2". VM generation. Should be set to "V2" only when using a VM **size** included in [the generation 2 VMs list](#). Defaults to "V1"]

dnsServer: Machine to use as a DNS server. This is an optional parameter; the value can be set to an empty string to use the default configuration. In the default configuration, every machine has access to the internet, and can query each other via ICMP.

The **Appendix B. JSON files for the virtual machines** provides sample JSON file for the CC-HOL-WDEV-01, CC-HOL-LTEST-01, and CC-HOL-LTEST-02 lab environment VMs.

Task 2: Upload the files to the blob containers

The final step is to upload the JSON files on the same storage container than the VHD files. These files will be later picked up by the lab building script.

1. In the Azure portal, go to the storage account (**compasshols** in the example) page.
2. On the left panel, select **Storage Explorer (preview)**
3. On the middle panel, click on **BLOB CONTAINERS** and select your container (**vhds-2020-04** in the example)
4. Click on **Upload** and select all the json files to upload

Share the VHD images

In this the, you will be learning how to find and make a link to the previously exported images.

Task 1: Get a link to an exported image

1. On your Storage account, click on **Storage Explorer (preview)**
2. Under **BLOB CONTAINERS** in the middle panel, click on your created container
3. You should now be able to see all previously exported images
4. Click on the image you want to export and then click on **Copy URL**
5. Share the copied link with whom it may concern

Understanding the lab workflow

A Dev & Test Lab is a powerful learning structure, allowing anyone to leverage a reusable Cloud playground for any technology. At the end of this guide, VHDs images and JSON descriptions are located on a Storage Account. From this point on, every step is automated. This part will detail whole creation process.

From model VMs to Storage Account [This guide]

Throughout this guide, you have created VMs from pre-existing templates already having an operating system and tailored them to your needs. Azure VMs have a main managed disk, called an **OS_Disk**, and a temporary storage disk. Parts "[Prepare the Windows-based development VM](#)" and "[Prepare the Linux-based testing VM](#)" are dedicated to edit the content of the Os_Disk to suit the lab goal.

Next, in part "[Export the VMs' Disks](#)" This Disk is then dissociated from the VMs and expose as its own entity, before being saved as a VHD file. During this process, VM is generalized, meaning that the Operating system is prepared to be used as a template to create other VMs. User specific information are changed to system-wide modification. Thus, any new user will have the same specific environment as the one created during this lab.

This VHD file however, doesn't contains any information on the VM runtime environment itself, as it's a disk image. In part "[Describing the lab through JSON](#)", a JSON file containing these specific runtime information is created, and the two file are stored together.

From Storage Account to reusable Image Definitions

Once both the VHD file and the JSON file are located on a storage account, the lab creation process continues with an automated process creating an [Azure Stored Image Gallery](#). This gallery is what is used internally to store image to be used in the [Azure Marketplace](#), it's an highly available VM template registry that combines both the VHD file JSON file information.

Within this gallery, multiple Image Definitions are contained, one for each VM. These Image definitions contains Image Versions. Similar to a version-control software, such as git, these versions are auto-labeled using [semantic versioning](#). Specific evolution is the image can thus be easily monitored, and breaking changes won't disturb any production setup. A new Image Version is created by making a snapshot of the current VHD file and freezing this snapshot into a reusable component.

This entire process is fully automated and available with the "[Import-VHDsToSharedImageGallery](#)" script is the [Lab script repository, using the branch "HyperVGen2"](#).

From Image Definitions to user VMs

The last step is simply using these Images Versions to create brand new VMs, this process is also fully automated using the "[Create-Vms](#)" [script](#) int the lab repository.

As these images are generalized, they need to be created with a new user account. That's precisely the point of the `credentials.csv` file accompanying the Lab script repository.

| image Name | Username | Credential Type | Credential Value |
|-----------------|------------|-----------------|-----------------------------|
| CC-HOL-LTEST-01 | azureadmin | SSHKey | ssh-rsa AAAAB3NzaC1yc2EAAA/ |
| CC-HOL-LTEST-02 | azureadmin | SSHKey | ssh-rsa AAAAB3NzaC1yc2EAAA/ |
| CC-HOL-WDEV-01 | CamfYEqD | Password | QHvBcReAw1u0dXWv8kg |
| HE-HOL-LDEV-01 | NvVKrhfj | Password | pB2CZybXRGJSmncsL7 |

Using [Create-Vms.ps1](#), the VMs are created and exposed to be connected to. The lab can be either set to "Private", "Public", "Shared".

| Dev Test Lab Name | Resource Group Name | Ip Config | Shared Image Gallery Name | Shut Down Time | Timezone Id | Lab Region | Lab Owners | Lab Users |
|-------------------|---------------------|-----------|---------------------------|----------------|---------------------------|------------|------------|-----------|
| testLabOfDespair | COMPASS-HOL-RG | public | compassholsig | 1900 | "W. Europe Standard Time" | westeurope | | |

A "Public" lab will assign a different IP address to all VM, a "Shared" lab will use one IP address, shared between all VMs. On these labs, the destination port of any incoming connection will be used to differentiate VMs. A "Private" lab is not exposing any VMs to the public. It can be regarded as "owners only".

Delete the lab environment pre-build

Lastly, you will deprovision any Azure resources that were created in support of the lab.

Task 1: Delete the Resource group in which you placed your Azure resources

1. From the Portal, navigate to the blade of your **CC-HOL-RG** resource group and select **Delete** in the command bar at the top.
2. Confirm the deletion by re-typing the resource group name and selecting **Delete**.

You should follow all steps provided *after* preparing the hands-on lab's environment.

If so, this concludes the setup of the hands-on lab.

Appendix A: Partial automation with Docker

The confidential computing lab setup process can be partially automated. Using the ARM templates, a reusable deploy script can be created. Likewise, saving VMs images to a storage account can be automated. A Docker container, usable on any platform, has been prepared to allow any interested user to setup this lab easily. This procedure assumes the user understand the manual method described above, and thus will be more concise

First, pull the lab docker image:

```
docker pull devlabs.azurecr.io/cclab
```

Next, create a `config.ps1` file on the host system. Although this is a PowerShell script, the file won't be executed on the host system. Thus, PowerShell doesn't need to be installed. The file content should be the following

```
# Resource Group name to create
$resGroupName = "CC-HOL-RG"

# Resource group region (non-correlated to physical machine regions)
$resGroupLocation = "westeurope"

# Name of the Azure subscription to use in the deployment
# Use "az account list" to find out
$subName = "<your-subscription-name>"

# SSH key Path to register in the Linux machines, for direct connection
# Please refer to the guide to create one
$sshPubKeyFilePath = "$HOME/.ssh/id_rsa.pub"

# Linux root username
$linuxUser = "azureadmin"

# Windows username and password
$windowsUser = "azuredemo"
$windowsPassword = "Password.1!!"

#Machine within the resource group you want to save an image of.
# If not set, all the VMs in the resource group are saved
$machineNames = @( "CC-HOL-LTEST-01", "CC-HOL-LTEST-02", "CC-HOL-WDEV-01" )

#Storage account resource group name and Location.
# Same resource group and location as the VMs by default.
$storageAccountResourceGroupName = $resGroupName
$storageAccountResourceGroupLocation = $resGroupLocation
#Provide storage account name where you want to copy the underlying VHD of the managed disk.
$storageAccountName = "compasshols"

#Name of the storage container where the downloaded VHD will be stored
$storageContainerName = "vhds-2020-04"

#Provide the key of the storage account where you want to copy the VHD of the managed disk.
$storageAccountKey = '<storage-key>'
```

Adapt these variables to your needs.

This file will then be share with the lab container, along with the directory where your SSH public key is located.

```
docker run -it -v $PWD\config.ps1:/app/config.ps1 -v $HOME\.ssh\::/root/.ssh <image_name>
```

Note Docker volumes needs absolute path, using relative path could lead to unexpected behavior.

Once logged into the container, the entire lab can be deployed with a single command:

```
./deploy.ps1
```

This process can take up to 20 minutes to complete. Once completed, login credentials for the lab VMs will be displayed as such:

```
All Done ! You can now connect to the Windows development Machine with RDP:
(ip : 40.120.57.182, user: azuredemo, password: Password.1!!)
--
Linux machine 1 is available via ssh azureadmin@40.120.58.95
Linux machine 2 is available via ssh azureadmin@40.120.59.201
```

You should be able to connect to any of the machines using SSH for Linux-based VMs and RDP for Windows-based VMs.

Steps **Prepare the Windows-based development VM**, **Prepare the second Linux-based testing VM** and the first two tasks of **Export the VMs' disks** can then be executed as normal, fine-tuning the machines for the lab. Executing these steps is mandatory and failing to complete them will lead to errors stating that the VM disk is in use.

If it doesn't exist yet, you can then create a storage account to receive the VM disk images. Be aware that the name of the storage account must be unique and must be less than 24 characters long.

```
./create-storage-account.ps1
```

Once these steps are finished, the export process can be executed with two commands:

```
Connect-AzAccount
./copy-disks.ps1
```

Your VMs images are now saved in the provided storage account.

The lab setup process is complete!

Appendix B. JSON files for the virtual machines

This section provides JSON files sample for the CC-HOL-WDEV-01, CC-HOL-LTEST-01, and CC-HOL-LTEST-02 lab environment VMs.

CC-HOL-WDEV-01.json

```
{
  "timestamp": "2020-05-13T18:27:35",
  "description": "CC - Lab",
  "publisher": "ArnaudJumelet",
  "osType": "Windows",
  "imageName": "CC-HOL-WDEV-01",
  "vhdFileName": "CC-HOL-WDEV-01",
  "size": "Standard_D2_v3",
  "storageType": "Standard",
  "OsState": "Generalized",
  "HyperVGeneration": "V1",
  "dnsServer": ""
}
```

CC-HOL-LTEST-01.json

```
{
  "timestamp": "2020-05-13T18:27:35",
  "description": "CC - Lab",
  "publisher": "ArnaudJumelet",
  "osType": "Linux",
  "imageName": "CC-HOL-LTEST-01",
  "vhdFileName": "CC-HOL-LTEST-01",
  "size": "Standard_DC1s_v2",
  "storageType": "Standard",
  "OsState": "Generalized",
  "HyperVGeneration": "V2",
  "dnsServer": ""
}
```

CC-HOL-LTEST-02.json

```
{
  "timestamp": "2020-05-13T18:27:35",
  "description": "CC - Lab",
  "publisher": "ArnaudJumelet",
  "osType": "Linux",
  "imageName": "CC-HOL-LTEST-02",
  "vhdFileName": "CC-HOL-LTEST-02",
  "size": "Standard_DC1s_v2",
}
```



```
"storageType": "Standard",  
"OsState": "Generalized",  
"HyperVGeneration": "V2",  
"dnsServer": ""  
}
```