RMA - UCL - ULB - UNamur - HE2B - HELB

Faculty of Science

Master in Cyber Security

Refactoring a cyber range system to support multiple hypervisors

Simon Wattier

Promoters: Master Thesis in Cyber Security
Prof. Thibault Debatty Prof. Jean-Michel Dricot

Academic year 2019 - 2020

"*La vie, c'est comme une bicyclette, il faut avancer pour ne pas perdre l'équilibre.*"

Albert Einstein, 1879 - 1955

"*Informatique : Alliance d'une science inexacte et d'une activité humaine faillible.*"

Luc Fayard, 2002

Acknowledgment

First of all, I would like to thank the people who have, from near and far, contributed to its development. I also particularly want to express my gratitude:

To my promotor, Mr. DEBATTY Thibault, for his follow-up and assistance in the completion of this thesis.

To my co-promotor, Mr. DRICOT Jean-Michel, for the time he has given me in researching subjects.

Finally, I thank the readers, but also my family and friends for their support and encouragement throughout this achievement.

Contents

1 Introduction 1

1.1 Objectives . . . 1

1.2 Structure . . . 1

1.3 Contributions . . . 2

2 State of the art 3

2.1 Virtualization . . . 3

2.1.1 What is virtualization? . . . 3

2.1.2 Hypervisors . . . 4

2.1.3 Virtualization types . . . 6

2.2 Cyber Ranges . . . 9

2.2.1 What is a Cyber Range? . . . 9

2.2.2 Architecture . . . 9

2.2.3 Different suppliers . . . 10

3 Analysis of an existing system 14

3.1 Mode of operation . . . 14

3.2 Composer for the libraries [26] . . . 17

3.3 Command classes . . . 21

3.4 Tests on GitLab . . . 23

3.5 Identified problem . . . 25

4 Proposed solution **28**

4.1 Design patterns 28

4.2 Interfaces 31

4.3 Dependency injection 32

4.3.1 What is DI? 32

4.3.2 PHP example 33

4.3.3 Java example 35

5 Implementation **37**

5.1 Problem definition 38

5.2 Code analysis 38

5.3 Creation of the interface 39

5.4 Environment variables 42

5.5 Dependency injection 44

5.6 Implementation with ESXi 45

5.6.1 Why VMware ESXi? 45

5.6.2 Installation and configuration 46

5.6.3 How to interface with ESXi? 46

5.6.4 Methods list 48

5.6.5 ESXi class 49

5.6.6 ESXiAdapter class 69

5.6.7 PHP Diagram 71

5.7 Problems encountered 71

6 Conclusion **73**

7 Future works **75**

Appendices **77**

A ESXi and VCenter installation and configuration 77

B Requests documentation 99

B.1 Authentication 99

B.1.1 Login 99

B.1.2 Logout 102

B.1.3 Session information 104

B.2 Information 106

B.2.1 Host 106

B.2.2 Datastore 108

B.2.3 Folder 110

B.2.4 Network 112

B.2.5 version 114

B.3 Virtual machine 116

B.3.1 Get the list of VMs 116

B.3.2 Deploy a VM 118

B.3.3 Deploy a VM 124

B.3.4 Delete a VM 127

B.3.5 Search a VM with its name 129

B.3.6 State of the VM 131

B.3.7 CPU 135

B.3.8 Memory 139

B.3.9 Network 143

C Diagrams 151

List of Figures

2.1 Hypervisor type 1 vs type 2 4

2.2 CPU performance per hypervisor 6

2.3 Example of a Vulnerability Lattice in Alpaca 13

3.1 Architecture of the RMA's solution 15

3.2 Test classes of the cyber range project 24

3.3 PHP diagram before any modifications 27

5.1 Request example from VMware documentation in PostMan 47

5.2 Command classes of the cyber range project after modifications 72

C.1 PHP diagram before any modifications (with methods included) 152

List of Tables

3.1 Table of composer operator 18

Chapter 1

Introduction

1.1 Objectives

The goal of this thesis is to detail and explain the operation of a cyber range solution but also to refactor a cyber range system to support multiple hypervisors and to implement it with VMware's ESXi product.
The choice to refactor the code aims to create independence of the cyber range system from the hypervisor on which it runs but also to implement good coding practices.

By refactoring the code, we will create a kind of ”universal plug”. Thanks to this operation, the cyber range will not have to monitor the type of hypervisor and will be able to simply launch its instructions.

A summary of the master thesis can be found in point 1.2.

1.2 Structure

This thesis is composed of several sections. The first, the introduction, is intended to give the objectives, the structure, and the contributions.

The second, the state of the art, is the research phase. This section contains the theoretical points necessary to understand all this dissertation, as well as what has been done for cyber range systems.

The third one, the analysis of an existing system, will evaluate in detail a current cyber range solution. We will cover its operation and its components. In this case, we will analyze the solution proposed by the Royal Military Academy. We identified multiple shortcomings and limitations, that could be fixed using dependency injection and design patterns.

The fourth one, the proposed solution, introduce design patterns, dependency injection and interfaces. We then show how these could beneficially be applied to the Cyber Range of the Royal Military Academy.

The last section, implementation, is the practical part of this paper. This will bring together the problem definition by analyzing the code, the creation of the interface to overcome this problem, the implementation with VMware ESXi and finally the problems encountered during the implementation.

Finally, the appendices section contains the step-by-step installation and configuration of the ESXi and the vCenter. It also contains the documentation of the different requests used to communicate with the vCenter API and their responses in several formats, depending on the HTTP client chosen to send the requests.

1.3 Contributions

The contribution of this thesis is multiple.

First of all, a high-level description of an existing cyber range solution was made in a paper that Professor Debatty and Professor Mees published. From this paper, we will analyze this cyber range solution in much more detail, presenting in detail the functioning and architecture of the cyber range solution. This contribution may be useful for students and researchers who will work on this project in the future.

Through our detailed analysis of this system but also of the code, it allowed us to bring to light several problems in it. This leads to the second contribution, which is to propose various improvements in the way the code and the different components work together. Indeed, we use design patterns, dependency injection and best coding practice.
The implementation of all these improvements aims to make the code more robust but also more maintainable by reducing, for example, the links between the different classes.

Finally, thanks to the implementation of the various improvements in the code, this will allow us to universalize the code through the use of interfaces.
We will also be able to implement the cyber range of the Royal Military School with another hypervisor (VMware ESXi) than the one currently implemented (VirtualBox).

Chapter 2

State of the art

Before discussing the implementation of the refactoring of the cyber range, we must address some theoretical points, which will be detailed herein state of the art.

First of all, we will start with a part explaining virtualization, the different types of existing hypervisors, and finally, the different types of virtualizations.

The last section will define the cyber ranges, their architecture, and the different existing types and suppliers existing at the moment.

2.1 Virtualization

We will discuss the definition of virtualization, the different types of hypervisors and their use, a comparison of the two types that exist, and the different types of virtualization that exist.

2.1.1 What is virtualization?

Virtualization is the virtual creation of something physical such as an operating system, server, application, storage, or network resources. [42]

It allows several operating systems such as Windows, Linux, etc. to run on a single physical server (the virtualization server) as if they were running on separate physical machines. Those operating systems are independent of each other.

In order to be able to run the virtual machines, we need to install a hypervisor on a physical server (see 2.1.2).

2.1.2 Hypervisors

The hypervisor is an operating system allowing virtualization. It creates a virtualization layer that separates the physical hardware from the virtual machine. Due to this virtualization layer, the VMs are not able to tell the difference between the physical and the virtual environment.[8][27]

They are two types of them: type 1 (bare-metal) and type 2 (hosted) hypervisors (figure 2.1).

The classification of these hypervisors is linked to their method of operation.

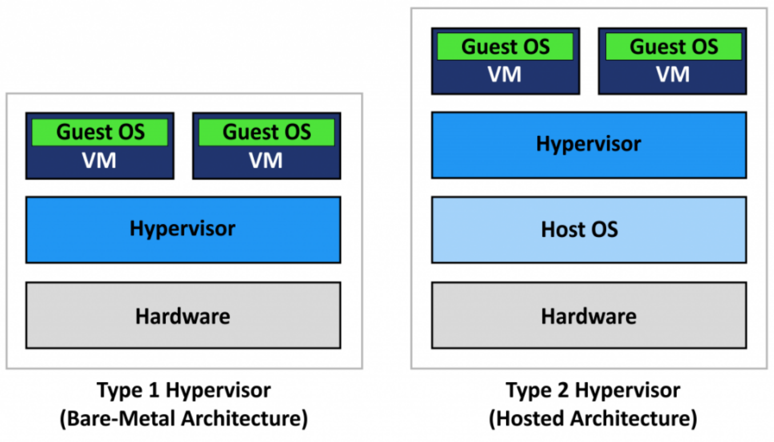


Figure 2.1: Hypervisor type 1 vs type 2

Source: https://www.nakivo.com/blog/hyper-v-virtualbox-one-choose-infrastructure/

2.1.2.1 Bare metal

Type 1 hypervisors are exploitation systems directly installed on the machine, running directly on the system hardware. Since the hypervisor has direct access to the hardware with no software in-between, it provides better performance and stability.

There are a few vendors available:

- VMware vSphere with ESXi

- KVM (Kernel-based Virtual Machine)
- Microsoft Hyper-V
- Citrix Hypervisor (Xen Server)

2.1.2.2 Hosted

Type 2 hypervisors are applications installed on a host operating system (like Windows, MacOS, Linux, ...) that provides virtualization services.

It relies on the host's pre-existing OS to manage calls to CPU, memory, storage, and network resources.

There are a few vendors available:

- Oracle VM VirtualBox
- VMware Workstation Pro / VMware Fusion
- Windows Virtual PC
- Parallels Desktop

2.1.2.3 Type 1 VS Type 2

The type of hypervisor chosen only depends on the needs of the person or the enterprise, as well as the size of the virtual environment. For personal use or smaller deployments, the type 2 hypervisor is the way to go. On the contrary, if the budget allows it and it is a bigger deployment for enterprise-level then the type 1 hypervisor is the ideal choice.

From a performance point of view, type 1 is the most efficient because it has no intermediate layer and works directly on the host's hardware, remaining the more performant compared to type 2.

In the 2016 publication (Performance analysis of selected hypervisors (Virtual Machine Monitors - VMMs) [14]) in which two researchers compared different hypervisors (including Microsoft Hyper-V, VMware ESXi, OVM, VirtualBox and Citrix XenServer) according to several criteria. The major components tested were the processor, RAM memory, hard disk, and network interface. These components were tested separately.
We can see in the following figure (2.2) that the performance at the processor level is more impacted for the type 2 hypervisor (VirtualBox). This is mainly due to the fact that type 1 hypervisors have direct access to system resources.

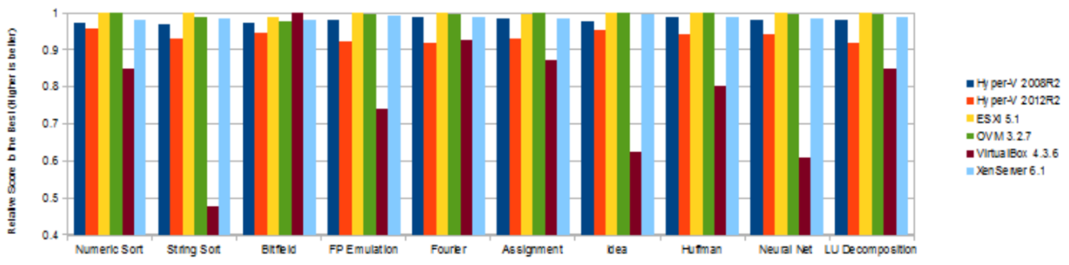


Figure 2.2: CPU performance per hypervisor

Source: [14]

Licensing is also an important factor to consider. Indeed, most type 2 hypervisors are free, while type 1 hypervisors have several types of licenses. It can be per server, per CPU, or even per core used.

2.1.3 Virtualization types

2.1.3.1 Data virtualization

The main goal of data virtualization is to create, from different and disparate sources, a unique and virtual representation of data without having to copy or to move them. That way, data is accessible by front-end solutions (applications, dashboard, portals) and can be manipulated in a unified way, regardless of their location. [15, 31, 37, 6]

2.1.3.2 Desktop virtualization

"Desktop virtualization is the concept of isolating a logical operating system (OS) instance from the client that is used to access it." [32]

Virtualized desktops are often hosted on a remote server, allowing an administrator to easily deploy and configure hundreds of simulated desktops. Virtualization enables virtualized desktops to adapt to changing business or user needs by adding, for example, more CPU core, more RAM, etc. The end-user can access his machine from any client (a computer, a tablet, a smartphone, ...).

There are several "types" of desktop virtualization, depending on whether the instance is running on a local machine or a remote machine:

- Host-based virtual machines: the user connects to a VM hosted in a data center;

- Shared hosted: the user connects to a shared desktop running on a server;
- Host-based physical machines: the OS runs on another device's physical hardware;
- OS image streaming: OS running on local hardware but boot from a remote disk image;
- Client-based virtual machine: a VM runs on a fully functional PC with a hypervisor in place.

Among the main benefits of the desktop virtualization, we can cite the lower total cost of ownership, the enhanced security, the reduction of energy costs, and the centralization of management. On the other hand, it has its drawbacks like the increased downtime in case of network problems and security risks in case of bad network configurations. [15, 45, 38]

Desktop virtualization is not to be confused with operating system virtualization (see 2.1.3.3).

2.1.3.3 Operating system virtualization

Unlike desktop virtualization (see 2.1.3.2), OS virtualization allows us to deploy multiple operating systems in a single machine.

It is *"a server virtualization technology that involves tailoring a standard operating system so that it can run different applications handled by multiple users on a single computer at a time".* [40]

It is a useful way to run Linux and Windows environments side-by-side without interfering with each other. The virtual environment accepts commands from different users using different applications.

2.1.3.4 Server virtualization

"Server virtualization is a virtualization technique that involves partitioning a physical server into a number of small, virtual servers with the help of virtualization software." [41]

Server virtualization is part of the general trend that is virtualization in IT environments such as storage virtualization (see 2.1.3.1), network virtualization (see 2.1.3.5), and operating system virtualization (see 2.1.3.3).

They are three kinds of server virtualization: [44]

- **Full Virtualization**: This type of virtualization uses a hypervisor (2.1.2) that directly communicates with the physical server's disk and CPU. The hypervisor keeps each virtual server independent while managing the physicals resources, which can impact server performance.
- **Para-Virtualization**: Unlike full virtualization, the entire network works together as a cohesive unit. It means that each virtual server is aware of each other, and thus the hypervisor uses less processing power to manage the operating systems.
- **OS-Level Virtualization**: It does not use a hypervisor. The virtualization capacity embedded in the operating system of the physical server executes all the tasks of a hypervisor. In this method, all the virtual servers need to run the same operating system.

Server virtualization [25, 44] can be used to increase the server availability (by speeding up the creation of new servers and deployment of an application, it reduces the time of server rebuilding) and to decrease the investment and equipment.
By integrating virtualized servers, it reduces the number of physical servers needed and thus, reduces the operating and caring costs like hardware buying, data center spaces, equipment cabinet, power consumption, and maintenance costs.
Server virtualization can also be used to eliminate server sprawl (when multiple underutilized servers take up more space and consume more resources than they can be justified by their workload [34]). In general, it uses server resources more efficiently and centralize server management (when a physical server is virtualized, it can be visualized as a software that can be easily manageable). [25, 44]

2.1.3.5 Network virtualization

"Network virtualization is a method of combining the available resources in a network to consolidate multiple physical networks, divide a network into segments or create software networks between virtual machines." [33]

Network virtual allows to completely reproduce a physical network, but without the hardware limitations of it. Moreover, network virtualization comes with significant advantages such as [23] greater flexibility and scalability because it is no longer bound to hardware limitations. It offers more manageability over the virtual network and also the possibility to have segregated networks that are isolated from each other. In addition, it drastically reduces the number of physical components such as switches, routers, servers, cables to create multiple and independent networks [15].

2.2 Cyber Ranges

After talking about virtualization in the previous 2.1 point, we are going to talk more easily about cyber ranges.

In this part, we will define a cyber range, explain the different architectures that exist, and finally, concrete examples of different types/suppliers of cyber range solutions.

2.2.1 What is a Cyber Range?

"A cyber range is a platform for the development, delivery and use of interactive simulation environments." [12]

A cyber range is an interactive platform, allowing the virtual representation of an infrastructure, which can be very simple with a few components up till a complex infrastructure composed of many different "hardware" such as servers, switches, routers, firewalls, etc. and thus improving the realism and the quality of the platform.
It allows the simulation of different scenarios, like the simulation of an attack in real-time, the defense of critical systems (blue teaming), the compromise of systems, pentesting (red teaming).

A cyber range serves multiple purposes and can be used to train people to certain technologies, as a certification platform, as Capture The Flag (CTF) exercises, for security testing, research and education, etc. In other words, Cyber Ranges are platforms on which users can practice, train, and improve their cybersecurity defense skills.

Cyber Ranges can be used by a multitude of different "typical" users, ranging from students to military agencies via researchers, security professionals, and corporates (private and government).

After having developed the different architecture that a cyber range can have 2.2.2, we will go through some major actors in the field of cyber ranges 2.2.3.

2.2.2 Architecture

There are generally three ways that you can build your cyber range's architecture on [17] : the physical, the virtual, and the hybrid architecture.

The **physical architecture** replicates the full physical architecture, including all the network elements and routers, switches, firewalls, servers, etc. One of the main advantages of this solution is that you can not have a more identical topology than you infrastructure.
On the other hand, physical architecture has many drawbacks. Indeed, this solution is the most expensive because we have to buy the material twice in order to obtain the same infrastructures.
In addition to having to buy additional hardware, you have to provide people to maintain the infrastructure, provide sufficient space in the server room, and have enough electrical and cooling power for all the additional hardware.
In addition, when the infrastructure of the cyber range changes or evolves, manpower is needed to reset, modify, and configure the hardware.

The **virtual architecture** replicates the physical architecture, but virtually every network element, firewalls, servers, etc. are simulated by virtual machines. Compared to physical architecture, it is cheaper to set up and maintain than physical architecture. It is also easier, as the scenario evolves, to reset, modify, and configure the different elements. Another advantage is that the virtual architecture can work on almost any type of hardware, no dedicated or specific equipment is needed. On the other hand, virtual architecture also has these weak points.
The simplicity of implementation and management gained with it will be impacted by ”raw” performance. It depends, of course, on the infrastructure and the scenario, but virtual infrastructures will show poorer performances than the physical ones.

The **hybrid architecture** combines the simplicity of virtual infrastructure with the power of physical architecture.
Indeed, the choice is free when it comes to virtualizing such equipment and using such physical equipment. Also, some specific devices are either not virtualizable or have not been virtualized.
Consequently, the hybrid architecture allows the advantages of virtual infrastructure to be combined with the possibility of connecting physical equipment to it.

2.2.3 Different suppliers

We are going to look at some of the big players in the cyber range.

Whether it is a ready-made solution or a custom solution that has been done by hand, we will detail some of them in the following section.

2.2.3.1 HNS plateform [29]

Founded in 2002, DIATEAM is specialized in cybersecurity and the production of highly innovative solutions.
As a pioneer company in cyber range and associated services, DIATEAM has developed a complete range of solutions that are marketed under the brand name HNS PLATFORM (https://www.hns-platform.com/). These solutions already equip many civil and military training centers in France and abroad. DIATEAM's HNS PLATFORM cyber range is a software and hardware simulation and virtualization platform, a **hybrid cyber range** that enables security teams to test, train, develop their expertise and reproduce realistic digital environments through the content of all forms.

They propose a lot of features, including multi-view, multi-user with multi-architecture/multi-topology, memory dumps for forensics, custom network links simulation (latency, packet loss and/or duplication...).

They also developed online learning, training, and experimentation platform based on the HNS platform, accessible directly from the web browser without any other plugin required (thanks to the full integration of HTML5 web client). The main advantage of this online platform is the fact that it does not require any installation or dependency, and that you can access it from wherever you are. [30]

2.2.3.2 Airbus [3]

Airbus has been offering a CyberRange solution since 2017.

It is a **hybrid platform**, which means that in order to meet the constraints of a complex environment, the platform is open to be connected with external equipment that is not virtualized.
CyberRange is also very scalable and can deploy pre-configured scenarios within 15 minutes for the number of users that you need.

Its main purposes are the pre-production tests, the training, the exercises, and the operational qualifications.
The main scenarios that will be found on this cyber range solution are the formation and training of the teams: Blue Team, Cyber Challenges, Ethical Hacking IT and ICS, preparation for crisis management, tests and evaluations of the customer's products security capabilities and the improvement of the cyber defense posture through attack scenarios.

The CyberRange, just like the HNS PLATFORM is available in a mobile box, in a bay, or accessible from a cloud.

2.2.3.3 Palo Alto Networks [5]

Palo Alto Networks launched its global Cyber Range initiative in 2017. By making dedicated cybersecurity training and simulation environments available to customers in Europe, the Middle East, Africa, the United States, and the Asia-Pacific region, the company aims to create and develop effective skills to meet today's and tomorrow's cybersecurity challenges.

It is a **virtual platform**, with a varied environment. On this plateform, each participant defends the company by using new generation firewalls to block attacks.
The exercises are managed by teams on-site, allowing the real-life reproduction of production environments as found in the corporate world.
Of course, the carried out exercises show the results obtained using the applications in the Palo Alto portfolio.

Like other solutions, this cyber range from Palo Alto offers several structures that we will quickly discuss below:

- Red Team: simulates malicious users attacking users via different vectors;
- Green Team: simulates legitimate users (on their computer, phone, etc.) connected to a network infrastructure managed by a Blue Team;
- Yellow Team: simulates users like the green team, but those users install malicious applications or click on phishing links;
- White Team: creates attack scenarios and monitor how the Blue Team defends against the Red Team attack while looking at the Green Team metrics.
- Blue Team: simulates those who manage the security and stability of the network and application infrastructure.

Also, participants in a cyber range event will receive learning credits.

2.2.3.4 Alpaca [11]

Alpaca is an open-source software package[1], with a **virtual infrastructure**, which is oriented for **training students** with various scenarios.

Unlike other cyber range solutions, Alpaca uses a vulnerability database coupled with a custom planning engine to simulate exploits sequences allowing an attacker to achieve a specific objective.

[1]https://github.com/StetsonMathCS/alpaca

Based on these factors, the solution can create a unique cyber range that groups together the vulnerabilities selected in the database.

To meet the needs of users, when a scenario is created, a minimum number of steps are required to be able to solve a challenge or exploit a specific vulnerability.
Depending on the scenario, there may be several ways to solve a challenge. In Alpaca, these different resolution paths are called ”Vulnerability Lattice”. An example of this ”vulnerability lattice” can be found in figure 2.3.

Once the lattice is found, a cyber range is built by automatically generating scripts to instantiate a virtual machine that contains all vulnerabilities that make up the lattice.

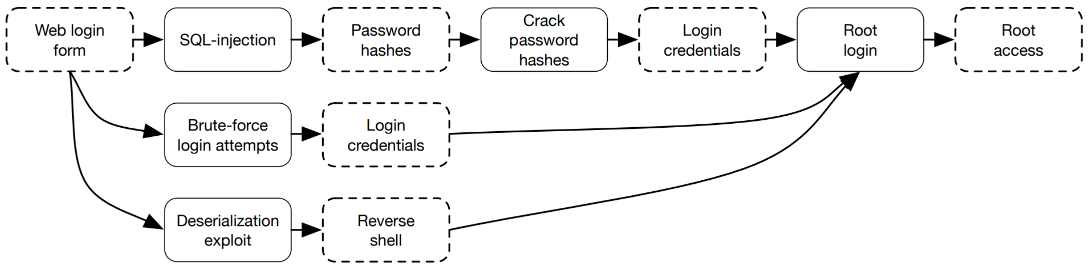


Figure 2.3: Example of a Vulnerability Lattice in Alpaca

Source: [10]

Chapter 3

Analysis of an existing system

In this section, we will look at the analysis of the cyber range solution proposed by the Royal Military Academy. We will begin by describing its mode of operation based on the paper written by Professor Debatty and Professor Mees [10].

We will then go into more detail in the code that makes up this solution. We will then talk about its implementation in practice by going through its composition for library management, the different "command" classes allowing to control the infrastructure and finally the tests that are implemented and performed on GitLab.

This analysis of the cyber range solution will allow a global understanding of the system, but also a more specific understanding of its components.

3.1 Mode of operation

The solution proposed by the RMA is not like the previous ones, i.e. here it has been **made "from scratch"** and is not "key in hand". It is also a fully **virtual infrastructure**, as explained in 2.2.2. Its main use is for **training purposes**.

As you can see on the following figure 3.1, the main components of the RMA cyber range are a hypervisor, a remote desktop gateway, and an orchestrator.

This solution offers many advantages:

- The scenario that will be implemented is written in a text file (yalm or json type) and facilitates version control, scenario updates, etc.;

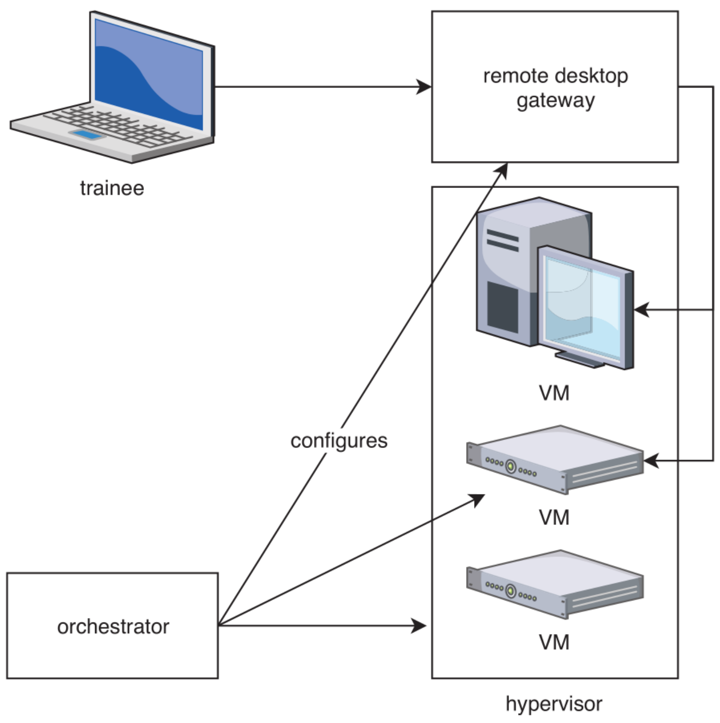


Figure 3.1: Architecture of the RMA's solution

Source: [10]

- The flexibility of the implementation makes it possible to reuse a scenario for 10 or 100 participants;
- Vagrant images can be used directly in the scenarios and imported as virtual machines, giving access to hundreds of pre-configured images;
- ”Scenario” files in JSON format allows more extensive configuration of virtual machines (see listing 3.1). Those JSON scenario files describe the machines that need to be instantiated, with their configurations such as the image of the virtual machine to be used, the number of processors, the amount of RAM, the network interfaces, and possibly commands to be executed for the update and installation of specific software.
  There are also ”playbook” files, in YML format, allowing you to specify information such as IP address assertion, installation of certain software or even certain commands to be executed (see listing 3.2).

Apache Guacamole is currently used for the remote desktop gateway. This allows users to connect to the cyber range and use the virtual machines from a browser without requiring additional plugins or flash to be enabled as Guacamole is a pure HTML5 gateway.

At the heart of the cyber range is the orchestrator. Starting from a defined scenario, it is responsible for provisioning the virtual machines (by deploying the appropriate images, configuring the virtual hardware of each virtual machine, IP addresses, user accounts, installing and configuring the required additional software), configuring the various virtual networks, but also creating accounts that will allow users to access their virtual machines from the remote desktop gateway.

To perform all of the above actions, the PHP code is the basis of everything and is responsible for all of them. We will discuss the different classes and steps that make it up in more detail in section 3.3. We will discuss in a general way how the classes work and what they are composed of, but we will also focus on some of the more important classes.

For the cyber range to work correctly, it requires the use of external libraries. Indeed, some of its components are based on libraries that already exist and are widely used elsewhere. To manage them as efficiently as possible, Composer has been implemented here. We will discuss this in more detail in point 3.2; but in simple words, Composer is a simplified PHP dependency management tool that will install and update them for us.

In addition to these various components, there's a part that we haven't talked about yet. In the cyber range, there is also a whole part that is dedicated to testing. These tests are carried out on the GitLab platform (https://gitlab.cylab.be/cylab/cyber-range). To make it simple, with each code modification, a "push" in technical terms, on GitLab, a series of tests is launched.

First of all, we will have Composer which will install all the libraries the program needs. Then, starting from defined scenario files, the program will be in charge of instantiating the virtual machines, configuring them, etc. Once all the tests have been passed, GitLab will provide us with a percentage of classes and methods that the tests have covered. If, on the contrary, an error occurred during the tests, we will have a console with a stack trace explaining the error(s) in our code. This allows us to verify the proper functioning of our code, all in an automated way.

As for the other points mentioned above, the tests have a dedicated part that will be discussed in more detail in point 3.4.

Currently, VirtualBox is the hypervisor responsible for running virtual machines and the network but other hypervisors could be supported.
Therefore the work of code refactoring to support other hypervisors will be based on this cyber range solution.

```
{
  "name":             "forensics-1819",
  "networks":  [],
  "instances":        2,
  "machines":  [
    { "ova":              "vagrant:cylab/ubuntu-16.04-64-desktop",
      "name":            "forensics",
      "networks": [
        { "mode": "bridged",
          "bridge_interface": "eno50"}
      ],
      "provision": [
        "sudo apt-get -y -qq update",
        "sudo apt-get install -y -qq python2.7 python-minimal"
      ],
      "cpu_count": 4,
      "memory": 4096,
      "playbook": "playbook.yml",
      "remote_desktop": true}
  ],
}
```

Listing 3.1: Example of scenario file

```
---
- hosts: all
  become: true
  tasks:
  - name: install packages
    apt:
      name: ['git', 'qemu', 'gnusim8085', 'python-pil', 'python-distorm3']
  - name: install volatility
    git:
      repo: 'https://github.com/volatilityfoundation/volatility.git'
      dest: /opt/volatility
```

Listing 3.2: Example of playbook file

3.2 Composer for the libraries [26]

Composer is a dependency management tool in PHP (similar to Bundler for Ruby applications). It allows you to declare the libraries needed for your project in a composer.json file that it will manage (install/update) for you.

This is not to be confused with a packet handler. Indeed, composer works with "packages" and libraries, but it only manages them on a "per-project" basis, by installing dependencies in a folder (often called vendor) inside the project.

But then, in which cases are we going to use Composer?

Let's assume, for example, that we have a project that relies on a large number of bookstores and that these same bookstores also depend on other bookstores.
Composer will allow us to declare in a JSON file the libraries we need, it will find out which versions of which packages can and should be installed, and then it will install them.

Finally, all dependencies of a project can be easily updated in only one command making it easier to maintain to the latest versions. For online repositories like Github, Gitlab, etc., Composer will also allow to leave out of the repository the dependent libraries and to have only the application code.

Composer also allows you to ”package” libraries that you would have created yourself and share them very quickly. There is already an online platform with a ton of packages available (packagist.org).

Now, how do we use Composer? The very first thing we need to do is obviously create the *composer.json* file. Then, the keyword for the packages will be *require* which will simply tell Composer which packages our project depends on.

As you can see in the following 3.3 listing, *require* takes into account the name of the package with the vendor as well as the version constraints.

```
{
    "require": {
        "vendor/package": "1.0.*"
    }
}
```

Listing 3.3: Example of composer.json

For **version constraints**, Composer is based on several operators that can be found in the following 3.1 table:

Symbol	Signification	Role	Example
>=	Version Range	Specify ranges of valid version, Valid operators are >, >=, <, <=, !=	”php”: ”>=1.0 <1.1 || >=1.2” includes version 1.0 to 1.1 and versions above 1.2
-	Hyphenated Version Range	Defines a range of versions	”php”: ”1.0 - 2.0” is equivalent to >=1.0.0 <2.1
*	Wildcard Version Range	Extends to all subversions	”symfony/symfony”: ”3.1.*” includes the 3.1.1 It is equivalent to ”>=3.1 <3.2”
~	Tilde Version Range	Extends to the next versions of the same level	”doctrine/omr”: ”~2.5” also concerns the 2.6, but not the 2.4 nor the 3.0 It is equivalent to ”>=2.5 <3.0.0”
^	Caret Version Range	Same as tilde, only if there is backward compatibility	”symfony/symfony”: ”^1.2.3” is equivalent to ”>=1.2.3 <2.0.0”

Table 3.1: Table of composer operator

The following listing, coming from the GetComposer website 3.4, is a summary of version example that is possible to make with different version [1].

```
"require": {
    "vendor/package": "1.3.2", // exactly 1.3.2

    // >, <, >=, <= | specify upper / lower bounds
    "vendor/package": ">=1.3.2", // anything above or equal to 1.3.2
    "vendor/package": "<1.3.2", // anything below 1.3.2

    // * | wildcard
```

[1] https://getcomposer.org/doc/articles/versions.md

```
    "vendor/package": "1.3.*", // >=1.3.0 <1.4.0

    // ~ | allows last digit specified to go up
    "vendor/package": "~1.3.2", // >=1.3.2 <1.4.0
    "vendor/package": "~1.3", // >=1.3.0 <2.0.0

    // ^ | doesn't allow breaking changes (major version fixed - following
        semver)
    "vendor/package": "^1.3.2", // >=1.3.2 <2.0.0
    "vendor/package": "^0.3.2", // >=0.3.2 <0.4.0 // except if major version
        is 0
}
```

Listing 3.4: Composer file summary

When all the necessary dependencies have been defined in the "composer.json" file, we will be able to install them. To do so, we will run the "php composer.phar install" command.

Once Composer has completed the installation and at the first command execution, it will write in a file, "composer.lock", the exact versions of the packages it has installed. This will lock the project to those specific versions, preventing us from automatically getting the latest versions of our dependencies.

A good practice is to upload the "composer.lock" file to the repository, so that everyone working on the project will work with the same versions of dependencies.

To go back to the Royal Military Academy's cyber range solution, you will find below 3.5 the "composer.json" file present in the program.

```
{
    "name": "rucd/vbox",
    "authors": [
        {
            "name": "Thibault Debatty",
            "email": "thibault.debatty@gmail.com"
        }
    ],
    "require": {
        "symfony/process": "^3.4",
        "symfony/console": "^3.4",
        "herrera-io/phar-update": "^2.0",
        "doctrine/orm": "^2.6",
        "php-di/php-di": "^5.4",
        "phpseclib/phpseclib": "~2.0",
        "vlucas/phpdotenv": "^3.1",
        "cylab-be/php-vbox-api": "^0.0.1",
        "psr/log": "^1.1",
        "monolog/monolog": "^1.24",
        "cylab-be/php-vagrant-cloud": "^0.0.5"
    },
    "require-dev": {
        "kherge/box": "^2.7",
        "phpunit/phpunit": "^6.5",
        "squizlabs/php_codesniffer": "^3.4",
        "slevomat/coding-standard": "^5.0"
    },
    "autoload": {
        "psr-4": {
            "Cylab\\Cyrange\\": "src/"
        }
    },
    "autoload-dev": {
        "psr-4": {
            "Cylab\\Cyrange\\": "tests/"
        }
    }
}
```

Listing 3.5: Composer file Cyber Range

There are several remarks to be made about this file.

- symfony/process: The process class executes a command in a sub-process. To avoid security problems, it takes care of the differences between the operating system and escaping arguments. It also replaces PHP functions such as exec, passthru, shell_exec and system;
- symfony/console: The Console component allows to create command-line commands and can be used for any recurring task (cronjobs, imports, etc.);
- php-di/php-di: This library is the dependency injection container for PHP. We'll discuss this further in the section 4;
- phpseclib/phpseclib: It is the PHP Secure Communications Library, with implementations of RSA, AES, SSH2, SFTP, X.509 etc.;
- vlucas/phpdotenv: It loads environment variables from a ".env" file into the program. It is useful for anything that is likely to change between deployment environments like database credentials, etc.;
- cylab-be/php-vbox-api: It is a PHP library developed by Pr. Debatty to drive VirtualBox;
- monolog/monolog: It is helful for sending your logs to files, sockets, inboxes, databases and various web services;
- cylab-be/php-vagrant-cloud: It is a PHP client for Vagrant Cloud that allows you to download Vagrant boxes directly.
- phpunit/phpunit: It is the PHP unit testing framework;
- squizlabs/php_codesniffer: It tokenizes PHP, JavaScript and CSS files and detects violations of a defined set of coding standards;

- slevomat/coding-standard: It works in pair with ”squizlabs/php_codesniffer” and complements Consistence Coding Standard by providing sniffs with additional checks.

Second, the difference between ”require” and ”require-dev” is that ”require-dev” is a list of packages needed to develop this package, or to run tests, etc. The root package development requirements are installed by default.

Finally, the keyword ”autoload” will allow us to use PHP classes without the need to require() or include() them. This is considered a feature of modern programming.
Using PSR-4 here is the newest autoloading standard in PHP, and it requires us to use namespaces. More technical information can be found in the technical documentation[2].

3.3 Command classes

Here we are in the command class section. In this part, we will approach in a more general way the various classes present in this folder as you can see on the figure 3.3.

These different classes are used to ”control” the whole VirtualBox environment. Each class has a specific action, either on a specific virtual machine or more generally on a scenario.

But generally speaking, the different classes are used to instantiate virtual machines, to configure them, to deploy a scenario, to create access accounts for the VPN but also for the Guacamole web interface. You can find bellow the different classes with their general explanation:

- **AbstractCommand:** Its role is to log commands and possible errors as well;
- **DeployResult:** It will return the result of deploying a virtual machine, as well as the fact that it needs RDP enabled;
- **GroupList:** It will return the list of existing virtual machine groups;
- **GroupUp:** It will start all virtual machines in a specified group, recursively;
- **GuacamoleAdapter:** It is an adapter acting as a wrapper between the Guacamole class and AbstractCommand;
  Supprimé
- **RDCommand:** This class extends the AbstractCommand class. However, this class is empty and therefore has no use. My advice would be to delete it too;

[2]https://getcomposer.org/doc/04-schema.md#psr-4

- **RDList:** It returns the list of Remote Desktop accounts configured for Guacamole. However, it also extends the class RDCommand which is empty ... It would be more judicious that this class be directly linked with the AbstractCommand class:
- **ScenarioDeploy:** It will deploy virtual machines and create guacamole accounts using a JSON scenario description. More details on this class can be found bellow;
- **ScenarioDestroy:** It will destroy a specified scenario by deleting virtual machines and associated Guacamole accounts;
- **ScenarioList:** It will return the list of existing scenarios;
- **Update:** This function is useless, it redirects to a ”manifest.json” hosted on the filetray site but which is no longer accessible. Moreover, it is never called in the code. This class should therefore be deleted.;
- **VagrantGet:** It allows you to download a box directly from Vagrant, by specifying its name;
- **VBoxAdapter:** It is an adapter acting as a wrapper between the VBox class and AbstractCommand;
- **VBoxCommand:** It allows to extract the name of a scenario from the name of a specified group;
- **VMCount:** It makes it possible to count the number of virtual machines present;
- **VMDestroy:** It allows deleting a virtual machine specified by som name or its UUID as well as its associated hard disks;
- **VMHalt:** It sends a stop signal from ACPI to a virtual machine specified by its name or UUID;
- **VMImport:** It makes it possible to import an OVA file by specifying the path and start the virtual machine. Optionally, you can preset the number of copies of the OVA file to be made;
- **VMKill:** It will kill a virtual machine specified by its name or UUID by forcing the shutdown;
- **VMList:** It will return the list of running virtual machines with their name, group, UUID and state;
- **VMReset:** It will reset a virtual machine specified by its name or UUID by by simulating the pressing of a reset button;
- **VMSuspend:** It will suspend a virtual machine specified by its name or UUID;
- **VMSuspendAll:** It will suspend all virtual machine, which can be useful for updates and reboots;
- **VMUp:** It will bring up (start it or resume it) a virtual machine specified by its name or UUID;
- **VMUpAll:** It will bring up (start it or resume it) all virtual machine.

- **VPNAdd:** It will add a VPN user by specifying a username and an expiration date;
- **VPNAddList:** It will add multiple VPN users by specifying a list of username and an expiration date;
- **VPNCommand:** This class contains several methods allowing it to perform several actions. It will be able to return an array containing all the usernames, check if a username is valid or make a backup of a configuration file;
- **VPNList:** It will list the VPN users with their username and expiration date;
- **VPNUser:** It describes the structure of a VPN type user with the username and expiration date.

To go into a little deeper about the **ScenarioDeploy class**, when configuring virtual machines, the role of ScenarioDeploy is to:

- Configure the properties of the virtual machines, i.e., their name, description, group membership;
- Configure the hardware of the virtual machines, such as the number of processors, the size of the RAM, the activation of the RDP (Remote Desktop Connection);
- Configure port forwarding rules for NAT via SSH;
- Configure the hostname of the machine, its password, the network configuration but also the installation of the different packet and the execution of the commands in the playbook (an example scenario in JSON format can be found 3.2);
- And finally, start the machine.

3.4 Tests on GitLab

Another essential part of the cyber range is testing. Indeed, there is a dedicated folder in the project, the ”test” folder, which will simulate the creation of scenarios by creating and configuring virtual machines, creating user accounts, testing the connection to these virtual machines with these accounts and then deleting everything.

You will find on figure 3.2 the link between the test classes.

You can find below the different test classes with their general explanation:

- **ExamplesTest**: It extends the ScenarioTestCase class, its role is to test the deployment of example scenarios. Basically, it will use the ScenarioDeploy class in the ”src” folder to deploy two scenarios that have passed to it, ”scenario.json”

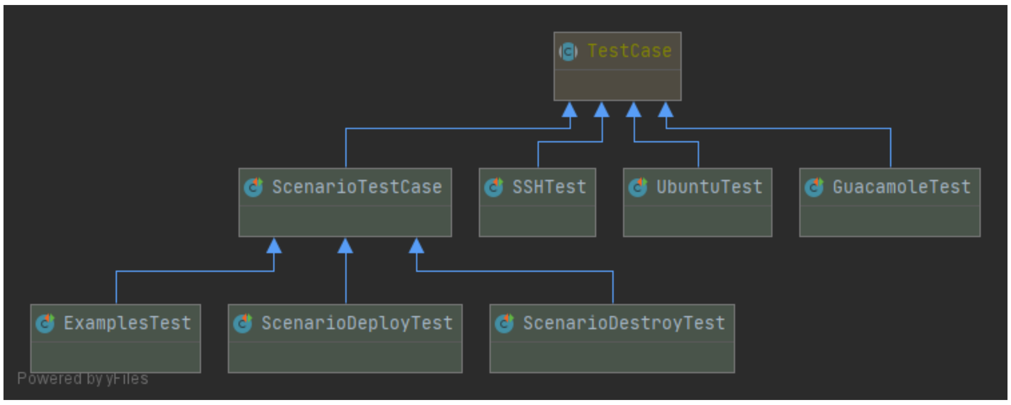


Figure 3.2: Test classes of the cyber range project

that we talked about above in 3.1.
The first is a more complex scenario consisting of 7 virtual machines. In these machines, there will be two student machines as well as a traffic generator, a honeypot, a server and two routers. This can represent a typical corporate network for the students to analyze.
The second is simply an Ubuntu 16.04 virtual machine with a bridged network interface.
After deploying them, it will wait for the machines to boot and then he will count them, to verify that the number of instantiated virtual machines matches the scenario.

- **ScenarioDeployTest**: It extends the ScenarioTestCase class, its role is to test the deployment of a specific scenario. This scenario is composed of 4 virtual machines. In these machines, we will find two test machines with four network interfaces (one per bridge to be connected to the internet, one in private mode for the intranet, one in internal mode to emulate a connection to a cloud and the last one in private mode, for a connection to a DMZ), a machine representing an attack with here two network interfaces (one per bridge to be connected to the internet, and one in internal mode to emulate a connection to a cloud) and a last machine representing a server with a single network card in internal mode to emulate the connection to a cloud.
After deploying them, it will wait for the machines to boot and then he will count them, check in the machine array that two specific names are present, check that the remote display is enabled on ports 16000 and 16001, check the IP address of the machines and finally test if some ports are open.

- **ScenarioDestroyTest**: It extends the ScenarioTestCase class, its role is to test the destruction of scenarios. In practice, this class will do two actions.
The first will be to import four virtual machines from specified ".ova" files. We will then assign these virtual machines to different groups, ask the class to delete machines belonging to a specified group, test if they have been deleted and lastly remove the remaining virtual machines.
The second is to delete Guacamole's accounts and connections. To do this, we will create a user with two different RDP connections, create another user from another scenario and then delete the users from a specified scenario. We will then

verify that this user has been removed before deleting the remaining users.

- **ScenarioTestCase**: Its role is to set up the MySQL database according to a ".sql" file that is given to it so that Guacamole can use it but also to connect to VirtualBox to "clean it", i.e. to go and delete the existing virtual machines and remove the existing DHCP servers.
- **GuacamoleTest**: Its role will be to create a new MySQL database by passing it a .sql schema file so that Guacamole can use it.
  Then it will create a user and test his connection to Guacamole in RDP.
  Finally, it will test the action of defining a password for a newly created user.
- **UbuntuTest**: This class have several roles.
  The first is to review the network interfaces of the VirtualBox server via SSH and compare them to a specified file.
  The second is to test different network interfaces by assigning IPs to them.
  The last one is to test the static configuration of interfaces. To do this, it will import a virtual machine, configure NAT and port forwarding, connect to SSH using port forwarding, configure the network interface and assign it an IP address and finally, after a reboot, test the IP address of this machine.
- **SSHTest**: Its role will be to test the SSH connection to the VirtualBox server from credentials located in environment variables.
  It will then try to transfer a file via SFTP from the local directory to the virtual machine, and then test if the transfer was successful.

3.5 Identified problem

When it comes to the definition of the problem, several ideas may come to mind.

On the one hand, we have the problem that the Royal Military Academy's cyber ranger solution is strongly linked to one single virtualization technology. Indeed, everything in the code is "hardcoded" to use the VirtualBox hypervisor and no other.

Besides, if we wanted to implement another solution for the hypervisor right now, it would be difficult. Indeed, since all classes are linked to VirtualBox, the changes that could be made would perhaps allow us to use another virtualization solution, but at the expense of VirtualBox. By this, I mean that the two solutions, in the current state of the code, would not be able to co-exist.

On the other hand, precisely because of this strong link with VirtualBox, it creates strong links between classes. This ultimately increases class coupling, which is defined by "[...] the degree of interdependence between software modules; a measure of how closely connected two routines or modules are; the strength of the relationships between modules" [3]. In our case, class coupling can be defined as a measure on how classes are connected or dependent on each other.

A common disadvantage of a large class coupling is that a change in one object or class often results in a "ripple effect" that requires changes in other classes as well. A particular object or class may also be more difficult to reuse and/or test because other objects or classes must be included.

Therefore the refactoring of the RMA cyber range system is necessary. Indeed, it can help to reduce the link between classes, to reduce class coupling, and to make it "universal", regardless of the choice of hypervisor solution.

[3] https://en.wikipedia.org/wiki/Coupling_(computer_programming)

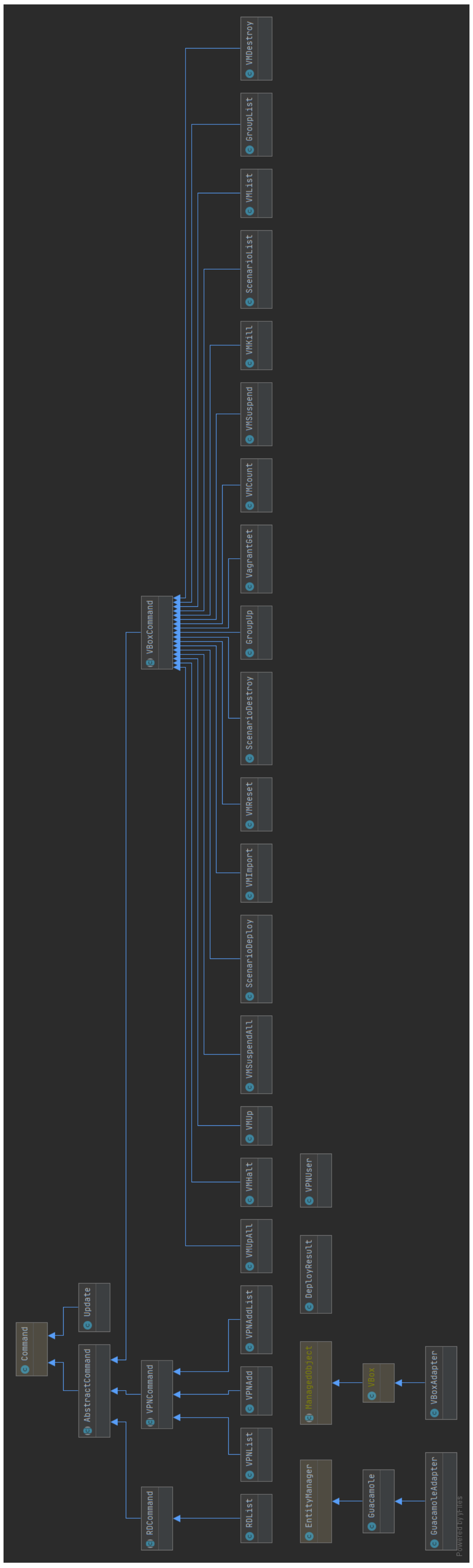


Figure 3.3: PHP diagram before any modifications

Chapter 4

Proposed solution

In this part, we are going to talk about the proposed solution, i.e. the elements that we are going to put in place to be able to solve the problems identified in the part 3.5.

In this part, we will discuss the design patterns, the interfaces and the dependency injection.

These notions discussed here will be useful in part 5, concerning the implementation of the proposed solution.

4.1 Design patterns

"In software engineering, a design pattern is a general repeatable solution to a commonly occurring problem in software design. A design pattern isn't a finished design that can be transformed directly into code. It is a description or template for how to solve a problem that can be used in many different situations." [36]

In other words, design patterns can be used to represent the most effective and widely used best practices in the programming language. In this way, they provide a kind of "guideline" for recurring problems by providing solutions to them. They can also be used as examples when implementing the code, but not as specific piece of code, but a general concept for solving a particular problem.

In programming, we use a design pattern to be able to accelerate the development process by providing proven development paradigms, also to anticipate issues that may only become visible later in the implementation and finally to improve code readability by providing some standardization.

The patterns can be structured in three different categories: [20]

- Creational Patterns
- Structural Patterns
- Behavioral Patterns

Creational design patterns. Creational design patterns give the program more flexibility and versatility by providing various mechanisms for object creation. Thanks to these patterns, it is possible to hide their creation logic, rather than instantiating objects directly using a new operator, but it also allows efficient reuse of existing code.

Here are some examples of creational design patterns:

- Abstract Factory: This creational design pattern allows to produce instances of objects of several families without specifying their concrete classes;
- Builder: The representation and construction of an object are separate. Thus, it is possible to to build representations and types of different complex objects step by step using the same construction code;
- Factory Method: The factory method creates an instance of several derived classes;
- Prototype: This allows to copy existing objects (because it is a fully initialized instance to be copied or cloned) without making the code dependent on other classes;
- Singleton: The singleton ensures that a class has only one instance while providing a global access point to that instance.

Structural design patterns. Structural design patterns concern the composition of classes and objects. They explain how to assemble them into larger structures using the concept of inheritance to achieve new functionality while maintaining efficient and flexible structures.

Here are some examples of structural design patterns:

- Adapter: Allows an incompatible object to collaborate by converting the interface of one object so that another object can understand it;
- Bridge: Allows you to divide a large class or a set of closely related classes into two distinct hierarchies (abstraction and implementation) that can be developed independently of each other by separating the interface of an object from its implementation;
- Composite: Allows you to compose tree-like objects and then work with these structures as if they were individual objects;

- Decorator: Add responsibilities to objects dynamically;
- Dependency Injection: Creates (”injects”) dynamically the dependencies between different objects based on a configuration file for example. This allows the dependencies between components to be expressed not in a static way in the code, but rather in a dynamic way at runtime;
- Facade: It is a class that provides a simple interface to an entire complex subsystem;
- Flyweight: It shares common parts between multiple objects, fitting more objects into the available amount of RAM. It efficiently shares the resources between objects.
- Proxy: It is an object representing another object. An agent controls access to the original object, allowing you to perform before or after the request is sent to the original object.

Behavioral design patterns. Behavioral design models are specifically concerned with communication between objects, but also with the allocation of responsibilities between objects.

Here are some examples of behavioral design patterns:

- Chain of responsibility: It is a way of passing a request between a chain of objects. When receiving one, each handler decides wether to process it or to pass it to the next one in the chain;
- Command: It transforms a command request into a stand-alone object containing all information. It allows us to have more control and options over the requests;
- Iterator: It allows sequential access to the elements of a collection without exposing its underlying representation;
- Mediator: It defines simplified communication between classes, reducing chaotic dependencies between objects;
- Memento: It captures and restore an object’s internal state without revealing the details of its implementation;
- Observer: It lets us define a subscription mechanism that notify any change of the object to a number of classes;
- State: It alters an object’s behavior when its state changes and appears as if the object changed its class;
- Strategy: It makes it possible to define an algorithm inside a class and make its objects interchangeable;
- Template method: It defines the skeleton of an algorithm in the superclass but lets subclasses override specific steps of the algorithm without changing its structure;
- Visitor: It defines a new operation to a class without change.

4.2 Interfaces

Another important point to address is the interface. The interface in object-oriented programming in PHP is widely used.

From the official documentation available on the PHP website, interface objects are defined as follows: [1]
"Object interfaces allow you to create code which specifies which methods a class must implement, without having to define how these methods are implemented."

In other words, an interface is a preview of what an object can do, on the same level as a kind of contract. The interface is defined as a traditional class, except that instead of the term *"class"*, the term *"interface"* will be used.

An interface contains only prototypes of methods without any data variables. It describes the methods to be implemented but not how to do it. This is done in the class that implements the interface. As is the nature of interfaces, all methods in the interface must be public.
The fact that a class implements an interface gives a certain guideline on the minimum methods to be defined. Note that a class can implement several interfaces.

Implementing interfaces in your code has multiple advantages.

On the one hand, as said above, it provides a kind of model, a flexible basic structure for the minimum methods to be defined for the classes that implement the interface.
On the other hand, it allows, for the caller of the object, to be concerned only with the interface and not with the object or even with the implementation of the methods of the object itself.
The interface allows classes that are not related to each other to implement the same set of methods, regardless of their position in the class inheritance hierarchy, and also allows them to define methods according to their requirements.

In our case, the interface will be handy. Indeed, before, when the program needed to call a function to control or drive the deployment of a scenario, a virtual machine, etc., the interface would be useful. It needed to call a "VBOX" object. In contrast, in the future, it will call a "HYPERVISOR" object, making the program more universal, and also allowing to disregard the technology and hypervisor used underneath.

[1] https://www.php.net/manual/en/language.oop5.interfaces.php

4.3 Dependency injection

After having covered the interfaces, we are now going to tackle the dependency injection part.

In this point, we will define and explain what dependency injection is, see a concrete example of the application of dependency injection in a PHP program and finally end with the exploration of dependency injection in another language, here object-oriented Java.

4.3.1 What is DI?

Dependency injection is a design pattern belonging to the structural design pattern (see 4.1 for definition).

The purpose of the dependency injection is *"to implement a loosely coupled architecture in order to get better testable, maintainable and extendable code."*[2]. It can also reduce code coupling by making classes or methods less tightly linked together.

We can use dependency injection to write modular, testable, maintainable and extensible code: [2]

- Modular: dependency injection allows the creation of completely autonomous classes or modules;
- Testable: this makes it easy to write testable code, e.g. with unit tests;
- Maintainable: as indicated above, as each class becomes modular, it becomes easier to manage;
- Extensible: it allows us to easily add new functionality to the code.

There are generally three types of dependency injection:

1. The constructor injection: it is the class constructor that provide the dependencies;
2. The setter injection: a setter method is exposed by the customer. This will be used by the injector to inject the dependency
3. The interface injection: an injector method is provided by the addiction. It will inject the dependency into all the clients she has passed on to her. Clients must implement an interface that exposes a setter method that accepts the dependency.

[2]https://designpatternsphp.readthedocs.io/en/latest/Structural/DependencyInjection/

Using dependency injection will set up a pattern called ”Inversion of Control” (IoC), where dependency control is reversed from the one called to the one calling [24].

It exists a lot of dependency injection containers in different languages. Here is a non-exhaustive list for a few different programming languages: [16]

- .NET: AutoFac, SimpleInjector, Ninject, StructureMap, Castle Windsor, Unity, Spring.NET;
- PHP: Laravel IoC, PHP DI, Zend DI, Symfony, Dice;
- Java: Pico container, Guice, Spring, Silk DI;
- JavaScript: di-lite, inverted, wire.js, bottle.js, pimple, cujo.js (Spring like).

What are the **advantages and disadvantages** of using dependency injection?

We can cite different benefits such as assistance for unit testing, easier application extensions and reduced class coupling. To discover the disadvantages, it is a bit more complex to learn, as a theoretical and comprehension base is required to be able to set it up without causing management problems, but it can also lead to more compilation time errors at runtime. Finally, the use of some frameworks can hinder the automation of some IDEs.[18]

How to choose a Dependency Injection container?

There are several parameters to consider when choosing a solution to implement dependency injection. It seems obvious, but obviously it will depend on the programming language used.
In a second step, it is advisable to use a container with an easy to understand API whose configuration is simple and readable, which has a large enough community and finally meets the needs of the code.

To sum up, a class should focus on the fulfilment of its responsibilities, not on the creation it needs to fulfil these responsibilities.
That is why we set up the dependency injection, to provide the necessary objects to the classes that need them.

4.3.2 PHP example

From a more practical point of view, let us now look at a **concrete example** of the application of dependency injection.

What could be better as a concrete example than the solution of the cyber range of the Royal Military Academy?

As seen above in the different types of dependency injection containers, we use PHP-DI here[3].
As it was explained in 3.2 how Composer managed libraries, PHP-DI is easy to install with Composer. We can even see in the 3.5 listing that *”php-di/php-di”: ”5.4”* is present.

One of the first steps to get started with the PHP-DI library will be to create a container. You can create a pre-configured container instance for easy development, or you can use the container builder to save definition files or modify other options.
In this case, the second option has been chosen; this can be seen in the listing 4.1 below on line 5 where the *”ContainerBuilder”* is used. We can also see that we indicate a file named ”injection.php” where the dependencies we are going to need are defined.

PHP-DI will load the definitions that have been made into the specified file and use them as instructions on how to create objects.
Despite this, objects are only created when they are requested by the container (e.g. *$container->get(...)*) or if they are to be injected into another object.
This means that we can have a large number of definitions in our file, and PHP-DI will not create all the objects unless we specifically ”request” it.

PHP-DI’s definitions are written using a specific language called ”DSL” (Domain Specific Language), which is written in PHP and based on helper functions. In our case, we will use ”constant magic” for the classes.

```
1 #!/usr/bin/env php
2 <?php
3 require_once __DIR__ . '/../vendor/autoload.php';
4 [...]
5 $builder = new \DI\ContainerBuilder();
6 $builder->addDefinitions(__DIR__ . ”/injection.php”);
7 $container = $builder->build();
8 [...]
```

Listing 4.1: Part of the Main.php file

Speaking precisely about this famous ”injection.php” file, we can see in the listing 4.2 below the definition of the objects.

In our case, we will have defined as object the class VBoxAdapater. This will allow us, in the program, to inject the VBoxAdapter class when we need the VBox class.

```
return [
    \Cylab\Vbox\VBox::class =>
        DI\object(\Cylab\Cyrange\Command\VBoxAdapter::class)
];
```

[3]https://php-di.org/

Listing 4.2: Injection.php file

4.3.3 Java example

We are going to show a dependency injection in another language, here, the java programming language. In absolute terms, the idea is almost the same, except for changes in the way of coding. To express this point, we will use the simple and concise example of the article [46].

In Java, a class is said to be dependent on another class if it uses an instance of that class, and this will be called a class dependency.
Let's take the example of a class that needs to access a logger service; it has a dependency on this service class in the listing 4.3.

Following coding best practice in Java, classes should be as independent as possible from other classes. This makes it easier to reuse these classes on the one hand, and on the other hand, to be able to test these classes independently of each other.

If an instance of a class is created by another class using the "new" operator, it cannot be used or tested independently of the other class. This case is called a hard dependency.

```java
package com.vogella.tasks.ui.parts;

import java.util.logging.Logger;

public class MyClass {

    private Logger logger;

    public MyClass(Logger logger) {
        this.logger = logger;
        // write an info log message
        logger.info("This is a log message.")
    }
}
```

Listing 4.3: Example of a normal class in Java with no hard dependencies

Several approaches exist when it comes to describing the dependencies of a class. The most common way in Java is to use so-called annotations. These annotations represent dependencies directly in the class.

As can be seen in the listing 4.4 below, the most common way described in the Java Specification Request 330 (JSR330) is to use the annotations "@Inject" and "@Named".

```java
// import statements left out

public class MyPart {

    @Inject private Logger logger;

    // inject class for database access
    @Inject private DatabaseAccessClass dao;

    @Inject
    public void createControls(Composite parent) {
        logger.info("UI will start to build");
        Label label = new Label(parent, SWT.NONE);
        label.setText("Eclipse 4");
        Text text = new Text(parent, SWT.NONE);
        text.setText(dao.getNumber());
    }
}
```

Listing 4.4: Example of a class using annotations to describe these dependencies

The place where dependency injection can be done is almost identical to PHP, except that the wording changes.

According to the document JSR330, the injection is done first in the constructor injection, in the field injection and finally in the injection method:

- in the constructor of the class, we talk about "construction injection";
- in a field, so it is called "field injection";
- in the parameters of a method, we speak of the "method injection".

Note that it is possible to use the principle of dependency injection on static and non-static fields and methods.

Chapter 5

Implementation

We are now in the implementation part. In this part, we will describe the different development steps that have been made during the realization of the thesis to be able to use the cyber range with several hypervisors, and then implement it with VMware's solution, ESXi.

We will begin by introducing the definition of the problems at present. Then we will do an analysis of the code to understand better how everything fits together and the changes that need to be made. After that, it will be detailed how the interface has been created, but also how the environment variables and dependency injection have been modified. Finally, we will talk about the implementation of VMware ESXi and the problems we encountered.

When looking at the main steps of the implementation, we can mention the following:

- Creation the Hypervisor interface;
- Modification of the VBox class to implement the Hypervisor interface;
- In ScenarioDeploy, make a list of the methods used and then put them in the Hypervisor interface;
- In the ScenarioDeploy class, replace the references to the VBox class by the Hypervisor interface. This way, the ScenarioDeploy class can be used with VBox or ESXi, or even with any other hypervisor (in the future), as long as there is a class that implements Hypervisor;
- Creation the ESXi class that implements the Hypervisor interface;
- Implemention with ESXi.

The idea is that it can be used with any of the two hypervisors (VirtualBox or ESXi).

5.1 Problem definition

For this part of defining the problem, several ideas may come to mind.

On the one hand, we have the problem that the Royal Military Academy's cyber range solution is strongly linked to one single virtualization technology. Indeed, everything in the code is ”hardcoded” to use the VirtualBox hypervisor and no other.

On the other hand, precisely because of this strong link with VirtualBox, it creates strong links between classes. This ultimately increases class coupling, which is defined by ”*[...] the degree of interdependence between software modules; a measure of how closely connected two routines or modules are; the strength of the relationships between modules.*”[1]. In our case, class coupling can be defined as a measure on how classes are connected or dependent on each other.

A common disadvantage of a large class coupling is that a change in one object or class often results in a ”ripple effect” that requires changes in other classes as well.
A particular object or class may also be more difficult to reuse and/or test because other objects or classes must be included.

It is for all these reasons that the refactoring of the RMA cyber range system is necessary, to reduce the link between classes, to reduce class coupling, and to correct bad programming habits.

5.2 Code analysis

The code was analyzed in the item 3. It was concluded that the programming code was very dependent on VirtualBox and that to remedy this problem, several solutions were going to be implemented.

We will start with the creation of an interface, to ”universalize” the code (see point 4.2) and then we will implement the injection of dependency (see point 4.3) making the cyber range compatible with other hypervisors. Finally, the implementation with VMware ESXi will be presented in point 5.6.

[1]https://en.wikipedia.org/wiki/Coupling_(computer_programming)

5.3 Creation of the interface

The first step is to create the ”Hypervisor” interface. As explained above in the point 4.2, the interface will be useful for universalization of the code, but it will also allow us to disregard the technology and hypervisor used underneath. Indeed, later in the code, we will modify the call to this interface.

To create the interface, we first need to check in our code which types of functions the program uses. Indeed, since the ”VBox” class will implement our interface, we must, as indicated in the point 4.2, indicate the prototype of the methods used.

```php
<?php

namespace Cylab\Cyrange;

interface Hypervisor
{
    /**
     * Hypervisor constructor.
     */
    public function __construct();

    public function importMultiple(string $ova, int $instances);

    public function allVMs();
}
```

Listing 5.1: Hypervisor interface

As can be seen in the listing above, prototype functions have already been written. We will skip one step to explain why the interface is not ”empty”.

Indeed, an analysis had been done beforehand to determine the indispensable functions that should be found in the interface, and which will later be redefined in another class managing another hypervisor.

Once the interface has been created, the following changes can be made. To do this, several steps will be required.

The first one will be that the class responsible for VirtualBox implements the newly created interface; that is to say the VBoxAdapter class.

In the following 5.2 listing, we can see how the class was just before its modification and also once the interface has been implemented by VBoxAdapter, the class definition can be found in the following 5.2 listing.

```php
// VBoxAdapter before implementing the interface

```

```
<?php
[...]
class VBoxAdapter extends \Cylab\Vbox\VBox
{
    // Class code
}

// VBoxAdapter after implementing the interface

<?php
[...]
class VBoxAdapter extends \Cylab\Vbox\VBox implements \Cylab\Cyrange\
    Hypervisor
{
    // Class code
}
```

Listing 5.2: VBoxAdapter class implementing the interface

Now that the VBoxAdapter class implements the Hypervisor interface well, we can proceed to the next steps in the code. These next steps will mainly involve refactoring.

To do so, it will be a rather simple principle that we will detail right away.
We will replace the places where the *getVbox()* function was called by the more generic *getHyperviseur()* function. Also, to make it more logical and tend towards this universalization of code, we will rename the variables called *$vbox* to *$hypervisor*.

As can be seen in the following listing 5.3, we have refactorized the code as described above.

```
// ScenarioDeploy before beeing refactored

class ScenarioDeploy extends VBoxCommand
{
    public function __construct(\Cylab\Vbox\VBox $vbox) {
        parent::__construct($vbox);
        // [...]
    }

// ScenarioDeploy after beeing refactored

<?php
class ScenarioDeploy extends VBoxCommand
{

    public function __construct(Hypervisor $hypervisor)
    {
        parent::__construct($hypervisor);
        // [...]
    }
```

Listing 5.3: ScenarioDeploy class being refactored

Then, we do exactly the same thing for the AbstractCommand class, i.e. to replace the *getVBox* function by the *getHypervisor* function; as we can see in the following listing 5.4.

```
// AbstractCommand function before beeing refactored

// [...]
protected function getVBox()
{
 return $this->vbox;
}

// AbstractCommand function after beeing refactored

// [...]
protected function getHypervisor()
{
 return $this->hypervisor;
}
```

Listing 5.4: AbstractCommand class being refactored

As we discussed in point 3.4, in addition to the ”normal” code, there is also a part of tests that are performed when a push is done on GitLab. This other part must also be adapted so that it uses the new methods and variables correctly.

To do this, it will be the same principle as what we have done above, i.e. by replacing *getVbox()* by *getHyperviseur()* and by replacing *$vbox* variables to *$hypervisor*.
An example of several of these code changes in different classes can be found below in the listing 5.5, 5.6 and 5.7.

Here we have an example of refactoring for the class ScenarioDeployTest:

```
public function testDeploy()
{
    $import = new ScenarioDeploy($this->getVbox());
    $import->setRemoteDesktopStartPort(16000);
    // rest of the code

// That becomes:

public function testDeploy()
    {
        $import = new ScenarioDeploy($this->getHypervisor());
        $import->setRemoteDesktopStartPort(16000);
        // rest of the code
```

Listing 5.5: Example of ScenarioDeployTest class refactoring

Here we have an example of refactoring for the class ScenarioDestroy:

```
public function destroyVMs($scenario)
    {
        // Destroy VM's
        $vbox = $this->getVBox();
        foreach ($vbox->allVMs() as $vm) {

// That becomes:

public function destroyVMs($scenario)
{
    // Destroy VM's
    $hypervisor = $this->getHypervisor();
    foreach ($hypervisor->allVMs() as $vm) {
```

Listing 5.6: Example of ScenarioDestroy class refactoring

Here we have an example of refactoring for the class ExampleTest:

```
public function testUbuntu1604()
    {
        $deployer = new ScenarioDeploy($this->getVbox());
        $deployer->setRemoteDesktopStartPort(16000);
        [...]
        $machines = $this->getVbox()->allVMs();
        $this->assertEquals(1, count($machines));
    }

// That becomes:

public function testUbuntu1604()
    {
        $deployer = new ScenarioDeploy($this->getHypervisor());
        $deployer->setRemoteDesktopStartPort(16000);
        [...]
        $machines = $this->getHypervisor()->allVMs();
        $this->assertEquals(1, count($machines));
    }
```

Listing 5.7: Example of ExampleTest class refactoring

5.4 Environment variables

During the dependency injection, as explained more theoretically in point 4.3, and as we will see in more detail in the next point 5.5, there is also a large amount of environment variables that are defined (as you can see in the listing 5.8 below). These are variables that are used by several different classes but in the same way and therefore they do not need to be defined in several different places.

There are two files where these environment variables are stored: the ”env.vbox” file and the ”env.test” file. The first one is for the main program, and the second one is for the tests performed on Gitlab. In our case, we will focus on the first one.

We can see in the following listing 5.8 the file containing the environment variables of the cyber range, before any modification.

This file contains several important environment variables. It contains, for example, the username, password, IP address and port of the VirtualBox server used, but also the location where the OVA images of the virtual machines are stored. The file also includes the IP address, username and password of the MySQL database for the Guacamole connection interface (presented in 3 above).

```
#
# Meant to run cyrange on the same host has vbox and mysql
#

VBOX_USER = "vbox"
VBOX_PASSWORD = "password"
VBOX_SERVER = "127.0.0.1"
VBOX_PORT = 18083
# where ova files are stored when we use relative paths
VBOX_ROOT = "/home/vbox"

VBOX_ADDRESS = "http://${VBOX_SERVER}:${VBOX_PORT}/"

GUAC_MYSQL_HOST = 127.0.0.1
GUAC_MYSQL_USER = root
GUAC_MYSQL_PASSWORD = "cyl@b2019!"

NETWORK_INTERNET=eno1

## Where vagrant images and deployment summaries are stored
CYRANGE_ROOT = "/tmp"
```

Listing 5.8: Environment variable file "env.vbox"

The first modification we are going to make to this file is the addition of a variable that we are going to name "HYPERVISOR". This variable will be useful when injecting dependencies, we will discuss this matter in more detail in the following point 5.5. This will allow us to determine in a simple way and in one place which hypervisor will be used when starting the program.

We will also add environment variables related to the ESX environment, which we will need later when dealing with the implementation with ESXi, detailed in point 5.6 below.

Therefore, we will find the IP address (which will have to be modified depending on the environment), the URL of the REST API, and the username and password of the system.

```
# Where we specify which hypervisor to use
HYPERVISOR = "vbox"

# ESXi environment variables
ESXI_IP = "127.0.0.1"
ESXI_URL = "https://{ESXI_IP}/rest"
ESXI_USER = "Administrator"
ESXI_PASSWORD = "P@ssword123"
```

```
ESXI_USER_PASS = "QWRtaW5pc3RyYXRvcjpQQHNzd29yZDEyMw=="
```

Listing 5.9: Updated environment variables

5.5 Dependency injection

We are now in the dependency injection part. We talked about the theoretical concept and also about its application in the cyber range of the Royal Military School.
As a little reminder, here is in listing 5.10 the current state of the *injection.php* file, before it was modified.

```
return [
    \Cylab\Vbox\VBox::class =>
        DI\object(\Cylab\Cyrange\Command\VBoxAdapter::class)
];
```

Listing 5.10: Injection.php before modification

Instead of retrieving the VBoxAdapter class when the VBox class is requested, we will put a condition here.

Indeed, in the previous point 3, we dealt with the refactoring of the MRE. For the cyber range to be used with several different hypervisors, it must be indicated at a place in the code. Thanks to the refactoring, the program simply expects to have a "Hypervisor" object returned to it.

Consequently, we used the "function()" argument of PHP-DI, allowing us, depending on the "HYPERVISOR" environment variable that had been defined, to inject the appropriate class. If we define in the above environment variable file 5.4 "*HYPERVISOR*" as being "*vbox*", then the VBoxAdapter class will be injected. On the contrary, if it had been defined as "*esxi*", then the ESXiAdapter class will be injected.

```
<?php
return [
    \Cylab\Cyrange\Hypervisor::class =>
        function () {
            if (getenv("HYPERVISOR") == "vbox") {
                return DI\object(\Cylab\Cyrange\Command\VBoxAdapter::class);
            } elseif (getenv("HYPERVISOR") == "esxi") {
                return DI\object(Cylab\Cyrange\Command\ESXiAdapter::class);
            }
        }
];
```

Listing 5.11: Injection.php after modification

5.6 Implementation with ESXi

We are now in the part where we talk about the implementation with ESXi. In this part, we will justify why we chose to use ESXi as another hypervisor, how it was installed and configured. We will detail how we decided to interface, communicate with it. We will explain the methods we need and finally implement them in a class, depending on the choice of interfacing with ESXi.

5.6.1 Why VMware ESXi?

In the previous sections, we explained what we had put in place to make the cyber range compatible with other hypervisors. In this case, we are going to implement it with another hypervisor.
When the question arose as to which hypervisor to use for this implementation, several points were discussed.

Firstly, the current solution runs with a type 2 hypervisor, which, as we know from our analysis earlier in the point 2.1.2.3 but also from the article [14] also referenced in this point, is generally less powerful than a type 1 hypervisor.

Indeed, as a small reminder, type 2 hypervisor is installed directly on the machine, by running directly on the system hardware. This allows it to have direct access to the resources, without having a system or application in between.

This is the reason why we have decided to **move towards a type 1 hypervisor**, compared to a type 2 hypervisor.

In a second step, we had to choose which type 1 hypervisor we were going to use.

On the one hand, we were interested in the thesis that was written by Mr. Michaux [28]. In his thesis, a section is dedicated precisely to the comparison of hypervisors; he concluded that VMware's solution, ESXi, was the most efficient.

On the other hand, ESXi is a solution, often combined with others, widely used in the business world. There is also a whole part of their site dedicated to documentation (as can be seen here by following the link). Finally, there is a whole community aspect available on their site, with dedicated people to answer possible problems.

For these reasons, in addition to moving towards a type 1 hypervisor, we have chosen to **use VMware's solution, ESXi**, to serve as the hypervisor with which we will implement the cyber range.

5.6.2 Installation and configuration

After choosing the hypervisor, we must now proceed to its installation.

In appendix A is a kind of ”tutorial”, explaining step by step the steps that have been performed both for the installation and configuration of the ESXi, but also for the installation and configuration of the vCenter.
The latter being essential to link the communications between our program and the hypervisor.

We will discuss the issue of interfacing with the ESX in the next point 5.6.3.

5.6.3 How to interface with ESXi?

5.6.3.1 Communication

Now that our system’s installed, we are going to have to be able to communicate with it. To do so, we had two choices:

- VMware PowerCLI;
- REST API.

On the one hand, we could use a module made to be used with PowerShell, ”VMware PowerCLI”.
This is defined as a *”[...] command-line and scripting tool built on Windows PowerShell, and provides more than 700 cmdlets for managing and automating vSphere, vCloud Director, vRealize Operations Manager, vSAN, NSX-T, VMware Cloud Services, VMware Cloud on AWS, VMware HCX, VMware Site Recovery Manager, and VMware Horizon environments.”*[2].

It is a pretty powerful tool, allowing us to do everything we need for our cyber range. The only problem with it was to know how to make our PHP program communicate with this tool, created mainly to run on a machine with a Windows operating system. Although it is, of course, also possible to install PowerShell on a computer running Linux

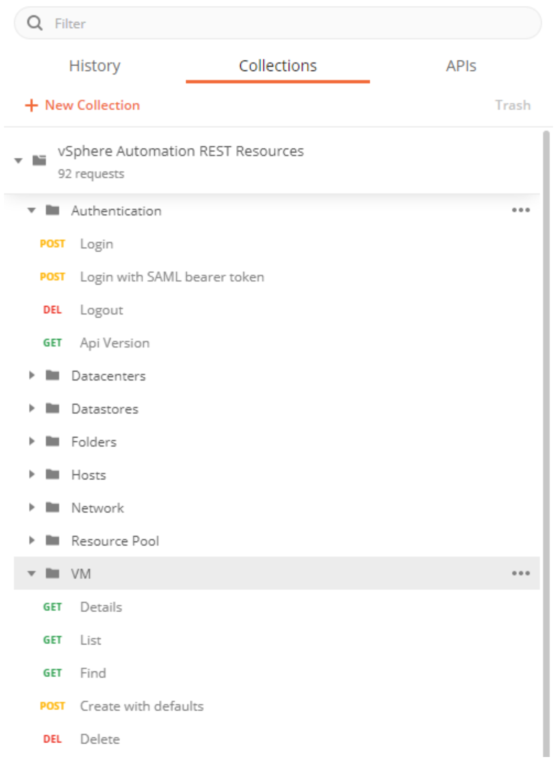
[2]https://code.vmware.com/web/tool/12.0.0/vmware-powercli

[3], this would have required more installation and configuration before starting, making it less efficient.

On the other hand, we have the possibility to drive our hypervisor thanks to REST API requests.

To be able to experiment and test the different available requests, VMware provides on its website what they call the "*vSphere Automation REST API Postman Resources and Samples*"[4]. Those are JSON files that will have to be coupled and imported with the Postman software. Postman is simply a development software that will allow you to test calls to APIs.

Several collections were available, but to base our research, we will use the collection entitled *vSphere-Automation-REST-Resources.postman_collection.json*. This provides the individual API resources. These are stand-alone queries that we will be able to run or use to set up an end-to-end workflow. An overview of the collection of available queries is shown in figure 5.1.



vSphere Automation REST Resources
92 requests
- Authentication
  - POST Login
  - POST Login with SAML bearer token
  - DEL Logout
  - GET Api Version
- Datacenters
- Datastores
- Folders
- Hosts
- Network
- Resource Pool
- VM
  - GET Details
  - GET List
  - GET Find
  - POST Create with defaults
  - DEL Delete

Figure 5.1: Request example from VMware documentation in PostMan

After comparing the two solutions outlined above, **we chose to use the API to make our requests** because of its documentation and its facilitation to be integrated with the rest of the program.

[3]https://github.com/PowerShell/PowerShell
[4]https://code.vmware.com/samples/2562/vsphere-automation-rest-api-postman-resources-and-samples

5.6.3.2 Framework

We know how we are going to communicate with ESXi, we need to look at how we are going to implement that into our program.

From the Postman software, we can export the requests in different languages. Since the whole cyber range has been coded in PHP, it is PHP that we are interested in. Postman offers us 3 ”framework” available to export our requests to PHP:

- **PHP - cURL**: object-oriented wrapper of the PHP cURL extension;
- **PHP - HTTP Request2**: uses pluggable adapters and provides an easy way to perform HTTP requests;
- **PHP - pecl http**: provides a convenient and powerful set of functionality for one of PHPs major applications.

Although some of these frameworks might have been interesting, we finally went to **Guzzle**[5]. Among the various advantages it offered, we could retain that it is a simple interface allowing to build request chains, POST requests, the use of HTTP cookies and JSON data download.

In addition, we also take advantage of the fact that we use Composer in our program to facilitate its installation. So we are going to modify our ”composer.json” file by adding the following line:

```
{
    [...]
    "require": {
        [...]
        "guzzlehttp/guzzle": "~6.0",
        [...]
    }
}
```

Listing 5.12: Adding Guzzle to Composer file

5.6.4 Methods list

Before looking in the VMware API documentation, we first need to perform an analysis of the methods and thus the requests we will need. To do so, we will use the ”ScenarioDeploy” class for this purpose.

After analyzing it, we made a list of the requests we needed:

[5]https://docs.guzzlephp.org/

- Login function
- Test if the session is still active, otherwise login again
- Creation of VMs
- Get list of VMs
- Import a VM from an OVA file
- Configure network of a VM
- Get the API version
- Get/Set the name of VM
- Get/Set state of the VM (also include Start/Pause/Reset/Suspend/Resume/Halt the VM)
- Destroy a VM
- Get network adapter
- Get/Set number of CPU
- Get/Set memory seize

We will find in appendix B the documentation of all the requests which were retained according to the list stated above. This appendix includes all the requests necessary for our program. These have been exported in the 3 frameworks we saw in point 5.6.3.2 (cURL, HTTP Request2 and pecl http) proposed by Postman.

The methods and queries used with Guzzle will be explained in the following section.

5.6.5 ESXi class

Now that we have defined how and with which framework we will communicate with the ESXi, the last part is to create functions to do so. That's why in this section, we will detail the different functions that have been created within the ESXi class, to allow us to manage our program, but also to send our requests through Guzzle.

This class has been designed with the best ways of coding in mind. That is to say that we have, for example, created a method specially made to send our HTTP requests and handle the different errors in a personalized way. This allows us to have cleaner code without unnecessary repetition while managing errors in the best possible way by allowing faster debugging thanks to custom errors.

In addition to detailing the various functions that have been written, a sample response has also been included where necessary.

5.6.5.1 Login function

The first function is the login function. It allows, from the environment variables discussed in section 5.4, to connect to the ESXi API to obtain a session. This session is sent afterwards in each request and is mandatory.
For the connection, we have two choices: either we use the username and password stored in the environment variables, which we must then encode in Base64 before sending the request, either we store directly in the environment variables the encoding specified just before, which will save us a few lines.

```
public function login()
{
    $api_url = getenv('ESXI_URL');
    // First option is to have user and password in env variable
    $user = getenv('ESXI_USER');
    $password = getenv('ESXI_PASSWORD');
    /* Second option is to have the Base64 of "user:password" in env variable
    * $authorization = getenv('ESXI_USER_PASS');
    */
    try {
        //Encode in Base64 the user:password
        $authorization = base64_encode($user . ':' . $password);
        $client = new Client(['headers' => ['Authorization' => 'Basic ' . $
            authorization . '']]);
        $response = $client->request('POST', $api_url . '/rest/com/vmware/cis
            /session');

        // Status code and reason phrase of the response
        if ($response->getStatusCode() == 200) {
            $array = json_decode($response->getBody(), true);
            $sessionId = $array['value'];
            putenv("SESSION_ID=$sessionId");
        } else {
            throw new \Exception("Failed to login: " . $response->
                getStatusCode() . " : "
                . $response->getReasonPhrase());
        }
    } catch (RequestException $e) {
        throw new \Exception("Failed to login", 1, $e);
    }
}
```

Listing 5.13: Login function

5.6.5.2 CheckConnected function

The purpose of this function is to check if our session is still active and to return the boolean "true" if it is and on the contrary the boolean "false" if it is no longer active. CheckConnected will be useful in the next function below.

```php
public function checkConnected()
{
    $apiUrl = getenv('ESXI_URL');
    $sessionId = getenv('SESSION_ID');
    try {
        $client = new Client();
        $response = $client->request('POST', $apiUrl . '/com/vmware/cis/
            session?~action=get', [
            'cookies' => 'vmware-api-session-id=' . $sessionId
        ]);
        if ($response->getStatusCode() == 200) {
            return true;
        } else {
            return false;
        }
    } catch (RequestException $e) {
        throw new \Exception("Failed to check session", 1, $e);
    }
}
```

Listing 5.14: CheckConnected function

5.6.5.3 Logout function

As its name suggests, this function will be used to disconnect us from the ESXi. In other words, it will delete our session.

```php
public function logout()
{
    $apiUrl = getenv('ESXI_URL');
    $sessionId = getenv('SESSION_ID');
    try {
        $client = new Client();
        $response = $client->request('DELETE', $apiUrl . '/com/vmware/cis/
            session', [
            'cookies' => 'vmware-api-session-id=' . $sessionId
        ]);
        if ($response->getStatusCode() == 200) {
            return true;
        } else {
            throw new \Exception("Failed to logout: " . $response->
                getStatusCode() . " : "
                . $response->getReasonPhrase());
        }
    } catch (RequestException $e) {
        throw new \Exception("Failed to logout", 1, $e);
    }
}
```

Listing 5.15: Logout function

5.6.5.4 myRequest function

Once the login function and the session verification requests were set up, another essential function was created, ”myRequest”. This is the function that will allow you to make HTTP requests like ”GET”, ”POST”, etc. to the ESXi. It needs as arguments of the method (if it’s a GET, POST, ...), the URL of the API on which the request will be made (we will see many examples just below) and any additional data.

First it will check to see if our session is still active. If this session is no longer active, the login function will be called. myRequest will then make the request to the API based on what it has received in input, and send everything back to the function that called it.

```
public function myRequest(string $method, string $uri, string $dataBody)
{
    if ($this->checkConnected() == false) {
        $this->login();
    }
    $apiUrl = getenv('ESXI_URL');
    $sessionId = getenv('SESSION_ID');
    try {
        // Create a client with a base URI and return the Response $client
        $client = new Client();
        return
            $client->request($method, $apiUrl . $uri, [
                'cookies' => 'vmware-api-session-id=' . $sessionId,
                'body' => $dataBody
        ]);
    } catch (RequestException $e) {
        throw new Exception("Failed to complete the request", 1, $e);
    }
}
```

Listing 5.16: myRequest function

5.6.5.5 vCenter version function

This function will simply return the version of the vCenter that is in use.

```
\begin{lstlisting}
public function getVcenterVersion()
{
    $response = $this->myRequest('GET', '/appliance/system/version', null);
    return $results = json_decode($response->getBody()->getContents(), true);
}
```

Listing 5.17: vCenter version function

5.6.5.6 Get Host function

This function returns a list with the ID, the name, and the connection and startup status of the hosts. A host is simply a server on which an ESXi is installed.

In our case, we only had one host, that is why in the following answer we see only one.

```php
public function getHostList()
{
    $response = $this->myRequest('GET', '/vcenter/host', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get list of hosts: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.18: Get Host function

```json
{
    "value": [
        {
            "host": "host-28",
            "name": "192.168.1.18",
            "connection_state": "CONNECTED",
            "power_state": "POWERED_ON"
        }
    ]
}
```

Listing 5.19: Get Host function response

5.6.5.7 Search Host function

This function will return the ID, name, connection status and startup status of the hosts by passing the name of the host we are looking for as a parameter.

```php
public function searchHost(string $host)
{
    $response = $this->myRequest('GET', '/vcenter/host?filter.names.1=' . $
        host, null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to search specific host " . $host . " :
            " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.20: Search Host function

```
{
    "value": [
        {
            "host": "host-28",
            "name": "192.168.1.18",
            "connection_state": "CONNECTED",
            "power_state": "POWERED_ON"
        }
    ]
}
```

Listing 5.21: Search Host function response

5.6.5.8 Get Datastore function

This function returns a list including the ID, name, type as well as the remaining disk space and the total capacity of all the datastores present.

A datastore is a manageable storage entity, usually used to store logs, virtual machine files, virtual machine disks, etc. It can be used to manage the data of a virtual machine.

```
public function getDatastoreList()
{
    $response = $this->myRequest('GET', '/vcenter/datastore', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get list of datastores: " . $response
           ->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.22: Get Datastore function

```
{
    "value": [
        {
            "datastore": "datastore-29",
            "name": "ESXi_local",
            "type": "VMFS",
            "free_space": 678754779136,
            "capacity": 992137445376
        }
    ]
}
```

Listing 5.23: Get Datastore function response

5.6.5.9 Search Datastore function

This function returns the ID, name, type as well as the remaining disk space and the total capacity of the datastore searched by looking with its name as an argument.

```
public function searchDatastore(string $datastore)
{
    $response = $this->myRequest('GET', '/vcenter/datastore?filter.names.1='
        . $datastore, null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to search specific datastore " . $
            datastore . " : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.24: Search Datastore function

```
{
    "value": [
        {
            "datastore": "datastore-29",
            "name": "ESXi_local",
            "type": "VMFS",
            "free_space": 678754779136,
            "capacity": 992137445376
        }
    ]
}
```

Listing 5.25: Search Datastore function response

5.6.5.10 Get Folder function

This function returns a list including the ID, name and the type of all the folders present.

A folder can be used to group objects of the same type together, allowing easier management. A folder can contain other folders, specific virtual machines, etc.

```php
public function getFolderList()
{
    $response = $this->myRequest('GET', '/vcenter/folder', null);
    $results = json_decode($response->getBody()->getContents(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get list of folders: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.26: Get Folder function

```json
{
    "value": [
        {
            "folder": "group-s24",
            "name": "datastore",
            "type": "DATASTORE"
        },
        {
            "folder": "group-v22",
            "name": "vm",
            "type": "VIRTUAL_MACHINE"
        }
    ]
}
```

Listing 5.27: Get Folder function response

5.6.5.11 Get Networks function

This function returns a list of all networks available on the server with their name, type and ID.

Simply put, networks are groups of ports assigned to interfaces. Often in a simple architecture, a virtual machine will be connected to the ”VM Network” group, allowing it to have access to the internet for example.

```php
public function getNetworkList()
{
    $response = $this->myRequest('GET', '/vcenter/network', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get list of networks: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.28: Get Networks function

```
{
    "value": [
        {
            "name": "Management Network",
            "type": "STANDARD_PORTGROUP",
            "network": "network-91"
        },
        {
            "name": "VM Network",
            "type": "STANDARD_PORTGROUP",
            "network": "network-30"
        }
    ]
}
```

Listing 5.29: Get Networks function response

5.6.5.12 Get VMs function

This function will return the list of all virtual machines present on the ESXi. This includes details such as the size of the RAM, the ID and name of the virtual machine as well as its status and the number of CPUs assigned to it.

```
public function getVmList()
{
   $response = $this->myRequest('GET', '/vcenter/vm', null);
   $results = json_decode($response->getBody()->getContents(), true);

   if ($response->getStatusCode() == 200) {
       return $results;
   } else {
       throw new \Exception("Failed to get list of VMs: " . $response->
           getStatusCode() . " : "
           . $response->getReasonPhrase(), 1);
   }
}
```

Listing 5.30: Get VMs function

```
{
   "value": [
       {
           "memory_size_MiB": 4096,
           "vm": "vm-102",
           "name": "DebianTest",
           "power_state": "POWERED_OFF",
           "cpu_count": 2
       },
       {
           "memory_size_MiB": 10240,
           "vm": "vm-31",
           "name": "VMware vCenter Server Appliance",
           "power_state": "POWERED_ON",
           "cpu_count": 2
       }
   ]
}
```

Listing 5.31: Get VMs function response

5.6.5.13 Create VM function

This function, as its name indicates, will allow you to create a virtual machine. It waits for a JSON configuration file as input. This JSON file, as we can see in the listing 5.33, regroups all the characteristics of the virtual machine including its location (on which datastore/host/folder, hence the importance of the requests explained above), its virtual hardware, the basic ISO file, the virtual hard disk with its size, and its network connection. Finally, if the creation of the virtual machine went well createVM() will return its ID to us.

```
public function createVM(array $specs)
{
    $dataBody = json_encode($specs);

    $response = $this->myRequest('POST', '/vcenter/vm', $dataBody);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to create VM: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.32: Create VM function

```
{
  "spec": {
    "name": "DebianTest",
    "guest_OS": "DEBIAN_10_64",
    "placement": {
        "datastore": "datastore-29",
        "host": "host-28",
        "folder": "group-v61"
    },
    "memory": {
        "size_MiB": 4096,
        "hot_add_enabled": true
    },
    "floppies": [],
    "cpu": {
      "hot_remove_enabled": true,
      "count": 2,
      "cores_per_socket": 2,
      "hot_add_enabled": true
    },
    "cdroms": [
      {
        "type": "IDE",
        "start_connected": true,
        "backing": {
            "iso_file": "[ESXi_local] ISOs/debian-10.3.0-amd64-netinst.iso",
            "type": "ISO_FILE"
        }
      }
    ],
    "disks": [
      {
        "new_vmdk": {
          "name": "DebianTest",
          "capacity": 17179869184
        }
      }
    ],
    "boot_devices": [
      {
        "type": "CDROM"
      }
    ],
    "boot": {
      "type": "DISK",
      "delay": 1,
      "retry_delay": 1,
      "retry": true
    },
    "nics": [
      {
        "type": "E1000E",
        "pci_slot_number": 0,
        "mac_type": "AUTOMATIC",
        "wake_on_lan_enabled": true,
        "start_connected": true,
        "allow_guest_control": true,
        "backing": {
          "type": "DISTRIBUTED_PORTGROUP",
          "network": "dvportgroup-90"
        }
      }
    ]
  }
}
```

Listing 5.33: JSON configuration file

```
{
    "value": "vm-121"
}
```

Listing 5.34: Create VM function response

5.6.5.14 Deploy OVF VM function

This function allows you to deploy OVF files. It expects as input a JSON configuration file, as shown in Listing 8, which will indicate where the virtual machine will be deployed. It also expects as argument the ID of the OVF file to be imported. If the deployment of the virtual machine went successfully, deployVM() will return its ID to us.

Small parenthesis here, generally in the program with VirtualBox implemented, the files that were imported were OVAs, which are specific to VirtualBox. However, we can, thanks to the "VMware OVF tool" software[6], manage to convert our OVA files into OVF so that it is functional with VMware.

```php
public function deployVM(array $specs, string $ovfId)
{
    $dataBody = json_encode($specs);
    $response = $this->myRequest('POST', '/vcenter/ovf/library-item/id:' . $
        ovfId .'?~action=deploy', $dataBody);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to deploy VM: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.35: Deploy OVF VM function

```json
{
    "deployment_spec": {
        "accept_all_EULA": true,
        "default_datastore_id": "datastore-14"
    },
    "target": {
        "folder_id": "group-v7",
        "host_id": "host-10",
        "resource_pool_id": "resgroup-9"
    }
}
```

Listing 5.36: JSON configuration file for OVF deployment

```json
{
    "value": "vm-125"
}
```

Listing 5.37: Deploy OVF VM function response

[6]https://www.vmwarearena.com/convert-ova-to-ovf-using-the-vmware-ovf-tool/

5.6.5.15 Delete VM function

The purpose of this function will be to delete a specific virtual machine. It expects as input the ID of the virtual machine in question.

If the deletion is performed correctly, the answer we will receive would be a "200 - OK" code. We will receive a code "400 - Bad Request" if, for example, the machine is still running. We will receive a code "404 - Not Found" if the ID we specified does not correspond to any existing virtual machine.

```
public function deleteVM(string $vmId)
{
    $response = $this->myRequest('POST', '/vcenter/vm' . $vmId, null);
    $results = json_decode($response->getBody()->getContents(), true);

    if ($response->getStatusCode() == 200) {
        return true;
    } else {
        throw new \Exception("Failed to delete specific VM " . $vmId . " : "
            . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.38: Delete VM function

```
 200 OK          -> deleted
 400 Bad Request -> still on for example
 404 Not Found   -> if vmId not correct
```

Listing 5.39: Delete VM function response

5.6.5.16 Search VM function

This function will search for the details of a virtual machine by specifying its name, not its ID as in other functions.

It will return the size of RAM memory, its ID, name, status, and the number of CPUs assigned to it.

```
public function searchVM(string $vmName)
{
    $response = $this->myRequest('GET', '/vcenter/vm?filter.names.1=' . $
        vmName, null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to search specific VM " . $vmName . " : 
            " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.40: Search VM function

```
{
    "value": [
        {
            "memory_size_MiB": 4096,
            "vm": "vm-102",
            "name": "DebianTest",
            "power_state": "POWERED_OFF",
            "cpu_count": 2
        }
    ]
}
```

Listing 5.41: Search VM function response

5.6.5.17 Get VM State function

This function will return the state of a specified virtual machine (powered off, running, etc.) by specifying its ID.

```
public function getStateVM(string $vmId)
{
    $response = $this->myRequest('GET', '/vcenter/vm/' . $vmId . '/power',
        null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get VM state: " . $response->
            getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.42: Get VM State function

```
{
    "value": {
        "clean_power_off": true,
        "state": "POWERED_OFF"
    }
}
```

Listing 5.43: Get VM State function response

5.6.5.18 Change VM State function

This function allows you to change the operating state of a virtual machine. It expects as input the id of this one, but also the change of state. Several states are available:

- **stop**: allows to power off the virtual machine;
- **start**: allows to power on the virtual machine;
- **suspend**: allows to suspend the virtual machine;
- **reset**: allows to reset the virtual machine (stimulates a press on a reset button).

When the function has been executed, it can return several different answers; if the status change has been successful, we will receive a "200 - OK", if the ID of the virtual machine passed was not correct, we will receive a "404 - Not Found" and finally, if we ask to start the machine when it is already running, we will receive a "400 - Bad Request" with the error message.

```
public function setStateVM(string $vmId, string $action)
{
    $response = $this->myRequest('POST', '/vcenter/vm/' . $vmId . '/power' .
        $action, null);
    $results = json_decode($response->getBody()->getContents(), true);

    if ($response->getStatusCode() == 200) {
        return true;
    } else {
        throw new \Exception("Failed change state of VM " . $vmId . " : " . $
            response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.44: Change VM State function

```
/*
 * JSON response 200, if OK
 * JSON response 404 Not Found, if VM not found
 * JSON response 400 Bad Request:
{
    "type": "com.vmware.vapi.std.errors.already_in_desired_state",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "Virtual machine is already powered on.",
                "id": "com.vmware.api.vcenter.vm.power.already_powered_on"
            },
            {
                "args": [],
                "default_message": "The attempted operation cannot be
                    performed in the current state (Powered on).",
                "id": "vmsg.InvalidPowerState.summary"
            }
        ]
    }
}
```

Listing 5.45: Change VM State function response

5.6.5.19 Get VM CPU counts function

This function is intended to return details about the number of CPUs in a virtual machine. To do so, it will need the ID of the virtual machine.

In addition to returning the number of CPUs, this function will also give us the number of cores per sockets, but also other information as we can see in the listing 5.47.

```
public function getCpuCount(string $vmId)
{
    $response = $this->myRequest('GET', '/vcenter/vm/' . $vmId . '/hardware/
        cpu', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get CPU count for VM " . $vmId . " :
            " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.46: Get VM CPU counts function

```
{
    "value": {
        "hot_remove_enabled": false,
        "count": 2,
        "hot_add_enabled": false,
        "cores_per_socket": 2
    }
}
```

Listing 5.47: Get VM CPU counts function response

5.6.5.20 Set VM CPU counts function

This function will allow us to set and/or change the number of CPUs of a specific virtual machine. It expects as input a JSON configuration file and the virtual machine ID.

Once the request is executed, the function will return the CPU information, just like the previous request.

```php
public function setCpuCount(array $specs, string $vmId)
{
    $dataBody = json_encode($specs);
    $response = $this->myRequest('PATCH', '/vcenter/vm/' . $vmId . '/hardware
        /cpu', $dataBody);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to set CPU count for VM " . $vmId . " :
            " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.48: Set VM CPU counts function

```json
{
  "spec": {
    "count": 2,
    "hot_remove_enabled": false,
    "cores_per_socket": 1,
    "hot_add_enabled": false
  }
}
```

Listing 5.49: JSON configuration file for setting VM CPU counts

```json
{
    "value": {
        "hot_remove_enabled": false,
        "count": 2,
        "hot_add_enabled": false,
        "cores_per_socket": 2
    }
}
```

Listing 5.50: Set VM CPU counts function response

5.6.5.21 Get VM Memory amount function

This function is intended to return the amount of memory that a virtual machine has. To do so, it will need the ID of the virtual machine.

```php
public function getMemoryAmount(string $vmId)
{
    $response = $this->myRequest('GET', '/vcenter/vm/' . $vmId . '/hardware/
        memory', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get memory amount for VM " . $vmId .
            " : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.51: Get VM Memory amount function

```
{
    "value": {
        "size_MiB": 4096,
        "hot_add_enabled": true
    }
}
```

Listing 5.52: Get VM Memory amount function response

5.6.5.22 Set VM Memory amount function

This function will allow us to set and/or change the amount of ram of a specific virtual machine. It expects as input a JSON configuration file and the virtual machine ID.

Once the request is executed, the function will return the memory information, just like the previous request.

```
public function setMemoryAmount(array $specs, string $vmId)
{
    $dataBody = json_encode($specs);
    $response = $this->myRequest('PATCH', '/vcenter/vm/' . $vmId . '/hardware
        /cpu', $dataBody);

    if ($response->getStatusCode() == 200) {
        return true;
    } else {
        throw new \Exception("Failed to set memory amount for VM " . $vmId .
            " : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.53: Set VM Memory amount function

```
{
  "spec": {
    "size_MiB": 2048
  }
}
```

Listing 5.54: JSON configuration file for setting VM Memory amount

5.6.5.23 Get Network info function

This request will return the ID of the network cards that are connected to a specific virtual machine. To do this, we need to provide the function with the ID of that virtual machine.

```php
public function getNetworkInfoVm(string $vmId)
{
    $response = $this->myRequest('GET', '/vcenter/vm/' . $vmId . '/hardware/
        ethernet', null);
    $results = json_decode($response->getBody(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get network info for VM " . $vmId . "
             : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.55: Get Network info function

```json
{
    "value": [
        {
            "nic": "4000"
        },
        {
            "nic": "4001"
        }
    ]
}
```

Listing 5.56: Get Network info function response

5.6.5.24 Get Network details function

Contrary to the previous request, this one will provide us, for a specific virtual machine, all the details of the network adapters. An overview of the query response can be found in Listing 5.58.

```php
public function getNetworkDetailsVm(string $vmId, string $nicId)
{
    $response = $this->myRequest('GET', '/vcenter/vm/' . $vmId . '/hardware/
        ethernet' . $nicId, null);
    $results = json_decode($response->getBody()->getContents(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to get network details for VM " . $vmId
            . " : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.57: Get Network details function

```
{
    "value": {
        "start_connected": true,
        "pci_slot_number": 0,
        "backing": {
            "connection_cookie": 677560597,
            "distributed_switch_uuid": "50 07 e6 b7 98 b5 ff 60-4c a7 96 06 4
                4 32 bd 51",
            "distributed_port": "20",
            "type": "DISTRIBUTED_PORTGROUP",
            "network": "dvportgroup-90"
        },
        "mac_address": "00:50:56:87:3e:78",
        "mac_type": "ASSIGNED",
        "allow_guest_control": true,
        "wake_on_lan_enabled": true,
        "label": "Network adapter 1",
        "state": "NOT_CONNECTED",
        "type": "E1000E"
    }
}
```

Listing 5.58: Get Network details function response

5.6.5.25 Add Network card function

This function will allow us to add a network card to a specific virtual machine. To do this, it waits for a JSON configuration file containing the type of network card, and in the "backing" part, the type and the network on which the card will be connected. We saw the different possible types in point 5.6.5.11.

In return, the function will return the ID of the network card just created.

```
public function addNetworkVm(array $specs, string $vmId)
{
    $dataBody = json_encode($specs);

    $response = $this->myRequest('POST', '/vcenter/vm/' . $vmId . '/hardware/
        ethernet', $dataBody);
    $results = json_decode($response->getBody()->getContents(), true);

    if ($response->getStatusCode() == 200) {
        return $results;
    } else {
        throw new \Exception("Failed to add network card for VM " . $vmId . "
            : " . $response->getStatusCode() . " : "
            . $response->getReasonPhrase(), 1);
    }
}
```

Listing 5.59: Add Network card function

```
{
    "spec": {
        "type": "E1000",
        "start_connected": true,
        "backing": {
            "type": "{NETWORK_TYPE}",
            "network": "{NETWORK_ID}"
        }
    }
}
```

Listing 5.60: JSON configuration file for adding Network card

```
{
    "value": "4003"
}
```

Listing 5.61: Add Network card function response

5.6.6 ESXiAdapter class

We end with the ESXiAdapter class. This class is similar to the VBoxAdapter class in the way they work. Indeed, this class will link the Hypervisor interface and the ESXi class.

We will, therefore, find here the methods present in the interface, which have been redefined using the adapted functions present in the ESXi class.

We have the "importMultiple()" function, which simply allows importing several virtual machines at once. Importing into ESXi requires in addition to VirtualBox a JSON configuration file telling it where to place this new virtual machine. In our case, we assumed that we had only one ESXi host, hence the previous requests to know the ID of the datastores, folder and host are more straightforward. We then make a loop allowing to import several virtual machines, depending on the number of instances specified.

As for the "allVMs()" function, it will simply return the list of virtual machines present on the ESXi host.

```
class ESXiAdapter extends \Cylab\Cyrange\ESXi implements \Cylab\Cyrange\
    Hypervisor
{
    public function __construct()
    {
        parent::__construct(
            getenv("ESXI_USER"),
            getenv("ESXI_PASSWORD"),
            getenv("ESXI_ADDRESS")
        );
    }

    public function importMultiple(string $ovf, int $instances)
    {
        try {
            $datastore = $this->getDatastoreList();
            $folder_id = $this->getFolderList();
            $host_id = $this->getHostList();
            $resource_pool_id = $this->getResourcePoolList();

            $specs = <<<JSON
                {
                    "deployment_spec": {
                    "accept_all_EULA": true,
                    "default_datastore_id": $datastore
                },
                    "target": {
                        "folder_id": $folder_id,
                        "host_id": $host_id,
                        "resource_pool_id": $resource_pool_id
                    }
                }
            JSON;

            $i = 1;
            $vms = array();
            while ($i <= $instances) {
                $vms[] = ESXi::deployVM($specs, $ovf);
            }
            return $vms;
        } catch (Exception $e) {
            echo ("Failed to import multiple OVF: " . $e);
        }
    }

    public function allVMs()
    {
        try {
            return $this->getVmList();
        } catch (Exception $e) {
            echo ("Failed to retreive VM list: " . $e);
        }
    }
}
```

Listing 5.62: ESXiAdapter class

5.6.7 PHP Diagram

To end this chapter, on the next page, you will find figure 5.2, taking a PHP diagram of the classes in the program after their modifications.

5.7 Problems encountered

We address one of the last parts of this thesis, the one that will talk about the problems that were encountered throughout the development of this thesis. We encountered several ”obstacles” during the realization of this thesis.

The first one, which happened at the very beginning, concerns the programming language. Indeed, it was a novelty for us to have to program in object-oriented PHP. Luckily, we had some necessary PHP programming skills when we created various websites. This allowed us, after having investigated more deeply how object-oriented was articulated, to understand all the code quickly, and then to be able to modify it.

A second, smaller obstacle was the adaptation of the tests running on GitLab. Indeed, after modifying the code in different classes, the tests no longer worked. It was, therefore, necessary to investigate and modify the tests according to the adaptations that had been made in the main code.

A final obstacle that can be cited was encountered when using ESXi. Indeed, to be able to use the API to control the hypervisor from our program, we had to install a vCenter. VMware provides a 60-day trial version. So once everything had to be reinstalled with a new license, to be able to take advantage of the 60-day trial again, allowing us to finish our tests on the requests.

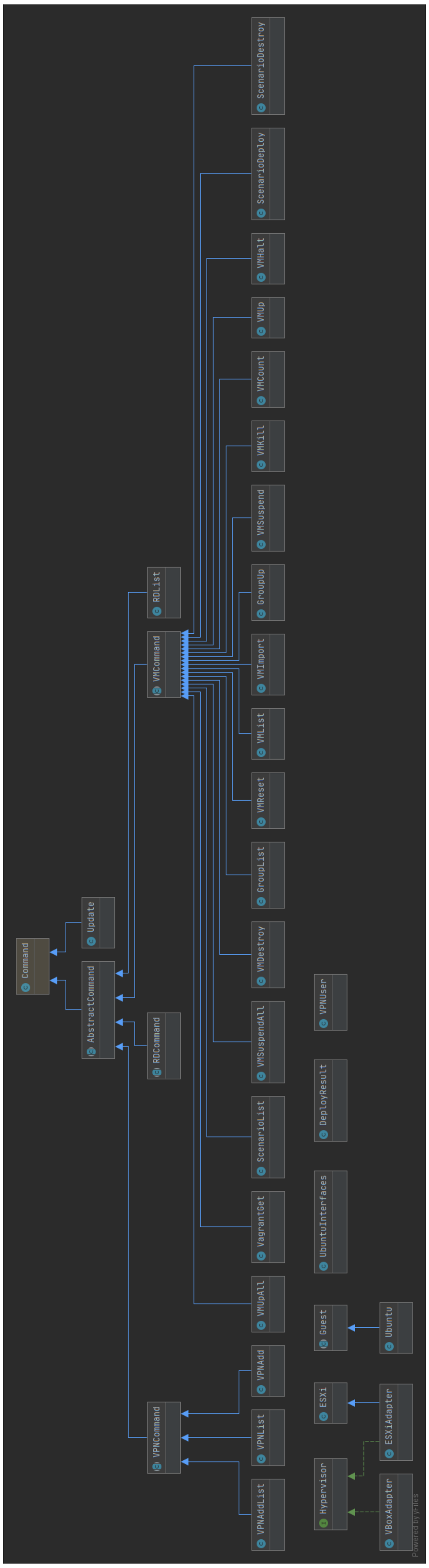


Figure 5.2: Command classes of the cyber range project after modifications

Chapter 6

Conclusion

There have long been ways to improve one's code, be it through the use of design patterns, coding best practices, or even more precisely in our case dependency injection or the use of interfaces. All these methods, techniques, and guides aim to improve the maintainability of your code or to facilitate the addition of new components or features. It is also beneficial during error debugging, for example.

From the point of view of the existing cyber range of the Royal Military Academy, an in-depth analysis of the code and the interactions between them has been carried out. This allowed us to have a complete analysis of the code to be used in future work. However, it also made it possible to shed light on various problems in the code, which brings us to this conclusion.

From the point of view of improving the code, several solutions have been made. Whether in the implementation of design patterns, dependency injection, the application of best coding practices, or the modification of individual classes. These improvements in the code have made it more maintainable, robust, and even easier to test. Another benefit of this code refactoring is discussed in the following paragraph.

In addition to the improvements mentioned in the previous point, we have taken advantage of this to universalize the cyber range to make it compatible with other hypervisors. Indeed, and thanks to the implementation of an interface, we were able to universalize the cyber range code to dissociate it from the hypervisor with which it was working. This allowed us to theoretically apply it to another hypervisor of a different type, VMware ESXi. An analysis was performed to determine how to install it, and then to communicate with it through the current cyber range. The equivalence of requests made with VirtualBox was found to work best with ESXi.

The limitation of this contribution lies in the fact that everything theoretically has been achieved, but unfortunately, it could not be tested on the hardware of the Royal Military School. This point will be further developed in section 7 regarding future works.

To conclude, this work led to the implementation of improvements in the existing code, allowing later to universalize the code to implement it with another hypervisor.

Chapter 7

Future works

Due to a lack of time and health circumstances at the moment, we were, unfortunately, unable to physically implement the ESXi on the ERM servers.

One of the future contributions would be to be able to install ESXi on one of the servers of the Royal Military School and to be able to see our program working with another hypervisor than VirtualBox.

On the other hand, it might be interesting to have two identical servers, one that would be installed and configured to use VirtualBox, and the other for ESXi. This would allow us to measure the performance of each hypervisor by basing our comparison on various criteria such as CPU consumption, RAM, disk usage, and why not also power consumption for defined scenarios. Consequently, we could compare the results obtained by Mr. Michaux during the realization of his thesis and see if the results corroborate each other.

A second contribution, which will be in line with continuing to improve the code, would be the refactorization of what has been called ”the orchestrator”. Indeed, all the different commands and especially the ”ScenarioDeploy” command takes care of creating the VMs based on the scenario. However, all its classes are linked together, and one, in particular, takes care of performing many tasks. It could be interesting to start on the principle of ”single responsibility” for each class.

Appendices

Appendix A

ESXi and VCenter installation and configuration

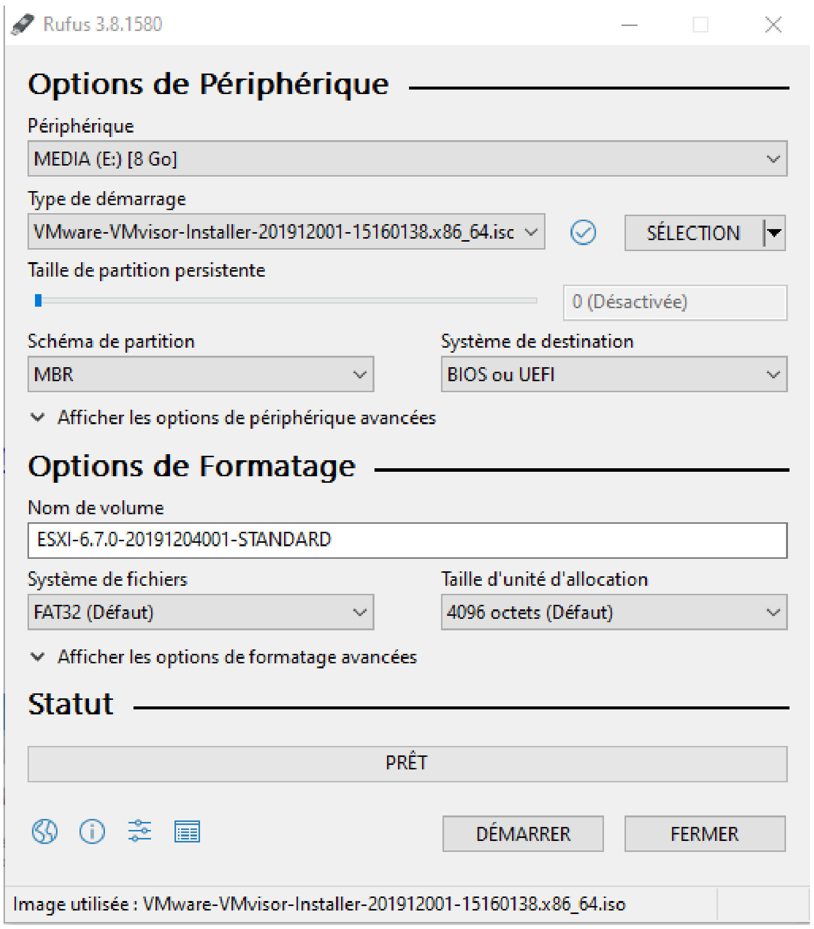
The next pages are a kind of tutorial that was made during the installation of the ESX. It includes the installation and configuration of the ESX, but also the installation and configuration of the vCenter, which allows us to use the Rest API.

ESXi and VCenter installation and configuration

ESXi installation

Creation of the bootable USB key

To create the bootable USB, we need to use Rufus (on Windows):



Rufus 3.8.1580

Options de Périphérique

Périphérique: MEDIA (E:) [8 Go]

Type de démarrage: VMware-VMvisor-Installer-201912001-15160138.x86_64.iso

Taille de partition persistente: 0 (Désactivée)

Schéma de partition: MBR

Système de destination: BIOS ou UEFI

Options de Formatage

Nom de volume: ESXI-6.7.0-20191204001-STANDARD

Système de fichiers: FAT32 (Défaut)

Taille d'unité d'allocation: 4096 octets (Défaut)

Statut

PRÊT

Image utilisée : VMware-VMvisor-Installer-201912001-15160138.x86_64.iso

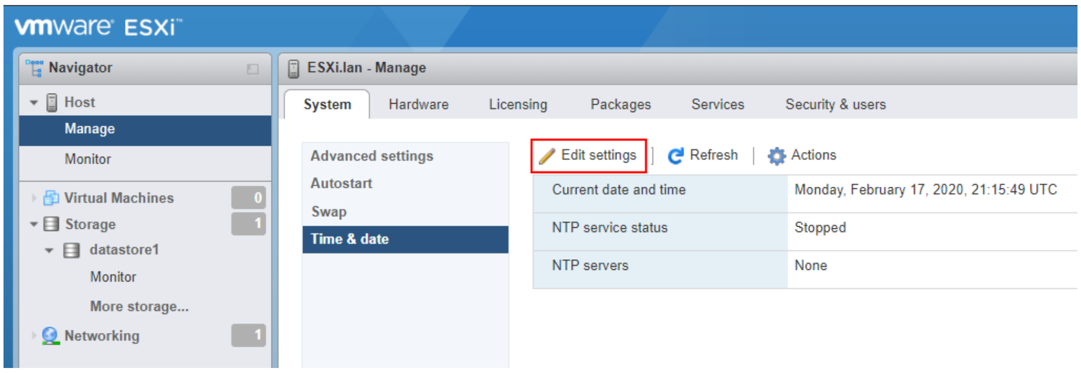
SÉLECTION
Afficher les options de périphérique avancées
Afficher les options de formatage avancées
DÉMARRER
FERMER

Then we need to boot the computer/server on that previously created bootable USB and follow the installation steps by filling out the hostname, IP address, root password, etc.

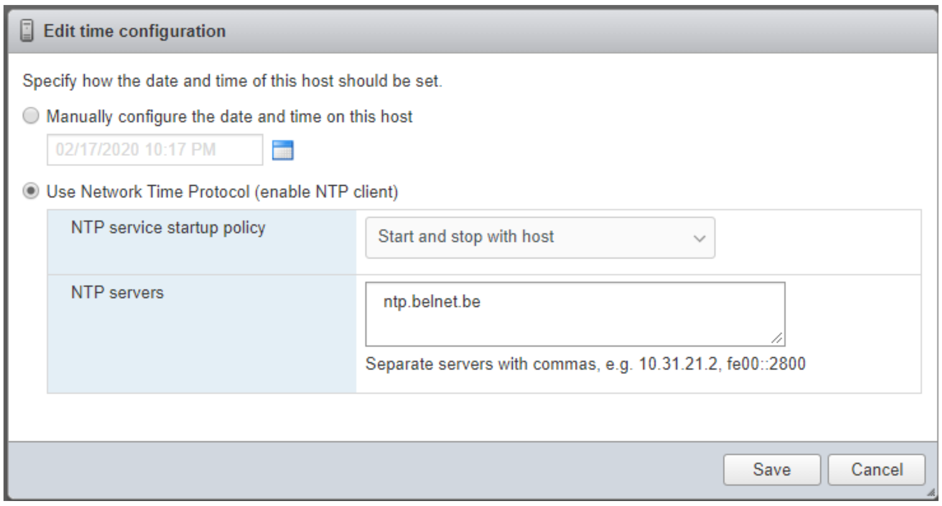
ESXi configuration

NTP

To be sure the time is set correctly, we specify the Belnet NTP server in the parameters.



Current date and time	Monday, February 17, 2020, 21:15:49 UTC
NTP service status	Stopped
NTP servers	None



Specify how the date and time of this host should be set.

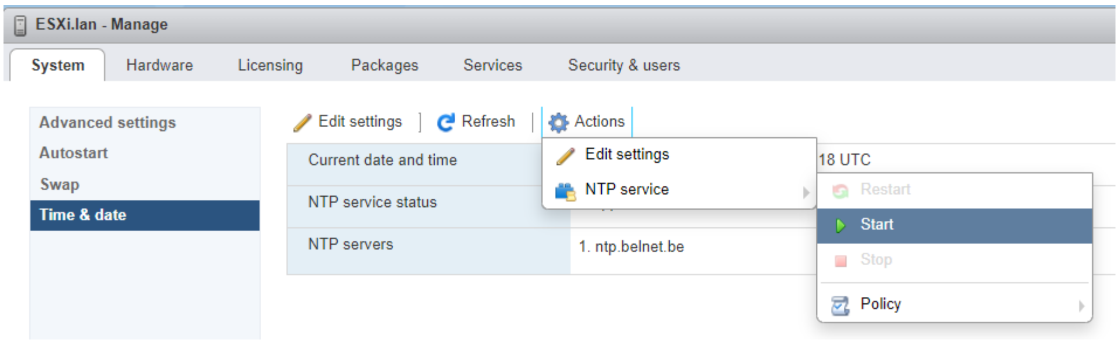
Manually configure the date and time on this host

02/17/2020 10:17 PM

Use Network Time Protocol (enable NTP client)

NTP service startup policy	Start and stop with host
NTP servers	ntp.belnet.be Separate servers with commas, e.g. 10.31.21.2, fe00::2800

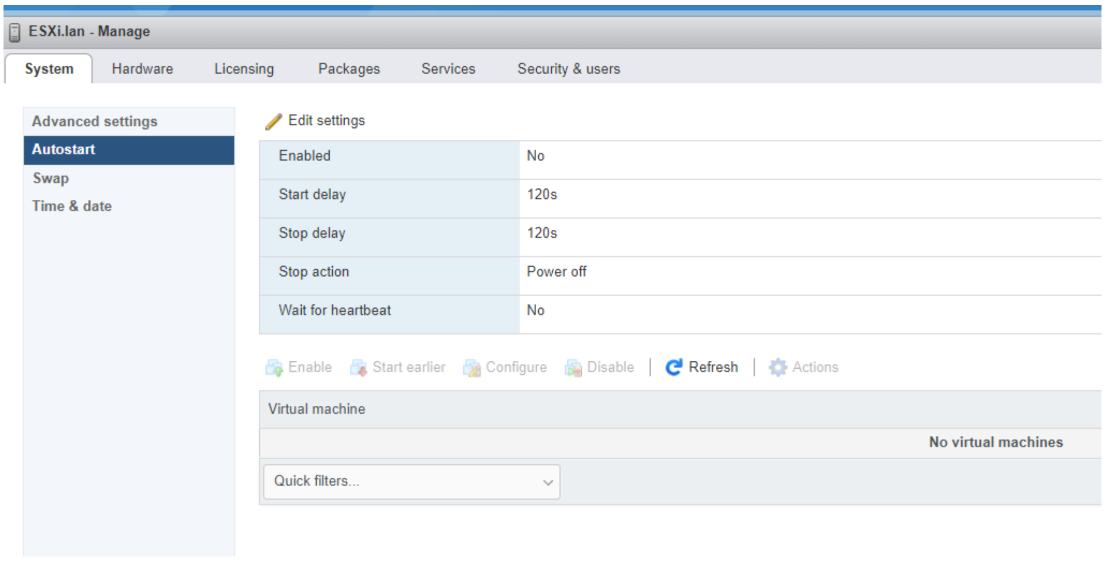
We then start the service.



Current date and time	18 UTC
NTP service status	
NTP servers	1. ntp.belnet.be

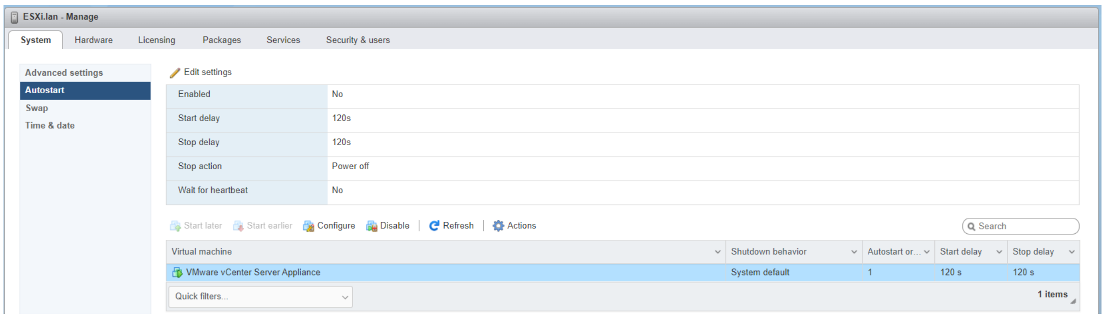
Auto-start

Here we will skip a few steps, but we will activate the automatic start of the vCenter later.



Enabled	No
Start delay	120s
Stop delay	120s
Stop action	Power off
Wait for heartbeat	No

Virtual machine
No virtual machines



Enabled	No
Start delay	120s
Stop delay	120s
Stop action	Power off
Wait for heartbeat	No

Virtual machine	Shutdown behavior	Autostart or...	Start delay	Stop delay
VMware vCenter Server Appliance	System default	1	120 s	120 s

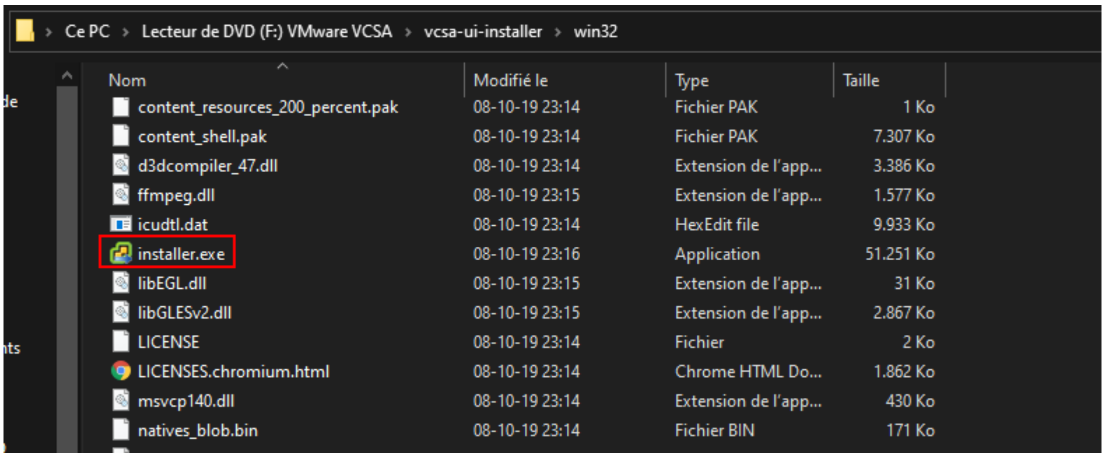
vCenter installation

The installation of the vCenter will be done via a machine connected to the same network as the ESXi, there is no longer any need to create a bootable USB key as before.

We need to get the ISO for installing the vCenter at this link: https://my.vmware.com/group/vmware/evalcenter?p=vsphere-eval

An account must be created in order to receive a 60-day trial license.

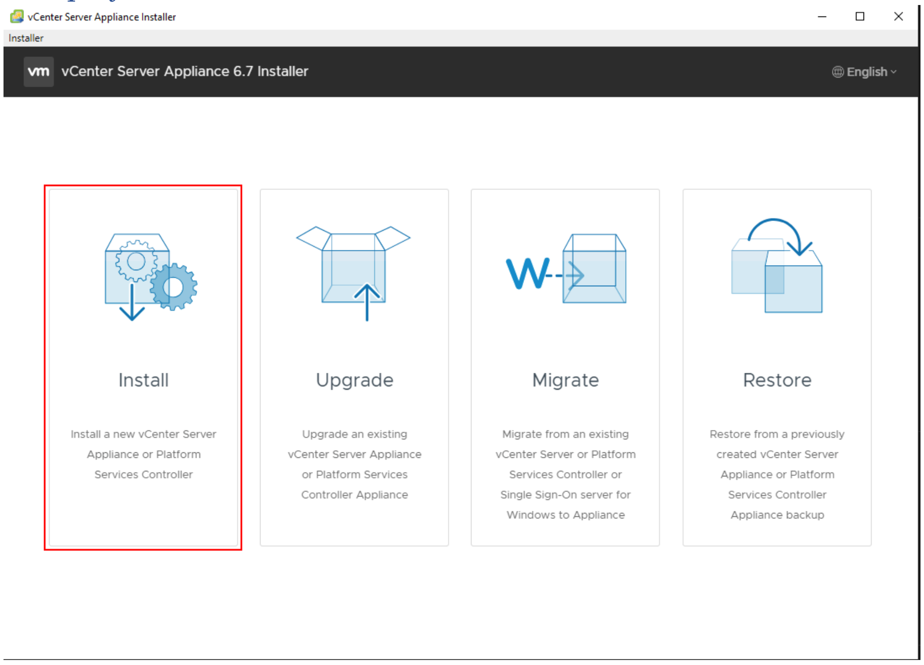
After mounting the ISO, we launch the installer.



Ce PC › Lecteur de DVD (F:) VMware VCSA › vcsa-ui-installer › win32

Nom	Modifié le	Type	Taille
content_resources_200_percent.pak	08-10-19 23:14	Fichier PAK	1 Ko
content_shell.pak	08-10-19 23:14	Fichier PAK	7.307 Ko
d3dcompiler_47.dll	08-10-19 23:14	Extension de l'app...	3.386 Ko
ffmpeg.dll	08-10-19 23:15	Extension de l'app...	1.577 Ko
icudtl.dat	08-10-19 23:14	HexEdit file	9.933 Ko
installer.exe	08-10-19 23:16	Application	51.251 Ko
libEGL.dll	08-10-19 23:15	Extension de l'app...	31 Ko
libGLESv2.dll	08-10-19 23:15	Extension de l'app...	2.867 Ko
LICENSE	08-10-19 23:14	Fichier	2 Ko
LICENSES.chromium.html	08-10-19 23:14	Chrome HTML Do...	1.862 Ko
msvcp140.dll	08-10-19 23:14	Extension de l'app...	430 Ko
natives_blob.bin	08-10-19 23:14	Fichier BIN	171 Ko

vCenter deployment



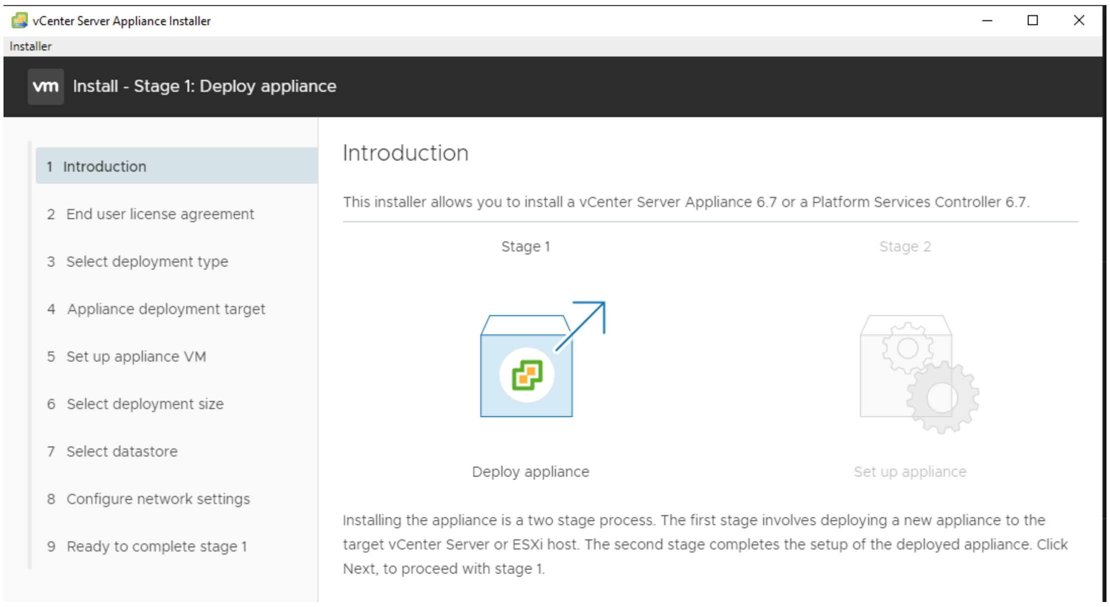
vCenter Server Appliance Installer

Installer

vCenter Server Appliance 6.7 Installer

English
Install
Install a new vCenter Server Appliance or Platform Services Controller
Upgrade
Upgrade an existing vCenter Server Appliance or Platform Services Controller Appliance
Migrate
Migrate from an existing vCenter Server or Platform Services Controller or Single Sign-On server for Windows to Appliance
Restore
Restore from a previously created vCenter Server Appliance or Platform Services Controller Appliance backup

The first step is the deployment of the appliance:


vCenter Server Appliance Installer

Installer

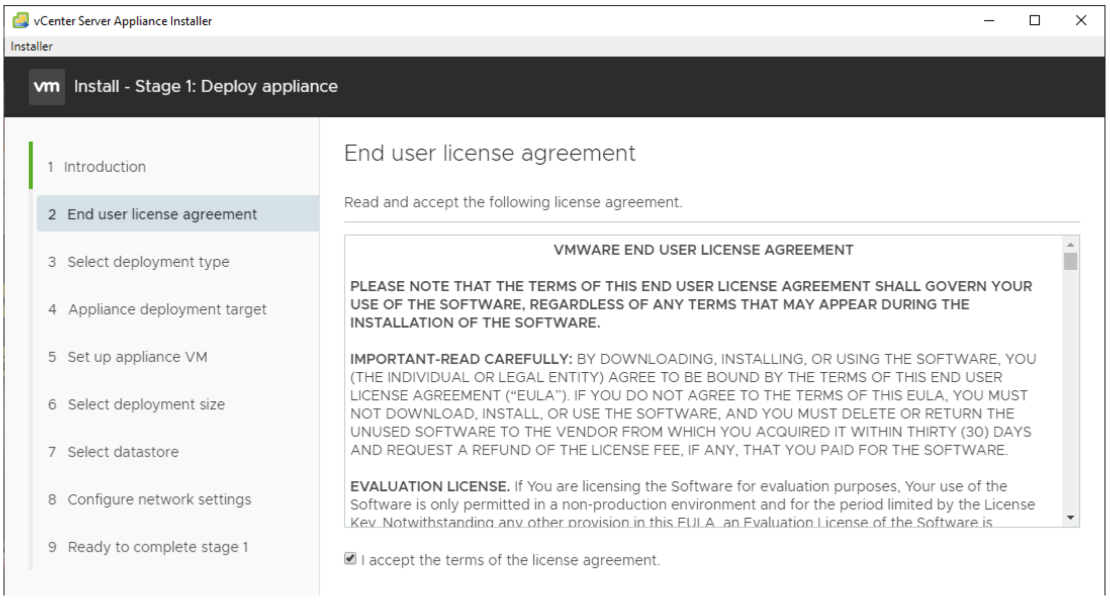
Install - Stage 1: Deploy appliance

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Introduction

This installer allows you to install a vCenter Server Appliance 6.7 or a Platform Services Controller 6.7.

Installing the appliance is a two stage process. The first stage involves deploying a new appliance to the target vCenter Server or ESXi host. The second stage completes the setup of the deployed appliance. Click Next, to proceed with stage 1.


vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy appliance

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

End user license agreement

Read and accept the following license agreement.

VMWARE END USER LICENSE AGREEMENT

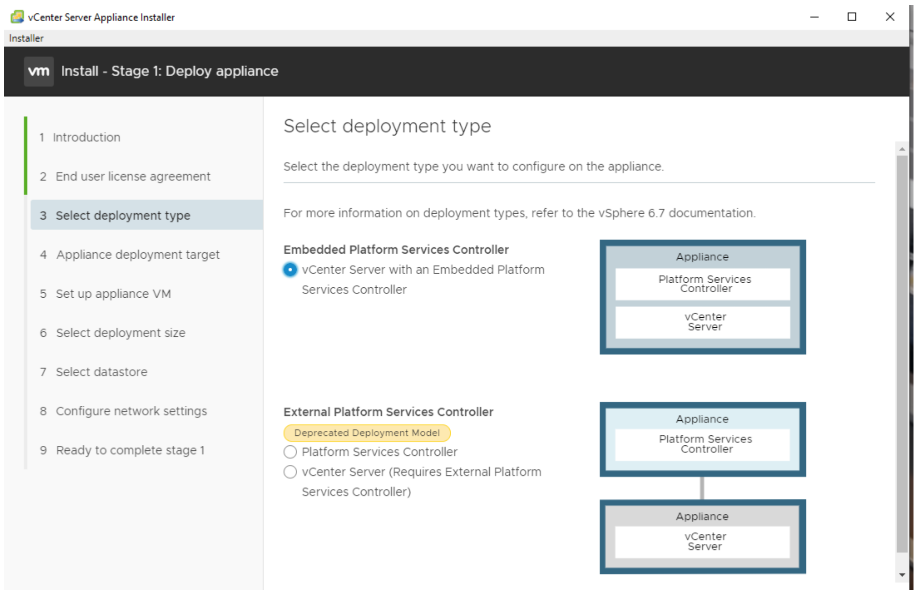
PLEASE NOTE THAT THE TERMS OF THIS END USER LICENSE AGREEMENT SHALL GOVERN YOUR USE OF THE SOFTWARE, REGARDLESS OF ANY TERMS THAT MAY APPEAR DURING THE INSTALLATION OF THE SOFTWARE.

IMPORTANT-READ CAREFULLY: BY DOWNLOADING, INSTALLING, OR USING THE SOFTWARE, YOU (THE INDIVIDUAL OR LEGAL ENTITY) AGREE TO BE BOUND BY THE TERMS OF THIS END USER LICENSE AGREEMENT ("EULA"). IF YOU DO NOT AGREE TO THE TERMS OF THIS EULA, YOU MUST NOT DOWNLOAD, INSTALL, OR USE THE SOFTWARE, AND YOU MUST DELETE OR RETURN THE UNUSED SOFTWARE TO THE VENDOR FROM WHICH YOU ACQUIRED IT WITHIN THIRTY (30) DAYS AND REQUEST A REFUND OF THE LICENSE FEE, IF ANY, THAT YOU PAID FOR THE SOFTWARE.

EVALUATION LICENSE. If You are licensing the Software for evaluation purposes, Your use of the Software is only permitted in a non-production environment and for the period limited by the License Key. Notwithstanding any other provision in this EULA, an Evaluation License of the Software is

☑ I accept the terms of the license agreement.

For the type of deployment, we choose the "embedded" version.


vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy appliance

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Select deployment type

Select the deployment type you want to configure on the appliance.

For more information on deployment types, refer to the vSphere 6.7 documentation.

Embedded Platform Services Controller

[✓ vCenter Server with an Embedded Platform Services Controller]

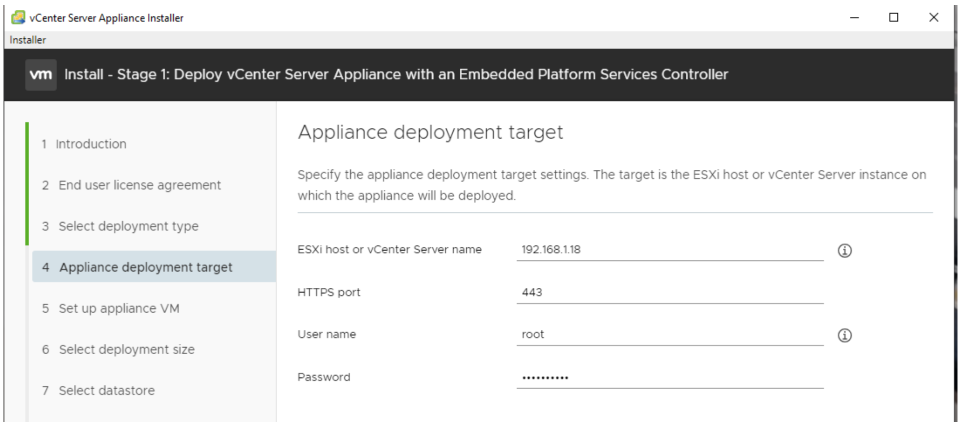
External Platform Services Controller

Deprecated Deployment Model

Platform Services Controller

vCenter Server (Requires External Platform Services Controller)

Here we enter the IP address of the ESXi on which the vCenter will be deployed and we accept the certificate.


vCenter Server Appliance Installer

Installer

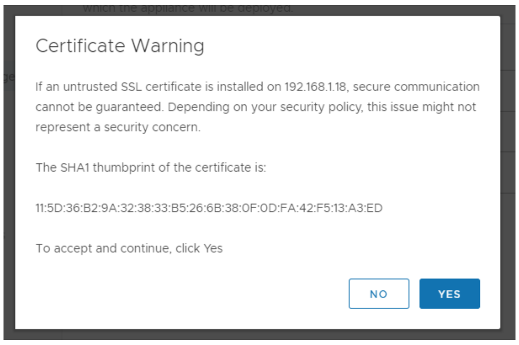
Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore

Appliance deployment target

Specify the appliance deployment target settings. The target is the ESXi host or vCenter Server instance on which the appliance will be deployed.

ESXi host or vCenter Server name	192.168.1.18
HTTPS port	443
User name	root
Password	••••••••••


Certificate Warning

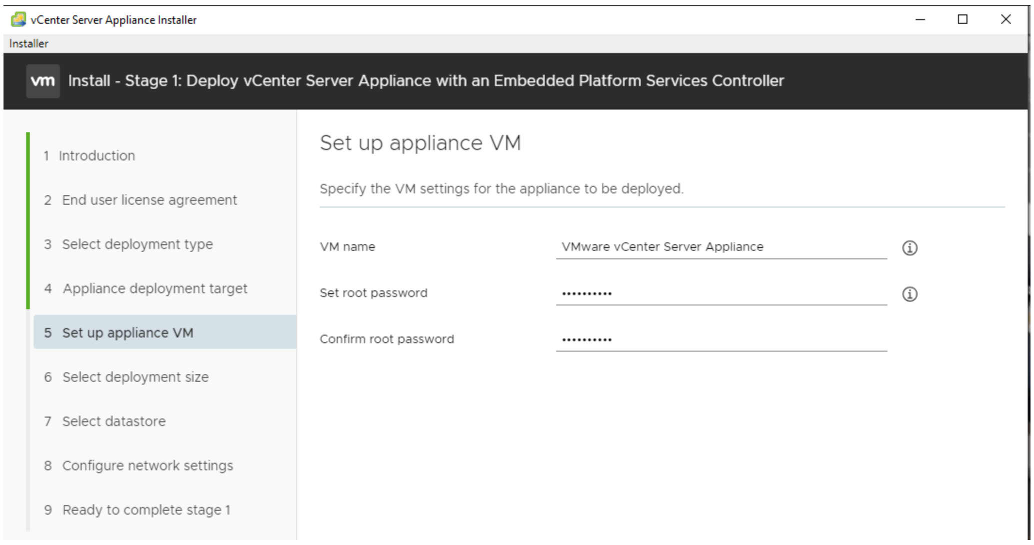
If an untrusted SSL certificate is installed on 192.168.1.18, secure communication cannot be guaranteed. Depending on your security policy, this issue might not represent a security concern.

The SHA1 thumbprint of the certificate is:

11:5D:36:B2:9A:32:38:33:B5:26:6B:38:0F:0D:FA:42:F5:13:A3:ED

To accept and continue, click Yes

We define the name and password of the vCenter.



vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

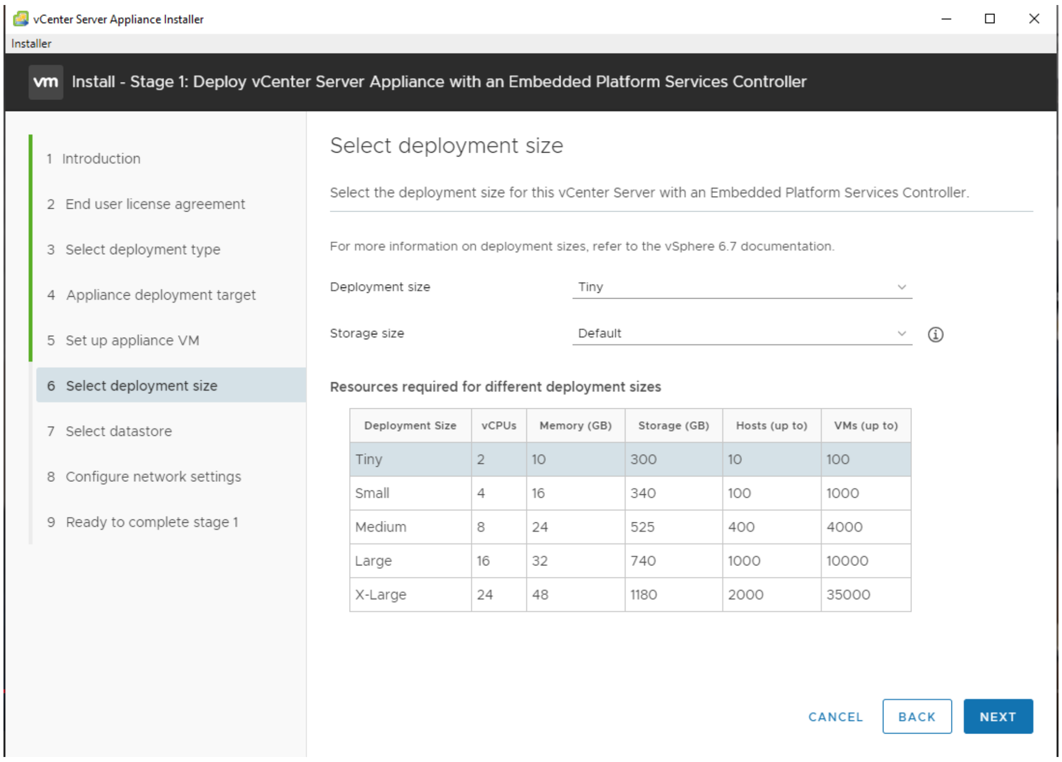
1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Set up appliance VM

Specify the VM settings for the appliance to be deployed.

VM name	VMware vCenter Server Appliance
Set root password	••••••••••
Confirm root password	••••••••••

Depending on the number of virtual machines running and the power of the server, we choose the size of the deployment ("Tiny" in our case).



vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Select deployment size

Select the deployment size for this vCenter Server with an Embedded Platform Services Controller.

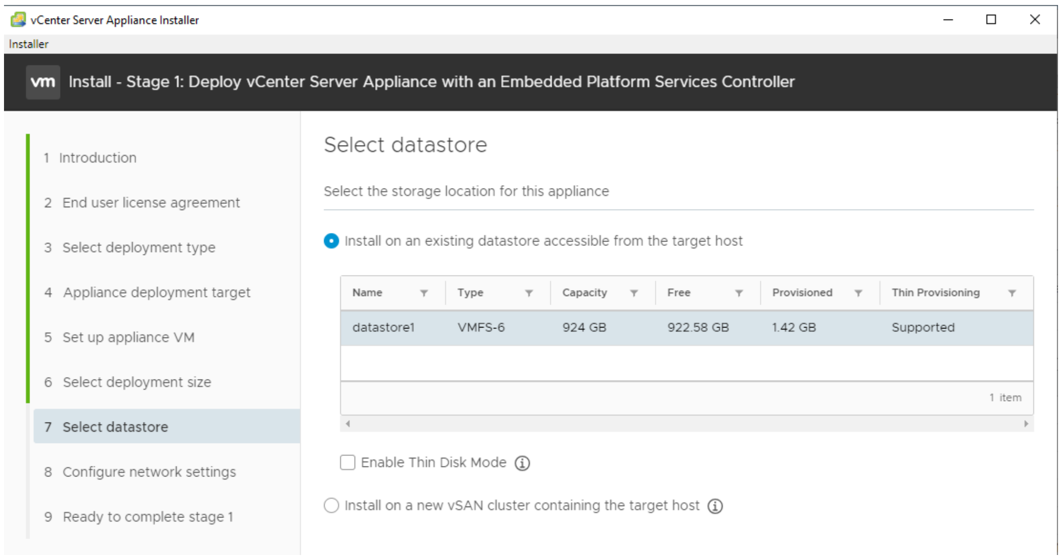
For more information on deployment sizes, refer to the vSphere 6.7 documentation.

Deployment size	Tiny
Storage size	Default

Resources required for different deployment sizes

Deployment Size	vCPUs	Memory (GB)	Storage (GB)	Hosts (up to)	VMs (up to)
Tiny	2	10	300	10	100
Small	4	16	340	100	1000
Medium	8	24	525	400	4000
Large	16	32	740	1000	10000
X-Large	24	48	1180	2000	35000

We choose the datastore on which the appliance will be installed (if there are several disks in the server for example).


vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Select datastore

Select the storage location for this appliance

● Install on an existing datastore accessible from the target host

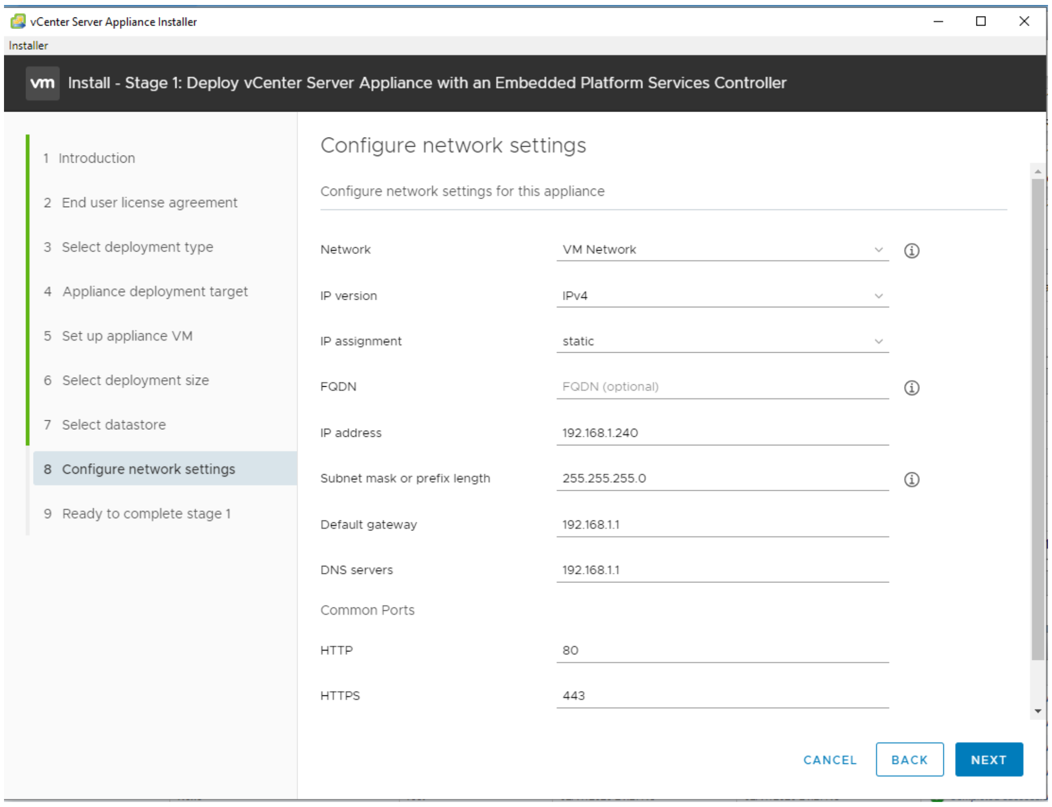
Name	Type	Capacity	Free	Provisioned	Thin Provisioning
datastore1	VMFS-6	924 GB	922.58 GB	1.42 GB	Supported

1 item

☐ Enable Thin Disk Mode

○ Install on a new vSAN cluster containing the target host

We define the vCenter network parameters


vCenter Server Appliance Installer

Installer

Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

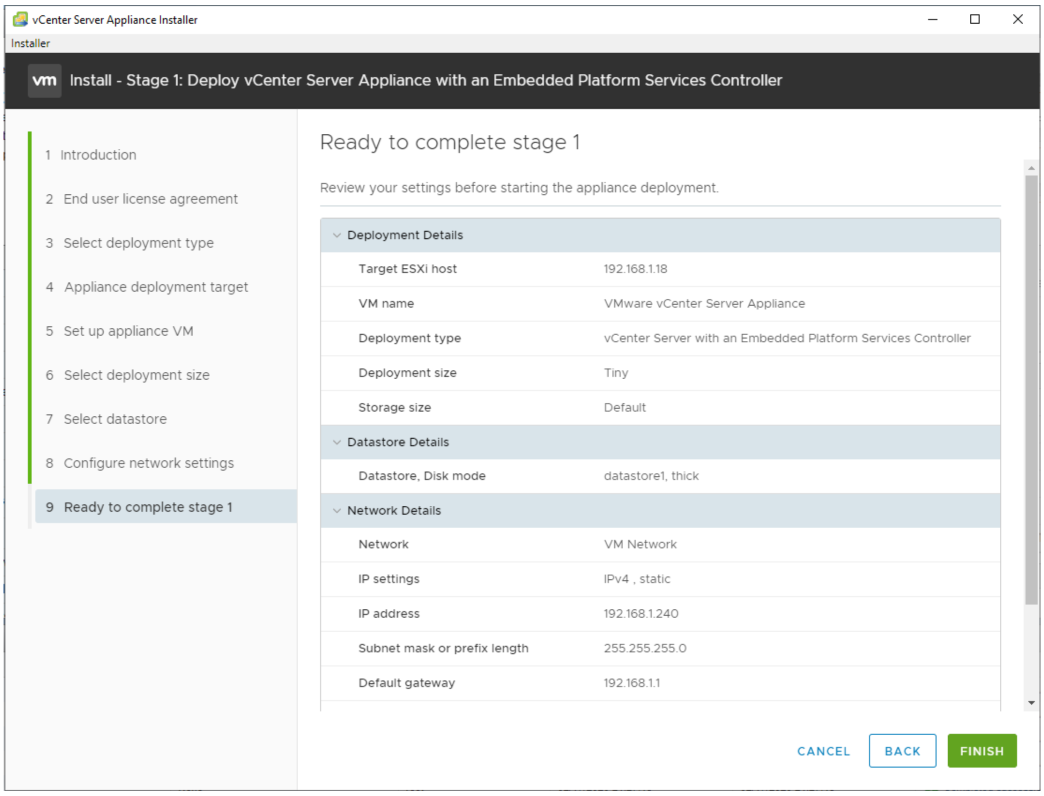
1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1

Configure network settings

Configure network settings for this appliance

Network	VM Network
IP version	IPv4
IP assignment	static
FQDN	FQDN (optional)
IP address	192.168.1.240
Subnet mask or prefix length	255.255.255.0
Default gateway	192.168.1.1
DNS servers	192.168.1.1
Common Ports	
HTTP	80
HTTPS	443

We click on "Finish" and the deployment begins.

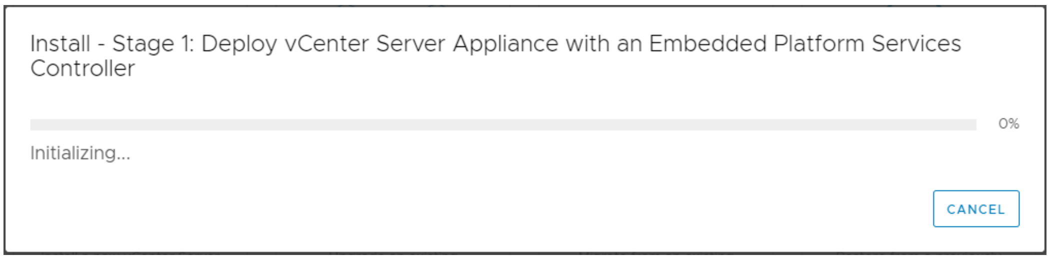


vCenter Server Appliance Installer
Installer
vm Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller
1 Introduction
2 End user license agreement
3 Select deployment type
4 Appliance deployment target
5 Set up appliance VM
6 Select deployment size
7 Select datastore
8 Configure network settings
9 Ready to complete stage 1
CANCEL
BACK
FINISH

Ready to complete stage 1

Review your settings before starting the appliance deployment.

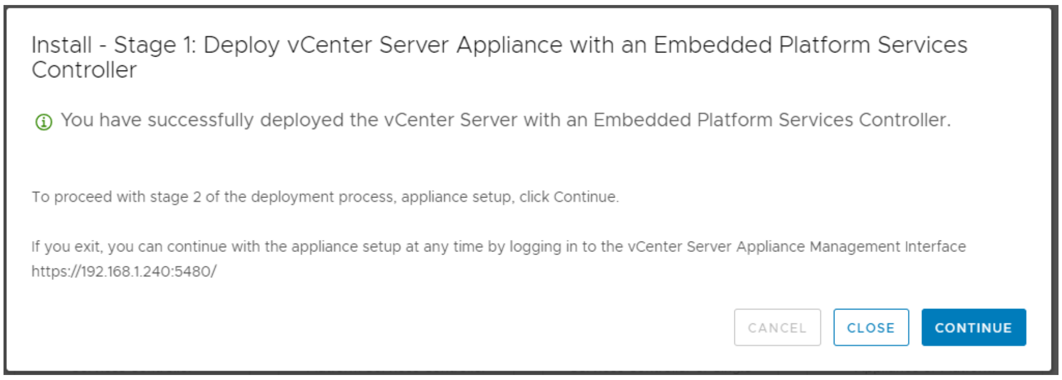
Deployment Details	
Target ESXi host	192.168.1.18
VM name	VMware vCenter Server Appliance
Deployment type	vCenter Server with an Embedded Platform Services Controller
Deployment size	Tiny
Storage size	Default
Datastore Details	
Datastore, Disk mode	datastore1, thick
Network Details	
Network	VM Network
IP settings	IPv4 , static
IP address	192.168.1.240
Subnet mask or prefix length	255.255.255.0
Default gateway	192.168.1.1



Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

0%

Initializing...



Install - Stage 1: Deploy vCenter Server Appliance with an Embedded Platform Services Controller

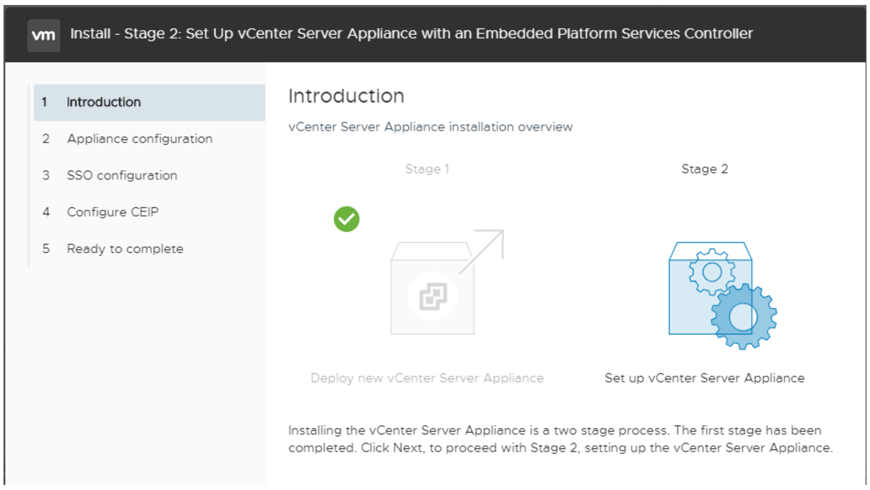
You have successfully deployed the vCenter Server with an Embedded Platform Services Controller.

To proceed with stage 2 of the deployment process, appliance setup, click Continue.

If you exit, you can continue with the appliance setup at any time by logging in to the vCenter Server Appliance Management Interface https://192.168.1.240:5480/

vCenter configuration

We will now configure the vCenter



Install - Stage 2: Set Up vCenter Server Appliance with an Embedded Platform Services Controller

1 Introduction
2 Appliance configuration
3 SSO configuration
4 Configure CEIP
5 Ready to complete

Introduction

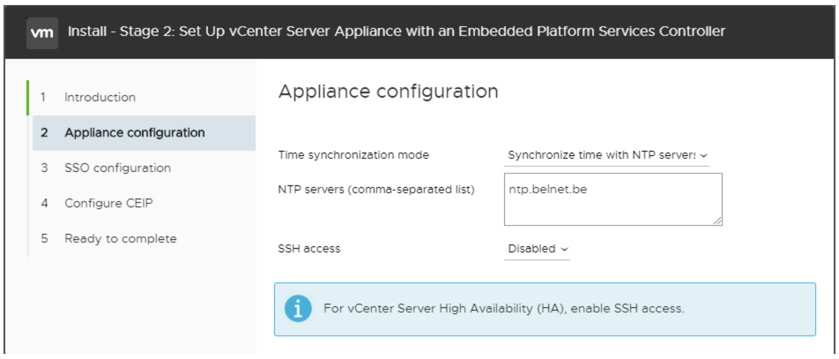
vCenter Server Appliance installation overview

Stage 1
Stage 2
Deploy new vCenter Server Appliance
Set up vCenter Server Appliance

Installing the vCenter Server Appliance is a two stage process. The first stage has been completed. Click Next, to proceed with Stage 2, setting up the vCenter Server Appliance.

NTP

We start with the NTP, as previously done with the ESXi.



Install - Stage 2: Set Up vCenter Server Appliance with an Embedded Platform Services Controller

1 Introduction
2 Appliance configuration
3 SSO configuration
4 Configure CEIP
5 Ready to complete

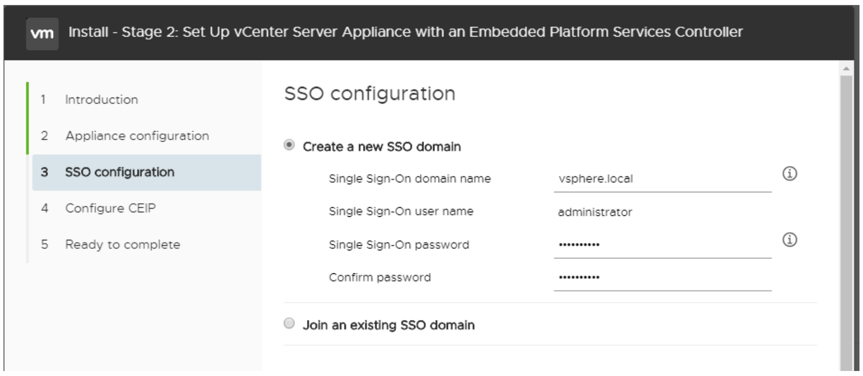
Appliance configuration

Time synchronization mode	Synchronize time with NTP servers
NTP servers (comma-separated list)	ntp.belnet.be
SSH access	Disabled

For vCenter Server High Availability (HA), enable SSH access.

SSO domain

We then configure the SSO domain. For our part, we will create a new one.



Install - Stage 2: Set Up vCenter Server Appliance with an Embedded Platform Services Controller

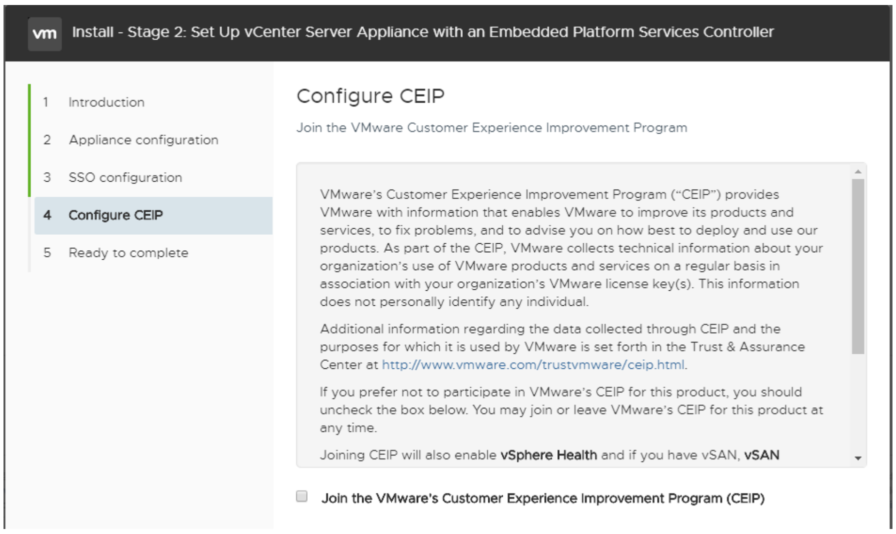
1 Introduction
2 Appliance configuration
3 SSO configuration
4 Configure CEIP
5 Ready to complete

SSO configuration

● Create a new SSO domain

Single Sign-On domain name	vsphere.local
Single Sign-On user name	administrator
Single Sign-On password	••••••••••
Confirm password	••••••••••

○ Join an existing SSO domain



vm
Install - Stage 2: Set Up vCenter Server Appliance with an Embedded Platform Services Controller
1 Introduction
2 Appliance configuration
3 SSO configuration
4 Configure CEIP
5 Ready to complete

Configure CEIP

Join the VMware Customer Experience Improvement Program

VMware's Customer Experience Improvement Program ("CEIP") provides VMware with information that enables VMware to improve its products and services, to fix problems, and to advise you on how best to deploy and use our products. As part of the CEIP, VMware collects technical information about your organization's use of VMware products and services on a regular basis in association with your organization's VMware license key(s). This information does not personally identify any individual.

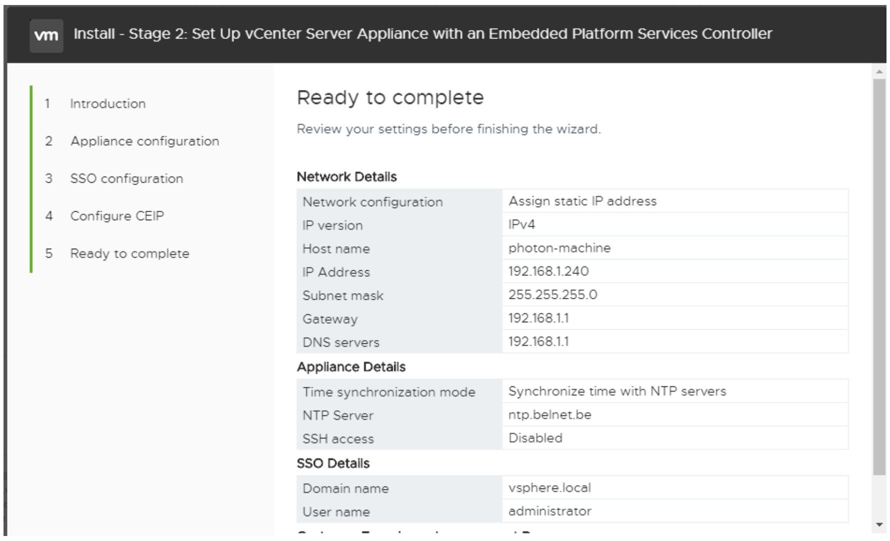
Additional information regarding the data collected through CEIP and the purposes for which it is used by VMware is set forth in the Trust & Assurance Center at http://www.vmware.com/trustvmware/ceip.html.

If you prefer not to participate in VMware's CEIP for this product, you should uncheck the box below. You may join or leave VMware's CEIP for this product at any time.

Joining CEIP will also enable **vSphere Health** and if you have vSAN, **vSAN**

☐ **Join the VMware's Customer Experience Improvement Program (CEIP)**

After reviewing the parameters, the second phase is executed.



vm
Install - Stage 2: Set Up vCenter Server Appliance with an Embedded Platform Services Controller
1 Introduction
2 Appliance configuration
3 SSO configuration
4 Configure CEIP
5 Ready to complete

Ready to complete

Review your settings before finishing the wizard.

Network Details

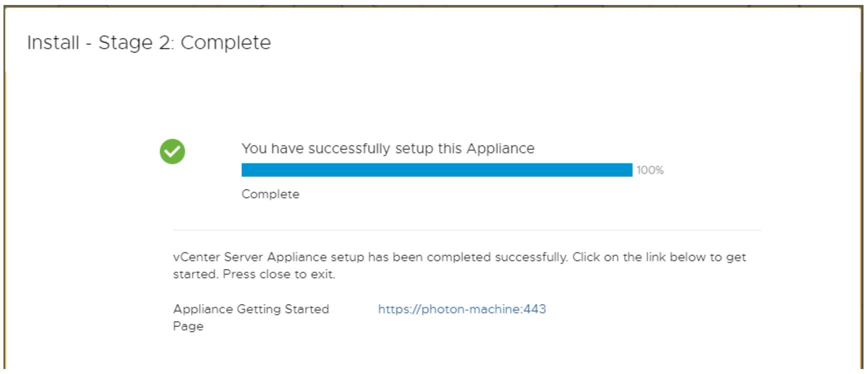
Network configuration	Assign static IP address
IP version	IPv4
Host name	photon-machine
IP Address	192.168.1.240
Subnet mask	255.255.255.0
Gateway	192.168.1.1
DNS servers	192.168.1.1

Appliance Details

Time synchronization mode	Synchronize time with NTP servers
NTP Server	ntp.belnet.be
SSH access	Disabled

SSO Details

Domain name	vsphere.local
User name	administrator



Install - Stage 2: Complete

You have successfully setup this Appliance

100%

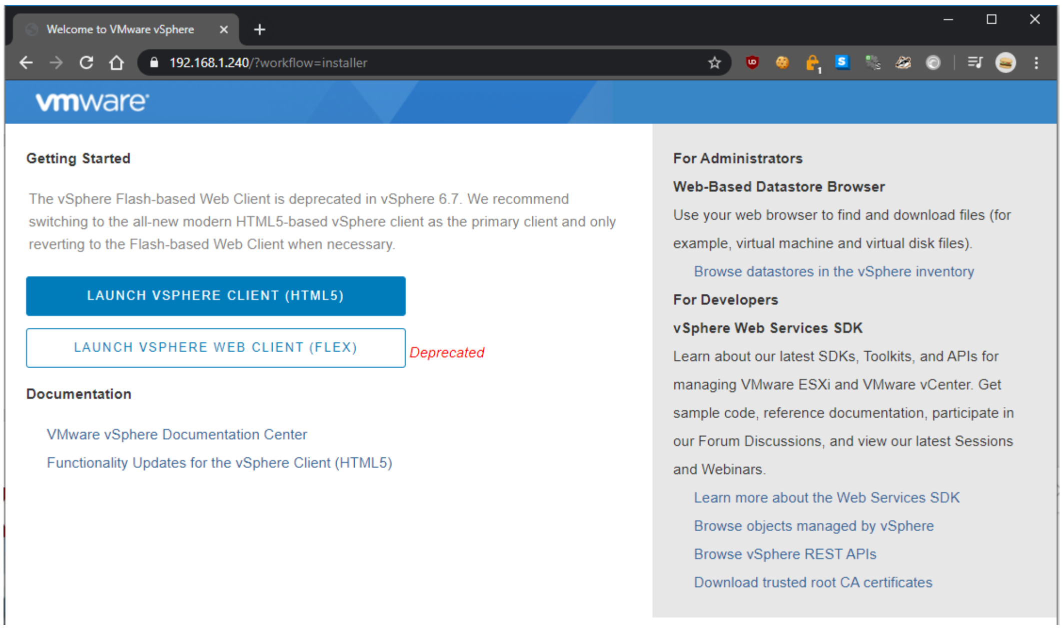
Complete

vCenter Server Appliance setup has been completed successfully. Click on the link below to get started. Press close to exit.

Appliance Getting Started Page	https://photon-machine:443

Dashboard

To continue the configuration, we go to the following link: https://[IP-VCENTER]/?workflow=installer



Getting Started

The vSphere Flash-based Web Client is deprecated in vSphere 6.7. We recommend switching to the all-new modern HTML5-based vSphere client as the primary client and only reverting to the Flash-based Web Client when necessary.

LAUNCH VSPHERE CLIENT (HTML5)

LAUNCH VSPHERE WEB CLIENT (FLEX) Deprecated

Documentation

VMware vSphere Documentation Center
Functionality Updates for the vSphere Client (HTML5)

For Administrators

Web-Based Datastore Browser

Use your web browser to find and download files (for example, virtual machine and virtual disk files).

Browse datastores in the vSphere inventory

For Developers

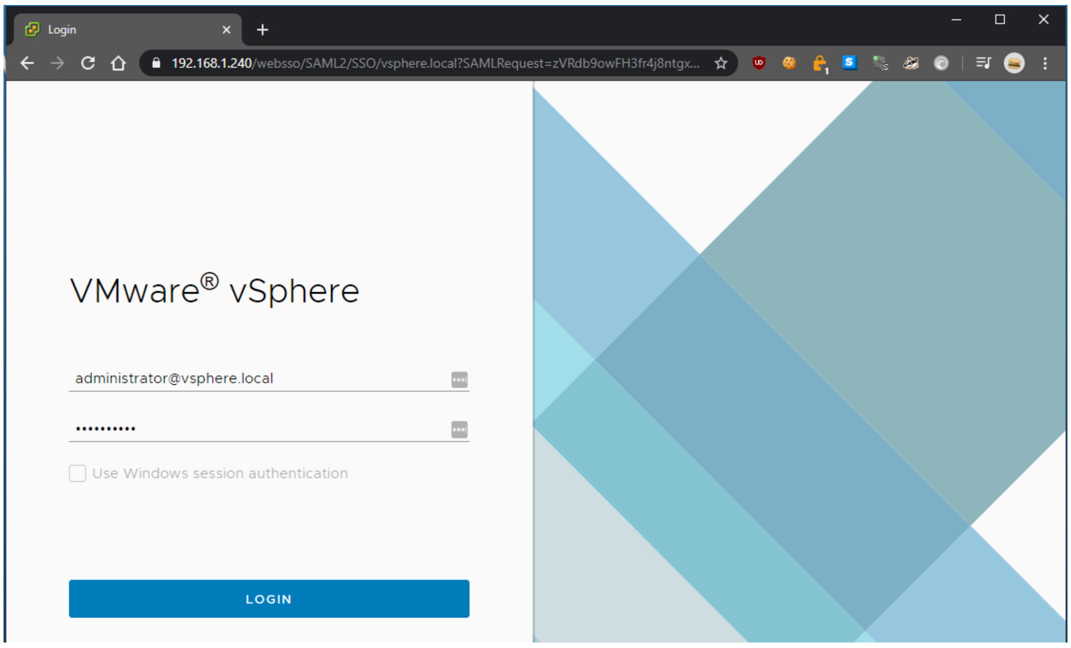
vSphere Web Services SDK

Learn about our latest SDKs, Toolkits, and APIs for managing VMware ESXi and VMware vCenter. Get sample code, reference documentation, participate in our Forum Discussions, and view our latest Sessions and Webinars.

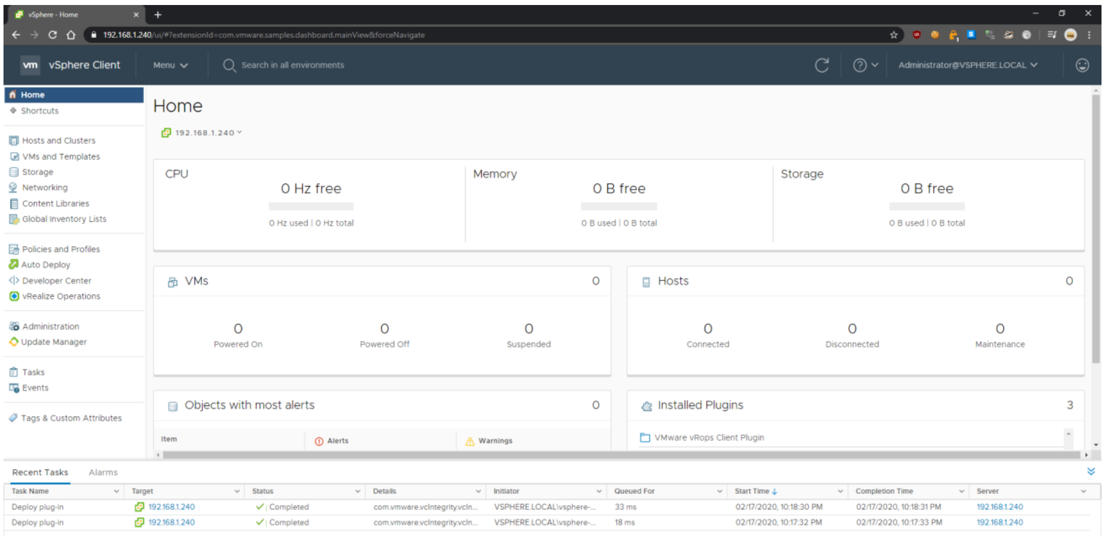
Learn more about the Web Services SDK
Browse objects managed by vSphere
Browse vSphere REST APIs
Download trusted root CA certificates

We can see that there is a section dedicated to the REST API.

We launch the vSphere client and connect with the account created previously.


Login
192.168.1.240/websso/SAML2/SSO/vsphere.local?SAMLRequest=zVRdb9owFH3fr4j8ntgx...
VMware® vSphere
administrator@vsphere.local
••••••••••
Use Windows session authentication
LOGIN

So that's the vCenter dashboard.

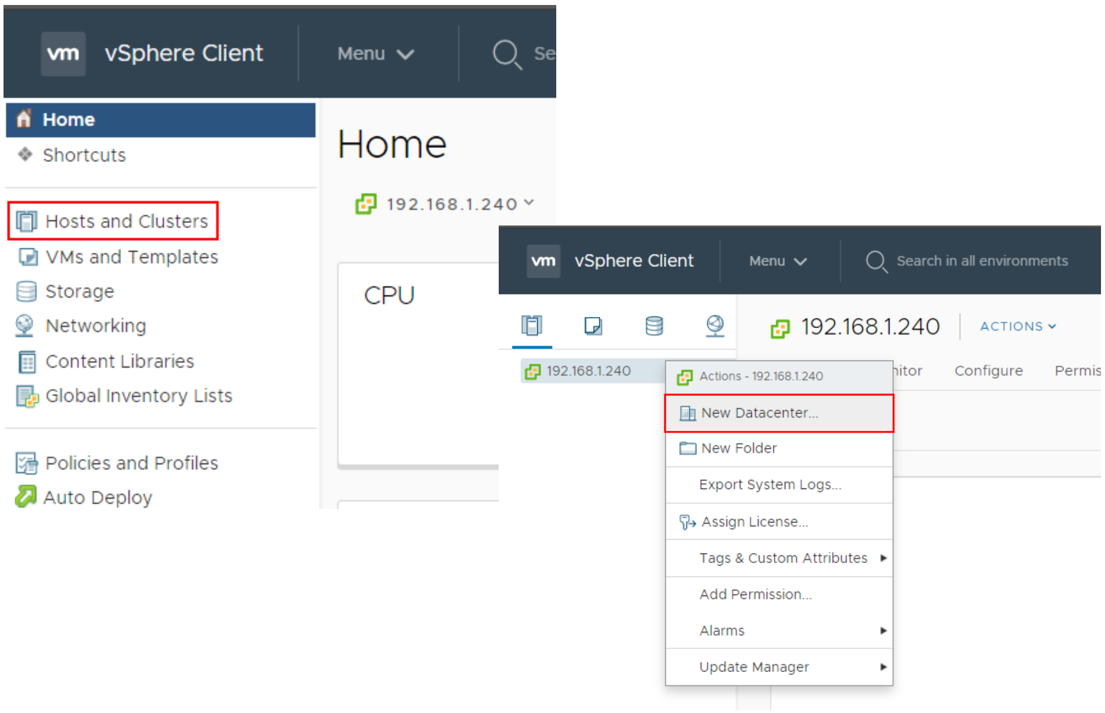


Task Name	Target	Status	Details	Initiator	Queued For	Start Time	Completion Time	Server
Deploy plug-in	192.168.1.240	✓ Completed	com.vmware.vcIntegrity.vcIn...	VSPHERE.LOCAL\vsphere-...	33 ms	02/17/2020, 10:18:30 PM	02/17/2020, 10:18:31 PM	192.168.1.240
Deploy plug-in	192.168.1.240	✓ Completed	com.vmware.vcIntegrity.vcIn...	VSPHERE.LOCAL\vsphere-...	18 ms	02/17/2020, 10:17:32 PM	02/17/2020, 10:17:33 PM	192.168.1.240

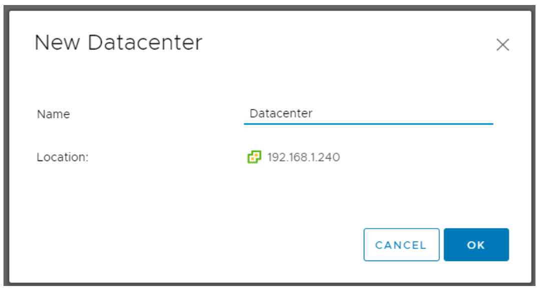
Hosts and clusters

We go to the "Hosts and Clusters" tab and then create a new Datacenter.

This will make it easier, if multiple servers are associated with the vCenter, for administrative tasks. This also allows you to manage the various ESXs only through the vCenter.



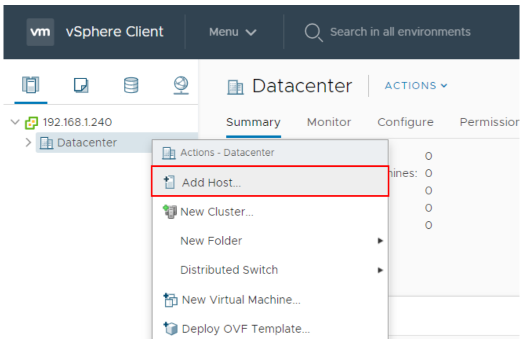
After giving it a name, we add our previously installed ESX to it.



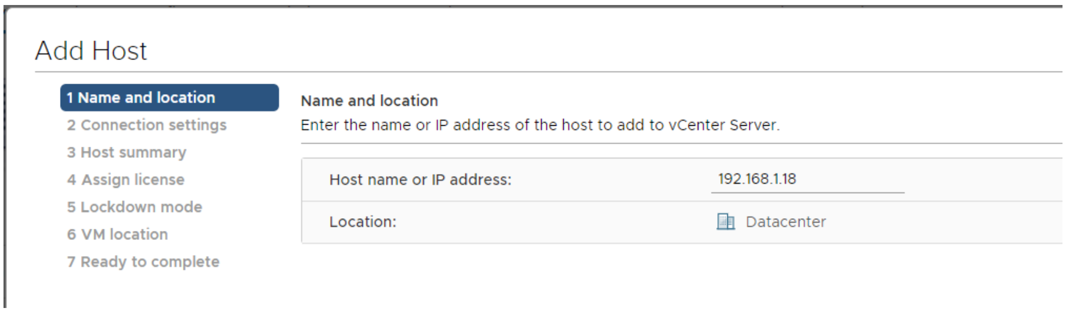
New Datacenter

Name: Datacenter

Location: 192.168.1.240



It is given the IP and the root password.

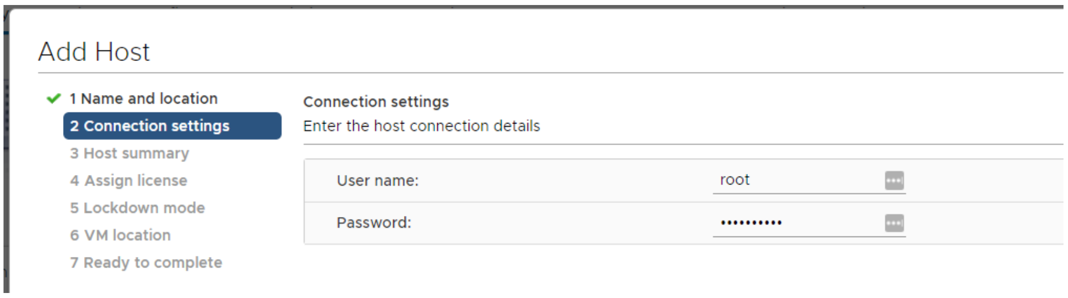


Add Host

Name and location
Enter the name or IP address of the host to add to vCenter Server.

Host name or IP address: 192.168.1.18

Location: Datacenter

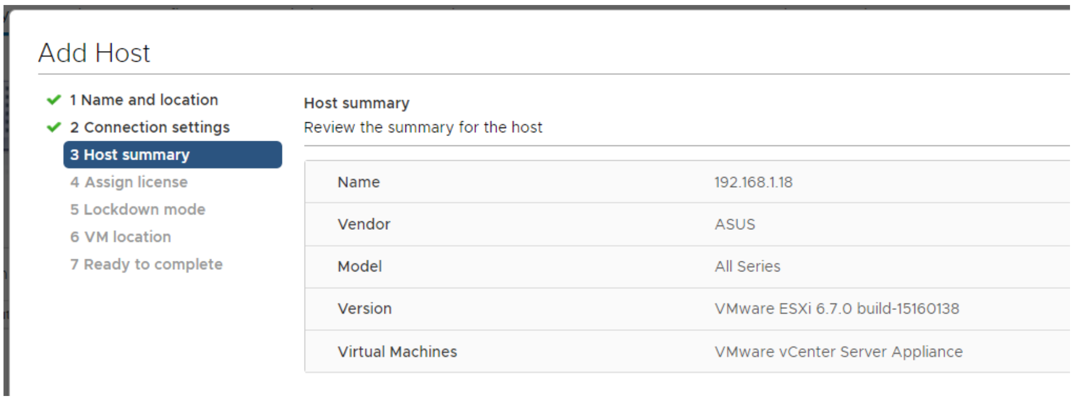


Add Host

Connection settings
Enter the host connection details

User name: root

Password: ••••••••••

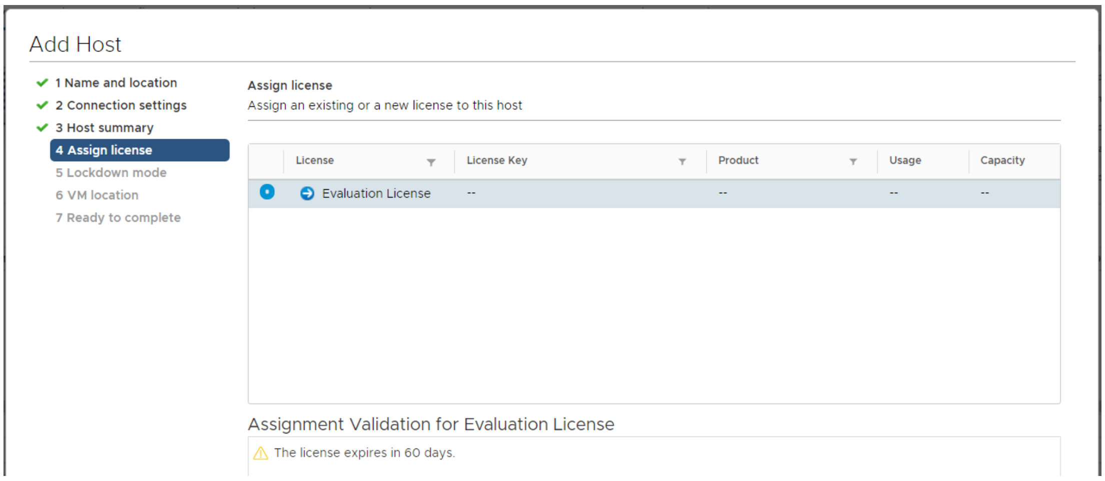


Add Host

Host summary
Review the summary for the host

Name	192.168.1.18
Vendor	ASUS
Model	All Series
Version	VMware ESXi 6.7.0 build-15160138
Virtual Machines	VMware vCenter Server Appliance

In the case of a paid license, it is at this stage that it is indicated.



Add Host

- ✓ 1 Name and location
- ✓ 2 Connection settings
- ✓ 3 Host summary
- 4 Assign license
- 5 Lockdown mode
- 6 VM location
- 7 Ready to complete

Assign license

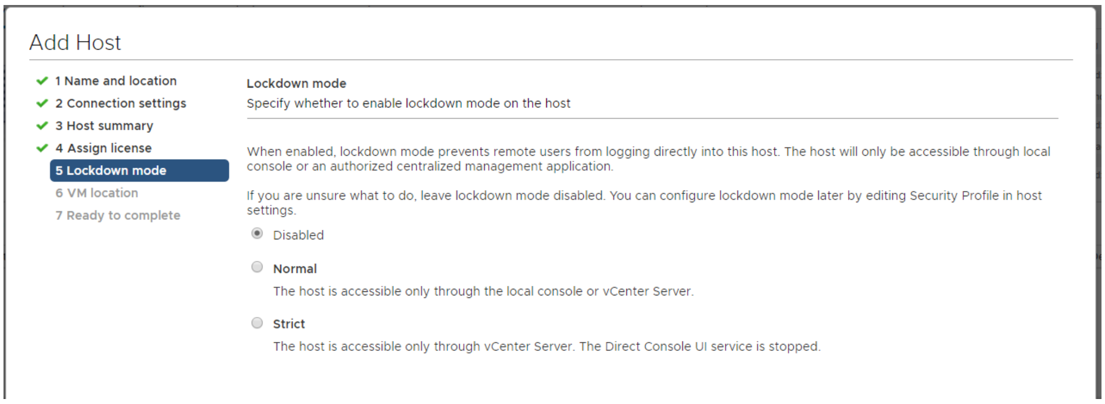
Assign an existing or a new license to this host

	License	License Key	Product	Usage	Capacity
●	Evaluation License	--	--	--	--

Assignment Validation for Evaluation License

⚠ The license expires in 60 days.

We must then define the lockdown mode. A summary table is below. In our case, we will choose the disabled mode.



Add Host

- ✓ 1 Name and location
- ✓ 2 Connection settings
- ✓ 3 Host summary
- ✓ 4 Assign license
- 5 Lockdown mode
- 6 VM location
- 7 Ready to complete

Lockdown mode

Specify whether to enable lockdown mode on the host

When enabled, lockdown mode prevents remote users from logging directly into this host. The host will only be accessible through local console or an authorized centralized management application.

If you are unsure what to do, leave lockdown mode disabled. You can configure lockdown mode later by editing Security Profile in host settings.

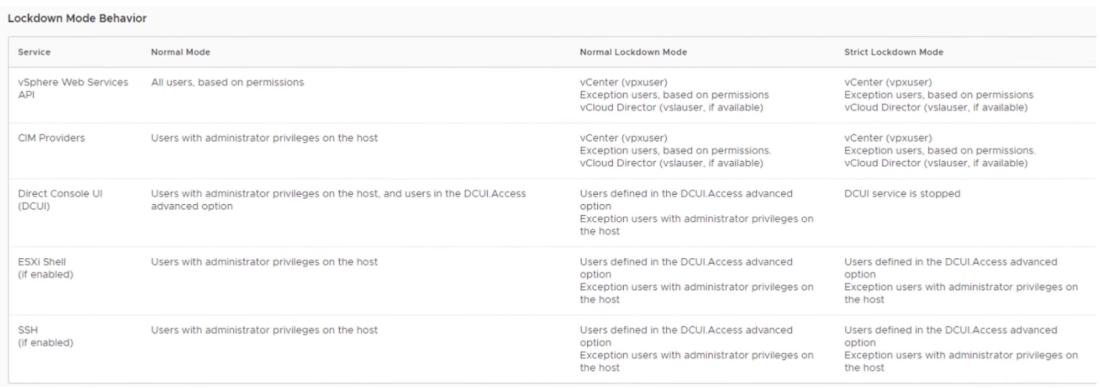
[✓ Disabled]

Normal

The host is accessible only through the local console or vCenter Server.

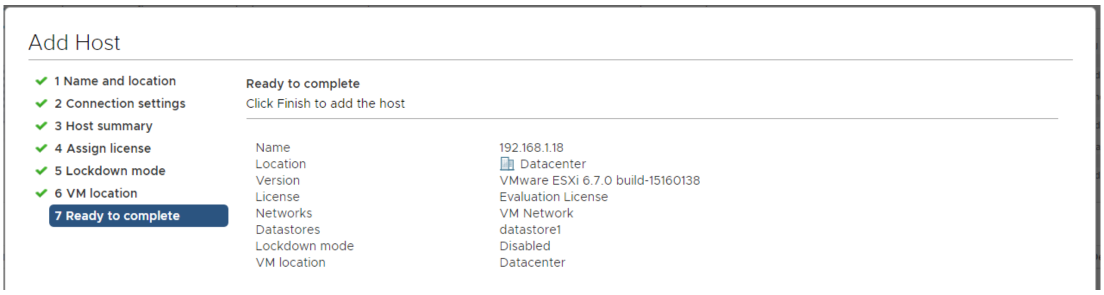
Strict

The host is accessible only through vCenter Server. The Direct Console UI service is stopped.



Lockdown Mode Behavior

Service	Normal Mode	Normal Lockdown Mode	Strict Lockdown Mode
vSphere Web Services API	All users, based on permissions	vCenter (vpxuser) Exception users, based on permissions vCloud Director (vslauser, if available)	vCenter (vpxuser) Exception users, based on permissions vCloud Director (vslauser, if available)
CIM Providers	Users with administrator privileges on the host	vCenter (vpxuser) Exception users, based on permissions. vCloud Director (vslauser, if available)	vCenter (vpxuser) Exception users, based on permissions. vCloud Director (vslauser, if available)
Direct Console UI (DCUI)	Users with administrator privileges on the host, and users in the DCUI.Access advanced option	Users defined in the DCUI.Access advanced option Exception users with administrator privileges on the host	DCUI service is stopped
ESXi Shell (if enabled)	Users with administrator privileges on the host	Users defined in the DCUI.Access advanced option Exception users with administrator privileges on the host	Users defined in the DCUI.Access advanced option Exception users with administrator privileges on the host
SSH (if enabled)	Users with administrator privileges on the host	Users defined in the DCUI.Access advanced option Exception users with administrator privileges on the host	Users defined in the DCUI.Access advanced option Exception users with administrator privileges on the host



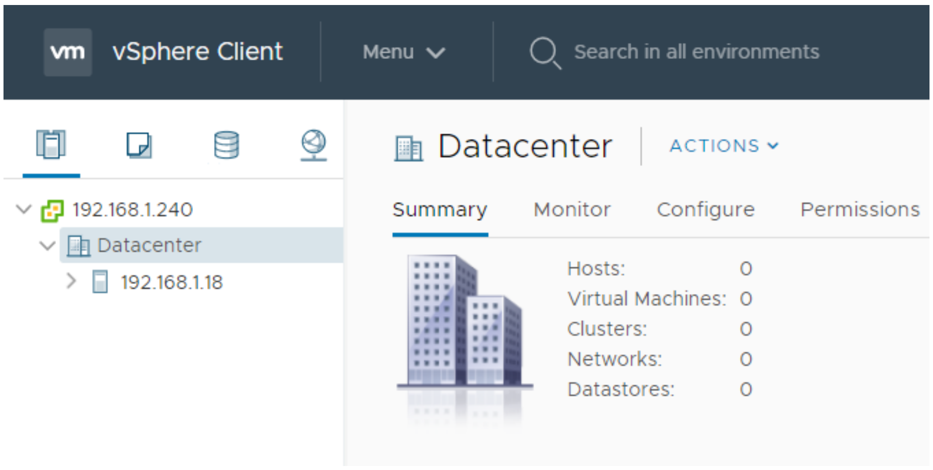
Add Host

- ✓ 1 Name and location
- ✓ 2 Connection settings
- ✓ 3 Host summary
- ✓ 4 Assign license
- ✓ 5 Lockdown mode
- ✓ 6 VM location
- 7 Ready to complete

Ready to complete
Click Finish to add the host

Name	192.168.1.18
Location	Datacenter
Version	VMware ESXi 6.7.0 build-15160138
License	Evaluation License
Networks	VM Network
Datastores	datastore1
Lockdown mode	Disabled
VM location	Datacenter

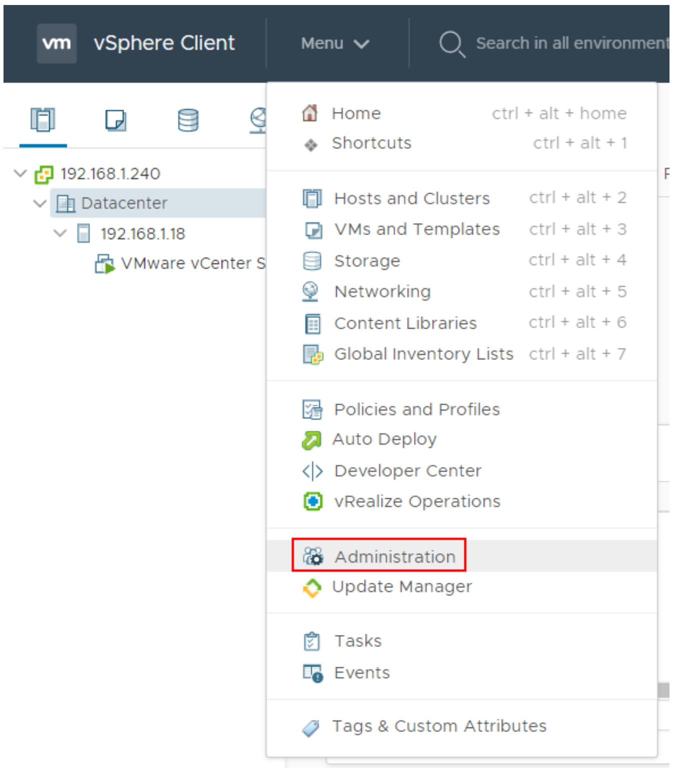
We can see that the host has been successfully added to the Datacenter.



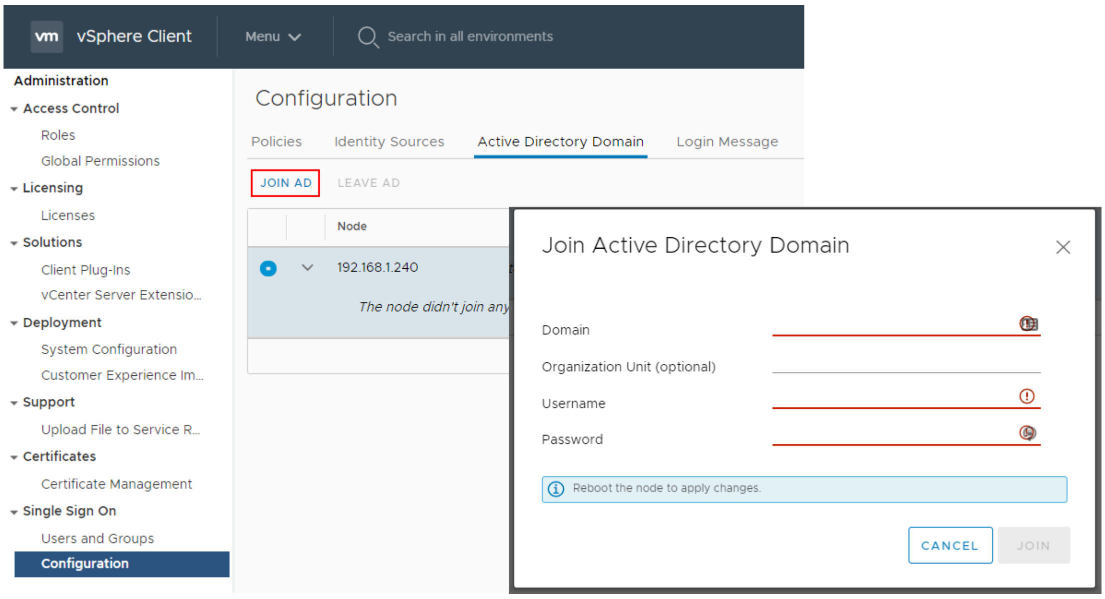
Hosts:	0
Virtual Machines:	0
Clusters:	0
Networks:	0
Datastores:	0

Access

We are now going to configure the access.

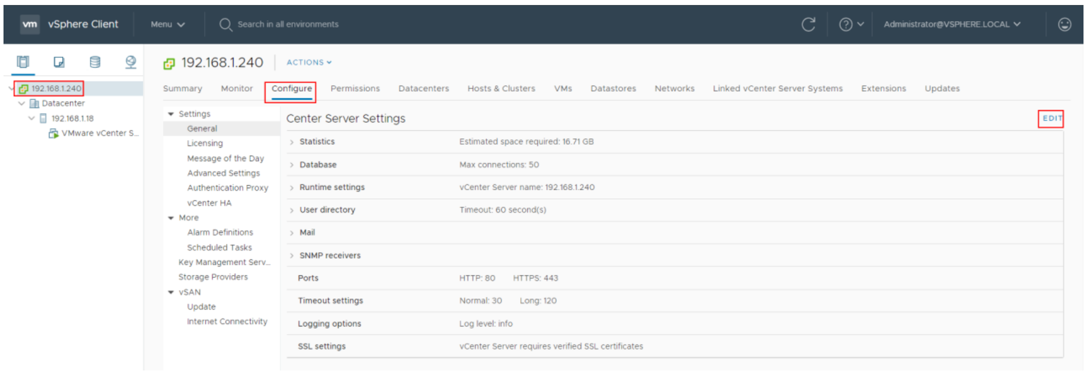


In the case of an existing Active Directory, this is where it should be joined:



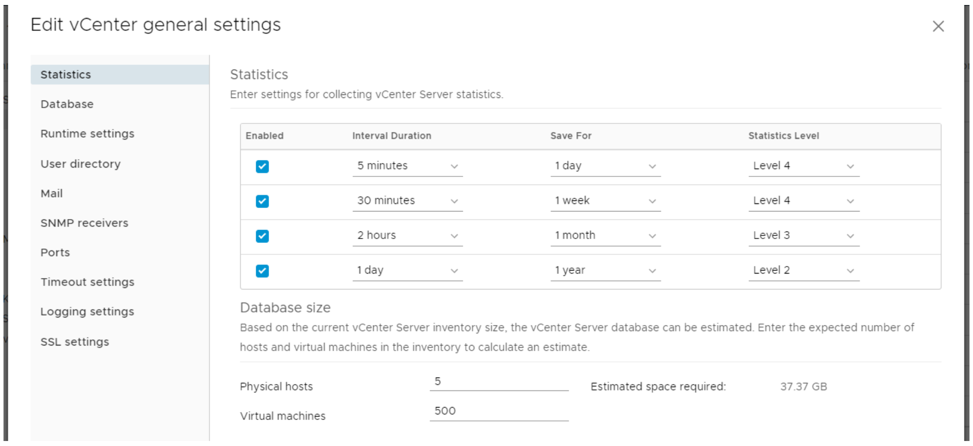
Statistics

We configure the different settings for collecting statistics from the vCenter Server.



Center Server Settings

Statistics	Estimated space required: 16.71 GB
Database	Max connections: 50
Runtime settings	vCenter Server name: 192.168.1.240
User directory	Timeout: 60 second(s)
Mail	
SNMP receivers	
Ports	HTTP: 80 HTTPS: 443
Timeout settings	Normal: 30 Long: 120
Logging options	Log level: info
SSL settings	vCenter Server requires verified SSL certificates



Edit vCenter general settings

Statistics

Enter settings for collecting vCenter Server statistics.

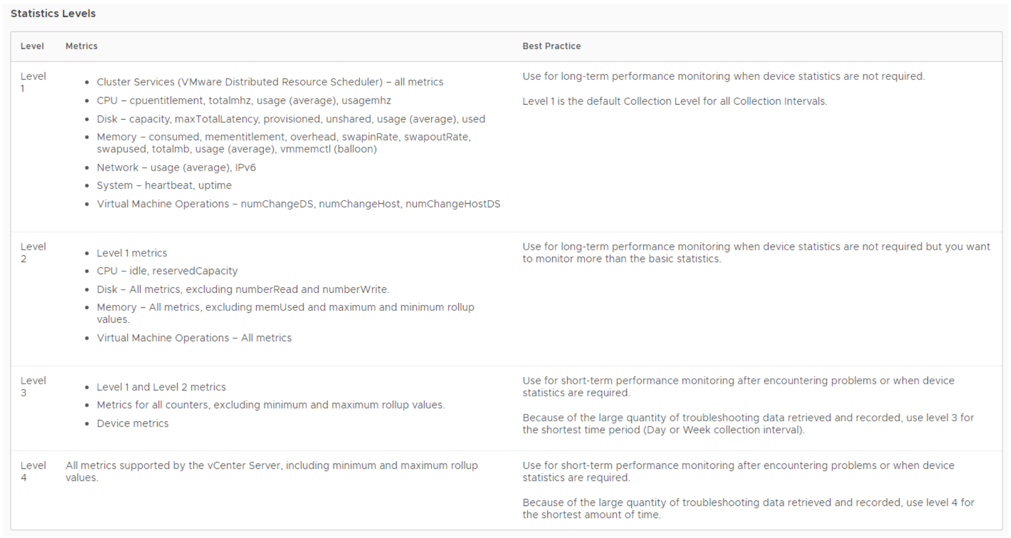
Enabled	Interval Duration	Save For	Statistics Level
✓	5 minutes	1 day	Level 4
✓	30 minutes	1 week	Level 4
✓	2 hours	1 month	Level 3
✓	1 day	1 year	Level 2

Database size

Based on the current vCenter Server inventory size, the vCenter Server database can be estimated. Enter the expected number of hosts and virtual machines in the inventory to calculate an estimate.

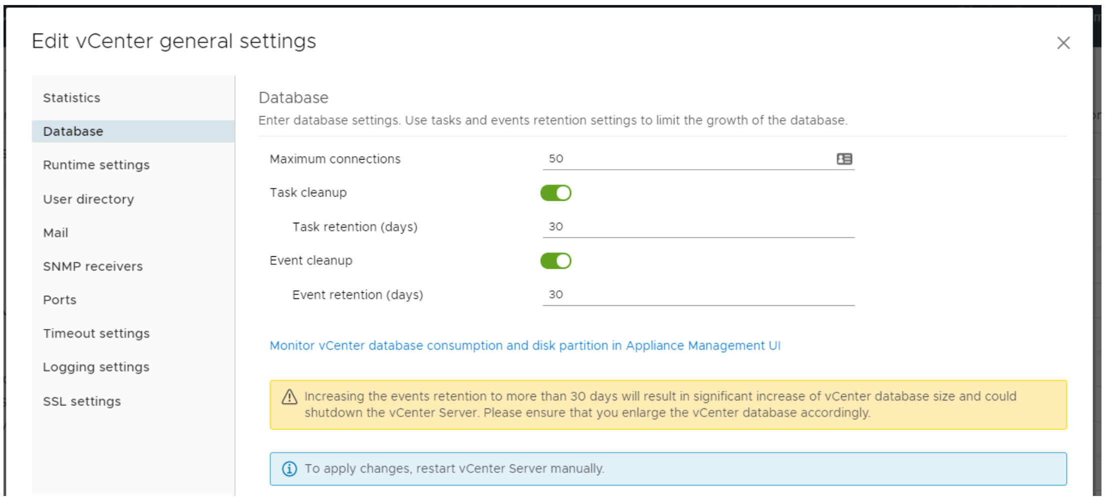
Physical hosts	5	Estimated space required:	37.37 GB
Virtual machines	500		

The full table of the different levels of statistics can be found below:



Statistics Levels

Level	Metrics	Best Practice
Level 1	• Cluster Services (VMware Distributed Resource Scheduler) – all metrics • CPU – cpuentitlement, totalmhz, usage (average), usagemhz • Disk – capacity, maxTotalLatency, provisioned, unshared, usage (average), used • Memory – consumed, mementitlement, overhead, swapinRate, swapoutRate, swapused, totalmb, usage (average), vmmemctl (balloon) • Network – usage (average), IPv6 • System – heartbeat, uptime • Virtual Machine Operations – numChangeDS, numChangeHost, numChangeHostDS	Use for long-term performance monitoring when device statistics are not required. Level 1 is the default Collection Level for all Collection Intervals.
Level 2	• Level 1 metrics • CPU – idle, reservedCapacity • Disk – All metrics, excluding numberRead and numberWrite. • Memory – All metrics, excluding memUsed and maximum and minimum rollup values. • Virtual Machine Operations – All metrics	Use for long-term performance monitoring when device statistics are not required but you want to monitor more than the basic statistics.
Level 3	• Level 1 and Level 2 metrics • Metrics for all counters, excluding minimum and maximum rollup values. • Device metrics	Use for short-term performance monitoring after encountering problems or when device statistics are required. Because of the large quantity of troubleshooting data retrieved and recorded, use level 3 for the shortest time period (Day or Week collection interval).
Level 4	All metrics supported by the vCenter Server, including minimum and maximum rollup values.	Use for short-term performance monitoring after encountering problems or when device statistics are required. Because of the large quantity of troubleshooting data retrieved and recorded, use level 4 for the shortest amount of time.



Edit vCenter general settings

Database

Enter database settings. Use tasks and events retention settings to limit the growth of the database.

Maximum connections	50
Task cleanup	On
Task retention (days)	30
Event cleanup	On
Event retention (days)	30

Monitor vCenter database consumption and disk partition in Appliance Management UI

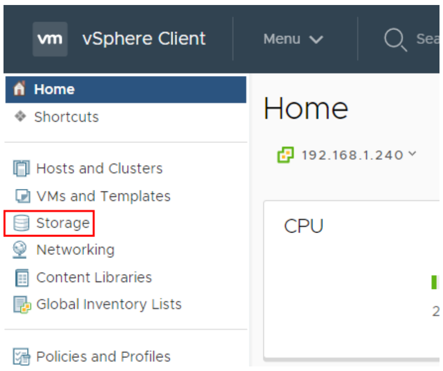
Increasing the events retention to more than 30 days will result in significant increase of vCenter database size and could shutdown the vCenter Server. Please ensure that you enlarge the vCenter database accordingly.

To apply changes, restart vCenter Server manually.

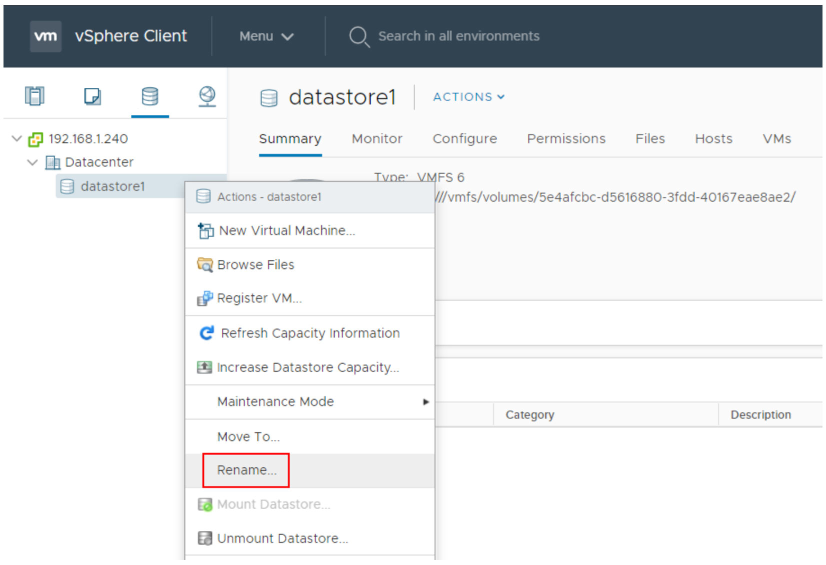
Administrative settings

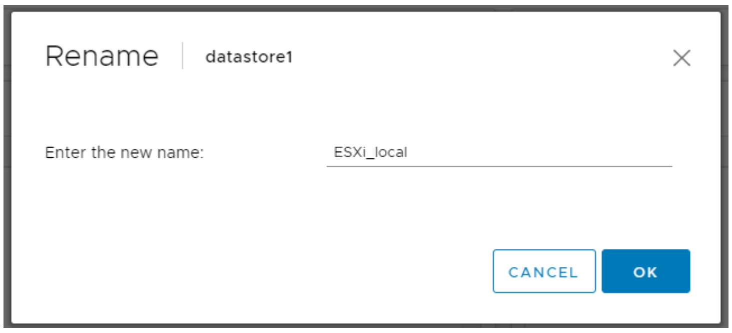
Here are some tweaks to make administration easier.

We go to the “storage” tab.



Here is the local ESXi storage. For easier administration, we rename “datastore1” by default to “ESXi_*Hostname*_LOCAL”

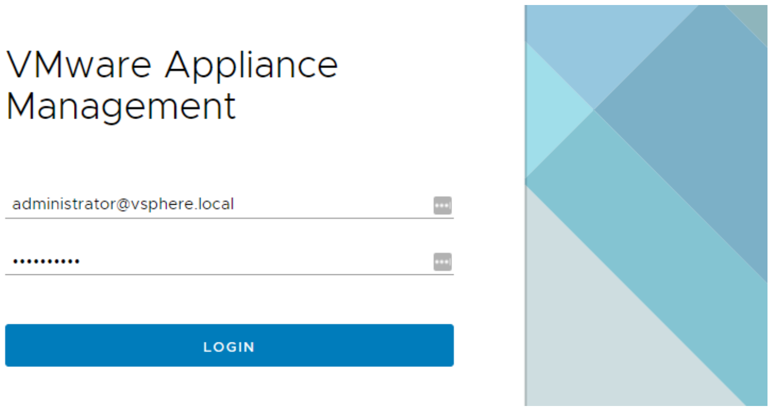




Enter the new name: ESXi_local

VMware Appliance Management

To set the time zone, we go to : https://[IP-VCENTER]:5480/login

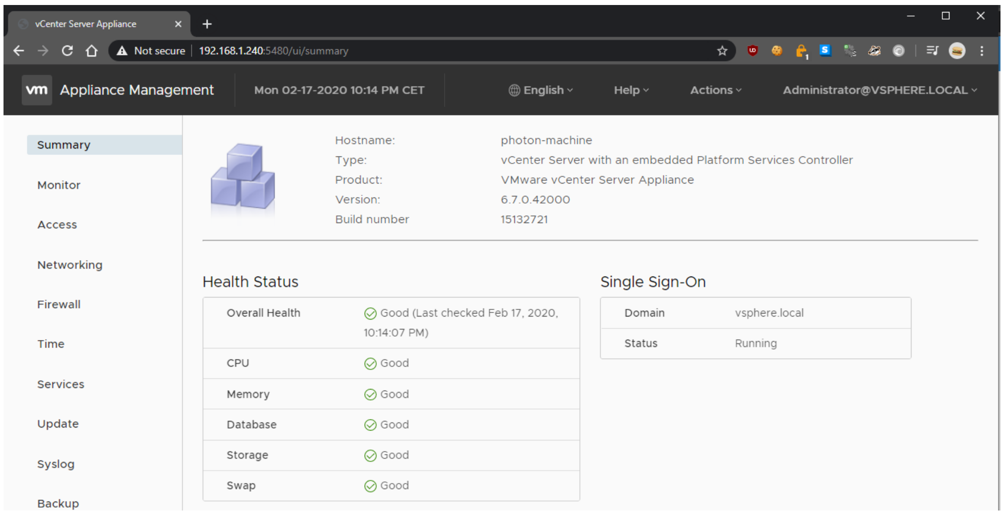


VMware Appliance Management

administrator@vsphere.local

••••••••••

LOGIN



vCenter Server Appliance

Not secure | 192.168.1.240:5480/ui/summary

Appliance Management | Mon 02-17-2020 10:14 PM CET | English | Help | Actions | Administrator@VSPHERE.LOCAL

Summary, Monitor, Access, Networking, Firewall, Time, Services, Update, Syslog, Backup

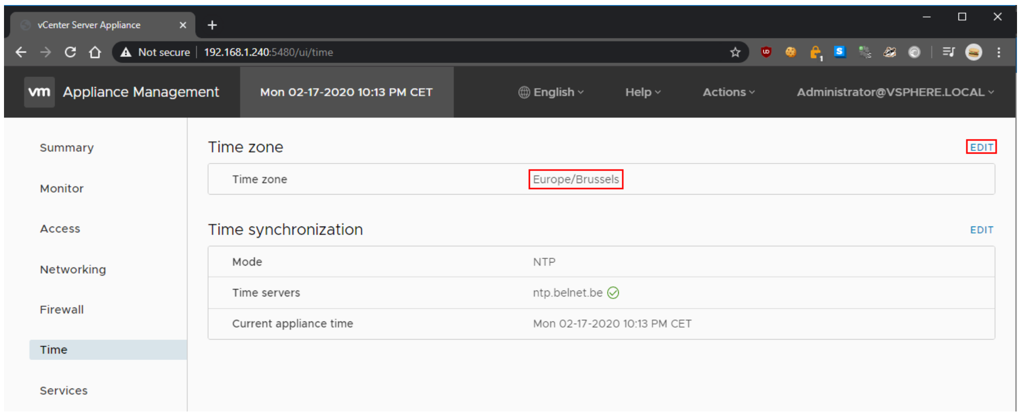
Hostname:	photon-machine
Type:	vCenter Server with an embedded Platform Services Controller
Product:	VMware vCenter Server Appliance
Version:	6.7.0.42000
Build number	15132721

Health Status

Overall Health	✓ Good (Last checked Feb 17, 2020, 10:14:07 PM)
CPU	✓ Good
Memory	✓ Good
Database	✓ Good
Storage	✓ Good
Swap	✓ Good

Single Sign-On

Domain	vsphere.local
Status	Running



vCenter Server Appliance

Not secure | 192.168.1.240:5480/ui/time

Appliance Management | Mon 02-17-2020 10:13 PM CET | English | Help | Actions | Administrator@VSPHERE.LOCAL

Summary, Monitor, Access, Networking, Firewall, Time, Services

Time zone — EDIT

Time zone	Europe/Brussels

Time synchronization — EDIT

Mode	NTP
Time servers	ntp.belnet.be ✓
Current appliance time	Mon 02-17-2020 10:13 PM CET

Appendix B

Requests documentation

This section will present the queries that were used when implementing the cyber range with the ESXi hypervisor.

The basis of these queries come from the template package available on the VMware website.

We will find for each request the three different frameworks proposed by Postman (cURL, HTTP Request2 and pecl http). For the requests whose answers were relevant, they have also been included.

B.1 Authentication

B.1.1 Login

Login to the specified vCenter and retrieve a session.
The base64 in the code the the base 64 encoding of the "username:password".
{IP_ADDRESS} is the IP of the vCenter.

PHP - cURL

```
<?php
$curl = curl_init();
curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/com/vmware/cis/session",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_HTTPHEADER => array(
    "Authorization: Basic YWRtaW5pc3RyYXRvckB2c3BoZXJlLmxvY2FsOlRpZ3JvdTAwNy4
      =”
  ),
));
$response = curl_exec($curl);
curl_close($curl);
echo $responses;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE

));
$request->setHeader(array(
  'Authorization' => 'Basic YWRtaW5pc3RyYXRvckB2c3BoZXJlLmxvY2FsOlRpZ3
      JvdTAwNy4='
));
$request->setBody('');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Authorization' => 'Basic YWRtaW5pc3RyYXRvckB2c3BoZXJlLmxvY2FsOlRpZ3
      JvdTAwNy4='
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Responses

```
200 OK
{
    "value": "31e7b314dba8d0e9c03351e9e7247d04"
}
```

```
401 Unauthorized
{
    "type": "com.vmware.vapi.std.errors.unauthenticated",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "Authentication required.",
                "id": "com.vmware.vapi.endpoint.method.authentication.
                    required"
            }
        ]
    }
}
```

B.1.2 Logout

Logout of the specified vCenter

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/com/vmware/cis/session",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "DELETE",
  CURLOPT_HTTPHEADER => array(
     "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session');
$request->setMethod(HTTP_Request2::METHOD_DELETE);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```php
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session');
$request->setRequestMethod('DELETE');
$body = new http\Message\Body;
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

B.1.3 Session information

Get current session info, check if we are still logged or not.

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/com/vmware/cis/session?~action=
      get",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session?~action=
    get');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/com/vmware/cis/session?~
    action=get');
$request->setRequestMethod('POST');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Responses
Response if session is active

```
200 OK
{
    "value": {
        "created_time": "2020-04-13T11:39:40.936Z",
        "last_accessed_time": "2020-04-13T11:43:12.461Z",
        "user": "root@localos"
    }
}
```

Response if the session is not active anymore

```
401 Unauthorized
{
    "type": "com.vmware.vapi.std.errors.unauthenticated",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "This method requires authentication.",
                "id": "vapi.method.authentication.required"
            }
        ]
    }
}
```

B.2 Information

B.2.1 Host

List all the hosts available with details such as its id, name, connection and power state.

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/host",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/host');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
```

```
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/host');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "host": "host-28",
            "name": "192.168.1.18",
            "connection_state": "CONNECTED",
            "power_state": "POWERED_ON"
        }
    ]
}
```

B.2.2 Datastore

List all the datastore available with details such as its id, name, type, free space and total capacity.

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/datastore",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/datastore');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/datastore');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "datastore": "datastore-29",
            "name": "ESXi_local",
            "type": "VMFS",
            "free_space": 678754779136,
            "capacity": 992137445376
        }
    ]
}
```

B.2.3 Folder

List all the folders present with details such as its id, name, and type.

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/folder",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/folder');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```php
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/folder');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "folder": "group-s24",
            "name": "datastore",
            "type": "DATASTORE"
        },
        {
            "folder": "group-v22",
            "name": "vm",
            "type": "VIRTUAL_MACHINE"
        },
        {
            "folder": "group-v32",
            "name": "Test",
            "type": "VIRTUAL_MACHINE"
        },
        {
            "folder": "group-v61",
            "name": "Discovered virtual machine",
            "type": "VIRTUAL_MACHINE"
        }
    ]
}
```

B.2.4 Network

List all the networks available and their IDs, name and type.

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/network",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/network');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```php
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/network');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "name": "Management Network",
            "type": "STANDARD_PORTGROUP",
            "network": "network-91"
        },
        {
            "name": "VM Network",
            "type": "STANDARD_PORTGROUP",
            "network": "network-30"
        },
        {
            "name": "DSwitch 1-VM Network",
            "type": "DISTRIBUTED_PORTGROUP",
            "network": "dvportgroup-90"
        },
        {
            "name": "DSwitch 1-Management Network",
            "type": "DISTRIBUTED_PORTGROUP",
            "network": "dvportgroup-88"
        }
    ]
}
```

B.2.5 version

Get the API version of the vCenter.

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/appliance/system/version",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/appliance/system/version');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/appliance/system/version')
    ;
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": {
        "summary": "Patch for VMware vCenter Server Appliance 6.7.0",
        "install_time": "2020-02-17T20:55:32 UTC",
        "product": "VMware vCenter Server Appliance",
        "build": "15132721",
        "releasedate": "December 5, 2019",
        "type": "vCenter Server with an embedded Platform Services Controller
            ",
        "version": "6.7.0.42000"
    }
}
```

B.3 Virtual machine

B.3.1 Get the list of VMs

This request will get the list of VMs with their details such as their name, ID, amount of RAM and CPU and the state of it.

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "memory_size_MiB": 4096,
            "vm": "vm-102",
            "name": "DebianTest",
            "power_state": "POWERED_OFF",
            "cpu_count": 2
        },
        {
            "memory_size_MiB": 10240,
            "vm": "vm-31",
            "name": "VMware vCenter Server Appliance",
            "power_state": "POWERED_ON",
            "cpu_count": 2
        }
    ]
}
```

B.3.2 Deploy a VM

This will create a DebianTest VM with 4GB of RAM, 2CPUs, a network interface, and the ISO attached to it.
For the parts like "datastore", "host", "folder", etc. they needed previous requests in order to have the correct "ID" (see B.2).
The complete list of guest OS can be found by following this link [1].

JSON file

```
{
  "spec": {
    "name": "DebianTest",
    "guest_OS": "DEBIAN_10_64",
    "placement": {
        "datastore": "datastore-29",
        "host": "host-28",
        "folder": "group-v61"
    },
    "memory": {
        "size_MiB": 4096,
        "hot_add_enabled": true
    },
    "floppies": [],
    "cpu": {
      "hot_remove_enabled": true,
      "count": 2,
      "cores_per_socket": 2,
      "hot_add_enabled": true
    },
    "cdroms": [
      {
        "type": "IDE",
        "start_connected": true,
        "backing": {
            "iso_file": "[ESXi_local] ISOs/debian-10.3.0-amd64-netinst.iso",
            "type": "ISO_FILE"
        }
      }
    ],
    "disks": [
      {
        "new_vmdk": {
          "name": "DebianTest",
          "capacity": 17179869184
        }
      }
    ],
    "boot_devices": [
      {
        "type": "CDROM"
      }
    ],
```

[1] https://vdc-download.vmware.com/vmwb-repository/dcr-public/1cd28284-3b72-4885-9e31-d1c6d9e26686/71ef7304-a6c9-43b3-a3cd-868b2c236c81/doc/operations/com/vmware/vcenter/vm.create-operation.html

```
    "boot": {
      "type": "DISK",
      "delay": 1,
      "retry_delay": 1,
      "retry": true
    },
    "nics": [
      {
        "type": "E1000E",
        "pci_slot_number": 0,
        "mac_type": "AUTOMATIC",
        "wake_on_lan_enabled": true,
        "start_connected": true,
        "allow_guest_control": true,
        "backing": {
          "type": "DISTRIBUTED_PORTGROUP",
          "network": "dvportgroup-90"
        }
      }
    ]
  }
}
```

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS} /rest/vcenter/vm",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS =>"{\r\n  \"spec\": {\r\n    \"name\": \"DebianTest\",\r
      \n    \"guest_OS\": \"DEBIAN_10_64\",\r\n    \"placement\": {\r\n
        \"datastore\": \"datastore-29\",\r\n      \"host\": \"host-28\",\r\
      n      \"folder\": \"group-v61\"\r\n    },\r\n    \"memory\": {\r\n
          \"size_MiB\": 4096,\r\n      \"hot_add_enabled\": true\r\n
      },\r\n    \"floppies\": [],\r\n    \"cpu\": {\r\n      \"
      hot_remove_enabled\": true,\r\n      \"count\": 2,\r\n      \"
      cores_per_socket\": 2,\r\n      \"hot_add_enabled\": true\r\n    },\r\n
          \"cdroms\": [\r\n      {\r\n        \"type\": \"IDE\",\r\n
      \"start_connected\": true,\r\n        \"backing\": {\r\n          \"
      iso_file\": \"[ESXi_local] ISOs/debian-10.3.0-amd64-netinst.iso\",\r\n
                \"type\": \"ISO_FILE\"\r\n        }\r\n      }\r\n    ],\r\n
          \"disks\": [\r\n      {\r\n        \"new_vmdk\": {\r\n          \"
      name\": \"DebianTest\",\r\n          \"capacity\": 17179869184\r\n
         }\r\n      }\r\n    ],\r\n    \"boot_devices\": [\r\n      {\r\n
          \"type\": \"CDROM\"\r\n      }\r\n    ],\r\n    \"boot\": {\r\n
        \"type\": \"DISK\",\r\n      \"delay\": 1,\r\n      \"retry_delay\":
      1,\r\n      \"retry\": true\r\n    },    \r\n    \"nics\": [\r\n
      {\r\n        \"type\": \"E1000E\",\r\n        \"pci_slot_number\": 0,\r
      \n        \"mac_type\": \"AUTOMATIC\",\r\n        \"wake_on_lan_enabled
      \": true,\r\n        \"start_connected\": true,\r\n        \"
```

```
        allow_guest_control\": true,\r\n           \"backing\": {\r\n              \"
        type\": \"DISTRIBUTED_PORTGROUP\",\r\n              \"network\": \"
        dvportgroup-90\"\r\n           }\r\n        }\r\n     ]\r\n  }\r\n}\r\n",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{
\n  "spec": {
\n    "name": "DebianTest",
\n    "guest_OS": "DEBIAN_10_64",
\n    "placement": {
\n        "datastore": "datastore-29",
\n        "host": "host-28",
\n        "folder": "group-v61"
\n    },
\n    "memory": {
\n        "size_MiB": 4096,
\n        "hot_add_enabled": true
\n    },
\n    "floppies": [],
\n    "cpu": {
\n      "hot_remove_enabled": true,
\n      "count": 2,
\n      "cores_per_socket": 2,
\n      "hot_add_enabled": true
\n    },
\n    "cdroms": [
\n      {
\n        "type": "IDE",
\n        "start_connected": true,
\n        "backing": {
\n            "iso_file": "[ESXi_local] ISOs/debian-10.3.0-amd64-netinst.iso
    ",
\n            "type": "ISO_FILE"
\n        }
\n        }
\n      ],
\n      "disks": [
\n        {
\n          "new_vmdk": {
\n            "name": "DebianTest",
\n            "capacity": 17179869184
\n          }
\n        }
\n      ],
\n      "boot_devices": [
\n        {
\n          "type": "CDROM"
\n        }
\n      ],
\n      "boot": {
\n        "type": "DISK",
\n        "delay": 1,
\n        "retry_delay": 1,
\n        "retry": true
\n      },
\n      "nics": [
\n        {
\n          "type": "E1000E",
\n          "pci_slot_number": 0,
\n          "mac_type": "AUTOMATIC",
\n          "wake_on_lan_enabled": true,
\n          "start_connected": true,
\n          "allow_guest_control": true,
\n          "backing": {
\n            "type": "DISTRIBUTED_PORTGROUP",
\n            "network": "dvportgroup-90"
\n          }
\n        }
\n      ]
\n    }
\n}
\n');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm');
$request->setRequestMethod('POST');
```

```
$body = new http\Message\Body;
$body->append('{
  "spec": {
    "name": "DebianTest",
    "guest_OS": "DEBIAN_10_64",
    "placement": {
        "datastore": "datastore-29",
        "host": "host-28",
        "folder": "group-v61"
    },
    "memory": {
        "size_MiB": 4096,
        "hot_add_enabled": true
    },
    "floppies": [],
    "cpu": {
      "hot_remove_enabled": true,
      "count": 2,
      "cores_per_socket": 2,
      "hot_add_enabled": true
    },
    "cdroms": [
      {
        "type": "IDE",
        "start_connected": true,
        "backing": {
            "iso_file": "[ESXi_local] ISOs/debian-10.3.0-amd64-netinst.iso",
            "type": "ISO_FILE"
        }
      }
    ],
    "disks": [
      {
        "new_vmdk": {
          "name": "DebianTest",
          "capacity": 17179869184
        }
      }
    ],
    "boot_devices": [
      {
        "type": "CDROM"
      }
    ],
    "boot": {
      "type": "DISK",
      "delay": 1,
      "retry_delay": 1,
      "retry": true
    },
    "nics": [
      {
        "type": "E1000E",
        "pci_slot_number": 0,
        "mac_type": "AUTOMATIC",
        "wake_on_lan_enabled": true,
        "start_connected": true,
        "allow_guest_control": true,
        "backing": {
          "type": "DISTRIBUTED_PORTGROUP",
          "network": "dvportgroup-90"
        }
      }
    ]
  }
}
');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": "vm-121"
}
```

B.3.3 Deploy a VM

Deploy a virtual machine from an OVF file. It needs a JSON file detailing where the VM will be stored.

JSON file

```
{
    "deployment_spec": {
        "accept_all_EULA": true,
        "default_datastore_id": "datastore-14"
    },
    "target": {
        "folder_id": "group-v7",
        "host_id": "host-10",
        "resource_pool_id": "resgroup-9"
    }
}
```

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/com/vmware/vcenter/ovf/library-
      item/id:{ovf_library_item_id}?~action=deploy",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS =>"{\r\n    \"deployment_spec\": {\r\n        \"
      accept_all_EULA\": true,\r\n        \"default_datastore_id\": \"
      datastore-29\"\r\n    },\r\n    \"target\": {\r\n        \"folder_id\":
       \"group-v7\",\r\n        \"host_id\": \"host-28\",\r\n        \"
      resource_pool_id\": \"resgroup-9\"\r\n    }\r\n}",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/com/vmware/vcenter/ovf/library-
    item/id:{ovf_library_item_id}?~action=deploy');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{
\n    "deployment_spec": {
\n        "accept_all_EULA": true,
\n        "default_datastore_id": "datastore-29"
\n    },
\n    "target": {
\n        "folder_id": "group-v7",
\n        "host_id": "host-28",
\n        "resource_pool_id": "resgroup-9"
\n    }
\n}');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/com/vmware/vcenter/ovf/
    library-item/id:{ovf_library_item_id}?~action=deploy');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
    "deployment_spec": {
        "accept_all_EULA": true,
        "default_datastore_id": "datastore-29"
    },
    "target": {
        "folder_id": "group-v7",
        "host_id": "host-28",
        "resource_pool_id": "resgroup-9"
    }
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": "vm-126"
}
```

B.3.4 Delete a VM

Delete a VM by specifying its id (ex: vm-123).

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "DELETE",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://192.168.1.240/rest/vcenter/vm/{VM_ID}');
$request->setMethod(HTTP_Request2::METHOD_DELETE);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}');
$request->setRequestMethod('DELETE');
$body = new http\Message\Body;
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Responses

```
200 OK
```

```
400 Bad Request
{
    "type": "com.vmware.vapi.std.errors.not_allowed_in_current_state",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "The attempted operation cannot be
                    performed in the current state (Powered on).",
                "id": "vmsg.InvalidPowerState.summary"
            }
        ]
    }
}
```

```
404 Not Found
{
    "type": "com.vmware.vapi.std.errors.not_found",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "The object 'vim.VirtualMachine:vm-122'
                    has already been deleted or has not been completely
                    created",
                "id": "vmsg.ManagedObjectNotFound.summary"
            }
        ]
    }
}
```

B.3.5 Search a VM with its name

Retrieve the information such as the amount of RAM, the ID, the power state or the number of processors of one specific VM. The request needs the VM name (ex: ”DebianTest”).

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm?filter.names.1={
      VM_NAME}",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm?filter.names.1={
    VM_NAME}');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm?filter.names.1
    ={VM_NAME}');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
{
    "value": [
        {
            "memory_size_MiB": 4096,
            "vm": "vm-102",
            "name": "DebianTest",
            "power_state": "POWERED_OFF",
            "cpu_count": 2
        }
    ]
}
```

B.3.6 State of the VM

B.3.6.1 Get the state of the VM

Retrieve the state of the VM. The request needs the VM ID (ex: vm-122).

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
```

```
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power')
    ;
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": {
        "clean_power_off": true,
        "state": "POWERED_OFF"
    }
}
```

B.3.6.2 Change the state of the VM

Change the state of a VM by specifying an ACTION:

- Power off: stop
- Power on: start
- Suspend: suspend
- Reset: reset

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power/{ACTION}
      ",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power/{ACTION}
    ');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```php
<?php
$client = new http\Client;
```

```
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/power/{
    ACTION}');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Responses

```
200 OK
```

```
400 Bad Request
{
    "type": "com.vmware.vapi.std.errors.already_in_desired_state",
    "value": {
        "messages": [
            {
                "args": [],
                "default_message": "Virtual machine is already powered on.",
                "id": "com.vmware.api.vcenter.vm.power.already_powered_on"
            },
            {
                "args": [],
                "default_message": "The attempted operation cannot be
                    performed in the current state (Powered on).",
                "id": "vmsg.InvalidPowerState.summary"
            }
        ]
    }
}
```

B.3.7 CPU

B.3.7.1 Get CPU counts

Retrieve the amount of CPU counts of a VM by specifying its ID (ex: vm-123).

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
CURLOPT_URL => " https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/cpu",
CURLOPT_RETURNTRANSFER => true,
CURLOPT_ENCODING => "",
CURLOPT_MAXREDIRS => 10,
CURLOPT_TIMEOUT => 0,
CURLOPT_FOLLOWLOCATION => true,
CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
CURLOPT_CUSTOMREQUEST => "GET",
CURLOPT_HTTPHEADER => array(
"Cookie: vmware-api-session-id={session-id}"
),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl(' https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/cpu'
    );
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/cpu');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": {
        "hot_remove_enabled": false,
        "count": 2,
        "hot_add_enabled": false,
        "cores_per_socket": 2
    }
}
```

B.3.7.2 Change CPU counts

Change the amount of CPU counts of a specific VM using its id (ex: vm-123).

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/cpu"
      ,
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "PATCH",
  CURLOPT_POSTFIELDS =>"{\n  \"spec\": {\n    \"count\": 2,\n    \"
      hot_remove_enabled\": false,\n    \"cores_per_socket\": 1,\n    \"
      hot_add_enabled\": false\n  }\n}",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl(' https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/cpu'
    );
$request->setMethod('PATCH');
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{\n  "spec": {\n    "count": 2,\n    "hot_remove_enabled":
     false,\n    "cores_per_socket": 1,\n    "hot_add_enabled": false\n  }\n}
    ');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl(' https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/cpu');
```

```
$request->setRequestMethod('PATCH');
$body = new http\Message\Body;
$body->append('{
  "spec": {
    "count": 2,
    "hot_remove_enabled": false,
    "cores_per_socket": 1,
    "hot_add_enabled": false
  }
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

B.3.8 Memory

B.3.8.1 Get the memory amount

Retrieve the amount of memory of a VM by specifying its ID (ex: vm-123).

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      memory",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl(' https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    memory);
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/memory');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": {
        "size_MiB": 4096,
        "hot_add_enabled": true
    }
}
```

B.3.8.2 Change the memory amount

Change the amount of memory count of a specific VM using its id (ex: vm-123).

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      memory",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "PATCH",
  CURLOPT_POSTFIELDS =>"{\n  \"spec\": {\n    \"size_MiB\": 2048\n  }\n}",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl(' https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    memory');
$request->setMethod('PATCH');
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{\n  "spec": {\n    "size_MiB": 2048\n  }\n}');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/memory');
$request->setRequestMethod('PATCH');
$body = new http\Message\Body;
$body->append('{
  "spec": {
    "size_MiB": 2048
  }
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

B.3.9 Network

B.3.9.1 Get network information

Retrieve all the IDs of network cards attached to a VM by specifying its ID (ex: vm-123).

PHP - cURL

```php
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      ethernet",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl(https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    ethernet');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/ethernet');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": [
        {
            "nic": "4000"
        },
        {
            "nic": "4001"
        },
        {
            "nic": "4002"
        },
        {
            "nic": "4003"
        }
    ]
}
```

B.3.9.2 Get network details

Retrieve the details of a specific NIC ID (retrieved in the request before) of a VM by specifying its ID (ex: vm-123).

PHP - cURL

```
<?php
```

```

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      ethernet/{NIC_ID}",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "GET",
  CURLOPT_HTTPHEADER => array(
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```php
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    ethernet/{NIC_ID}');
$request->setMethod(HTTP_Request2::METHOD_GET);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```php
<?php
$client = new http\Client;
$request = new http\Client\Request;
```

```
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/ethernet/{NIC_ID}');
$request->setRequestMethod('GET');
$request->setOptions(array());
$request->setHeaders(array(
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

PHP - cURL

```
{
    "value": {
        "start_connected": true,
        "pci_slot_number": 0,
        "backing": {
            "connection_cookie": 677560597,
            "distributed_switch_uuid": "50 07 e6 b7 98 b5 ff 60-4c a7 96 06 4
                4 32 bd 51",
            "distributed_port": "20",
            "type": "DISTRIBUTED_PORTGROUP",
            "network": "dvportgroup-90"
        },
        "mac_address": "00:50:56:87:3e:78",
        "mac_type": "ASSIGNED",
        "allow_guest_control": true,
        "wake_on_lan_enabled": true,
        "label": "Network adapter 1",
        "state": "NOT_CONNECTED",
        "type": "E1000E"
    }
}
```

B.3.9.3 Add network card

Add an ethernet card to a VM by specifying its ID (ex: vm-123). We need to specify the type and the network id (can be found in B.2.4)

PHP - cURL

```
<?php

$curl = curl_init();

curl_setopt_array($curl, array(
  CURLOPT_URL => " https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      ethernet",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "POST",
  CURLOPT_POSTFIELDS =>"{\r\n     \"spec\": {\r\n         \"type\": \"E1000\",\
     r\n         \"start_connected\": true,\r\n         \"backing\": {\r\n
             \"type\": \"{NETWORK_TYPE}\",\r\n             \"network\": \"{
     NETWORK_ID}\"\r\n         }\r\n     }\r\n}",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    ethernet');
$request->setMethod(HTTP_Request2::METHOD_POST);
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{
\n     "spec": {
\n         "type": "E1000",
\n         "start_connected": true,
\n         "backing": {
\n             "type": "{NETWORK_TYPE}",
\n             "network": "{NETWORK_ID}"
\n         }
\n     }
\n}');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
  echo $response->getBody();
}
else {
  echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
  $response->getReasonPhrase();
}
}
catch(HTTP_Request2_Exception $e) {
echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/ethernet');
$request->setRequestMethod('POST');
$body = new http\Message\Body;
$body->append('{
    "spec": {
        "type": "E1000",
        "start_connected": true,
        "backing": {
            "type": "{NETWORK_TYPE}",
            "network": "{NETWORK_ID}"
        }
    }
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Response

```
200 OK
{
    "value": "4003"
}
```

B.3.9.4 Change network card

Change the ethernet card attached to a specific VM ID (ex: vm-123).
By default, if a VM only have one network card, the NIC ID will be 4000. We need to specify the type and the network id (can be found in B.2.4).

PHP - cURL

```
<?php

$curl = curl_init();

```

```
curl_setopt_array($curl, array(
  CURLOPT_URL => "https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
      ethernet/{NIC_ID}",
  CURLOPT_RETURNTRANSFER => true,
  CURLOPT_ENCODING => "",
  CURLOPT_MAXREDIRS => 10,
  CURLOPT_TIMEOUT => 0,
  CURLOPT_FOLLOWLOCATION => true,
  CURLOPT_HTTP_VERSION => CURL_HTTP_VERSION_1_1,
  CURLOPT_CUSTOMREQUEST => "PATCH",
  CURLOPT_POSTFIELDS =>"{\r\n    \"spec\": {\r\n        \"backing\": {\r\n
            \"type\": \"{NETWORK_TYPE}\",\r\n            \"network\": \"{
      NETWORK_ID}\"\r\n        }\r\n    }\r\n}",
  CURLOPT_HTTPHEADER => array(
    "Content-Type: application/json",
    "Content-Type: text/plain",
    "Cookie: vmware-api-session-id={session-id}"
  ),
));

$response = curl_exec($curl);

curl_close($curl);
echo $response;
```

PHP - HTTP Request2

```
<?php
require_once 'HTTP/Request2.php';
$request = new HTTP_Request2();
$request->setUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/hardware/
    ethernet/{NIC_ID}');
$request->setMethod('PATCH');
$request->setConfig(array(
  'follow_redirects' => TRUE
));
$request->setHeader(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$request->setBody('{
\n    "spec": {
\n        "backing": {
\n            "type": "{NETWORK_TYPE}",
\n            "network": "{NETWORK_ID}"
\n        }
\n    }
\n}');
try {
  $response = $request->send();
  if ($response->getStatus() == 200) {
    echo $response->getBody();
  }
  else {
    echo 'Unexpected HTTP status: ' . $response->getStatus() . ' ' .
    $response->getReasonPhrase();
  }
}
```

```
catch(HTTP_Request2_Exception $e) {
  echo 'Error: ' . $e->getMessage();
}
```

PHP - pecl http

```
<?php
$client = new http\Client;
$request = new http\Client\Request;
$request->setRequestUrl('https://{IP_ADDRESS}/rest/vcenter/vm/{VM_ID}/
    hardware/ethernet/{NIC_ID}');
$request->setRequestMethod('PATCH');
$body = new http\Message\Body;
$body->append('{
    "spec": {
        "backing": {
            "type": "{NETWORK_TYPE}",
            "network": "{NETWORK_ID}"
        }
    }
}');
$request->setBody($body);
$request->setOptions(array());
$request->setHeaders(array(
  'Content-Type' => 'application/json',
  'Content-Type' => 'text/plain',
  'Cookie' => 'vmware-api-session-id={session-id}'
));
$client->enqueue($request)->send();
$response = $client->getResponse();
echo $response->getBody();
```

Appendix C

Diagrams

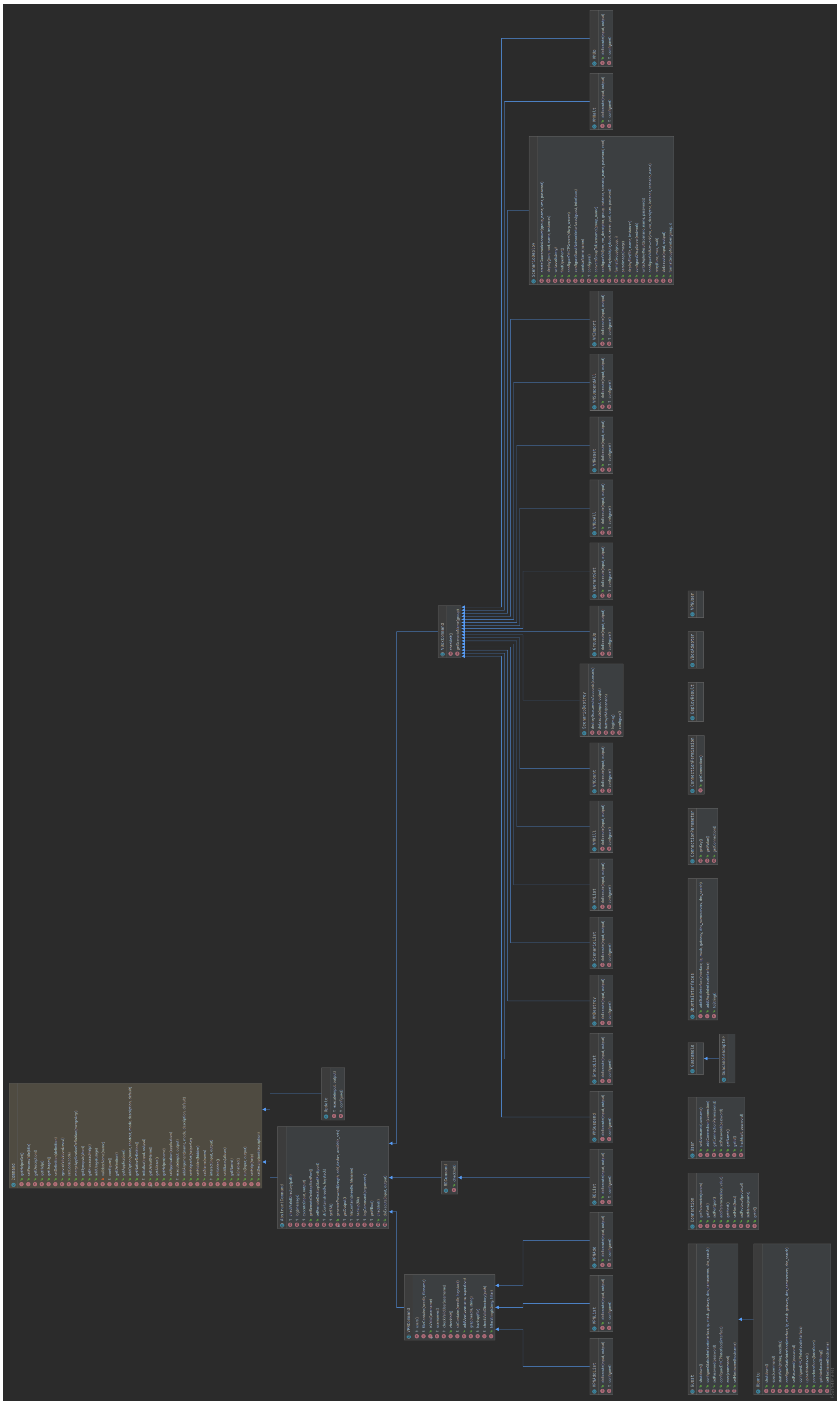


Figure C.1: PHP diagram before any modifications (with methods included)

Bibliography

[1] European Defence Agency. Common staff target for military cooperation on cyber ranges in the european union, last visited on July 19, 2020. https://www.eda.europa.eu/docs/default-source/procurement/annex-a---cyber-ranges-cst.pdf.

[2] Sarfraz Ahmed. Dependency injection in php, last visited on July 24, 2020. https://codeinphp.github.io/post/dependency-injection-in-php/.

[3] Airbus. Cyberrange, last visited on July 19, 2020. https://airbus-cyber-security.com/products-and-services/prevent/cyberrange/.

[4] Airbus. Cyberrange - integration and simulation platform of it/ot systems, last visited on July 19, 2020. https://airbus-cyber-security.com/wp-content/uploads/2018/03/CyberRange_Brochure_0907_EN-1.pdf.

[5] Palo Alto. Cyber defense training center and exercises, last visited on August 1, 2020. https://www.paloaltonetworks.com/solutions/initiatives/cyberrange-overview.

[6] Radhwan Y. Ameen and Asmaa Y. Hamo. Survey of server virtualization. *CoRR*, abs/1304.3557, 2013.

[7] Samdare B. When do we need interfaces in php?, last visited on July 24, 2020. https://www.geeksforgeeks.org/when-do-we-need-interfaces-in-php/.

[8] Stephen J. Bigelow. What's the difference between type 1 and type 2 hypervisors?, last visited on Augustus 8, 2019. https://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors.

[9] Jeff Daniels. Server virtualization architecture and implementation. *XRDS*, 16(1):8–12, September 2009.

[10] T. Debatty and W. Mees. Building a cyber range for training cyberdefense situation awareness. In *2019 International Conference on Military Communications and Information Systems (ICMCIS)*, pages 1–6, 2019.

[11] Joshua Eckroth, Kim Chen, Heyley Gatewood, and Brandon Belna. Alpaca: Building dynamic cyber ranges with procedurally-generated vulnerability lattices. In *Proceedings of the 2019 ACM Southeast Conference*, ACM SE '19, page 78–85, New York, NY, USA, 2019. Association for Computing Machinery.

[12] European Cyber Security Organisation (ECSO). Understanding cyber ranges: From hype to reality, last visited on July 18, 2020. https://ecs-org.eu/press-releases/understanding-cyber-ranges-from-hype-to-reality.

[13] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. Design patterns: Abstraction and reuse of object-oriented design. In Oscar M. Nierstrasz, editor, *ECOOP' 93 — Object-Oriented Programming*, pages 406–431, Berlin, Heidelberg, 1993. Springer Berlin Heidelberg.

[14] Waldemar Graniszewski and Adam Arciszewski. Performance analysis of selected hypervisors (virtual machine monitors - vmms). *International Journal of Electronics and Telecommunications*, 62, 08 2016.

[15] Red Hat. What is virtualization?, last visited on Augustus 9, 2019. https://www.redhat.com/en/topics/virtualization/what-is-virtualization.

[16] Kristijan Horvat. Dependency injection in practice, last visited on July 24, 2020. https://www.mono.hr/pdf/Dependency-Injection-in-practice-CodeCAMP.pdf.

[17] ixiacom. Cyber range: Improving network defense and security readiness, last visited on July 17, 2020. https://support.ixiacom.com/sites/default/files/resources/whitepaper/915-6729-01-cyber-range.pdf.

[18] Bhavya Karia. A quick intro to dependency injection: what it is, and when to use it, last visited on August 3, 2020. https://www.freecodecamp.org/news/a-quick-intro-to-dependency-injection-what-it-is-and-when-to-use-it-7578c84fa88f/.

[19] P. Lavin. *Object-Oriented PHP: Concepts, Techniques, and Code.* No Starch Press Series. No Starch Press, 2006.

[20] Dominik Liebler and contributors. Designpatternsphp, last visited on July 21, 2020. https://readthedocs.org/projects/designpatternsphp/downloads/pdf/latest/.

[21] Karissa Miller and Mahmoud Pegah. Virtualization: Virtually at the desktop. In *Proceedings of the 35th Annual ACM SIGUCCS Fall Conference*, SIGUCCS '07, pages 255–260, New York, NY, USA, 2007. ACM.

[22] R. Morabito, J. Kjällman, and M. Komu. Hypervisors vs. lightweight virtualization: A performance comparison. In *2015 IEEE International Conference on Cloud Engineering*, pages 386–393, 2015.

[23] N. M. Mosharaf Kabir Chowdhury and R. Boutaba. Network virtualization: state of the art and research challenges. *IEEE Communications Magazine*, 47(7):20–26, July 2009.

[24] Matthieu Napoli and contributors. Php di - documentation, last visited on July 24, 2020. https://php-di.org/doc/.

[25] J. Nie. A study on the application cost of server virtualisation. In *2013 Ninth International Conference on Computational Intelligence and Security*, pages 807–811, Dec 2013.

[26] Jordi Boggiano Nils Adermann and many community contributions. A dependency manager for php, last visited on July 29, 2020. https://getcomposer.org/.

[27] phoenixNAP. What is a hypervisor? types of hypervisors 1 & 2, last visited on Augustus 8, 2019. https://phoenixnap.com/kb/what-is-hypervisor-type-1-2.

[28] Michaux Pierre. Analyse et mise en place d'un simulateur et d'un hyperviseur incluant des scénarios d'entrainement destinés aux étudiants du master en cybersécurité, last consulted on July 15, 2020.

[29] HNS platform. Hns cyber range, last visited on July 19, 2020. https://www.hns-platform.com/cyberrange/.

[30] HNS platform. Hns online cyber range, last visited on July 19, 2020. https://www.hns-platform.com/wp-content/uploads/2018/06/plaquette_2018_online_EN_BD_1.0.pdf.

[31] Margaret Rouse. Data virtualization, last visited on Augustus 9, 2019. https://searchdatamanagement.techtarget.com/definition/data-virtualization.

[32] Margaret Rouse. Desktop virtualization, last visited on Augustus 9, 2019. https://searchvirtualdesktop.techtarget.com/definition/desktop-virtualization.

[33] Margaret Rouse. Network virtualization, last visited on Augustus 13, 2019. https://searchservervirtualization.techtarget.com/definition/network-virtualization.

[34] Margaret Rouse. Server sprawl, last visited on Augustus 13, 2019. https://searchdatacenter.techtarget.com/definition/server-sprawl.

[35] Margaret Rouse. Server virtualization, last visited on Augustus 9, 2019. https://searchservervirtualization.techtarget.com/definition/server-virtualization.

[36] Alexander Shvets. Dive into design patterns, last visited on July 19, 2020.

[37] Techopedia. Data virtualization, last visited on Augustus 9, 2019. https://www.techopedia.com/definition/1007/data-virtualization.

[38] Techopedia. Desktop virtualization, last visited on Augustus 9, 2019. https://www.techopedia.com/definition/601/desktop-virtualization.

[39] Techopedia. Network virtualization, last visited on Augustus 13, 2019. https://www.techopedia.com/definition/655/network-virtualization.

[40] Techopedia. Operating system virtualization (os virtualization), last visited on Augustus 9, 2019. https://www.techopedia.com/definition/660/operating-system-virtualization-os-virtualization.

[41] Techopedia. Server virtualization, last visited on Augustus 9, 2019. https://www.techopedia.com/definition/688/server-virtualization.

[42] Techopedia. Virtualization, last visited on Augustus 7, 2019. https://www.techopedia.com/definition/719/virtualization.

[43] V. E. Urias, W. M. S. Stout, B. Van Leeuwen, and H. Lin. Cyber range infrastructure limitations and needs of tomorrow: A position paper. In *2018 International Carnahan Conference on Security Technology (ICCST)*, pages 1–5.

[44] VMware. Server virtualization, last visited on Augustus 13, 2019. https://www.vmware.com/topics/glossary/content/server-virtualization.

[45] VMWare. Virtualization, last visited on Augustus 9, 2019. https://www.vmware.com/be/solutions/virtualization.html.

[46] Lars Vogel. Using dependency injection in java - introduction - tutorial, last visited on August 3, 2020. https://www.vogella.com/tutorials/DependencyInjection/article.html.

[47] H. Zhang, Y. Wang, X. Qiu, W. Li, and Q. Zhong. Network operation simulation platform for network virtualization environment. In *2015 17th Asia-Pacific Network Operations and Management Symposium (APNOMS)*, pages 400–403, Aug 2015.