CENTRALIZIRANA ADMINISTRACIJA KORIŠTENJEM POWERCLI-A

Boljat, Filip

Undergraduate thesis / Završni rad

2019

Degree Grantor / Ustanova koja je dodijelila akademski / stručni stupanj: **Algebra University College / Visoko učilište Algebra**

Permanent link / Trajna poveznica: https://urn.nsk.hr/urn:nbn:hr:225:647538

Rights / Prava: In copyright / Zaštićeno autorskim pravom.

Download date / Datum preuzimanja: **2024-04-26**

ALGEBRA VISOKO UČILIŠTE

Repository / Repozitorij:

Algebra Univerity College - Repository of Algebra Univerity College

zir.nsk.hr

dabar DIGITALNI AKADEMSKI ARHIVI I REPOZITORIJI

VISOKO UČILIŠTE ALGEBRA

ZAVRŠNI RAD

CENTRALIZIRANA ADMINISTRACIJA KORIŠTENJEM POWERCLI-A

Filip Boljat

Zagreb, prosinac 2018.

Student vlastoručno potpisuje Završni rad na prvoj stranici ispred Predgovora s datumom i oznakom mjesta završetka rada te naznakom:

„Pod punom odgovornošću pismeno potvrđujem da je ovo moj autorski rad čiji niti jedan dio nije nastao kopiranjem ili plagiranjem tuđeg sadržaja. Prilikom izrade rada koristio sam tuđe materijale navedene u popisu literature, ali nisam kopirao niti jedan njihov dio, osim citata za koje sam naveo autora i izvor, te ih jasno označio znakovima navodnika. U slučaju da se u bilo kojem trenutku dokaže suprotno, spreman sam snositi sve posljedice uključivo i poništenje javne isprave stečene dijelom i na temelju ovoga rada“.

U Zagrebu, datum.

Filip Boljat

Predgovor

Prvenstveno bih se htio zahvaliti svom mentoru i profesoru Vedranu Dakiću za svu podršku i mentorstvo koje mi je pružio tijekom pisanja završnog rada te zahvaliti na stečenom znanju koje sam dobio kroz njegove kolegije. Htio bih se zahvaliti i stručnom povjerenstvu koje je prepoznalo značaj i korisnost ovog završnog rada. Nikako ne bih htio zaboraviti zahvaliti se i ostalim profesorima koji su me podučavali kroz ove tri godine i u svakom trenutku bili spremni pomoći i odgovoriti na svako postavljeno pitanje kako bi još dodatno obogatili svoje znanje.

Prilikom uvezivanja rada, Umjesto ove stranice ne zaboravite umetnuti original potvrde o prihvaćanju teme završnog rada kojeg ste preuzeli u studentskoj referadi

Sažetak

U ovom završnom radu kreirao sam skriptu za kreiranje, brisanje i vraćanje virtualnih mašina u prijašnje stanje. Glavna motivacija za uzimanje ovakve teme bila mi je osobno zanimanje u području skriptiranja, od prvog puta kada sam se upoznao sa skriptiranjem želio sam naučiti kako pisati i razumjeti što skripta radi. Uz to, spoznaja da je korištenje skripti nešto što svaka ozbiljna IT organizacija koristi još dodatno me motiviralo da kreiram skriptu. Za potrebe pisanja skripte koristio sam PowerShell, odnosno PowerShell ISE, i PowerCLI, dodatni modul VMware-a koji mi je omogućio da pomoću njega upravljam virtualnim mašinama kreiranim na VMware hipervizoru kroz PowerShell. PowerShell je Microsoftov novi, poboljšani naredbeni redak koji se koristi u svrhu automatizacije zadataka i upravljanja konfiguracijama. Objektno je orijentiran, što znači da koristi objekte unutar svojeg koda koji su poznatiji pod imenom komandleti. PowerShell također ima ugrađen i skriptni dio, nazvan PowerShell ISE. Unutar njega, moguće je kreirati i testirati skripte te ga dodatno nadograditi novim modulima koji proširuju njegovu funkciju. Instalacija PowerShell-a nije bila potrebna jer je on već predinstaliran na svakom novom Windows operativnom sustavu.

Uz PowerShell, koristio sam i VMware-ovu virtualizacijsku tehnologiju pomoću koje sam kreirao virtualne mašine nad kojima sam testirao svoju skriptu. Ti VMware-ovi proizvodi uključuju Workstation Pro, vSphere ESXi hipervizor i PowerCLI. Workstation Pro i ESXi u osnovi vrše istu ulogu virtualizacije, no ESXi nije moguće instalirati na sustavu koji ima operativni sustav, dok Workstation Pro jest, ali PowerCLI se jedino može koristiti za ESXi zbog čega sam morao koristiti oba proizvoda kako bih virtualizirao ESXi unutar Workstation Pro-a i time omogućio da se ESXi pokreće na mom računalu. Kasnije sam unutar ESXi-a kreirao nove virtualne mašine koje su se koristile za potrebe testiranja skripte.

Pisanje same skripte provođeno je unutar PowerShell ISE. Sama skripta kroz tekstualno sučelje korisniku prikazuje glavni meni sa svim opcijama koje skripta nudi. Te opcije uključuju kreiranje potpuno novih virtualnih mašina ili kloniranja korištenjem linked klonova, brisanje ili vraćanje tih istih mašina u prijašnje stanje. Skripta je testirana i uspješno obavlja sve svoje funkcije, a ušteda vremena koju ostvaruje u odnosu na ručno obavljanje prije navedenih funkcija opravdava korisnost korištenja skripti.

Ključne riječi: PowerShell, PowerShell ISE, Workstation Pro, vSphere, ESXi, PowerCLI, skripta

Overview

In my thesis I made a script which creates, deletes and reverts virtual machines in their previous state. My main motivation for taking this kind of topic was my personal interest in the field of scripting, since the first time I got introduced to scripting I wanted to learn how to write and understand what a script does. In addition to that, realization that every serious IT organization uses scripts motivated me further to create a script. For the purpose of writing my script I used PowerShell, PowerShell ISE and PowerCLI, an additional VMware module which allowed me to manage virtual machines created inside the VMware hypervisor through PowerShell. PowerShell is Microsoft's new and improved command prompt which is used for the purpose of automating tasks and managing configurations. It is object oriented, which means that it uses objects inside its code which are more familiar under the name of commandlets. PowerShell also has an integrated scripting part, called PowerShell ISE. With it, it's possible to create and test scripts aswell as additionaly upgrading it with new modules which extend its functionality. Installation of PowerShell was not needed since it's already preinstalled on every new Windows operating system.

Alongside PowerShell, I used VMware's virtualization technology to create virtual machines which were used to test my scripts. These VMware products include Workstation Pro, vSphere ESXi hypervisor and PowerCLI. Workstation Pro and ESXi essentially do the same job, but ESXi requires to be installed on a OS free system, while Workstation Pro can be installed as a software, but PowerCLI can only be used for ESXi and for that reason I had to use both products to virtualize ESXi inside Workstation Pro and enable myself to run ESXi on my computer. Later on, I created new virtual machines inside ESXi to which were used for testing purposes for my script.

Writing of the script was done with PowerShell ISE. The script iteself through a text interface present the user a main menu with all the options that script offers. These options include creating completely new virtual machines or cloning them using linked clones, deleting or reverting those virtual machines in their previous state. The script has been tested and it successfully does its functions and the time save it achieves compared to the manual process of the beforementioned functions justifies its usefulness for using scripts.

Key words: PowerShell, PowerShell ISE, Workstation Pro, vSphere, ESXi, PowerCLI, script

Sadržaj

1. Uvod 1

2. Powershell 2

2.1. Što je PowerShell 2

2.1.1. Verzije PowerShell-a 2

2.2. Sistemski zahtjevi 3

2.3. Korištenje PowerShell-a 4

2.3.1. PowerShell struktura naredbi 4

2.3.2. Aliasi 6

2.3.3. Varijable 6

2.3.4. Upis i ispis informacija 7

3. VMware ..... **Pogreška! Knjižna oznaka nije definirana.**

3.1. Što je VMware Workstation 10

3.1.1. Sistemski zahtjevi 10

3.1.2. Instalacija 11

3.2. Što je VMware vSphere 12

3.2.1. VMware ESXi 12

3.2.2. Instalacija 12

3.2.3. vSphere Web Client 13

3.3. Što je PowerCLI 15

3.3.1. Sistemski zahtjevi 15

4. Priprema okoline 17

4.1. Instalacija PowerCLI-a 18

4.2. Priprema VMware okoline 21

4.3. Kôd u radu 25
4.3.1. Main Menu 31
4.3.2. Deploy 32
4.3.3. Revert 33
4.3.4. Delete 34
Zaključak 35
Popis kratica 37
Popis slika 38
Popis tablica 39
Popis kôdova 40
Literatura 41

1. Uvod

U današnjem poslovnom svijetu informacijska tehnologija (engl. Information Technology, skraćeno IT) postala je glavna okosnica svake organizacije koja svakim danom sve više raste i postaje kompleksnije a samim time i zahtjeva veću alokaciju vremena i resursa za njeno upravljanje. Jedna od tih tehnologija je i virtualizacija. Virtualizacija je koncept koji se počeo razvijati sredinom prošlog stoljeća, a danas je jedan od glavnih metoda za optimalno iskorištavanje ograničenih računalnih resursa i prostora. No unatoč tome što pomaže organizacijama uštediti gdje god je ušteda moguća, virtualizacija i dalje zahtjeva određenu mjeru administracije kako jednostavnih, tako i složenih zadataka. Upravo su ti jednostavni, elementarni zadaci najveći problem organizacija jer zahtjevaju alokaciju resursa, odnosno ljudske snage, za obavljanje ponavljajućih poslova koji su samo temelj za buduće, kompleksnije zadatke kojima organizacija ostvaruje profit. Zbog toga je potrebno na neki način automatizirati te osnovne, jednostavne zadatke i osloboditi što je više moguće ljudske snage te je alocirati na druge, kompleksnije zadatke.

Jedno od rješenja je korištenje PowerShell, točnije PowerCLI skripti, za tu automatizaciju. PowerCLI omogućuje administratorima virtualnih mašina da kreiraju skripte za kreiranje (engl. Deploy), brisanje (engl. Delete), vraćanje virtualnih mašina u prijašnje stanje (engl. Revert) i mnogih drugih opcija bez njihove prisutnosti i potrebe za ručnim obavljanjem ponavljajućih poslova na svakoj virtualnoj mašini.

Upravo zato je srvha ovog rada instalirati PowerCLI modul za PowerShell, pripremiti testnu okolinu kreirajući nekoliko virtualnih mašina korištenjem VMware ESXi hipervizora te napraviti interaktivnu skriptu koja će kroz tekstualno korisničko sučelje (engl. Text User Interface, skraćeno TUI) omogućiti korisniku da odabere koje opcije želi, a na temelju odabranih opcija skripta će, bez potrebe prisutnosti korisnika, obaviti zadani posao i time omogućiti korisniku da obavlja druge, kompleksnije poslove.

2. Powershell

2.1. Što je PowerShell

Windows PowerShell je Windows-ova naredbena ljuska (engl. command-line shell) osmišljena specifično za sistem administratore. Windows PowerShell nudi interaktivni ekran (engl. prompt) i skriptno okruženje koje se može koristiti nezavisno ili zajedno, u kombinaciji. Za razliku od drugih shell-ova, koji prihvaćaju i vraćaju određeni tekst, Windows PowerShell je izgrađen na .NET okviru (engl. framework), čime prihvaća i vraća .NET Framework objekte. Ova fundamentalna promjena donosi potpuno nove alate i metode za upravljanje i konfiguriranje Windows-om.

Windows PowerShell uvodi koncept command-leta (skraćeno cmdlet), jednostavnog naredbenog alata ugrađenog u shell. Svaki cmdlet se može koristiti zasebno, no njihova prava snaga se ostvaruje njihovim međusobnim kombiniranjem kako bi se obavljao neki kompleksni zadatak. Windows PowerShell uključuje više od stotine osnovnih cmdlet-a, a ujedno postoji i mogućnost pisanja vlastitih cmdlet-a i dijeljena istih sa drugim PowerShell korisnicima.

Kao i ostali shell-ovi, Windows PowerShell daje korisniku pristup datotečnom sustavu (engl. file system) na računalu. Uz to, Windows PowerShell pružatelji usluga (engl. providers) omogućuju pristup drugim pohranama podataka (engl. data store), poput registara i digitalnim certifikatima.

2.1.1. Verzije PowerShell-a

Trenutno, Microsoft nudi podršku za dvije različite verzije PowerShell-a, Powershell 5.1 i Powershell 6.0.

Powershell 5.1 je najnovija Windows PowerShell verzija, što znači da se može instalirati i koristiti jedino na Windows operativnom sustavu. S druge strane, PowerShell 6.0, tj. PowerShell Core je nova, više platformska (engl. cross-platform) verzija PowerShella koja je otvorenog standarda (engl. Open Source) te je podržana na Windows, Mac OS X i Linux baziranim operativnim sustavima. Višeplatformska priroda PowerShell Core-a znači da se skripte koje korisnik napiše na jednom operativnom sustavu mogu pokretati i koristiti na drugim operativnim sustavima i obrnuto. Microsoft aktivno radi na PowerShell Core-u, dok

za klasični PowerShell pružaju produženu podršku, što znači da Microsoft ne planira dodavati nove značajke na PowerShell, ali će izbacivati zakrpe za kritične greške i sigurnosne nadogradnje.

PowerShell Core 6.0 trenutno nije snažan kao i PowerShell 5.1, a jedan od glavnih razloga je taj što PowerShell 5.1 koristi .NET Framework dok PowerShell Core koristi .NET Core, koji ima manje mogućnosti. Glavne razlike između PowerShell i PowerShell Core-a mogu se vidjeti u tablici ispod:

	PowerShell	PowerShell Core
Verzija	1.0 do 5.1	6.0
Podržane platforme	Windows (klijent i server)	Windows, Mac OS, Linux
Ovisnost	.NET Framework	.NET Core
Korištenje	Oslanja se na .NET Framework runtime	Oslanja se na .NET Core runtime
Politika ažuriranja	Samo zakrpe za kritične bug-ove	Potpuno ažuriranje (značajke, bug-ovi)

Tablica 2.1 Usporedba PowerShell i PowerShell Core-a

2.2. Sistemski zahtjevi

Kako sam za potrebe ovog rada koristio Windows PowerShell 5.1 nabrojat ću samo zahtjeve za navedenu verziju PowerShell-a. Tablica ispod prikazuje zahtjeve za instalaciju PowerShell 5.1:

Operativni sustav	Zahtjevi
Windows Server 2016	Predinstalirano
Windows Server 2012 R2	Windows Management Framework 5.1
Windows Server 2008 R2 SP1	Windows Management Framework 5.1

Windows 10	Predinstalirano
Windows 8.1	Windows Management Framework 5.1
Windows 7 SP1	Windows Management Framework 5.1

Tablica 2.2 Sistemski zahtjevi za PowerShell 5.1

Kao što se može vidjeti iz tablice, PowerShell je na najnovijim verzijama Windows-a, kako klijentskog tako i serverskog, već predinstaliran što znači da je samo potrebno udovoljiti sistemskim zahtjevima operativnog sustava kojeg instaliravamo kako bi PowerShell ispravno radio. Valja napomenuti da starije verzije operativnih sustava također imaju predinstaliran Windows PowerShell, no taj PowerShell je starije verzije od 5.1 i stoga zahtjeva nadogradnju. Ukoliko želimo nadograditi postojeći PowerShell na starijim verzijama operativnih sustava, sve što je potrebno je instalirati Windows Management Framework 5.1, okvir koji donosi nadogradnje, zakrpe, dodatne funkcionalnosti i usluge za Windows PowerShell, Windows PowerShell Desired State Configuration (skraćeno DSC), Windows Remote Management (skraćeno WinRM), Windows Management Instrumentation (skraćeno WMI).

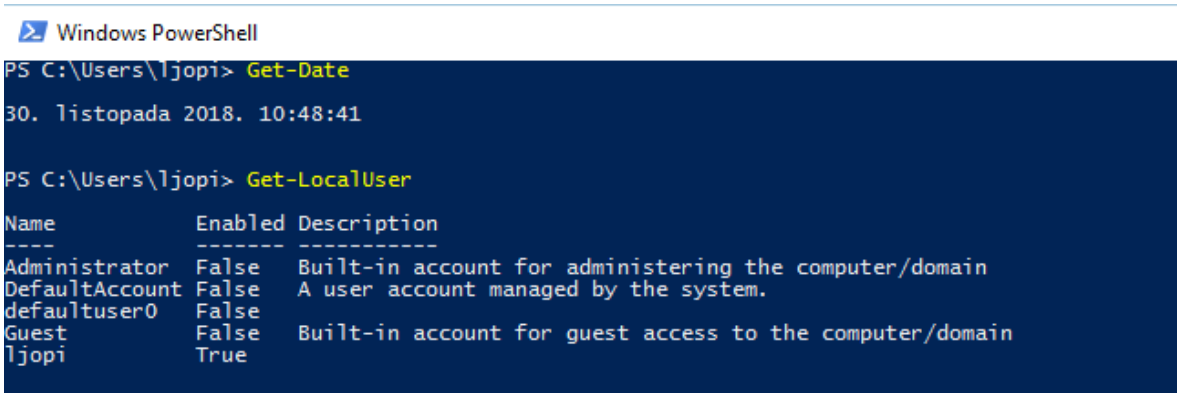
2.3. Korištenje PowerShell-a

Prije nego krenemo sa korištenjem PowerShell-a, potrebno je upoznati se sa samim programom kao i njegovom strukturom naredbi, terminologijom, funkcijama i mogućnostima koje nudi. Valja napomenuti da se u ovom radu spominju samo funkcije koje će se koristiti prilikom rađenja skripte, te da PowerShell ima i nudi puno više funkcija i mogućnosti.

2.3.1. PowerShell struktura naredbi

PowerShell koristi „glagol-imenica“ strukturu naredbi za svoje naredbe. Svako ime cmdleta se sastoji od standardnog glagola (engl. verb) povezanog sa specifičnom imenicom (engl. noun). PowerShell glagoli nisu uvijek glagoli temelji na Engleskom jeziku, ali oni uvijek izriču specifičnu radnju unutar PowerShell-a. Imenice unutar PowerShell-a vrše istu ulogu kao i imenice u bilo kojem jeziku, opisuju specifične tipove objekata koji su ključni u

sistemskoj administraciji. Glagoli s druge strane daju opis što imenica čini. Prednost ovakve strukture naredbi je jednostavnost razumijevanja što sama naredba radi. Naprimjer, ako želimo saznati koji je današnji datum, potrebno je upisati naredbu *Get-Date*. S druge strane, ako želimo saznati koji sve lokalni korisnici postoje na računalu, potrebno je upisati *Get-LocalUser*. Primjer ovih dviju naredbi može se vidjeti i na slici ispod:



Slika 2.1 Primjer PowerShell strukture naredbi

Kao što je bilo navedeno ranije, naredbe korištene u tradicionalnim naredbenolinijskim sučeljima (engl. interface) nemaju uvijek konzistentna imena parametara. Parametri su često napisani sa jednim znakom ili skraćenicama riječi radi jednostavnosti pisanja ali zato nisu intuitivni za nove korisnike.

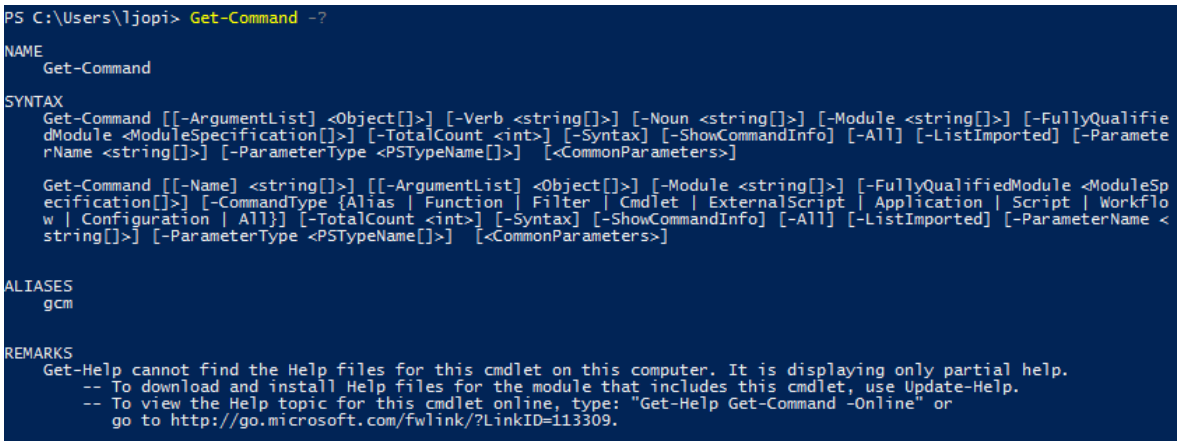
Za razliku od većine tradicionalnih naredbenolinijskih sučelja, PowerShell procesira parametre direktno i standardizira pisanje parametara. Ime parametra uvijek ima znak „-“ stavljen na početak svojeg imena, što se može vidjeti u sljedećem primjeru:

Get-Command -Name Write-Host

Ime parametra je *Name*, ali se piše *-Name* kada se koristi kao parametar.

2.3.1.1 Help parametar

Ukoliko želimo saznati više o pojedinom cmdlet-u, kao npr. njegovu sintaksu ili postojeće aliase za taj cmdlet, potrebno je upisati *-?* parametar nakon cmdlet-a.



Slika 2.2 Help parametar

2.3.1.2 Uobičajeni parametri

PowerShell ima nekoliko uobičajenih parametara. Ovi parametri su upravljani od strane PowerShell pogona (engl. engine). Uobičajeni parametri se uvijek ponašaju na isti način,a oni su: *WhatIf, Confirm, Verbose, Debug, Warn, ErrorAction, ErrorVariable, OutVariable, i OutBuffer.*

2.3.2. Aliasi

Iako su PowerShell naredbe specifične i konzistentne, nekada su i poprilično duge. Naredba poput *Set-WinDefaultInputMethodOverride* je primjer jedne dugačke, specifične naredbe. Iako je ime navedene naredbe prilično jasna i točno objašnjava što radi, njena dužina je čini neprivlačnom i nepraktičnom u slučaju kada je korisnik mora koristiti više puta. Zbog toga postoje PowerShell aliasi. Alias je ništa više nego drugo ime za naredbu.

Unutar PowerShell-a mogu se pronaći već ugrađeni (engl. built in) aliasi. Za popis svih aliasa koji postoje može se koristiti naredba *Get-Alias*. Također se mogu kreirati i novi aliasi naredbom *Set-Alias* ili *New-Alias*, no problem kod ručno kreiranih aliasa je taj što postoje onoliko dugo koliko je naš PowerShell prozor otvoren, što znači da nakon što ugasimo PowerShell svi naši aliasi nestaju. Za potrebe ovog rada nisam radio vlastite aliase, ali zato jesam koristio postojeće, unaprijed ugrađene aliase.

2.3.3. Varijable

Kao što smo već napomenuli, PowerShell ima svoj skriptni jezik koji ćemo koristiti kako bi napravili skriptu za upravljanje virtualnim mašinama. Uobičajeno je kod skriptiranja koristiti

imenovane objekte poznatije kao varijable. Kada se koriste unutar PowerShell-a, varijable specificira $ znak ispred njihovog imena, te omogućuju pohranjivanje bilo koje vrijednosti ili objekta unutar PowerShell-a. Radi lakšeg razumijevanja, slika ispod prikazuje primjer korištenja varijable:

Slika 2.3 Primjer korištenja varijable

Kreirao sam varijablu imena *varijabla* i pohranio u nju vrijednost *probna varijabla*. Nakon pozivanja kreirane varijable PowerShell mi za odgovor vraća pohranjenu vrijednost. Upravo ova mogućnost pohranjivanja objekata unutar varijable pokazat će se korisnom prilikom kreiranja skripte za upravljanjem virtualnim mašinama.

2.3.4. Upis i ispis informacija

Zadnja funkcija koja je relevantna za ovaj rad je upis i ispis informacija. PowerShell za upis koristi Read-Host cmdlet dok za ispis nudi *Write-Host* i *Write-Output* cmdlet-e.

2.3.4.1 Read Host

Powershell-ov Read-Host cmdlet je osmišljen da prikazuje tekstualni upit koji prikuplja bilo što što korisnik upiše. Kako bi bilo jednostavnije za razumijeti, slika ispod prikazuje primjer korištenja Read-Host cmdlet-a.

Slika 2.4 Read-Host cmdlet

Na temelju slike možemo izvući dva zaključka:

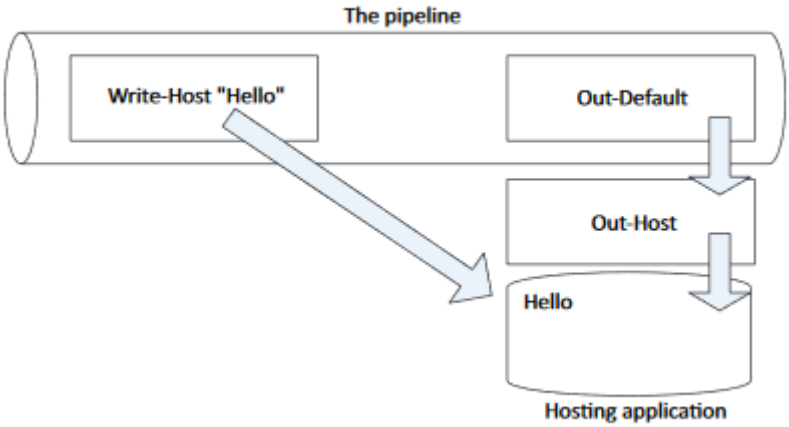
- cmdlet automatski stavlja dvotočje (znak ":") na kraju upita
- što god korisnik upiše se odmah vraća i ispisuje kao rezultat upisa

Read-Host se uglavnom ne koristi samostalno, nego se koristi u kombinaciji s varijablama kako bi se korisničke upisane vrijednosti privremeno pohranile i koristile u daljnje svrhe. Slika 2.5 prikazuje taj primjer.

Slika 2.5 Read host korišten u kombinaciji sa varijablom

2.3.4.2 Write-Host

Write-Host cmdlet je jedan on načina pomoću kojeg možemo prikazati ispis informacija. Kao što slika 2.6 prikazuje, Write-Host pokreće se unutar cjevovoda (engl. pipeline) kao i svaki drugi cmdlet, ali ne postavlja ništa unutar istog pipeline-a. Umjesto toga, ispisuje informacije direktno na ekran aplikacije.

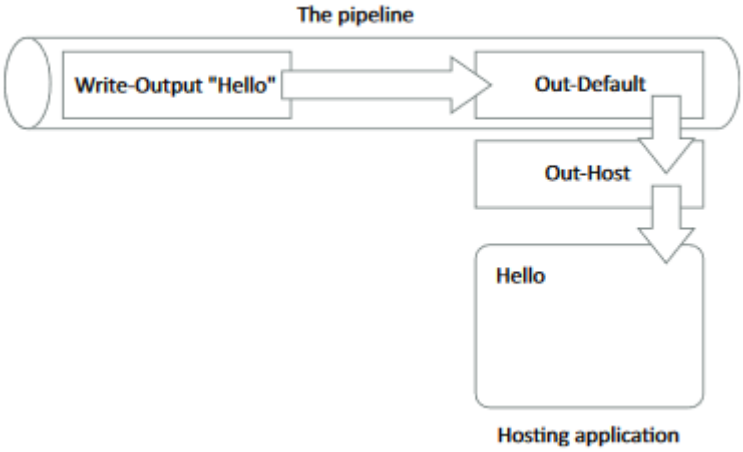


Slika 2.6 Write-Host princip rada

Write-Host bi se trebao koristiti samo kada se ukaže potreba za prikazivanjem specifične poruke, ali nije prikladan za prikazivanje ispisa iz skripte ili naredbe.

2.3.4.3 Write-Output

Za razliku od Write-Host cmdlet-a, Write-Output šalje objekte u pipeline. Slika 2.7 ilustrira kako Write-Output funkcionira.



Slika 2.7 Write output princip rada

- Write-Output stavlja String objekt Hello unutar pipeline-a.
- Kako trenutno ništa nije unutar pipeline-a, Hello putuje na kraj pipeline-a, gdje se Out-Default uvijek nalazi.
- Out-Default šalje objekt Out-Host-u.
- Out-Host pita PowerShell-ov sustav formatiranja da formatira objekt. U ovom primjeru je to jednostavna riječ, tj. String, sustav formatiranja vraća tekst string-a.
- Out-Host ispisuje formatirani rezultat na ekran.

Write-Output je shell-ov zadani (engl. default) cmdlet, što znači da svaki put kada upišemo nešto unutar PowerShell-a što nije naredba, shell šalje što god da smo upisali Write-Output cmdlet-u.

3. VMware

3.1. Što je VMware Workstation

VMware Workstation je softverski hipervizor koji se pokreće na x64 verzijama Windows i Linux operativnim sustavima te omogućuje korisniku da kreira i pokreće više virtualnih mašina na jednoj fizičkoj mašini. Svaka virtualna mašina može pokrenuti svoj operativni sustav bez ometanja rada druge virtualne mašine ili samog fizičkog stroja. Trenutno postoje dvije verzije VMware Workstationa, besplatni VMware Workstation Player koji se može koristiti u nekomercijalne svrhe i komercijalni VMware Workstation Pro, koji nudi veći broj funkcionalnosti i sigurnosti nego njegova besplatna verzija. Za potrebe ovog rada koristio sam VMware Workstation Pro jer nudi mogućnost korištenja slika u vremenu (engl. snapshot), što mi je omogućilo da slobodno testiram svoju skriptu i svoju virtualnu mašinu vratim u prijašnje, funkcionalno stanje u slučaju da uzrokujem neki kvar testiranjem.

3.1.1. Sistemski zahtjevi

Sljedeća tablica prikazuje sistemske zahtjeve za instalaciju VMware Workstation Pro-a:

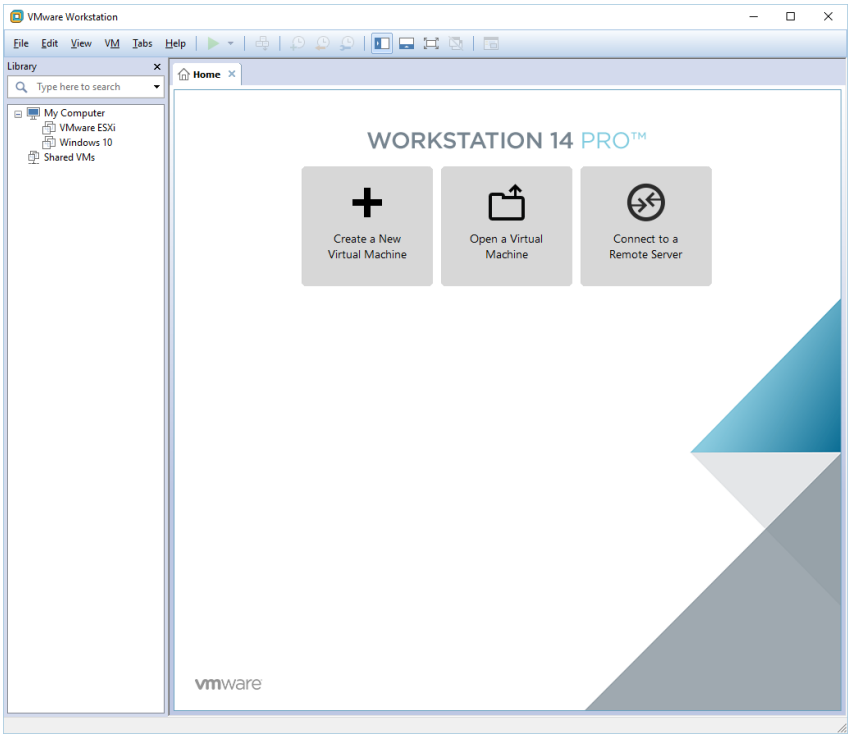
Arhitektura	64-bit x86 Intel ili AMD procesor
Brzina procesora	1.3GHz ili brže
Memorija	2GB RAM minimum / 4GB RAM preproučeno
Diskovni prostor	1.2GB slobodnog prostora za aplikaciju
Operativni sustav	• Windows Server 2008 i novije • Windows 7 i novije • Ubuntu 8.04 i novije • RHEL 5 i novije • CentOS 5.0 i novije

	• Oracle Linux 5.0 i novije • openSUSE 10.2 i novije • SUSE Linux 10 i novije

Tablica 3.1 Sistemski zahtjevi VMware Workstation Pro

3.1.2. Instalacija

Instalacija VMware Workstation Pro-a je trivijalna. Sve što je potrebno je pokrenuti čarobnjak za instalaciju te odabrati lokaciju instalacije Workstation-a, označiti želimo li da se Workstation automatski ažurira i odabrati želimo li Workstation ikonu na radnoj površini i start meniju. Nakon što smo prilagodili Workstation instalaciju svojim željama kreće i sama instalacija nakon koje je potrebno upisati ispravni licencni ključ ukoliko želimo koristiti aplikaciju i nakon 30 dana. Time završava proces instalacije i možemo pokrenuti VMware Workstation Pro po prvi puta. Slika ispod prikazuje primjer pokrenute aplikacije.



Slika 3.1 WMWare Workstation Pro početni zaslon

3.2. Što je VMware vSphere

VMware vSphere nije specifičan proizvod ili softver. VMware vSphere je komercijalni naziv za cijeli paket VMware proizvoda. VMware vSphere paket proizvoda sastoji se od virtualizacije, managementa i sučelja koje sve navedeno objedinjuje. Dvije glavne komponente vSphere-a jesu ESXi server i vCenter Server. ESXi je hipervizor, pomoću kojeg se kreiraju i pokreću virtualne mašine dok je vCenter Server softver koji nam pomaže da centralno upravljamo cijelom virtualizacijskom infrastrukturom. Drugim riječima, ukoliko imamo više VMware ESXi hostova koji tvore našu virtualizacijsku infrastrukturu, vCenter Server omogućuje nam centralizirano upravljanje svih naših ESXi hostova neovisno o njihovoj međusobnoj udaljenosti. Valja napomenuti da je za korištenje vCenter Server-a potrebno koristiti plaćenu verziju ESXi-a jer besplatna verzija ESXi hipervizora nema vCenter podršku, a kako sam ja za potrebe ovog rada koristio besplatnu ESXi verziju te ujedno i samo jedan ESXi host, nisam vidio potrebu za korištenjem vCenter Server-a.

3.2.1. VMware ESXi

VMware ESXi, to jest Elastic Sky X integrated, je bare metal, tip 1 hipervizor koji služi za kreiranje i pokretanje virtualnih mašina. Kao tip 1 hipervizor, ESXi nije softverska aplikacija koja se instalira na postojeći operativni sustav, nego već uključuje zasebni operativni sustav i njegove komponente, poput Linux kernela.

3.2.2. Instalacija

Instalacija VMware ESXi-a postiže se na isti način kao i instalacija bilo kojeg drugog operativnog sustava, korištenjem prijenosnog medija poput USB-a ili CD-a, tj. DVD-a. Za potrebe ovog rada, ja sam simulirao instalaciju ESXi-a unutar VMware Workstation-a, točnije, kreirao sam novu virtualnu mašinu, preuzeo .iso datoteku koja sadrži ESXi i „umetnuo“ je u DVD pogon moje virtualne mašine kako bi se instalacija ESXi-a pokrenula. Sama instalacija je jednostavna i jedino što zahtjeva je da odaberemo na koji disk želimo da se hipervizor instalira, jezik tipkovnice i lozinku za root korisnika. Nakon instalacije, potrebno je ponovno pokrenuti računalo nakon čega nas na početnom zaslonu dočekuje ekran prikazan na slici ispod:

Slika 3.2 VMware ESXi početni zaslon

Ukoliko želimo konfigurirati, testirati ili resetirati management mrežu, uklanjati moguće nastale probleme (engl. troubleshoot) ili pogledati sistemske logove kao i informacije sustava, potrebno je na tipkovnici pritisnuti F2 te upisati lozinku root-a koju smo postavili tokom same instalacije ESXi hipervizora. Ako želimo započeti sa kreiranjem i upravljanjem virtualnih mašina i ostalih komponenti poput storage-a i networkinga moramo koristiti vSphere Web Client.

3.2.3. vSphere Web Client

vSphere Web Client je web bazirana aplikacija pomoću koje se korisnik povezuje na svoj ESXi host te njime i upravlja. vSphere Web klijent korisniku prezentira grafičko korisničko sučelje (skraćeno GUI) koje se sastoji od navigatora objekata, glavnog radnog prostora i ploče sa zadacima i alarmima. Sam web klijent se instalira prilikom instalacije VMware vSphere-a i nije ga potrebno zasebno kupovati niti preuzimati.

3.2.4. Kloniranje

VMware unutar svoje virtualizacijske tehnologije nudi mogućnost kloniranja virtualnih mašina. Klon je identična kopija već postojeće virtualne mašine, koju se naziva roditelj (engl. parent). Nakon što se operacija kloniranja izvrši, klonirana virtualna mašina može biti potpuno nezavisna od svog roditelja ili može dijeliti virtualni disk sa svojim roditeljem. Te dvije vrste kloniranja nazivaju se Full Clone i Linked Clone. Neovisno o tome koji tip kloniranja se koristi, promjene koje nastaju na klonu ne utječu na roditelja kao što ni promjene nastale na roditelju ne utječu na klon.

3.2.4.1 Full Clone

Full Clone je nezavisna virtualna mašina koja ne zahtjeva konstantnu vezu sa roditeljem. Zato što full clone ne dijeli virtualni disk sa roditeljem, ovakav način kloniranja generalno daje bolje performanse nego linked clone. Međutim, full clone način kloniranja traje duže nego linked clone i zauzima više diskovnog prostora.

3.2.4.2 Linked Clone

Linked Clone je za razliku od Full Clone-a klon napravljen iz prijašnjeg stanja roditelja (odnosno snapshot-a). Svi podaci koji su u tom stanju dostupni na roditelju ostaju dostupni i na linkanom klonu. Sve nove promjene koje nastaju na roditelju ne prenose se na klon kao što se ni nove promjene nastale na klonu ne prenose na roditelja. Ono što je bitno za napomenuti je da linkani klon mora imati omogućen pristup roditelju jer bez tog pristupa klon je onemogućen.

Prednost linkanih klonova jest taj što se kreiraju brzo, što nam omogućava da kreiramo nove unikatne virtualne mašine za svaki zadatak koji imamo, a ujedno i uvelike štede diskovni prostor jer svaki klon koristi disk svog roditelja, a sve nove promjene zapisuje na puno manji disk, poznatiji kao delta disk. Upravo zbog ovih prednosti Linked Clone se vrlo često koristi unutar organizacija kako bi se optimalno iskoristilo vrijeme i uštedilo što je više moguće prostora na disku.

3.3. Što je PowerCLI

VMware PowerCLI je VMware-ov PowerShell modul koji proširuje mogućnosti PowerShell-a da razumije i koristi VMware okruženje. Tokom instalacije, koje ćemo se dotaknuti malo kasnije, PowerCLI instalira VMware specifične PowerShell komandlete koji nam omogućuju da radimo sa VMware vSphere okruženjem.

3.3.1. Sistemski zahtjevi

VMware PowerCLI se može instalirati na sljedećim operativnim sustavima:

OS tip	64-bit
Server	• Windows Server 2016 • Windows Server 2012 R2 • Windows Server 2008 R2 Service Pack 1
Workstation	• Windows 10 • Windows 8.1 • Windows 7 Service Pack 1 • Ubuntu 16.04 • macOS 10.12

Tablica 3.2 PowerCLI sistemski zahtjevi

Kao što se vidi iz tablice, PowerCLI se može instalirati samo na Microsoft Windows serverima, dok se za radne stanice uz Windows operativne sustave još i uključuje Ubuntu i macOS. Razlog tome je što je za instalaciju PowerCLI-a potreban PowerShell, što jedino Windows bazirani serveri trenutno imaju, dok se kod radnih stanica proširuje i na Linux i BSD bazirane operativne sustave zbog PowerShell Core-a, kojeg smo spominjali na početku ovog rada.

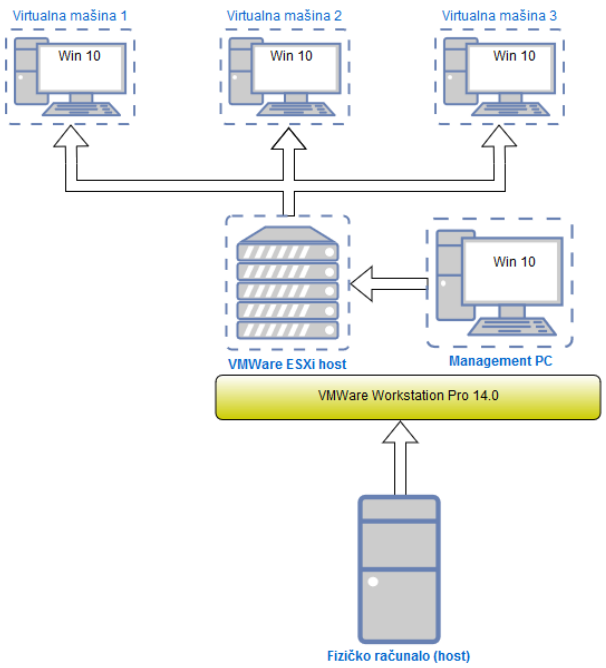
Nadalje, ako želimo uspješno instalirati PowerCLI na našem sustavu, moramo osigurati da je sljedeći set softvera instaliran:

OS tip	.NET verzija	PowerShell verzija
Windows	.NET Framework 4.5, 4.5.x, 4.6, 4.6.x ili 4.7.x	Windows PowerShell 3.0, 4.0, 5.0 ili 5.1
Ubuntu	.NET Core 2.0	PowerShell Core 6.0.2
macOS	.NET Core 2.0	PowerShell Core 6.0.2

Tablica 3.3 Set softvera potrebnog za uspješnu instalaciju PowerCLI-a

4. Priprema okoline

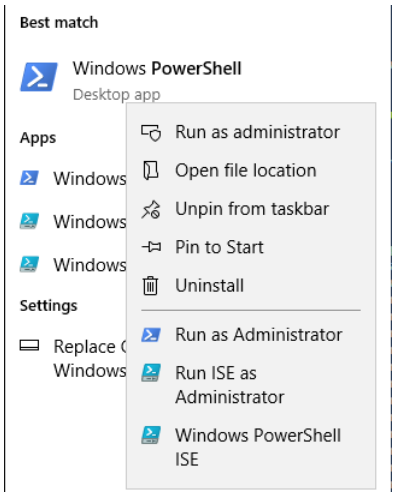
Prije nego što krenemo sa kreiranjem interaktivne skripte za centraliziranu administraciju virtualnih mašina, potrebno je pripremiti okolinu kojom ćemo moći i testirati samu skriptu. Za potrebe ovog rada koristio sam VMware Workstation 14 Pro hipervizor na kojem sam kreirao VMware vSphere 6 hipervizor na kojem se nalaze 3 Windows 10 virutalne mašine te jednu Windows 10 virtualnu mašinu radi potrebe pristupa i administracije ESXi hosta. Razlog zašto sam odabrao ovakvu infrastrukturu jest taj što se VMware Workstation bazira na softverskoj virtualizaciji, odnosno tip 2 virtualizaciji, koja mi omogućava da ga instaliram na postojećem operativnom sustavu i jednostavno upravljam njima. VMware vSphere hipervizor je, za razliku od VMware Workstation-a, baziran na hardverskoj virtualizaciji, odnosno bare metal virtualizaciji kod kojeg virtualizacijski softver direktno koristi hardverske resurse, što znači da se ne može instalirati na postojećem operativnom sustavu nego sam hipervizor sadrži svoj vlastiti operativni sustav. Upravo radi tog razloga najbolje rješenje mi je bilo napraviti infrastrukturu kojoj je temelj softverski hipervizor (VMware Workstation Pro), a zatim na tome kreirati bare metal hipervizor radi lakšeg upravljanja i administracije, te, na kraju, na virtualiziranom bare metal hipervizoru napraviti virtualne mašine koje će biti korištene u svrhu testiranja PowerCLI skripte. Kako bi bilo lakše i jednostavnije razumijeti opisanu infrastrukturu, slika ispod ju prikazuje.



Slika 4.1 Infrastruktura korištena za tesitranje skripte

4.1. Instalacija PowerCLI-a

Kako bi mogli raditi PowerCLI skripte potrebno je instalirati PowerCLI modul na administrativnoj Windows 10 mašini. Proces instalacije dotičnog modula radi se kroz PowerShell, stoga je potrebno i pokrenuti PowerShell. Važno je za napomenuti da je instalacija modula moguća jedino ako smo administrator, to jest, ako imamo povišena prava. Upravo zbog tog razloga potrebno je pokrenuti PowerShell kao administrator, što je najjednostavnije desnim klikom na PowerShell i odabirom kartice Run As Administrator.



Slika 4.2 Pokretanje PowerShell-a kroz Windows Start meni

Nakon što smo pokrenuli PowerShell u administrativnom modu, možemo krenuti na proces instalacije PowerCLI modula. Potrebno je potražiti dotični modul PowerShell komandletom (skraćeno cmdlet) *Find-Module*. Ovaj cmdlet traži module iz online galerije koji odgovaraju upitu koji korisnik zadaje. U ovom slučaju, koristio sam parametar *-Name VMware.PowerCLI* za traženje modula po njegovom imenu što mi je vratilo željeni modul.

```
PS C:\Windows\system32> Find-Module -Name VMWare.PowerCLI

Version    Name                                Repository           Description
-------    ----                                ----------           -----------
10.2.0.... VMware.PowerCLI                     PSGallery            This Windows PowerShell module contains VMware.P...
```

Slika 4.3 Find-Module cmdlet

Nakon što smo pronašli željeni modul možemo ga instalirati cmdlet-om *Install-Module*. Kao što ime cmdleta predlaže, njime se preuzima jedan ili više modula iz online galerije te instalirava na lokalno računalo. Kao i kod *Find-Module* cmdlet-a, parametrom *-Name* specificiramo naziv modula koji želimo instalirati.

```
PS C:\Windows\system32> Install-Module -Name VMware.PowerCLI
```

Slika 4.4 Install-Module cmdlet

Za provjeru uspješnosti instalacije modula možemo koristiti **Get-Module** cmdlet, točnije, Get-Module VMware.* -ListAvailable naredbu koja će nam izlistati sve instalirane module koji na početku svog imena sadržavaju VMware parametar.

```
PS C:\Windows\system32> Get-Module VMWare.* -ListAvailable


    Directory: C:\Program Files\WindowsPowerShell\Modules


ModuleType Version    Name                                ExportedCommands
---------- -------    ----                                ----------------
Script     6.7.0.8... VMware.DeployAutomation             {Add-DeployRule, Add-ProxyServer, Add-ScriptBundle, Copy-D...
Script     6.7.0.8... VMware.ImageBuilder                 {Add-EsxSoftwareDepot, Add-EsxSoftwarePackage, Compare-Esx...
Manifest   10.2.0.... VMware.PowerCLI
Script     6.7.0.9... VMware.Vim
Script     10.1.0.... VMware.VimAutomation.Cis.Core       {Connect-CisServer, Disconnect-CisServer, Get-CisService}
Script     10.0.0.... VMware.VimAutomation.Cloud          {Add-CIDatastore, Connect-CIServer, Disconnect-CIServer, G...
Script     10.1.0.... VMware.VimAutomation.Common
Script     10.1.0.... VMware.VimAutomation.Core           {Add-PassthroughDevice, Add-VirtualSwitchPhysicalNetworkAd...
Script     6.5.4.7... VMware.VimAutomation.HA             Get-DrmInfo
Script     7.5.0.8... VMware.VimAutomation.HorizonView    {Connect-HVServer, Disconnect-HVServer}
Script     10.0.0.... VMware.VimAutomation.License        Get-LicenseDataManager
Script     10.2.0.... VMware.VimAutomation.Nsxt           {Connect-NsxtServer, Disconnect-NsxtServer, Get-NsxtService}
Script     10.0.0.... VMware.VimAutomation.PCloud         {Connect-PIServer, Disconnect-PIServer, Get-PIComputeInsta...
Script     10.1.0.... VMware.VimAutomation.Sdk
Script     10.0.0.... VMware.VimAutomation.Srm            {Connect-SrmServer, Disconnect-SrmServer}
Script     10.1.0.... VMware.VimAutomation.Storage        {Add-KeyManagementServer, Copy-VDisk, Export-SpbmStoragePo...
Script     1.2.0.0    VMware.VimAutomation.StorageUtility Update-VmfsDatastore
Script     10.1.0.... VMware.VimAutomation.Vds            {Add-VDSwitchPhysicalNetworkAdapter, Add-VDSwitchVMHost, E...
Script     10.0.0.... VMware.VimAutomation.Vmc            {Connect-Vmc, Disconnect-Vmc, Get-VmcService, Connect-VmcS...
Script     10.0.0.... VMware.VimAutomation.vROps          {Connect-OMServer, Disconnect-OMServer, Get-OMAlert, Get-O...
Script     6.5.1.7... VMware.VumAutomation                {Add-EntityBaseline, Copy-Patch, Get-Baseline, Get-Complia...
```

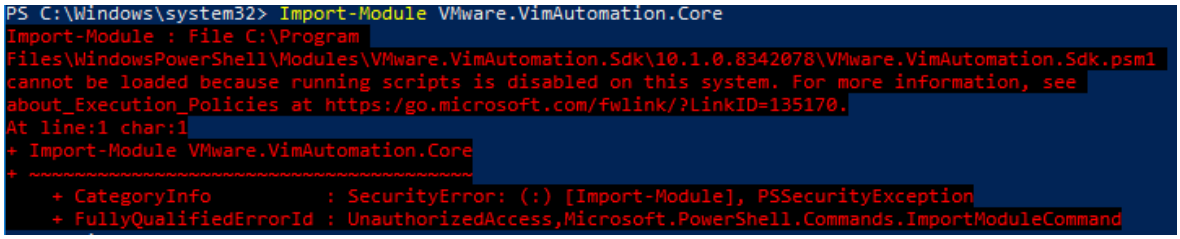
Slika 4.5 Get-Module cmdlet

Kao što možemo vidjeti, VMware PowerCLI modul je instaliran. Za provjeru uspješnosti instalacije probao sam upisati jednu od PowerCLI naredbi, Connect-VIServer. Ovom naredbom trebao sam se moći spojiti na svoj ESXi host, no naišao sam na grešku.

```
PS C:\Windows\system32> Connect-VIServer -Server 192.168.157.100
Connect-VIServer : The 'Connect-VIServer' command was found in the module 'VMware.VimAutomation.Core',
but the module could not be loaded. For more information, run 'Import-Module VMware.VimAutomation.Core'.
At line:1 char:1
+ Connect-VIServer -Server 192.168.157.100
+ ~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : ObjectNotFound: (Connect-VIServer:String) [], CommandNotFoundException
    + FullyQualifiedErrorId : CouldNotAutoloadMatchingModule
```

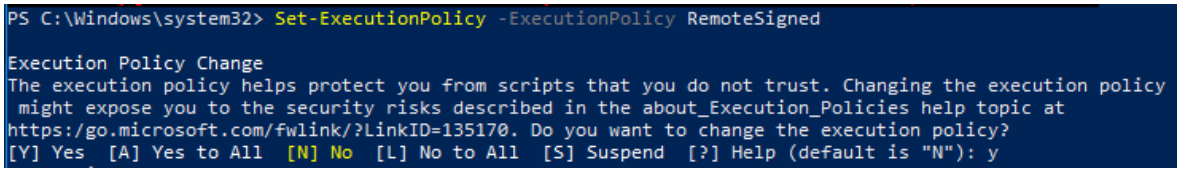
Slika 4.6 Connect-VIServer prva greška

Kao što vidimo na slici, upisao sam dobru naredbu, ali modul u kojem se ta naredba nalazi nije se mogao učitati. Stoga sam probao učitati modul upisavši Import-Module VMware.VimAutomation.Core što mi je vratilo novu grešku koja govori da učitavanje modula nije moguće jer je pokretanje skripti onemogućeno.



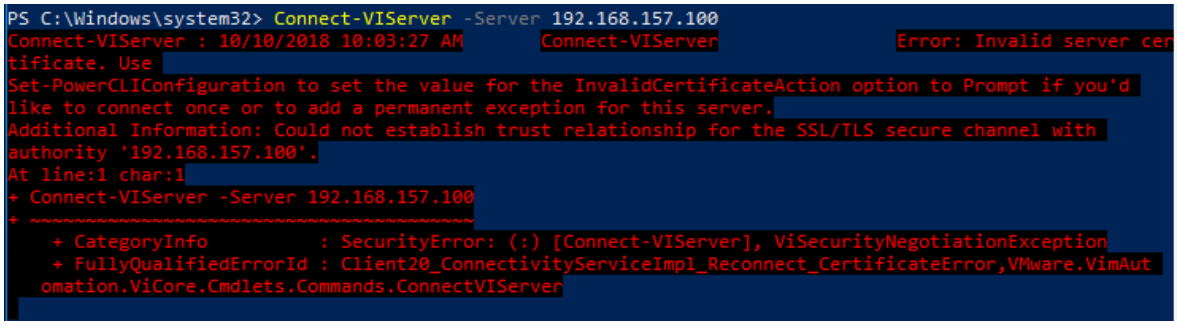
Slika 4.7 Import-Module prva greška

Kako bi omogućio pokretanje skripti, potrebno je bilo upisati cmdlet *Set-ExecutionPolicy -ExecutionPolicy RemoteSigned* koji omogućuje pokretanje onih skripti koje su preuzete sa interneta i koje su potpisane od strane pouzdanog izdavača.



Slika 4.8 Set-ExecutionPolicy cmdlet

Nakon što sam omogućio pokretanje skripti, opet sam probao importati modul što je ovoga puta uspjelo. Ponovno sam probao spojiti se na ESXi host sa *Connect-VIServer* cmdlet-om, no opet sam dobio grešku, ali ovoga puta radilo se o nevažećem certifikatu.



Slika 4.9 Connect-VIServer druga greška

Kako je ovo testna okolina i nije potrebna tolika mjera sigurnosti, odlučio sam ukloniti bilo kakvu potrebu za certifikatima sa *Set-PowerCLIConfiguration -InvalidCertificateAction ignore -confirm:$false* cmdlet-om. Ponovno sam se probao spojiti na svoj ESXi host i ovoga puta je uspjelo čime sam potvrdio da se PowerCLI modul uspješno instalirao.

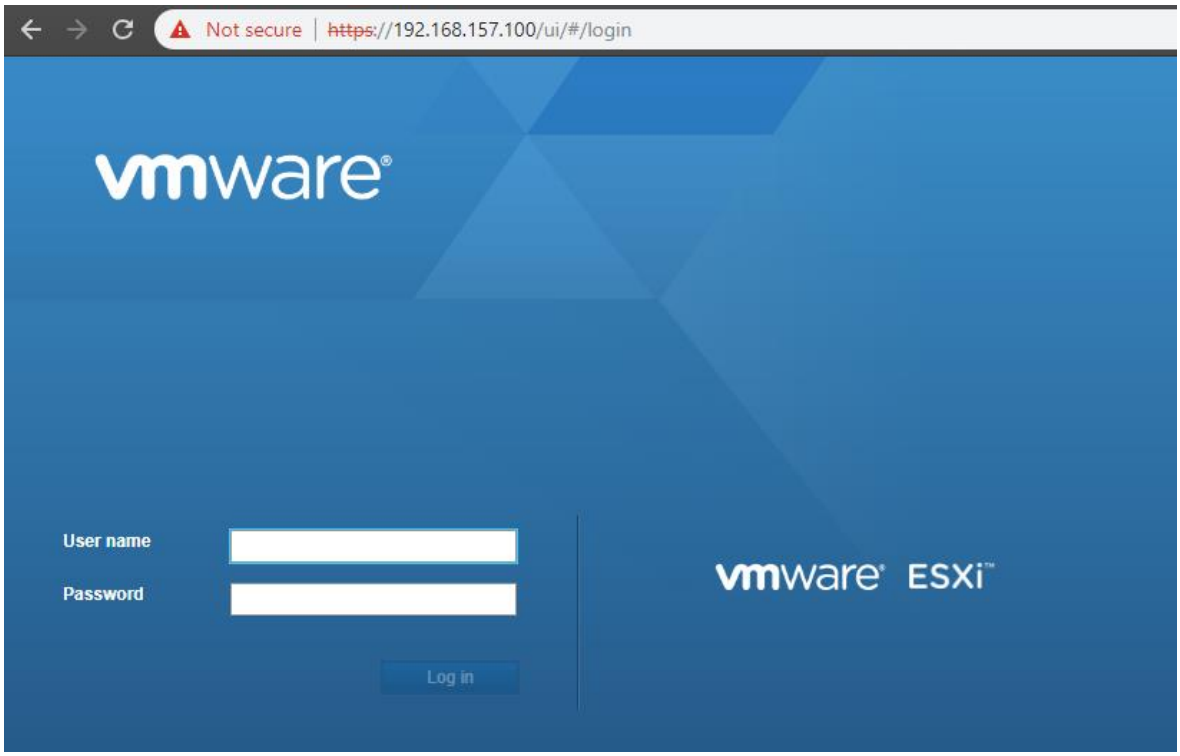
```
PS C:\Windows\system32> Connect-VIServer -Server 192.168.157.100

Name                           Port  User
----                           ----  ----
192.168.157.100                443   root
```

Slika 4.10 Connect-VIServer povezivanje

4.2. Priprema VMware okoline

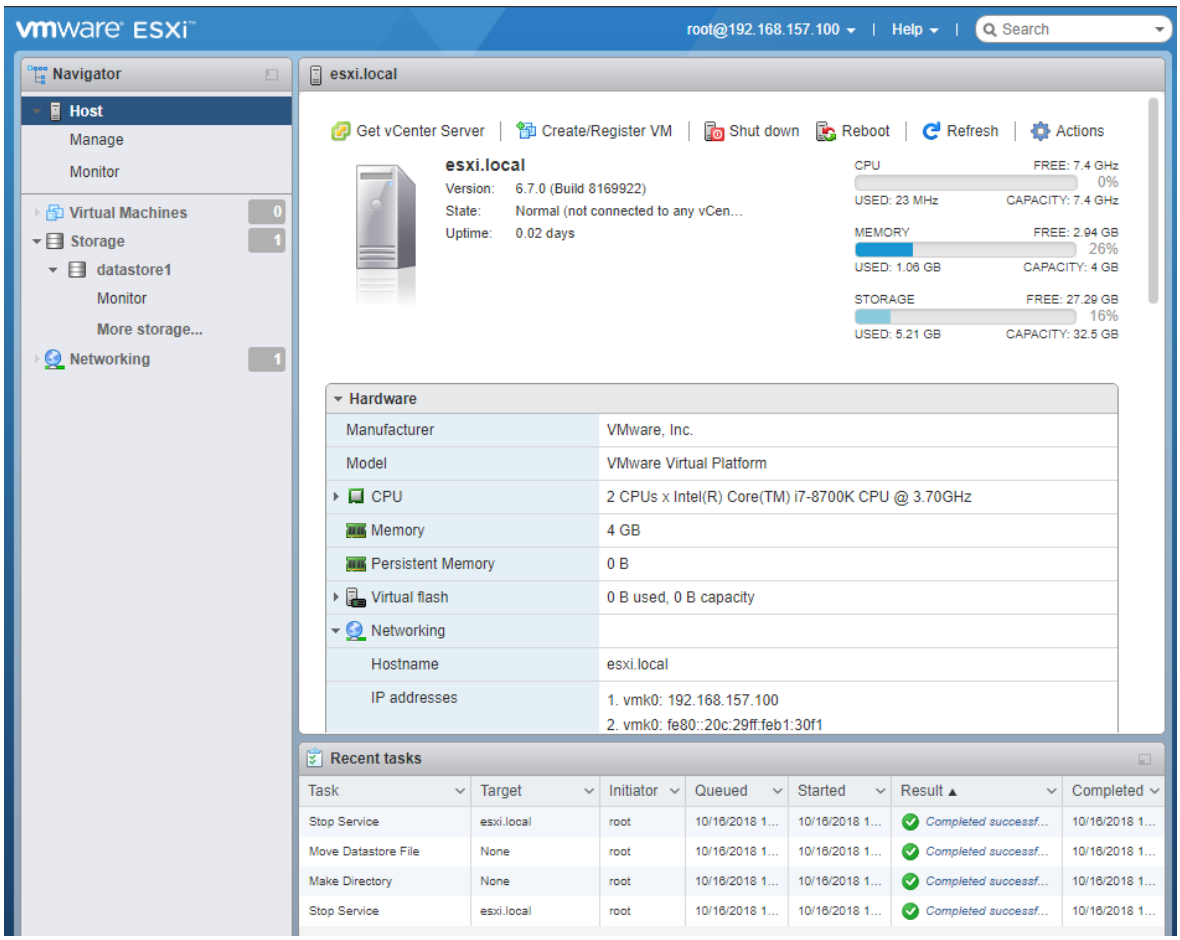
Nakon što smo uspješno instalirali i testirali PowerCLI modul na klijentskoj mašini, potrebno je pripremiti testnu okolinu koju ćemo koristiti za testiranje skripte za administraciju virtualnih mašina koju ćemo kasnije raditi. Tu okolinu pripremit ćemo unutar ESXi hosta jer PowerCLI ne podržava VMware Workstation.

Za početak potrebno je spojiti se na ESXi host. To ćemo učiniti ili putem *Connect-VIServer* cmdlet-a ili upisivanjem adrese ESXi hosta unutar web preglednika. Radi jednostavnosti i lakšeg snalaženja kao i upravljanja odabrao sam koristiti web preglednik. Nakon upisivanja adrese ESXi hosta dočekat će nas početni ekran na kojem se od nas traži da upišemo korisničko ime i lozinku. Slika prikazuje početnu stranicu.



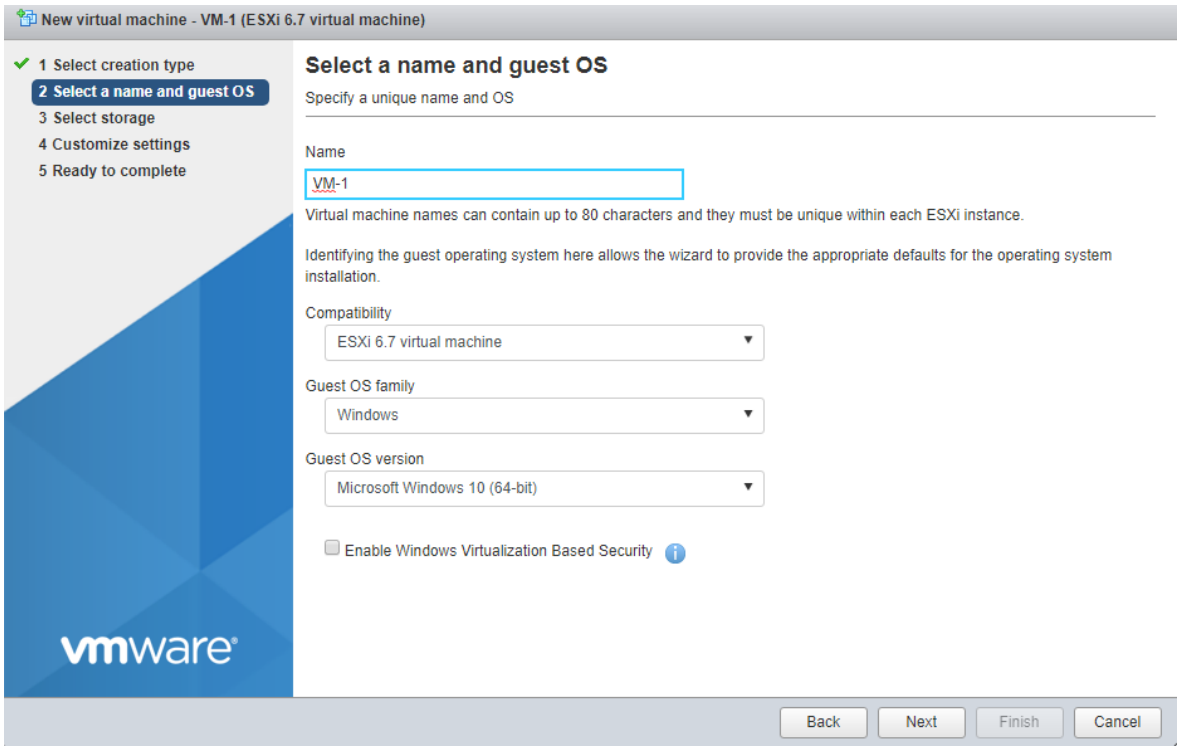
Slika 4.11 VMware vSphere web login sučelje

Nakon upisivanja točnih kredencijala i klikom na gumb Log in dočekuje nas početna stranica ESXi hosta u kojem se vrši sva administracija i nadzor.



Slika 4.12 VMware vSphere web početni zaslon

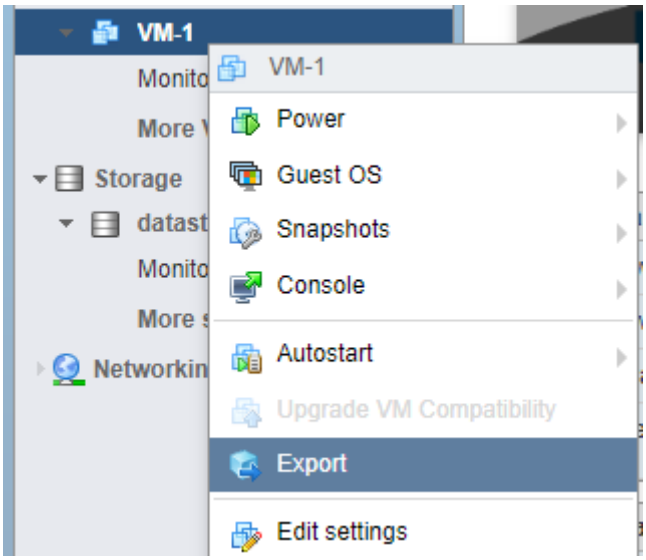
Kao što se vidi na slici, trenutno nemamo kreiranu niti jednu virtualnu mašinu. Kako bismo krenuli sa kreacijom virtualne mašine potrebno je kliknuti na Create/Register VM te zatim pratiti čarobnjak za kreaciju nove virtualne mašine. Unutar tog čarobnjaka udabiremo ime, vrstu OS-a kao i njegovu verziju, mjesto pohrane (odnosno datastore) konfiguracijskih datoteki i virtualnih diskova te broj CPU-ova, memorije, veličinu diska i mrežnu karticu koju će ta virtualna mašina imati. Nakon što smo završili sa kreiranjem naše virtualne mašine, ESXi host nam javlja da se virtualna mašina kreirala i ta ista mašina se prikazuje unutar kartice Virtual Machines.



Slika 4.13 Kreiranje virtualne mašine

Kao što je spomenuto na početku, za potrebe ovog testiranja kreirat ću 3 virtualne mašine, stoga ću još morati kreirati preostale 2 virtualne mašine. Iako bih preostale 2 mašine mogao kreirati na isti način kao i prvu, tj. početi „od nule“ i pratiti čarobnjak, takav pristup nije skalabilan, odnosno nije primjenjiv u situaciji gdje je potrebno kreirati 10, 50, 100 ili više identičnih virtualnih mašina. Zbog toga postoje alternative koje ubrzavaju proces kreiranja virtualnih mašina, a koje se u poslovnom svijetu svakodnevno primjenjuju. Jedna od tih alternativa je kloniranje. Ukratko, kloniranje je postupak kopiranja postojeće virtualne mašine i kreiranja nove i nezavisne virtualne mašine. Ovu metodu sam koristio i ja za kreiranje preostale 2 mašine. Još jedna alternativa, za koju je preduvjet imati instaliran PowerCLI, je pisanje skripte za kreiranje virtualnih mašina. Ova opcija je najbolja jer omogućuje brzo kreiranje velikog broja virtualnih mašina, no razlog zašto se nisam odlučio za ovu opciju jest taj što mi trebaju samo 2 dodatne virtualne mašine koje su potpuno identične kao i prva.

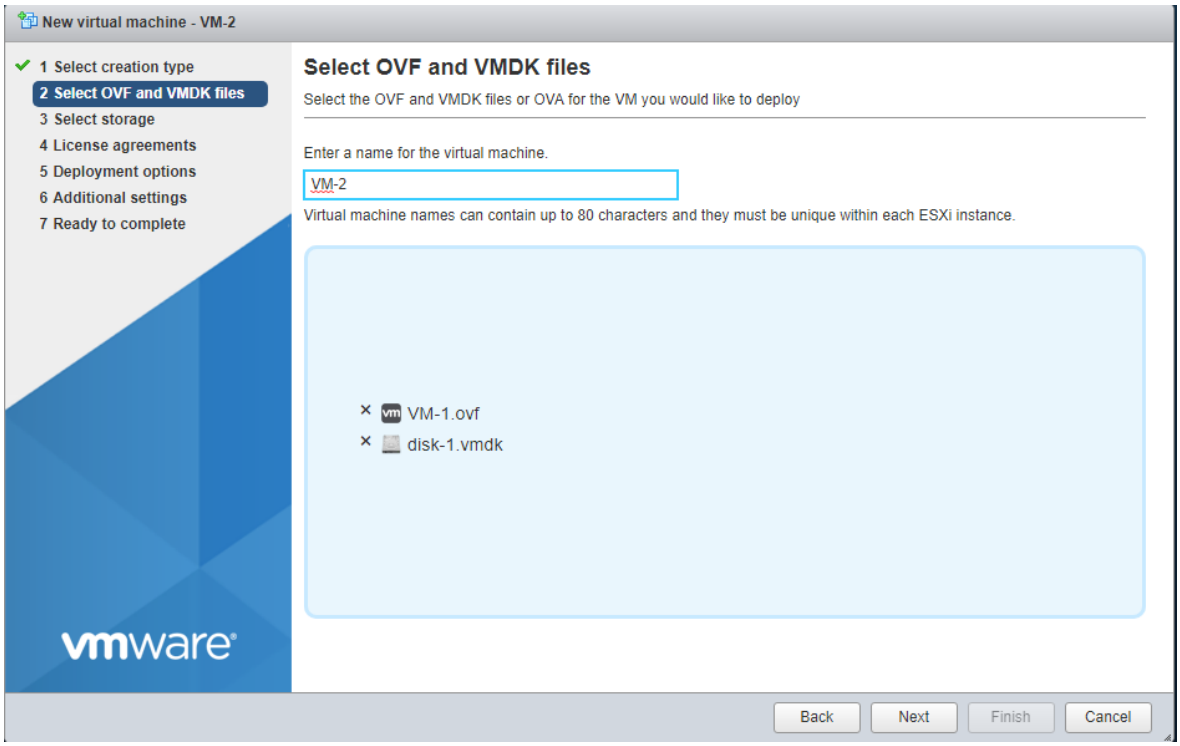
Za proces kloniranja koristit ću značajku Export unutar VMware ESXi web sučelja.



Slika 4.14 Export virtualne mašine

Klikom na Export započet ću proces preuzimanja OVF paketa. OVF, odnosno Open Virtualization Format je otvoreni standard za pakiranje i distribuiranje virtualnih uređaja kao što su virtualne mašine. OVF paket sadrži nekoliko datoteka u jednom direktoriju, a to je OVF deskriptor, XML datoteka koja sadrži metapodatke OVF paketa (kao što je ime, hardverski zahtjevi, reference na druge datoteke i sl.), image diska (u ovom slučaju .vmdk) i opcionalne certifikacijske datoteke i druge pomoćne datoteke. VMDK, odnosno Virtual Machine Disk, je otvoreni format razvijan od strane VMware-a koji virtualizira tvrdi disk virtualne mašine.

Nakon preuzimanja OVF paketa na računalo može se kreirati nova virtualna mašina istim postupkom kao i što smo kreirali prvu, no umjesto odabira prve opcije, Create a new virtual machine, odabiremo drugu opciju, Deploy a virtual machine from an OVF or OVA file. Sve što je zatim potrebno jest da upišemo ime nove virtualne mašine i odaberemo OVF i VMDK datoteke koje smo prethodno preuzeli.



Slika 4.15 Kloniranje virtualne mašine

Na isti način napravio sam i treću virtualnu mašinu i time sam si pripremio okolinu za testiranje skripte.

4.3. Kôd u radu

Za samo skriptiranje koristio sam PowerShell-ovu ekstenziju, PowerShell ISE. PowerShell ISE, odnosno Integrated Scripting Environment je dodatna aplikacija koja omogućava korisniku da pokreće naredbe kao i u normalnom PowerShellu, ali ujedno može i pisati, testirati i otklanjati poteškoće (engl. debug) nastale na napisanim skriptama. Sam program je jednostavan za razumijeti i koristiti te ne zahtjeva da se ulaže puno vremena u upoznavanje samog programa, a svojom strukturom i načinom rada najviše podsjeća na Microsoft Visual Studio, stoga ako osoba ima iskustva u korištenju tog programa, PowerShell ISE mu neće biti nikakva nepoznanica.

Kôd ispod prikazuje strukturu cijele skripte sa napomenom da su svi komentari izbrisani radi jednostavnijeg pregleda same skripte:

```
$vServer = '192.168.157.99'
$vUser = 'administrator@vsphere.lab'
$vPassword = 'Pa$$w0rd'
Connect-VIServer -Server $vServer -User $vUser -Password $vPassword
function mainMenu {
    $mainMenu = ''
    while($mainMenu -ne 'q') {
        Clear-Host
        Write-Host "`n`t`t VMware ESXi script`n"
        Write-Host "Main Menu"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "1"; Write-Host
-NoNewline "]"; `
            Write-Host " Deploy"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "2"; Write-Host
-NoNewline "]"; `
            Write-Host " Revert"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "3"; Write-Host
-NoNewline "]"; `
            Write-Host " Delete"
        $mainMenu = Read-Host "`nSelection (press q to quit)"
        if($mainMenu -eq 1) {
            Deploy}
        if($mainMenu -eq 2) {
            Revert}
        if($mainMenu -eq 3) {
            Delete}}}
function Deploy {
    $Deploy = 'a'
    while($Deploy -ne '') {
        Clear-Host
        Write-Host "`n`t`t VMware ESXi script`n"
```

```
        Write-Host " Deploy"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "1"; Write-Host
-NoNewline "]"; `
            Write-Host " Deploy new Virtual Machines"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "2"; Write-Host
-NoNewline "]"; `
            Write-Host " Create Virtual Machines via Linked Clones"
        $Deploy = Read-Host "`nSelection (leave blank to go back)"
        if($Deploy -eq 1){
            $VM_count = Read-Host "Enter the number of Virtual Machines to
Deploy"
            $numcpu = "1"
            $MBram = "16"
            $MBguestdisk = "4"
            $Typeguestdisk ="Thin"
            $guestOS = "winNetStandardGuest"
            $ds = "datastore1"
            #$Cluster = "CHANGEME"
            $Folder = "VirtualMachines"
            $VM_prefix = "VM-"
            for($i=1; $i -le $vm_count; $i++){
                $x="{0:D2}" -f $i
                $VM_name= $VM_prefix + $x
                $ESXi=Get-VMHost -state connected
                Write-host "Creating $VM_name" -foreground green
                New-VM -Name $VM_name -VMHost $ESXi -numcpu $numcpu -
MemoryMB $MBram -DiskMB $MBguestdisk -DiskStorageFormat $Typeguestdisk -
Datastore $ds -GuestId $guestOS -Location $Folder
                Write-host "Power On of the VM $VM_name initiated" -
foreground green
                Start-VM -VM $VM_name -confirm:$false -RunAsync}
            Write-Host "Script execution complete." -ForegroundColor Green
```

```
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}
        if($Deploy -eq 2) {
            $VM_count = Read-Host "Enter the number of Virtual Machines to
Deploy"
            $BaseVM ="Win10Template"
            $ds="datastore1"
            $MasterSnapShot="master"
            $VM_prefix = "LinkedClone-"
            $Folder = "LinkedClones"
            for($i=1; $i -le $vm_count; $i++){
                $x="{0:D2}" -f $i
                $VM_name= $VM_prefix + $x
                $ESXi=Get-VMHost -state connected
                Write-host "Creating $VM_name" -foreground green
                New-VM -Name $VM_name -VM $BaseVM -VMHost $ESXi  -Datastore
$ds -LinkedClone -ReferenceSnapshot $MasterSnapShot -Location $Folder
                Write-host "Power On of the VM $VM_name initiated" -
foreground green
                Start-VM -VM $VM_name -confirm:$false -RunAsync}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}}}


function Revert {
    $Revert = "a"
    while($Revert -ne ""){
        Clear-Host
        Write-Host "`n`t`t VMware ESXi script`n"
        Write-Host "Revert"
```

```
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "1"; Write-Host
-NoNewline "]"; `
            Write-Host " Revert Powered Off Virtual Machines"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "2"; Write-Host
-NoNewline "]"; `
            Write-Host " Revert Powered Off Linked Clones"
        $Revert = Read-Host "`nSelection (leave blank to go back)"
        if($Revert -eq 1) {
            foreach($vm in Get-VM -Location "VirtualMachines"){
                if($vm.PowerState -eq "PoweredOff"){
                    $snap = Get-Snapshot -VM $vm | Sort-Object -Property
Created -Descending | Select -First 1
                    Set-VM -VM $vm -Snapshot $snap -Confirm:$false -
RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true) }
        if($Revert -eq 2) {
            foreach($vm in Get-VM -Location "LinkedClones"){
                if($vm.PowerState -eq "PoweredOff"){
                    $snap = Get-Snapshot -VM $vm | Sort-Object -Property
Created -Descending | Select -First 1
                    Set-VM -VM $vm -Snapshot $snap -Confirm:$false -
RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true) }}}


function Delete {
    $Delete = "a"
    while($Delete -ne ""){
```

```
        Clear-Host
        Write-Host "`n`t`t VMware ESXi script`n"
        Write-Host "Delete"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "1"; Write-Host
-NoNewline "]"; `
            Write-Host " Delete Powered off Virtual Machines"
        Write-Host -NoNewline "`n["; Write-Host -NoNewline "2"; Write-Host
-NoNewline "]"; `
            Write-Host " Delete Powered off Linked Clones"
        $Delete = Read-Host "`nSelection (leave blank to go back)"
        if($Delete -eq 1) {
            foreach($vm in Get-VM -Location "VirtualMachines"){
                if($vm.PowerState -eq "PoweredOff"){
                    Remove-VM -VM $vm -DeletePermanently -Confirm:$false
-RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}
        if($Delete -eq 2) {
            foreach($vm in Get-VM -Location "LinkedClones"){
                if($vm.PowerState -eq "PoweredOff"){
                    Remove-VM -VM $vm -DeletePermanently -Confirm:$false
-RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}}}
mainMenu
```

Kôd 4.1 Skripta za automatizaciju deploy-a, revert-a i brisanja virtualnih mašina

Kao što se može vidjeti, sama skripta je poprilično velika, stoga ćemo je podijeliti na četiri cjeline i objasniti svaku cjelinu kako bi nam bilo lakše razumijeti što sama skripta zapravo radi te kako to radi.

4.3.1. Main Menu

Main Menu, odnosno glavni meni najvažniji je dio skripte jer korisniku prikazuje sve mogućnosti same skripte. Sama skripta započinje sa varijablama vServer, vUser i vPassword koje od korisnika očekuju da upiše IP servera, korisničko ime i lozinku koje će se koristiti za povezivanje na ESXi host. Samu primjenu vidimo već u sljedećem redu gdje se gore navedene varijable koriste sa Connect-VIServer komandletom:

```
Connect-VIserver -Server $vServer -User $vUser -Password $vPassword
```

Sljedeći dio kôda je sam glavni meni napisan u obliku funkcije. Funkcije u PowerShell-u su liste izjava koje imaju ime koje im mi dodjelimo. Funkcioniraju na isti način kao i cmdleti: kada pozovemo funkciju, ona izvrši niz izjava koje smo ranije definirali. Unutar same funkcije kreirana je nova varijabla mainMenu koja se koristi unutar while petlje, a sama petlja provjerava je li korisnik upisao slovo „q“ za izlaz iz skripte. Sve dok korisnik ne upiše slovo „q“ koji se zatim dodjeljuje varijabli mainMenu pomoću `Read-Host` cmdleta, kôd unutar while petlje će se neprestano izvoditi, odnosno u ovom slučaju prikazivati glavni izbornik i sve njegove opcije.

```
function mainMenu {
    $mainMenu = ''
    while($mainMenu -ne 'q'){
        KOD ZA PRIKAZ GLAVNOG IZBORNIKA
        $mainMenu = Read-Host "`nSelection (press q to quit)"}
```

Kôd 4.2 mainMenu funkcija

Zadnji dio Main Menu funkcije je parametar koji provjerava koju opciju, točnije, broj je upisao korisnik. Ukoliko je upisan broj „1“, pokreće se funkcija „Deploy“, ako je upisan broj „2“ funkcija Revert i broj „3“ funkcija Delete.

4.3.2. Deploy

Funkcija Deploy započinje na isti način kao i mainMenu funkcija. Kreirana je varijabla Deploy i ponovno se koristi while petlja za prikaz podopcija (engl. sub-options) koju Deploy funkcija nudi sve dok korisnik ne upiše traženi znak čime se prekida izvršavanje te while petlje. U mainMenu funkciji znak za izlazak iz skripte je „q“, no ovaj puta umjesto da se na isti način izlazi iz Deploy-a, koristi se tipka Enter kako bi se vratilo na prethodnu „opciju“, odnosno na glavni meni.

```
function Deploy {

    $Deploy = 'a'

    while($Deploy -ne '') {

        KOD ZA PRIKAZ PODOPCIJA

        $Deploy = Read-Host "`nSelection (leave blank to go back)"}
```

Kôd 4.3 Deploy while petlja

Ponovno imamo dvije podopcije, kreiranje novih virtualnih mašina i kreiranje linkanih klonova. Nakon što se jedna od tih podopcija odabere, od korisnika se traži da upiše koliko novih virtualnih mašina ili klonova želi. Pri objašnjavanju koristit ću samo kôd za kreiranje novih virtualnih mašina jer su kôdovi obju podopcija u osnovi isti, stoga nije potrebno objašnjati obje podopcije. Nakon upisivanja željenog broja, započinje se sa kreiranjem novih virtualnih mašini pomoću sljedeće linije koda:

```
New-VM -Name $VM_name -VMHost $ESXi -numcpu $numcpu -MemoryMB $MBram -
DiskMB $MBguestdisk -DiskStorageFormat $Typeguestdisk -Datastore $ds -
GuestId $guestOS
```

Kôd 4.4 kôd za kreiranje novih virtualnih mašina

Naravno, ta linija koda kreira samo jednu virtualnu mašinu, stoga je potrebno koristiti for petlju koja će brojati od 1 do broja upisanog od strane korisnika kako bi se kreiralo točno onoliko virtualnih mašina koliko je i korisnik zatražio. Dakle, cijela for petlja izgleda ovako:

```
for($i=1; $i -le $vm_count; $i++){
                $x=”{0:D2}” -f $i
                $VM_name= $VM_prefix + $x
                $ESXi=Get-VMHost -state connected
                Write-host “Creating $VM_name” -foreground green
```

```
                New-VM -Name $VM_name -VMHost $ESXi -numcpu $numcpu -
MemoryMB $MBram -DiskMB $MBguestdisk -DiskStorageFormat $Typeguestdisk -
Datastore $ds -GuestId $guestOS

                Write-host "Power On of the VM $VM_name initiated" -
foreground green

                Start-VM -VM $VM_name -confirm:$false -RunAsync}
```

Kôd 4.5 for petlja za kreiranje virtualnih mašina

Unutar same for petlje postoje par opcionalnih naredbi koje „uljepšavaju“ izvršavanje skripte, a to je varijabla *$x* te *Write-Host* naredbe. Svrha *$x* varijable je da formatira numeriranje svake virtualne mašine tako da svaka mašina u svom imenu ima minimalno dva broja. Dakle, ako kreiramo 5 virtualnih mašina, njihovo ime bit će VM-01, VM-02 itd. U drugu ruku, *Write-Host* cmdlet služi samo kao dodatna informacija korisniku skripte da je samo izvršavanje skripte proteklo uspješno.

4.3.3. Revert

Revert funkcija ima identičnu strukturu kao i Deploy funkcija, te kao i Deploy ima dvije podopcije. Prva podopcija je da se revertaju virtualne mašine koje su isključene dok je druga podopcija revert isključenih linkanih klonova.

Kod ispod prikazuje kod samo za prvu podopciju jer su obje podopcije u osnovi iste, uz minimalne razlike:

```
if($Revert -eq 1) {
            foreach($vm in Get-VM -Location "VirtualMachines") {
                if($vm.PowerState -eq "PoweredOff"){
                    $snap = Get-Snapshot -VM $vm | Sort-Object -Property
Created -Descending | Select -First 1
                    Set-VM -VM $vm -Snapshot $snap -Confirm:$false -
RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true) }
```

Kôd 4.6 kôd za prvu podopciju

Kao što se može vidjeti, Revert funkcija koristi foreach cmdlet koji funkcionira na sličan način kao i for petlja, samo što automatski prolazi kroz svaki parametar dobiven kroz određeni argument, u ovom slučaju to je Get-VM cmdlet. Za obje podopcije koristi se if operator kako bi se provjerilo u kojem stanju su virtualne mašine, odnosno jesu li upaljene ili ugašene. U ovom slučaju, if operator provjerava stanje svake virtualne mašine i ako je ugašena, radi revert virtualne mašine korištenjem snapshota.

4.3.4. Delete

Delete funkcija funkcionira na isti način kao i Revert, razlika je jedino kôd koji se izvršava. Postoje dvije podopcije, prva podopcija briše samo ugašene virtualne mašine dok druga podopcija briše samo ugašene linkane klonove:

```
if($Delete -eq 1) {
            foreach($vm in Get-VM -Location "VirtualMachines"){
                if($vm.PowerState -eq "PoweredOff"){
                    Remove-VM -VM $vm -DeletePermanently -Confirm:$false
-RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}
        if($Delete -eq 2) {
            foreach($vm in Get-VM -Location "LinkedClones"){
                if($vm.PowerState -eq "PoweredOff"){
                    Remove-VM -VM $vm -DeletePermanently -Confirm:$false
-RunAsync}}
            Write-Host "Script execution complete." -ForegroundColor Green
            Write-Host "Press any key to return to the previous menu..."
-ForegroundColor Yellow
            [void][System.Console]::ReadKey($true)}}}
```

Kôd 4.7 Obje podopcije Delete funkcije

Zaključak

Virtualizacija kao tehnologija uvelike je pomogla organizacijama u uštedi novca, prostora i vremena, no moguće je još dodatno uštediti vrijeme automatiziranjem te iste tehnologije korištenjem skripti. Kroz ovaj rad ja sam koristio PowerShell ISE za pisanje svoje skripte za osnovno upravljanje virtualnim mašinama.

Prije kreiranja same skripte bilo je potrebno upoznati se sa PowerShell-om i njegovim objektnim načinom rada. Iako je PowerShell vrlo moćan alat koji omogućuje da se koristi na vrlo kompleksan način, vrijeme koje sam trebao uložiti da ga naučim koristiti na osnovnoj razini bilo je izrazito kratko, što je velika prednost u odnosu na konkurenciju. Uz vrlo kvalitetnu dokumentaciju i pregršt tutoriala vrlo je lako naći odgovor na svako pitanje koje bi osoba mogla imati te definitivno preporučam upoznati se sa korištenjem PowerShell-a. Čak i kada sam naišao na greške prilikom pisanja svoje skripte, greška je bilo jasno označena i objašnjena što je uzrok greške te je bilo vrlo lako naći rješenje za uklanjanje same greške pretraživanjem po dokumentaciji ili na internetu.

Sljedeća komponenta koju sam morao koristiti jesu VMware-ovi virtualizacijski programi, Workstation i vSphere. Oba programa odlična su u obavljanju svojeg posla, to jest virtualizacije, a svojim intuitivnim, ali kompleksnim i opširnim grafičkim sučeljem nude uvid u svaki detalj koje bi korisnika moglo zanimati. Valja napomenuti da sam za pisanje svoje skripte mogao koristiti i Microsoftovu virtualizacijsku tehnologiju, Hyper-V, no razlog zašto sam se odlučio za VMware je taj što sam istraživanjem pronašao da VMware-ova tehnologija zauzima 75% tržišta, odnosno da se VMware koristi u 75% organizacija koje koriste virtualizaciju dok je Hyper-V korišten u samo 17% organizacija, stoga mi je logično bilo opredijeliti se za onu virtualizacijsku tehnologiju sa kojom ću se u bliskoj budućnoj vjerovatnije sresti. Još jedan od faktora bilo je prijašnje negativno iskustvo sa HyperV-em. Često mi se znalo događati da bi mi se datoteka virtualne mašine korumpirala ako sam ju više puta vraćao u prijašnje stanje te sam morao isponova napraviti drugu ili sam imao problema sa ostvarivanjem komunikacije između virtualne mašine i fizičkog računala kao i još nekolicinu drugih manjih problema, stoga mi se VMware činio kao najbolja opcija.

Pisanje same skripte zahtjevalo je malo više vremena jer sam se morao upoznati sa logikom pisanja skripti, ali jednom kada sam shvatio na koji način funkcionira bilo je vrlo lagano nastaviti pisati ostatak skripte. Čak i kada sam naišao na problem prilikom pisanja ili nisam

znao kako napraviti neku od funkcija, pretraživanjem po internetu brzo bih došao do željenog odgovora. Uz to, veliku pomoć pri testiranju skripte bio je vizualni log unutar vSphere sučelja koji mi je uvelike pomogao da brzo uvidim i otklonim sve greške koje su se nalazile u mom kodu. Sama skripta, iako jednostavna, po mome je mišljenju vrlo učinkovita u onome što radi i smatram da bi se budućom nadogradnjom mogla uvelike poboljšati i ostvariti još veću uštedu vremena.

Za kraj bi htio istaknuti isplativost i učinkovitost pisanja i korištenja skripti. Uspoređivanjem vremena koje mi je bilo potrebno da samostalno kreiram tri virtualne mašine i vremena da skripta napravi tri virtualne mašine jasno se vidi koliko značajno skripta uštedi vrijeme a uz to potpuno eliminira mogućnost ljudske pogreške. Naravno, ova ušteda vremena eksponencijalno raste kako se broj virtualnih mašina koje se treba kreirati povećava, a isto pravilo vrijedi i za brisanje ili vraćanje u prijašnje stanje. Uz to, od velikog je značaja i poznavanje kako pisati skripte i razumijevanje što skripta radi. Tim znanjem osoba može samostalno prilagoditi skriptu prema svojim potrebama ili čak nadodati nove funkcionalnosti stoga čvrsto preproučam da se upozna sa barem jednim skriptnim jezikom.

Popis kratica

IT	*Information Technologies*	informacijske tehnologije
CLI	*Command Line Interface*	komandno linijsko sučelje
ESXi	*Elastic Sky X Integrated*	Elastic Sky X Integrated
TUI	*Text User Interface*	tekstualno grafičko sučelje
DSC	*Desired State Configuration*	željeno stanje konfiguracije
WMI	*Windows Management Instrumentation*	komandno linijsko sučelje
GUI	*Graphical User Interface*	grafičko korisničko sučelje
OVF	*Open Virtualization Format*	otvoreni virtualizacijski format
XML	*EXtensible Markup Language*	EXtensible Markup Language
VMDK	*Virtual Machine Disk*	disk virtualne mašine
OVA	*Open Virtualization Appliance*	otvoreni uređaj za virtualizaciju
ISE	*Integrated Scripting Environment*	integrirano skriptno okruženje
IP	*Internet Protocol*	internet protokol

Popis slika

Slika 2.1 Primjer PowerShell strukture naredbi 5

Slika 2.2 Help parametar 6

Slika 2.3 Primjer korištenja varijable 7

Slika 2.4 Read-Host cmdlet 7

Slika 2.5 Read host korišten u kombinaciji sa varijablom 8

Slika 2.6 Write-Host princip rada 8

Slika 2.7 Write output princip rada 9

Slika 3.1 WMWare Workstation Pro početni zaslon 11

Slika 3.2 VMware ESXi početni zaslon 13

Slika 4.1 Infrastruktura korištena za tesitranje skripte 17

Slika 4.2 Pokretanje PowerShell-a kroz Windows Start meni 18

Slika 4.3 Find-Module cmdlet 18

Slika 4.4 Install-Module cmdlet 19

Slika 4.5 Get-Module cmdlet 19

Slika 4.6 Connect-VIServer prva greška 19

Slika 4.7 Import-Module prva greška 20

Slika 4.8 Set-ExecutionPolicy cmdlet 20

Slika 4.9 Connect-VIServer druga greška 20

Slika 4.10 Connect-VIServer povezivanje 21

Slika 4.11 VMware vSphere web login sučelje 21

Slika 4.12 VMware vSphere web početni zaslon 22

Slika 4.13 Kreiranje virtualne mašine 23

Slika 4.14 Export virtualne mašine 24

Slika 4.15 Kloniranje virtualne mašine 25

Popis tablica

Tablica 2.1 Usporedba PowerShell i PowerShell Core-a 3

Tablica 2.2 Sistemski zahtjevi za PowerShell 5.1 4

Tablica 3.1 Sistemski zahtjevi VMware Workstation Pro 11

Tablica 3.2 PowerCLI sistemski zahtjevi 15

Tablica 3.3 Set softvera potrebnog za uspješnu instalaciju PowerCLI-a 16

Popis kôdova

Kôd 4.1 Skripta za automatizaciju deploy-a, revert-a i brisanja virtualnih mašina..... 30

Kôd 4.2 mainMenu funkcija..... 31

Kôd 4.3 Deploy while petlja..... 32

Kôd 4.4 kôd za kreiranje novih virtualnih mašina..... 32

Kôd 4.5 for petlja za kreiranje virtualnih mašina..... 33

Kôd 4.6 kôd za prvu podopciju..... 33

Kôd 4.7 Druga podopcija Delete funkcije..... 34

Literatura

[1] JONES, D; HICKS, J. *Learn Windows PowerShell in a Month of Lunches*. ISBN-13: 978-1-61729-416-7

[2] HTTPS://PUBS.VMWARE.COM/VSPHERE-51/INDEX.JSP#COM.VMWARE.VSPHERE.SCRIPTING.DOC/GUID-3C87E067-CF9B-4348-8F27-C278439A71EF.HTML, 2018.

[3] HTTPS://WWW.VMWARE.COM/SUPPORT/DEVELOPER/WINDOWSTOOLKIT/WINTK40/DOC/VIWIN_ADMG.PDF, 2018.

[4] HTTP://THESOLVING.COM/VIRTUALIZATION/HOW-TO-INSTALL-AND-CONFIGURE-VMWARE-POWERCLI-VERSION-10/, 2018.

[5] HTTPS://WWW.MICHELLELAVERICK.COM/2018/04/SETTING-UP-VMWARE-POWERCLI-10-X/, 2018.

[6] HTTPS://DOCS.MICROSOFT.COM/EN-US/POWERSHELL/SCRIPTING/SETUP/INSTALLING-WINDOWS-POWERSHELL?VIEW=POWERSHELL-6, 2018.

[7] http://www.mustbegeek.com/difference-between-vsphere-esxi-and-vcenter/, 2018.

[8] https://docs.VMware.com/en/VMware-Workstation-Pro/12.0/com.VMware.ws.using.doc/GUID-A471ADE7-3F03-462A-BA6D-F7BE3C9204EC.html, 2018.

[9] http://www.VMwarearena.com/what-is-VMware-vsphere-beginners-guide-to-VMware-virtualization/, 2018.

[10] https://www.petri.com/more-choices-powershell, 2018.

[11] https://www.itprotoday.com/powershell/prompting-user-input-powershell, 2018.