Automatizovaný deployment linuxových serverů a aplikací na platformě Hyper-V

Bc. Milan Cibulka

Diplomová práce
2021

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav informatiky a umělé inteligence

Akademický rok: 2020/2021

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Milan Cibulka**
Osobní číslo:	**A17770**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Automatizovaný deployment linuxových serverů a aplikací na platformě Hyper-V**
Téma práce anglicky:	**The Automated Deployment of Linux Servers and Applications on the Hyper-V Platform**

Zásady pro vypracování

1. Vypracujte literární rešerší na dané téma.
2. Navrhněte automatizované technické řešení podle požadavků.
3. Zdůvodněte výběr jednotlivých komponentů technického řešení.
4. Realizujte a otestujte výsledné technické řešení.
5. Připravte přehledný uživatelský manuál nasazení serverů a aplikací.
6. Věnujte pozornost zabezpečení.

Forma zpracování diplomové práce: **Tištěná/elektronická**

Seznam doporučené literatury:

1. Ansible. Ansible [online]. 2020 [cit. 2020-11-26]. Dostupné z: https://www.ansible.com/
2. Vault. Vault [online]. 2020 [cit. 2020-11-26]. Dostupné z: https://www.vaultproject.io/
3. Vagrant. Vagrant [online]. 2020 [cit. 2020-11-26]. Dostupné z: https://www.vagrantup.com/
4. Design and use of virtualization technology in cloud computing [online], 2017. Hershey PA, USA: IGI Global [cit. 2020-11-26]. ISBN 9781522527862. Dostupné z: doi:10.4018/978-1-5225-2785-5
5. Hyper-V on Windows Server, 2016. Microsoft [online]. [cit. 2020-11-26]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-on-windows-server
6. HOCHSTEIN, Lorin a René MOSER. Ansible: up and running: automating configuration management and deployment the easy way. Second edition. Beijing: O'Reilly, 2017. ISBN 978-1-4919-7980-8.
7. Debian GNU/Linux Installation Guide: Appendix B. Automating the installation using preseeding. Debian.org [online]. Debian, 2019 [cit. 2020-12-03]. Dostupné z: https://www.debian.org/releases/buster/amd64/apb.en.html
8. Debian Wiki: PXEBootInstall: Installing Debian using network booting. Debian Wiki [online]. Debian, 2019 [cit. 2020-12-03]. Dostupné z: https://wiki.debian.org/PXEBootInstall

Vedoucí diplomové práce: **doc. Ing. Martin Sysel, Ph.D.**
Ústav počítačových a komunikačních systémů

Konzultant diplomové práce: **Ing. Ivan Masár**
Ústav informatiky a umělé inteligence

Datum zadání diplomové práce: **15. ledna 2021**
Termín odevzdání diplomové práce: **17. května 2021**

UNIVERZITA TOMÁŠE BATI VE ZLÍNĚ
Fakulta aplikované informatiky
DĚKAN
L.S.

doc. Mgr. Milan Adámek, Ph.D. v.r.
děkan

prof. Mgr. Roman Jašek, Ph.D. v.r.
ředitel ústavu

Ve Zlíně dne 15. ledna 2021

Prohlašuji, že

- beru na vědomí, že odevzdáním diplomové práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že diplomová práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk diplomové práce bude uložen v příruční knihovně Fakulty aplikované informatiky Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má UTB ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – diplomovou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování diplomové práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky diplomové práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem diplomové práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na diplomové práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne 17.5.2021

Bc. Milan Cibulka, v. r.
podpis studenta

ABSTRAKT

Tato Diplomová práce se zabývá problematikou automatizace nasazování virtuálních serverů na platformě Microsoft Hyper-V. V teoretické části jsou popsány obecné možnosti virtualizace výpočetního prostředí, principy automatické instalace operačního sytému Linux a uveden přehled nástrojů pro automatizaci správy serverové infrastruktury. Praktická část se zaměřuje na návrh a realizaci automatizovaného nasazení virtuálních strojů v prostředí Microsoft Hyper-V s ohledem zejména na využití Knihovnou Univerzity Tomáše Bati ve Zlíně. Součástí práce je také přehledný manuál k nasazení virtuálního serveru.

Klíčová slova: Hyper-V (software), virtuální stroj, Packer, Ansible, Linux, virtualizace (počítače)

ABSTRACT

This diploma thesis deals with the issue of an automation of a virtual server deployment on the Microsoft Hyper-V platform. The theoretical part describes the general possibilities of a computing environment virtualization, the principles of an automatic installation of the Linux operating system and provides an overview of tools for the automating server infrastructure management. The practical part focuses on the design and the implementation of the automated deployment of virtual machines in the Microsoft Hyper-V environment, regarding to be used especially in the library of the Tomas Bata University in Zlín. In addition, this thesis includes also a clear manual for deploying the virtual server.

Keywords: Hyper-V (software), Virtual Machine, Packer, Ansible, Linux, virtualization (computers)

Na tomto místě bych rád poděkoval vedoucímu práce panu doc. Ing. Martinu Syslovi, Ph.D za odborné vedení, čas a připomínky při zpracovávání mojí diplomové práce. Také bych chtěl velmi poděkovat panu Ing. Ivanu Masárovi za odborný dohled a konzultace při realizaci praktické části této práce.

Prohlašuji, že odevzdaná verze diplomové práce a verze elektronická nahraná do IS/STAG jsou totožné.

OBSAH

ÚVOD

Virtualizace výpočetního prostředí je velmi populární, především protože nabízí efektivní využití fyzických prostředků, čímž snižuje náklady na provoz a výdaje spojené s pořizováním dalšího hardwaru. Mezi další klady pak patří zjednodušení správy, snadná migrace služeb mezi fyzickými servery bez výpadku, rychlejší zálohování a obnova dat nebo například snadné škálování.

Hlavním cílem této práce je vytvoření automatizovaného řešení pro proces nasazování virtuálních strojů v prostředí Microsoft Hyper-V, které by mělo nalézt své uplatnění v provozu služeb Knihovny Univerzity Tomáše Bati ve Zlíně. Automatizace zde přinese benefity v podobě rychlejšího a snadnějšího vytváření nových serverů a omezí chyby způsobené lidským faktorem.

Teoretická část diplomové práce popisuje typy a metody virtualizace výpočetního prostředí a uvádí stručný přehled virtualizačního softwaru. Dále se zabývá automatickou instalací operačního systému Linux. Následně věnuje pozornost nástrojům pro práci s platformou Microsoft Hyper-V. Poslední kapitola v teoretické části se věnuje nástrojům pro automatizaci správy serverové infrastruktury.

Praktická část práce je rozdělena na tři kapitoly. V první z nich jsou dle stanovených požadavků navrženy dvě varianty řešení pro automatizaci procesů spojených s nasazením virtuálních serverů na platformě Microsoft Hyper-V. Další kapitola se zabývá realizací, při které je vyhodnocena vhodnější varianta řešení. Podrobně popisuje tvorbou znovupoužitelného obrazu virtuálního stroje pomocí technologie Packer, nasazení při použití nástroje Ansible a komplikace, které vznikly při realizaci. Poslední kapitola je věnována bezpečnosti.

I. TEORETICKÁ ČÁST

1 VIRTUALIZACE VÝPOČETNÍHO PROSTŘEDÍ

Virtualizace v podstatě vytváří softwarovou iluzi fyzických prostředků, ať už na úrovni jednotlivých komponent nebo celého počítače. Což umožňuje vytvářet více nezávislých prostředí při sdílení jednoho fyzického zdroje. [1]

Pojem virtualizace se v informatice objevil už v 60. letech 20. století, kdy společnost IBM spustila na svých sálových počítačích operační systém OS/370, který umožňoval jeden fyzický počítač rozdělit na několik virtuálních strojů (VM, Virtual Machine) s vlastním plnohodnotným operačním systémem. Důvodem pro vznik virtualizace byla především vysoká pořizovací cena hardwaru a tehdejší systém sdílení počítačů více uživateli sebou nesl velká rizika poškození dat a programů mezi uživateli. Jeden uživatel také mohl znemožnit práci ostatním, např. pokud jeho konání zapříčinilo zhroucení celého počítače nebo jeho program nadměrné vytížil všechny hardwarové prostředky. [1][2]

S nástupem levných osobních počítačů zájem o virtualizaci výrazně poklesl, ale s růstem jejich výkonu nabízí virtualizace lepší využití hardwaru a ve výsledku nižší náklady na provoz. Ve vysoké míře nachází uplatnění v oboru služeb poskytujících webhosting, kde umožnuje velmi rychle nabídnout klientům požadované prostředky, bez nutnosti pořizování nového hardwaru. Důležitou roli hraje také při tvorbě softwaru. Umožňuje udržovat důkladně oddělené vývojové prostředí nebo testování při použití různých operačních systému a konfiguracích. [1][3]

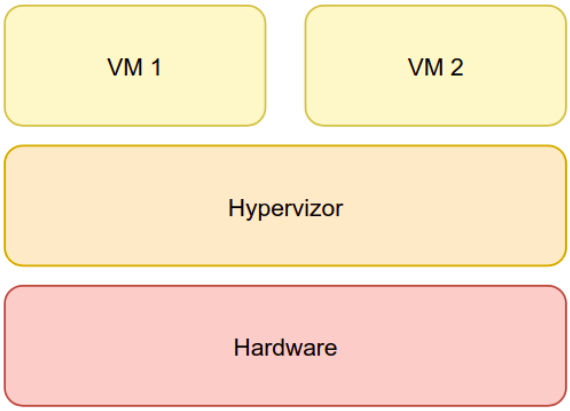
Ve spojitosti s virtualizací je často spojován výraz cloud. Jedná se o síť vzájemně propojených serverů poskytujících různé služby, které navenek působí jako jeden celek. Jedním z možných řešení může být cloud veřejný, kde jsou prostředky sdíleny veřejně pomocí internetu. Naopak privátní nabízí služby pouze v rámci soukromé sítě. Kombinací vzniká hybridní cloud, vhodný především když uživatel svá citlivá data chce udržet ve vlastní soukromé síti ale zároveň některé služby využívat veřejně. Velkou výhodou je flexibilita, se kterou lze měnit využívané služby podle potřeby. Není třeba investice do vlastních serverů. Poskytovatel si účtuje poplatky pouze v míře, v jaké byl cloud využívá zákazníkem. Veřejný cloud s distribuovanými servery po celém světě může být zajímavým prostředkem k expanzi firmy na zahraničí trhy. Z pohledu bezpečnosti pak může nabídnout i lepší zabezpečení, než by nabízelo běžné firemní řešení, jelikož cloud většinou poskytují větší týmy specialistů a například zálohování dat bývá součástí základní nabídky. [4][5][6]

1.1 Typy virtualizace

Základní software, jenž umožňuje provoz VM je označován jako hypervizor nebo také Virtual Machine Monitor. Vytváří vrstvu mezi virtuálními stroji a hardwarovou vrstvou fyzického počítače, zároveň udržuje oddělené výpočetní prostředí pro jednotlivé hostující systémy. V závislosti na tom, zda běží přímo na fyzické vrstvě nebo až v prostředí operačního sytému jsou rozlišovány dva typy tzv. nativní nebo hostovaný. [7][8]

1.1.1 Typ 1 (nativní)

V anglické literatuře bývá označován také jako bare metal (holé železo), což napovídá že je spouštěn bezprostředně na hardwaru, nahrazuje operační systém a má přístup přímo k fyzickým prostředkům. Ty následně poskytuje hostujícím virtuálním strojům. Hypervizor typu 1 vyžaduje podporu hardwarové akcelerace, která umožňuje provádět složité operace spojené se správou virtuálních prostředků. Tento typ nabízí nejlepší využití výkonu hardwaru, proto nachází uplatnění především v serverovém prostředí. Mezi významné technologie využívající tento přístup patří Microsoft Hyper-V, KVM nebo VMware vSphere. [9][10]

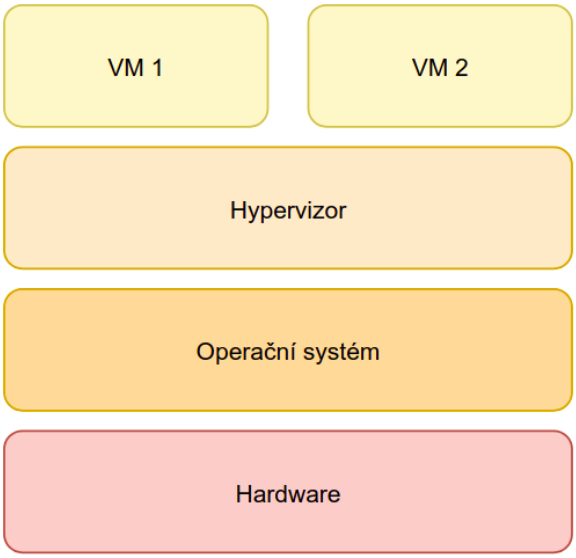


Obrázek 1 – Architektura virtualizace s využitím hypervizoru typu 1 (nativní) [vlastní]

1.1.2 Typ 2 (hostovaný)

Je spouštěn v prostředí operačního systému, jako softwarová vrstva nebo aplikace, tudíž nemá přímý přístup k hardwaru. Všechny aktivity musí být zpravovány přes hostitelský operační systém a dochází k určité ztrátě výkonu. Takové řešení, ale může být výhodné v případě kdy uživatel chce provozovat více operačních systémů na jednom osobním počítači.

Typickými nástroji s hypervizorem tohoto typu jsou VMware Workstation nebo Oracle VirtualBox. [9][10]



Obrázek 2 – Architektura virtualizace s využitím hypervizoru typu 2 (hostovaný)
[vlastní]

1.2 Metody virtualizace

Virtualizaci je možné provádět na různých úrovních, může probíhat na úrovni celého virtuálního stroje, ale může se jednat jen o některé jeho komponenty např. operační paměť, procesor nebo uložiště. Případně se nemusí jednat vůbec o virtualizaci hardwaru, ale pouze o virtualizaci na úrovni operačního systému. S tím se váže použití specifických metod, za pomoci, kterých toho lze dosáhnout. Každá má své výhody i nevýhody a nachází uplatnění ve svém oboru použití. [11][12][13]

1.2.1 Emulace

Při emulaci dochází k úplné softwarové virtualizaci, kdy jsou všechny fyzické komponenty včetně procesoru nahrazeny virtuálními. Díky tomu nabízí jedinečnou možnost provozu aplikací, které hostitelský hardware nepodporuje. Nejčastěji se jedná o procesorovou nekompatibilitu, kdy program zkompilovaný pro jednu procesorovou architekturu nelze spouštět v kombinaci s procesorem založeným na jiné architektuře. Emulace pracuje na principu překladu jednotlivých instrukcí, které programu slouží k práci s hardwarem. Toto řešení sice

umožňuje pracovat i s fyzicky nedostupnými prostředky, ale za cenu vysokých nároků na výpočetní výkon. [11][12]

Nutnost překladu lze do jisté míry zrychlit technikou zvanou dynamický překlad, při níž se instrukce analyzují ještě před jejich spuštěním bezpečné instrukce tak je umožňuje spouštět bez překladu a citlivé instrukce před jejich provedením překládá. Přeložené instrukce následně mohou být uloženy do mezipaměti a urychlit opakované spuštění. [11]

1.2.2 Virtualizace s hardwarovou asistencí

Tento druh virtualizace tak jako emulace vytváří iluzi fyzických prostředků pro virtuální stroje, od emulace se však liší tím, že slouží k vytváření virtuálních strojů, které disponují typově stejnými prostředky jako hostující fyzický stroj. Hypervizor využívá podpory virtualizace ze strany hardwaru a řídí přidělování fyzických zdrojů virtuálním strojům. Přední výrobci procesorů Intel a AMD podporu virtualizace u svých produktů označují jako Intel VT-x a AMD-V, podpora je ale nutná i ze strany dalších komponent. Hostované virtuální stroje jsou od sebe dokonale izolovány a také nemají možnost rozeznat, že běží ve virtuálním prostředí. Tato metoda díky nízké režii nabízí velmi dobré zhodnocení výkonu fyzických prostředků. [11][14]

1.2.3 Paravirtualizace

Stejně jako Virtualizace s hardwarovou asistencí slouží k vytváření virtuálních strojů, které jsou hardwarově podobné tomu fyzickému. Avšak své využití nachází v případě, že virtualizace na úrovni fyzických prostředků není dostupná. Hypervizor tak hostujícím virtuálním strojům nabízí speciální aplikační rozhraní (API) pomocí kterého mohou přistupovat k hardwaru. Tato metoda však vyžaduje modifikací jádra operačního systému virtuálních strojů případně instalaci specifických ovladačů. Díky tomu tyto stroje získávají informaci o tom, že běží ve virtuálním prostředí, mezi sebou jsou však udržovány v dokonalé izolaci. Paravirtualizace dosahuje nejnižší míry režie z nabízených metod. [11][14][15]

1.2.4 Virtualizace na úrovni operačního systému

Pro tuto metodu je specifické že již nevirtualizuje na úrovni hardwaru, ale virtualizovaná prostředí běží nad společným jádrem hostitelského operačního systému, který má přístup k fyzickým prostředkům, tudíž není nutné vytvářet virtuální zařízení ani speciální aplikační rozhraní. Tím umožňuje velmi efektivní využití výkonu srovnatelné s paravirtualizací,

hostující operační systém však nevyžaduje žádnou úpravu ani instalaci specifických ovladačů. K provozu je nutná pouze instalace virtualizačního softwaru v prostředí hostitele. [13][16]

Nevýhodou muže být že metoda umožňuje provoz pouze instancí operačního systému sejného typu jako je hostitelský, v případě Linuxu však dovoluje provoz různých distribucí, neboť sdílené je jen jádro operačního systému. Virtualizační software spravuje přidělované prostředky jednotlivým virtuálním prostředím a zajišťuje izolaci mezi nim, ale nezabezpečuje již takovou míru izolace od hostitelského systému, jako je tomu u ostatních metod virtualizace. S tím roste i riziko výskytu bezpečnostních chyb. [13][16][17]

1.3 Virtualizační software

Pro realizaci virtualizace výpočetního prostředí je nutný výběr vhodného softwaru. Vývojem nástrojů v této oblasti se zabývá mnoho společností a nabízí širokou škálu produktů. Některé jsou poskytovány zdarma u jiných je třeba zakoupit licenci. Podstatné rozdíly se nachází v oblastech využití, kde nabízí určitou specializací např. pro virtualizaci desktopů, serverů nebo cloud. Následující přehled představuje aktuálně nejrozšířenější technologie. [13][18][19]

1.3.1 VMware

Společnost VMware byla jednou z prvních organizací, která začala vyvíjet virtualizační software. Nyní se jedná o nevětší společnost na trhu se softwarem pro virtualizaci. Nabízí opravdu velké množství produktů, řešení a školení. Převážná část distribuce je komerční, ale některé nástroje jsou v omezené verzi dostupné zdarma. [20][21]

Pro serverovou virtualizaci slouží WMware vSphere, což je ucelený placený balík nástrojů pro provoz serverové infrastruktury. Obsahuje sadu prostředků pro správu a monitorování. Platforma stojí na nativním hypervizoru (typ 1) vSphere Hypervisor, též označován jako VMware ESXi. Lze jej pořídit i v samostatné verzi zdarma, přináší sebou však limitaci v počtu procesorových jader na virtuální stroj, a to maximálně 8. Zároveň neobsahuje některé funkcionality nebo možnost vytvářet clustery. [22]

Desktopovou virtualizaci zastupují nástroje VMware Workstation pro Windows nebo Linux a VMware Fusion pro macOS. Na rozdíl od serverového řešení jsou spouštěny v prostředí operačního sytému a využívají hostovaný hypervizor (typ 2). Oba produkty je možné pořídit ve verzi Pro nebo Player. Verze Pro je plnohodnotným komerčním produktem a umožňuje

využívat všechny dostupné funkce. Zdarma lze získat pro osobní užití verzi Player, která je omezená v některých pokročilých funkcích, jako je například simulace virtuální sítě, vytváření replik nebo šifrování virtuálních počítačů. [22]

VMware nabízí také virtualizaci na vlastní cloudové platformě VMware Cloud Foundation, nebo nástroje pro populární cloudové řešení jako jsou Amazon Web Services, Microsoft Azure, Google Cloud Platform a další. [22]

1.3.2 Microsoft Hyper-V

Microsoft vyvíjí virtualizační technologii Hyper-V která je aktuálně dostupná u systémů Windows server a 64bitových verzí systému Windows 10, kromě základní licence Home edition, u které možnost funkce Hyper-V není podporována, avšak při použití neoficiálních metod a s porušením smluvních podmínek lze Hyper-V provozovat i zde. [23] V nabídce je také samostatný systém Hyper-V Server, který je zdarma a neobsahuje operační systém Windows s grafickým uživatelským prostředím (GUI). Hypervizor běží přímo na fyzickém stroji, proto se řadí do kategorie nativních (typ 1). [24]

Společnost Microsoft dále pak rozvinula technologii Hyper-V do podoby produktu Microsoft Azure Hypervisor na kterém provozuje cloudovou službu Azure. [25]

1.3.3 Xen Project

Tým Xen Project je globální komunita zabývající se vývojem open source virtualizační platformy Xen Project jejíž základem je hypervizor typu 1. Původně byl vyvinut společností XenSource kterou následně převzala firma Citrix a hypervizor použila pro vlastní komerční produkt Citrix Hypervisor (dříve XenServer). Samotný hypervizor nadále zůstal dostupný v podobě otevřeného kódu, podílet na vývoji se může kdokoliv, Xen Project na svém webu přímo nabízí řadu školících materiálů pro nové přispěvatele. Využití nachází i v dalších komerční řešeních například Oracle VM Server, nebo Amazon Web Services. [26][27]

1.3.4 Kernel-based Virtual Machine (KVM)

KVM je virtualizační technologie jenž je zabudována do linuxového jádra od verze Linux 2.6.20. Neběží na operačním systému, ale je jeho součástí a tím přetváří Linuxový systém na hypervizor typu 1. Původně byla vyvíjena společností Qumranet a později odkoupena firmou Red Hat, která tvoří komerční linuxové produkty. Ve svých distribucích Red Hat Enterprise Linux a Red Hat Virtualization rozšiřuje možnosti hypervizoru KVM a dodává

nadstandardní nástroje pro správu. Virtualizaci s pomocí KVM zvolil například také Google pro své řešení služeb Google Cloud. [28][29]

1.3.5 Oracle Virtualization

Společnost Oracle nabízí hned několik produktů pro virtualizaci. K dispozici je například vlastní řešení nástrojů pro správu KVM jako Oracle Linux Virtualizaton Manager nebo platforma založená na hypervizoru Xen Project pod názvem Oracle VM Server. Nejpopulárnějším nabízenou technologii je ale určitě Oracle VM VirtualBox. Je dostupný zdarma a lze jej provozovat na operačních systémech Windows, Linux i macOS. Slouží pro vytváření desktopové virtualizace a je založen na hypervizoru typu 2. Za pomoci přehledného grafického uživatelského prostředí umožňuje jednoduše spouštět, ale také vytvářet virtuální stroje a spravovat jejich běh. [30][31]

1.3.6 Qemu

Open source nástroj Qemu může pro virtualizaci využívat hypervizory jako KVM nebo Xen. Hlavní výhody ale nabízí v podobě plné emulace platformy, kdy dokáže vytvářet virtuální stroje s rozdílnou procesorovou architekturou, než existuje na fyzickém hardwaru. Dovoluje například na osobním počítači s procesorem na architektuře AMD64 provozovat systém vyžadující procesor s architekturou ARM. [32][33]

2 AUTOMATICKÁ INSTALACE OPERAČNÍHO SYSTÉMU LINUX

První uvedení do chodu, každého virtuálního či fyzického počítače vyžaduje instalaci operačního systému, což je poměrně zdlouhavý proces a běžně vyžaduje interakci uživatele. Při častém opakování, nebo mnohonásobném požadavku v jeden moment už se může jednat o vážnou komplikaci. Hlavní linuxové distribuce však nabízí poměrně jednoduché řešení v podobě automatizované instalace. Tato možnost je zpravidla postavená na principu kdy se instalátoru předá předpřipravený soubor s odpověďmi na otázky, které by jinak musel uživatel provádějící instalaci zodpovědět. Soubor lze předat přímo s instalačním mediem, to však vyžaduje zásah do instalačního obrazu. Jinou možností je stažení z webového serveru až při instalaci. [34][35]

Alternativou může být také síťové zavedení s využitím technologie Preboot Execution Environment (PXE). Jde o techniku, kdy je operační systém zaveden ze sítě bez použití instalačního média jako je CD nebo USB disk. Tento proces vyžaduje předchozí vytvoření zaváděcího serveru a celkové nabízí poměrně složitý způsob instalace operačního systému. Využití však může najít v prostředí kde není možné žádným způsobem provést instalaci z CD. [36]

2.1 Instalace operačního systému Debian

Debian a linuxové distribuce na něm založené využívají takzvaného přednastavení (preseeding). Odpovědi na otázky, které by musel uživatel vyplnit ručně jsou sepsány do jednoho souboru s přednastavením, který je při zahájení instalace předán instalátoru. Ke spuštění automatické instalace slouží příkaz:

auto url=“adresa souboru“

Instalační program následně stáhne soubor s přednastavením z uvedené URL a provede instalaci s využitím zapsaných předvoleb. Pro instalaci se souborem uloženým přímo na instalačním médiu může být využit příkaz pro automatickou instalaci ve formě:

auto file=“cesta k souboru“

Každý řádek souboru s přednastavením obsahuje jeden zápis odpovědi na jednu otázku. Případně dlouhý řádek může být rozdělen s využitím znaku „\“ na konci první části. Řádky, které začínají znakem „#“ jsou považovány za komentáře. Typicky je jeden řádek rozdělen do čtyř polí oddělených mezerami ve tvaru:

{vlastník} {identifikátor/typ otázky} {datový typ} {hodnota}

- *{vlastník}* představuje označení programu, jehož parametr je nastavován aktuálním řádkem. Pro většinu nastavení bude vlastníkem instalační program Debian-Installer zastoupený zkratkou „d-i“.
- *{identifikátor/typ otázky}* označuje otázku a její parametr který je nastavován například „passwd/username“ kde „passwd“ představuje sekci nastavení uživatele a „username“ že je nastavován parametr uživatelské jméno.
- *{datový typ}* označuje datový typ nastavované hodnoty, možnosti můžou být například string (textový řetězec), boolean (logická hodnota), select (výběr z hodnot) a další.
- *{hodnota}* představuje samotnou hodnotu nastavovaného parametru.

[34][35][37]

2.2 Instalace operačního systému Red Hat Enterprise Linux

Operační systém Red Hat Enterprise Linux (RHEL) a distribuce na něm založené jako například Fedora nebo CentOS využívají k automatické instalaci obdobný přístup jako distribuce založené na Debianu. Průběh instalace je řízen instalačním programem Anaconda. Příkazy nutné k vykonání instalace se zapisují do tzv. Kickstart souboru. Ten se předává při zavádění instalátoru v parametru:

inst.ks=“URL nebo cesta k souboru“

K vytváření komentářů se na začátku řádku uvádí znak „#“. Standardní příkaz se zapisuje ve tvaru:

{příkaz} --{parametr}={hodnota}

nebo pouze:

{příkaz} {hodnota}

Většina z možných příkazů jsou nepovinné, avšak několik je přímo vyžadovaných. Mezi tyto příkazy patří:

- *lang* – nastavení kódování jazyka
- *keyboard* – jazyk klávesnice
- *timezone* – volba časového pásma

- *auth* – způsob authentifikace
- *rootpw* – heslo pro uživatele root
- *bootloader* – nastavení zavaděče operačního systému

Kickstart soubor lze dělit do sekcí označených na začátku znakem „%“ a ukončených příkazem %end. Je možné vytvořit tři různé sekce s názvem packeges, pre nebo post. V sekci packeges jsou definovány balíčky, které se při instalaci operačního systému mají nainstalovat. Není třeba vypisovat všechny balíčky, některé jsou sloučeny do předdefinovaných skupin. Tyto skupiny jsou na začátku názvu označeny znakem „@“, například skupina s názvem @gnome-desktop sdružuje všechny balíčky, které jsou nutné pro provoz desktopového prostředí Gnome. Do sekce pre jsou zapisovány příkazy, které jsou vykonány ihned po spuštění instalace operačního systému. V sekci post jsou pak uvedeny příkazy, které budou vykonány po instalaci operačního systému. [34][35][38]

3 VITRUALIZACE NA PLATFORMĚ HYPER-V

Technologie Hyper-V, jak už bylo uvedeno v přehledu virtualizačního softwaru, je produktem společnosti Microsoft a slouží k provozu virtualizovaných počítačových systémů na fyzickém hostiteli. Tyto virtuální systémy je možné spravovat a používat stejně jako by šlo o fyzické systémy, avšak existují ve virtualizovaném izolovaném prostředí. Přístup mezi virtuálními systémy a fyzickými prostředky je na platformě Hyper-V a je spravován nativním hypervizorem (typ 1). Ten poskytuje virtualizované hardwarové prostředky s jejichž pomocí jsou tvořeny virtuální stroje. Tyto virtuální stroje umožňují provozovat nejen operační systém Windows ale také různé Linuxové distribuce nebo FreeBSD. [24][39]

3.1 Základní nástroje

Virtuální stroje na platformě Hyper-V je možné spravovat pomocí rutin (commandlet, cmdlet) vylepšeného prostředí příkazového řádku PowerShell který je standardně součástí systému Windows. PowerShell může být zároveň i skriptovacím jazykem. Může být tedy jednoduše využit i k automatizování úloh, vytvářením skriptů, vykonávajících více činností. Druhou možnou variantou správy Hyper-V virtuálních strojů je pak pomocí nástroje s grafickým uživatelským prostředím Hyper-V Manager. Pomocí obou nástrojů však lze vykonávat všechny důležité činnosti jako je nastavení funkcí Hyper-V, úkony spojené s řízením životního cyklu virtuálních strojů, správa virtuálních disků a další. [24][40]

3.2 Nástroje pro automatizaci

K efektivní práci s virtuální infrastrukturou je vhodné všechny procesy co nejvíce automatizovat. Hyper-V nabízí částečnou možnost automatizace správy virtuálních strojů v podobě PowerShell scriptů. Na platformě Hyper-V existují však další nástroje, které mohou pomoci automatizovat úkony spojené s vytvářením nových virtuálních strojů, správou jejích životního cyklu nebo například zjednodušit přenositelnost. Mezi tyto nástroje patří například spřízněné technologie Packer a Vagrant, za pomoci kterých je možné podstatně zkrátit čas nutný k nasazení nového virtuálního stroje a nabízí využití i na jiných platformách, než je Hyper-V.

3.2.1 Packer

Packer je nástroj s otevřeným zdrojovým kódem určený k vytváření obrazů virtuálních strojů pro více platforem při použití jednoho konfiguračního zdroje. Obrazy strojů dokáže vytvářet na všech hlavních operačních systémech, a to paralelně pro více platforem. K tomu využívá konfigurační šablony strukturované ve formátu JSON. [41][42]

Hlavními stavebními prvky Packer šablon jsou takzvaní:

- stavitelé (builders)
- poskytovatelé (provisioners)
- postprocesory (post-procesors)

Platforma, na které je sestavení obrazu spuštěno, je označována jako stavitel. Tím může být například Hyper-V, VirtualBox nebo VMware. K spuštění skriptů v prostředí sestavovaného obrazu virtuálního stroje jsou využíváni poskytovatelé, například shell. Po dokončení obrazu mohou být spuštěny postprocesory, které mohou výsledný obraz vyexportovat pro použití na jiných platformách. Například obraz vytvořený na platformě Hyper-V může být převeden na obraz pro VirtualBox nebo zabalen jako Vagrant box. [41][42]

3.2.2 Vagrant

Vagrant je nástroj pro práci s virtuálními prostředími. Tento nástroj poskytuje jednoduchého klienta v prostředí příkazového řádku, který spravuje tato prostředí. Stejně jako Packer má tento nástroj otevřený zdrojový kód, a proto jej může kdokoliv bezplatně používat. Jednou z výhod technologie Vagrant je, že při použití na různých platformách využívá stále stejný přístup, čímž tvoří jednotné rozhraní napříč různými virtualizačními platformami. Základním konfiguračním souborem je Vagrantfile, který využívá jazyk Ruby. Tento soubor obsahuje konfiguraci a zároveň udává, jaký obraz virtuálního stroje se má spustit. Vagrant využívá pro ukládání obrazů formát nazývaný Vagrant box. Tyto obrazy je možné získat z oficiálního uložiště této technologie Vagrant Cloud nebo si vyrobit vlastní s využitím nástroje Packer. [43][44]

4 NÁSTROJE PRO SPRÁVU SERVEROVÉ INFRASTRUKTURY

Na trhu existuje široká nabídka nástrojů, které usnadňují práci při správě serverové infrastruktury. Tato infrastruktura lze definovat jako soubor všech dostupných zařízení a prvků v rámci spravovaného prostředí jako je síť propojující virtuální nebo hardwarové prvky ať už servery či osobní počítače a další síťové prvky.

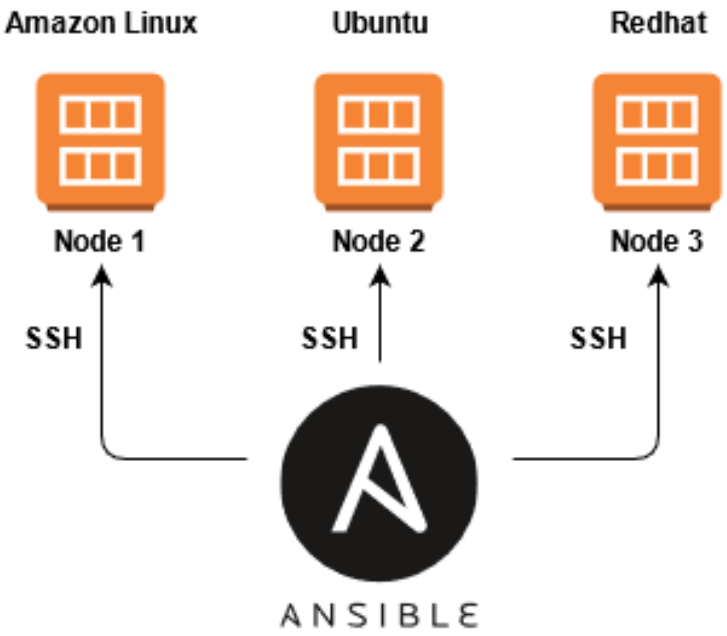
Technologie zajišťující správu takové infrastruktury si tak kladou za cíl automatizaci procesů spojených s instalací a konfigurací serverů a dalších síťových zařízení. Snaží se nabídnout efektivní řešení pro snížení manuálních a opakujících se úkolů. Umožňují souběžnou práci s velkým množstvím spravovaných zařízení. Často se snaží popisovat co největší část infrastruktury pomocí kódu, aby byla jednoduše udržitelná, editovatelná a bylo ji možné verzovat.

Jednotlivé nástroje se pak liší v jazycích využívaných k tvorbě konfiguračních souborů, v přístupu, jakým infrastrukturu řídí nebo v modelu architektury spravované infrastruktury. Některé nástroje je možné získat bezplatně, ale existují i profesionální komerční produkty, které se většinou zaměřují na nabídku přehledného uživatelského rozhraní s pokročilými funkcemi pro vizualizaci procesů a reportování událostí. [45][46]

4.1 Ansible

Ansile je pravděpodobně nejpopulárnější nástroj pro automatizaci správy serverů. Jedná se o open-source program, ale dostupné je i komerční řešení od společnosti Red Hat která nad ním postavila vlastní produkt Red Hat Ansible Platform, který nabízí oproti standardnímu přístupu přes příkazový řádek i grafické uživatelské prostředí s různými možnostmi vizualizace, plánování akcí, nebo zobrazení upozornění na události. K dispozici také dává vlastní kolekce modulů a analytické nástroje. [47][48][49]

Ve své podstatě je tvořen sadou skriptů a knihoven napsaných v jazyce Python. Za pomoci těchto scriptů tzv. modulů je umožněno provádět konfigurační úkony na spravovaných systémech tzv. uzlech (node). Moduly mají definovaný seznamu argumentů, na jehož základě provádí požadované operace. Pomocí argumentů je určen požadovaný stav nastavovaných hodnot, změny jsou provedeny pouze v situaci kdy se aktuální hodnota neshoduje s tou požadovanou. Takový přístup bývá označován jako desired-state approach. Následkem opakovaného spouštění jednoho příkazu tak není opětovně provedení změn na spravovaném uzlu. [50][51]



Obrázek 3 – Diagram správy pomocí nástroje Ansible [52]

Ansible v porovnaní s konkurenčními nástroji jako jsou např. Puppet nebo Chef, nevyžaduje přítomnost žádného klienta na koncových uzlech, ale většina modulů potřebuje pro svůj běh podporu jazyka Python. Proto také Ansible nabízí modul s názvem raw, který může spouštět operace na spravovaných uzlech i bez toho. Měl by však být využit pouze v nezbytných případech, a to primárně pro možnost instalace podpory jazyka Python v uzlech, kde není dostupná. Pro přístup k řízeným uzlům je výchozí a doporučenou volbou zabezpečený komunikační protokol Secure Shell (SSH), ale například pro komunikaci se systémem Windows lze využít i Windows Remote Management (WinRM) protokol. [34][53][54]

Pro Ansible je podstatným souborem tzv. inventář (inventory), obsahuje důležité informace o spravovaných uzlech, zápis je ve formátu INI, měly by v něm být specifikovány jména uzlů, jejich IP adresy, porty uživatelská jména, hesla a další. Spravované zařízení zde můžou být sdružovány do skupin. S využitím inventáře je možné spouštět moduly které se provádí činnost na uzlech zapsaných v inventáři. Je možné spouštět jednotlivé moduly pomocí příkazového řádku, ale především je důležitá možnost z modulů skládat takzvané scénáře (playbooks). Ty jsou zapsané pomocí přehledného formátu YAML Ain't Markup Language (YAML), jenž standardně slouží pro ukládání strukturovaných dat. [50]

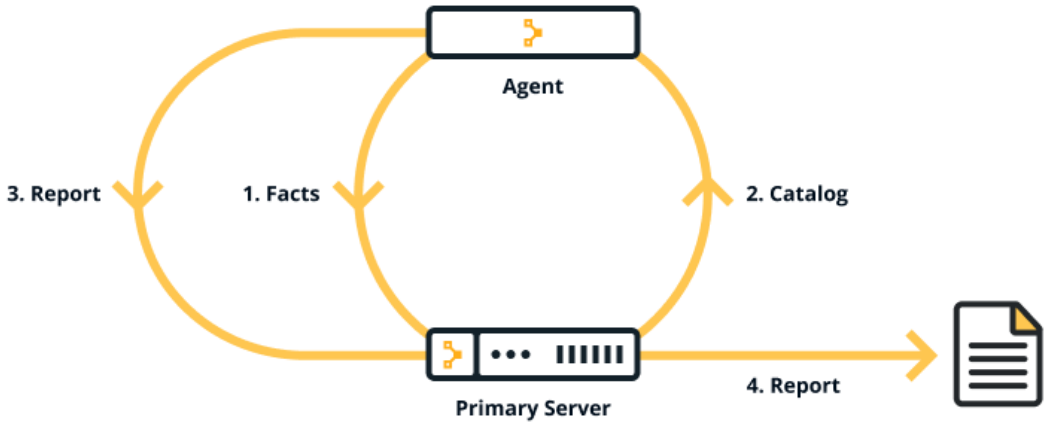
4.2 Puppet

Puppet byl v roce 2005 vytvořen firmou PuppetLabs. Je napsán v jazyce Ruby a pro psaní konfigurace používá vlastní jazyk založený právě na Ruby s názvem Puppet language. Tento jazyk je deklarativního typu, což znamená, že za pomocí tohoto jazyku se definuje, jak má stroj po aplikaci konfigurace vypadat, ale ne jakým způsobem se má požadované konfigurace dosáhnout.

Nástroj Puppet je dostupný ve dvou verzích, a to jako Open Source Puppet a Puppet Enterprise. Verze s otevřeným zdrojovým kódem obsahuje všechny důležité nástroje k provozování Puppet infrastruktury. Komerční verze Pupper Enterprise nabízí navíc užitečné nástroje pro efektivnější práci s touto technologií jako grafické uživatelské rozhraní a možnost vizualizace procesů.

Na spravovaných zařízeních vyžaduje Puppet instalaci klientské aplikace, takzvaného agenta. V rámci infrastruktury je nutná přítomnost také řídicího stroje, který je označován jako master nebo primary server. Puppet pracuje na principu kdy master poskytuje agentům na řízených strojích tzv. manifest, což je soubor s definicí konfigurace. Na jeho základě pak agenti realizují veškeré úkony nutné k tomu, aby systém dosáhl příslušného stavu. Agenti tak mohou například instalovat softwarové balíčky, případně je také odebírat nebo upravovat konfigurační soubory. Řídící stroj však nemusí spravovat jen vzdálené agenty, ale i sám sebe.

Z jednoduchých manifestů je možné skládat složitější kolekce takzvané moduly, tím lze dosáhnout zapouzdření celé konfigurace pro konkrétní použití například modul pro konfiguraci Apache serveru. Ucelená konfigurace tvořená rozsáhlou kombinací manifestů, která je aplikována na celý spravovaný stroj pak bývá označována jako katalog. Informace o stavu spravovaných uzlů je možné získat pomocí reportů. O vykonaných úkonech odesílá Agent svému nadřízenému serveru informace v podobě takzvaných faktů (Facts). Na obrázku (Obrázek 4) je vyobrazen zjednodušený model infrastruktury tvořené nástrojem Puppet. [45][55]



Obrázek 4 – Diagram infrastruktury řízené nástrojem Puppet [56]

4.3 Chef

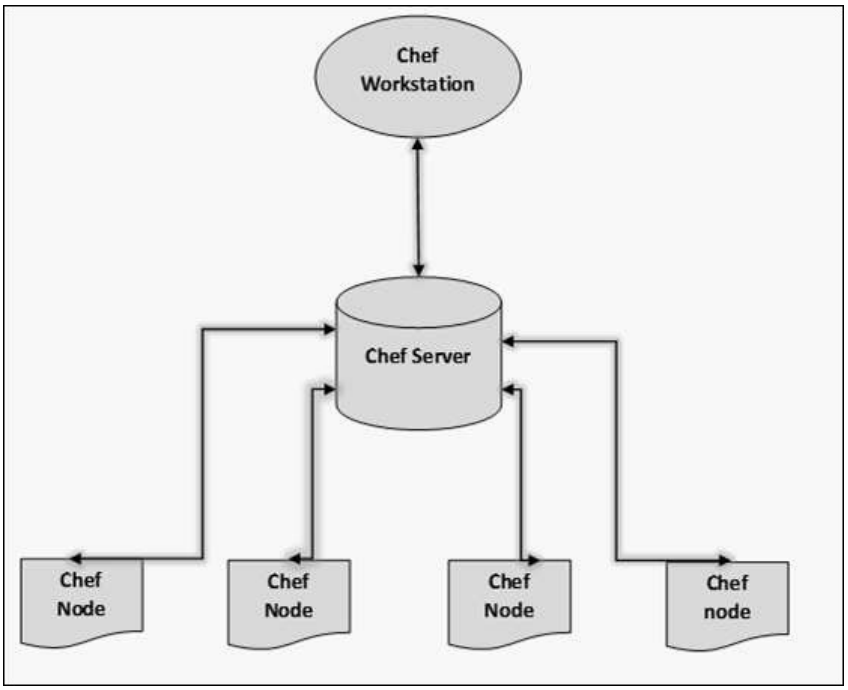
Chef je technologie s otevřeným zdrojovým kódem, která je vyvinutá společností Opsocode. Cílem této technologie je co nejpodrobněji popsat celou infrastrukturu v souborech pomocí kterých lze automatizovat nasazení nových strojů s požadovanou konfigurací a softwarovou výbavou.

Technologie Chef je napsaná v jazyce Ruby, který také využívá pro tvorbu kódu infrastruktury. Používá takzvaně imperativní přístup ke konfiguraci, kdy popisuje, jak přesně dosáhnout požadovaného stavu řízeného systému, na rozdíl od nástroje Puppet, který využívá deklarativní přístup.

Základním konfiguračním prvkem je zde takzvaný recept (receipe). Obsahuje seznam zdrojů nutných pro konfiguraci systému. Tyto recepty jsou ukládány do takzvaných kuchařek (cookbooks). Kuchařky jsou základní bloky, které se nahrávají na centrální řídicí server (Chef Server).

Samotná infrastruktura (Obrázek 5) se pak skládá ze tří vrstev, kde první vrstvu tvoří pracovní stanice (Chef Workstation), která je nainstalována na lokálním počítači, kde probíhá vývoj nových kuchařek. Pomocí nástroje Knife jsou nahrávány do druhé vrstvy, kterou tvoří řídící Chef Server. Ten slouží právě k distribuci kuchařek pro spravované uzly (Chef Nodes). Na těchto uzlech je nutná přítomnost klientské aplikace Chef Client. Jednotliví klienti po získání kuchařky z Chef Serveru aplikují přijaté nastavení v prostředí spravovaného uzlu. Dané uzly mohou mít různé druhy konfigurace v závislosti na přiřazených rolích. Chef

nabízí i možnost kdy není nutná přítomnost celé infrastruktury, ale uzel může být řízen napřímo s využitím nástroje Chef-Solo. [57][58]



Obrázek 5 – Diagram Chef infrastruktury [59]

4.4 SaltStack

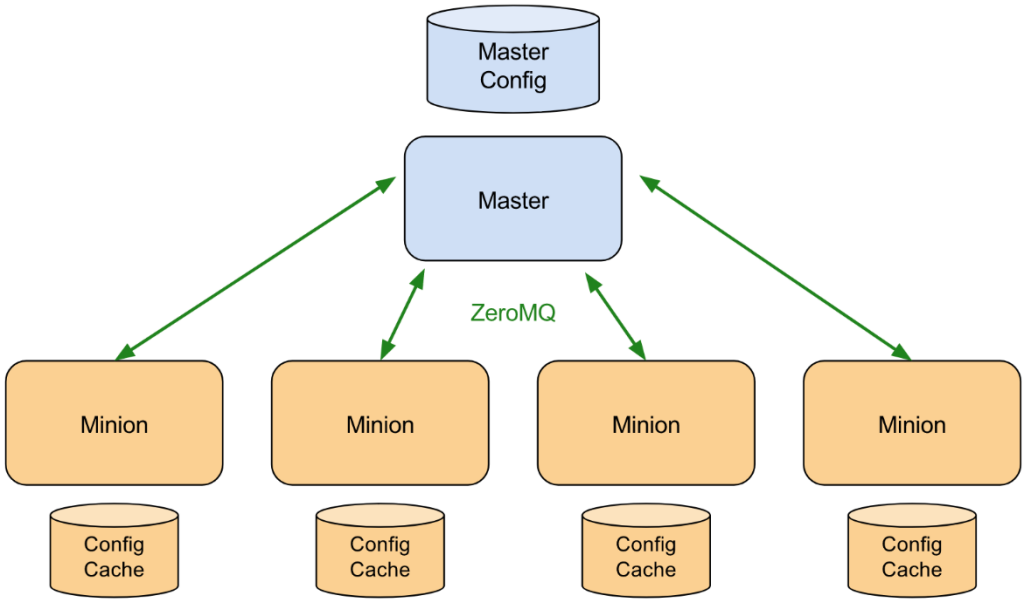
SaltStack je nástroj s otevřeným zdrojovým kódem vyvíjená společností SaltStack, kterou však na konci roku 2020 odkoupila společnost VMware. Tato technologie využívá deklarativního přístupu pro popis infrastruktury, podobně jako Puppet.

Základem správy konfigurace je model klient-server, kdy pro označení řídícího serveru slouží označení salt-master a jednotlivý klienti nainstalovaní na spravovaných serverech jsou nazýváni salt-minion. Existuje ale také varianta, kdy je aplikační server spravován sám sebou v takzvaném master-less módu.

Standardně se vyžívá infrastruktury, která je řízena centrálním serverem salt-master (Obrázek 6). Ten slouží jako uložiště konfiguračních dat a řídící centrum, které spouští vzdálené příkazy a zajišťuje stav řízených strojů. Salt-minion na spravovaných strojích je pak zodpovědný za provádění pokynů odeslaných řídícím serverem, podávání správ o výkonu úloh a poskytování údajů o hostiteli.

Komunikace mezi servery probíhá za použití knihovny pro asynchronní předávání zpráv ZeroMQ, která poskytuje velmi rychlou síťovou komunikaci.

SaltStack je navržen pro co možná největší modularitu a veškeré základní funkce jsou zapouzdřovány do stavových modulů, za pomoci kterých je vytvářena výsledná konfigurace. Konfigurace je zapisována do takzvaných vzorců ve formě strukturovaných dat formátu YAML. Salt-master směruje spravované servery na konkrétní sady vzorců, ale případně lze vzorce aplikovat i napřímo. Salt-minion následně vykoná činnosti spojené s dosažením stavu požadovaného moduly v poskytnutém vzorci. [60][61][62]



Obrázek 6 – Diagram infrastruktury řízené technologií SaltStack [63]

II. PRAKTICKÁ ČÁST

5 NÁVRH AUTOMATIZOVANÉHO ŘEŠENÍ

Návrh řešení byl vytvořen na základě požadavků a potřeb Knihovny Univerzity Tomáše Bati ve Zlíně zastoupené IT specialistou Ing. Ivanem Masárem, jakožto konzultantem této diplomové práce. Hlavním cílem bylo nalézt vhodné technologie pro automatizaci procesů spojených se správou stávající serverové infrastruktury. Při konzultaci s panem Ing. Masárem byly nalezeny konkrétní úkony, které je možné automatizovat a přispět tím k zefektivnění práce, zkrátit čas nutný k výkonu těchto akcí a také minimalizovat chyby způsobené lidským faktorem.

Přehled zkoumaných procesů k automatizaci:

- Vytvoření nového VM
- Instalace operačního systému VM
- Instalace dodatečného softwaru
- Nastavení a konfigurace
- Údržba systému

Formou rešerše byly prozkoumány dostupné nástroje pro správu serverů a v kombinaci s předpřipraveným obrazem VM nabízí řešení pro všechny dané požadavky. Také pro tvorbu samotného obrazu byla nalezena vhodná automatizovaná technologie. Při návrhu byl kladen důraz na možnost využití i mimo platformu Hyper-V. Návrh řešení se řídí také dalšími požadavky, jako přehlednost a srozumitelnost řešení, jednoduchost zavedení do stávající infrastruktury nebo například minimální nutnost provádět u administrátorů dodatečná školení.

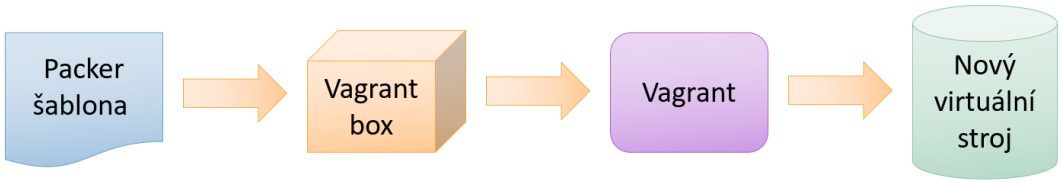
5.1 Obraz virtuálního stroje

Pro sestavení obrazu Hyper-V VM a automatizaci instalace operačního systému byla zvolena technologie Packer od společnosti HashiCorp. Především pro širokou podporu platforem, pro které lze tímto nástrojem vytvářet obrazy. Jedna připravená šablona tak jednoduše poslouží při tvorbě obrazu i jinde než na Hyper-V. A v budoucnu může výrazně usnadnit migraci serverové infrastruktury na jinou platformu.

S využitím technologie Packer byly navrženy dvě varianty řešení, kde první počítala s řízením životního cyklu pomocí technologie Vagrant a druhá by využila základní rutiny pro Hyper-V v prostředí Windows PowerShell.

5.1.1 Varianta č.1

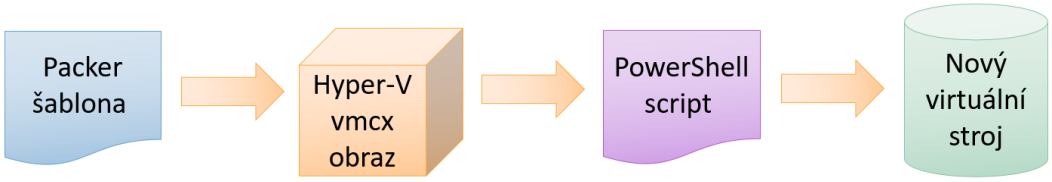
Jednou z možností technologie Packer je vytvářet obrazy tzv. boxy pro spřízněnou technologii Vagrant. V tomto boxu je v komprimované podobě uložen hotový Hyper-V VM. Ten pak lze Vagrant příkazem uvést do chodu, pomocí zápisu konfiguračních atributů do VagrantFile lze měnit konfiguraci virtuálních strojů. Vagrant přichází s velmi jednoduchým způsobem, jak řídit životní cyklus VM.



Obrázek 7 – Znázornění varianty č. 1 [vlastní]

5.1.2 Varianta č.2

Základním produktem sestavení obrazu pomocí technologie Packer je přímo nativní virtuální stroj Hyper-V, s tím lze pracovat pomocí grafického uživatelského prostředí Windows Hyper-V Manager nebo s využitím rutin v prostředí Windows PowerShell. Tyto rutiny jsou připraveny pro všechny důležité úkony, které jsou pro práci s Hyper-V VM důležité, umožňují stroj zkopírovat, modifikovat konfiguraci, spouštět a také zastavovat. Sdružením rutin do skriptů lze docílit automatizace všech kroků nutných k řízení životního cyklu virtuálního stroje.



Obrázek 8 – Znázornění varianty č. 2 [vlastní]

5.2 Správa serverů

Pro automatizaci při správě serverů byla zvolena technologie Ansible především pro jednoduchost použití kdy na spravovaných uzlech není třeba instalovat žádné klienty. Pro přístup k těmto uzlům využívá komunikaci pomocí zabezpečeného protokolu SSH.

Velkou výhodou je také že všechny scénáře, které bude administrátor využívat jsou zapsány ve velmi přehledné struktuře YAML a není tedy k vytváření těchto souborů nutné mít znalost konkrétního programovacího jazyka, tak jako tomu je u technologii Puppet nebo Chef. K výběru nástroje Ansible přispěla také jeho velká popularita a přítomnost široké komunity jeho uživatelů na internetu, což může přispět ke snadnějšímu řešení případných problémů. Výhodou je také to, že se jedná o program s otevřeným zdrojovým kódem a je tedy dostupný zdarma. Pokud však provozovatel bude chtít využívat tuto technologii v rozšířené verzi například o specializované nástroje pro centralizované ovládání infrastruktury pomocí grafického uživatelského prostředí, je možnost kdykoliv přejít na placenou komerční verzi Red Hat Ansible Automation Platform. Tato komerční verze rovněž obsahuje nástroje pro automatizační analýzu, vizualizaci nebo reportování běžících procesů ve spravované infrastruktuře.

6 REALIZACE

Návrh řešení byl realizován s využitím vzdáleného přístupu k testovacímu prostředí Windows Server 2019 Datacenter, které bylo poskytnuto Knihovnou Univerzity Tomáše Bati ve Zlíně. Jako první byla testována technologie Vagrant a její přínos pro infrastrukturu, aby bylo možné rozhodnout, která varianta z návrhu pro vytvoření obrazu virtuálního stroje bude aplikována v reálném provozu. Následně byl pomocí nástroje Packer vytvořen základní obraz virtuálního stroje. A na konec s využitím technologie Ansible byly vytvořeny obslužné scénáře důležité pro správu serverů.

6.1 Testování technologie Vagrant

Prvním krokem byla instalace nástroje Vagrant, instalační soubor je k dispozici z oficiálních stránek technologie ze sekce downloads dostupné z [64]. Po instalaci je připraven k použití jako nástroj v příkazovém řádku. Ověřit úspěšnost instalace si lze například příkazem:

vagrant info

který vrátí informace o aktuálně nainstalované verzi. Společnost Hashicorp také vlastní repositář hotových Vagrant boxů s názvem Vagrant Cloud, kde je k dispozici široká nabídka balíčků s různými systémy a připravených pro použití na různých platformách. Stažení a spuštění obrazu pomocí nástroje Vagrant lze provést velmi jednoduše ve dvou krocích. V prvním kroku provedením příkazu:

vagrant init

za který jako parametr stačí jen napsat název boxu v případě, že box pochází z repositáře Vagrant Cloud se vytvoří základní Vagranfile, jenž slouží pro následné spuštění, při němž je požadovaný box i stažen, a to pomocí příkazu:

vagrant up

Samotný box může být také umístěn i v jiném například soukromém uložišti ale v tom případě je nutné při provádění *vagrant init* uvést celou URL nebo lze také využít cestu na lokálním uložišti na které je požadovaný box dostupný.

Pro testování byl z repositáře Vagrant Cloud vybrán box s aktuálně nejnovější stabilní verzí operačního systému Debian (generic/debian10). Spuštěním příkazu *vagrant up* byl vytvořen nový virtuální stroj běžící v prostředí Hyper-V. Kontrola byla provedena pomocí nástroje Hyper-V Manager, po připojení k tomuto stroji bylo zjištěno, že v prostředí tohoto stroje

není dostupné připojení k internetu. Vyřešení téhle nepříjemné situace bylo nakonec poměrně dost komplikované.

Výsledkem samotného testování technologie Vagrant bylo zjištění, že varianta založená na této technologii není natolik zajímavá, aby byla použita. Konfiguraci a nasazení je možné podobně efektivně provést i za pomoci PowerShell skriptu, jenž nabízí druhá varianta. Nástroj Vagrant by však mohl přinést hodnotu v případě, že by byla prováděna virtualizace na více různých platformách, kdy zajišťuje jednotné rozhraní.

6.2 Odstranění komplikací se síťovým připojením

V prostředí Hyper-V byl administrátorem, který prostředí připravoval již před začátkem testování, spuštěn jeden virtuální stroj s operačním systémem Windows. Po připojení k němu bylo zjištěno, že také nemá dostupné připojení k internetu. Zkoumání tohoto problému vedlo k možnosti, že pravděpodobně je chyba ve virtuálním síťovém přepínači, který tyto stroje používají k přístupu do sítě. Dle návodu z Microsoft dokumentace [65] byl vytvořen nový, avšak problém vyřešen nebyl.

Jelikož samotné testovací prostředí bylo vytvořeno jako virtuální stroj na Hyper-V, další možné řešení bylo nalezeno v oficiálním návodu na spuštění vnořené virtualizace. [66] V něm jsou uvedeny dvě možnosti, jak zpřístupnit připojení k internetu:

- První uvedená možnost řešení vyžaduje na první úrovni povolit podvržení MAC adresy (MAC address spoofing), což nepřicházelo v úvahu, jelikož k této úrovni má přístup pouze hlavní administrátor.
- Proto byla otestována druhá možnost, ta navrhuje vytvořit nový virtuální switch, který umožňuje provozovat překlad síťových adres (Network Address Translations, NAT). Tato technika se standardně využívá na síťovém prvku, přes který přistupují zařízení z lokální sítě do internetu. Zajišťuje, že všechny zařízení v lokální síti budou přistupovat do internetu pod veřejnou IP adresou tohoto síťového prvku. [67] Výsledkem tohoto pokusu bylo pouze znemožnění vzdáleného přístupu k tomuto testovacímu prostředí.

Zprovoznění přístupu si vyžadovalo asistenci pana Ing. Ivana Masára, jakožto správce tohoto testovacího prostředí. Následně se však objevila nová komplikace. V prostředí nástroje Hyper-V Manager již nebylo možné spustit správce virtuálních síťových přepínačů Virtual Switch Manager, pokus o spuštění vracel pouze dialogové okno s informací o chybě popsané

zprávou „An error occurred while trying to retrive a list of virtual switches.“ v překladu „Při pokusu o načtení seznamu virtuálních přepínačů došlo k chybě.“. Řešením tohoto problému se zabývá článek dostupný z [68]. Nabízí tři možné řešení:

- první možnost doporučuje povolit protokol Hyper-V Extensible Virtual Switch v nastavení síťového adaptéru, jenž využívá hostitelský systém k připojení ke zbytku sítě, po prověření bylo zjištěno že tato možnost je již aktivní.
- Druhé řešení uvádí, že některým uživatelům pomohlo vytvořit nový virtuální síťový přepínač pomocí příkazového řádku PowerShell a při následném spuštění nástroje Virtual Switch Manager již byl problém odstraněn. Po provedení tohoto kroku bohužel k nápravě nedošlo.
- Třetí bod navrhuje řešení v podobě deaktivace a následné znovu aktivace celé služby Hyper-V v systému Windows, tato možnost ale také nebyla úspěšná.

Konečným řešením bylo kontaktování hlavního administrátora, který celé toto testovací prostředí odstranil a vytvořil nové, ve kterém již síťově připojení pracovalo správně.

6.3 Vytvoření základního obrazu

Úvodním krokem použití technologie Packer byla instalace. Tento nástroj byl pro snadné testování výsledků nainstalován bezprostředně v testovacím prostředí Windows Server. Obrazy vytvořené pomocí technologie Packer jsou však přenositelné a v podobě pro Hyper-V mohou být vytvořeny v jakémkoli systému, kde je tato možnost k dispozici. Instalaci Packer lze provést dle oficiálního návodu [69] pomocí správce softwarových balíčků Chocolatey. Jednoduše příkazem:

choco install packer

Samotnou instalací Chocolatey lze provédst také snadno dle návodu [70] z příkazového řádku PowerShell a to pomocí rutiny:

Set-ExecutionPolicy Bypass -Scope Process -Force; [System.Net.ServicePointManager]::SecurityProtocol = [System.Net.ServicePointManager]::SecurityProtocol -bor 3072; iex ((New-Object System.Net.WebClient).DownloadString('https://chocolatey.org/install.ps1'))

Automatická instalace operačního systému a vytvoření obrazu virtuálního stroje s použitím technologie Packer vychází z šablony v podobě JSON struktury, ve které jsou zapsány všechny důležité parametry.

S využitím Packer dokumentace [71] byla vytvořena šablona pro sestavení základního obrazu. Ukázka této šablony je dostupná v příloze P IV. Určitě nejdůležitější položkou každé takové šablony je instalační médium operačního systému, od kterého se odvíjí další parametry důležité pro instalaci samotného operačního systému. Vedle těchto informací šablona obsahuje i konfiguraci virtuálního hardware, který pro samotnou tvorbu obrazu není až tak důležitý, pouze v případě, že by byly nastaveny nízké hodnoty, mohlo by se sestavení zbytečně prodloužit, případně vůbec neprovést, pokud by konfigurace nesplňovala minimální požadavky instalovaného operačního systému.

U každého nově vytvářeného virtuálního stroje, na základě připraveného obrazu, bude hardwarová konfigurace pravděpodobně přenastavena dle potřeb konkrétního nasazení. Požadavkem bylo určeno vytvořit obraz s aktuální stabilní verzí operačního systému Debian, což aktuálně odpovídá vydání Debian 10 s kódovým označením Buster. [72] K instalaci tohoto systému byl využit minimální obraz instalačního CD pro síťovou instalaci, tento obraz obsahuje pouze minimální množství softwaru, které je důležité pro spuštění instalace. Zbývající programové balíčky jsou staženy pomocí internetu až při samotné instalaci. [73]

K automatizované instalaci operačního systému Debian se využívá soubor s přednastavením (preseed), tento soubor obsahuje především nastavení, které při běžné instalaci získává instalátor formou odpovědí na kladené otázky uživateli, který provádí instalaci. Soubor s přednastavením však nabízí i možnosti, které jsou u běžné instalace nedostupné. Například instalaci softwarových balíčků, které nejsou součástí základní nabídky, případně také umožňuje spouštět vlastní příkazy přímo v průběhu instalace. Všechny důležité informace k automatizované instalaci operačního systému Debian jsou přehledně s využitím ukázek uvedeny v dokumentaci dostupné z [37]. V níž je také dostupný ukázkový příklad s velkým množstvím komentářů popisujících dostupné možnosti nastavení. S využitím tohoto příkladu byl také sestaven soubor s přednastavením pro instalaci pomocí nástroje Packer.

Spuštění sestavení podle šablony, například s názvem *debian.json,* se provede příkazem:

packer build debian.json

Packer následně zaregistruje a spustí nový virtuální stroj v prostředí Hyper-V, provede instalaci operačního systému a virtuální stroj vypne. Na základě výsledného obrazu lze vytvářet nové virtuální stroje. Packer může výsledný obraz také zpracovat pomocí takzvaných postprocesorů a připravit tak obraz pro využití další technologií, například tímto způsobem lze obraz vyexportovat jako Vagrant box.

Pro aktuální řešení to sice nebylo nutné, ale při testování byla vytvořena i šablona, která této možnosti využívá a může sloužit jako pomůcka pro případné budoucí rozvíjení práce s touto technologií. Od šablony v Příloze P IV se liší pouze přidáním následujícího bloku:

```
"post-processors": [
  {
    "output": "{{ user `boxes_directory` }}/{{user `vm_name`}}.{{.Provider}}.box",
    "type": "vagrant"
  }
]
```

Blok post-procesors je zodpovědný za export obrazu do zvolené podoby v parametru type, v tomto případě je to Vagrant. V parametru output je nastavena cesta s názvem souboru kde se má výsledný Vagrant box uložit, zde je vytvořena za pomoci proměnný boxes_directory jenž představuje cílový adresář a název boxu je tvořen kombinací názvu virtuálního stroje a názvu platformy pro kterou je sestavován.

6.3.1 Možné komplikace realizace

Automatizované sestavení obrazu virtuálního stroje navrhnutým způsobem může také přinést různé komplikace, na které je dobré si dát pozor a nejlépe se jim vyhnout. Při této realizaci došlo k několika situacím, především z důvodu nepozorného čtení dokumentace, které prodloužily práci na vytvoření obrazu virtuálního stroje.

Výčet komplikací, které nastaly při realizaci:

- Nástroj Packer při spuštění sestavení vytvoří lokální webový server, pomocí kterého poskytuje soubory z nastaveného adresáře. Hlavní využití nachází pro předání souboru s přednastavením pro automatickou instalaci. Po zavedení instalace je spuštěn příkaz, který spustí automatickou instalaci ta jako parametr přebírá URL, z které má být stažen soubor s přednastavením. Při testování se však projevil problém, kdy instalátor nedokázal požadovaný soubor získat. V prostření hostitelského systému Windows bylo však v prostředí příkazového řádku PowerShell pomocí příkazu:
  wget http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_file.cfg,
  kde:
  - *{{ .HTTPIP }}* znázorňuje IP adresu hostitelského serveru
  - *{{ .HTTPPort }}* znázorňuje port, na kterém Packer vytváří webový server

Bylo ověřeno, že soubor dostupný je. Nakonec se ukázalo že aplikace Packer je blokována systémem Windows Defender Firewall a pro odstranění této komplikace je nutné přidat pravidlo prostřednictvím následující PowerShell rutiny:

New-NetFirewallRule -DisplayName "packer" -Direction Inbound -Profile Domain -Program "C:\Program Files\packer.exe" -Action Allow

- Packer po spuštění sestavení v prostředí Hyper-V vytvoří nový virtuální stroj a automaticky otevře okno s rozhraním obrazovky právě vytvořeného virtuálního stroje. Příkaz, který je spuštěn po zavedení instalátoru operačního systému je pomocí nástroje Packer zapsán prostřednictvím simulace vstupu z klávesnice, proto je nutné dát pozor, zda v prostředí hostitelského operačního systému není na klávesnici zapnutý Caps Lock. Operační systém Debian rozlišuje velká a malá písmena, tudíž by byl zadaný příkaz chybně zapsán velkými písmeny, a proto by tento proces skončil s chybovou hláškou.
- Pro sestavení nástrojem Packer je po instalaci samotného operačního systému důležitý přístup prostřednictvím protokolu SSH. Pomocí kterého následně může provádět další úpravy v systému. Pro zprovoznění této možnosti je nutné do přednastavení zahrnout instalaci balíčku openssh-server. Aby program Packer byl schopen zjistit IP adresu vytvořeného virtuálního stroje, je nutná v prostředí operačního systému Debian při provozu na platformě Hyper-V také instalace balíčku hyperv-daemons. Tato skutečnost může být v dokumentaci poměrně jednoduše přehlédnuta. [74]

6.4 Využití technologie Ansible

Řídicí nástroj Ansible nebylo nutné instalovat v prostředí testovacího serveru, jelikož pro svou funkci využívá vzdáleného přístupu prostřednictvím zabezpečeného protokolu SSH. S jeho pomocí lze spravovat jak zařízení s operačním systémem na bázi Linuxu, tak i s operačním systémem Windows. Vzhledem k tomu, že má Ansible tyto možnosti, byl k realizaci instalován na osobním počítači s operačním systémem založeném na Debianu, a to Ubuntu 20.04, nikoliv na vzdáleném serveru jako tomu bylo v případě testování předchozích technologií. Z důvodu, že pan Ing. Ivan Masár, který by toto řešení následně využíval, používá ke své práci operační systém Debian, byl tento způsob řešení ideální.

Nástroj Ansible je dostupný ze standardního repositáře softwarových balíčků pro linuxové distribuce založené na Debianu. Instalaci lze tedy provést spuštěním příkazu:

sudo apt-get install ansible

Základní konfiguraci lze nastavit v souborech ansible.cfg a hosts standardně vytvořených při instalaci v adresáři /etc/ansible. Kde ansible.cfg obsahuje konfigurační nastavení s komentáři k výchozímu nastavení, které by mělo být pro většinu potřeb dostačující. Ale při realizaci této práce se objevila nesrovnalost v cestě k Python interpretru, a proto v tomto souboru byla tato cesta přenastavena na správnou. Soubor hosts je možné využít jako základní inventář spravovaných uzlů. [75]

6.4.1 Práce s Windows uzly

Základní Ansible kolekce nejsou uzpůsobeny pro práci s jinými než linuxovými systémy. K dispozici jsou však rozšiřující kolekce, které je možné doinstalovat. Pro samotnou práci s uzly, které provozují operační systém Windows je nutné doinstalovat kolekci pomocí příkazu:

ansible-galaxy collection install ansible.windows

Tato kolekce obsahuje moduly umožňující interakci s prostředím spravovaného stroje. Jako například kopírování souborů z lokálního systému do vzdáleného uzlu, práci se vzdálenými soubory, spouštění skriptů, instalovat a odstraňovat software nebo spravovat nastavení Windows.

Před prací s těmito uzly je nutné nejprve v inventáři, případně přímo v playbooku pro dané spojení definovat proměnné ansible_conection a ansible_shell_type. Například pro využití komunikace pomocí SSH a práci s využitím konzole PowerShell je třeba zadat:

ansible_connection=ssh

ansible_shell_type=powershell

6.4.2 Přístup k spravovaným uzlům

Přístupové jméno a heslo lze ukládat přímo do inventářů s informacemi o spravovaných uzlech nebo využít možnosti protokolu SSH a k přihlášení na řízených uzlech využít SSH klíč. Tento klíč lze vygenerovat pomocí příkazu:

ssh-keygen

Uživatel následně bude vyzván k zadání cesty pro uložení vygenerovaného klíčového páru, dalším krokem je zadání hesla, které si zde uživatel může, ale nemusí zvolit. Po dokončení

se vytvoří soubory s privátním a veřejným klíčem. Jejich názvy mohou být zvoleny uživatelem nebo jsou ve výchozím nastavení pojmenovány id_rsa, kde veřejný klíč je navíc označen koncovkou .pub. Veřejný klíč je určen k uložení do systému, ke kterému chce uživatel přistupovat. [76]

Při realizaci řešení byl pro snadnější testování vytvořen klíčový pár bez volitelného hesla. Veřejný klíč byl nahrán na server Windows, do souboru authorized_keys nacházející se v domovském adresáři uživatele ve složce s názvem .ssh. Velmi důležitým krokem je nastavení práv k tomuto souboru. Je nutné odstranit přístup všem uživatelům kromě uživatele, v jehož adresáři se soubor nachází. [77][78]

6.4.3 Inventáře (inventory)

Pro spouštění obslužných scénářů je nutné si nejprve připravit takzvaný inventář, který obsahuje seznam spravovaných uzlů a proměnných k nim patřícím. Jednotlivé uzly lze zapisovat pomocí IP adresy, případně i doménového jména. Inventáře je možné zapisovat ve více formátech, kde nejběžnější jsou INI nebo YAML. Ve výchozím formátu INI pak může inventář vypadat například takhle:

adresa1.priklad.com
[webservers]
web1.priklad.com
web2.priklad.com
[dbservers]
db1.priklad.com
db2.priklad.com

Spravované uzly jsou sdružovány do skupin. Existují dvě výchozí skupiny s názvy all a ungruped. Skupina all sdružuje všechny hostitele zaznamenané v inventáři. Skupina unguped obsahuje všechny uzly, které nejsou přiřazeny žádné jiné skupině, kromě all. V příkladu uvedeném výše do této skupiny spadá adresa1.priklad.com. Vlastní skupiny lze vytvářet zápisem uzlů do sekcí označených názvem skupiny v hranatých závorkách, jako jsou v příkladu znázorněny skupiny webserves (pro webové servery) a dbservers (pro databázové servery) a tím je rozdělit. [79]

K jednotlivým skupinám je následně možné přiřazovat proměnné s údaji o uživateli, jako může být přihlašovací jméno a heslo a další údaje pro specifikaci připojení. Slouží k tomu sekce označená názvem skupiny s koncovkou „:vars“. Jako například:

[webservers:vars]

ansible_user=jmeno
ansible_passwoed=heslo

Pro použití při řízení Windows serveru, který vytváří virtualizační platformu byl vytvořen následující inventář:

[utb_win]
10.13.32.17
[utb_win:vars]
ansible_user=m_cibulka
ansible_connection=ssh
ansible_shell_type=powershell

Pro případ, že bude tento inventář rozšiřován, byl testovací server zapouzdřen do skupiny s názvem utb_win. K přístupu na testovací server není nutné díky vytvořenému připojení pomocí SSH klíče zadávat heslo, ale vzhledem k tomu, že uživatelská jména na osobním počítači, na kterém byl Ansible provozován, a na vzdáleném serveru se liší, bylo nutné uvést uživatelské jméno do proměnné ansible_user. Dále pro připojení k Windows serveru je nutné specifikovat typ připojení v proměnné ansible_connection, kde bylo zvoleno ssh. Poslední proměnná v tomto inventáři ansible_shell_type udává, jaký typ příkazového řádku je na spravovaném uzlu nastaven jako výchozí. [79]

Druhý použitý inventář s využitím u nově vytvořených linuxových virtuálních strojů je následující:

[debian_servers]
10.13.34.229
[debian_servers:vars]
ansible_user=utb
ansible_password=utbutb
ansible_become_pass=utbutb

Do obrazu virtuálního stroje nebyl z důvodu zjednodušení nahrán žádný SSH klíč, jelikož by jej bylo nutné po nasazení odstranit a nahradit novým. Z toho důvodu, aby neexistovalo jednotné přihlášením do všech virtuálních strojů založených na tomto obrazu. Proto pro nastavení nového virtuálního stroje je nutné uvést uživatelské jméno i heslo do proměnných ansible_user a ansible_password. K vykonání některých akcí spojených s konfigurací je potřeba administrátorský přístup, k tomu Ansible využívá metodu become, která však vyžaduje zadání hesla pro sudo, jenž je uvedeno v proměnné ansible_became_pass. [79]

Uvedené inventáře jsou také přiložené v elektronické podobě v příloze P I.

6.4.4 Obslužné scénáře (playbooks)

V rámci realizace byla vytvořena sada základních Ansible scénářů, jenž jsou také součástí přílohy P I. Tato sada obsahuje velmi jednoduchý scénář pro otestování základních funkcí ve verzi jak pro Linux, tak pro Windows s názvem hello_world.yml.

Verze pro Windows:

```
- name: Hello world from windows host
  hosts: all

  tasks:
  - name: Copy a single file
    ansible.windows.win_copy:
      src: ./scripts/hello_world.ps1
      dest: C:\hello_world.ps1
  - name: Execute a command in the remote shell
    win_shell: C:\hello_world.ps1
    register: output
  - name: Remove a file, if present
    ansible.windows.win_file:
      path: C:\hello_world.ps1
      state: absent
  - debug: msg="{{ output.stdout_lines }}"
```

Obecně každý Ansible scénář začíná parametrem name, jenž by měl jednoduše popsat, co daný scénář bude vykonávat. Následně parametr hosts uvádí skupinu spravovaných uzlů, pro které bude spuštěn. Je možné přidat i další parametry související s konfigurací připojení nebo jinak ovlivňující chování při spuštění scénáře, stejně tak mohou být uvedeny jako proměnné v inventáři. Dále obsahuje seznam úkolů k provedení pod klíčovým slovem tasks. Jednotlivé úkoly jsou voláním konkrétního Ansible modulu. Zápis úkolů zpravidla začíná názvem prováděné akce v parametru name a pokračuje voláním modulu s nastavením konkrétních hodnot požadovaných daným modulem.

Testovací scénář hello_world.yml pro operační systém Windows se skládá ze čtyř úkolů:

- Kopírování přiloženého PowerShell skriptu hello_world.ps1 za pomoci modulu ansible.windows.win_copy na disk C:\. Přiložený skript obsahuje pouze příkaz: *echo "Hello, World!"*
- Spuštění skriptu v prostředí PowerShell na spravovaném uzlu. K tomu je využit modul ansible.windows.win_shell. Přesto, že by moduly měly být vždy uváděny s plným názvem obsahující i název kolekce, ve které je daný modul obsažen, aby nedošlo k záměně s modulem z jiné kolekce. Při testování se nepodařilo s tímto zápisem daný

modul spustit. Proto je ve scénáři uveden pouze jako win_shell. Výsledek spuštění skriptu je zapsán do proměnné output zadané do parametru register.

- Smazání skriptu hello_world.ps1 za pomoci modulu ansible.windows.win_file, který slouží k práci se soubory. S nastaveným parametrem state na hodnotu absent, pak slouží k mazání.
- Pomocí modulu debug je vytvořena zpráva, jenž je vypsána do terminálu. V tomto případě je do zprávy umístěn výpis, který byl zapsán do proměnné output při spouštění skriptu. Pokud vše proběhlo správně, vypsaným textem bude řetězec „Hello, World!“.

Pro Linux pak existuje obdobný testovací scénář:

```
- name: Hello world from linux host
  hosts: all

  tasks:
  - name: Copy a file
    ansible.builtin.copy:
      src: ./scripts/hello_world.sh
      dest: /home/utb/hello_world.sh
      mode: 0711
  - name: Execute a command in the remote shell
    shell: /home/utb/hello_world.sh
    register: output
  - name: Remove a file, if present
    ansible.builtin.file:
      path: /home/utb/hello_world.sh
      state: absent
  - debug: msg="{{ output.stdout_lines }}"
```

- Kopírování bash skriptu probíhá pomocí modulu ansible.builtin.copy, přebírá parametry s cestou k souboru, cílovou cestou a nastavením práv k souboru. Použitý bash script obsahuje:

  ```
  #!/bin/bash
  echo "Hello, World!"
  ```

- Ke spuštění skriptu je využit modul ansible.builtin.shell, u nějž se objevil stejný problém, jako u ansible.windows.win_shell, proto je uveden jen jako shell. Výstup je také zapsán do proměnné output.

- Pro práci se soubory slouží modul ansible.builtin.file, přebírá parametr path s cestou k souboru a parametr state určující akci. V tomto případě je uvedena hodnota absent pro smazání.
- Na konec je pomocí modulu debug vypsán text z proměnné output. Pokud vše proběhlo správně, objeví se textový řetězec „Hello, World!“.

Dále sada obsahuje užitečné scénáře pro Windows, pomocí kterých lze měnit výchozí prostředí příkazového řádku.

Scénář default_shell_cmd.yml nastavuje jako výchozí příkazový řádek cmd.exe:

```
- name: set the default shell to cmd
  hosts: all

  tasks:
  - name: set the default shell to cmd
    ansible.windows.win_regedit:
      path: HKLM:\SOFTWARE\OpenSSH
      name: DefaultShell
      state: absent
```

Naopak scénář default_shell_ps.yml nastavuje jako výchozí PowerShell:

```
- name: set the default shell to PowerShell
  hosts: all

  tasks:
  - name: set the default shell to PowerShell
    ansible.windows.win_regedit:
      path: HKLM:\SOFTWARE\OpenSSH
      name: DefaultShell
      data: C:\Windows\System32\WindowsPowerShell\v1.0\powershell.exe
      type: string
      state: present
```

Oba scénáře využívají modul ansible.windows.win_regedit, který pracuje s Windows registry a je u něj možné nastavit tyto parametry:

- *path* – cesta k požadovanému registru
- *name* – název položky daného registru
- *data* – hodnota položky registru
- *type* – datový typ hodnoty registru
- *state* – označení pro vložení nebo smazání

Nejdůležitějším scénářem je start_new_vm.yml, jenž slouží k nasazení nového virtuálního stroje na základě připraveného obrazu:

```
- name: Create and start new VM from image
  hosts: all

  tasks:
  - name: Copy script to Hyper-V Server
    ansible.windows.win_copy:
      src: ./scripts/start_new_vm.ps1
      dest: C:\Local-Hyper-V\start_new_vm.ps1
  - name: Execute a command in the remote shell
    win_shell: C:\Local-Hyper-V\start_new_vm.ps1 -sourceVmPath "C:\Local-Hyper-
        V\images\debian_10\Virtual Machines\EEA589F5-4A9C-419C-A2B8-
        B092774D51FF.vmcx" -vmsFolder "C:\Local-Hyper-V" -newVmName debian_new -
        procesorsCount 2 -memorySize 512MB -diskSize 30GB
    register: output
  - debug: msg="{{ output.stdout_lines }}"
```

Samotný scénář je postaven na stejném principu jako výše uvedený testovací scénář hello_world.yml.

- PowerShell skript řídící nasazení je zkopírován na virtualizační server.
- Skript je spuštěn s řadou parametrů specifikující konfiguraci virtuálního stroje.
    - sourceVmPath představuje cestu k připravenému obrazu uloženému v prostředí hostitelského serveru zajišťujícího virtualizaci.
    - vmsFolder označuje adresář, ve kterém bude nový virtuální stroj uložen.
    - newVmName přebírá název nového virtuálního stroje.
    - procesorsCount udává počet procesorů.
    - memorySize představuje velikost operační paměti
    - diskSize udává velikost pevného disku
- Výstup provedeného skriptu je zapsán do proměnné a vypsán do terminálu, kde byl scénář spuštěn. Tímto výstupem je podrobný popis konfigurace, ze které je nejdůležitějším parametrem dynamicky přiřazená IP adresa, kterou je nutné si zaznamenat pro následnou práci s tímto strojem.

Pro následné nastavení síťové konfigurace je připraven scénář network_interfaces.yml:

```
- name: Set eth0 ip address
  hosts: all
  become: yes

  tasks:
```

```
- name: Set eth0 ip address
  community.general.interfaces_file:
    dest: /etc/network/interfaces
    iface: eth0
    address_family: inet
    option: "{{ item.option }}"
    value: "{{ item.value }}"
    state: present
  register: output
  loop:
    - { option: method, value: static }
    - { option: address, value: 10.13.34.229 }
    - { option: netmask, value: 255.255.252.0 }
    - { option: gateway, value: 10.13.32.1 }
    - { option: dns-server, value: 10.13.34.230 }
- debug: msg="{{ output }}"
```

Tento scénář využívá modul community.general.interfaces_file jenž není dostupný v základní instalaci, proto je nutné jej doinstalovat pomocí příkazu:

ansible-galaxy collection install community.general

Pro spuštění je vyžadována řada parametrů, které je nutno dosadit:

- *dest* – cesta k souboru s nastavením sítě, standardně je v prostředí debian umístěn v /etc/network/interfaces
- *iface* – název rozhraní
- *address_family* – typ IP adresy, standardně inet označuje typ IPv4
- *option* – název nastavovaného parametru
- *value* – hodnota nastavovaného parametru
- *state* – volba zápisu nebo mazání

K nastavení všech potřebných parametrů bylo využito možnosti vytvoření smyčky, pomocí které je nastavovaný parametr a jeho hodnota cyklicky vyplněna připravenými daty a samotný modul je v rámci jednoho spuštění scénáře vykonán několikrát, až do vyplnění všech požadovaných hodnot.

K zprovoznění základního Apache2 serveru slouží scénář apache2_install.yml:

```
- name: Apache2 server install
  hosts: all
  become: yes

  tasks:
    - name: install apache2
```

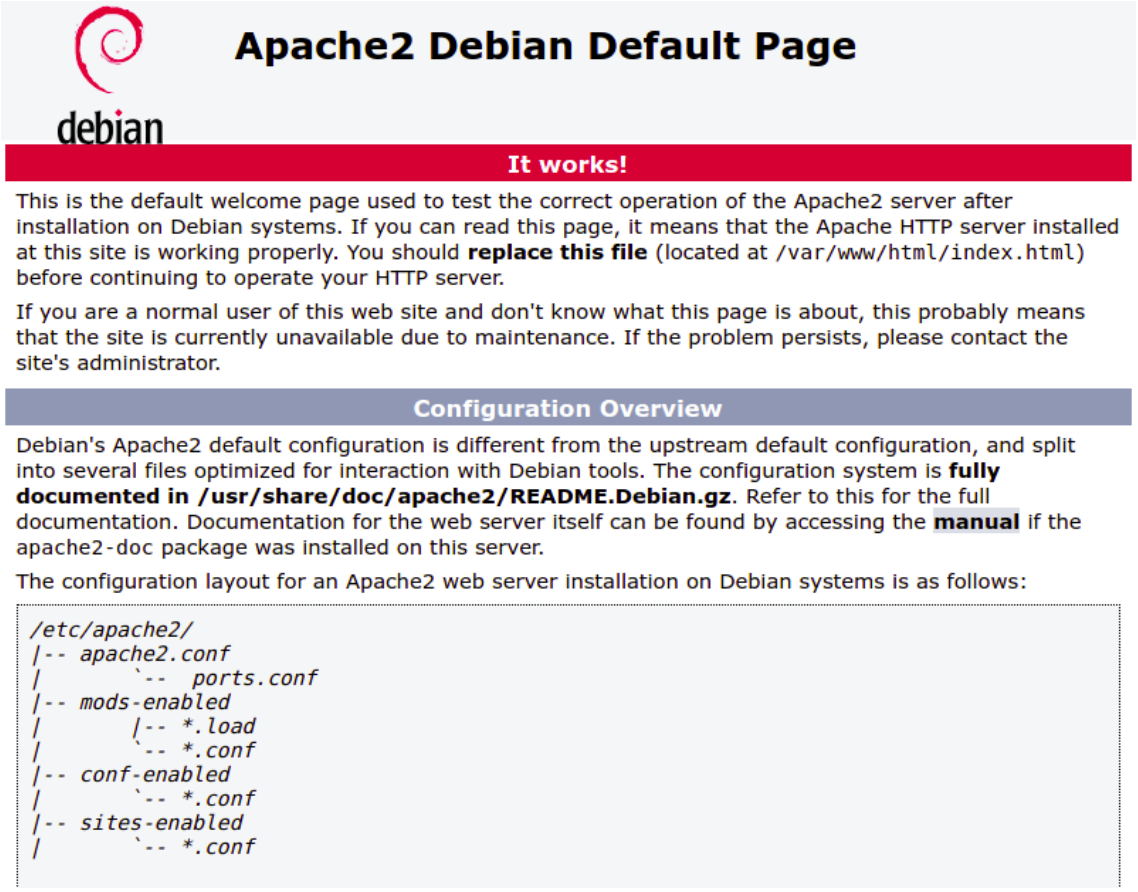
```
    ansible.builtin.apt: name=apache2 update_cache=yes state=latest

  - name: enabled mod_rewrite
    apache2_module: name=rewrite state=present
    notify:
      - restart apache2

handlers:
  - name: restart apache2
    service: name=apache2 state=restarted
```

- V prvním kroku je za pomoci modulu ansible.builtin.apt, který standardně slouží pro instalaci softwarových balíčků, nainstalován webový server Apache2.
- Dalším krokem je povolení Apache modulu mod_rewrite.
- Nakonec proběhne restart Apache serveru.

Po vykonání tohoto scénáře si lze funkčnost Apache serveru ověřit zadáním IP adresy spravovaného uzlu do prohlížeče a na standardním portu 80 bude dostupná výchozí uvítací stránka Apache serveru.



Obrázek 9 – Ukázka uvítací stánky Apache serveru [vlastní]

7 BEZPEČNOST

Při tvorbě této práce byla provedena pouze ukázková realizace a při nasazení do provozu je nutné myslet na bezpečnost a dořešit všechny možné bezpečnostní nedostatky, jejichž řešení je nad rámec této diplomové práce. Obraz virtuálního stroje je založen s velmi jednoduchým heslem pro uživatele root i administrátorského uživatele, který může využívat sudo, tyto hesla by měly být minimálně nahrazena silnějšími. Uživateli root je zakázán přístup přes SSH již v základní konfiguraci, a však druhý účet by mohl být narušen útokem hrubou silou. Lepším řešením by mohlo být nepovolit přístup přes SSH s ověřením hesla žádnému uživateli a využívat pouze přístup ověřený pomocí SSH klíče zabezpečeného heslem. [80]

Heslo uživatele na vytvořených strojích se také objevuje přímo v inventáři pro správu těchto serverů. Zajímavým řešením by mohlo být udržování hesel za pomoci nějakého správce citlivých údajů jako například Vault od společnosti HashiCorp. Tato technologie nabízí možnost získávat hesla, certifikáty, klíče a další citlivé údaje až na základě autentizace uživatele, takže neoprávněný uživatel nebude mít možnost se k citlivým údajům dostat. Vault dokáže také monitorovat kdy, kdo a k jakým údajům přistupoval. Také může být nápomocný při pravidelné obměně hesel. [81]

Vhodné je zabývat se i zabezpečením celé platformy už na úrovni hypervizoru. Hyper-V pracuje s nativním hypervizorem (typ 1), tudíž jednotlivé provozované virtuální stroje nemají žádnou informaci o tom, že běží ve virtuálním prostředí. Možnost nějakého napadení hypervizoru nebo ostatních strojů na něm běžících z prostředí některého z nich je tedy minimální. Určitou cestou by mohlo být využívání sdílených datových uložišť nebo síťová komunikace mezi nimi. S šířením škodlivého kódu si pak mohou poradit antivirové softwarové prostředky. Nejnebezpečnějším prvkem všech řešení, bývá často člověk a poškození na základě jeho nedbalosti, případně i úmyslu. Samotný přístup ke stroji, na kterém je virtualizace provozována by měl být výhradně z lokání sítě. Pro bezpečnější přístup pak může být vhodné zavést více faktorové ověřování. Důležité je také správné přidělení oprávnění, logování přístupů, sledování činnosti uživatelů a monitorování stavu celé virtualizační vrstvy a jednotlivých virtuálních strojů. [82]

ZÁVĚR

V rámci této diplomové práce byl na základě požadavků Knihovny Univerzity Tomáše Bati ve Zlíně vytvořen návrh řešení automatizovaného nasazení virtuálních serverů na platformě Microsoft Hyper-V s následnou ukázkovou realizací. Veškeré aspekty navrhovaného řešení byly konzultovány se zástupcem z knihovny panem Ing. Ivanem Masárem, aby výsledné řešení co nejvíce vyhovovalo potřebám zadavatele. Hlavním cílem řešení této práce je úspora času a lidské energie, která je jinak nutná k zavádění nových virtuálních serverů. Automatizované nasazovaní serverů snižuje také riziko chyb, které by mohly vznikat při manuálním zavádění.

Při ukázkové realizaci byla vytvořena šablona pro automatické vytváření obrazu virtuálního stroje s využitím nástroje Packer. Tato technologie přináší výhodu v možnosti použití této šablony i na jiných platformách, než je Microsoft Hyper-V, pro kterou je tento projekt určen. K automatizaci nasazování a správy virtuálních serverů byl zvolen nástroj Ansible a pro práci s ním byla vytvořena sada scénářů.

Součástí práce je také přehledný manuál popisující využití připravených šablon, scénářů a skriptů, díky kterému uživatel dokáže snadno zavést nový virtuální server bez hlubšího nastudování použitých technologií nebo předchozích znalostí. Tento manuál byl vytvořen s použitím nástroje Microsoft Publisher.

Samotou realizaci provázelo spoustu menších či větších komplikací, které se následně podařilo zdárně vyřešit. Zásadním problémem, jenž značně zpomalil postup ve vývoji práce, byla nemožnost připojení se k internetové síti v testovacím prostředí vnořeného Microsoft Hyper-V poskytnutého Knihovnou UTB ve Zlíně. Tento problém se nedařilo vyřešit žádnou z nabízených možností. Řešení přineslo až smazání nefunkčního prostředí administrátorem z knihovny a následné vytvoření nového prostředí, kdy síť začala fungovat bez problémů.

Aktuální připravené praktické řešení by bylo vhodné v budoucnu ještě rozvinout o scénáře, které by pokryly všechny potřeby práce s virtuální infrastrukturou, jako je například specifická konfigurace, instalace dalších softwarových nástrojů na virtuální stroje nebo možnost automatizovaného nahrazení již existujících strojů za nové verze. Pro aplikaci v produkčním prostředí bude nutné věnovat pozornost zabezpečení, jako je uvedeno v poslední kapitole práce. Zejména si dát pozor na práci s hesly.

Toto řešení může být zajímavé usnadnění práce nejen pro zadavatele, ale i pro jiné firmy a instituce, které pracují s virtuální infrastrukturou a zároveň může být inspirací pro subjekty, které prostředí virtuálních serverů ještě nevyužívají.

SEZNAM POUŽITÉ LITERATURY

[1] Zpravodaj ÚVT MU: Techniky virtualizace počítačů (2) [online]. Brno, 2007 [cit. 2021-5-12]. Dostupné z: http://webserver.ics.muni.cz/bulletin/articles/545.html

[2] Design and use of virtualization technology in cloud computing [online], 2017. Hershey PA, USA: IGI Global [cit. 2021-5-12]. ISBN 9781522527862. Dostupné z: doi:10.4018/978-1-5225-2785-5

[3] KRÁL, Jan Virtualizace operačních systémů: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2016. 97 s. Vedoucí práce byl doc. Ing. Dan Komosný, Ph.D.

[4] Cloudové výpočetní služby | Microsoft Azure: Co je cloud? [online]. Seattle, ©2021 [cit. 2021-5-12]. Dostupné z: https://azure.microsoft.com/cs-cz/overview/what-is-the-cloud/

[5] Správa sítí a outsourcing IT | Quadro Net: Pochopte co je cloud, cloud computing a další pojmy [online]. Praha: Quadro Net, ©2021 [cit. 2021-5-12]. Dostupné z: http://www.quadronet.cz/pochopte-co-je-cloud-cloud-computing-a-dalsi-pojmy/

[6] SystemOnLine.cz - ekonomické a informační systémy v praxi: Je cloud pro firmy vždy ideální řešení? [online]. Brno: CCB, 2020 [cit. 2021-5-12]. Dostupné z: https://www.systemonline.cz/virtualizace/je-cloud-pro-firmy-vzdy-idealni-reseni.htm

[7] SMEJKAL, T. Návrh na optimalizaci serverů využitím virtuálního prostředí. Brno: Vysoké učení technické v Brně. Fakulta podnikatelská. 2009. 50s. Vedoucí bakalářské práce Ing. Viktor Ondrák, Ph.D.

[8] STANČÍK, M. Optimalizace IT infrastruktury za pomocí virtualizace serverů. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2014. 93 s. Vedoucí diplomové práce Ing. Viktor Ondrák, Ph.D.

[9] Medium – Where good ideas find you.: Type 1 and Type 2 Hypervisors: What Makes Them Different [online]. Bombaj, 2019 [cit. 2021-5-12]. Dostupné z: https://medium.com/teamresellerclub/type-1-and-type-2-hypervisors-what-makes-them-different-6a1755d6ae2c

[10] Red Hat - We make open source technologies for the enterprise: VIRTUALIZATION What is a hypervisor? [online]. Raleigh, ©2021 [cit. 2021-5-

12]. Dostupné z: https://www.redhat.com/en/topics/virtualization/what-is-a-hypervisor

[11] KAFKA, Jan. VIRTUALIZACE APLIKACE PRO MONITOROVÁNÍ SÍTĚ ROK. Praha, 2009. Bakalářská práce. Vysoká škola ekonomická v Praze, Fakulta informatiky a statistiky, Katedra informačních technologií. Vedoucí práce Ing. Luboš Pavlíček.

[12] SystemOnLine.cz - ekonomické a informační systémy v praxi: Virtualizace v kostce K čemu, jak a proč? [online]. Brno: CCB, 2010 [cit. 2021-5-12]. Dostupné z: https://www.systemonline.cz/clanky/virtualizace-v-kostce.htm

[13] KOMÍNEK, Jiří. Porovnání virtualizačních technologií. Jihlava, 2013. Bakalářská práce. VYSOKÁ ŠKOLA POLYTEC HNICKÁ JIHLAVA, Obor Počítačové systémy, Katedra elektrotechniky a informatiky. Vedoucí práce Ing. Luboš Pavlíček.

[14] Virtualizace operačních systémů [online]. Radek Beran, 2006 [cit. 2021-5-12]. Dostupné z: http://beranr.webzdarma.cz/virtualizace.html

[15] Root.cz - informace nejen ze světa Linuxu: Úvod do virtualizace pomocí XENu [online]. Miroslav Suchý, 2007 [cit. 2021-5-12]. Dostupné z: https://www.root.cz/clanky/uvod-do-virtualizace-pomoci-xenu/

[16] AbcLinuxu.cz - Linux na stříbrném podnose: Virtualizace na úrovni jádra operačního systému [online]. Jaroslav Tomeček, 2007 [cit. 2021-5-12]. Dostupné z: https://www.abclinuxu.cz/clanky/system/virtualizace-na-urovni-jadra-operacniho-systemu

[17] Diit.cz - Vybráno z IT: Není virtuál jako virtuál [online]. Filip Marvan, 2012 [cit. 2021-5-12]. Dostupné z: https://diit.cz/clanek/virtualizace-na-urovni-operacniho-systemu

[18] STREIT, J. Virtualizace a konsolidace serverů. Brno: Vysoké učení technické v Brně, Fakulta podnikatelská, 2011. NEXT s.r.o., 100 s. Vedoucí bakalářské práce Ing. Viktor Ondrák, Ph.D.

[19] Virtualizační technologie a jejich využití ve vzdělávání [online]. Olomouc, © 2020 [cit. 2021-5-17]. Dostupné z: https://www.pdf.upol.cz/fileadmin/userdata/PdF/katedry/ktiv/Studijni_materialy/Klement/2020/VTV_2020_cviceni.pdf

[20] Virtualizace, konzultace a správa pro řešení na VMware vSphere, Veeam, Linux a Windows | OldanyGroup: VMWARE [online]. Praha: OldanyGroup, ©2021 [cit. 2021-5-12]. Dostupné z: http://www.oldanygroup.cz/vmware-110/

[21] MÜLLER, Marek. Virtuální desktopy na platformě VMware, Hyper-V a Citrix. Hradec Králové, 2016. Diplomová práce. Univerzita Hradec Králové, Fakulta informatiky a managementu, Katedra informačních technologií. Vedoucí práce Ing. Jan Budina.

[22] VMware - Delivering a Digital Foundation For Businesses: VMware Products [online]. © 2021 [cit. 2021-5-12]. Dostupné z: https://www.vmware.com/products.html

[23] Oficiální domovská stránka Microsoft: Does enabling Hyper-V on Windows 10 Home breaks EULA [online]. Xiaowei He, 2021 [cit. 2021-5-16]. Dostupné z: https://docs.microsoft.com/en-us/answers/questions/254205/does-enabling-hyper-v-on-windows-10-home-breaks-eu.html

[24] Oficiální domovská stránka Microsoft: Introduction to Hyper-V on Windows 10 [online]. 2018 [cit. 2021-5-12]. Dostupné z: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/

[25] Quora - A place to share knowledge and better understand the world: What technology is Azure running on? Is it on Hyper-V or some other kind of virtualization technology? [online]. Abhishek Pradhan, 2018 [cit. 2021-5-12]. Dostupné z: https://www.quora.com/What-technology-is-Azure-running-on-Is-it-on-Hyper-V-or-some-other-kind-of-virtualization-technology

[26] Home - Xen Project: WHY XEN PROJECT? [online]. [cit. 2021-5-12]. Dostupné z: https://xenproject.org/users/why-xen/#

[27] Home - Xen Project: GETTING STARTED [online]. [cit. 2021-5-12]. Dostupné z: https://xenproject.org/developers/getting-started-devs/

[28] Red Hat - We make open source technologies for the enterprise: VIRTUALIZATION What is KVM? [online]. Raleigh, © 2021 [cit. 2021-5-12]. Dostupné z: https://www.redhat.com/en/topics/virtualization/what-is-KVM

[29] MasterDC – Specialisté na firemní IT infrastrukturu: Typy virtualizace serverů – je lepší KVM nebo LXC? [online]. Jan Můčka, 2020 [cit. 2021-5-12]. Dostupné z: https://www.master.cz/blog/typy-virtualizace-serveru-kvm-nebo-lxc/

[30] ORACLE: Oracle VM VirtualBox [online]. © 2020 [cit. 2021-5-12]. Dostupné z: https://www.oracle.com/a/ocom/docs/oracle-vm-virtualbox-ds-1655169.pdf

[31] Oracle Česká republika | Integrated Cloud Applications and Platform Services: Virtualizace Oracle [online]. Praha: Oracle Czech, © 2021 [cit. 2021-5-12]. Dostupné z: https://www.oracle.com/cz/virtualization/

[32] Root.cz - informace nejen ze světa Linuxu: QEMU - qemulátor [online]. Petr Krčmář, 2005 [cit. 2021-5-12]. Dostupné z: https://www.root.cz/clanky/qemu-qemulator/

[33] Arch Linux: QEMU [online]. 2021 [cit. 2021-5-12]. Dostupné z: https://wiki.archlinux.org/title/QEMU?fbclid=IwAR131FuJd8PQTjT80nOoA_6sEYxPu3eGq6tnekQXMPqDeF7AG4zLrTECS2U

[34] Sokol, Jan. Infrastruktura pro centralizovanou automatickou instalaci serverů. Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2017.

[35] CHURÝ, Jan. Automatizovaná instalace virtuálních stroj ˚u propojených IPv6 sítí. Brno, 2013. Bakalářská práce. MASARYKOVA UNIVERZITA, Fakulta informatiky. Vedoucí práce RNDr. Adam Rambousek.

[36] Debian -- The Universal Operating System: PXEBootInstall Installing Debian using network booting [online]. 2020 [cit. 2021-5-12]. Dostupné z: https://wiki.debian.org/PXEBootInstall

[37] Debian GNU/Linux Installation Guide: Appendix B. Automating the installation using preseeding. Debian.org [online]. Debian, 2019 [cit. 2021-5-12]. Dostupné z: https://www.debian.org/releases/buster/amd64/apb.en.html

[38] Red Hat Customer Portal: CHAPTER 7. STARTING KICKSTART INSTALLATIONS [online]. © 2021 [cit. 2021-5-16]. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/8/html/performing_an_advanced_rhel_installation/starting-kickstart-installations_installing-rhel-as-an-experienced-user

[39] Hyper-V on Windows Server, 2016. Microsoft [online]. [cit. 2021-5-12]. Dostupné z: https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/hyper-v-on-windows-server

[40] Oficiální domovská stránka Microsoft: Co je PowerShell? [online]. 2021 [cit. 2021-5-16]. Dostupné z: https://docs.microsoft.com/cs-cz/powershell/scripting/overview?view=powershell-7.1

[41] Enterprise Software Delivery | CloudBees: Using Packer and Vagrant to Build Virtual Machines [online]. Motlik, 2017 [cit. 2021-5-16]. Dostupné z: https://www.cloudbees.com/blog/packer-vagrant-tutorial/

[42] HashiCorp: Infrastructure enables innovation: Why Use Packer? [online]. [cit. 2021-5-16]. Dostupné z: https://www.packer.io/intro/why

[43] Zdroják - o tvorbě webových stránek a aplikací: Úvod do Vagrantu [online]. Martin Svačinka, 2015 [cit. 2021-5-16]. Dostupné z: https://zdrojak.cz/clanky/uvod-do-vagrantu/

[44] Opensource.com | Opensource.com: What is Vagrant? [online]. © 2021 [cit. 2021-5-16]. Dostupné z: https://opensource.com/resources/vagrant

[45] ŠTĚPÁNEK, Jan. Automatická správa laboratoře s OS Windows pomocí technologie Puppet. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra počítačů. Vedoucí práce Ing. Ondřej Votava.

[46] KADLÍČEK, Jan. Automatická konfigurace virtuální infrastruktury. Brno, 2020. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce Ing. Václav Uher, Ph.D.

[47] Ansible is Simple IT Automation [online]. © 2021 [cit. 2021-5-12]. Dostupné z: https://www.ansible.com/

[48] WALT Labs - Cloud Migration & Application Modernization: Ansible, Puppet, Chef, Salt: What Should I Use? [online]. 2020 [cit. 2021-5-12]. Dostupné z: https://waltlabs.io/ansible-puppet-chef-salt-what-configuration-management-tool-should-i-use/

[49] Red Hat Ansible Automation Platform: An expansive foundation for building and operating automation at scale [online]. © 2019 [cit. 2021-5-12]. Dostupné z: https://www.redhat.com/cms/managed-files/ma-ansible-automation-platform-data-sheet-f20327pr-201911-en.pdf?int-cmp=7013a000002DDP9AAO&extIdCarryOver=true&sc_cid=701f2000001OH7YAAW

[50] HUDEC, Miroslav. Automatizace konfigurace síťových prvků pomocí Ansible. Praha, 2016. Bakalářská práce. České vysoké učení technické v Praze, Fakulta elektrotechnická, Katedra telekomunikační techniky. Vedoucí práce Ing. Miloš Kozák, Ph.D.

[51] HOCHSTEIN, Lorin a René MOSER. Ansible: up and running: automating configuration management and deployment the easy way. Second edition. Beijing: O'Reilly, 2017. ISBN 978-1-4919-7980-8

[52] Paul Zhao Projects – Medium: Ansible installation using one controller and 3 AWS instances [online]. Paul Zhao, 2020 [cit. 2021-5-16]. Dostupné z: https://medium.com/paul-zhao-projects/ansible-installation-using-one-controller-and-3-aws-instances-1862e941bd4c

[53] Ansible Documentation: Installing Ansible [online]. Red Hat, 2021 [cit. 2021-5-12]. Dostupné z: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

[54] Ansible Documentation: ansible.builtin.raw – Executes a low-down and dirty command [online]. Red Hat, 2021 [cit. 2021-5-12]. Dostupné z: https://docs.ansible.com/ansible/latest/collections/ansible/builtin/raw_module.html

[55] AbcLinuxu.cz - Linux na stříbrném podnose: Puppet a konfigurační nástroj Augeas [online]. 2013 [cit. 2021-5-16]. Dostupné z: https://www.abclinuxu.cz/blog/kenyho_stesky/2013/4/puppet-a-konfiguracni-nastroj-augeas

[56] Powerful infrastructure automation and delivery | Puppet: Introduction to Puppet [online]. © 2021 [cit. 2021-5-16]. Dostupné z: https://puppet.com/docs/puppet/7.6/puppet_overview.html

[57] KÁBRT, Tomáš. Automatické nasazení aplikací v cloudu řízené modelem výkonnosti. Praha, 2015. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií, Katedra počítačových systémů. Vedoucí práce Ing. Tomáš Vondra.

[58] RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity: Chef - Overview [online]. © 2021 [cit. 2021-5-16]. Dostupné z: https://www.tutorialspoint.com/chef/chef_overview.htm

[59] RxJS, ggplot2, Python Data Persistence, Caffe2, PyBrain, Python Data Access, H2O, Colab, Theano, Flutter, KNime, Mean.js, Weka, Solidity: Chef - Architecture [online]. © 2021 [cit. 2021-5-16]. Dostupné z: https://www.tutorialspoint.com/chef/chef_architecture.htm

[60] DigitalOcean Community | DigitalOcean: An Introduction to SaltStack Terminology and Concepts [online]. Justin Ellingwood, 2015 [cit. 2021-5-16]. Dostupné z: https://www.digitalocean.com/community/tutorials/an-introduction-to-saltstack-terminology-and-concepts#salt-commands

[61] Zdroják - o tvorbě webových stránek a aplikací: SaltStack – sůl pro vaše servery [online]. Tomáš Fejfar, 2013 [cit. 2021-5-16]. Dostupné z: https://zdrojak.cz/clanky/saltstack-sul-pro-vase-servery/

[62] VMware - Delivering a Digital Foundation For Businesses: Support for SaltStack [online]. 2021 [cit. 2021-5-16]. Dostupné z: https://www.vmware.com/support/acquisitions/saltstack.html

[63] Life with SaltStack: SaltStack [online]. © 2021 [cit. 2021-5-16]. Dostupné z: http://zohararad.github.io/presentations/saltstack-config-management-that-rocks/

[64] HashiCorp: Infrastructure enables innovation: Download Vagrant [online]. [cit. 2021-5-12]. Dostupné z: https://www.vagrantup.com/downloads

[65] Oficiální domovská stránka Microsoft: Create a virtual network [online]. 2016 [cit. 2021-5-12]. Dostupné z: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/quick-start/connect-to-network

[66] Oficiální domovská stránka Microsoft: Run Hyper-V in a Virtual Machine with Nested Virtualization [online]. 2016 [cit. 2021-5-12]. Dostupné z: https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/user-guide/nested-virtualization

[67] AbcLinuxu.cz - Linux na stříbrném podnose: NAT [online]. Praha: Michal Křenek, 2009 [cit. 2021-5-12]. Dostupné z: https://www.abclinuxu.cz/slovnik/nat

[68] Appuals: Fix: An Error Occurred while Trying to Retrieve a List of Virtual Switches in Hyper-V 2019 [online]. Kevin Arrows, 2020 [cit. 2021-5-12]. Dostupné z: https://appuals.com/an-error-occurred-virtual-switches-in-hyper-v-2019/

[69] HashiCorp Learn: Install Packer [online]. Kevin Arrows, 2020 [cit. 2021-5-12]. Dostupné z: https://learn.hashicorp.com/tutorials/packer/get-started-install-cli

[70] Chocolatey Software | Chocolatey - The package manager for Windows: INSTALLING CHOCOLATEY [online]. Chocolatey Software, © 2021 [cit. 2021-5-12]. Dostupné z: https://chocolatey.org/install

[71] HashiCorp Learn: Packer Documentation [online]. [cit. 2021-5-12]. Dostupné z: https://www.packer.io/docs

[72] Debian -- The Universal Operating System: Verze Debianu [online]. 2021 [cit. 2021-5-12]. Dostupné z: https://www.debian.org/releases/

[73] Debian -- The Universal Operating System: Síťová instalace z minimálního CD [online]. 2019 [cit. 2021-5-12]. Dostupné z: https://www.debian.org/CD/netinst/

[74] HashiCorp Learn: Hyper-V Builder (from an ISO) [online]. [cit. 2021-5-12]. Dostupné z: https://www.packer.io/docs/builders/hyperv/iso

[75] Ansible Documentation: Installing Ansible [online]. Red Hat, 2021 [cit. 2021-5-12]. Dostupné z: https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.html

[76] DigitalOcean Community | DigitalOcean: How to Set Up SSH Keys on Debian 9 [online]. Hanif Jetha, 2018 [cit. 2021-5-12]. Dostupné z: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-debian-9

[77] Windows OS Hub | IT knowledge base for system administrators: Configuring SSH Key-Based Authentication on Windows 10/ Server 2019 [online]. 2020 [cit. 2021-5-12]. Dostupné z: http://woshub.com/using-ssh-key-based-authentication-on-windows/

[78] Stack Overflow - Where Developers Learn, Share, & Build Careers: Setting up OpenSSH for Windows using public key authentication [online]. 2013 [cit. 2021-5-12]. Dostupné z: https://stackoverflow.com/questions/16212816/setting-up-openssh-for-windows-using-public-key-authentication/50502015

[79] Ansible Documentation: How to build your inventory [online]. 2021 [cit. 2021-5-16]. Dostupné z: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html

[80] DigitalOcean Community | DigitalOcean: How to Set Up SSH Keys on Debian 9 [online]. Hanif Jetha, 2018 [cit. 2021-5-16]. Dostupné z: https://www.digitalocean.com/community/tutorials/how-to-set-up-ssh-keys-on-debian-9

[81] HashiCorp: Infrastructure enables innovation: Manage Secrets and Protect Sensitive Data [online]. [cit. 2021-5-16]. Dostupné z: https://www.vaultproject.io/

[82] Computerworld.cz | Deník pro IT profesionály: Zabezpečení virtuálních počítačů na úrovni hypervisoru (1) [online]. Security World, 2009 [cit. 2021-5-16]. Dostupné z: https://computerworld.cz/securityworld/zabezpeceni-virtualnich-pocitacu-na-urovni-hypervisoru-1-47035

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AMD	Advanced Micro Devices
API	Application Programming Interface
ARM	Advanced RISC Machine
CD	Compact Disc
GUI	Graphic User Interface
IP	Internet Protocol
IT	Information Technology
JSON	JavaScript Object Notation
KVM	Kernel-based Virtual Machine
MAC	Media Access Control
NAT	Network Address Translations
PXE	Preboot Execution Environment
RHEL	Red Hat Enterprise Linux
RISC	Reduced Instruction Set Computer
SSH	Secure Shell
URL	Uniform Resource Locator
USB	Universal Seriál Bus
UTB	Univerzita Tomáše Bati
VM	Virtual Machine
WinRM	Windows Remote Management
YAML	YAML Ain’t Markup Language

SEZNAM OBRÁZKŮ

SEZNAM PŘÍLOH

PŘÍLOHA P II: MANUÁL NASAZENÍ VIRTUÁLNÍHO STROJE

Manuál

Automatizovaného nasazení virtuálních serverů na platformě Hyper-V

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Bc. Milan Cibulka

Vytvoření základního obrazu virtuálního stroje

Instalace potřebného softwaru

Pro automatizované vytvoření nového obrazu virtuálního stroje na platformě Hyper-V je nutné mít v prostředí, ve kterém bude obraz vytvářen, nainstalovaný nástroj Packer. Instalaci lze provést pomocí správce softwarových balíčků Chocolatey. Jednoduše z příkazového řádku Powershell příkazem:

```
choco install packer
```

Samotnou instalací Chocolatey lze provédst pomocí rutiny:

```
Set-ExecutionPolicy Bypass -Scope Process -Force;
[System.Net.ServicePointManager]::SecurityProtocol
=
[System.Net.ServicePointManager]::SecurityProtocol
-bor 3072; iex ((New-Object Sys-
tem.Net.WebClient).DownloadString('https://
chocolatey.org/install.ps1'))
```

Vytvoření základního obrazu virtuálního stroje

Sestavení obrazu na základě šablony

Ukázková šablona je uložena v elektronické podobě v Příloze P I i s dalšími potřebnými soubory, které využívá k automatizovanému vytvoření obrazu virtuálního stroje s operačním systémem Debian 10. Tato šablona je připravena ke spuštění, které se provede příkazem:

```
packer build debian.json
```

Po dokončení sestavení se vedle souboru s šablonou vytvoří adresář s názvem build, jenž obsahuje požadovaný obraz virtuálního stroje.

Vytvoření základního obrazu virtuálního stroje

Ukázková šablona pro tvorbu VM pomocí Packer

```
  {
    "variables": {
      "vm_name": "debian-10.9-{{timestamp}}",
      "iso_url": "https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/
debian-10.9.0-amd64-netinst.iso",
      "iso_checksum":
"sha512:47d35187b4903e803209959434fb8b65ead3ad2a8f007eef1c3d3284f356ab9955aa7e1
5e24cb7af6a3859aa66837f5fa2e7441f936496ea447904f7dddfdc20",
      "ssh_username": "utb",
      "ssh_password": "utbutb",
      "output_directory": "./build",
      "boxes_directory": "./boxes",
      "scripts_directory": "./scripts",
      "cpus": "2",
      "memory": "1024",
      "disk_size": "21440"
    },
    "builders": [
      {
        "vm_name": "{{ user `vm_name` }}",
        "type": "hyperv-iso",
        "cpus": "{{ user `cpus` }}",
        "memory": "{{ user `memory` }}",
        "disk_size": "{{user `disk_size`}}",
       "output_directory": "{{ user `output_directory` }}",
        "iso_url": "{{ user `iso_url` }}",
        "iso_checksum": "{{ user `iso_checksum` }}",
        "communicator": "ssh",
        "ssh_port": 22,
        "ssh_username": "{{ user `ssh_username` }}",
        "ssh_password": "{{ user `ssh_password` }}",
        "ssh_wait_timeout": "10000s",
        "shutdown_command": "echo {{user `ssh_password`}} | sudo -S /sbin/
shutdown -hP now",
        "http_directory": "./http",
        "boot_wait": "20s",
        "boot_command": [
         "<esc><wait>",
          "auto <wait>",
          "url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_file.cfg <wait>",
          "<enter><wait>"
        ]
      }
    ],
    "provisioners": [
      {
        "type": "shell",
       "execute_command": "echo {{user `ssh_password`}} | {{.Vars}} sudo -S -E
sh -eux '{{.Path}}'",
        "scripts": [
          "{{ user `scripts_directory` }}/update.sh",
          "{{ user `scripts_directory` }}/packages.sh"
        ]
      }
    ]
}
```

Nasazení nového virtuálního stroje

Instalace potřebného softwaru

Pro práci s připravenými Ansible scénáři je nutné mít tento software nainstalovaný. U linuxových distribucí založených na Debianu je možné instalaci provést pomocí příkazu:

```
sudo apt-get install ansible
```

Připravené scénáře využívají také moduly, které nejsou dostupné v základní instalaci, proto je nutné doinstalovat pomocí příkazů:

```
ansible-galaxy collection install ansible.windows
ansible-galaxy collection install community.general
```

Nasazení nového virtuálního stroje

Nasazení

Pro nasazení je nutné nastavení připraveného inventáře pro konkrétního uživatele a Hyper-V server. Jako například inventář pro uživatele m_cibulka na testovacím serveru 10.13.32.17:

```
[utb_win]
10.13.32.17
[utb_win:vars]
ansible_user=m_cibulka
ansible_connection=ssh
ansible_shell_type=powershell
```

K nasazení nového virtuálního stroje na základě připraveného obrazu slouží scénář s názvem start_new_vm.yml z adresáře s připraveným inventářem lze nasazení spustit pomocí příkazu:

```
ansible-playbook -i hosts_inventory playbooks/
start_new_vm.yml
```

Po úspěšném dokončení scénáře se v terminálu vypíše podrobná konfigurace spuštěného virtuálního stroje, kde je nutné získat jeho IP adresu a zaznamenat ji do připraveného inventáře pro linuxové uzly. Příklad:

```
[debian_servers]
10.13.34.229
[debian_servers:vars]
ansible_user=utb
ansible_password=utbutb
ansible_become_pass=utbutb
```

Nasazení nového virtuálního stroje

Konfigurace

Po nastavení inventáře je možné spustit připravený scénář pro síťovou konfiguraci se statickou IP adresou. V souboru s názvem network_interfaces.yml je možné nastavit požadované parametry jako název rozhraní, typ IP adresy, metoda přidělení (statická nebo dynamická), IP adresa, maska sítě, brána sítě nebo DNS server. Spuštění lze povést obdobně jako u samotného nasazení příkazem:

```
ansible-playbook -i hosts_inventory playbooks/
network_interfaces.yml
```

K zprovoznění základního Apache2 serveru slouží scénář apache2_install.yml.

```
ansible-playbook -i hosts_inventory playbooks/
apache2_install.yml
```

Po spuštění tohoto scénáře je nainstalován funkční Apache2 server.

PŘÍLOHA P III: POWERSHELL SKRIPT PRO NASAZENÍ VIRTUÁLNÍHO STROJE

```
param ($sourceVmPath, $vmsFolder, $newVmName, $procesorsCount, $memorySize, $diskSize)

Write-Output "Source Path: $sourceVmPath"
$newVmBasePath = Join-Path $vmsFolder $newVmName
Write-Output "New Base Path: $newVmBasePath"
$newVmPath = Join-Path $newVmBasePath "Virtual Machines"
Write-Output "New Path: $newVmPath"
$newVhdPath = Join-Path $newVmBasePath "Virtual Hard Disks"
Write-Output "New disk path: $newVhdPath"

#vytvoreni noveho VM
Import-VM -Path $sourceVmPath -Copy -GenerateNewId -VirtualMachinePath $newVm-Path -VhdDestinationPath $newVhdPath
Write-Output "New virtual machine created"

#prejmenovani
$newVmBasePath = Join-Path $vmsFolder $newVmName
$vm = Get-VM | Where-Object {$_.Path.StartsWith($newVmBasePath)}
Rename-VM -VM $vm -NewName $newVmName
Write-Output "New virtual machine renamed"

#nastaveni poctu virtualnich procesoru
Set-VMProcessor $newVmName -Count $procesorsCount
Write-Output "Number of procesors was set to $procesorsCount"

#nastaveni velikosti operacni pameti
Set-VMMemory $newVmName -StartupBytes $memorySize
Write-Output "Memory size was set to $memorySize"

#nastavení velikost disku
$disk = Get-VMHardDiskDrive -VMName $newVmName
Resize-VHD -Path $disk.Path -SizeBytes $diskSize
Write-Output "Disk size was set to $memorySize"

#spusteni nového VM
Start-VM -Name $newVmName

#ziskani informaci o novem VM
Start-Sleep -s 45
$networkInfo = Get-VM -Name $newVmName | Select -ExpandProperty Networkadapters | Format-list *
Write-Output "New virtual machine info:"
Write-Output $networkInfo
```

PŘÍLOHA P IV: ŠABLONA NÁSTROJE PACKER

```
{
  "variables": {
    "vm_name": "debian-10.9-{{timestamp}}",
    "iso_url":    "https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-10.9.0-amd64-netinst.iso",
    "iso_checksum": "sha512:47d35187b4903e803209959434fb8b65ead3ad2a8f007eef1c3d3284f356ab9955aa7e15e24cb7af6a3859aa66837f5fa2e7441f936496ea447904f7dddfdc20",
    "ssh_username": "utb",
    "ssh_password": "utbutb",
    "output_directory": "./build",
    "boxes_directory": "./boxes",
    "scripts_directory": "./scripts",
    "cpus": "2",
    "memory": "1024",
    "disk_size": "21440"
  },
  "builders": [
    {
      "vm_name": "{{ user `vm_name` }}",
      "type": "hyperv-iso",
      "cpus": "{{ user `cpus` }}",
      "memory": "{{ user `memory` }}",
      "disk_size": "{{user `disk_size`}}",
        "output_directory": "{{ user `output_directory` }}",
      "iso_url": "{{ user `iso_url` }}",
      "iso_checksum": "{{ user `iso_checksum` }}",
      "communicator": "ssh",
      "ssh_port": 22,
      "ssh_username": "{{ user `ssh_username` }}",
      "ssh_password": "{{ user `ssh_password` }}",
      "ssh_wait_timeout": "10000s",
      "shutdown_command": "echo {{user `ssh_password`}} | sudo -S /sbin/shutdown -hP now",
      "http_directory": "./http",
      "boot_wait": "20s",
      "boot_command": [
       "<esc><wait>",
        "auto <wait>",
        "url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed_file.cfg <wait>",
        "<enter><wait>"
      ]
    }
  ],
  "provisioners": [
    {
      "type": "shell",
```

```
          "execute_command": "echo {{user `ssh_password`}} | {{.Vars}} sudo -S -E sh -eux
'{{.Path}}'",
        "scripts": [
          "{{ user `scripts_directory` }}/update.sh",
          "{{ user `scripts_directory` }}/packages.sh"
        ]
      }
    ]
}
```