VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

AUTOMATIZACE POMOCÍ ANSIBLE VE WINDOWS

AUTOMATION USING ANSIBLE IN WINDOWS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **JIŘÍ KLÍČ**
AUTHOR

VEDOUCÍ PRÁCE **Mgr. ROMAN TRCHALÍK, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/20483/2017/xklicj00

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Klíč Jiří**
Obor: Informační technologie
Téma: **Automatizace pomocí Ansible ve Windows**
Ansible for Windows
Kategorie: Počítačové sítě

Pokyny:

1. Seznamte se nástrojem Ansible a dalšími existujícími podobnými nástroji pro nasazení v OS MS Windows 7, 10 a 2012 Server.
2. Navrhněte a vytvořte nástroj (sadu skriptů) pro evidenci HW (typy komponent, sériová čísla, verze firmware, apod.).
3. Navrhněte a vytvořte nástroj pro sledování zdrojů (CPU, RAM, disk I/O, síťová rozhraní, apod.). Výstupy pak znázorněte v grafu (např. MS Excel).
4. Navrhněte sadu skriptů pro běžnou správu, např. přenášení souborů, instalace aplikace, získání seznamu instalovaných programů, apod.
5. Proveďte vyhodnocení a srovnání s tradičními nástroji, které Windows nabízí (např. Desired State Configuration).

Literatura:

- Ansible Documentation. Red Hat, Inc. [online]. Rev. 2017-10-01 [cit. 2017-10-01]. Available at URL: < http://docs.ansible.com/>.
- Windows PowerShell Desired State Configuration Overview. Microsoft. 2017 [cit. 2017-10-01]. Available at URL: < https://docs.microsoft.com/en-us/powershell/dsc/overview>.
- Dále podle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Trchalík Roman, Mgr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav informačních systémů
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zabývá především možnostmi, jak automatizovat správu počítačů a získávání dat o počítačích. Práce se zaměřuje na automatizaci počítačů s operačním systémem Windows spravovaných z operačního systému Linux. Automatizace je zde prováděna z textového uživatelského rozhraní. Seznamuje čtenáře s využitým nástrojem Ansible a jeho komponenty a využitím. Probírají se zde jak metody získávání potřebných dat, tak i jejich význam a interpretace. V práci jsou získávána jak statická data, tak i data měnící se s časem. Také jsou zde rozebírány možnosti správy softwarového vybavení počítače.

Abstract

The main purpose of this bachelor thesis is dealing with possibilities of automation of the computer management and acquisition of data about computers. This thesis focuses on automation of Windows-based computers managed by Linux operating system. The automation is performed by text user interface. It presents to the reader the used tool named Ansible and its components and usages. Here are discussed both methods of collecting of required data and their meaning and interpretation. Some data which are collected in this work are static, the rest of them use to change their values by time. There are also discussed possibilities how to manage the software equipment of the computer.

Klíčová slova

Automatizace, Ansible, Windows, HW data, DSC, Desired State Configuration, konfigurace

Keywords

Automation, Ansible, Windows, HW data, DSC, Desired State Configuration, configuration

Citace

KLÍČ, Jiří. *Automatizace pomocí Ansible ve Windows*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Mgr. Roman Trchalík, Ph.D.

Automatizace pomocí Ansible ve Windows

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Mgr. Romana Trchalíka Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jiří Klíč
11. května 2018

Poděkování

Rád bych poděkoval svému vedoucímu práce, panu Mgr. Romanu Trchalíkovi za odbornou pomoc, konzultace a návrhy jak mou práci dále vylepšovat.

Obsah

Kapitola 1

Úvod

Podíváme-li se v současné době na firmy a na způsob, jakým mají zajištěnou správu počítačů, můžeme je rozdělit do dvou kategorií. První kategorií jsou velké firmy, které mají více správců a k tomu až tisíce zaměstnanců a počítačů. Tito správci musejí mít většinu akcí zatomatizovanou, jinak by nemohli stíhat řešit věci, které se od nich požadují. Druhou kategorií je menší firma, čítající pár (desítek) zaměstnanců. Správce mají buďto vlastního, nebo platí firmu která jim tyto služby zajišťuje. Pokud mají správce vlastního, je možné, že tento správce dělá vše ručně. Od instalací programů až po instalace celého počítače. Mnohdy o nějaké automatizaci, jejich možnostech a výhodách ani nemusel slyšet. Nebo o ní třeba slyšel, ale může si myslet, že naučení se jí, by mu zabralo více času než ruční správa.

Tato práce se zabývá hromadnou správou zařízení, na kterých běží operační systém Windows. Tato zařízení mohou být spravována jednotlivě, avšak již při pár zařízeních je to velice zdlouhavé, a administrátor by mohl tento čas využít efektivněji. Pokud navíc zařízení není lehce přístupné (je např. na jiné pobočce, v jiném městě) může si několika minutová úprava vyžádat i několik hodin kvůli přepravě. Začali tak vznikat programy, za jejichž pomoci se může administrátor vzdáleně připojit a nastavení upravit. Nástroj pro tyto účely, pojmenovaný vzdálená plocha, implementoval i Microsoft do některých verzí Windows.

Postupem času však přibývá stále více osobních počítačů, i serverových stanic. Vznikají tak nástroje, které umožňují hromadnou správu těchto zařízení. Jedním z těchto nástrojů je i freeware nástroj Ansible.

Cílem této práce, je vytvořit sadu nástrojů, které za pomoci nástroje Ansible pomohou administrátorovi ulehčit hromadnou správu. Nejdříve vytvoříme nástroje pro evidenci hardwaru a jeho stavu. Dalším krokem potom bude vytvoření nástrojů, které budou sbírat dlouhodobé statistiky o využití prostředků dané stanice. Nesmíme také zapomínat na práci se soubory a aplikacemi, čemuž se bude věnovat předposlední část této práce. V poslední částí se zabývám podobnými nástroji.

Kapitola 2

Analýza problematiky

Pro obsluhu personálních počítačů a serverů dříve sloužila pouze příkazová řádka. Nebylo však příliš pohodlné s ní pracovat a pro běžného uživatele je zbytečně náročná. Postupem času ji začala nahrazovat grafická uživatelská rozhraní. Ta však kladou větší nároky na prostředky, čas pro vývoj a pro jednoduché programy je leckdy zbytečná. Textový režim však nezmizel. U většiny programů lze navíc tímto způsobem docílit lepších výsledků. Při automatizaci se také využívá textový režim. V této práci je využíván textový nástroj Ansible a textový režim nástroje WMI.

2.1 Ansible

Jedná se o sadu nástrojů, která umí shromažďovat informace, vykonávat příkazy a řešit problémy na mnoha stanicích současně. Jde o freeware nástroj pro operační systém Linux, původně určený pro hromadnou správu zařízení s OS Linux. Od verze 1.7 však umožňuje i správu zařízení běžících na OS Windows. Od verze Windows 10 a představení jeho subsystému pro Linux, může Ansible běžet i zde. Pro úspěšné spuštění Ansible je potřeba mít nainstalovaný Python verze 2.6 nebo vyšší. Jeho velkou výhodou je, že na cílové stanici není třeba nic instalovat, stačí povolit příslušný port ve firewallu a spustit nasluchací proces. Tento nástroj má i velmi dobře zpracovanou dokumentaci [2], která byla využita při vývoji.

Připojení

Pro připojení k Windows lze použít několik možností (viz tabulka 2.1). V této práci je používán CredSSP, který umožňuje přihlašování jak přes lokální účty, tak přes doménové přihlášení. Ten vyžaduje na cílové stanici TLS verze 1.2., která je přítomna v základní instalaci OS Windows 8, Windows server 2012 a výše. Pro některé starší Windows systémy lze tuto verzi doinstalovat.

Inventory

Pro snadnou správu zařízení využívá Ansible tzv. inventory. Jedná se o soupis skupin zařízení pod jednotným označením. Mohu například vytvořit skupinu Brno, do které vložím IP adresy, či doménová jména pro zařízení umístěné na pobočce v Brně, a druhou skupinu Praha. Pokud budu chtít zjistit, jestli jsou v Brně zapnutá zařízení, budu pracovat pouze se skupinou Brno.

Option	Local Accounts	Active Directory Account
Basic	Yes	No
Certificate	Yes	No
Kerberos	No	Yes
NTLM	Yes	Yes
CredSSP	Yes	Yes

Tabulka 2.1: Tabulka srovnávající možnosti přihlášení[3]

Ad-hoc Příkazy

Chceme-li Ansible využívat příležitostně, či pro základní úkony, máme k dispozici ad-hoc příkazy. Ty zadáváme jednoduše z příkazové řádky. Např pro zjištění, zda máme vše nastaveno správně a cílová stanice je dostupná, použijeme příkaz (win_)ping. Příkaz spustíme zadáním příkazu `ansible windows -m win_ping`, kde windows je výčet stanic, uložený v inventory souboru.

Playbook

Pokud chceme naplno využít potenciál Ansiblu a nevypisovat pokaždé ty stejné příkazy jeden po druhém, využijeme k tomu tzv. playbooky. Ty jsou psány v YAML formátu. Každý jeden příkaz je tzv. `play`, ty lze libovolně skládat za sebe.

Nejprve si určíme, s kým budeme komunikovat. Lze vypsat hosta, nebo můžeme použít inventory. Dále zadáme uživatelské jméno a heslo. Opět máme dva způsoby zadání. Vypsat přímo to playbooku nebo do souboru. Později se lze přepnout na jiný účet.

Play se nemusí provádět jen na vzdáleném PC, ale můžeme se přepnout i pro práci na lokálním zařízení.

Po vykonání playbooku nám Ansible vypíše, kolik úkolů se podařilo, kolik úkolů něco změnilo, jestli se nějaké zařízení nepodařilo kontaktovat a kolik úkolů nebylo provedeno kvůli chybě.

Zápis playbooku, kterým se chceme dotázat, zda je daná stanice aktivní, vypadá následovně. Nejdříve zvolíme název playbooku, poté jeho cíle, popř. proměnné a nakonec nadefinujeme požadované operace.

```
- name: HW Info
  hosts: windows
  tasks:

    - name: ping
      win_ping:
```

Výstup má poté formát, který lze vidět v další ukázce. Na prvním řádku vidíme spuštění playbooku z příkazové řádky. Můžeme si všimnout názvu, pokusu o kontaktování cílů, vykonání našeho požadavku a rekapitulace. Jedna z cílových stanic se nepodařila kontaktovat, a tak je vynechána z dalších úloh.

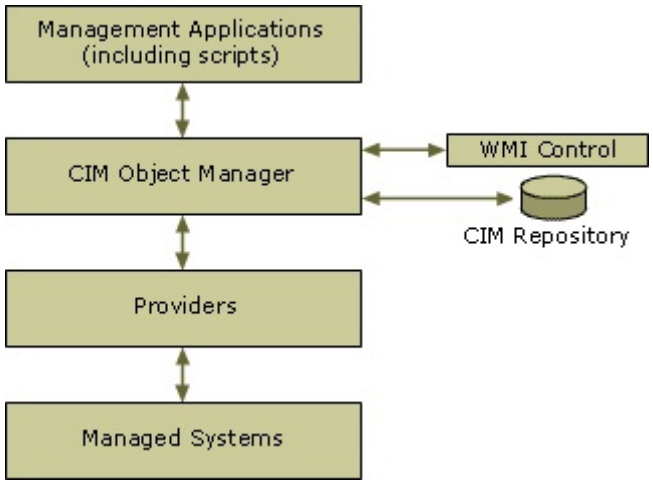
```
jirka@jirka-VirtualBox:~/Plocha/ansible/Macros$ ansible-playbook PlayPing.yml

PLAY [HW Info] *****************************************************************

TASK [Gathering Facts] *********************************************************
ok: [10.0.10.2]
fatal: [10.0.10.21]: UNREACHABLE! => ....

TASK [ping] ********************************************************************
ok: [10.0.10.2]
to retry, use: --limit @/home/jirka/Plocha/ansible/Macros/PlayPing.retry

PLAY RECAP *********************************************************************
10.0.10.2                  : ok=2    changed=0    unreachable=0    failed=0
10.0.10.21                 : ok=0    changed=0    unreachable=1    failed=0
```

2.2 Spouštěcí skript

Pro co možná nejjednodušší využití vytvořených nástrojů pro Ansible, byl vytvořen také skript pro spouštění. Jedná se o jednoduchý skript, který lze spustit v několika režimech. Tyto režimy se volí argumentem a v případě že argument zadán není, či je zadán špatně, skript vypíše chybu a ukončí se. Skript bere v potaz první, popř. i druhý argument, a ostatní ignoruje. Pokud se tedy správce přepíše, a zadá argumenty tři, skript zkontroluje pouze první ,nebo první dva, a na jeho základě rozhodne.

2.3 WMIC

Jde o textový režim služby Windows Management Instrumentation, určený pro získávání informací ze vzdálených počítačů. Tato služba dokáže zjistit o počítači nepřeberné množství informací. V této práci je využívána pro sběr hardwarových informací o koncových stanicích. Pro téměř všechny výstupy je využit tzv. `CIM - Common Information Model`. Jedná se o objektově orientovaný datový model shromažďující informace o různých částech PC. Architekrutu `WMI` a `CIM` lze vidět na obrázku 2.1.



Obrázek 2.1: Architektura WMI[7]

Kapitola 3

Evidence HW

Pro správce systému je důležité mít přehled o počítačích, o které se stará a o jejich stavu a konfiguraci. Při psaní nástrojů pro sběr hardwarových informací je důležité mít na paměti, že není důležité vybrat co nejvíce informací o zařízení, ale především vybrat ty podstatné. To bylo velkou součástí této části práce, kdy možnosti výpisů jsou velmi komplexní a vybrat z nich ty nejdůležitější není jednoduché. Občas se zdálo, že informace byla důležitá, ale odstupem času byl tento názor přehodnocen, a naopak. Pro sběr těchto informací je využíváno podlužby WMI a to konkrétně `Win32 Provider` [5].

3.1 Základní stavy měřených komponent

`CIM` využívá u některých výstupů tzv. výčtový typ. Místo delšího výpisu se tedy zobrazuje pouze číslo, které je poté nutné dohledat. Pro lepší přehlednost jsou zde uvedeny některé důležité a často používané informace. Další získávané informace budou popsány u patřičných komponent.

- `Availability` - Jedná se dostupnost, či stav daného zařízení, je vypsáno u každého zařízení, kde je to možné.. Např. číslo 3 znamená `běží/plný výkon`, číslo 4 má význam `pozor`, číslo 7 znamená `vypnuto`, číslo 18 znamená `pozastaveno` atd.
- `Capabilities` - Jako jeden mála záznamů vrací více hodnot. Jde o možnosti přístupu k mediu. Např. číslo 2 znamená `sekvenční přístup`, číslo 3 naopak `náhodný přístup`, číslo 5 znamená, že médium podporuje `šifrování` atd.
- `Caption` - Důležitý záznam `Caption` krátce a výstižně popisuje dané zařízení. Jedná se o jednořádkový zápis.
- `ConfigManagerError` - V případě nefunkčnosti, či problému se zařízením může pomoci celočíselný záznam `ConfigManagerError`. Např. číslo 0 znamená `zařízení nefunguje správně`, číslo 10 znamená `zařízení nelze spustit`, číslo 22 znamená `zařízení je zakázáno` a podobně.
- `Status` - Tento záznam nám udává, v jakém stavu se zařízení aktuálně nachází. Na rozdíl od jiných, podobných záznamů však nejde o výčtový typ. Je to umožněno především tím, že jde o jedno, či dvou slovné stavy. Výstupem může být např stav `OK`, `Error`, `Unknown` atd.

Pro lepší představu o využití a významu těchto stavů přidávám část výpisu informací o discích.

```
Availability  Capabilities  Caption                    ConfigManagerErrorCode  Status
3             {3, 4, 10}    Samsung SSD 850 EVO 250GB  0                       OK
```

3.2 Evidované komponenty

3.2.1 Systémové prostředky

CPU

Informace o procesoru jsou důležité, jelikož jde o hlavní komponentu počítačů.

Prvním informací, která nám říká zda jde o 32, či 64 bitový systém je `AdressWidth` záznam. Následuje záznam `Architecture` který nám říká, o jaký typ architektury se jedná. Může jím být např. `x86/x64/PowerPC` atd. Dalšími položkami jsou `Availability` a `Caption`. Položka `DeviceID` má význam především ve víceprocesorových systémech, kdy se specifikuje, o který procesor se jedná. `ExtClock` udává základní frekvenci pro přenos dat z RAM paměti nebo do ní. Další položkou je `Manufacturer`, která udává výrobce procesoru. `MaxClockSpeed` udává maximální frekvenci procesoru. Nebere však v potaz přetaktování nastavené již z výroby (např HyperBoost u společnosti Intel). Položka `Name` specifikuje model procesoru. Následující dvě položky `NumberOfCores` a `NumberOfLogicalProcessors` udávají počet fyzických jader procesoru a počet logických jader(Hypetreading). Poslední položkou je `Role`, která určuje funkci procesoru. Jde například o matematický, či centrální procesor.

RAM

Operační systémy, stejně jako pod ním běžící procesy, vyžadují stále větší operační paměti. Ty se tak musí často vyměňovat za novější a kapacitně větší. Také je dobré mít přehled, kolik paměti a jakého typu ve stroji je. Proto jsou o nich shromažďovány informace.

Záznam `BankLabel` nám určuje, do které zásuvky je daný modul operační paměti zasunutý. Tato informace se nám hodí zejména v případě, že chceme daný modul měnit. Nemusíme potom vytahovat jeden po druhém a zkoušet, jestli je to ten, který chceme vyměnit. Značení zásuvek najdeme v dokumentaci základové desky, popř. mohou být vyznačeny přímo na desce. `Capacity` udává velikost operační paměti daného modulu v bytech. Následuje `Caption`, a za ním `ConfiguredClockSpeed`, podle kterého zjistíme, na jaké frekvenci operační paměti běží. Hodnota je vypsána v MHz. Dalším záznamem je záznam `FormFactor`, který číselnou hodnotou udává typ paměti. Následujícím záznam `HotSwappable` udává, zda je modul možné vyměnit během chodu počítače. Jde o velmi málo rozšířenou funkcionalitu, některé servery a operační systémy toto však umožňují. `Manufacturer` udává číslo výrobce modulu. `MaxVoltage` udává maximální možné napětí v milivoltech, pro které je daný modul specifikován. Tato informace je důležitá, pokud chceme paměti taktovat na vyšší frekvence a zvyšovat u toho napětí. na druhou stranu nám záznam `MinVoltage` udává nejnižší certifikované napětí.

Pro zjištění typu modulu využijeme záznam `MemoryType`. Bohužel tento záznam rozpozná nanejvýš typ DDR2, a tak v době, kdy používáme DDR4, není naplno využitelný. Důležitým záznamem je záznam `PartNumber`, který udává konkrétní modul. To může být stěžejní informace při hledání správné náhrady, se stejnými parametry za modul, nebo pokud chceme přidat moduly další. `SerialNumber` potom udává sériové číslo. Záznam `Speed`

udává maximální možnou rychlost modulu, udanou v MHz. Následuje `Status` a po něm záznam `Tag`. Jde o unikátní identifikátor modulu.

3.2.2 Úložná zařízení

CD-ROM

Z dnešního pohledu může jít o již zastaralé periferní zařízení, ale na velkém množství počítačů jej stále najdeme a může být využíván.

Jako první záznam je ude evidován `Availability`. Následují záznamy `Capabilities`, `Caption` a `ConfigManagerErrorCode`. `DeviceID` jednoznačne identifikuje zařízení v systému. Následující záznam `Drive` nám ukazuje, na jaké písmeno je mechanika v systému namapována. Pro zjištění výrobce mechaniky, slouží záznam `Manufacturer`. Dále je tu záznam `NeedsCleaning`, který nám říká, zda je potřeba mechaniky vyčistit. Následující čtyři záznamy `SCSIBus`, `SCSILogicalUnit`, `SCSIPort` a `SCSITargetId`, které jednoznačně určují zařízení na sběrnici SCSI. Jako poslední položku je zde sériové číslo mechaniky, zobrazené záznamem `SerialNumber`.

Disk

Se stále se zvyšující datovou náročností jak programů, tak například multimediálních souborů je důležité mít přehled o tom, jaké disky máme nainstalované, jak jsou připojeny a jaká je jejich kapacita.

jako první záznam máme znovu `Availability`. Následuje položka `Capabilities`, která může být složena z více hodnot výčtového typu. Následuje `Caption` a `ConfigManagerError`. Položka `DeviceID` specifikuje disk v rámci systému. Každý disk má rozdílné ID. Další důležitou položkou je `FirmwareRevision`, neboli verze firmwaru disku. Pokud by výrobce zjistil, že je ve firmwaru chyba, vydá opravu. Správce si poté zjistí, zda nemá onu špatnou verzi a v případě potřeby provede update.

Jestliže bude chtít správce nějaký disk vyměnit, bude ho zajímat, jakým konektorem je patřičný disk připojen. Pro to slouží záznam `InterfaceType`. Ten může nabývat pěti hodnot, kterými jsou SCSI, HDC, IDE, USB a 1394, neboli FireWire. SATA zde bohužel chybí, a tak jsou tyto disky označeny jako IDE. Následuje `Manufacturer` a `Model`, které určují výrobce disku a konkrétní model. Dále zde máme záznam `partitions` určující na kolik oddílů máme daný disk rozdělený.

Následuje čtveřice informací `SCSIBus`, `SCSILogicalunit`, `CSCIPort` a `SCSITargetId`, definujících disk na sběrnici SCSI. Po této čtveřici zde máme `SerialNumber`, které zobrazuje sériové číslo disku udané výrobcem. V neposlední řade je evidována položka `Size` udávající velikost disku v bytech. Posledním záznamem je `Status`, který popisuje stav disku.

3.2.3 Rozšiřující karty

Síťové karty

V dnešní době je pro některé operace velmi důležitý Internet. I proto je zahrnut v tomto nástroji.

Jako první záznam je evidován `AdapterType`, který nám říká, jakým typem spojení jsme připojeni. Může jít na příklad o `Wireless`, neboli bezdrátové spojení, `Ethernet 802.3`, neboli kabelové připojení, `Token Ring 802.5` apod. Dalším záznamem je `AutoSense`, který

nabývá hodnot `TRUE` nebo `FALSE` a říká nám, zda umí toto rozhraní automaticky zjistit rychlost připojeného zařízení.

Následují záznamy `Availability`, `Caption` a `ConfigManagerError`. Dalším záznamem je `DeviceID` jednoznačně určující zařízení v systému a záznam `GUID`. Jde o sto dvaceti osmi bitové číslo typu integer, které by mělo být unikátní v globálním měřítku pro identifikaci zdroje. Slovo mělo je zde uvedeno záměrně, neboť toto není garantováno. Je zde však opravdu miniaturní pravděpodobnost, že by dvě zařízení vygenerovala dvě stejná čísla. Následuje záznam `MACAddress`, který zařízení identifikuje v lokální síti. Zde by se opět mělo jednat o unikátní číslo, kdy první polovinu má každý výrobce přidělenou a druhou polovinu si určuje sám. Tato adresa jde však změnit, a navíc některá levná zařízení tuto unikátnost nedodržují a může tak na síti vznikat problém při doručování.

Záznam `Manufacturer` nám říká, kdo je výrobcem daného síťového adaptéru. Následující záznam `NetConnectionID` nám vrátí název, pod kterým je adaptér veden v záložce `Síťová připojení` v ovládacích panelech. `NetConnectionStatus` vrací číselnou hodnotu, která udává, zda je zařízení připojeno, odpojeno, či je hardware zakázán atd. Zda je adaptér povolen lze také zjistit z následujícího záznamu `NetEnabled`, který je typu boolean. Záznam `PhysicalAdapter` je taktéž typu boolean a uvádí, zda se jedná o fyzické, či logické zařízení.

Následuje záznam `SeviceName`, který vypíše název služby, tento název je většinou kratší než název zařízení. Velice důležitou informací může být rychlost daného rozhraní, kterou nám ukáže záznam `Speed`. Posledním záznamem je `Status`.

Grafické karty

Grafický procesor je taktéž důležitou součástí počítače, jelikož zajišťuje vykreslování na monitor. Mimo to však umožňuje pomocí specializovaných nástrojů provádět výpočty, které by procesoru zabrali několikanásobně delší čas.

`AdapterCompatibility` nám zobrazí výrobce grafické karty (NVIDIA/AMD/Intel...). Některé výpočty mohou být náročné na paměť, proto je dobré vědět, jakou kapacitou disponuje grafická karta. Pro toto zjištění nám poslouží záznam `AdapterRAM`, vracející hodnotu v bytech. Následuje `Availability`, `Caption`, `ConfigManagerError` a `DeviceID`.

Poté je evidována dvojice `DriverDate` a `DriverVersion`, díky kterým můžeme přehledně poznat, jakou verzi ovladače máme nainstalovanou na naší stanici. Neméně důležitou součástí pro zjišťování stavu a závad zařízení je konfigurační soubor, cestu k němu, a konkrétní sekci lze zobrazit pomocí `InfFilename` a `InfSection`. V současné době na trh proniká stále větší počet monitorů se zvyšující se hodnotou obnovovací frekvence. Maximální možnou hodnotu obnovovací frekvence připojeného monitoru lze zobrazit pomocí `MaxRefreshRate`. Pro zobrazení podporovaného protokolu pro připojení, který nás zajímá např. v případě že chceme grafickou kartu vyměnit, použijeme záznam `ProtocolSupported`. Následuje záznam `Status`. Záznam `VideoModeDescription` nám vypíše momentálně používané rozlišení ve formátu `šířka x výška x počet barev`. Hodnoty výška a šířka jsou udané v pixelech. Posledním záznamem je `VideoProcessor`, což je řetězec volného tvaru popisující grafickou kartu.

Řadič SCSI

SCSI je v dnešní době chápáno jako rychlé propojení disků. Tato sběrnice však umožňuje propojovat různorodá zařízení s počítačem.

Jako první je tu opět záznam `Availability`, následovaný záznamem `Caption` a také `ConfigManagerErorCode`. Poté vidíme záznam `DeviceID`. Záznam `DeviceMap` nám říká, v

jakém pořadí jsou zařízení uvedena v tomto SCSI řadiči. Pro zjištení názvu SCSI řadiče použijeme záznam `DriverName`. Dalším záznamem jménem `HardwareVersion` zjistíme verzi řadiče a záznamem `Index` zase indexové číslo řadiče v registrech systému. Pro zjištění jména výrobce využijeme záznam `Manufacturer`. Také nás zajímá šířka sběrnice. Tuto informaci nám sdělí záznam `MaxDataWidth`. Maximální možný počet přímo komunikujících zařízení zjistíme ze záznamu `maxNumberControlled`. Důležitým parametrem je také maximální dosažitelná rychlost přenosu. Tu nám udává záznam `MaxTransferRate`. Posledním záznamem je `Status`.

3.2.4 Ostatní

Základní informace

Jedná se o obecnou konfiguraci počítače. Pro tento výpis je použit příkaz `systeminfo`. Ten vypíše název stanice, verzi operačního systému, typ zařízení(server/PC), výrobce a označení modelu, procesor, verzi BIOSu, nastavený jazyk a časové pásmo, velikost fyzické i virtuální operační paměti, doménu, nainstalované opravné balíčky Windows, základní informace o síťové kartě a další. Tyto informace lze považovat za základní informace o stanici, se kterými lze dále pracovat, popřípadě zjišťovat konkrétnější specifikace systému a periférií.

Boot

Nejedná se sice o čistě hardwarovou záležitost, má to ovšem co dělat s diskem a jsou zde zobrazeny důležité informace. Ve výpisu lze vidět např. bootovací složku a cestu k dočasnému adresáři `tmp`.

Ventilátory

Pro spoustu serverů může být velice důležité chlazení, proto jsou v této práci evidovány i ventilátory.

Prvním dvojicí záznamů jsou `Availability` a `Caption`. Záznam `DeviceID` nám jednoznačně určuje zařízení v systému. Následuje záznam `Status` a jako poslední zde vystupuje záznam `VariableSpeed`. Pokud je jeho hodnota `TRUE`, pak ventilátor podporuje variabilní rychlost otáček.

3.3 Výstup

Pro zlepšení čitelnosti a přehlednosti získaných dat je dobré zamyslet se nad formátem zápisu dat. Při použití vytvořeného nástroje má správce dvě možností výstupů, a to textový soubor a tabulkový formát `csv`.

Obyčejný textový soubor je méně přehledný, ale nepotřebuje žádné dodatečné programy pro zobrazení. Jednotlivé položky jsou odděleny mezerami a jsou zarovnány do sloupců.

Pro zobrazení tabulkového formátu `csv` stačí taktéž standardní nástroj pro psaní textu, ale jednotlivé položky jsou oddělené čárkami namísto mezer se zarovnáním, a soubor je tak velmi špatně čitelný. Pro dobré zobrazení je lepší nainstalovat patřičný software (např. MS Excel pro Windows, LibreOffice Calc atd). Formát `csv` však nepodporuje více listů v jednom souboru a pro spojený soubor `all.ods` je tento software nutný. Výhodou je lepší přehlednost a možnost seřadit si výstupy dle kritérií.

Základní režim

Základní režim se spustí argumentem `basic`. Tento režim spustí nástroj Ansible za účelem zjištění základních informací o počítači. Výstup je uložen v podsložce `Basic info`, výstupní složky. Pro každý počítač je zde samostatný soubor, pojmenovaný dle názvu počítače.

Textový režim

Druhým režim se zapne zadáním argumentu `text`. Základním formátem výstupu nástroje `wmic` je `table`, který každý záznam s jeho hodnotami zapisuje na samostatný řádek. Nevýhodou takovéhoto zápisu je velmi široký záznam při výpisu více hodnot, který je méně intuitivní a pohodlný než vertikální posuv. Výhodou je ovšem velmi dobrá porovnatelnost s ostatními zařízeními. Tento formát je použit u všech výstupů získaných pomocí `wmic`. Struktura výstupních souborů je odlišná od prvního režimu. Začínáme opět v základní výstupní složce, ale každý počítač tu tentokrát má vlastní složku, která nese název tohoto počítače. V tomto adresáři se poté nachází složka pojmenovaná `TXT`, která obsahuje adresáře, jejichž jména jsou data a časy spuštění nástroje. V těchto adresářích už jsou poté jednotlivé výstupy.

Další možností je volba `list`, kdy jsou jednotlivé názvy a jejich hodnoty zapisovány na samostatný řádek stylem `nazev=hodnota`. Pro tento projekt to však není vhodné, z důvodu horšího přehledu při více záznamech stejného typu. Dále tento nástroj podporuje výstup ve formátu značkovacího jazyka `XML`, webové tabulky `hform` a `htable` a tabulkového formátu `csv`.

Tabulkový režim

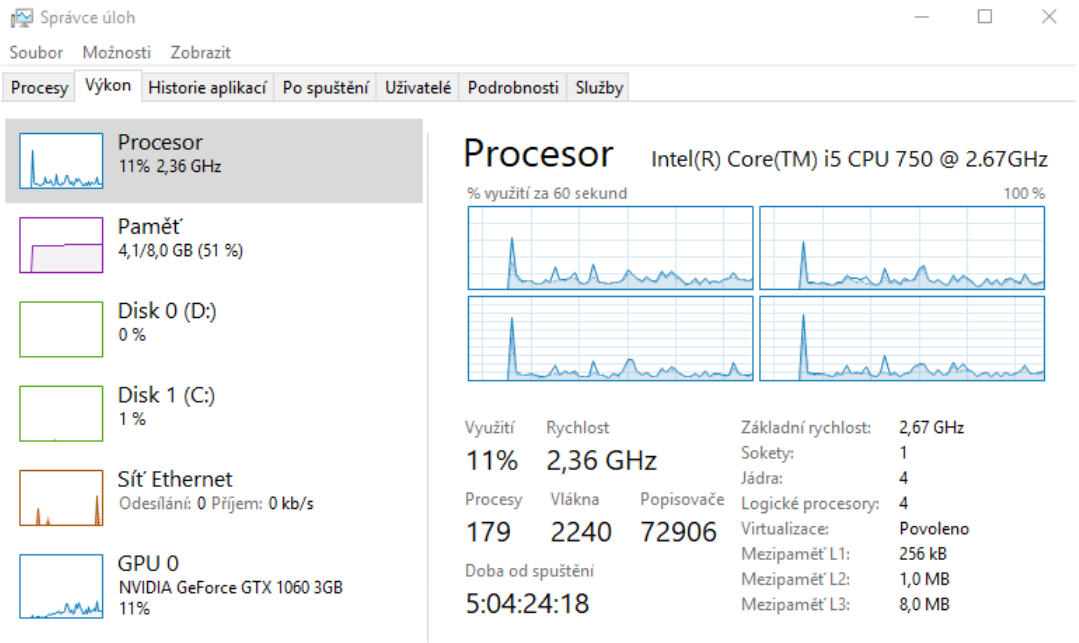
Ve třetím režimu jde o tabulkové zobrazení. Tento režim lze spustit argumentem `csv`. Jeho výstupy jsou ve formátu `csv`. Adresářová struktura je podobná jako u textového režimu. Jediným rozdílem je zde přejmenování složky `TXT` na složku `CSV`. Při experimentování s tímto režimem bylo zjištěno, že pokud jsou ve výstupu uvozovky, `wmic` je špatně ukončuje, a nelze pak tento formát využít bez úpravy souboru. Ve výstupech této práce však takovéto výstupy nejsou nezískávány, a tak je možné pracovat se soubory bez úpravy.

Tento režim však oproti předchozím nezískává pouze data, která uloží, ale spojuje výstupy do jednoho. Původní záměr byl spojit všechny výstupy do jednoho, avšak jeden z těchto výstupů se při pokusu o spojení, či otevření nástrojem `Gnumerical` jeví jako poškozený. Bohužel se nepodařilo zjistit proč, a tak je tento výstup v spojování vynechán. Pro převod z formátu `csv` do formátu `ods` je používán nástroj `ssconvert`, který je součástí balíčku tabulkového procesoru `Gnumeric`.Ten mimo převod formátů a dalších funkcí dokáže z tabulkových souborů vytvořit spojený soubor, kde každý původní soubor je jeden list nového souboru formátu `ods`. Není nutné ani původní soubory do toho formátu převádět, lze využít soubory získané rovnou z výstupu.

Kapitola 4

Využití prostředků

V minulé kapitole jsme získali statické informace o hardwaru. V této kapitole se budeme zabývat získáváním, ukládáním a reprezentací využití prostředků v reálném čase. Tyto data budou pro správce povětšinu času důležitější než data získaná v předchozí kapitole, z důvodu reprezentace stavu daného zařízení v daném čase. Sledováním těchto statistik může správce zjistit zvyšující se využití některého z těchto prostředků, a včas zareagovat dokoupením, či výměnou daného hardwaru. Může tak předejít problémům s delší odezvou, či úplnou nedostupností zařízení v nejvíce využívaný čas. Zároveň mu tyto statistiky mohou pomoci tyto problémy vyřešit u zařízení, u kterých k nim dochází. Předpokládaným výstupem jsou grafy podobné těm, na obrázku 4.1.



Obrázek 4.1: Správce úloh - grafy využití prostředků

4.1 Sběr dat

Pro získávání těchto dat na cílových stanicích s operačním systémem Microsoft Windows se jeví jako ideální cesta využití integrovaného nástroje systému `Sledování výkonu`. Tento

nástroj umožňuje vytvářet tzv. `Sady kolekcí dat`, pod kterými si uživatel zvolí, jaké čitače by chtěl sledovat. V těchto kolekcích lze například nastavit dobu sledování, periodu registrace dat(vzorek každých x sekund), umístění výstupu či formát výstupu. Výchozím formátem výstupu je graf, zobrazitelný v této aplikaci.

Pro automatizaci je však lepší textové prostředí. Naštěstí většina systémových nástrojů toto prostředí má, a ani tato aplikace není výjimkou. Textové prostředí lze zavolat spuštěním programu `logman.exe` a předáním příslušných parametrů. Pro vytvoření kolekce dat existují dva způsoby, jedním z nich je import ze souboru, a druhým je vypsání všech parametrů v příkazové řádce. Pro naše účely (co nejmenší zásah do cílové stanice) je použito zadání parametrů přes Ansible do Powershellu.

Parametry

Pro správnou funkčnost těchto kolekcí je třeba správně vybrat a použít parametry. Pro vytvoření kolekce slouží parametr `-create`, následovaný typem záznamu, který je v našem případě `counter`, neboli čítač. Další možností je například kolekce typu `alert`, neboli upozornění. Poté se zadá jméno této kolekce dat, které musí být unikátní. Následuje přepínač `-c`, za kterým se uvede, které čítače chceme pod danou kolekcí dat sledovat. Linux bohužel neumí zobrazit výchozí výstup tohoto nástroje, jímž je graf, a proto je jako výstup zvolen tabulkový soubor csv. Tento výstup specifikuje přepínač `-f`, následovaný požadovaným typem výstupu. Důležitá je také četnost vzorků, která se specifikuje přepínačem `-si`, za který napíšeme číslo, udávající interval v sekundách, po kterém se zapíše hodnota. Pokud nechceme použít standardní cestu výstupu (`%systemdrive%\PerfLogs\Admin\Nová sada kolekcí dat`) lze využít přepínač `-o`, následovaný cestou pro výstup. Jako poslední přepínač je využita volba `-rf`, následovaná číslem, které udává, jak dlouho (v sekundách) má daná kolekce dat data sbírat.

4.2 Sledované komponenty a parametry

Nástroj `sledování prostředků` umožňuje sledovat velké množství čítačů, většinu z nich však pro tento nástroj, zaměřený na obecné informace o systémových prostředcích, není nutné sledovat. Zde se tedy podíváme na výčet čítačů, které jsou potřebné a důležité. Pro výběr těchto čítačů byl využit návod z podpory Microsoftu [4] (s přihlédnutím k jeho stáří) a vlastní zkušenosti s pozorováním výkonnosti systému.

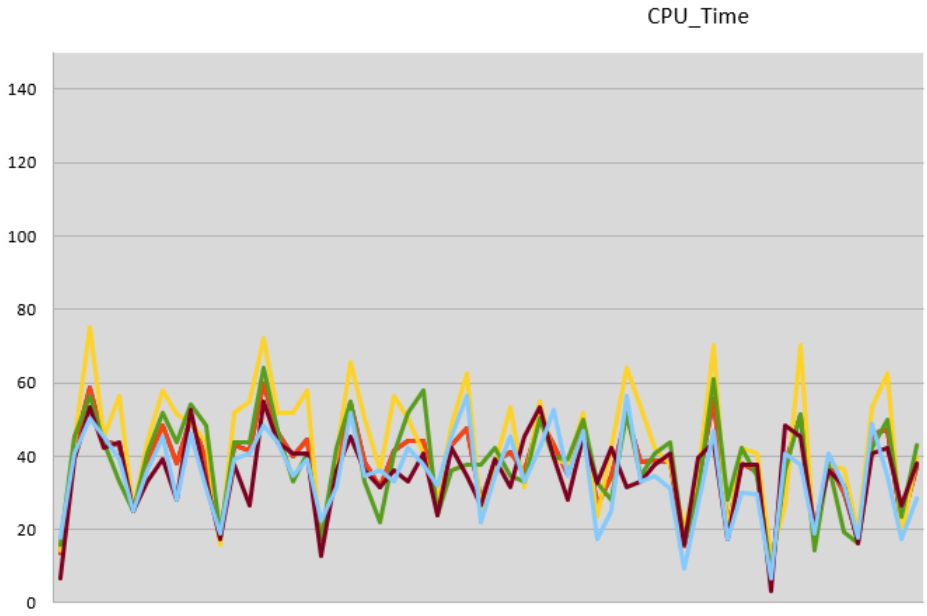
4.2.1 CPU

Využití procesoru je pro nás důležité na stanicích, na kterých probíhají náročné operace, například matematické výpočty, nebo obsluha požadavků dalších stanic. V případě velkého vytížení by tato stanice mohla přestat odpovídat na požadavky, nebo by tyto požadavky trvaly dlouhou dobu. U procesoru je tedy sledován `čas procesoru` a `Výkon procesoru`.

Čas procesoru

Výstupem tohoto čítače je hodnota 0-100%, která udává, jakou dobu strávil procesor(jádro) zpracováváním aktivního vlákna. Tato hodnota se vypočítá odečtením procenta času stráveného ve vláknu nečinnosti od hodnoty 100%. Každé jádro má takovéto vlákno, v němž se akumuluje čas, kdy nejsou připravena žádná jiná vlákna pro zpracování. V moderních procesorech může tento čítač bohužel podhodnotit reálné využití, z důvodu vnitřního intervalu

vzorkování systémových hodin, které je rovno 10 ms. Grafovou část výstupu (bez legendy) lze vidět na obrázku 4.2.



Obrázek 4.2: % času procesoru

Výkon procesoru

Tento čítač nám udává procentuální výkon procesoru(jader) vykonávání požadovaných operací. Tyto procenta jsou odvozena od maximální udávané frekvence procesoru. Novější procesory však díky inovativním technologiím mohou překročit tuto hodnotu, a graf se tak může pohybovat nad hranicí sto procent.

Délka fronty procesoru

Pokud chceme zjistit počet vláken čekajících ve frontě procesoru, využijeme k tomu tento čítač. Pokud se jedná o víceprocesorový počítač, je nutné tuto hodnotu vydělit počtem procesorů. Vše by mělo být v pořádku v případě, že je tato hodnota menší než deset vláken na jeden procesor.

4.2.2 Operační paměť

V operační paměti se nacházejí data programů, se kterými pracuje procesor. Pokud nebudeme mít data uložená, a něco se pokazí, o svá data přijdeme. Proto je dobré mít přehled o volné paměti, ale i o chybách které se stanou. Následující čítače mohou správce upozornit na nutnou výměnu, či doplnění modulů.

Volné megabajty

Již od počátku éry počítačů se zvyšují nároky programů, ale i operačního systému na velikost operační paměti. Správce tedy uvítá, pokud má možnost vidět využití operační paměti v daný čas. Pro zisk těchto dat je v systému přítomný čítač `počet MB k dispozici`, díky kterému se dozvíme dostupné Megabajty v operační paměti.

Bajty v mezipaměti

Tato hodnota nám udává, kolik bajtů se nenachází ve fyzických modulech RAM, ale dočasně uložených na pevných discích. Jelikož je operační paměť je rychlejší, a tyto bajty zatěžují více disk, je dobré se při větší hodnotě tohoto čítače zamyslet nad zakoupením modulů, či jejich výměnou.

Bajty nestránkovaného fondu

Tento čítač nám udává číselnou hodnotu počtu bajtů, které se nachází v nestránkovaném fondu. To je část paměti, ve které se nachází objekty, které po celou dobu své existence musejí být v paměti a nelze je zapsat na disk. Pokud čítač nabývá větších čísel (poměrově k paměti), může docházet k paměťovým únikům.

Bajty stránkovaného fondu

Tento čítač nám, narozdíl od předchozího, určuje počet bajtů, které se nachází ve stránkovaném fondu. To je část disku určená pro objekty, které mohou být uloženy na disk v případě, že se s nimi nepracuje. Stejně jako u nestránkového fondu, i zde platí, že pokud hodnota nabývá vyšších hodnot, může docházet k paměťovým únikům.

Stránky/s

jak již napovídá název, tento čítač nám udává číselnou hodnotu počtu stránek čtených z disku a zapisovaných na disk. Ty řeší hardwarové chyby stránkování. Pokud tento čítač dosahuje větších hodnot, pravděpodobně dochází k nadměrnému stránkování z důvodu paměťových úniků.

4.2.3 Logické a fyzické disky

Údaje o těchto zařízeních využijeme především u diskových polí, ale i u standardních stanic. Fyzický disk je fyzické zařízení umístěné v počítačové skříni, popř. v diskovém poli. Naproti tomu logický disk je pouze logické uspořádání fyzických disků v systému. Logický disk může být spojen z několika fyzických disků, či na jednom fyzickém disku může být několik logických disků. Z tohoto důvodu sbírám informace o obou typech. Údaje, které zde jsou sledovány jsou volné místo a zápis/čtení z/na disk.

Volné místo

Význam volného místa na disku asi není potřeba vysvětlovat. V systému existují dva čítače pro tuto problematiku. Jeden z nich zobrazuje volné místo v Megabajtech, zatímco druhý zobrazuje tuto hodnotu v procentech. Na první pohled by se mohlo zdát, že stačí využít pouze jeden z nich. Při zamyšlení se nad touto problematikou však vyvstává najevo, že disky se vyrábí v různých kapacitách, a tak pouze jedna z těchto informací nedává relevantní výsledek.

Čtení a zápis

Čtení z disku probíhá v případě, že chceme zobrazit uložené informace. K této činnosti bude často docházet např. u e-shopů, kde si budou zákazníci chtít zobrazit informace o produktu. V případě zahlcení by tak mohlo docházet k nedostupnosti dat. Pro získání informací zde

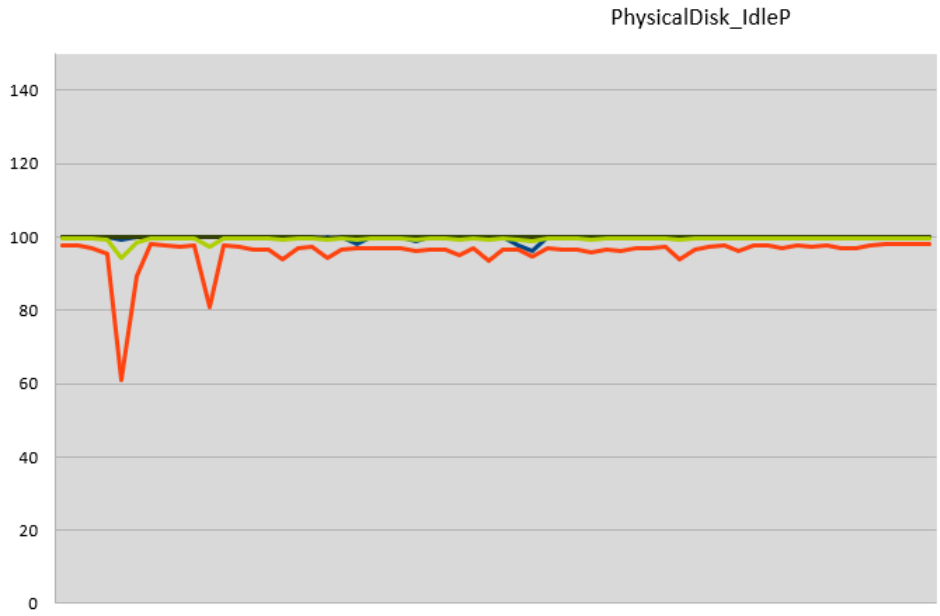
využívám čítač `Bajty čtení s disku/s`, který vrací počet bajtů, které byly přeneseny z disku za sekundu.

Zapisování na disk využijeme naopak v případě sběru a ukládání informací, např. při výpočtech a simulacích. Při přetížení by mohlo dojít k zaplnění bufferu, a teoreticky i ztrátě dat, proto je nutné mít o zápisu přehled. K informacím o zápisu lze využít čítač `Bajty zapisování na disk/s`, které vrací (podobně jako v předchozím případě) počet bajtů, které byly na disk zapsány v daném sekundě.

Počítače, disková pole či datová centra nemají rozdělené disky na čtení a zápis, a tak je dobré mít přehled jak o čtení, tak o zápisu na daný disk. Pro tuto funkcionalitu využívám čítač `Bajty disku/s`, který vrací počet bajtů, které byly za danou sekundu, vystaveny, či zapsány na disk.

Čas disku

Vytížení disku nelze určit pouze z počtu zapsaných a přečtených bajtů, ale je nutné sledovat i procentuální využití disku v čase. K tomuto nám poslouží čítač `%doba nečinnosti`. Tento čítače nám určují jaké procento času disk nevykonával žádnou operaci čtení a zápisu. Specifičtější výsledek bychom mohli získat z čítače `% čtení/% zápis čtení disku`, které však mohou vracet hodnoty větší než 100%, např. z důvodu využívání RAID polí. Výstup tohoto čítače lze pozorovat na obrázku 4.3.



Obrázek 4.3: %času nečinnosti disku

Délka fronty disku

Dalším důležitou informací je pro správce délka fronty disku. Pro její přesnější určení využívám čítač `střední délka fronty disku`, který vrací průměrný počet požadavků na disk, které byly umístěny do fronty.

4.2.4 Síťová rozhraní

Pro komunikaci s okolními zařízeními jsou nutné síťové rozhraní(pomineme-li rozhraní jako FireWire apod.). Správci se tak hodí informace např. o využití šířky pásma u dané síťové karty atd.

Příjem a odesílání

Při sledování informací o síťové kartě uvažujeme podobně jako u sledování informací o discích. Je pro nás důležité sledovat nejen příchozí a odchozí počet bajtů, ale i celkový přenos, neboli součet příchozích a odchozích bajtů. Čítače které v této části používám jsou `Bajty přiajté/s`, `Bajty odeslané/s` a `Bajty celkem/s`. Pokud se velikost celkově přenesených bajtů blíží maximální propustnosti karty, správce by měl uvažovat nad kartou další, či přerozdělením zátěže.

Délka výstupní fronty

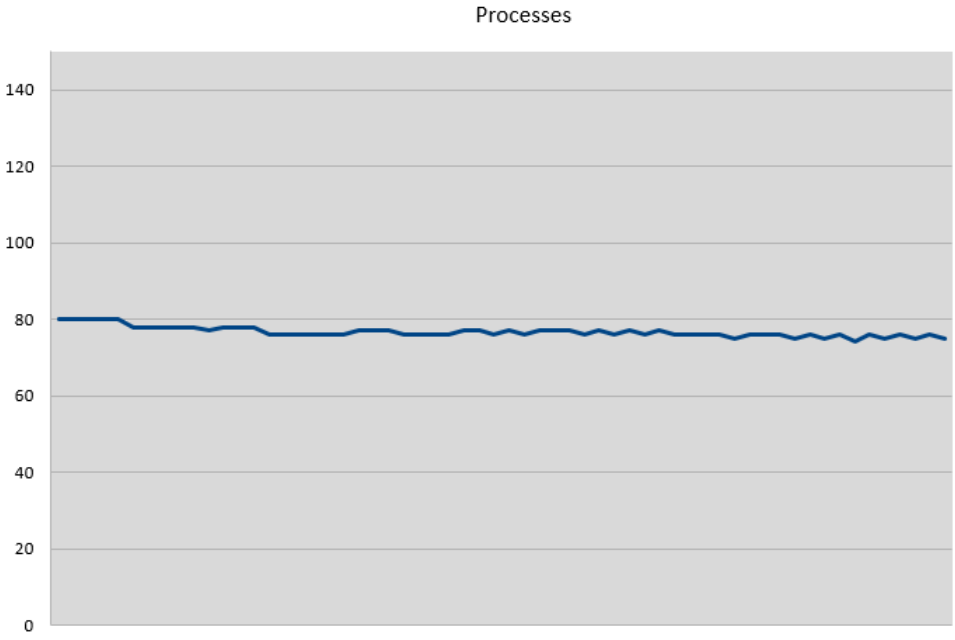
Tento čítač `délka fronty výstupu` nám udává délku fronty čekajících paketů na odeslání. Pokud tato hodnota nabývá čísla větších hodnot, je nutné najít problém a odstranit ho, aby nedocházelo k zahlcení a následné ztrátě paketů.

Počet navázaných spojení

Další, poměrně důležitou informací při zjišťování problémů může být počet navázaných spojení. Pokud se bude tento počet lišit od standardního počtu, může jít o útok na dostupnost služby, či problém v některé aplikaci. Sleduji zde tedy počet navázaných IPv4 a IPv6 TCP spojení.

4.2.5 Procesy

Poslední, ale neméně důležitou součástí této sady sledovacích čítačů je počet běžících procesů. Podobně jako u počtu navázaných spojení v případě síťových rozhraní, i zde může neobvyklý počet procesů znamenat nějakou chybu. Sledování prostředků bohužel nemá pro počet procesů čítač, a tak bylo nutné vytvořil čítač vlastní za pomocí powershell příkazů. Nejdříve ze všeho je zapsán nadpisový řádek výstupního souboru s textem `Datum,Počet`. Poté probíhá cyklus, s předem definovanou délkou trvání, ve kterém jsou získávána data. Jako první je získáno datum ve formátu `MM/dd/yyy HH:mm:ss` a uloženo ho do proměnné. Dále je získán počet procesů za pomocí dále specifikovaných příkazů. První z nich, kterým je `get-process`, vypíše všechny aktivní procesy. tento výstup je následně projit sérií příkazů `sort-object`, `get-unique` a `measure`, které vrátí statistiky, ze kterých je, za pomocí příkazu `Select -expand count`, vyberán pouze počet. Následně je získaný čas a počet zapísán na konec připraveného souboru. Poté už je jen využit příkaz `start-sleep -second`, za kterým ještě doplním číslo. Tento příkaz počká stanovenou dobu, aby byly vzorky brány stejně často, jako u čítačů. Výsledek lze vidět na obrázku 4.4.



Obrázek 4.4: Počet procesů

4.3 Reprezentace dat

Pro výstup je využit textový souboru `CSV`. Ten však není ideální pro rychlou a přehlednou reprezentaci dat. Proto je s ním dále pracováno a je upraven. Nejdříve ze všeho je za pomocí textového nástroje aplikace LibreOffice `soffice` převeden na formát `ODS`, který již není textový, a dovoluje tak ukládat např. grafy, vzorce apod. Následně je vytvořen graf za použití makra vytvořeného v LibreOffice, za použití jazyka Basic. Nejdříve ze všeho zjistíme počet využitých řádků a sloupců v souboru. Pokud je řádků více než devět set, což odpovídá patnácti minutám při logování každou sekundu, je vypočítána pro každý sloupec průměrná hodnota, a dále je brána v potaz pouze ona. Toto opatření je zavedeno z důvodu úspory místa na disku, a pro lepší práci s těmito soubory. Při větším počtu řádků, z kterých je následně tvořen graf, by mohla být špatná odezva programu pro prohlížení a editaci. Následně zjišťuji nejvyšší číslo, které se v souboru objeví. Tato funkcionalita je implementována z důvodu procentuálních výstupů některých čítačů. Automaticky vytvářený graf adaptuje rozsah y osy v závislosti na nejvyšším čísle, a porovnání výsledků několika výstupů by tak nebylo lehké určit. Jelikož některé čítače mohou překročit hodnotu sta procent, je za hraniční hodnotu považováno číslo sto padesát. Poté je vytvořen graf z hodnot v souboru a soubor uložen.

Kapitola 5

Nástroje pro běžnou správu

V minulých kapitolách jsme se zabývali získáváním informací o počítačích. Častokrát však chceme spravovat zařízení co se týče programů, aktualizací a správy souborů. V této kapitole bude řešena právě tuta zmíněná problematika. U playbooků vytvářených v této kapitole se pravděpodobně budou daleko častěji měnit parametry potřebné ke správnému fungování, a tak je zde přidána možnost psát tyto parametry do příkazové řádky při spouštění nástroje za pomocí skriptu.

5.1 Práce se soubory

Pro běžnou správou složek a souborů byl vytvořen playbook, jehož jediným modulem je modul `win_file`. Uživatel tak příkazem zadá, co chce s danou složkou či souborem udělat. Příkladem spuštení může být např: `win file 'path="C:\\složka" state=directory'`. Ansible bohužel neumí jedním příkazem stahovat i nahrávat soubory, a tak bylo nutné tyto dvě činnosti rozdělil do dvou playbooků. Oba z těchto příkazů jsou již využívány v playboocích v minulých kapitolách, ale až nyní nás budou zajímat jejich podrobnosti. Volitelné parametry, které budou vysvětleny později, mají výchozí hodnotu, a tak, pokud nejsou nastaveny uživatelem, budou využity tyto hodnoty.

Nahrávání

Pro nahrávání informací nám tvůrci Ansiblu vytvořili příkaz `win_copy`, který je odvozen od příkazu pro linux, jímž je příkaz `copy`. Dvěma nejhlavnějšímy parametry jsou `src` a `dest`. Ty nám, jak lze předpokládat z jejich názvu, definují zdrojovou a cílovou adresu a jsou povinné. Mimo ně jsou využity ještě volitelné pomocné parametry `force`, `local_follow` a `remote_src`.

Zadání takovéhoto příkazu může vypadat např následujícím způsobem: `win upload 'src="/home/jirka/soubor"\ dst="D:\\ans/"'`. Uvozovky jsou nutné v případě, že název cesty obsahuje mezeru. Co se týče lomítek a zpětných lomítek, Ansible doporučuje pro windows stanice používat zpětná lomítka, jak lze vidět v příkladu(nutné použít escape sekvenci při zadávání). Pokud je však cílem složka, je nutné použít obyčejné lomítko.

Zdrojem pro kopírování může být složka, nebo i soubor. Cesta může být zadána jak relativním, tak absolutním umístěním. V případě, kdy je zdrojem složka, je tato složka zkopírována rekurzivně (včetně podsložek a jejich obsahu). Jsou zde však dvě možnosti kopírování složky. Pokud cesta končí názvem složky, do cílové složky se zkopíruje tato složka včetně obsahu. Pokud však cesta končí lomítkem za názvem složky, budou do cíle

zkopírovány pouze soubory z této složky. Třetí a poslední možností je výběr pouze jednoho souboru.

Destinací může být taktéž složka, či soubor. Pokud je zdrojem složka, cíl musí být také složka. Pokud je zdrojem soubor, a cílem složka, vytvoří se kopírovaný soubor v této složce s původním názvem. Pokud je ve stejném případě destinací soubor, kopírovaný soubor se nakopíruje a přejmenuje. Jestliže je cílem soubor, a složka v které má být umístěn neexistuje, příkaz se nezdaří. V ostatních případech budou vytvořeny soubory i složky.

Význam volitelných parametrů, stejně jakou zde popsaných povinných lze nalézt v dokumentaci [1].

Stahování

Co se stahování týče, zde je využito modulu `fetch`, který je společný pro stanice s operačním systémem Windows i Linux. Stejně jako u nahrávání, i zde jsou použity dva povinné parametry `src` a `dest` a k tomu pomocné volitelné parametry `flat`, `fail_on_missing` a `validate_checksum`.

Příkladem může být `win download 'src="D:\\ans\\soubor" dst="/home/jirka/"'`. Platí zde vše, co již bylo napsáno u příkazu pro nahrávání, tedy použití uvozovek a zpětných lomítek v escape sekvenci.

Zdrojem zde musí být jen a pouze jeden soubor, což může být velká nevýhoda oproti nahrávání dat. Cíl může být složka i soubor. Tento příkaz umí vytvořit složky i soubory, jejichž otcovský adresář neexistoval.

5.2 Správa aplikací

Správce může také chtít mít přehled o tom, jaké aplikace jsou na stanicích nainstalované. Např. z důvodu firemní politiky, či bezbečnostních důvodů. Nežádané aplikace může chtít odinstalovat. Naproti tomu se mu může hodit hromadná instalace při instalaci nových počítačů, či přechod na jinou verzi aktuálního softwaru. pro všechny tyto případy jsem tedy vytvořil další sadu playbooků.

Instalace aplikací

Nyní se dostáváme k části, která byla asi nejtěžší částí. Při zjišťování, jak automaticky instalovat programy, byly vyzkoumány na dvě možnosti. Jednou z nich je použití Ansible modulu `win_package`, který je zde využit. Druhou možností bylo využití vestavěných příkazů systému Windows. Pro co nelepší výsledek byly vyzkoušeny obě, a vybrána z nich ta, která byla pro tuto problematiku ideální. Bohužel obě volby s sebou nesly jak pozitivní, tak negativní důsledky, a bylo tak velmi těžké se rozhodnout, kterou z těchto možností nakonec implementovat.

Nejdříve bylo experimentováno s možností, která se osvědčila při odinstalaci. Ta se na první pohled zdála být velmi vhodnou. Kód by byl vlastně stejný jako u odinstalace pouze s tím rozdílem, že místo odinstalačního souboru by se spouštěl instalační. Nemusel by tedy vytvářet další playbook, uživatel by pouze zadával cestu ke spustitelnému programu. Taky by byla zachována možnost sepsat seznam instalačních programů, které by se poté spustili.

Druhá možnost, tedy využití modulu poskytovaného nástrojem Ansible, zde dává větší smysl. Jsou zde možnosti volby, kdy se má daná aplikace nainstalovat pouze v případě, že je na cílovém systému přítomna složka, služba či konkrétní verze nějaké služby/pro-

gramu. Také se můžeme dotazovat na očekávaný návratový kód, který nám může říci, že je např. potřeba restartovat systém. Bohužel největším mínusem této možnostmi je právě Ansible, který při registrování parametrů přidává na začátek a konec uvozovky, a powershell poté tento parametr nenalezne. Zde by bylo řešením nebrat žádné parametry z příkazové řádky, ale všechny je mít napsané v playbooku. Cílem však bylo vytvořit lépe modifikovatelné parametry, a tak bylo rozhodnuto použít pouze některé parametry. Zároveň s tímto nelze sepsat seznam aplikací a umístění jejich instalačních balíčků, které mají být nainstalovány. To však může správce vyřešit vytvořením skriptu, který bude opakovaně volat tento playbook s potřebnými parametry. Parametry které jsou využívány jsou `creates_path`, `creates_service` a `creates_version`. Dále je využíván parametr `path`, specifikující umístění instalačního souboru a `arguments`, do kterého se vepisují parametry instalátoru. Příkladem pro instalaci aplikace 7-Zip tak může být příkaz: `win install 'path="C:\\7Z.exe" creates_path="C:\\program Files\\7-Zip" arguments="/s"'`.

Odinstalace aplikací

Při odinstalaci jsem mohl znovu využít `win_package`, který mi však přišel náročnější na implementaci a přitom při experimentování nepřinášel žádné výhody oproti druhému způsobu. Využito tedy bylo vestavěných příkazů systému Windows. Je nutné spouštět tichou(silent) odinstalaci, jinak dojde k zaseknutí nástroje. Tichá instalace se nejčastěji spouští argumentem `/S` nebo `/SILENT`.

Při spouštění nástroje má uživatel na výběr, zda chce odinstalovat jednu, či více aplikací výběrem ze souboru, nebo pouze jednu aplikací vepsáním do příkazu. Argument který toto rozhoduje se nazývá `file`, a jeho defaultní hodnota je `"no"`. Pokud tedy není změněn, uživatel zadává pouze parametr, který specifikuje cestu k odinstalačnímu souboru: `win uninstal 'src="\"C:\\Program Files\\7-Zip\\Uninstall.exe\" /S"'`. Pokud se rozhodne pro druhou možnost, příkaz bude vypadat následujícím příkladem: `win uninstal 'file="/home/jirka/soubor s cestami"'`. Po odinstalaci některých aplikací je zapotřebí ještě vymazat některé zbylé soubory. V opačném případě by příště nemusela fungovat tichá instalace. V případě prohlížeče Mozilla Firefox se jedná o skrytou složku s umístěním `C:\Users\user\AppData\Roaming\Mozilla`, v případě aplikace PSPad zase zůstane složka `C:\Program Files (x86)\PSPad editor` a v ní v podsložce `Uninst` soubor `unins000.exe`.

Seznam nainstalovaných aplikací

Někdy můžeme chtít získat seznam nainstalovaných aplikací, jindy nás zase může zajímat pouze to, zda máme určitou aplikaci nainstalovanou, nebo jaké aplikace máme nainstalované od daného výrobce. pro všechny tyto případy slouží playbook `PlayAppList`, který spustíme zapomocí skriptu a parametru `win app`. Parametry, které jsou využívány, jsou `where` pro určení, zda chceme seznam celý nebo pouze vybrané aplikace. Dále pak parametr `file` pro určení, zda chceme výstup uložit do souboru nebo vypsat do konzole. V tomto playbooku definuji čtyři volby výpisu.

Prvním z nich je možnost výpisu listu na základě nějaké podmínky a výstup ve formátu `csv`. Příkladem takovéhoto spuštění je například následující text: `win app 'where="yes" condition="\$\_.DisplayName -like "*DOOM*"" dst="/home/jirka/soubor"'`. Pravidla pro filtrování zde definuje powershell modul `Where-Object`, který hledá dané pravidlo v daném záznamu. Záznam se definuje řetězcem `$_.`, následovaným jménem sloupce. Další možností je použití pouze `where` argumentu bez file argument. Výstupem je tedy výpis dle zvolených parametrů do konzole. Třetí možností je výběr pouze argumentu `file`, který

vybere všechny aplikace a uloží je do zvoleného souboru. Poslední možností je nezádání parametru `where` ani `file`. Tuto možnost nedoporučuji, neboť výstupem bude velmi dlouhý seznam aplikací v konzoli.

U všech výše zmíněných možností jsou vybírány z původního listu pouze některé záznamy, kterými jsou `DisplayName`, což je název produktu, `InstallLocation`, který specifikuje, kde je aplikace nainstalována, `DisplayVersion`, neboli verzi aplikace a `Publisher`, kde se zobrazí vydavatele produktu. Pro možnost odinstalace je zjišťován ještě záznam `UninstallString` a `QuietUninstallString`.

V případě kdy chce uživatel získat soubor s výstupem, je tento výstup ukládán ve formátu `csv` na cílové stanici, poté stáhnut a na vzdálené stanici vymazán.

Nejdříve bylo využito nástroje `WMIC` a výpis nainstalované aplikace z modulu `product`. Tento modul však zobrazuje pouze aplikace, které byly nainstalovány za pomocí instalačního formátu Windows. Nyní tedy tento nástroj funguje díky získávání dat z registrů. Původní záměr byl výstup uložit do jedné proměnné. Ansible však tzv. registruje výstup i v případě že nebyla splněna podmínka pro vykonání příkazu, a tak musím mít čtyři proměnné, z nichž je poté vybrána ta, která se má použít pro výstup.

DisplayName	**DisplayVersion**	**Publisher**
Visual Studio Enterprise 2017	15.5.27130.2036	Microsoft Corporation
Nexus Mod Manager	0.63.14	Black Tree Gaming
BlackArmor Discovery	1.20.0931.004	Seagate

Tabulka 5.1: Část výpisu nainstalovaných programů

Instalace aktualizací

Aktualizace systému Windows jsou poslední roky poměrně častou záležitostí. Mnohdy mohou ochraňovat počítač před nejnovějšími viry, či přidávat do systému nové či vylepšené funkce. I na ně mysleli programátoři Ansiblu a vytvořili modul pro jejich stahování. Jedná se o modul s názvem `win-updates`. Ten má parametry `category_names`, což jsou názvy kategorií updatů. Ty musí být vepsány přímo do skiptu. Následuje parametr `log_path`, který specifikuje umístění souboru, v případě že ho chceme, s výstupem na cílové stanici (musí existovat nadsložka). jako poslední je zde parametr `state`, který specifikuje, zda chceme pouze vypsat dostupné aktualizace, nebo dostupné aktualizace nainstalovat.

Playbook byl přidán do skriptu, a jeho spuštení může být např. následující: `win update 'log_path="\yes" state="searched" src="C:\\anstmpfile.txt"'`. `log_path` má defaultní hodnotu nastavenou na `"no"`. `src` zde specifikuje cestu pro dočasné uložení souboru na cílové stanici. Tento výstup je poté nakopírován do složky s výstupy.

Kapitola 6

Automatizace pomocí DSC

V předchozích kapitolách jsme se zabývali používáním Linuxového nástroje Ansible na správu stanic Windows. Jaké jsou však jiné možnosti, které může správce využít v případě že nechce používat Ansible, popřípadě ani linuxovou distribuci? Zde se přímo nabízí nástroj, který má stejné zaměření jako Ansible, ale vytvořil ho Microsoft a je implementován v nástroji Winodws PowerShell od verze 4.0. Tímto nástrojem je DSC - Desired State Configuration.

Jedná se o deklarativí prostředí pro konfiguraci a správu systémů, která se skládá ze tří hlavních komponent [6]. Těmi jsou `Configuration - konfigurace`, `Resources - zdroje` a `LCM - Local Configuratiom Maneger`.

Konfigurace

Konfigurační částí se myslí PowerShell skripty, které využívají zdroje k tomu, aby uvedli cíl do požadovaného stavu. Tyto skripty vypadají podobně jako zdrojové kódy. Celý blok skriptu je obalen blokem `Condiguration Name`, který definuje název konfigurace. Následují bloky `Node`, které určují, pro které stanice je daný skript určen. Poslední částí jsou bloky využívající zdroje. V celém těle konfigurace lze využívat PowerShell příkazy stejně jako normálně.

Zdroje

Zdroji jsou zde myšleny prostředky, s pomocí kterých lze přistupovat k požadovaným objektům a pracovat s nimi. DSC má již některé zdroje implementované (soubor, uživatel, registr...). Pokud bychom potřebovali nějaké jiné, je možné využít tzv. `DSC Resource Kit`, což je sada experimentálních zdrojů, vytvořená PowerShell týmem. Pokud by nám ani tyto nestačili, je zde močnost si vytvořit své.

LCM

LCM je služba, která zajišťuje, aby byl skript na dané stanici spuštěn a správně vykonán. Je také zodpovědný např. za interval přijímaná konfigurací, či určování částečných konfigurací. Pro nastavení LCM slouží speciální konfigurace.

Architektura distribuce konfigurací

Budeme-li se zabývat možnými způsoby doručení konfigurací na cílové stanice, existují v zásadě dvě možnosti, které můžeme využít. Jednou z nich je mód `PUSH - Odeslání`, tím druhým `PULL - Vyžádání`.

Push mód spočívá v tom, že konfigurace jsou manuálně odesílány na cílové stanice. Výhoda spočívá v ukládání všech konfigurací na správcově počítači, nebo také to, že se pro to nemusí nastavovat (popř. kupovat) nový server. Nevýhodou naproti tomu je, že nevíme, zda jsou požadované počítače připojené, a nemají změněnou IP adresu (např. notebooky).

Naproti tomu PULL mód je zcela automatizovaný. Cílová stanice se v pravidelných intervalech dotazuje serveru, zda pro ni má nějakou konfiguraci. Jako výhodu můžeme uvést (ne)přítomnost notebooků. Jakmile se totiž dostanou do sítě, automaticky si vyžádají novou konfiguraci. jako nevýhoda je tu však nutnost nastavení nového serveru pro tuto činnost. Distribuční server může konfigurace rozesílat za pomocí protkolu `SMB - Server Message Block`, též zvaný jako `CIFS - Common Internet File System`, nebo za pomocí `HTTP/S - HyperText Transfer Protocol`.

Srovnání s Ansible

U Ansible byl postup takový, že se nainstalovaly prerekvizity a Ansible, a poté se vytvořili playbooky, které se spouštěli. U desired state configuration máme některé kroky stejné, některé řešit nemusíme a jiné jsou naopak přidané. Instalovat se musí pouze na starších operačních systémech(Windows 8 a níže, Windows Server 2012 a níže). Co se týče samotných skriptů, zde nejdřív vytváříme konfigurační soubor s příponou `ps1`, což je PowerShell skript. Ten zkompilujeme, a poté spustíme.

Pro nasazení ve velkých společnostech je lepší použít DSC, z důvodů přítomnosti v operačním systému a lepší propracovanosti nasazení. Pro skromnější používání je však Ansible plně dostačující nástroj, který sice má své omezení a neduhy (řešení escape sekvencí atd), ale lze s ním nastavit vše, co potřebujeme. A co neumí přes své moduly, toho lze docílit za použití PowerShell příkazů, i když to vyžaduje na cílové stanici PowerShell verze 5.0.

Kapitola 7

Prostředí pro vývoj a testování

Linux

Pro vývoj a provozování nástroje byla použita 64 bitovou distribuci Linuxu. Konkrétně šlo o Ubuntu verze 16.04 LTS. Ta byla nainstalována jako virtuální stroj ve virtualizačním prostředí Oracle VM VirtualBox. Pro plynulý běh systému, bylo virtuálnímu počítači přiděleno 1,5 GB RAM a 2 jádra procesoru bez omezení výkonu. Jako hostitelský operační systém byl použit Windows 10. Ansible bylo využíváno ve verzi 2.4. Pro jeho bezproblémový běh byl nainstalován python verze 2.7.

Windows

Pro testování nástroje bylo využito dvou počítačů. Jedním z nich byl hostitelský počítač s Windows 10. Jako druhý stroj posloužil další virtuální počítač, na kterém běžel Windows server 2012. Tento virtuální počítač měl přiděleno 3 GB RAM a 2 jádra procesoru bez omezení výkonu. Bylo však nutné vypnout Windows Update, neboť tím byl velmi zatěžován disk a docházelo tak k prodlevě při komunikaci s počítačem. Zde bylo již ze začátku práce konzultováno s vedoucím této práce, že mohou nastat některá úskalí z důvodu použití virtuálního stroje. V některých případech má své prostředky virtualizované (např. CD-ROM), a tak je ve výpisu něco navíc oproti hostitelskému stoji, to však nevadí. Horší situace nastává v případě, že virtuální stroj nemá přístup některým součástem fyzického počítače, a u některých výpisů tak můžou vznikat problémy. To se záhy potvrdilo při pokusu o výpis některých součástí počítače, kdy nástroj vrátil chybu, že tato instance není spuštěna. Nejprve jsem to řešil nastavením ignorování této chyby, aby mohl proběhnout zbytek nástroje. Postupem času a přemýšlením nad potřebnými a nepotřebnými informacemi však tento problém vymizel, neboť se jednalo o informace, které nebyli důležité.

Kapitola 8

Závěr

Se stále se zvyšujícím počtem zařízení je nemožné spravovat je jednotlivě, a právě, ale nejen, pro tyto případy jsou velmi užitečná a hodnotná řešení pro automatizaci správy. Na trhu je spousta méně, či více povedených produktů, a každému se hodí nějaké.

Cílem této práce bylo vytvořit sadu nástrojů za pomoci nástroje Ansible, které pomohou správci řešit situace s hromadnou správou zařízení. Před vytvářením samotného nástroje jsem se seznámil s již zmíněným nástrojem Ansible, který je velice zdařilý a dobře se používá. Dále jsem se seznámil s Windows nástroji PowerShell a WMIC, a jejich možnostmi. Také jsem se seznámil s problematikou evidence komponent počítače a jejich důležitými vlastnostmi.

Výsledkem práce je sada nástrojů, která se spouští za pomocí bash skriptu. Ta umožňuje získávat informace o hardwaru a jeho využití, pracovat se soubory a správu programů. Většina výstupů je podána textově, popř. s grafem u výstupů, u nichž to dává smysl, a podávají přehlednou zprávu o stanicích.

Co se týče rozšiřitelnosti práce, nabízí se zde spousta možností, jako například úprava registrů, konfigurace cílové stanice dle vzoru a podobně. V průběhu vývoje byla také vydána nová verzi Ansible, která u některých modulů značně rozšiřuje možnosti a význam jejich použití.

Literatura

[1] Ansible: *Win_copy - Copies files to remote locations on windows hosts.* [Online; navštíveno 4.4.2018].
URL http://docs.ansible.com/ansible/latest/modules/win_copy_module.html

[2] Inc, R. H.: *Ansible Documentation.* [Online; navštíveno 10.10.2017].
URL http://docs.ansible.com/ansible/latest/index.html

[3] Inc, R. H.: *Authentication Options.* [Online; navštíveno 12.10.2017].
URL https://docs.ansible.com/ansible/2.5/user_guide/windows_winrm.html#authentication-options

[4] Microsoft: *Use PerfMon to Diagnose Common Server Performance Problemsr.* [Online; navštíveno 23.3.2018].
URL https://technet.microsoft.com/en-us/library/2008.08.pulse.aspx

[5] Microsoft: *Win32 Provider.* [Online; navštíveno 25.10.2017].
URL https://msdn.microsoft.com/en-us/library/aa394388(v=vs.85).aspx

[6] Microsoft: *Windows PowerShell Desired State Configuration Overview.* [Online; navštíveno 17.4.2018].
URL https://docs.microsoft.com/en-us/powershell/dsc/overview

[7] Microsoft: *WMI Architecture.* [Online; navštíveno 4.5.2018].
URL https://technet.microsoft.com/en-us/library/cc180678.aspx

Příloha A

Obsah CD

- source/
  - Zdrojové soubory nástroje
- docsource/
  - Zdrojové soubory technické zprávy
- xklicj00-Automatizace-Windows-Ansible.pdf
  - Technická zpráva
- README.txt
  - Návod pro zprovoznění nástroje