МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра комп'ютерних інформаційних технологій

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
_____Савченко А.С.
"____"_____2020 р.

ДИПЛОМНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
"МАГІСТРА"

ЗА СПЕЦІАЛІЗАЦІЄЮ "ІНФОРМАЦІЙНІ УПРАВЛЯЮЧІ СИСТЕМИ ТА ТЕХНОЛОГІЇ (ЗА ГАЛУЗЯМИ)"

Тема: "Методи створення та імплементації скриптів в корпоративному середовищі"

Виконавець: Літовальцев Тарас

Керівник: к.т.н., доцент Моденов Юрій Борисович

Нормоконтролер: _____ Шевченко О.П.

Київ 2020

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки, комп'ютерної та програмної інженерії

Кафедра Комп'ютерних інформаційних технологій

Галузь знань, спеціальність, спеціалізація: 12 "Інформаційні технології", 122 "Комп'ютерні науки", "Інформаційні управляючі системи та технології (за галузями)"

ЗАТВЕРДЖУЮ
Завідувач кафедри
Савченко А.С.
" " 2019 р.

З А В Д А Н Н Я

на виконання дипломної роботи студента

Літовальцева Тараса Олександровича

1. Тема роботи: " Методи створення та імплементаці скриптів в корпоративному середовищі"

Затверджена наказом ректора від " " за №

2. Термін виконання роботи: з 2019р. до лютого 2020р.

3.Вихідні данні до роботи:Огляд середовищ скриптів, основних команд ,якими користується системний адміністратор в повсякденних задачах. Огляд Операційних систем та можливостей скриптів виконаних них. Огляд основних інструментів та шаблонів для автоматизації бізнес процесів. Розглядаються технології віртуалізації та методи створення IT інфраструктури.

4. Зміст пояснювальної записки: 1)Було розглянуто стандартні інструменти та середовища виконання скриптів. 2)Огляд семантики скриптових мов та їх практичність при виконанні задач. 3) Запропоновано рішення в процессі побудови інфраструктури . 4) Визначено основні концепції побудови IT

інфраструктури . 5) Написання та впровадження скриптів на сервері для моніторингу та віддаленного керування .

5. Перелік обов'язкового ілюстративного матеріалу: таблиця, рисунки, діаграми, графіки, а також слайди презентації доповіді у PowerPoint .

6. Календарний план-графік

№ з/п	***Завдання***	***Термін виконання***	***Підпис керівника***
1.	Огляд основних команд Windows	31.10.18 – 09.11.18	
2	Створення командних файлів	10.11.18 – 20.11.18	
3	Порівняняння скриптових мов їх та можливостей	21.11.18 – 30.11.18	
4	Дослідження моделі IT інфраструктури	01.12.18 – 10.12.18	
5	Побудова письмового плану проектування IT інфраструктури	11.12.18 – 20.12.18	
6	Опис призначення ITSM	21.12.18 – 31.12.18	
7	Розглянути скрипти в доменній структурі компанії	01.01.19 – 10.01.19	
8	Огляд поняття ‘’Інфраструктура-як сервіс’’	11.01.19 – 20.01.19	
9	Створення доповіді та слайдів до неї	21.01.19 – 30.01.19	
10.	Оформлення та друк пояснювальної записки дипломної роботи	31.01.19 – 04.02.19	

7. Консультація з окремого(мих) розділу(ів) роботи:

Розділ	Консультант (посада, П.І.Б.)	Дата, підпис	
		Завдання видав	Завдання прийняв

8. Дата видачі завдання _____

Керівник дипломної роботи . _____ Моденов Ю.Б. .

Завдання прийняв до виконання _____

(підпис випускника) (ПІБ)

РЕФЕРАТ

Пояснювальна записка до дипломної роботи "Методи створення та імплементації сериптів в корпоративному середовищі": 121 с., 36 рис., 1 табл., 10 літературних джерел.

Об'єкт дослідження: Корпоративне середовище як частину IT інфраструктрури і роль скриптів в ній.

Мета роботи: дослідити методи створення та імплементації скриптів в корпоративному середовищі.Дослідити семантику та практичність інструментів запуску скриптів а також їх пряму залежність з бібліотеками.

Методи дослідження: розробка концепції технології створення доменного середовища компанії з елементом віддаленного управління та запуску скриптів для застосування налаштувань на усіх машинах які присутні в середовищі. Досліджено методи створення користувачів в напівавтоматичному режимі.

Результати магістерської роботи: Було створенно концепцію написання скриптів та їх запуску як частини процесу управління інфраструктурними сервісами які вже є вмонтованими в систему. Скрипти забезпечують автоматизацію процесів та послідовність виконнання сервісів які звертаються до ядра системи взаємодіючи з програмною оболонкою виконання скриптів.

МЕТОДИ СТВОРЕННЯ ТА ІМПЛЕМЕНТАЦІЇ СКРИПТІВ, КОМАНДИ, КОМАНДНІ ПРОГРАМИ , POWER SHELL , IT INFRACTURE , ITSM, PYTHON, ANSIBLE , OPEN STACK, КОНТРОЛЕР ДОМЕНУ.

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

API -Прикладни́й програмний інтерфейс.

XML - Розши́рювана мо́ва розмі́тки (англ. *Extensible Markup Language*)

JSON (англ. *JavaScript Object Notation*) — текстовий формат обміну даними

SDK - (від англ. software Development Kit)-набір утиліт та інструментів для розробки

Active Directory -контроллер домену

CMD- коммандний рядок

Power Shell(PS)- оболонка виконання скриптів

ITSM -система управління IT інфраструктури

OPEN STACK -хмарне рішення для впровадження інфраструктури.

PS -POWER SHELL -командна оболонка

VM – віртуальна машина

. . . . . .

ЗМІСТ

ВСТУП

У сучасному програмуванні в мережі,скрипти (сценарії) - це окремі послідовності дій, створені для автоматичного виконання завдання. Якщо готового сценарію немає, користувач виконує ці дії вручну з відповідними витратами часу і можливостями появи помилок. Для написання скриптів використовуються спеціальні мови програмування, які так і називаються - скриптові. Відповідно, скриптова мова програмування - це набір лексичних, семантичних і синтаксичних правил для створення і редагування скриптів. Навіщо потрібні скрипти коли всі операції в системі можна робити вручну, послідовно і при цьому існує менша вірогідність помилки в процесі виконання операції.Зазвичай у такому випадку у адміністратора є можливість вимкнути процес примусово, хоча це грозить тим, що в подальшому система як програмний продукт може працювати не коректно. В загальному будь який процес для інженера відбувається через графічний інтерфейс для наглядності та зручності, але є один момент – витрата апаратних ресурсів при запуску графічного ядра, що веде за собою повільність процесів які запущені фоново.Призначення скриптів наступне:наприклад ви можете встановити програмне забезпечення на цільову машину написавши декілька рядків коду, прицьому описавши умови перевірки успішності виконання операції. Для таких випадків існують вже готові системи конфігурації але скрипти та шаблони себе не напишуть самі тому для кваліфікованого адміністратора важливим кроком для осилення цієї предметної області є вивчення семантики мови, порівняння кількості рядків коду з іншими мовами та методи їх впровадження на машину яка являється мастером або конфігуратором для інших цільових машин-іншими словами головний сервер управління, максимально ізольований та малонавантажений. На сервері повинні існувати тільки ті бібліотеки та програмні продукти, котрі не навантажують його та дозволяють йому виконувати скрипти без можливості бути перерваними. Для цього існують скрипти, вони економлять час адміністратора тим що замість пройти одну операцію сотні раз на сотні різних машинах, все може бути записано в один скрипт, який буде запускатись у визначений час та перевіряти умови відповідності стану машин в загальній інфраструктурі до стандарту визначеного у документації і т.д. Скрипти це магія. В даній роботі я спробую максимально підтвердити доцільність використання скриптів в корпоративному середовищі.

РОЗДІЛ 1.

ЗНАЙОМСТВО ЗІ СКРИПТАМИ

1.1 Скрипти мають наступні цілі:

SEO-скрипти (шаблони) для просування сайтів. Зазвичай під їх управлінням працюють спеціалізовані програми автоматизації цього процесу. Найбільш відомі - ZennoPoster, Human Emulator;

Системи для збору статистики відвідувань (лічильники відвідуваності). Ці скрипти найчастіше створюються із застосуванням JavaScript;

Сценарії для звернення до баз даних. Тут лідирує мову PHP;

Скрипти для роботи гостьових книг і створення коментарів до записів. Найчастіше застосовується комбінація PHP і JavaScript;

Скрипти для динамічного відображення сайтів. В цьому випадку скриптова мова визначається мовою написання CMS;

Скрипти для зміни частини сторінки сайту без її перезавантаження. При реалізації використовуються технології Ajax. У цьому випадку на перший план виходять асинхронний JavaScript і XML. Веб-додатки проводять обмін даних з сервером в «тлі», зміни на сторінках сайту відбуваються без їх повного перезавантаження. Користувачі зазвичай не помічають таких змін, і їм не потрібно розуміти, що таке скриптова мова програмування, щоб відмінно взаємодіяти з сайтом.

Переваги скриптів

Їх застосування дає можливість вносити програмні зміни без побоювання зруйнувати всю систему. Якщо скрипт написаний з помилкою, то при його виконанні вони будуть видані в результаті. При цьому сайт залишиться працездатним.

Використання скриптів дає можливість отримувати проблемно орієнтований набір команд. У цьому випадку один рядок сценарію дозволяє виконувати такий же обсяг дій, як програма з багатьох десятків рядків на

компільовані мови. На цьому прикладі наочно видно, що таке сценарій в програмуванні і наскільки його застосування прискорює вирішення завдань. З використанням скриптів успішно реалізується кроссплатформенность виконання завдань. Чудовим прикладом є JavaScript - одні й ті ж сценарії цією мовою без проблем виконують браузери в різних операційних системах.

Недоліки скриптів

Помітний більший час виконання. У переважній більшості випадків інтерпретовані сценарії вимагають для виконання набагато більше часу і комп'ютерних ресурсів.

До сих пір для таких мов не створена якісна середовище розробки рівня IDE.

В просування і рекламу цих мов вкладаються недостатні кошти. Як це не парадоксально, відносна доступність і умовна безкоштовність сценарних мов призводять до того, що у розробників просто не вистачає коштів на маркетинг і рекламу. Тому для багатьох великих грошових проектів вибираються Java або C #.

Типи скриптів

За ступенем швидкодії вони поділяються на мови динамічного розбору (sh, COMMAND.COM) і вимагають попередньої компіляції, такі як Perl. Також скриптові мови розбиваються на кілька великих груп по застосуванню.

Виділяють:

Командно-сценарні (JCL, sh, bash, csh, ksh, AppleScript, COMMAND.COM і cmd.exe, VBScript);

Прикладні (AutoLISP, JScript, JavaScript, ActionScript, Game Maker Language, VBA і ін.);

Універсальні сценарні (Tcl, Lua, Perl, PHP, Python, REBOL, Ruby, PowerShell).

Приклади скриптових мов

Найбільш відомі: PHP, Perl, Python, AngelScript, JavaScript, JScript і інші. Всі вони є високорівневими. За своїм механізмом дії скриптові мови зазвичай інтерпретуються, а не компілюються.

1.2Теорія правильних скриптів

Чим різниться скрипт і програма? Зовсім не використовуваним мовою або наявністю інтерфейсу.

Головна різниця - в наявності у програми величезною оболонки, не пов'язаної «вмістом» програми. Залежно від платформи, це можуть бути сторінки керівництва, підтримка декількох мов, наявність функціоналу по установці / видалення, виконання угод про інтерфейс (командного рядка, або інших засобів взаємодії), інтерфейси в загальному реєстрі і т.д ... Програма повинна вміти працювати в Будь-який документованої середовищі, передбачати різні ситуації (найкрутіше з цим у програм під unix, які використовують ./configure для визначення, власне, де вони, що можна, а що не можна на цій (черговий) платформі).

Скрипт - в протилежному значенні призначений для вирішення конкретної проблеми «тут і зараз». Ніхто не очікує від скрипта, який відсилає статистику, здатності робити це одночасно на solaris'e, freeBSD і windows embedded standard з cygwin'ом на борту.

За математико-програмістки уявленням, між скриптами адміністрування та програмами немає різниці. Вони працюють за однаковими принципами, взагалі кажучи, виконують майже одне і те ж.

Різниця між скриптом і програмою - адміністративна.

Практично будь-яка програма має в собі ТРИ важливі складові:

1.-Нетривіальний алгоритм.

2.-Техпідтримку, напрацьовані кращі практики використання, типові схеми впровадження і готові конфігурації

3.-Правильну інтеграцію в робоче середовище в будь-який дозволеної (документованої) конфігурації.

Докладніше про ці складові . 1) Алгоритм. У будь-якої програми є по-перше якась ідея (що, власне, робить програма), по-друге - обв'язка. Читання конфігов, висновок в сіслог, оповіщення по пошті і ще тисяча не пов'язаних з основним завданням операцій. Але програму використовують не заради

читання конфігов і записи в лог, а заради того, що вона робить. Відповідно, зазвичай ідея полягає у виконанні якихось дій по якомусь алгоритму. Нетривіальна ідея. У фактичному коді це може бути менше, ніж читання xml-конфіга, але при цьому саме робочий алгоритм - суть програми. Він може бути або «обробляють дані» (на кшталт SQL'я), або математичним (на кшталт md5sum), або працюють з конкретними особливостями конкретної залізяки (формату файлу) - але він завжди вимагає високої кваліфікації в предметної області для адекватного розуміння принципів роботи. Зрозуміло, що код OpenSSL може читати будь-який програміст. Зрозуміло, що алгоритм роботи OpenSSL може зрозуміти тільки хороший математик .

Але ми пишемо не про програмах а про скрипти. Скрипт не повинен реалізовувати нетривіальні алгоритми. Якщо ви у себе в скрипті пишіть аналог base64 - це поганий скрипт. Якщо ви у себе в скрипті пишіть відправку повідомлень по smtp методом «відкрили сокет, записали» - це огидний скрипт. Якщо ви у себе в скрипті ловите дані з когось порту і пишіть туди відповідь (для управління УПСом) - це писар якийсь, а не скрипт.

Скрипт не повинен містити в собі алгоритм в термінах «предметної області». У скрипта немає предметної області, скрипт - обгортка навколо програм, які вже працюють з предметними областями. У деяких випадках скриптова мова може надавати повний інструментарій:

If md5.md5sum (open. ($ check) .read ())! \u003d url.openurl ($ control) .read ():
Smtp.sendmail ($ from, $ to, 'data check failed', 'md5sum of $ check does not match control sum form $ contol.').

Це скрипт. Просто скрипт. Не дивлячись на те, що він реалізує великий обсяг роботи. А ось якщо у вас md5 - клас, оголошений в скрипті 5 рядками вище з імплементацією md5 (або url, або open, або smtp, etc) - це вже спроба походити на програму. Але програма - це набагато складніше, ніж алгоритм.

2). Будь-яка програма повинна володіти відомим поведінкою. Математики пропонують описувати поведінку програми в всеосяжних термінах; Практика же каже, що зазвичай крім алгоритму програма ще містить баги і фичи, які

впливають на її поведінку, до яких треба адаптуватися. Адаптуватися до них куди простіше, коли є деяка практика використання програми.
«KDC has been valid once but invalid now» - якщо це повідомлення від скрипта - все, ховати. Прямо тут, на місці. У програми це цілком розумне повідомлення по якому можна гуглити і з'ясовувати, що саме не так. Це прямий наслідок наявності в програмі якоїсь предметної логіки, специфічною і вимагає від користувачів не вивчати її наскрізь, а прийняти бехівіорістіческі. Тобто як набір тверджень про поведінку програми. «Дана версія програми не розуміє файли більше 2Гб в розмірі». Це не вкладається в алгоритм (а якщо вкладеться - буде займати так з тому дискретної математики) - але це потрібно знати в практичному сенсі. «Дана програма погано себе веде в умовах симетричного навантаження на аплоад / даунлоад, краще запустити дві копії, кожна з яких буде працювати в свою сторону симетрично »- розуміння чому потребують титанічних зусиль, простіше прийняти це як даність. Чим складніше алгоритм, тим більше життя потрібно витратити на його дослідження, адаптацію і глибоке вивчення. На все життя не вистачить, значить, простіше прийняти як даність і сконцентруватися на важливому.
Скрипт повинен бути кришталево зрозумілий кожному, хто його подивиться (з поправками на знання скриптового мови). Ніяких (if every in self .__ datarange__ is not in any map (__ systable __. Lang, __localtable__.map, lambda (a, b): [a in b or b in a for every __sys __. Pair (a, b)])) Raise 'Missed i18n constitution'.

3) Скрипт вирішує завдання “тут і зараз”. Програма вирішує завдання “там і завжди” (з поправкою на досвід експлуатації з п.2). Коли ви пишете скрипт, ви робите так, щоб воно працювало у вашій системі. Воно не годиться для вільного використання в інших системах (хоча може бути ЛЕГКО (див п.1) адаптовано). Програма повинна бути адаптується до купи варіантів застосування, реалізація цієї адаптації в скрипті призводить до втрати його простоти і перетворенню його, власне, в програму. Крім того (на жаль і ах), але знання ЯК ПРАВИЛЬНО писати програму не еквівалентне написання

правильного алгоритму. Ви можете написати приголомшливу бібліотеку, але якщо ви не зможете запустити її на машині, у якій понеділок перший день тижня (або другий - кому як пощастить), то гріш ціна вашій бібліотеці. Необхідність думати про це - це вже написання програм - скрипту таке допустимо (хоча і не бажано).

Ну і ще важлива відмінність між скриптами і програмами. Програми (в формі бібліотек) можуть «нашаровуватися» один на одного. Цією програмою потрібен libYYY, яка використовує libZZZ і libAAA, при цьому libAAA використовує libZZZ і libc. Це нормально.

Скрипти ж НЕ ПОВИННІ ЗАЛЕЖАТИ ОДИН ВІД ОДНОГО. Ситуація, коли скрипт залежить від сервісів іншого скрипта, який залежить від третього - ненормальна.

Зауважимо, мова йде про залежність. Цілком можна уявити собі скрипт, який викликає інші скрипти і видає узагальнений результат по ним, але це вже межа. Трохи складніше (наприклад, «запустити скрипт А якщо скрипт Б нічого не відпрацював») - вже за межею фолу. Не добре. А якщо скрипт А чи не відпрацював але не повідомив про це? Або трохи відпрацював, але потім відвалився так, що скрипту Б не вийде доробити (а ми, як автори скрипта А, і подумати не могли про подібне)?

Що ж в загальному повинен робити хороший скрипт? Зрощувати кілька програм в конкретну систему. Можете вважати програми за деталі конструктора. А сам конструктор - за скрипт.

Буває так, що скрипти перероджуються в програми. Раптово в скрипті з'являється якась логіка (алгоритм), яка стає нетривіальна (і корисна). У цей момент потрібно зловити це - і не полінуватися витратити в три рази більше часу, але зробити її програмою. Забезпечити її «м'ясом», яке відрізняє програму від скрипта. Додати сотню перевірок умов, замінити всі константи на конфігуруються змінні, приготувати її для роботи в «незвичних» умовах. Бажано зробити її публічною (тоді може наробитися практика використання).

Звичайний ***пайп*** вдає із себе практично ідеальний інструмент для конструювання простих програм:

Lssomething | Grep 'bla-bla' | sendmail root@host.ru -s 'bla-bal for something'.

Грань, в якій закінчується скрипт знайти складно. Скажімо так, цикл - ще терпимо. Перевірка умови - нормально. Але ось перевірка умови в циклі (більше, ніж вихід з циклу) - це вже погано. Якщо ж у вас цикл, в якому по перевірці умови запускається цикл - це 100% програма. Якщо у неї немає всього того, що повинно бути у програми, значить це просто дуже погана програма. Але ніяк не скрипт.

1.3 Командний рядок

Командний рядок Windows є стандартним засобом діагностики, налаштування і управління компонентами операційної системи і прикладних програмних забезпеченням. На відміну від графічного середовища користувача, командний рядок є більш гнучким і універсальним інструментом, що дозволяє вирішувати багато завдань адміністрування системи стандартними засобами Windows без установки додаткового програмного забезпечення.

Способи запуску командного рядка

Командний рядок Windows може бути запущена як і будь-яке інше стандартний додаток - через Головне меню Windows, з використанням діалогу Виконати (комбінації клавіш Win + R, Win + X), а також з використанням відкриття Провідником (за подвійним клацанням) ярлика або файлу C : \\ WINDOWS \\ System32 \\ cmd.exe. В результаті запуску відкриється вікно командного рядка із запрошенням до введення команд. Додаток cmd.exe часто називають командним процесором або інтерпретатором команд, а його основне вікно - консоллю Windows. Команди консолі є рядок символів, яка може містити вбудовані команди командного процесора (HELP, ECHO і т.п.), імена та шляхи виконуваних або командних файлів (C: \\ Windows \\ System32 \\ ping.exe), а також додаткові параметри, Якщо вони потрібні для виконання конкретної команди (***ping.exe i.ua***). Результати виконання команд відображаються у вікні консолі Windows і нерідко залежать від наявності

достатніх прав у поточного користувача. Для виконання команд в контексті облікового запису Адміністратора в операційних системах Windows Vista - Windows 10 необхідно використовувати режим *Запуск від імені Адміністратора.*

Довідник по командам CMD Windows

У різних версіях ОС сімейства Windows набір підтримуваних команд, їх параметри командного рядка, синтаксис і відображення результатів виконання можуть відрізнятися. Одна і та ж утиліта командного рядка може бути присутнім в стандартному постачанні однією версією Windows, але відсутні в іншій, або входити до складу додаткових коштів, як наприклад, Resource Kit або Software Development Kit.

В даній частині розділу представлено опис не тільки внутрішніх команд CMD, а й стандартних утиліт командного рядка, що входять до складу конкретних версії операційної системи (ОС) сімейства Windows. Деякі з наведених команд застаріли, і більше не підтримуються розробником або не мають сенсу в сучасних операційних системах, що відзначається в описі команди.

Список команд постійно оновлюється і включає в себе всі новітні команди, що додаються в стандартну поставку при виході нових версій або оновлень ОС Windows 10.

1.3 КОМАНДНІ ФАЙЛИ

Командні файли (пакетні файли, скрипти, сценарії,) - це звичайні текстові файли з розширенням .bat або .cmd, рядки яких представляють собою спеціальні команди командного процесора (інтерпретатора команд) і / або імена виконуваних файлів з параметрами. Командний процесор - це спеціальна програма, яка є обов'язковим елементом практично будь-якій операційній системи, головним призначенням якої є надання користувачеві можливості виконання певних програм без їх компіляції і створення виконуваних файлів. Для операційних систем DOS і Windows9X як інтерпретатора команд використовувався командний процесор command.com, для всіх інших ОС сімейства Windows (NT / 2k / XP / Vista / 7/8/10 і старше) - cmd.exe.

Незважаючи на безперервне вдосконалення засобів створення і виконання сценаріїв з використанням об'єктно-орієнтованих мов, звичайна командний рядок і прості командні файли, як і раніше залишаються основним інструментом для виконання рутинних дій, діагностики мережевих проблем, автоматизації процесів резервного копіювання і т.п. При всіх недоліках реалізації командного процесора в Windows, гідної альтернативи йому немає, і очевидно в найближчому майбутньому, не буде.

Як уже згадувалося вище, командний файл - це звичайний текстовий файл з набором команд, які послідовно виконуються командним процесором CMD Windows. Рядки командних файлів можуть містити специфічні команди самого процесора команд (FOR, ECHO, REM і т.п.) або імена виконуваних модулів (net.exe, regedit.exe, sc.exe) Командний процесор може бути запущений в інтерактивному режимі через Пуск - Виконати - CMD.EXE. В даному режимі, ви побачите вікно CMD.EXE (консолі) із запрошенням до введення команд. Можливий список більшості консольних команд можна отримати ввівши команду:

HELP

Довідкову інформацію по конкретній команді можна отримати, вказавши її назву в якості параметра команди HELP:

HELP Ім'я команди

Якщо ви працюєте в русифікованої версії Windows, то врахуйте, що в середовищі командного процесора символи національного алфавіту використовуються в DOS-кодуванні. Для перемикання між кодовими сторінками Windows і DOS використовується команда

CHCP номер сторінки

CHCP 866 - використовувати кодову сторінку 866 (DOS)

CHCP 1251 - використовувати кодову сторінку 1251 (WINDOWS)

Для перегляду і редагування командних файлів, що містять символи російського алфавіту потрібно використовувати редактор з підтримкою DOS-кодування. Якщо ви використовуєте стандартний додаток 'Блокнот'

(notepad.exe), то для правильного відображення символів російського алфавіту потрібно вибрати шрифт Terminal, за допомогою меню Правка - Шрифт ...
Зовнішній вигляд вікна CMD.EXE (консолі Windows) можна змінити за допомогою команди
COLOR
В якості аргументів для команди використовуються 2 шістнадцятиричні цифри, що задають колір фону і колір символу.
COLOR F0 - символи білого кольору на чорному тлі (використовується за умовчанням).
COLOR F0 - чорні символи на білому тлі.
COLOR 0E - світло-жовті символи на чорному тлі.
HELP COLOR - підказка для команди COLOR.
Робота з командним процесором передбачає використання двох стандартних пристроїв - пристрої введення (клавіатури) і пристрої виведення (дисплей). Однак, є можливість змінити стандартно використовуються пристрої введення-виведення за допомогою спеціальних символів - символів перенаправлення
- перенаправлення виведення
- перенаправлення вводу
Для виведення довідки не на екран а, наприклад, в файл з ім'ям help.txt, можна використовувати наступну команду:
HELP help.txt
При виконанні даної команди, в поточному каталозі буде створено файл з ім'ям help.txt, вмістом якого буде результат виведення команди HELP. Якщо файл help.txt існував на момент виконання команди, його вміст буде перезаписано. Для того, щоб дописати дані в кінець існуючого файлу, використовують подвоєння символу перенаправлення виведення - "
Приклад:
HELP GOTO myhelp.txt - в файл myhelp.txt буде видана довідка по використанню команду GOTO
HELP COLOR myhelp.txt - в кінець файлу myhelp.txt буде дописана довідка по

використанню команди COLOR

Найпростіший приклад перенаправлення вводу:

Cmd.exe commands.txt - командний процесор не чекатиме введення команд з клавіатури, а вважає їх з файлу commands.txt. Фактично, вказаний текстовий файл в даному випадку є командним файлом.

При запуску командного процесора можна вказати конкретну команду в якості аргументу командного рядка:

Cmd.exe / C HELP FOR - виконати команду HELP FOR і завершитися (параметр командного рядка або ключ / C)

Cmd.exe / K HELP FOR - виконати команду HELP FOR і перейти в режим очікування подальшого введення команд (ключ / K)

Детальну довідку по використанню cmd.exe можна отримати, ввівши в якості аргументу ключ /?

Cmd.exe /?

Крім символів перенаправлення вводу-виводу в командному рядку можуть використовуватися символи об'єднання команд - & (амперсанд) і | (Вертикальна риса)

& - одиночний амперсанд використовується для поділу декількох команд в одній командному рядку.

Команда1 & команда2 - виконується перша команда, потім друга команда.

&& - умовне виконання другої команди. Вона буде виконана, якщо код завершення (або код повернення) першої команди дорівнює нулю, тобто Команда виконана успішно. Успішність виконання команди визначається значенням спеціальної змінної середовища ERRORLEVEL.

Команда1 && команда2 - виконується команда1, а команда2 виконується, тільки якщо перша була виконана успішно.

|| - умовне виконання другої команди. Якщо перша команда завершилася з кодом повернення не рівним нулю (неуспішно), то виконується команда, наступна за подвійний вертикальної рисою.

Команда1 || Команда2 - якщо команда1 виконана неуспішно, то запускається на виконання команда2

Команду, наступну після знаків об'єднання, не потрібно укладати в подвійні лапки, інакше командний процесор подвоїть їх і повідомить про помилку.

Виконання командного рядка

Cmd.exe / C 'HELP IF' & 'HELP IF'

Завершиться виконанням першої команди і повідомленням про помилку для другої:

" HELP 'не є внутрішньою або зовнішньою командою, що виконується програмою або пакетним файлом.

Файли з розширенням .bat або .cmd в середовищі Windows стандартно відкриваються командним процесором аналогічно розглянутому вище прикладу з перенаправленням введення, коли послідовність команд зчитується ні з пристрою введення, а з текстового файлу.

Використання змінних в командних файлах.

При роботі з командними файлами, дуже важливе значення має такий інструмент як змінні оточення (environments) - змінні, значення яких визначають середу, в якій виконуються команда або пакетний файл. Іноді їх називають змінними середовища. Значення, що приймаються цими змінними формуються при завантаженні Windows, реєстрації користувача в системі, старті або завершення деяких додатків, і, крім того, можуть бути задані за допомогою спеціальної команди **SET**

SET змінна \u003d рядок

Змінна - ім'я змінної середовища.

Рядок - рядок символів, що привласнюється зазначеної змінної.

Наприклад, командний рядок

SET myname \u003d Vasya

Створює змінну myname, приймаючу значення Vasya.

Значення, присвоєне будь-якої змінної, є для обробки в командному рядку або в командному файлі з використанням її імені, укладеного в знаки відсотка -%.

Наочний приклад - команда видачі тексту на дисплей ECHO у вигляді:
ECHO date - виведе на екран слово 'date', а команда
ECHO% date% виведе на екран значення змінної date - поточну дату в форматі операційної системи.
Грамотне використання стандартних змінних оточення дозволяють створювати командні файли, які будуть однаково виконуватися на різних комп'ютерах, незалежно від їх конкретної конфігурації і призначених для користувача налаштувань. На практиці, за допомогою команди SET зазвичай задається і модифікується шлях пошуку виконуваних програм - змінна оточення PATH.
SET PATH \u003d C: \\ Windows; C: \\ windows \\ system32
Після виконання даної команди, пошук виконуваних файлів буде виконуватися в каталозі C: \\ Windows, і, якщо результат неуспешен, в C: \\ windows \\ system32
При необхідності виконати програму, наприклад, myedit.exe, розміщену в каталозі C: \\ NewProgs необхідно або вказати повний шлях виконуваного файлу, або зробити поточним каталогом каталог з програмою і використовувати тільки його ім'я. Якщо в командному рядку не задано повний шлях, а тільки ім'я виконуваного файлу - myedit.exe то спочатку буде виконуватися пошук файлу myedit.exe в поточному каталозі, і якщо він не буде знайдений - в каталогах, список яких визначається значенням змінної **PATH.**
Символ; Є роздільником елементів у списку шляхів пошуку. Якщо в наведеному прикладі, поточним каталогом не є C: \\ NewProgs, і в інших каталогах, заданих значенням змінної PATH. Немає виконуваного файлу myedit.exe, то запуск програми myedit.exe завершиться помилкою. Однак, якщо модифікувати значення змінної PATH, додавши в неї вимагається каталог, то вказівку повного шляху виконуваного файлу стає необов'язковим. Команда
SET PATH \u003d C: \\ NewProgs;% path%
Змінить поточне значення PATH, додавши каталог C: \\ NewProgs в початок списку. Для додавання каталогу в кінець списку використовується дещо інша конструкція:

SET PATH \u003d% path%; C: \\ NewProgs

Виконання команди SET без параметрів дозволяє отримати поточні значення змінних оточення:

NUMBER_OF_PROCESSORS \u003d 1 - кількість процесорів

OS \u003d Windows_NT- тип ОС

Path \u003d C: \\ WINDOWS \\ system32; C: \\ WINDOWS; C: \\ Program Files \\ Far - шлях пошуку файлів.

PATHEXT \u003d .COM; .EXE; .BAT; .CMD; .VBS; .VBE; .JS; .JSE; .WSF; .WSH - розширення для виконуваних файлів.

PROCESSOR_ARCHITECTURE \u003d x86 - архітектура процесора.

PROCESSOR_IDENTIFIER \u003d x86 Family 6 Model 8 Stepping 1, AuthenticAMD - ідентифікатор процесора.

PROCESSOR_LEVEL \u003d 6 - рівень (номер моделі) процесора.

PROCESSOR_REVISION \u003d 0801 - версія процесора.

ProgramFiles \u003d C: \\ Program Files - шлях до папки 'Program Files'

PROMPT \u003d $ P $ G - формат запрошення командного рядка $ P - шлях для поточного каталогу $ G - знак ".

SystemDrive \u003d C: - буква системного диска.

SystemRoot \u003d C: \\ WINDOWS - каталог ОС Windows.

Значення деяких змінних по команді SET не відображаються, хоча і присутні в системі. В основному, це змінні, що приймаються значення яких динамічно змінюються:

Передача параметрів командного файлу.

Дуже корисною особливістю роботи з командними файлами є можливість отримувати значення параметрів командного рядка і використовувати їх в операціях усередині самого командного файлу.

BAT-файл параметр1 параметр2 ... параметрN

У самому командному файлі перший параметр буде доступний як значення змінної% 1, другий -% 2 і т.п. Шлях і ім'я самого командного файлу є як значення змінної% 0. Для прикладу створимо командний файл, завданням якого

буде видача на екран значень введених при його запуску параметрів командного рядка. Для виведення тексту на екран використовується команда ECHO текст, проте якщо 'текст' замінити на% 0, - то буде видано ім'я командного файлу,% 1 - перший аргумент, заданий в рядку запуску,% 2 - другий і т.д.

Створюємо, наприклад, командний файл params.bat такого змісту:

Echo off echo Це командний файл% 0

Echo Перший параметр \u003d% 1

Echo Другий параметр \u003d% 2

Echo Третій параметр \u003d% 3

І запускаємо його на виконання такої команди:

Params.bat FIRST second 'two words'

Параметри командного рядка, що містять пробіли повинні полягати в подвійні лапки.

У першому рядку наведеного вище командного файлу використовується команда echo off, призначена для того, щоб оброблювані командним процесором рядки не видавалися на екран.

При обробці вхідних параметрів необхідно знати, чи були вони взагалі задані в командному рядку. Перш ніж виконувати будь-яких вхідних параметрів, що передаються командному файлу, можна перевірити, чи є значення змінної% 1 порожнім, що можна зробити, уклавши її, наприклад в подвійні лапки, і перевіривши отриманий результат на наявність цих лапок, наступних підряд:

If '% 1' EQU '' goto error.....: error

Echo Повинен бути заданий хоча б один вхідний параметр

Exit

У командних файлах великого розміру важко обійтися без довідкової інформації у вигляді коментарів, для чого використовується конструкція

REM пробіл текст

Рядки, що починаються з REM пробіл, вважаються коментарями і командним процесором ігноруються.

Rem ECHO OFF вимикає режим виведення змісту рядків командного файлу на екран

REM буде виводитися тільки результат їх виконання.

ECHO OFF

Echo Перший параметр \u003d% 1

Echo Другий параметр \u003d% 2

Echo Третій параметр \u003d% 3

Якщо до імені команди першим символом додається @, то незалежно від режиму ECHO (ON або OFF), висновок оброблюваного рядка не виконується. Спробуйте в даному прикладі 'ECHO OFF' замінити на '@ECHO OFF' - результат говорить сам за себе. Рядок, яка вимикає режим виведення, що не буде видаватися на екран. Висновок вмісту оброблюваних рядків (відлуння) на екран встановлюється по команді ECHO ON і, звичайно, використовується з метою діагностики.

Rem ECHO ON включає режим виведення змісту рядків командного файлу на екран

REM буде сама командний рядок і результат її виконання,

REM але рядки, що починаються символом @ не виводитися будуть ECHO ON

@echo Перший параметр \u003d% 1

@echo Другий параметр \u003d% 2

@echo Третій параметр \u003d% 3

Переходи і мітки.

У командних файлах можна використовувати команди умовного переходу, що змінюють логіку їх роботи в залежності від виникнення певних умов. В якості ілюстрації створимо командний файл, метою якого буде привласнення заздалегідь певної літери для знімних носіїв. Умови такі - є 2 знімних диска, один з яких повинен бути видно в провіднику як диск X: а другий - як диск Y: незалежно від того, в який порт USB вони підключені і які букви присвоєні їм операційною системою. Для прикладу, будемо вважати, що реальні диски можуть бути підключені як F: або G: Розпізнавання дисків будемо виконувати

за наявністю файлу з певним ім'ям (найкраще такий файл зробити прихованим в кореневому каталозі знімного диска і назвати його як-небудь незвично):
Flashd1.let - на першому диску
Flashd2.let - на другому
Таким чином, завдання командного файлу полягає в тому, щоб перевірити наявність на змінних дисках F: і G: файлів Flashd1.let або Flashd2.let і, в залежності від того, який з них є присутнім, привласнити диску букву X: чи Y:
Для пошуку файлу на диску скористаємося командою IF EXIST:
IF EXIST имя_файла команда
Як команди, яка буде виконана при задоволенні умови використовуємо SUBST, яка призначена для зіставлення каталогу і віртуального диска.
SUBST X: C: \\ - - створити віртуальний диск X :, вмістом якого буде кореневої каталог диска C:
Для вирішення поставленого завдання, створюємо командний файл, наприклад з ім'ям setletter.bat, такого змісту:
@ECHO OFF
IF EXIST G: \\ *flashd1.let SUBST X: G:* \\
IF EXIST F: \\ *flashd1.let SUBST X: F:* \\
IF EXIST G: \\ *flashd2.let SUBST Y: G:* \\
IF EXIST F: \\ flashd2.let SUBST Y: F: \\

Після виконання цього командного файлу у вас з'являться диски X: і Y: Однак, якщо такий файл виконати повторно, команда SUBST видасть повідомлення про помилку - адже диски X: і Y: вже існують.
Тому, бажано обійти виконання SUBST, якщо віртуальні диски X: і Y: вже створені, або видаляти їх, використовуючи SUBST з параметром -d перед підключенням. Спробуємо змінити командний файл setletter.bat з використанням команди переходу GOTO, що здійснює передачу управління рядку пакетного файлу на вказану мітку.
GOTO мітка

В якості мітки використовується рядок символів, що починається з двокрапки. Зробимо зміни в нашому командному файлі, щоб не виникало повідомлень про помилку:

```
@ECHO OFF
REM якщо не існує X: - то перейдемо на мітку SETX
IF NOT EXIST X: \\ GOTO SETX
REM якщо існує X: - перейдемо на перевірку наявності Y:
GOTO TESTY
: SETX
IF EXIST G: \\ flashd1.let SUBST X: G: \\
IF EXIST F: \\ flashd1.let SUBST X: F: \\
: TESTY
REM якщо Y: існує - завершимо командний файл.
IF EXIST Y: \\ GOTO EXIT
IF EXIST G: \\ flashd2.let SUBST Y: G: \\
IF EXIST F: \\ flashd2.let SUBST Y: F: \\
REM вихід з командного файлу
: EXIT
Exit
```

При виконанні зміненого таким чином командного файлу, повідомлення про помилку при виконанні SUBST зникне.

Звичайно, даний приклад не може вважатися взірцем програмування, але головна мета полягає не в написання оптимального командного сценарію, а в демонстрації принципів використання міток і переходів. У всіх наступних прикладах, наскільки це можливо, використовується саме такий підхід - важлива не оптимальність початкового тексту, а його максимальна простота для розуміння.

Одним з найважливіших прийомів при написанні складних командних файлів є аналіз успішності виконання конкретної команди або програми. Ознаки помилок при виконанні команд можна відстежувати, аналізуючи спеціальну

змінну **ERRORLEVEL**, значення якої формується при виконанні більшості програм. Зазвичай ERRORLEVEL дорівнює нулю, якщо програма завершилася без помилок і одиниці - при виникненні помилки. Можуть бути й інші значення, якщо вони передбачені в виконуваної програмою.
Як команди в рядку командного файлу можна використовувати також командний файл. Причому, для передачі з поверненням назад до точки виконання викликає командного файлу використовується команда CALL.
Спробуємо створити командний файл test.bat, такого змісту:
@ECHO OFF
ECHO Виклик 1.bat
CALL 1.bat
ECHO Повернення.
У цьому ж каталозі, створіть другий файл під ім'ям 1.bat, що містить команду

PAUSE - призупинити виконання командного файлу до натискання будь-якої клавіші.
@ECHO OFF
Pause
При виконанні командного файлу test.bat буде видано на екран повідомлення **‘’Виклик 1.bat’’**

І управління отримає командний файл 1.bat з однією єдиною командою pause. Після розпочата клавіші на клавіатурі управління буде повернуто викликав командному файлу на рядок 'ECHO Повернення.' І на екран буде видано
Повернення.
Якщо ж у файлі test.bat прибрати CALL, залишивши '1.bat', то, виконається командний файл 1.bat, і повернення в test.bat виконуватися не буде.

Викликаний командний файл може створювати змінні і присвоювати їм певні значення, які будуть доступні для обробки в зухвалій файлі. Спробуйте змінити файл test.bat наступним чином:

@ECHO OFF

ECHO Виклик 1.bat

CALL 1.

Приклади командних файлів.

Використання утиліт командного рядка і командних файлів нерідко дозволяють вирішити багато проблем пов'язані з повсякденним експлуатацією комп'ютерної техніки. Більшість системних адміністраторів і грамотних користувачів продовжують ними користуватися, не дивлячись на те, що в Windows проявилося нове, більш потужне і сучасне засіб управління системою - WMI (Windows Management Instrumentation) і багатофункціональна оболонка користувача Power Shell. Очевидно, не в останню чергу, це обумовлено простотою реалізації і достатньою ефективністю командного рядка для вирішення повсякденних завдань обслуговування системи. Нижче наведені прості приклади з коментарями, які демонструють деякі можливості і способи застосування .cmd і .bat

Командний рядок:

Nf.bat myfile.txt - створити файл з ім'ям myfile.txt в поточному каталозі.

Nf.bat C: \\ myfile.txt - створити файл в кореневому каталозі диска C:

Nf.bat '% USERPROFILE% \\ myfile.txt' - створити файл в каталозі профілю поточного користувача.

Можливість створення файлів в системних каталогах залежить від налаштувань безпеки системи і прав користувача, в контексті облікового запису якого виконується команда. Багато команд можуть бути виконані тільки користувачем з правами адміністратора.

Розширення командного файлу (.bat) можна не набирати і команда ще більше спрощується:

Nf myfile.txt

У тексті командного файлу присутній перевірка, чи задано ім'я створюваного файлу в командному рядку (if '% 1%' EQU '' goto error), і якщо не задано - виводиться повідомлення про помилку і командний файл завершує свою роботу.

У плані вдосконалення функціоналу, можна додати в цей командний файл перевірку на наявність файлу з ім'ям, зазначеним в командному рядку і попередженням користувача про його можливу перезапису.

Виконання команд за розкладом.

В операційних системах WINDOWS XP і старше існує утиліта командного рядка AT.EXE, що дозволяє управляти завданнями для планувальника завдань Windows, і таким чином, виконати команду або пакетний файл в зазначений час на локальному або віддаленому комп'ютері.

В операційних системах Windows 7 і старше, утиліта at.exe присутній, але визнана застарілою і Нерекомендовані до використання в майбутньому. Замість неї рекомендується використовувати schtasks.exe, яка має більші можливості, але складніше у використанні. Приклади використання сучасної утиліти є в розділі зі списком команд, а на даній сторінці, все ж скористаємося простий класичної AT. Команда AT призначена для запуску команд і програм в зазначений час по певних днях. Для використання команди AT необхідно, щоб була запущена служба планувальника завдань (в сучасних ОС Windows, за замовчуванням ця служба запущена завжди). .

Приклади команди

AT [\\\\ ім'я_комп'ютера] [[код] [/ DELETE] | / DELETE [/ YES]]

AT [\\\\ ім'я_комп'ютера] час [/ INTERACTIVE] [/ EVERY: день [...] | / NEXT: день [...]] 'команда'

\\\\ ім'я_комп'ютера - ім'я віддаленого комп'ютера. Якщо цей параметр опущений, завдання ставиться до локального комп'ютера.

Код - порядковий номер запланованe завдання. Вказується якщо потрібно скасувати вже заплановану завдання за допомогою ключа / delete.

/ Delete - скасувати заплановану завдання. Якщо код завдання опущений, скасовуються всі завдання, заплановані для зазначеного комп'ютера.
/ Yes - не буде підтверджувати з'єднання під час скасування усіх запланованих завдань.
Час - Час запуску команди.
/ Interactive - інтерактивний режим, дозвіл взаємодії завдання з користувачем. Завдання, запущені без цього ключа невидимі для користувача комп'ютера.
/ Every: день [...] Запуск завдання здійснюється за вказаними днях тижня або місяця. Якщо дата опущена, використовується поточний день
Місяці.
/ Next: день [...] Завдання буде запущена в наступний вказаний день тижня (наприклад в наступний четвер). Якщо дата опущена, використовується поточний день місяця.
'Команда' - Команда або ім'я командного файлу

Приклади використання:

- Перегляд списку запланованих завдань: **AT**
- - Видалення вже спланованих завдань:

AT 3 / DELETE - видалення завдання з номером 3
AT / DELETE / YES - видалення всіх завдань без запиту підтвердження

- **Створення інтерактивних завдань**

At \\\\ ***SERVER 15:21 / interactive notepad.exe*** - на комп'ютері SERVER о 15:21 запустити видиме для користувача додаток 'Блокнот' (notepad.exe)
AT 15:30 / interactive regedit.exe - о 15:30 запустити видимий редактор реєстру на своєму комп'ютері.

- Аналог 'будильника' - спливаючі вікна з текстом, що нагадують про необхідність будь-яких дій. Для посилки повідомлення віддаленому користувачеві в операційних системах Windows 2k / XP використовується утиліта **NET.EXE** в режимі відправки повідомлення **SEND**. На комп'ютерах повинна бути запущена служба повідомлень, інакше **NET SEND** не працюватиме. В ОС Windows Vista і більш пізніх, відправку

повідомлень потрібно виконувати за допомогою утиліти **msg.exe**, оскільки команда net send в цих ОС більше не підтримується.

AT 17:30 net.exe send COMP Пора додому - о 17:30 відправити повідомлення 'Пора додому' користувачеві комп'ютера COMP

AT 17:30 msg.exe * / server: COMP Пора додому - то ж, що і в попередньому випадку, але з використанням команди msg

AT 17:39 msg.exe * Пора додому - запланувати на 17:39 відправку текстового повідомлення 'Пора додому' всім користувачам даного комп'ютера в середовищі Windows Vista / 7/8

AT \\\\ PROXY 15:30 net.exe send COMP2 Test Message - створити завдання на комп'ютері PROXY, щоб о 15:30 їм було відправлено повідомлення 'Test Message' на комп'ютер COMP2

AT 15:45 net.exe send ім'я свого комп'ютера Task Scheduler test - о 15:45 на своєму комп'ютері показати повідомлення 'Task Scheduler test'

Для доступу до віддаленого комп'ютера і створення завдань, користувач, який виконує команду AT повинен володіти відповідними правами по відношенню до віддаленої системи.

Створювані командою AT завдання доступні для обробки в середовищі користувача за допомогою оснастки 'Призначені завдання' Windows. Пуск - Панель управління - Призначені завдання - тут можна переглядати, редагувати і вилучати командою AT завдання. У Windows Vista / 7-8 використовується меню - **Панель управління - Адміністрування - Планувальник завдань.**

Зупинка і запуск системних служб.

Для зупинки і запуску служб з командного рядка, в будь-якої версії Windows, можна скористатися командою **NET.EXE**

NET.EXE STOP <ім'я служби>

NET.EXE START <ім'я служби>

Як параметр команди можна використання як короткий, так і повне ім'я служби ('N Dnscache' - короткий, 'DNS-клієнт' - повне ім'я служби). Ім'я служби, що

містить прогалини полягає в подвійні лапки. Приклад перезапуску служби 'DNS-клієнт'

Net stop 'DNS-клієнт'

Net start 'DNS-клієнт'

Те ж, з використанням короткого імені:

Net stop Dnscache

Net start Dnscache

Повне ім'я служби можна скопіювати з 'Панель управління' - 'Адміністрування' - ***'Служби' - Ім'я служби - 'Властивості' - 'Коротке ім'я'***.

Те ж саме, але в режимі командного рядка: ***'Пуск' - 'Виконати' - services.msc***.

Для управління службами набагато зручніше скористатися ***утилітою PsService.exe з утиліт PsTools від Microsoft Sysinternals***. Утиліта не вимагає установки і працює в будь-який OS Windows. Крім запуску і зупинки, дозволяє виконати пошук конкретної служби на комп'ютерах локальної мережі, опитати стан і конфігурацію служби, змінити тип запуску, призупинити службу, продовжити, перезапустити.

Для роботи з системними службами в Windows XP і старше, можна використовувати утиліту sc.exe, що дозволяє не тільки зупинити / запустити службу, а й опитати її стан, параметри запуску і функціонування, змінити конфігурацію, а також працювати не тільки з системними службами, але І з драйверами. При наявності відповідних прав, можна управляти службами не тільки на локальній, а й на віддаленій машині. Приклади:

Sc.exe stop DNSCache - зупинити службу DNSCache на локальному комп'ютері.

Sc \\\\ 192.168.0.1 query DNSCache - опитати стан служби DNSCache на комп'ютері с IP-адресою 192.168.0.1

Sc \\\\ COMP start DNSCache запустити службу DNSCache на комп'ютері COMP

Підказку по роботі з утилітою можна отримати, ввівши:

Sc /?

Діалог з користувачем

Для діалогу з користувачем можна використовувати команду:

SET / P ім'я змінної \u003d текст

При виконанні якої, на екран видається текстове повідомлення текст і очікується введення відповіді. Наприклад, виконаємо запит введення пароля і дамо його значення змінної 'pset':

Set / p pset \u003d 'Enter password -'

Echo Password is -% pset%

Недоліком даного способу є неможливість продовження виконання командного файлу при відсутності відповіді користувача, тому дуже часто, замість команди set використовуються інші засоби, в тому числі і сторонні програми. У складі операційних систем сімейства Microsoft Windows є утиліта командного рядка CHOICE дозволяє досить просто реалізувати діалог з користувачем і проаналізувати введені їм дані, проте в різних версія ОС утиліта може бути присутнім в стандартному постачанні (Windows 7) або входити в набори додаткових програмних інструментів (Resource Kit Windows XP) Найпростіша версія - CHOICE.COM Завантажити (1.7кб), що працює у всіх ОС сімейства Windows.

CHOICE видає користувачеві текстове повідомлення і чекає вибору одного з заданих варіантів відповіді (натискання клавіш на клавіатурі). За результатами вибору формується змінна ERRORLEVEL, значення якої дорівнює порядковому номеру вибору. За замовчуванням варіантів вибору два - Y або N. Якщо відповідь дорівнює **Y - то ERRORLEVEL \u003d** 1, якщо N - то **ERRORLEVEL \u003d 2**. Можна використовувати більш 2-х варіантів вибору і є можливість задати вибір за замовчуванням, коли користувач за певний час не натиснула жодної клавіші. **Формат командного рядка:**

CHOICE [/ C [:] choices] [/ N] [/ S] [/ T [:] c, nn] [text]

/ C [:] choices - визначає допустимі варіанти вибору. Якщо не задано - YN

/ N - не видавати варіанти вибору.

/ S - малі та великі літери відрізняються.

/ T [:] c, nn - Вибір за замовчуванням дорівнює 'c' через 'nn' секунд

Text - Рядок тексту виводиться в якості запиту

Створимо командний файл, що демонструє використання **CHOICE.** Він буде реагувати на натискання клавіш '1', '2', 3 'i' 0 '. При натисканні' 0 'виконується завершення, а при натисканні інших - повідомлення користувачу. Якщо протягом 10 секунд нічого не натиснуто - завершення.

```
@ECHO OFF
: CHOICE
CHOICE / C: 1 230 / T: 0,10 Ваш варіант
IF% ERRORLEVEL% EQU 4 GOTO EXIT
Echo Ваш вибір \u003d% ERRORLEVEL%
GOTO CHOICE
: EXIT
```

Тепер, використовуючи CHOICE ви можете створювати командні файли, логіка роботи яких може визначатися користувачем.

Пошук комп'ютерів із запущеним додатком

В операційних системах Windows XP і старше є стандартна утиліта для отримання списку процесів, що виконуються в системі tasklist.exe. Вона має дуже непоганими можливостями і може використовуватися для пошуку виконуються додатків як на локальному, так і віддаленому комп'ютері. Також, можна скористатися допоміжною утилітою **PSList**.exe з пакета **PSTools** від **Microsoft Sysinternals**. Обидві утиліти можна використовувати для пошуку додатків, що виконуються на локальному або віддаленому комп'ютері, але PSlist має більш простий функціонал, менш відома серед комп'ютерних фахівців і обрана в якості засобу вирішення поставленого завдання. Для того, щоб використовувати стандартну утиліту **tasklist.exe**, будуть потрібні зовсім незначні переробки, пов'язані з різницею в синтаксисі команд.

Завдання полягає в тому, що потрібно створити командний файл, який би виявляв в локальній мережі комп'ютери з виконується програмою, ім'я якої (початкова частина імені), задається як параметр при запуску, наприклад, game. При виявленні, наприклад, потрібно надіслати користувачеві комп'ютера **ADMINCOMP** і виявлене додаток примусово завершити.

Для пошуку будемо використовувати утиліту **Pslist.exe** і аналізувати її код повернення. Значення змінної **ERRORLEVEL** рівне нулю означає, що утиліта виявила на віддаленому комп'ютері процес, що задовольняє умовам пошуку. Ім'я процесу для пошуку будемо ставити в якості параметра при запуску командного файлу. Дамо нашому командному файлу ім'я psl.bat. Запуск з параметром буде виглядати наступним чином:

Psl.bat game

У разі, коли в командному файлі використовується параметр, не завадить перевірити, чи не встановлено він в командному рядку при запуску, і, якщо не заданий, завершити виконання, відобразивши попередження користувачеві. Якщо ж параметр заданий, виконання файлу буде продовжено, наприклад, до мітці 'PARMOK':

@echo off

If '% 1' NEQ '' GOTO PARMOK

ECHO Потрібно задати ім'я процесу для пошуку

Exit

: PARMOK

На наступному кроці потрібно забезпечити послідовне формування IP-адрес комп'ютерів для командного рядка утиліти PSlist. Як простий приклад, можна скористатися присвоєнням тимчасової змінної значення постійної складової адреси для локальної мережі (наприклад - 192.168.0.) І обчислюється значення молодшої частини (наприклад, в діапазоні 1-254).

Наприклад, нам необхідно перевірити комп'ютер в діапазоні адрес:

192.168.0.1 - 192.168.0.30:

Set IPTMP \u003d 192.168.0. - старша частина адреси

Set / A IPLAST \u003d 1 - молодша частина. Ключ / А означає обчислюється числове вираз

Set IPFULL \u003d% IPTMP %% IPLAST% - значення повного IP-адреси.

Командний рядок для PSlist буде виглядати Следующим чином:

Pslist \\\\% IPFULL%% 1

Тепер залишилося тільки циклічно запускати PSlist, додаючи в кожному циклі одиницю до молодшої частини адреси, поки її значення не досягне 30 і аналізувати значення ERRORLEVEL після виконання. Для аналізу результату будемо виконувати перехід командою:

GOTO REZULT% ERRORLEVEL%

Забезпечує перехід на мітку REZULT0 при виявленні процесу і на REZULT1 - при його відсутності.

Остаточний вміст командного файлу:

@echo off

If '% 1' NEQ '' GOTO PARMOK

ECHO *Потрібно задати ім'я процесу для пошуку*

Exit

: PARMOK

Set IPTMP *\u003d 192.168.0.*

Rem *Задамо початкове значення 'хоста' IP- адреси*

Set / A IPLAST \u003d 1

Rem M0 *- мітка для організації циклу*

: M0

Rem Змінна IPFULL *- повне значення поточного IP-адреси*

Set IPFULL \u003d% IPTMP %% IPLAST%

Rem Якщо 'хвіст' більше 30 - на вихід

IF% IPLAST% GTR 30 GOTO ENDJOB

Pslist *\\\\% IPFULL%% 1*

GOTO REZULT% ERRORLEVEL%

: REZULT0

Rem Якщо знайдено додаток-відправимо повідомлення на ***ADMINCOMP***

Net send ADMINCOMP Запущено% 1 -% IPFULL%

Rem І завершимо програму за допомогою PSkill

Pskill *\\\\% IPFULL%% 1*

: REZULT1

Rem Сформуємо наступний IP-адреса

Set / A IPLAST \u003d% IPLAST% + 1

Rem Перейдемо на виконання наступного кроку

GOTO M0

Rem Завершення роботи

: endjob

Exit

В завершенні додам, що для того, щоб цей скрипт працював, PSlist.exe і PSkill.exe повинні бути доступні в шляхах пошуку виконуваних файлів, наприклад в каталозі WINDOWS \\ SYSTEM32. Користувач, який запускає його, повинен володіти правами адміністратора по відношенню до об'єкту сканування комп'ютерів. І, якщо поточний користувач таким не є, то в параметри запуску утиліт PSlist.exe і PSkill.exe потрібно додати ключі, що задають ім'я користувача і пароль. В якості альтернативи утиліті PSKill.exe можна скористатися стандартною Taskkill.exe. Використання стандартних утиліт tasklist.exe і taskkill.exe обмежена тим фактором, що їх параметри командного рядка не допускають можливість завдання імені користувача і пароля для доступу до віддаленого комп'ютера.

Пошук комп'ютерів із запущеним додатком за списком

У попередньому прикладі використовувався прямий перебір IP-адрес комп'ютерів в локальній мережі, що не завжди ефективно, оскільки в процедуру опитування можуть бути залучені і вимкнені комп'ютери. Вирішимо задачу іншим способом - створимо текстовий файл зі списком комп'ютерів і опитаємо їх з цього списку.

Список можна отримати з мережевого оточення з використанням команди:

Net.exe view comps.txt

Після виконання такої команди файл comps.txt буде містити список такого вигляду:.

Ім'я сервера Нотатки

2 порожніх рядки

\\\\ AB1

\\\\ AB2

\\\\ ALEX

\\\\ BUHCOMP

\\\\ PC2

\\\\ SA

\\\\ SERVER

Команда виконана успішно.

Обробляти вміст цього текстового файлу будемо за допомогою команди **FOR** з ключем / F:

FOR / F ['ключі']% змінна IN (ім'я файлу) DO команда [параметри]

Дана команда дозволяє отримати доступ до рядків в текстовому файлі з використанням ключів:

Skip \u003d n - пропустити n рядків від початку файлу (в нашому прикладі - 4 рядки)

Eol \u003d <символ> - не використовувати рядки, що починаються з заданого символу. (В нашому випадку - пропустити останній рядок, що починається з кириличного символу 'К' - 'Команда виконана успішно'

Tokens \u003d n - брати для обробки n-е слово в рядку (в нашому випадку - 1-е слово)

Остаточний вигляд команди:

FOR / F 'eol \u003d До skip \u003d 4 tokens \u003d 1' %% I IN (comps.txt)

DO (

Pslist.exe -u admin -p pass %% I% 1

IF NOT ERRORLEVEL 1 net.exe send ADMINCOMP %% i% 1

)

Звернемо увагу - в пакетних файлах для змінних команди FOR використовується два знака відсотка (запис %% змінна замість% змінна) і імена змінних враховують регістр букв (% і відрізняється від% I).

Працювати це буде наступним чином - пропускаються перші 4 рядки текстового файлу зі списком комп'ютерів, і далі в циклі змінної I присвоюється значення першого слова (текст від початку рядка до роздільник - пропуск), виконується утиліта PSlist.exe, для якої в якості імені комп'ютера використовується Значення цієї змінної. Якщо ERRORLEVEL менше 1 - завдання з шуканим ім'ям присутній в списку процесів і виконується відправка повідомлення за допомогою NET SEND.

Остаточний вміст командного файлу:

@echo off

If '% 1' NEQ '' GOTO PARMOK

ECHO *Потрібно задати ім'я процесу для пошуку*

Exit

: PARMOK

REM Створимо текстовий файл comps.txt зі списком комп'ютерів за допомогою **NET.EXE VIEW**

Net view / DOMAIN: MyDomain> comps.txt

REM FOR / F 'параметри' - використання даних з файлу

REM eol \u003d K - не використовувати рядки, що починаються з 'K' - 'Команда виконана успішно'

REM skip \u003d 4 - пропустити перші 4 рядки в файлі

REM tokens \u003d 1 - брати для обробки 1-е слово в рядку

FOR / F 'eol \u003d До skip \u003d 4 tokens \u003d 1' %% i in (comps.txt) do (

Pslist.exe -u admin -p pass %% i% 1

IF NOT ERRORLEVEL 1 net.exe send% COMPUTERNAME% *Комп'ютер - %% і процес -% 1*

Вимкнення комп'ютерів за списком, створеному на основі мережевого оточення.

Попередній приклад наводить на думку, що можна було б, наприклад, створити пакетний файл для швидкого вимикання всіх комп'ютерів в локальній

мережі. Вимикання проводиться утилітою стандартною утилітою Shutdown.exe (опис в розділі зі списком команд CMD Windows - команда Shutdown). Як і в попередньому прикладі, спочатку створюється файл зі списком комп'ютерів на основі мережевого оточення, а потім виконується їх почергове вимикання, за умови, що немає потреби вимикати комп'ютер є 'свій', на якому виконується даний командний файл. **Вміст файлу**:

Rem @echo off

REM Тут потрібно задати

REM ім'я домену або робочої групи для яких будується список машин для виключення

Set MyDomain \u003d ім'я домену

REM

REM Створимо текстовий файл comps.txt зі списком комп'ютерів за допомогою **NET VIEW**

Net view / DOMAIN:% MyDomain%> comps.txt

REM

REM FOR / F 'параметри' - використання даних з файлу

REM eol \u003d K - не використовувати рядки, що починаються з 'K' - 'Команда виконана успішно'

REM skip \u003d 4 - пропустити перші 4 рядки в файлі

REM tokens \u003d 1 - брати для обробки 1-е слово в рядку

FOR / F 'eol \u003d До skip \u003d 4 tokens \u003d 1' %% i in (comps.txt) do (

REM Свій комп'ютер вимикати Не будемо

REM Якщо ім'я комп'ютера не дорівнює COMPUTERNAME - вимикаємо

IF / I %% i NEQ% COMPUTERNAME% shutdown -f -s -t 0 -m %% i

Командний файл повинен виконуватися в контексті облікового запису користувача з правами адміністратора по відношенню до вимикати комп'ютер. Ім'я домену або робочої групи задається в рядку:

Set MyDomain \u003d

У реальному житті зі списку вимикаються комп'ютерів потрібно виключити кілька штук, для чого зручно використовувати команду FIND в ланцюжку з net.exe в скрипті формування списку на основі мережевого оточення. Дана команда використовується для пошуку рядків у текстовому файлі за шаблоном. Ключ / V використовується для пошуку рядків не збігаються з шаблоном. Для виключення комп'ютерів, виключаючи server1 ... server4 зручно використовувати такий варіант:

Net view | Find '\\\\' | Find / v 'сервер1' | Find / v 'сервер2' | Find / v 'сервер3' | Find / v 'сервер4'> comps.txt

FOR / F 'tokens \u003d 1' %% i in (comps.txt) do shutdown.exe -f -s -m %% i

Пошук в локальній мережі ввімкнених комп'ютерів.

В даному прикладі мова йде про створення командного файлу, що дозволяє 'зібрати' список IP-адрес вузлів локальної мережі, включених на даний момент часу. Нічого принципово нового в плані створення командних файлів тут немає, але тим не менш, завдання пошуку включених вузлів в локальній мережі зустрічається досить часто, і вирішити її описаними вище способами, з використанням ping.exe і net view вдається далеко не завжди, оскільки в сучасних Версіях операційних систем сімейства Windows налаштування брандмауерів за замовчуванням, задають досить жорсткі правила, які блокують мережеві з'єднання ззовні, і відповіді на ехо-запит. Іншими словами, включений в локальну мережу комп'ютер (або інший мережний пристрій) може не відображатися в мережевому оточенні Windows і не відповідати на ехо-запити ('пінг').

Прийом, який використовується для отримання списку ввімкнених мережевих пристроїв при вирішенні даного завдання залишається тим самим - необхідно визначити характерні відмінності у вихідних повідомленнях стандартних мережевих утиліт в разі, коли пристрій вимкнений, і коли - захищене параноїдальними брандмауера. Для цього можна скористатися стандартною командою arp дозволяє переглядати вміст таблиць дозволу IP-адрес в фізичні (MAC) адреси.

При будь-якій передачі пакетів IP-протоколу з даного комп'ютера на IP-адресу іншого мережевого пристрою в локальній мережі, програмні засоби мережевих протоколів виконують процедуру визначення фізичної адреси мережевого адаптера одержувача (MAC-адресу одержувача). Всім мережевих пристроїв відправляється спеціальний циркулярний запит (запит який буде прийнятий усіма комп'ютерами даної підмережі), що означає 'чий MAC - адреса відповідає такому-то IP-адресою'. Якщо будь-яка мережеве пристрій впізнали свій власний IP-адреса, воно відправить ARP-відповідь, що містить відповідний MAC-адресу, який буде збережений в спеціальній таблиці відповідності адрес IP і MAC, що зберігається в оперативній пам'яті комп'ютера, що відправив ARP-запит. Запис інформації в дану таблицю виконується тільки при необхідності передачі будь-яких даних по протоколу IP, що можна ініціювати, наприклад, пингования опитуваного пристрою. Навіть якщо брандмауера повністю закриті всі з'єднання ззовні і блокується протокол ICMP (пристрій не 'пінг»), в буферній пам'яті сервісу ARP буде присутній запис відповідності IP і MAC, якщо пристрій було підключено до локальної мережі і брало участь в процедурі вирішення адреси. Таким чином, відсутність відповіді на пінг і наявність запису для пінгуемого IP-адреси в ARP-кеш є ознакою того, що пристрій включено і присутній в локальній мережі.

Для перегляду вмісту ARP-кеш можна скористатися командою

Arp -a - відобразити всі записи в таблиці ARP

Приклад відображення таблиці ARP:

Інтерфейс: 192.168.0.29 --- 0xa

Адреса в Інтернеті Фізична адреса Тип

192.168.0.1 00-1e-13-d6-80-00 динамічний

192.168.0.3 60-eb-69-08-18-d2 динамічний

Інтерфейс: 192.168.234.1 --- 0xf

Адреса в Інтернеті Фізична адреса Тип

192.168.234.255 ff-ff-ff-ff-ff-ff статичний

224.0.0.22 01-00-5e-00-00-16 статичний

224.0.0.252 01-00-5e-00-00-fc статичний

239.255.255.250 01-00-5e-7f-ff-fa статичний

Як видно з наведеної таблиці, наприклад IP - адреса 192.168.0.1 відповідає фізичній адресі мережного адаптера, рівний 00-1e-13-d6-80-00. Якщо ж мережевий адаптер з даними адресою буде недоступний, то такого запису в таблиці не буде.

Для розуміння алгоритму опитування мережі необхідно врахувати наступне: Дозвіл адрес ARP використовується тільки при передачі даних по IP-протоколу в межах сегмента локальної мережі, що задається маскою. Так, наприклад, для прикладу з IP адресою 192.168.0.1 і маски 255.255.255.0 це буде діапазон IP від 192.168.0.1 до 192.168.0.254. Звернення ж до будь-якого іншою адресою буде виконуватися через пристрої маршрутизації. Тобто При виконанні команди

Ping i.ua

У таблиці ARP буде не MAC-адресу мережевого адаптера, відповідного IP-адресою вузла з ім'ям 'yandex.ru', а MAC-адресу шлюзу, через який виконалася відправка IP-пакета для доставки одержувачу в 'чужій' мережі.

Нижче наводиться простий приклад командного файлу, що визначає список ввімкнених мережевих пристроїв локальної мережі за результатами виконання команд **PING і ARP.**

@ECHO OFF

REM Встановлюємо значення змінної IPTMP - постійної частини IP-адреси

Set IPTMP \u003d 192.168.213.

REM Мінлива N задає кількість опитуваних IP-адрес

Set N \u003d 128

Rem Мінлива IPMIN задає початковий IP-адреса. (Повна адреса складається з IPTMP IPMIN)

Set / A IPMIN \u003d 1

REM результати будуть записані в файл oprosipplus.txt

ECHO% DATE% Опитування ARP% N% адрес починаючи з%

IPTMP %% IPMIN% >> oprosipplus.txt

Rem M0 - мітка для організації циклу

: M0

Rem Мінлива IPFULL - повне значення поточного IP-адреси, що складається з IPTMP I IPMIN

Set IPFULL \u003d% IPTMP %% IPMIN%

Rem Якщо 'хвіст' більше N - на завершення роботи

IF% IPMIN% GTR% N% GOTO ENDJOB

REM якщо «хвіст» менше N - продовжимо опитування

REM виконаємо пінг і перевіримо наявність запису в ARP-кеш

Ping -n 1% IPFULL%

Arp -a | Find / I '% IPFULL%'

REM Якщо запис є - запишемо цю подію в журнал oprosipplus.txt

If% ERRORLEVEL% \u003d\u003d 0 Echo% IPFULL% >>

OPROSipPLUS.txt

Rem Сформуємо наступні IP-адреси

Set / A IPMIN \u003d% IPMIN% + 1

Rem Перейдемо на виконання наступного кроку

GOTO M0

Rem Завершення роботи

: endjob

Exit

Помилки, що спостерігаються при написанні командних файлів.

Командний файл вручну виконується успішно, але запущений за допомогою планувальника не працює.

Зазвичай, це викликано тим, що ви не враховуєте той факт, що на момент виконання вашого командного файлу змінні середовища можуть бути зовсім іншими, ніж на момент його написання і запуску з командного рядка. Наприклад, в командному файлі використовується під час запуску програми

myprog.exe, що знаходиться в каталозі SCRIPTS на диску D:. Якщо в командному файлі використовується ім'я виконуваного модуля без повного шляху

MYPROG.EXE

І якщо каталог D: \\ SCRIPTS не прописаний в шляхах пошуку (змінна PATH) то модуль MYPROG.EXE може бути знайдений і виконаний тільки якщо поточним каталогом є D: \\ SCRIPTS. Але якщо ви вкажете повний шлях до myprog.exe:

D: \\ SCRIPTS \\ myprog.exe

То програма буде знайдена і виконана в будь-якому випадку.

Крім того, нерідко програма, зазначена в командному файлі використовує для пошуку своїх компонент (dll, ini і т.п.) власний каталог. Але на момент її виконання поточним каталогом може бути будь-який (найчастіше - системний каталог Windows). Природно, компоненти не перебувають і програма не виконується. Для усунення проблеми додайте в командний файл команди, що забезпечують перехід в потрібний каталог. Наприклад, програма myprog.exe повинна виконуватися в каталозі D: \\ SCRIPTS:

Rem Змінимо поточний диск D:

Rem перейдемо в каталог SCRIPTS

CD D: \\ SCRIPTS

Myprog.exe

Також для переходів по каталогам можна скористатися командами pushd і popd, опис і приклади використання яких є в розділі сайту зі списком команд Windows.

Неправильно відображаються російські імена файлів, служб і т.п.

Причина в тому, що при створенні командних файлів ви використовували текстовий редактор, в якому російські символи представлені не в DOS-кодуванні. Якщо в наведеному вище прикладі перезапуску служби 'DNS-клієнт' ви використовуєте невірну кодування, то російська частина імені служби не

буде визначена через невірну кодування і буде видано повідомлення, що зазначена служба не встановлена. Щоб уникнути проблем з російськими символами в командних файлах, використовуйте редактор з підтримкою DOS-кодування, наприклад, вбудований редактор файлового менеджера Far Manager. Перемикання між кодуваннями в редакторі здійснюється натисканням F8. За допомогою FAR можна легко здійснювати перекодування, скопіювавши (вирізавши) текст в буфер обміну, потім натиснувши F8 і вставивши текст з буфера.

Командний файл виконується на одному комп'ютері успішно, але на іншому - не працює.

Це може бути викликано застосуванням в командних файлах абсолютних значень для дисків, файлів і каталогів замість змінних середовища оточення. Замість C: \\ WINDOWS правильніше використовувати% SYSTEMROOT%, тому, що на іншому комп'ютері система може бути встановлена в інший каталог або на інший диск. Намагайтеся замість імені командного файлу використовувати змінну% 0 і її підстановочні варіанти (% ~ d0 - диск з якого запущений сценарій,% ~ dp0 - повний шлях і т.д.).

Рядки зі змінними, які приймають значення імен файлів і каталогів краще брати в лапки. Командний рядок

DIR% ProgramFiles%

Не видасть вам вмісту каталогу C: \\ Program Files, оскільки через наявність пробілу буде інтерпретована як

DIR C: \\ Program

Командний рядок

DIR '% ProgramFiles%'

Виконається вірно.

Намагайтеся використовувати команди **Setlocal і Endlocal**, щоб не залишати сміття з змінних, створених або модифікованих командним файлом.

Використання командних файлів в сценаріях реєстрації користувачів.

Командні файли зручно використовувати для виконання будь-яких дій при реєстрації користувача в домені. Робиться це за допомогою вкладки Profile властивостей користувача домену.

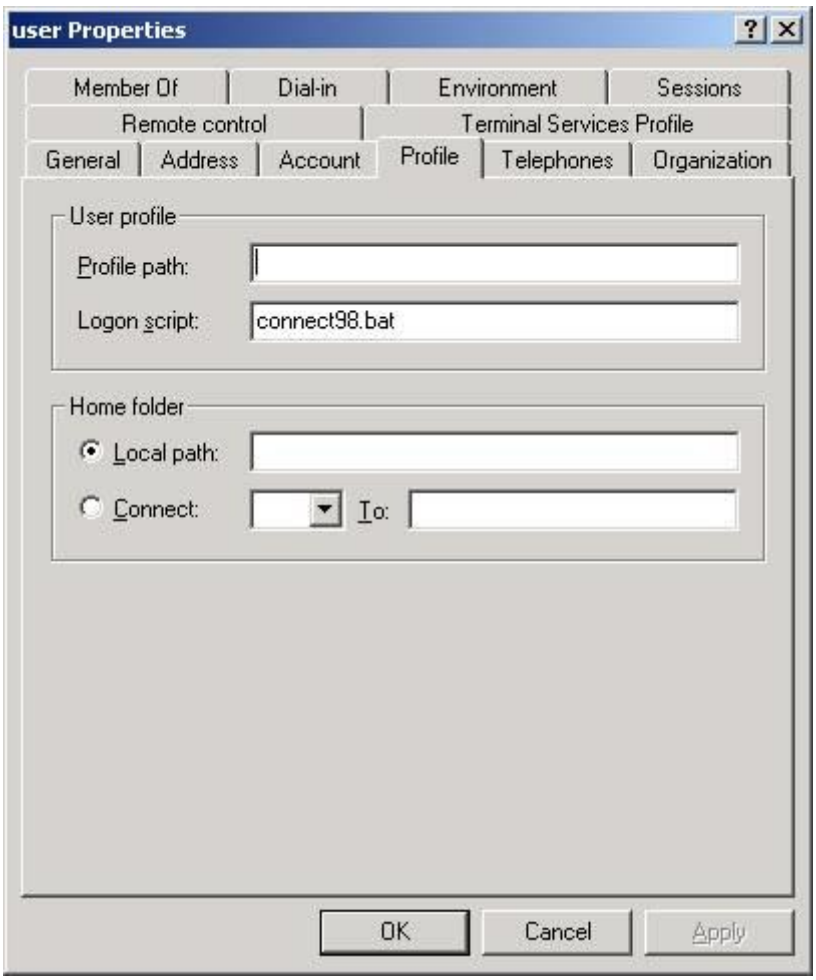

Рис.1 Реєстрація користувачів

Самі командні файли повинні знаходитися в мережевій теці **Netlogon (WINDOWS \\ SYSVOL \\ DOMAIN \\ SCRIPTS)** контролера домену.

Однак найбільш поширеною областю застосування командних файлів є їх використання в сценаріях групових політик, що дозволяють централізовано виконувати адміністративні дії з управління великою кількістю комп'ютерів і користувачів в домені, що значно полегшує роботу системних адміністраторів великих комп'ютерних мереж.

ВИСНОВОК ДО РОЗДІЛУ 1

У цьому розділі ми дізнались про базові команди якими оперує командний рядок операційної системи, їх поєднання у командні файли, цілі та методи

використання. Насправді, в першому розділі ми проаналізували первинний метод створення скриптів на “первісному рівні”, так як далі я опишу налаштування конфігурацій більш складних систем віртуалізації, та інструменту управління доменною структурою. В цьому розділі було проаналізовано достатньо-необхідний обсяг команд та їх алгоритмів для управління з точки зору користувача системи або адміністратора з обмеженим доступом. Ціль розділу була описати призначення скриптів, їх типи і тому була обрані самі прості і маловикористовуванні в нинішній час скрипти які запускаються у стандартній командній оболонці, хоча я можу бути впевненим що їх актуальність не зникне. Інструментарій командної оболонки достатньо широкий, тому іноді ми не можемо не оперувати стандартною командною оболонкою,задля економією часу. З власного досвіду можу сказати, що більш сучасні командні оболонки як power shell блокуються в цілях безпеки . Стандартна оболонка має безпосередній доступ до ядра і не представляє значної безпеки в межах домену поскільки, як раніше було сказано – у неї обмежений функціонал. В даному розділі ми розглядали командну оболонку операційної системи Windows, котра порівнюючи до ОС LINUX має обмежений доступ до ядра.

В другому розділі ми розглянемо модель IT інфраструктури та її елементів .

РОЗДІЛ 2.

ІТ інфраструктура

2.1 ІТ-інфраструктура - це складна багатокомпонентна інтегрована система, яка є комплексом інформаційних технологій (програмних і апаратних засобів) і забезпечує діяльність організації. Комп'ютерне обладнання, програмне забезпечення, мережеві служби, сервіси, електронна пошта, моніторингові системи, політики інформаційної безпеки, системи контролю, системи резервного копіювання та зберігання даних, оргтехніка, телефонія і т.д. - все це складові ІТ-інфраструктури підприємства.

Залежно від бізнес-моделі суб'єктагосподарювання і розмірів компанії ІТ-інфраструктура може бути дуже різною. На сьогоднішній день існує велика кількість різних технологій і рішень від різних виробників. Їх вибір для побудови ІТ-інфраструктури повинен грунтуватися на вирішенні головного завдання ІТ-інфраструктури - відповідати потребам бізнесу, забезпечувати безперервність бізнес-процесів, доступність і безпеку даних.

Інфраструктурні рішення

- Комплексні рішення, системна інтеграція
- Віртуалізація
- Проектування і монтаж СКС (ЛВС)
- Зберігання та доступність даних
- Доменна служба Active Directory
- Електронна пошта
- Термінальний сервер
- IP-телефонія
- Впровадження систем 1С
- Система моніторингу ІТ-інфраструктури та оповіщення
- Бездротові мережі Wi-Fi
- Системи управління ІТ-інфраструктурою, ITSM

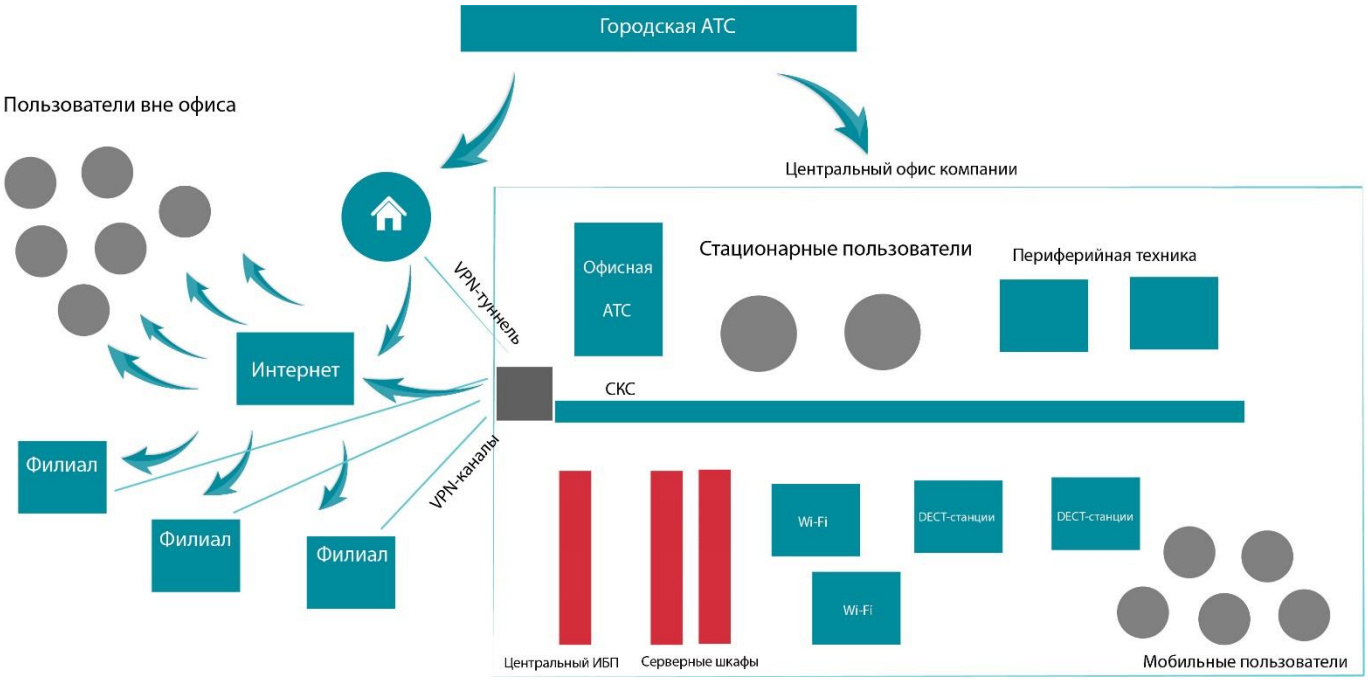

Рис.2.1 Приклад схеми IT-інфраструктури:

Організація роботи IT сфери - це не просто певні IT рішення. Це сукупність великих інтегрованих систем, які забезпечують діяльність фірми в цілому. IT інфраструктура вимагає не тільки правильної і грамотної експлуатації. Інфраструктура IT - це набір систем, програмного забезпечення, якими не вийде управляти як стандартними автоматизованими процесами. У той же час, після того, як проводиться установка всіх систем інфраструктури IT, пускати на самоплив процеси не можна. Саме для цього розробляється IT-стратегія. Це комплекс систем. Він включають в себе пріоритети, завдання та плани, завдяки яким можна домогтися того, що IT інфраструктура відповідатиме рішенням, які представляють бізнес.

Документація на IT-інфраструктуру

Управління, настройка і реалізація вимагає отримання найбільш повної інформації, представленої базами даних. Така діяльність важлива не тільки для керівника, але і для підлеглих, IT директора, керівників бізнес відділів.

До документації, необхідної для роботи підприємства, варто віднести:

- План IT-стратегії;
- Корпоративний тезаурус;
- Стандарти організації IT;
- Опис процесів, регламентів, SLA;
- Схема інформаційних потоків;

Інформаційні ресурси IT-підрозділу

Сюди входить конфігураційна база і електронна бібліотека документації. Перша складова включає в себе бази всіх значущих компонентів. До цього напрямку можна віднести включення серверів, основних робочих місць, системних і мережевих сервісів (наприклад, Microsoft Exchange). А також автоматизовані системи, підсистеми, ролі користувачів.

Дуже важливо організувати доступ всіх зацікавлених користувачів до електронної бібліотеки документації. Доступ проводиться так, щоб було видно:

- Ясність структури документації;
- Можливість швидкого знаходження необхідного документа;
- Впевненість в актуальності знайденої інформації.

Саме тому найчастіше створюють електронну бібліотеку в корпоративній мережі.

IT-інфраструктура: просто про головне

Інфраструктура IT підприємства - це набір засобів, необхідний для роботи додатків, які використовує бізнес. Це може бути електронна пошта, білінг, банківські перекази. Для зростаючого підприємства важлива наявність власної IT сфери, а малий бізнес може користуватися готовим продуктом, тобто це може бути вже існуюча схема, якою користуються на правах оренд

Планування IT-інфраструктури

Для того, щоб приступити до планування майбутньої IT-інфраструктури необхідно:

- - Провести аналіз бізнес-процесів організації;
- - Провести аудит IT-інфраструктури (якщо планується модернізація IT-інфраструктури або міграція на нову);
- - Провести аналіз доступних на ринку рішень, продуктів, технологій і оцінити вартість їх володіння (витрати на придбання, експлуатацію, обслуговування);
- - Розрахувати бюджети і співвіднести можливості з потребами;

Результатом етапу планування IT-інфраструктури є затверджена цільова архітектура, яка відповідає потребам бізнесу як з точки зору ефективності, так і за економічними показниками.

По закінченню планування IT-інфраструктури розробляється технічне завдання для виконавців по впровадженню, яке описує:

- -специфікацію необхідного обладнання та програмного забезпечення;
- -технології і рішення, які будуть використовуватися, їх застосування;
- -топологію мережі, схему мережі;
- -Опис загальних параметрів ключових компонентів IT-інфраструктури;
- -Опис необхідних робіт і їх обсягу;
- -програма і методику випробувань (тестування);
- -здавання IT-інфраструктури в експлуатацію.

Етапи створення IT-інфраструктури компанії:

В першу чергу проводиться розробка, а потім - затвердження технічного завдання. Саме технічне завдання є документацію, в якій представлені вимоги замовника щодо майбутньої інформаційної системи. Далі розробляється робочий проект, потім він реалізовується. Після цього проводиться забезпечення виконавчої документації. На етапі впровадження IT-інфраструктури підприємства виконується

- Забезпечення мережі інфраструктури;
- Розташування автоматичної телефонної станції;
- Поставку обладнання, програмного забезпечення, після чого проводиться його настройка;
- Включення систем для візуалізації, зазвичай це Microsoft Hyper-V, VMware vSphere 5;
- Впровадження мережевої служби (або декількох) по протоколу TCP / IP;
- Реалізація Windows, служби каталогів, розгортання файлових серверів і т. д.

IT-інфраструктура: створення, впровадження, обслуговування

Уявити бізнес без використання IT практично неможливо. Бізнес часто має велику кількість автоматизованих процесів, вони надійні і якісні. Але при цьому кожен з них вимагає того, щоб проводилося ефективне управління, впровадження програмного забезпечення, щоб кожна схема працювала передбачувано і чітко. Створення IT інфраструктури призводить до зниження витрат на операційні роботи.

Також виконуються всі необхідні роботи, які забезпечують оптимальний рівень захисту всіх наявних систем в компанії, в тому числі заходи стосуються мережевої безпеки. Додатково детально прописуються цілі, яких досягають за допомогою інформаційних технологій. Мінімізуються ризики, які можуть бути пов'язані з технічною складовою роботи організації, раціоналізуються рішення, що забезпечують скорочення витрат на придбання програмного забезпечення та реалізацію мережі всередині підприємства.

Що стосується **впровадження IT інфраструктури**, то тут існують такі компоненти:

- Системний комплекс управління базами даних, функції безпеки за рахунок резервного копіювання та інших важливих дій;
- Мережі, що дозволяють передавати сукупності інформації корпоративного напрямку;
- Операції для периферійного обладнання та обчислювальних ресурсів;
- Заходи для впровадження програмного забезпечення, що обслуговує всі системи, спрямовані на стабільну діяльність вашого підприємства.

Проведення обслуговування ІТ інфраструктури також є важливим етапом життєдіяльності, куди входять всі необхідні дії. Це можуть бути клієнтські бази даних, серверна частина, різні варіанти периферійного обладнання. Також виконується обслуговування програмної частини систем контролю доступу, кабельних мереж, АТС, відеоспостереження і т. д.

Навіщо потрібна ІТ-інфраструктура?

В першу чергу такий напрямок дозволяє сформувати цілі та завдання, які дозволяють вирішувати будь-які проблеми, пов'язані з бізнесом. Також за допомогою інформаційних технологій може вирішити довгострокові завдання, поставлені напрямком діяльності компанії і її ІТ-стратегією. Розвиток цього напрямок є необхідністю для кожного співробітника компанії, а не тільки для керівництва і спеціального ІТ відділу. Навіть банальна відправка електронного листа - це сфера, яка відноситься до мережі інформаційних технологій, тому нехтувати проектуванням і плануванням цієї сфери не можна.

Етапи впровадження ІТ-інфраструктури для офісу: від створення до обслуговування

Перед тим як проводити впровадження ІТ інфраструктури в організації, необхідно ретельно перевірити поточний стан справ і провести аудит. Такі заходи необхідні для того, щоб дізнатися все про наявну навантаженні, а також для збору відомостей по всім процесам, які протікають в компанії. Далі проводяться операції, які дозволяють з'ясувати необхідність покупки нового обладнання, його характеристики, кількість, вартість ліцензій ПО, розробляються методи, що поєднують систему воєдино.

Співробітники організації бачать зміни тільки на етапі впровадження ІТ інфраструктури. Це нова техніка, комп'ютери, ПЗ, всілякі навчальні програми, що стосуються складних питань.

Склад інформаційної інфраструктури підприємства

До складу інформаційної ІТ-інфраструктури входять такі компоненти, як:

- Комунікаційне середовище, включаючи провідні та безпровідні мережі;
- Програмне забезпечення;
- Все необхідне для роботи системи обладнання.
- Устаткування для ІТ-інфраструктури є весь комплекс серверів, робочих станцій, систем зберігання даних, мережевого устаткування, периферійних пристроїв та обчислювальної техніки.

Експлуатація та модернізація ІТ-інфраструктури

В процесі експлуатації технічні фахівці замовника або виконавця стежать за станом ІТ-інфраструктури, проводять планово-профілактичні роботи з обладнанням і програмним забезпеченням, виконують регламентні операції, збирають і аналізують зворотний зв'язок від користувачів організації і керівників по використанню.

Якщо в процесі експлуатації IT-інфраструктури в організації виникають нові бізнес-процеси, змінюються наявні, організація змінюється, розвиваємося - може виникнути потреба в модернізації IT-інфраструктури.

Під модернізацією може вважатись практично будь-які зміни IT-інфраструктури, мета яких підвищити доступність, безпеку і ефективність її використання:

- -Нарощування потужностей у зв'язку з розвитком організації (придбання комп'ютерів, серверів, ліцензій, дисків, пам'яті і т.д.);
- -використання нових систем, служб, сервісів в діючу інфраструктуру в зв'язку зі змінами потреб бізнесу (засоби колективної роботи, CRM, ERP, система документообігу, двофакторна аутентифікація і т.д.);
- -використання засобів захисту інформації в зв'язку зі змінами законодавства або появою нових напрямків бізнесу (необхідність забезпечувати збереження персональних даних, банківської таємниця, державної таємниця і т.д.).

Ефективне управління інфраструктурою (ITSM), моніторинг і аудит

В епоху глобалізації бізнес постійно піддається впливам ззовні, проекти і системи стають все більш складними, при цьому час на їх реалізацію скорочується. Для оптимізації багатьох процесів існують різні IT-рішення, проте більшість з них підвищує вимоги до фахівців IT-служби. Процесний же підхід не тільки покращує якість сервісів, а й знижує витрати. Абревіатура ITSM розшифровується як Information Technology Service Management, або «управління IT-послугами». Таке управління передбачає сервісний підхід до роботи IT-служби, коли фахівці IT-департаменту надають іншим відділам компанії послуги відповідно до угоди про рівень послуг. Що потрібно зробити для організації такого підходу, описано в ITIL - бібліотеці інфраструктури інформаційних технологій. Управління IT-інфраструктурою підприємства: складові успіху У бібліотеці ITIL зберігається набір документів, які використовуються для практичного впровадження принципів ITSM. Ідея бібліотеки ITIL з'явилася ще в 1980 р з ініціативи британського уряду. Робота над нею велася з 1986 по 1989 рр. Перша редакція була випущена в 1992 р На основі ITIL був розроблений міжнародний стандарт ISO 20000 для управління і обслуговування IT-сервісів. ITIL описує такі процеси, як: Управління проблемами і інцидентами. Інциденти - будь-які ситуації, які вимагають реакції. Це можуть бути запити від користувачів, збої в системі. Для найбільш успішної реалізації даного процесу, завдання якого виявити і усунути проблеми всередині компанії, мінімізувати ризик їх виникнення, організується спеціальна служба - Service Desk. Управління конфігураціями. Цей процес допомагає отримувати достовірну та актуальну інформацію про IT-інфраструктурі. Управління змінами. Мета процесу - допустити тільки потрібні і розумні зміни. Управління релізами. Це, власне, реалізація змін і контроль збереження IT-інфраструктури при їх впровадженні. Управління рівнем сервісу. Завдання -

виявити оптимальний рівень сервісу, не допускати падіння якості послуг, усуваючи неякісні послуги. Управління фінансами. Забезпечує підтримку фінансовим бізнес-процесів. Управління потужністю. Мета процесу - знайти оптимальну потужністю для реалізації основних завдань. Якщо потужність маленька, то швидкості не вистачає, що, в свою чергу, гальмує роботу. Якщо потужність занадто велика - вона не використовується, а це даремно витрачені гроші. Управління безперервністю. У разі виникнення надзвичайних ситуацій IT-інфраструктура повинна продовжувати роботу. До таких ситуацій відноситься пожежа, відключення живлення, повінь та ін. Управління доступністю. Доступність безпосередньо впливає на рівень сервісу. Відповідно до стандарту ISO / IEC 20000 «Інформаційна технологія. Менеджмент послуг », всі процеси зібрані в п'ять ключових груп: Надання сервісів (управління рівнем сервісу, управління доступністю і безперервністю, управління потужністю, а також управління інформаційною безпекою, бюджет і облік витрат); Управління взаємодією (взаємодія з бізнесом, з постачальниками і т.д.); Процеси дозволу (управління проблемами і інцидентами); Контроль (управління змінами та конфігураціями); Управління релізами. У 2010 р міжнародний стандарт ISO / IEC 20000: 2005 в Росії був переведений в розряд національного - ГОСТ Р ISO / IEC 20000-2010 «Інформаційна технологія. Менеджмент послуг ». Крім ITIL і ISO / IEC 20000: 2005 існують авторські адаптації ITIL - MOF, HP IT Service Management.

Етапи побудови ITSM-системи

1. **Аудит системи управління і планування (обстеження IT-процесів, структури підприємства та IT-інфраструктури).**

Щоб вирішувати проблеми, потрібно для початку виявити їх, тому ITSM-проект завжди починається з аудиту. На цьому етапі відбувається аналіз усіх процесів і виявляється їх стан на даний момент, виконується обстеження IT-інфраструктури. Проводиться аналіз продуктивності всіх підсистем, виявлення «вузьких місць» в бізнес-процесах, інвентаризація програмного забезпечення і ін. Під час аудиту всі IT-процеси оцінюються також і з точки зору відповідності потребам організації. Для кожного процесу визначається поточний і цільовий рівень зрілості. На підставі цих висновків і опрацьовуються подальші поліпшення.

2. **Визначення цільової моделі**.

Розробляється індивідуальна концепція розвитку управління IT, описуються вимоги, яким кожен процес повинен відповідати в майбутньому. Це досить трудомістка процедура, зате вона дозволить створити цілісну систему управління IT, яка зможе врахувати і нові можливості, і стратегію бізнесу. Прогнозування результатів роботи дозволяє оптимізувати процеси управління. Концепція розвитку повинна враховувати не тільки процеси і технології (все їх тонкощі, схеми і особливості), але і персонал, який буде брати участь в роботі. План щодо поліпшення послуг допоможе оцінити витрати і прийняти рішення про стратегію розвитку IT-служби.

3. **Оперативне усунення інцидентів і рішення запитів користувачів (зовнішніх і внутрішніх).**

Це організація роботи служби підтримки (Service Desk). Служба підтримки клієнтів компанії і внутрішніх користувачів допомагає домогтися наступних результатів: Чітка регламентація процесу підтримки; Автоматична обробка всіх вступників звернень; Оцінка задоволеності кінцевих користувачів.

4. **Моніторинг IT-інфраструктури (забезпечення контролю над змінами).**

Контроль над змінами в інфраструктурі - ще одне важливе завдання. Для цього проводиться інвентаризація програмно-апаратних засобів і забезпечується автоматизована підтримка актуальної інформації про інфраструктуру. Завдяки моніторингу відбувається оперативне виявлення збоїв, а процес внесення змін в інфраструктуру регламентований. Ще один плюс - підготовка звітів по роботі процесу відбувається автоматично. Таким чином, керівництво постійно отримує інформацію, необхідну для поліпшення роботи та вдосконалення послуг.

5. **Управління процесами планування, розгортання і надання IT-послуг.**

2.3 Система управління сервісами інфраструктури (ITSM)

У певний момент розвитку компанії керівництво IT-підрозділу може зіткнутися з ситуацією, коли рішення інцидентів займає надто багато часу, користувачі виявляються незадоволені наданими послугами, а внутрішня організація роботи перетворюється в організований хаос.

ITSM - це сервісний підхід до управління IT, коли діяльність IT-підрозділу розглядається як перелік послуг, які воно надає іншим відділам відповідно до SLA. Тобто, завдання ITSM зробити так, щоб IT-відділ став повноправним учасником бізнесу і виступав в ролі сервіс-провайдера для підрозділів компанії. У цьому випадку він перестає бути допоміжним елементом, відповідальним тільки за роботу окремих серверів, мереж і додатків.

Повний перехід на сервісну основу дозволить IT-підрозділам будь-якої компанії не тільки перетворитися з витратного підрозділи в центр отримання прибутку, а й пропонувати свої IT-послуги за межами власної організації. Однією з головних складових ITSM є формалізація процесів IT-відділу. Для кожного процесу визначається послідовність виконання робіт, необхідні ресурси і витрати часу, засоби автоматизації і контролю якості. Коли процес визначений можна виміряти його продуктивність, а також співвідношення якість / ціна. При цьому відзначимо, що ITSM-підхід не зачіпає деталі технічного управління процесами, а спрямований на структурування внутрішньої роботи IT-підрозділу.

Реалізація ITSM також включає в себе формалізацію регламенту роботи

співробітників, визначення зон відповідальності і повноважень персоналу, критерії якості роботи і формування механізмів контролю та моніторингу стану процесів, що допомагає підвищити інформаційну безпеку.

Реалізація ITSM - ServiceNow

Сьогодні, згідно зі звітом Gartner, одним з найпопулярніших ITSM-сервісів на ринку є платформа ServiceNow. «IT Гільдія» - офіційний партнер компанії ServiceNow Ltd., світового лідера серед постачальників ПО для управління службою підтримки, досить давно працює з платформою ServiceNow, забезпечуючи інтеграцію, адміністрування і технічний супровід. Тому ми на власному досвіді переконалися, що ServiceNow - це гнучка платформа, що надає великі можливості конфігурації під процеси клієнта.

За останні роки компанія ServiceNow трохи змінила вектор розвитку свого продукту, який тепер виходить за рамки IT. Сервіс дозволяє управляти більшістю бізнес-напрямків. Процеси, які бізнес використовує для управління, дуже схожі на процеси, що відбуваються в IT-відділі. Змінюються лише завдання, однак архітектура межелементних взаємозв'язків і відносини між виконавцем і споживачем залишаються колишніми.

Це дозволяє застосувати модель ITSM для фінансової служби, HR, служби кадрів, маркетингу, а також за допомогою єдиної платформи управляти аналітикою, розробкою, ресурсами, проектами та іншими напрямками бізнесу. А також для типових ITSM-практик, наприклад, обробки запитів користувачів. Далі, ми розглянемо кілька рішень, які дозволяє застосувати SaaS-платформа ServiceNow.

Управління IT і управління ризиками (GRC)

GRC (Governance, Risc, Compliance) - це сукупність процесів і продуктів, що беруть участь у визначенні та досягненні бізнес-цілей і одночасно знижують ризики. GRC в ServiceNow реалізована трьома додатками, повністю інтегрованими в основну платформу.

Перше додаток називається ServiceNow Risk Management і забезпечує централізований процес виявлення, оцінки, реагування та постійного моніторингу корпоративних і IT-ризиків, які можуть негативно вплинути на бізнес. Додаток має графічний інтерфейс для створення профілів і залежностей з метою зіставлення та моделювання ризиків.

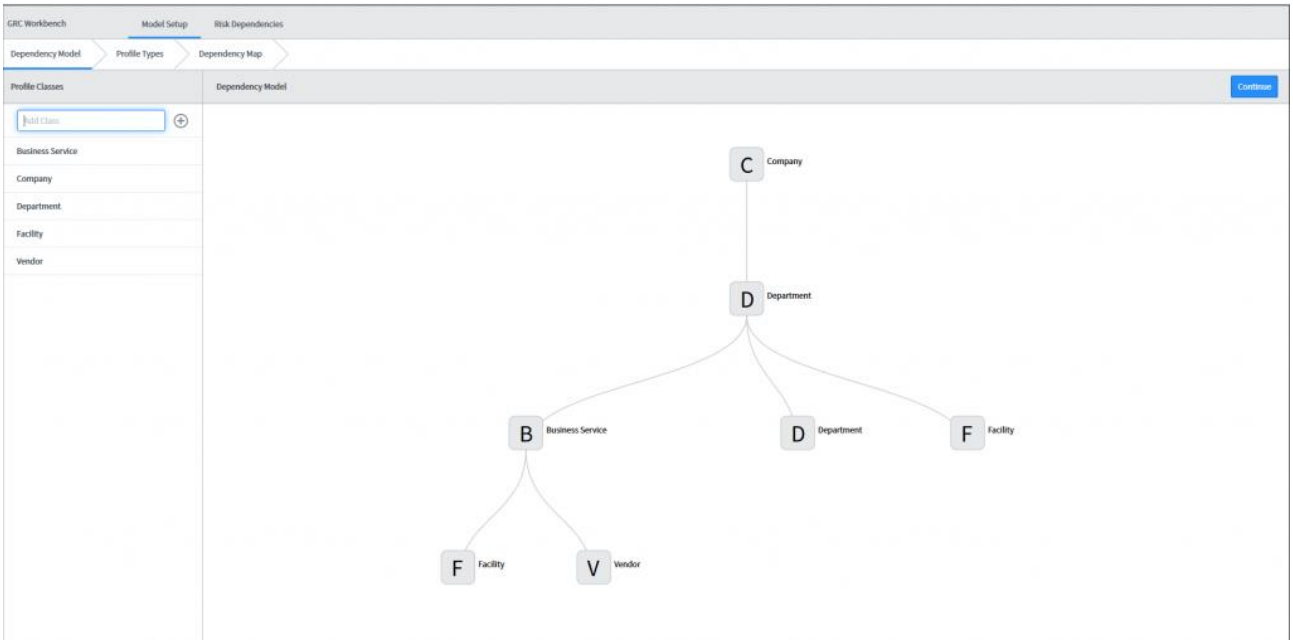

Рис.2.3 Топологія інфраструктури

Воно також дозволяє групувати звіти про ризики в керовані категорії і зберігати всі потенційні ризики в централізованому сховищі, що дає можливість переглядати зведену інформацію і на її підставі швидко визначати проблемні області.

Другий додаток - це Policy and Compliance Management, і воно забезпечує централізований процес створення і управління політиками, стандартами і процедурами внутрішнього контролю. Наприклад, модуль Compliance містить оглядову інформацію про відповідності, а також списки офіційних документів і посилань компанії.

Рис.2.4 Інтерфейс ITSM

В офіційних документах визначаються політики, ризики, елементи управління, аудити та інші процеси. Кожен документ прикріплюється до певного запису, а відповідні списки в запису містять індивідуальні умови цього документа.
Третьою додаток GRC-комплексу - це Audit Management. Воно дає можливість планувати аудиторські завдання, контролювати виконання і надавати звіти відповідальних осіб. Звітність про взаємодію гарантує, що стратегія управління ризиками організації та дотриманням законодавства ефективна. Процес аудиту включає в себе створення, планування, визначення області охоплення і проведення аудиторських завдань, а також звітність про результати.
Таким чином. GRC в ServiceNow, будучи частиною потужної платформи автоматизації бізнесу, допомагає вирішувати велику кількість важливих завдань за рахунок інтеграції з існуючими процесами і використання загальних даних з основними сервісами управління.

Управління розробкою ПЗ (Agile Development)

Платформа ServiceNow також включає в себе додатки для гнучкої розробки. Рішення ServiceNow Agile Development (SDLC) дозволяє управляти методами гнучкої розробки Scrum при роботі над програмним забезпеченням і його супроводом протягом усього життєвого циклу. Додаток Agile Development являє собою послідовний процес для середовища розробки програмного забезпечення.
Платформа дає можливість проводити щоденні наради (Daily Scrum) з членами команди з метою обговорення поточних проектів, запланованих робіт і труднощів. Тут також є можливість вести беклог продукту, що представляє собою список призначених для користувача історій, упорядкованих за ступенем важливості. Product Backlog можна регулярно переглядати і доповнювати в міру появи нових вимог і перегляду пріоритетів.
На додаток до беклогу продукту рішення представляє можливість працювати з беклогом релізів, що містить список історій, виконання яких необхідне для випуску поточної версії розробляється. Як правило, процес прийняття рішень заснований на шкалі часу для релізів, ступенем важливості історій в беклоге продукту і їх складності.

Управління IT-інфраструктурою (IT Operations Management)

IT Operations Management (ITOM) дозволяє впоратися з завданням централізації управління та впорядкування виник хаосу в IT-підрозділах компанії. Основна увага в ITOM приділяється управлінню IT-інфраструктурою, яка формує основу для надання послуг. Програмне забезпечення ITOM допомагає автоматизувати процеси, пов'язані з наданням інфраструктури, розподілом потужностей, управлінням продуктивністю і поточним обслуговуванням всіх елементів IT-інфраструктури.

Основним завданням впровадження ITOM є досягнення наочності і контролю за різними елементами IT-інфраструктури компанії з метою легкого та гнучкого управління. Це дозволяє надавати послуги згідно SLA, використовувати IT-інфраструктуру найбільш ефективно, гнучко реагувати на зовнішні та внутрішні потреби і краще здійснювати обслуговування і підтримку.
ITOM також дозволяє збирати інформацію з кількох джерел, що включають всі основні елементи IT-інфраструктури. Ці дані можуть бути проаналізовані, а потім надані у вигляді звітів про IT-операції за такими параметрами, як навантаження, продуктивність і дії користувачів. Це полегшує раннє виявлення проблем, їх діагностику і швидке усунення.

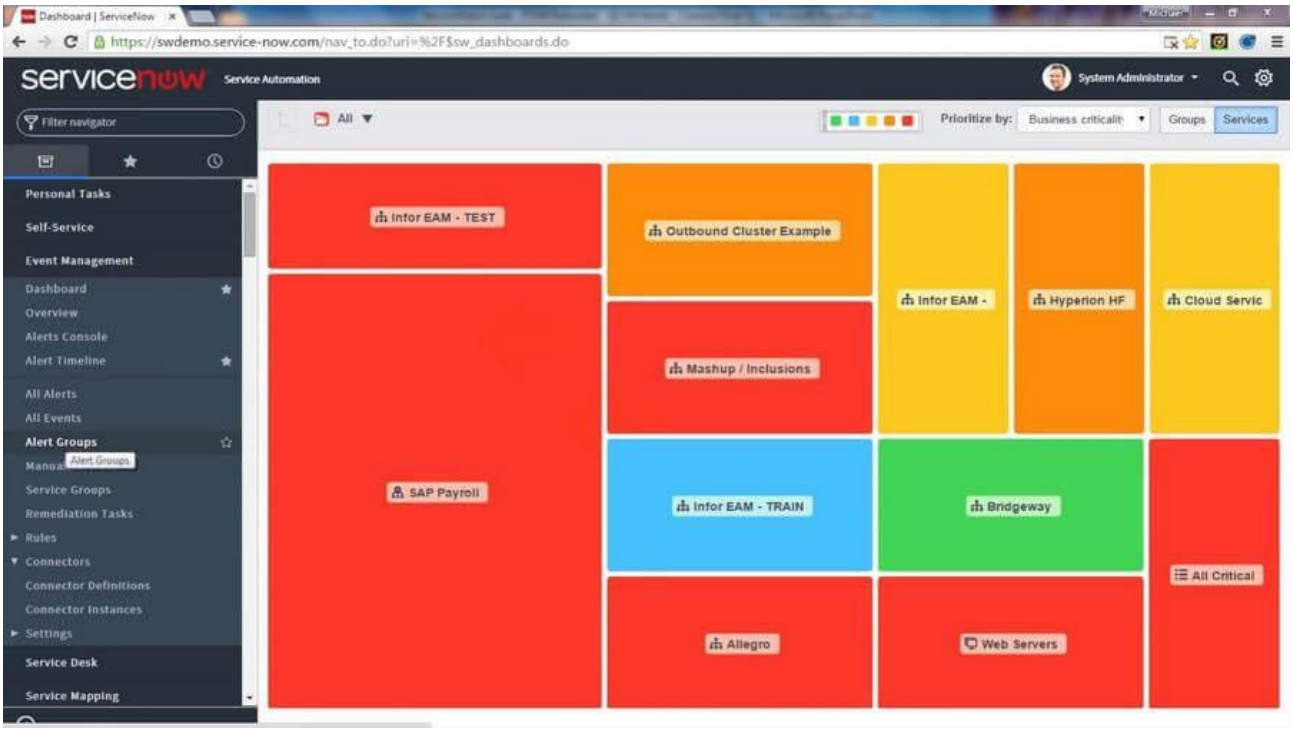

Рис.2.5. Представлення груп в мережі інфрастуктури.

Застосування ITOM робить прозорим використання інвестицій в IT, дозволяє гнучко виділяти IT-ресурси, оперативно виявляти проблеми і вирішувати їх, ефективно здійснювати обслуговування інфраструктури.
І це лише кілька функцій, які реалізує платформа ServiceNow. У списку послуг, що надаються можна знайти такі пункти, як управління фінансами , управління конфігураціями , управління портфелем проектів та інші.

2.4 КОНТРОЛЛЕР ДОМЕНУ

Контроллер домену створити надійний фундамент, розвивати взаємовигідні відносини з клієнтами, підтримувати високий рівень послуг. Основним елементом ефективної корпоративної мережі є контролер домену Active Directory, керуючий багатьма службами і дає багато переваг. Є два шляхи побудови IT-інфраструктури - стандартний і випадковий, коли

прикладаються мінімально достатні зусилля для вирішення виникаючих завдань, без побудови чіткої і надійної інфраструктури. Наприклад, побудова тимчасової мережі у всій організації і відкриття загального доступу до всіх потрібних файлів і папок, без можливості контролю дій користувачів. Очевидно, цей шлях небажаний, так як в кінці кінців доведеться розбирати і правильно організовувати хаотичне нагромадження систем, інакше воно не зможе функціонувати - і ваш бізнес разом з ним. Тому, чим раніше ви приймете єдине правильне рішення будувати корпоративну мережу з контролером домену - тим краще для вашого бізнесу в довгостроковій перспективі. І ось чому.

"Домен - базова одиниця IT-інфраструктури на базі ОС сімейства Windows, логічне і фізичне об'єднання серверів, комп'ютерів, обладнання та облікових записів користувачів."

Контролер домену (КД) - окремий сервер з ОС Windows Server, на якому запущені служби Active Directory, що роблять можливою роботу великої кількості ПО, що вимагає КД для адміністрування. Прикладами такого ПО є поштовий сервер Exchange, хмарний пакет Office 365 і інші програмні середовища корпоративного рівня від компанії Microsoft.

Крім забезпечення коректної роботи цих платформ, КД дає бізнесу і організаціям наступні переваги:

- Розгортання термінального серверу . Термінальний сервер в хмарі дозволяє значно заощадити ресурси і зусилля, замінюючи постійне оновлення офісних ПК одноразової інвестицією в розміщення "тонких клієнтів" для підключення до потужного хмарного сервера.
- Підвищена безпека . КД дозволяє задавати політики створення паролів і змушувати користувачів застосовувати більш складні паролі ніж дата свого народження, qwerty або 12345.
- Централізований контроль прав доступу . Замість ручного оновлення паролів на кожному комп'ютері окремо, адміністратор КД може централізовано змінити всі паролі за одну операцію з одного комп'ютера.
- Централізоване управління груповими політиками . Засоби Active Directory дозволяють створити групові політики і задати права доступу до файлів, папок та інші ресурси мережі для певних груп користувачів. Це в рази спрощує настройку облікових записів нових користувачів або зміна параметрів існуючого режиму.
- Наскрізний вхід . Active Directory підтримує наскрізний вхід, коли при введенні свого логіна і пароля для домену, користувач автоматично підключається до всіх інших службам типу пошти і Office 365.

- Створення шаблонів налаштування комп'ютерів . Налаштування кожного окремого комп'ютера при додаванні його в корпоративну мережу може бути автоматизована за допомогою шаблонів. Наприклад, за допомогою спеціальних правил можуть бути централізовано відключені CD-приводи або USB-порти, закриті певні мережеві порти і так далі. Таким чином, замість ручного налаштування нової робочої станції, адміністратор просто включає її в певну групу, і всі правила для цієї групи застосовується автоматично.

Коли впроваджувати контролер домену ACTIVE DIRECTORY в Корпоративну мережу?

Рекомендується задуматися про те, щоб налаштувати контролер домену для вашої компанії вже при підключенні до мережі понад 10 комп'ютерів, так як набагато легше поставити необхідні політики для 10 машин, ніж для 50. Крім того, так як цей сервер не виконує особливо ресурсномістких завдань, на цю роль цілком може підійти потужний настільний комп'ютер.

Однак важливо пам'ятати, що на цьому сервері зберігатимуться паролі доступу до мережевих ресурсів і база даних користувачів домену, схема прав і групових політик користувачів. Необхідно розгорнути резервний сервер з постійним копіюванням даних для забезпечення безперервності роботи контролера домену, а це зробити набагато швидше, простіше і надійніше використовуючи серверну віртуалізацію, що надається при розміщенні корпоративної мережі в хмарі. Це дозволяє уникнути наступних проблем:

- Помилкові настройки DNS-сервера , що призводить до помилок локації ресурсів в корпоративній мережі і в мережі Інтернет
- Некоректно налаштовані групи безпеки , що призводять до помилок прав доступу користувачів до мережевих ресурсів
- Некоректні версії ОС . Кожна версія Active Directory підтримує певні версії десктопних ОС Windows для тонких клієнтів
- Відсутність або неправильне налаштування автоматичного копіювання даних на резервний контролер домену.

Рис 2.6 Функціональні елементи контролеру домену.

Як бачимо,налаштування контроллеру домена Active Directory несе численні зручності та переваги для бізнесу та організацій будь-якого розміру.

ВИСНОВОК ДО РОЗДІЛУ 2

Було розглянуто концепцію існування та створення IT інфраструктури,етапи, планування і методи забезпечення економічності в умовах розвитку бізнес процесів. Було вивчено поняття ITSM і її роль в корпоративному середовищі. Було розглянуто переваги контролеру домену , в умовах великої к-сті співробітників в компанії.

В умовах стрімкого розвитку бізнесу стоїть задача систематизувати процеси, створити ієрархію , зобразити її , керувати нею. Всі ці данні нам може віалізувати ITSM система. Здавалось би раніше на заводі все зображали на папері та вели облік процесів там же, але в час розвитку технологій інформація -як корпоративна таємниця не повинна потрапити до рук зловмисників, для цього створюється модель контролів доступу (ієрархія доступу) – як при розробці так і при інцидентах. Інциденти в стабільно працюючій і налагодженій системі відбуваються нечасто. Система ITSM дозволяє аналізувати їх і зрозуміти корінь проблеми. Контроллер Домену потрібен для практичного керування правами доступу і створення груп. Існує безліч інтеграцій за допомою API з ITSM тому в цьому розділі я вирішив зробити акцент саме на данних технологіях.

В наступному розділі ми будемо імплементувати та керувати данними системи за допомогою скриптів.

РОЗДІЛ 3

АВТОМАТИЗАЦІЯ ЗАДАЧ В КОРПОРАТИВНОМУ СЕРЕДОВИЩІ

3.1 Рішення адміністративних задач

Запитайте будь-якого адміністратора Windows про його проблеми, і у верхній частині списку буде велика кількість роботи і постійний брак часу щоб зробити її. Вони знають про засоби автоматизації, можливо навіть обізнані про можливості WMI і Powershell, але у них немає часу щоб освоїти ці технології. Це ганебна ситуація, тому що прийнято вважати, що до 70% бюджету IT організації витрачається на те «щоб все було в робочому стані» (прим перекладача: в оригіналі 'keep the lights on.'). Автоматизація може скоротити витрати з цих 70%, за рахунок чого звільняться час і гроші для задач .
Крім того, можливо, що вони цікавилися WMI або Powershell і прийшли до висновку, що ці інструменти дуже складні для освоєння. Це зрозуміло, особливо з огляду на трудомісткість роботи з WMI через VBScript, а також відсутність прикладів, які пояснювали б використовувані прийоми. Навіть мене наводять страху деякі приклади скриптів Powershell розміщені в інтернеті, що ж відчувають ті, хто тільки почали розбиратися з цією темою. Адміністратори, відмовившись від використання цих інструментів упускають можливості зменшити свою робоче навантаження, пропускаю можливість провести автоматизацію своїх процесів.
Знизити планку входу до використання WMI ставилося за мету цієї книги. Приклади надаються в ній можуть бути використані без змін або з мінімальним їх кількістю. Крім того, ви отримаєте більш глибоке розуміння WMI, яке може бути використане для роботи з областями раніше недоступними для вашого контролю. PowerShell сконструйований так, щоб зробити використання WMI простіше і зрозуміліше ніж в скриптових мовах раніше, PowerShell движок автоматизації від Microsoft, яка, крім усього іншого, забезпечує полегшений доступ до багатих наборам інструментів управління, доступних в WMI.
Разом PowerShell і WMI надають набір перевірених методів, які дозволять вам керувати вашою системою легко і швидко. Ви зможете автоматизувати багато типових завдань, які часто вимагають занадто багато уваги інженера.

Насамперед, опишемо проблему яка стоїть перед сучасним системним адміністратором. Існує цілий ряд питань, які впливають на будь-яке середовище -Windows, значного розміру

-Збільшення кількості систем

-Підвищується складність моделі інфраструктури

-Зростаюча швидкість змін в інструментах та оновленнях ПО.

Я доведу, чому PowerShell і WMI забезпечують великий набір інструментів для вирішення цих проблем. Отримання результату від використання PowerShell передбачає незначних початкових витрат часу на його вивчення, особливо при використанні WMI. Вивчивши мову і автоматизувавши ваші щоденні процеси ви зможете досягти відмінної віддачі від витрат часу на процес навчання Супроводжується розділ двома прикладами демонстрації потужності цієї комбінації технологій. Перший приклад показує, як можна відключити всі сервери в центрі обробки даних за допомогою однієї команди, а другий показує, як ви можете перевірити налаштування багатьох машин за один прохід.

Почнемо з розгляду обов'язків сучасного адміністратора Windows і проблем з якими він стикається.

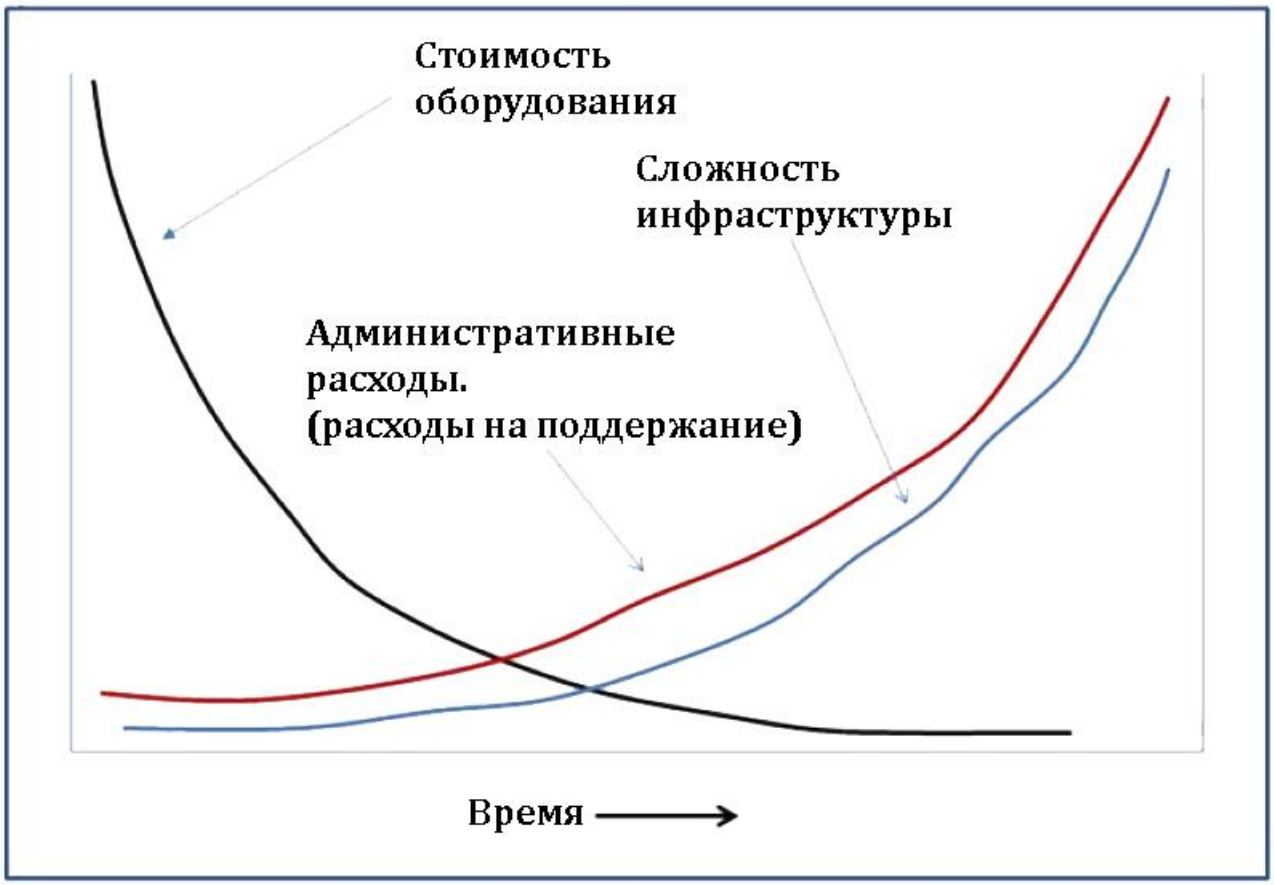

Рис.3.1 Залежність між вартістю та витратами на підтримку обладнання та складністю інфоаструктури

Здешевіння потужності апаратури призводить до двох інших графіків, котрі відображають круту зростання складності інфраструктури і ще швидше зростання адміністративних витрат на підтримку її в робочому стані. Постійне ускладнення інфраструктури та зростання витрат на неї, є на даний момент основною проблемою IT департаментів.

Для скорочення бюджету потрібно зламати криву зростання експлуатаційних витрат і PowerShell з WMI допоможе вам в цьому. Для початку давайте розберемося - звідки з'являється складність і зростання витрат на адміністрування?

Чи дійсно вам потрібен кожен сервер що ви створили? Багато, якщо не більшість організацій мають занадто багато серверів. Це відбувається з ряду причин:

А)Зниження вартості апаратної потужності - це призводить до того що при високому завантаженні простіше докупити новий сервер ніж шукати як оптимізувати наявні.

Б) Самостійні закупи відділів або закупівлі під проекти – цей процес відбувається через питаня приналежності сервера, департаменти або «проектувальники» не хочуть щоб на їх серверах хтось «був присутній і вносив зміни». Вони не бажають надавати іншим свої ресурси.

«Один додаток - один сервер» - поділ додатків так щоб проблема в одного не впливала на іншого, це правило все ще може діяти для критично важливих бізнес-додатків, але це не обов'язково для другого або третього ешелону додатків. І безумовно, це не потрібно для цілей тестування і навчання.

Повільна реакція або ригідність IT відділів - відсутність контролю і занедбаність процесів в IT департаменті призводить до розброду в проектах і безладної зміни систем.

Навантаження на системного адміністратора росте швидше, ніж темп росту машинної потужності через час, необхідний для перемикання уваги між машинами (зазвичай потрібно нове віддалене підключення), додаткову складність Кожна машина і його застосування залучені в життя системного адміністратора. Є кілька переваг для **віртуалізації:**

- Скорочується число фізичних серверів
- Знижується вимога до ЦОД організацій, в тому числі через простору, електроенергії та витрат на кондиціонування
- Більш точно використовуються потужності фізичних активів, що веде до більшої віддачі інвестицій

Організація в цілому отримує вигоду від віртуалізації, але навантаження адміністратора збільшується. Якщо у вас було 100 серверів до віртуалізації, і ви устанановили замість них 4 фізичних хоста і вірутуалізували на них 100

серверів, то у вас вийде 104 системи якими потрібно керувати. Крім того що їх стало на 4 більше, складність збільшиться через те що платформа віртуалізації може ввести іншу операційну систему в інфраструктуру або додати нові інструменти. Збільшення загальної кількості (фізичних плюс віртуальних) систем, також означає, що зростає ймовірність подій, під час еволюції інфраструктури.

Зміни в середовищі

Зміни можна розглядати як найгіршу головну біль адміністратора. На жаль середовища не статичні, зміни відбуваються завжди:

- Операційна система і додатки отримують регулярно патчі
- Виходять нові версії програм
- Простір зберігання потребує перенастроювання для відповідності мінливим типових завдань користувачів.
- Типові сценарії роботи додатків вимагають змін апаратної частини або апаратної модернізації.
- Віртуалізація та інші проривні технології змінюють екосистему і створюють нові можливості і конфігурації.

Ці проблеми накладаються на десятки, сотні або навіть тисячі машин, накладаються на щоденну активність, таку як моніторинг і резервне копіювання.

Ця ситуація не може тривати нескінченно. Компанії не можуть пом'якшувати постійно зростаючі витрати на адміністрування, сьогоднішні економічні реалії забороняють використовувати альтернативні механізми такі як нескінченне збільшення доходів щоб не помічати зростання витрат IT відділів. Ситуація повинна бути вирішена за рахунок скорочення витрат на експлуатацію, але адміністратори не можуть зробити це через те, що безліч змін приносять нові і нові технології не знижують навантаження і так відбувається нескінченно.

Зростаюча складність

Складність - реальна проблема. Вона виникає через низку причин:

- Кілька операційних систем несуть різні набори інструментів і термінологію, різниця є навіть між двома версіями Windows.
- Різні типи додатків, таких як бази даних, електронна пошта, служби каталогів Active Directory, і веб-додатки, вимагають різних навичок, різних інструментів, мають різні вимоги і створюють різне навантаження на сервера.
- Багато машин виконують однакові або схожі ролі, але непомітні особливості в їх реалізації, недокументовані можливості збільшують ймовірність помилки і роблять структуру складніше.

Складність також зростає через неповноту знань і навичок самих адміністраторів. Занадто часто проект вводить нову технологію і від системних адміністраторів очікується що вони відразу ж підхоплять і почнуть керувати системою. У них є необхідні навички? Чи є у них час, щоб вивчити тонкощі вноситься технології? На жаль, відповідь на обидва питання часто негативний. Адміністратор в такій ситуації прийме єдине вірне рішення - він почне робити так як вміє. Іноді, якщо нова технологія є зміною версії від продукту, адміністратор продовжить використовувати старі методи, навіть якщо в новій версії з'явився набагато кращий спосіб виконати завдання.

Відсутність навичок і знань призводить до помилок, і ці помилки коштують грошей, часто їх можна розглядати як втрату доходу для організації. Це ставить під стрес адміністраторів і призводить до недовіри з боку керівництва бізнесу. IT відділ часто виключений з дискусії про впровадження нових технологій, занадто пізно він дізнається про те що керівництво вирішило піти на зміни їхнього середовища, і цикл помилок повториться знову.

Крім того, мало що йдуть великі зміни вносяться проектами, системні адміністратори стикаються з безліччю дрібних змін, необхідних, щоб тримати середовище в безпеці і забезпечувати стабільність роботи.

Автоматизація - шлях для прориву вперед

Рішенням для подолання цих проблем є автоматизація рутинних операцій. Доручити машині робити просту, повторювану роботу це те заради чого ми робили їх!

Автоматизація розуміється по різному. Нижче наведена ієрархія автоматизації.

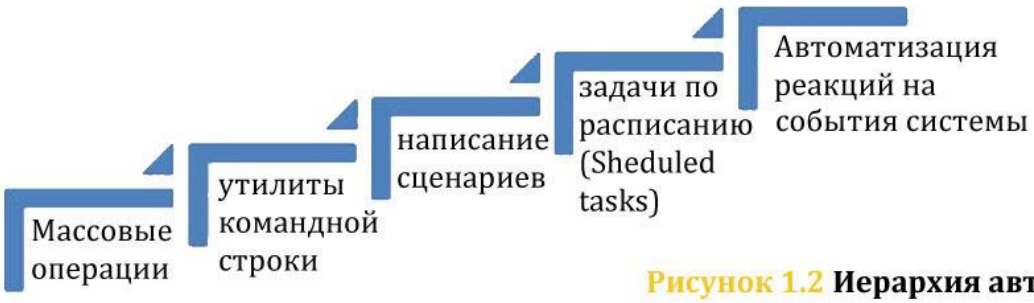

Рис.3.2 Ієрархія автоматизації

Щоб впровадити рівень вище цієї ієрархії потрібно відповісти на питання - «чи буде користь від переходу на більш складний рівень?». Я знаю цілий ряд організацій, яким вистачає можливостей стандартного інструментарій Windows і кілька інструментів масової розсилки команд *(прим. Перекладача: незнаю як точно перевести «a few bulk-editing tools»)*. Інші намагаються максимально використовувати планувальник завдань або навіть створюють

відповіді на події в журналі. Автоматизація для більшості організацій являє собою суміш інструментів командного рядка, сценаріїв, і завдань за розкладом. Наступне питання - «**Як ми можемо автоматизувати мої адміністративні завдання?».** PowerShell надає набір інструментів командного рядка (званих «командлети») які можна використовувати інтерактивно вводячи в консоль. У міру того як складність вирішуваних завдань стає більше і більш амбіційними, відбувається поява скриптів. У PowerShell ви можете використовувати одні команди одне написання і один стиль для командного рядка і при написанні сценаріїв.

PowerShell прекрасний інструмент При необхідності ви можете легко перейти на наступний рівень складності, і так далі, а WMI буде на самому верху цієї східці. *(прим перекладача. Richard Siddaway в книзі «powershell in depth» написав «якщо ви не використовуєте WMI в зв'язці з powershell то втрачаєте та 60% потужності»)* . WMI відкриє Вам доступ до стандартного набору інструментарію управління системою, яку ви зможете використовувати локально або на віддаленій машині, потенційно ви зможете працювати навіть з не Windows системами *(прим. Перекладача він має на увазі CIM)* . Сценарії можуть бути запущені інтерактивно або можуть бути заплановані на час. Але перш ніж ми заглибимося в подробиці нам потрібно оглянути автоматизацію в цілому.

У цьому розділі, ми будемо концентруватися на сценаріях як основному засобі автоматизації. Звичайно, ви могли б зробити багато з командного рядка підключаючись до віддалених машин. Однак, перевага сценаріїв в тому що ви можете використовувати код повторно, кожен раз економлячи час на написання і налагодження коду. Ця тема докладно розглядається в розділі 4 книги «PowerShell in practice» видавництва Manning 2010. Заплановані завдання і «автоматичні реакції на події»[10] занадто сильно залежать від вашої конкретної інфраструктури, в подальшому ми почнемо розглядати як ви можете зробити автоматичні відповіді на події, що відбуваються в вашій системі.

Давайте розглянемо приклад. Припустимо вам потрібно визначити кількість вільного простору на диску C декількох машин у вашому середовищі. Один із способів це прийти в ЦОД, підключиться до кожної машині по черзі і подивитися вільний простір диска C. Записати відповідь і повторити для наступної машини.Трохи простіший варіант - використовувати **RDP** для підключення до кожної машині і вручну вивантажувати інформацію. Таким чином ви не будете виходити з за свого столу. Але ви як і раніше повинні зробити дуже багато маленьких дій, ви як і раніше втрачаєте дуже багато часу. І рішення яке мені подобається - використовувати для цієї мети PowerShell, код

приведений в лістингу нижче. Не хвилюйтеся якщо ви прямо зараз не розумієте його. Ми повернемося до цього сценарію,коли будемо розглядати керування дисковою підсистемою.

Приклад починається з переліку контактів комп'ютер моєї лабораторії. Цей список передається по конвеєру в командлет **ForEach-Object** (foreach) який викликає **Get-WmiObject** для кожного сервера зі списку із запитом даних про логічному диску C. Потім отримана інформація форматується і виводиться у вигляді таблиці

Лістинг 1.1 Визначити вільне місце на диску:

```
"dc02", "W08R2CS01", "W08R2CS02", "W08R2SQL08", "W08R2SQL08A",
"WSS08" | foreach {
Get-WmiObject -Class Win32_LogicalDisk -ComputerName $_ -Filter
"DeviceId='C:'" } |
Format-Table SystemName, @{Name="Free";
Expression={[math]::round($($_.FreeSpace/1GB), 2)}} -auto
```

Вільний простір перераховується з байтів в гігабайти, щоб зробити результати більш зрозумілими. Зверніть увагу що PowerShell розуміє скорочення GB, а також KB, MB, TB і PB.

Результат роботи скрипта виглядає наступним чином:

```
SystemName Free
---------- ----
W08R2CS01 119.04
W08R2CS02 118.65
W08R2SQL08 114.8
W08R2SQL08A 115.17
WSS08 111.41
DC02 118.53
```

Ряд удосконалень можна внести в цей сценарій:

- Помістити імена комп'ютерів в CSV файл (як ми будемо робити в лістингу 1.4)
- Додати результат роботи в Excel, або базу даних щоб можна було бачити тенденцію зміни місця на диску
- Запланувати виконання завдання в планувальнику

У мене працює такий сценарій з двома першими поліпшеннями. Він регулярно повідомляє про місце на дисках, можна подивитися тенденції. Після написання у мене є інструмент який можна запустити за кілька секунд, опитає кожну машину сам і поверне інформацію.

Мені знадобилося всього кілька хвилин щоб написати його і я економлю час коли я запускаю його знову і знову. Саме таким способом PowerShell допомагає

заощадити час. Jeffrey Snover, архітектор PowerShell написав - «Я твердо вірю, що економіка визначає, що люди роблять і що вони не роблять. PowerShell розроблений з нуля, щоб бути розширюваної, високо рівневої, задачі орієнтованої абстракцією, здешевлює витрати на адміністрування і підтримку. »

1.3.3 Зламати криву зростання

На малюнку 1.1. ми бачили, безперервне зростання складності організації і витрат на адміністрування. Це безперервне збільшення складності і витрат рано чи пізно призведе до втрати балансу і зупинки росту через витрати на управління, потрібен спосіб щоб зламати цей ріст.

PowerShell може допомогти зупинити зростання витрат, забезпечуючи наступне:

- надає набір інструментів інтерактивної роботи з сервером і додатками
- Працює у всіх системах Windows *(прим перекладача: в оригіналі застосовано слово estate точний переклад - маєтку, володіння. Мається на увазі, що це core технологія Microsoft і всі системи так чи інакше мають командлети)*
- Надає один універсальний підхід роботи з різними системами (прим перекладача: все командлети одноманітність, не потрібно вивчати список ключів консольної утиліти)
- Вбудовані можливості віддаленого управління
- Вбудовані можливості для подальшого ускладнення автоматизації

PowerShell методи підвищують продуктивність і ефективність. А за допомогою PowerShell і WMI ви можете розраховувати на подальші підвищення вашого зростання контролю над системою.

3.2Автоматизація створення віртуальних машин

Створення нової віртуальної машини - це рутина, що забирає багато часу. І чим більше інфраструктура і організація, тим більше процедур, пов'язаних з цим процесом. Ми автоматизували це процес за допомогою PowerShell.

Існує **ITSM**-система, в даному прикладі це **ServiceNow**, ми створили відповідну web-форму в сервісному каталозі. Для «замовлення» нової машини менеджеру необхідно заповнити поля і підтвердити «замовлення», після цього запускається ланцюжок процесів, і на виході отримуємо готову до використання машину.

Які параметри потрібно вибрати в GUI, щоб створити нову віртуальну машину? :

Рис.3.3 Опції первинного налаштування віртуальної машини

VM Description: опис віртуальної машини

Тут потрібні деякі пояснення. У нашому рішенні активно використовується PowerShell 5.1, тому поки Windows-only, в майбутньому ми постараємося додати підтримку Unix-машин і перейдемо на PowerShell Core.

OS , операційна система. Ніяких особливих перешкод використовувати Windows 2008 (R2) немає, але ми використовуємо 2012R2 або 2016.

VM Size , розмір віртуальної машини. У кожного це може бути визначено по-своєму, в даному прикладі Small 1CPU-4Gb Ram, Medium 2CPU-8Gb, Large 4-16.

VM Storage, Disk 0 (C: \) має фіксований розмір, який ви не можете змінити,

доступний тільки селектор Fast / Slow storage. «Fast» - це може бути Storage Tier з SSD, а «Slow» - це storage на «звичайних» HDD (звичайно - SAN). Disk1 (Disk2 і далі) також мають селектор вибору типу Storage, а також поля для введення бажаного розміру в гигабайтах, Letter для розділу і розмір кластера (що важливо для SQL Server).
Trust , визначаємо, що машина повинна бути Domain-joined чи ні, з доступом з Public Network чи ні.
Type , тип машини. Майже кожну машину можна визначити, як front-end або back-end додатки або ж other у всіх залишилися випадках. На основі обраного типу ми зможемо в подальшому визначити найбільш підходящу підмережа для машини.
Environment, В інфраструктурі замовника є два дата центру: Primary (Production) і Secondary (Dev / test), DC пов'язані між собою швидкому каналом зв'язку і забезпечують відмовостійкість. За домовленістю все віртуальні машини в ***Primary DC*** мають IP-адресів, що починаються на 10.230, а в ***Secondary DC*** - на 10.231.
(SLA) Service Level Agreement , цей параметр впливає на якість обслуговування даної машини.
Додатки . Ми додали можливість установки і налаштування SQL Server. Необхідно вибрати версію, instance name і collation. Також можливо і Web Server роль і багато іншого.
Тепер нам потрібно визначити, як зберігати вибрані значення. Ми вирішили, що найбільш зручний формат - JSON-файл.Як рішення в середовищі використовується ITSM ServiceNow; менеджер, після того як вибрав всі необхідні значення, натискає кнопку «order» і після цього ServiceNow передає всі параметри нашого PowerShell-скрипту (на back-end ServiceNow), який і створить JSON-файл. Виглядає це приблизно так: (**ДОДАТОК В**)
Зазначу, що в веб-формі відсутнє ім'я віртуальної машини і IP-адреса. Отримати ці значення можна автоматично наступним чином:
Ім'я машини , в ITSM ServiceNow є спеціальний розділ: **CMDB** (**Configuration Management Data Base**), в цій базі зберігаються всі записи про існуючі віртуальних машинах, їх статус, команда підтримки та інше. Ми створили близько 200 резервних записів зі статусом Allocated. Щоб отримати ім'я для віртуальної машини ми робимо REST-запит до CMDB і отримуємо першу «вільну» запис і міняємо її статус з Allocated на Pending install.
IP адреса і VLAN, Ми розгорнули **IPAM** в нашій мережі - це вбудована feature в Windows Server 2016, яка дозволяє управляти IP-адресами в вашій мережі. Зовсім не обов'язково використовувати всі можливості IPAM (DHCP, DNS, AD), а використовувати її тільки як базу даних IP-адрес з потенційним розширенням функціоналу. Скрипт, який створює JSON файл, робить запит до IPAM на предмет першого вільного IP адреси в підмережі. А підмережа VLAN (x / 24 підмережа) визначається на основі вибраних значень SLA, Environment, Trust і Type.

Файл-конфігурації готовий, всі поля на місці, можна створювати машину. Постає питання «як зберігати облікові дані для всіх наших скриптів?». Ми використовуємо пакет **CredentialManager** . Цей пакет працює з вбудованим **Windows Credential Manager API** для зберігання паролів. Приклад створення пароля:

New-StoredCredential -Target "ESXi" -UserName "testdomain.eu\vmwareadm" -Password "veryultraP@ssw00rd." -Type Generic -Persist LocalMachine

Пароль буде доступний для читання в межах даної машини і облікового запису.

$ESXiAdmin = Get-StoredCredential -Type Generic -Target ESXi

У нас є сервер, на якому зберігаються всі конфігурації з GIT репозиторію, тепер ми можемо надійно відслідковувати всі зміни в конфігураціях: хто, що, де і коли.

На цьому сервері налаштований scheduled task: перевіряти папку с конфігураціями і писати в Windows Event Log про всі зміни.

Через 15 хвилин scheduled task напише в Windows EventLog, що виявлений новий файл-конфігурація.

Прийшов час перевірити цю конфігурацію. В першу чергу нам потрібно переконатися, що файл має коректне форматування:

$Configuration=(Get-Content -Raw $File | Out-String | ConvertFrom-Json)

Якщо все добре, настав час приступати до створення машини і запуску BuildVM. ps1 скрипт.

У BuildVM.ps1 ми перевіряємо, що файл-конфігурація має опис всіх характерис тик віртуальної машини: size, env, sla, type, storage, ram, network.

Обов'язково перевіримо, чи є в інфраструктурі машина з таким же ім'ям (Check VM.ps1).

Підключаємося через **VMWare PowerShell CLI** д о нашої vSphere:

$VmWareAdmin = Get-StoredCredential -Type Generic -Target ESXi
Connect-VIServer -Server "vSphereSrv" -Credential $VmWareAdmin | Out-Null

Перевіряємо, чи є машина з таким же ім'ям в інфраструктурі

$VM=Get-VM $server -ErrorAction SilentlyContinue

І відключаємося:

*Disconnect-VIServer * -Force -Confirm:$false*

Перевіряємо, машину на недоступність по WinRM

$ping=Test-NetConnection -ComputerName $Configuration.Hostname -CommonTCPPort WINRM -InformationLevel Quiet -ErrorAction SilentlyContinue

Якщо в $ VM і $ ping порожньо, то можна створювати нову віртуальну машину. Обробляються ситуації, коли машина вже створена в ESXi вручну або ж ця маш ина в іншому дата-центрі.Існує підготовлений образ віртуальний машини, який був фіналізовано **sysprep** і сконвертовані в **template в нашому vSphere**.В образ і збережений локальний адміністратор з відомим нам паролем.Данний облікови й запис «не злітає» після sysprep, що дозволить нам отримати доступ до кожної машини з цього шаблона, а пізніше маємо можливість змінити пароль в цілях безпеки.

3.2.1 Створення віртуальної машини

Знайдемо відповідний SLR-кластер:

$Cluster=Get-Cluster -Name $Configuration.VM.SLA

Перевірка наявності достатнього місця на Datastore:

$DatastoreCluster = Get-DatastoreCluster |Where-Object {$_.Name -like $Datastore1Name}
$Datastore1 = Get-Datastore -Location $DatastoreCluster |sort -Property "FreeSpaceGB" |select -Last 1
IF ($Datastore1.FreeSpaceGB -le "200"){
Write-Host -foreground red "STOP: Not enough datastore capacity for DISK" $vdisk.Id
Break
}

Перевіримо наявність достатнього об'єму пам'яті:

$VMHost = Get-VMHost -Location $Cluster |sort -Property "MemoryUsageGB" |select -First 1
IF ($VMHost.MemoryUsageGB -le "20"){
Write-Host -foreground red "STOP: No enough ESXi host capacity"
Break
}

Беремо наш шаблон

$VMTemplate = Get-Template -Name 'Win2016_Std_x64_Template'

Створюємо нову віртуальну машину

New-VM -Name $Configuration.Hostname.ToUpper() -VMHost $VMHost -ResourcePool $ResourcePool -Datastore $Datastore -Template $VMTemplate -Location "AutoDeployed VMs"

Важливо підключити мережевий інтерфейс до підмережі з включеним DHCP.

Запускаємо віртуальну машину

Start-VM $VM

І зберігаємо базовий опис машини(конфігурацію), для подальшого визначення машини на рівні гіпервізора **VMWare**.

Set-Annotation -Entity $VM -CustomAttribute "Change request" -Value $Configuration.Request -Confirm:$false
Set-VM $VM -Notes $Configuration.Description -Confirm:$false

Машина увімкнулась і тепер ми можемо дізнатися отриманий MAC-адресу:

$vMAC = (($VM | Get-NetworkAdapter | Select-Object -Property "MacAddress").MacAddress).Replace(':','')

Збережемо це значення в наш JSON-файл

$Configuration.Network.MAC=$VMAC
ConvertTo-Json -InputObject $Configuration -Depth 100 | Out-File "C:\Configs\TargetNodes\Build\$Hostname.json" -Force

Тут саме час зробити підтвердження (**commit**) в наш Git репозиторій, що машина створена і має свій унікальний MAC.

Машина починає власну ініцалізацію для доступності та видимості її в

середовищі (після sysprep), налаштувавши обладнання і початкову конфігурацію.
Давайте дочекаємося, коли буде доступна наша машина по **WinRM** с скриптом EstablishConnection.ps1.
Дізнаємося яку IP адресу отримала машина від DHCP:

```
#Тут $MAC = $vMAC

while($isOnline -ne $true){

    if((Get-DhcpServerv4Lease -ClientId $MAC -ScopeId $StagingDHCPScope -ComputerName $DHCPServer -ErrorAction Ignore).IPAddress.IPAddressToString){
    $tempIP=(Get-DhcpServerv4Lease -ClientId $MAC -ScopeId $StagingDHCPScope -ComputerName $DHCPServer).IPAddress.IPAddressToString

    break

  }

  else{

    if($isOnline -ne $true){

      Write-Host "`r$i`t" -NoNewline

      $i++

    }

  }
```

Зачекаємо, доки машина буде доступна по **WinRM**:

```
$LocalAdmin = Get-StoredCredential -Type Generic -Target LocalAdmin

$i=0

$isOnline=$false

while($isOnline -ne $true){

  if(Invoke-Command -ComputerName $tempIP -ScriptBlock{ Get-ItemProperty -Path "Registry::\HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Component Based Servicing" } -Credential $LocalAdmin -ErrorAction SilentlyContinue){

    $isOnline=$true

    break

  }

  else{

    if($isOnline -ne $true){
```

```
        Write-Host "`r$i" -NoNewline
        $i++
        Start-Sleep -Seconds 1
    }
}
```

Проміжний результат:-машина готова до управління.

Тепер створюємо **конфігурацію бажаного стану (DSC)** Для налаштування бажаної конфігурації ми використовуємо частину **PowerShell - DSC (Desired State Configuration). У мережі є налаштований DSC Pull Server: dscpull.testdomain.eu.**

Нижче конфігурація нашого DSC Pull Server.

```
Node $NodeName
{
    WindowsFeature DSCServiceFeature
    {
        Ensure = "Present"
        Name   = "DSC-Service"
    }

    xDscWebService PSDSCPullServer
    {
        Ensure                  = "Present"
        EndpointName            = "PSDSCPullServer"
        Port                    = 8080
        PhysicalPath            = "$env:SystemDrive\inetpub\PSDSCPullServer"
        CertificateThumbPrint   = $certificateThumbPrint
        ModulePath              =
"$env:PROGRAMFILES\WindowsPowerShell\DscService\Modules"
        ConfigurationPath       =
"$env:PROGRAMFILES\WindowsPowerShell\DscService\Configuration"
        State                   = "Started"
        DependsOn               = "[WindowsFeature]DSCServiceFeature"
        RegistrationKeyPath     =
"$env:PROGRAMFILES\WindowsPowerShell\DscService"
        AcceptSelfSignedCertificates = $true
        UseSecurityBestPractices = $true

    }

    File RegistrationKeyFile
    {
```

```
        Ensure          = 'Present'
        Type            = 'File'
        DestinationPath =
"$env:ProgramFiles\WindowsPowerShell\DscService\RegistrationKeys.txt"
        Contents        = $RegistrationKey
      }
   }
```

Він доступний за адресою: https://dscpull.testdomain.eu:8080
Його Endpoint: https://dscpull.testdomain.eu:8080/PSDSCPullserver.svc
На всіх клієнтів pull сервера повинен бути встановлений PowerShell 5.1
Якщо не встановлено PowerShell 5.1 виконуємо команду нижче:
$PSVersionTable.PSVersion.Major –lt 5
Встановлюємо PowerShell 5.1:

```
Write-Host "Download PowerShell 5.1"
Invoke-Command -ComputerName $Node -ScriptBlock {
[System.Net.ServicePointManager]::SecurityProtocol=[System.Net.SecurityProtocol
Type]::Tls12;Invoke-WebRequest -Uri
"https://dscpull.testdomain.eu:8080/Files/Updates/WMF.msu" -OutFile
C:\TEMP\WMF.MSU  }

Write-Host "Extract PowerShell 5.1"

   Invoke-Command -ComputerName $Node -ScriptBlock {Start-Process -FilePath
'wusa.exe' -ArgumentList "C:\temp\WMF.msu /extract:C:\temp\" -Wait -PassThru   }

   Write-Host "Apply PowerShell 5.1"

   Invoke-Command -ComputerName $Node -ScriptBlock {Start-Process -FilePath
'dism.exe' -ArgumentList "/online /add-package
/PackagePath:C:\temp\WindowsBlue-KB3191564-x64.cab /Quiet" -Wait -PassThru }

   Write-Host "PowerShell 5.1 has been installed"
```

В нашій мережі також розгорнуто PKI-сервер. Ця умова для безпечного шифрування облікових даних збережених в DSC mof файлах (Mof файли - це «мова» на якому спілкуються Pull Server і його клієнти). Коли клієнт намагається зареєструватися на Pull Server, необхідно вказати Thumbprint сертифіката і надалі Pull Server буде використовувати цей сертифікат для шифрування паролів. Нижче ми розглянемо, як це працює.
Імпортуємо Root CA нашої нової машини:

```
Invoke-Command -ComputerName $server -ScriptBlock{
      $PKI="-----BEGIN CERTIFICATE-----
MIIF2TCCA8GgAwIBAgIQSPIjcff9rotNdxbg3+ygqDANBgkqhkiG9w0BAQUFADAe
**********************************************************************
znafMvVx0B4tGEz2PFss/FviGdC3RohBHG0rF5jO50J4nS/3cGGm+HGdn1w/tZd0
a0FWpn9VCOSmXM2It+tSW1f4nZVt6T2kr1ZlTxkDhT7HMSGsrX/XJswzCkDGe3dE
qrVVjNUkhVTaeeBWdujB5J6mcx7YkNsAUhODiS9Cf7FnYnxLFA72M0pijI48P5F0
ShM9HWAAUIrLkv13ug==
```

-----END CERTIFICATE-----"
$PKI | Out-File RootCA.cer
Import-Certificate RootCA.cer -CertStoreLocation Cert:\LocalMachine\Root | s elect Thumbprint | Out-Null

} -Credential $LocalAdmin | Out-Null

Для подальшої роботи нам потрібна пара RSA-ключів.
Згенеруємо самопідписаний сертифікат[4] і тимчасово будемо працювати з ним
Тепер ми можемо зареєструватися на Pull Server (Додаток Б)
Сервер автоматично перейменується і перезавантажиться. Тепер ми можемо ви конати Join Domain.(**ДОДАТОК Г**)
DSC зашифрував облікові дані від сервісного облікового запису з правами Domain Admin: testdomain.eu \\ service_DomainJoin_001 самоподпісанного сертифікатом. DSC Client своїм Private Key розшифровує облікові дані і застосовує всі модулі конфігурації с зазначеними доменним обліковими даними **. В даному випадку виконує Domain Join в зазначену organization unit.**

{ GroupName = @('Administrators')
Ensure = 'Present'
MembersToInclude = @(testdomain-eu\dscstaging')
}

Цей модуль додає dscstaging в локальні адміністратори для подальшої настройки.
Після перезавантаження, ми зможемо зайти на машину з доменними обліковими даними.
Чекаємо, коли сервер отримає сертифікат від нашого PKI (у нас налаштований auto enrollment) і в подальшому будемо працювати з сертифікатом, випущеним нашим PKI.
$vmcert=Invoke-Command -ComputerName $server -ScriptBlock{ return Get-ChildItem -Path cert:\LocalMachine\My | where {$_.EnhancedKeyUsageList.FriendlyName -eq "Document Encryption"-and $_.Issuer -eq "CN=TestDomain Issuing CA, DC=testdomain, DC=eu"} } -ErrorAction Ignore
Тепер знову реєструватуємось на Pull Server з оновленим **thumbprint.**
Все, машина domain-joined, і ми можемо використовувати її так, як нам зручно.

Установка SQL Server

У JSON- файлі описані вимоги по MS SQL Server, для установки і налаштуванн я SQL Server ми також використовуємо **DSC**. Ось як виглядає конфігурація:
Configuration $Node{
WindowsFeature "NetFramework35"{
Name = "NET-Framework-Core"
Ensure = "Present"
Source = "\\$DscHostFQDN\Files\Updates"
}

WindowsFeature "NetFramework45"{

```
        Name = "NET-Framework-45-Core"
         Ensure= "Present"
          }
          SqlSetup "MSSQL2012NamedInstance"{
          InstanceName         = $MSSQL.InstanceName
              Features           = $MSSQL.Features
ProductKey          = $ProductKey
 SQLCollation         = $MSSQL.Collation
SQLSysAdminAccounts   = @('testdomain-EU\SQLAdmins',' testdomain-EU\Backup'
)
InstallSharedDir     = "C:\Program Files\Microsoft SQL Server"
 InstallSharedWOWDir   = "C:\Program Files (x86)\Microsoft SQL Server"
 InstallSQLDataDir     = $MSSQL.DataRoot
 SQLUserDBDir         = $MSSQL.UserDBDir
SQLUserDBLogDir       = $MSSQL.UserLogDir
SQLTempDBDir         = $MSSQL.TempDBDir
SQLTempDBLogDir       = $MSSQL.TempDBLogDir
SQLBackupDir         = $MSSQL.BackupDir
SourcePath          = $SQLSource
 SAPwd               = $SA
 SecurityMode         = 'SQL'
UpdateSource         = ".\Updates"
Action              = "Install"
ForceReboot          = $True
SQLSvcAccount         = $SqlServiceCredential
AgtSvcAccount        = $SqlServiceCredential
ISSvcAccount         = $SqlServiceCredential
 BrowserSvcStartupType = "Automatic"
DependsOn           = '[WindowsFeature]NetFramework35', '[WindowsFeature]NetF
ramework45'
}
```

Де $ MSSQL визначено:

```
$MSSQL=$Configuration.Applications | where {$_.Application -eq "Microsoft SQL S
erver 2012"}
```

$ MSSQL.InstanceName зазначений в нашому Json файлі. Застосування даної конфігурації виконає установку MS SQL Server со всіма оновленнями в папці Updates і перезавантажить сервер, якщо це необхідно.

Машина готова.

3.2 SERVICE -NOW

В Service-Now доступно кілька API [5]. Ми використоаємо Rest API.

Для отримання списку машин зі статусом Allocated використовується запит виду:

instance.service-now.com/cmdb_ci_server_list.do?sysparm_query=install_status=1
6 ^ u_subtype = ^ ORDERBYname

В PowerShell це виглядає так:

```
$url="https://instance.service-now.com/api/now/table/cmdb_ci_server?sysparm_query=install_status=16^u_subtype=^ORDERBYname"
$uri= new-object System.Uri("https://instance.service-now.com/")
#берем учетные записи нашей сервисной учетной записи
$credentials = (Get-StoredCredential -Type Generic -Target DSC).GetNetworkCredential()
$credentials = new-object System.Net.NetworkCredential $credentials.UserName, $credentials.SecurePassword
Add-Type -AssemblyName System.Net.Http
$handler = New-Object  System.Net.Http.HttpClientHandler
$handler.CookieContainer = New-Object System.Net.CookieContainer
$handler.UseCookies=$true
$handler.Credentials=$credentials
$HttpClient = New-Object System.Net.Http.HttpClient($handler)
$HttpClient.BaseAddress= $uri
$Header = New-Object System.Net.Http.Headers.MediaTypeWithQualityHeaderValue("application/json")
$HttpClient.DefaultRequestHeaders.Accept.Clear()
$HttpClient.DefaultRequestHeaders.Accept.Add($Header);
$response=$HttpClient.GetAsync($url)
$respStream=$response.Result.Content.ReadAsStringAsync()
$Servers = $respStream.Result | ConvertFrom-Json
#Отримуєм об’єкти Configuration Items каталога
$ServersCI=$Servers.result
```

Перший об'єкт масиву і є потрібний нам hostname.

Якщо машина готова, то можна змінити статус машини в **Service-Now**, для цього користуэмось скриптом UpdateCI.ps1:

```
param(
   $CI,
   [ValidateSet("Allocated","In use","Pending install")]
   $NewStatus='In use'
)

$url="https://instance.service-now.com/api/now/table/cmdb_ci_server?sysparm_query=name=$CI"
$uri= new-object System.Uri("https://instance.service-now.com/")
$credentials = (Get-StoredCredential -Type Generic -Target DSC).GetNetworkCredential()
$credentials = new-object System.Net.NetworkCredential $credentials.UserName, $credentials.SecurePassword
Add-Type -AssemblyName System.Net.Http
$handler = New-Object  System.Net.Http.HttpClientHandler
$handler.CookieContainer = New-Object System.Net.CookieContainer
$handler.UseCookies=$true
```

```
$handler.Credentials=$credentials
$HttpClient = New-Object System.Net.Http.HttpClient($handler)
$HttpClient.BaseAddress= $uri
$Header = New-Object System.Net.Http.Headers.MediaTypeWithQualityHeaderValu
e("application/json")
$HttpClient.DefaultRequestHeaders.Accept.Clear()
$HttpClient.DefaultRequestHeaders.Accept.Add($Header);
$response=$HttpClient.GetAsync($url)
$respStream=$response.Result.Content.ReadAsStringAsync()
$Servers = $respStream.Result | ConvertFrom-Json
$ServerCI=$Servers.result[0]
$update=@{}
if($NewStatus -eq "In use"){
   $update.install_status=1
}
if($NewStatus -eq "Pending install"){
   $update.install_status=4
}
$stringcontent =  New-Object System.Net.Http.StringContent((ConvertTo-Json -Input
Object $update -Depth 100),[System.Text.Encoding]::UTF8, "application/json");
$result=$HttpClient.PutAsync("https://instance.service-now.com/api/now/table/cmdb
_ci_server/$($ServerCI.sys_id)", $stringcontent)
```

Для отримання таблиці і записів використовуються REST API GET запити, для зміни запису PUT / POST запити, в тілі якого поля які необхідно змінити.

3.3 КОМПЛЕКСНА АВТОМАТИЗАЦІЯ З ANSIBLE

Рішення Ansible гарантують максимальну гнучкість. Це дозволяє спільноті знаходити все нові способи використання модулів Ansible для автоматизації часто виконуваних операцій на багатьох рівнях, в тому числі в поєднанні технологіями OpenStack.

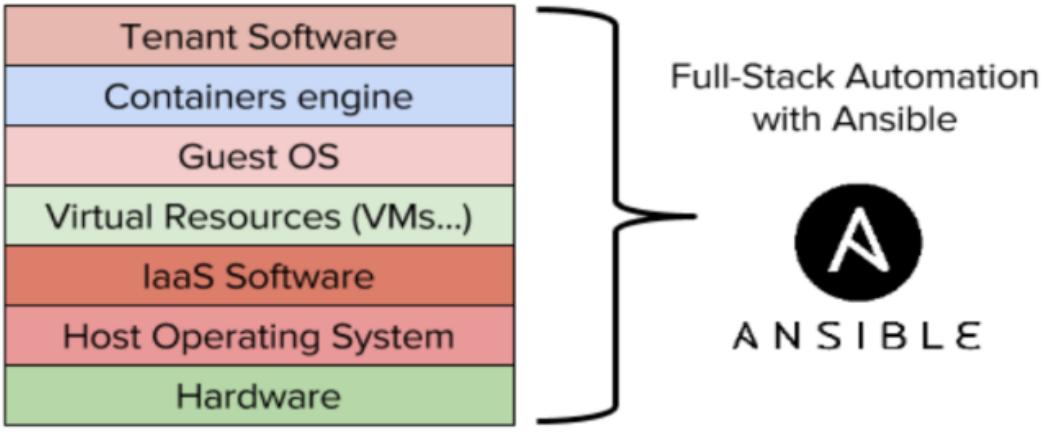

Рис 3.4 Технології з якими поєднаний Ansible

Для початку обговоримо рівні комплексної автоматизації, показані на діаграмі вище. Внизу у нас обладнання (сервери, системи зберігання даних та мережеве обладнання). Далі йде операційна система (Linux або Windows). На Linux ви можете встановити OpenStack, щоб абстрагуватися від фізичних ресурсів центру обробки даних і розгорнути програмну версію обчислювальних ресурсів, мережі та сховища. Поверх OpenStack розгортаються певні орендарем служби, необхідні для створення віртуальних машин, на яких будуть запускатися програми. Нарешті, ви повинні забезпечити управління операційною системою (Linux або Windows), щоб розгортати додатки і робочі навантаження, які вам дійсно потрібні (бази даних, веб-сервери, сервери мобільних додатків і т. Д.). Якщо ви використовуєте контейнери (наприклад, Docker або Rkt), ви будете упаковувати ці додатки в образи, які будуть потім розгорнуті поверх гостьовий ОС. Крім того, в деяких мовах з'явилася концепція серверів додатків, яка додає ще один рівень (наприклад, J2EE).

3.3.1 Можливості для управління, що надаються рішеннями Ansible

Ansible надає модулі для управління кожним шаром. Навіть для мережевого обладнання, точніше, якщо підходити до цього з технічної точки зору, для мережевої операційної системи, такої як IOS або NXOS (див. Повний список мережевих модулів Ansible).

1.) **Стандартна взаємодія з операційною системою**: установка пакетів, зміна

або примусове застосування вмісту або прав доступу до файлів, управління службами, створення і видалення користувачів і груп користувачів і т. д.

- Linux і BSD через SSH (перший і самий популярний варіант використання).
- Windows через PowerShell (починаючи з версії 1.7).

2.) **Програмне забезпечення** для реалізації концепції «інфраструктура як » (IaaS): встановіть програмне забезпечення IaaS і супутні компоненти (бази даних, балансувальник навантаження, файли конфігурації, служби та інші допоміжні інструменти).

- Інсталятор OpenStack-ansible, який використовується в деяких збірках OpenStack від інших постачальників. Не забувайте про те, що платформа Red Hat OpenStack використовує не Ansible, а Heat і Puppet. У наступних релізах Ansible буде використовуватися для виконання певних перевірок і надання операторам допомоги в процесі установки оновлень і нових версій.
- Інсталятор CloudStack - це також проект на основі Ansible.

3.) **Віртуальні ресурси:** налаштовуйте ресурси, наприклад віртуальну машину або екземпляр, визначаючи розмір, права доступу, контент, профіль безпеки, параметри мережевого підключення і т. д.

- Модулі OpenStack Ansible (починаючи з Ansible 2.0), наприклад Nova або Neutron . Вони засновані на бібліотеці OpenStack Shade, яка використовується всіма інструментами CLI в OpenStack.
- Ці модулі також можуть управляти не такими вже й віртуальними мережевими ресурсами через netconf (починаючи з версії 2.2).
- Модулі VMware vSphere Ansible [6].
- RHV, або oVirt, або Libvirt тільки для KVM.
- Також є модулі для постачальників хмарних послуг, таких як Amazon , Google Cloud , Azure і Digital Ocean .

4.) **Гостьова ОС**: компоненти аналогічні таким у ОС хоста. Але як ви дізнаєтеся, скільки у вас гостьових систем?

- Інструмент Ansible Dynamic Inventory буде динамічно опитувати рівень IaaS і ВМ з метою виявлення доступних в даний час примірників. При цьому зазначається ім'я вузла, IP-адреси і налаштування безпеки, тобто управління активами стає динамічним. Така можливість буде особливо корисна для тих, хто використовує групи автомасштабирование (Auto Scaling Groups) у своїй хмарної інфраструктурі, оскільки список примірників кардинально змінюється з плином часу.

5.) **Модуль управління контейнерами (на вибір).**

- Docker : зверніть увагу, що старий модуль Docker в версії Ansible 2.1 буде замінений на новий власний компонент.
- Губернаторів.
- Атомний господар.

6.) **Орендоване ПО**: бази даних, веб-сервери, балансувальник навантаження, модулі обробки даних і т. д.

- Ansible Galaxy - репозиторій алгоритмів (playbook) для розгортання самого популярного ПО, і це результат спільної роботи тисяч членів спільноти.
- Ви також можете керувати веб-інфраструктурою на зразок JBoss , дозволяючи Ansible визначати, як додаток буде розгортатися на сервері додатків.

Рекомендації по установці останньої версії Ansible у віртуальному середовищі Python Як ви вже зрозуміли, деякі функції доступні тільки в новітніх версіях Ansible, наприклад 2.2. Проте у вашій версії ОС може бути більш стара версія. Наприклад, в RHEL 7 або CentOS 7 ви знайдете тільки Ansible 1.9.

З огляду на те, що Ansible - це інструмент командного рядка, написаний на мові Python, який не забороняє наявності декількох версій в системі, нам, можливо, не знадобиться посилення безпеки в Ansible, яке пропонує наша збірка, і ми захочемо випробувати останню версію.

Однак, як і у випадку з будь-яким іншим програмним забезпеченням Python,

існує багато залежностей, тому не рекомендується використовувати неперевірені нові бібліотеки разом з системними. Бібліотеки можуть спільно використовуватися іншими компонентами вашої системи, і неперевірені новіші версії здатні порушити роботу інших додатків. Найпростіше встановити останню версію Ansible з усіма залежностями в ізольовану папку під вашою непривілейованої обліковим записом. Це буде віртуальне середовище Python (***Python Virtual Environment*** , virtualenv), і якщо ви все зробите правильно, то зможете безпечно випробувати останні модулі Ansible для комплексної оркестровки. Зрозуміло, ***не рекомендується***застосовувати таку практику у виробничому середовищі. Це не більше ніж вправа, що допомагає вам поліпшити свої навички в області DevOps.

1. Встановлюємо обов'язкові компоненти (pip, virtualenv)

В даному випадку нам знадобиться тільки одна «загальносистемна» бібліотека Python - virtualenvwrapper. Крім цього, нам не слід виконувати команду sudo pip install, оскільки це замінить системні бібліотеки Python на більш нові неперевірені версії. Довіряти в даному випадку ми можемо тільки бібліотеці virtualenvwrapper. Віртуальне середовище - ефективний механізм установки і тестування нових модулів Python у нашому непривілейованому обліковому запису користувача .

$ sudo yum install python-pip

$ sudo pip install virtualenvwrapper

$ sudo yum install python-heatclient python-openstackclient python2-shade

Крім того, якщо ваш обліковий запис відмінна від sudoer, ви можете використовувати наступні команди.

wget https://bootstrap.pypa.io/get-pip.py

python get-pip.py –user .local/bin/pip install virtualenvwrapper --userexport PATH=$PATH:~/.local/bin *#Remember to include this in your .bashrc*

2. Встановлюємо нове віртуальне середовище, де ми буде розгортати останню версію Ansible

Для початку створіть каталог для зберігання віртуальних середовищ.

$ mkdir $HOME/.virtualenvs

Потім додаємо наступні рядки в ваш файл .bashrc:

export WORKON_HOME = $ HOME / .virtualenvs

source /usr/bin/virtualenvwrapper.sh

Тепер виконайте команду ***source*** для цього файлу.

$ source ~/.bashrc

На цьому етапі створюються посилання-оболонки, але тільки при першому запуску. Виконуємо команду ***workon*** , щоб переглянути список середовищ, він не повинен бути порожнім. Далі ми створюємо нове середовище ***virtualenv*** з ім'ям ***ansible2*** , воно буде активовано автоматично і отримає доступ до пакетів, встановленим за замовчуванням за допомогою RPM(віддалене управління пакетами).

$ mkvirtualenv ansible2 --system-site-packages

Для виходу з середовища virtualenv введіть deactivate , а для повторного входу - workon.

$ deactivate

$ workon ansible2

Заходимо в створене віртуальне середовище virtualenv і встановлюємо Ansible2 через PIP (як звичайний користувач, без root-прав)

Бачимо, що запит на введення команди в оболонці змінився і в дужках вказано ім'я virtualenv, тобто було створено таке собі тестове віртуальне середовище всередині сереверу і при цьому нам не потрібно мати права доступу, оскільки середовище є ізольованим.

(ansible2) $ pip install ansible

Ця команда встановить тільки залежності для Ansible 2, використовуючи наші загальносистемні пакети Python, доступні в RPM (завдяки мітці system-site-packages, який ми використовували раніше). Альтернативний варіант існує якщо ми звертаємось до стороннього репозиторію з готовим рішенням.

(ansible2) $ pip install git+git://github.com/ansible/ansible.git@devel

(ansible2) $ ansible --version

Якщо вам потрібно буде видалити середовище virtualenv і всі її залежності, просто введіть команду *rmvirtualenv ansible2*.

ПРИМІТКА. Якщо наведені вище команди не спрацювали, переконайтеся, що в ваших системах встановлені бібліотеки GCC і OpenSSL, оскільки деякі залежності Python використовують останні криптографічні модулі.

4. Встановлюємо залежності клієнта OpenStack

Перша команда з наведеного нижче фрагмента коду гарантує, що у вас будуть останні стабільні версії OpenStack API, проте ви також можете скористатися командою ***pip install,*** щоб отримати останню версію інструменту командного рядка (CLI). Друга команда надає останню версію бібліотеки Python Shade для підключення до останніх версій OpenStack API за допомогою ansible, незалежно від конкретного інструменту CLI.

(ansible2) $ yum install python-openstackclient python-heatclient

(ansible2) $ pip install shade --upgrade

5. Протестуйте своє розгортання

(ansible2) $ ansible -m ping localhost

localhost | SUCCESS => {

"changed": false,

"ping": "pong"

}

ПРИМІТКА. Ми можемо запускати цю версію ansible за межами віртуального середовища virtualenv, тому слід завжди команду workon ansible2 перед usi.

3.Оркестрування OpenStack з використанням Ansible

Використовуючи Ansible для оркестровки OpenStack, ми, схоже, ігноруємо той факт, що офіційним модулем оркестровки для цієї платформи все ж є Heat. Фактично Ansible Playbook надає практично ті ж можливості, що і

шаблон HOT (HOT - це синтаксис на основі YAML для Heat, новішої версії AWS CloudFormation). Проте багато професіоналів в області DevOps вважають за краще не витрачати час на вивчення нового синтаксису, і вони вже консолідують весь процес для своєї гібридної інфраструктури.

Команда Ansible розуміє це, тому використовує Shade,[7] Офіційну бібліотеку з проекту OpenStack, для створення інтерфейсів для виклику OpenStack API. На момент написання цієї статті версія Ansible 2.2 включала модулі для виклику наступних API.

Keystone: користувачі, групи, ролі, проекти.

- Nova: сервери, пари ключів, групи і прапори безпеки.
- Neutron: порти, мережа, підмережі, маршрутизатори, плаваючі IP-адреси.
- Ironic: вузли, інтроспектіва.
- Об'єкти Swift.
- Tom Cat.
- Образи Glance.

З точки зору Ansible необхідно організувати взаємодію з сервером, де система зможе завантажувати облікові дані OpenStack і створювати HTTP-з'єднання з різними OpenStack API. Якщо цей сервер - ваша машина (localhost), то Ansible буде працювати локально, завантажувати облікові дані Keystone і ініціювати взаємодію з OpenStack.

Давайте розглянемо приклад. Ми будемо використовувати модулі Ansible OpenStack для підключення до Nova і запуску невеликого плейбука з образом Cirros. Для початку ми завантажимо останній образ Cirros, якщо ви не зробили цього раніше. Ми скористаємося існуючим ключем SSH поточного користувача. Ви можете завантажити цей playbook за цим посиланням на Github.[**ДОДАТОК Г**]

Після запуску ми бачимо IP-адресу плебука. Запишемо його. Тепер можна використовувати Ansible для підключення до цього плейбука через SSH. Передбачається, що за замовчуванням мережу Nova підтримує

підключення з нашої робочої станції (в нашому випадку через мережу постачальника послуг). Давайте розглянемо, що станеться, якщо ми створимо багато примірників, але забудемо записати їх IP-адреси. Чудовим прикладом використання інструменту Dynamic Inventory для OpenStack є аналіз поточного стану наших орендованих віртуалізованих ресурсів і отримання всіх IP-адрес серверів, щоб ми могли, наприклад, перевірити версію їх ядра. Наприклад, це прозоро робиться за допомогою рішення Ansible Tower, яке буде періодично проводити інвентаризацію і формувати оновлений список серверів OpenStack, що підлягають управлінню.

Перш ніж виконувати ці операції, переконаємось, що у нас немає застарілих файлів cloud.yaml в каталозі ~ / .config / openstack, / etc / openstack або / etc / ansible. Скрипт Dynamic Inventory спочатку буде шукати змінні середовища (OS_ *), а потім - перераховані файли.

#ensure you are using latest ansible version

$ workon ansible2

$ wget https://raw.githubusercontent.com/ansible/ansible/devel/contrib/inventory/openstack.py

$ chmod +x openstack.py

$ ansible -i openstack.py all -m ping

bdef428a-10fe-4af7-ae70-c78a0aba7a42 | SUCCESS => {

"changed": false,

"ping": "pong"

}

343c6e76-b3f6-4e78-ae59-a7cf31f8cc44 | SUCCESS => {

"changed": false,

"ping": "pong"

}

ВИСНОВОК ДО РОЗДІЛУ 3

У цьому розділі було розглянуто основні інструменти для налаштування інфраструктури , а також управління віртуалізацією за допомогою PS,Python та Ansible playbooks. Проміжні етапи впровадження елементів інфраструктури були реалізовані на рівні бази данних. Основним мотором цих операцій(оркестратором) -виступав Ansible- найбільш прогресивне відкрите ПО.

Ми написали декілька основних конфігурацій для машин в середовищі і маємо змогу їх моніторити. Скрипти ми використовували для автоматизації рутинних задач які стоять перед системним адміністратором використовуючи уже відомий нам power shell та WMI. Дуже велику перевагу я вбачаю в тому що інструменти не заважають одне одному . Скрипт на powershell може бути записаний в ANSIBLE плейбук з відповідними умовами виконання. Для архітектора IT інфраструктури (DEVOPS) це найбільш операбельні інструменти. У відкритому доступі існує безліч документації а спільнота вносить свій вклад в розвиток продукту. Тут і поєднується комерційна і оупенсоурс складова, і від цього бізнес тільки виграє.

.

ВИСНОВКИ

Було створенно концепцію написання скриптів та їх запуску як частини процесу управління інфраструктурними сервісами які вже є вмонтованими в систему. Скрипти забезпечують автоматизацію процесів та послідовність виконнання сервісів які звертаються до ядра системи взаємодіючи з програмною оболонкою виконання скриптів. Існує безліч моделей найбільш стійких архітектур, таких як гібридні наприклад коли хмарні сховища та сервіси взаємсодіють з локальною мережею існує ефект деякої дзеркальної синхронізації . Для бізнесу в даному контексті часу немає меж та обмежень . Інформація та доступи повинні бути відкриті і опрацьовані робітниками будь де і будь коли. Саме для забезпечення стабільною роботи і створюються скрипти на будь-який смак і відповідно до задачі та бізнес очікувань. Єдине питання було як ми можемо використати один скрипт на 1000 машин. Так , це не легко вважаючи що у кожної машини є свої власні апаратні та програмні особливості. Оркестрація та створення конфігураційних файлів є першим кроком до справжнього розуміння потрібності скриптів. Якщо ми хочемо виконати операції симетрично або ассиметрично ми створюємо 2 скрипти але методи їх імплементаціх полягають у виборі продукту і побудови правильної структурної моделі їх виконання . Адміністратор повинен сформувати правильну бізнес задачу в першу чергу для себе , включити в неї всі недоліки , і спробувати їх мінімізувати не порушуючи основну структуру, при цьому враховуючи час на виконання операцій що теж повинно бути розраховано в рамках технічного завдання. Таким чином створюється дуже спрощена архітектура завдяки багатьом інструментам про які йшла мова в данній роботі. В даній роботі я вирішив піти від простого і закінчити безпосередньо актуальними технологіями та алгоритмізацією команд ядра .

ДОДАТОК А

ОСНОВНІ КОМАНДИ КОМАНДНОГО РЯДКА

APPEND - дозволяє програмам відкривати файли даних із зазначених папок так, як ніби вони знаходяться в цій папці.

ARP - перегляд і зміна таблиць ARP(прив'язка Мак адреси до Ip адреси) (Address Resolution Protocol)

ASSOC - перегляд або зміна зіставлень розширень файлів додатків

AT - управління планувальником завдань

ATTRIB - зміна атрибутів файлів

Auditpol - управління політиками аудиту.

BASH - командна оболонка BASH в підсистемі Windows для Linux (**WSL**).-про неї ми поговоримо пізніше.

BCDBOOT - копіювання в системний розділ файлів завантаження і створення нового сховища конфігурації завантаження (BCD)

BCDEDIT - редагування сховища даних конфігурації завантаження (BCD)

BOOTCFG - управління параметрами меню завантаження і відновлення Windows 10

BOOTIM - редагування параметрів завантаження в файлі boot.ini

BOOTREC - відновлення завантажувальних записів і конфігурації завантаження Windows

BOOTSECT - редагування завантажувальних секторів для забезпечення завантаження NTLDR або BOOTMGR

BREAK - включити або вимкнути обробку комбінації клавіш CTRL + C в DOS

CACLS - редагування списків управління доступом до файлів (ACL - Access Control List)

CALL - виклик з командного файлу підпрограм або інших командних файлів

CD - зміна каталогу (Change Directory)

CHANGE - зміна налаштувань сервера терміналів. Контексти - LOGON, PORT, **USER**

CHGLOGON - зміна налаштувань сервера терміналів, аналогічно **CHANGE LOGON**

CHGPORT - зміна налаштувань сервера терміналів, аналогічно CHANGE PORT

CHGUSR - зміна налаштувань сервера терміналів, аналогічно CHANGE USER

CHCP - перегляд або зміна поточної кодової сторінки

CHKDSK - перевірка диска (Check Disk)

CheckNetIsolatin - управління доступом додатків до інтерфейсу замикання на себе (localhost)

CHKNTFS - перевірка ознаки системну помилку і управління перевіркою диска при завантаженні Windows

CHOICE - реалізація призначеного для користувача введення в командному файлі

CIPHER - відображення або зміна шифрування файлів на томах NTFS

CLEARMGR - управління очищенням дисків Windows

CLIP - перенаправлення виведення утиліт командного рядка в буфер обміну Windows

CLS - очищення екрану в командному рядку

CMD - запуск нової копії інтерпретатора командного рядка

CMDKEY - створення, відображення, видалення і збереження імен користувачів і паролів

COLOR - зміна кольору тексту і фону у вікні CMD

COMMAND - запуск нової копії інтерпретатора командного рядка MS-DOS

COMP - порівняння вмісту файлів

COMPACT - управління стисненням і розпакуванням файлів в розділах NTFS

CONVERT - перетворення файлової системи з FAT в NTFS

COPY - копіювання файлів і каталогів

Cscript - сервер сценаріїв Windows з консольним інтерфейсом

DATE - відображення або зміна дати

DEBUG - запуск відладчика DOS-Windows XP

DEFRAG - дефрагментація диска

DEL - видалення одного або декількох файлів

DevCon - управління пристроями в командному рядку

DIANTZ - теж що і MAKECAB, створення архівів .cab.

DIR - відображення списку файлів і каталогів

DISKCOMP - порівняння вмісту двох гнучких дисків

DISKCOPY - копіювання вмісту одного гнучкого диска на інший

DISKPART - управління розділами і дисками з командного рядка

DISM - управління компонентами образрв WIM.

DISPDIAG - висновок дампов з діагностичною інформацією про графічної підсистеми.

DJOIN - автономне приєднання комп'ютера до домену.

DOSKEY - редагування і повторний виклик команд Windows, створення макросів

DRIVERQUERY - відобразити інформацію про встановлені драйвери.

DxDiag - засіб діагностики DirectX.

ECHO - висновок тексту на екран консолі

EDIT - запуск текстового редактора

ENDLOCAL - кінець локальних змін змінних оточення в командному файлі

ERASE - аналогічно команді DEL - видалення файлів

ESENTUTL - обслуговування баз даних Extensible Storage Engine для Windows

EVENTCREATE - запис повідомлення в журнал подій Windows

EXIT - вихід з процедури або командного файлу

EXPAND - розпакування стислих файлів CAB-файлів.

EXTRACT - витягання вмісту, розпакування CAB-файлів в Windows (EXTRAC32)

FC - порівняння вмісту файлів

FIND - пошук рядка символів в файлі

FINDSTR - пошук рядків у файлах з використанням регулярних виразів

FOR - організація циклічної обробки результатів виконання інших команд, списків, і рядків в текстових файлах

FORFILES - виконання зазначеної команди для кожного файлу з заданої групи

FORMAT - форматування диска

FSUTIL - управління файловою системою

FTP - консольний FTP-клієнт

FTYPE - перегляд і зміна розширень файлів і зіставлених їм додатків

GETMAC - відображення фізичної адреси мережевого адаптера (MAC-адреси)

GOTO - команда безумовного переходу в командному файлі

GPRESULT - відображення результуючої політики (RSoP)

GPUPDATE - оновлення групових політик.

HELP - виклик довідки командного рядка Windows

HOSTNAME - відображення імені комп'ютера

ICACLS - управління списками доступу (ACL)

IF - оператор умовного виконання команд в пакетному файлі

IPCONFIG перегляд і управління конфігурацією протоколу IP

LABEL - редагування міток томи дисків

LOGMAN - управління монітором оцінки продуктивності системи

LOGOFF - завершення сеансу користувача

MAKECAB - створення стислих файлів формату CAB

MBR2GPT - перетворення дисків MBR в GPT

MEM - висновок довідки про використання пам'яті в MS-DOS

MD - створення нового каталогу

MKLINK - створення символічного посилання на файл або каталог

MODE - конфігурація системних пристроїв в середовищі CMD

MORE - посторінковий вивід в консолі

MOUNTVOL - управління точками монтування томів

MOVE - переміщення файлів і каталогів

MOVEFILE - переміщення або видалення зайнятих файлів після перезавантаження сторінки

MSG - відправлення повідомлень користувачам.

MSTSC - підключення до віддаленого робочого столу.

NBTSTAT - перегляд статистичних даних NETBIOS через TCP / IP (NetBT)

NET - управління ресурсами локальної мережі

NETCFG - відображення і зміна конфігурації компонентів мережі

NETSH - командна мережева оболонка (Network Shell)

NETSTAT - відображення статистики мережевих з'єднань

NSLOOKUP - перегляд даних DNS в командному рядку

OPENFILES - управління відкритими по мережі або локально файлами

PATH - відображення або зміна шляхів пошуку файлів

PATHPING - трасування маршруту з можливістю оцінки якості ділянок траси

PAUSE - пауза при виконанні командного файлу

PING утиліта перевірки доступності вузла

PKGMGR - управління програмними пакетами Windows

PNPUTIL - конфігурація драйверів пристроїв PnP

POPD - повернення в каталог, раніше запомненний за допомогою команди PUSHD

POWERCFG - настройка параметрів системи електроживлення Windows

PRINT - друк текстового файлу

PROMPT - зміна рядка запрошення в консолі

PUSHD - зберегти поточний шлях каталогу і перейти в зазначений

PSR - записати дії користувача (Problem Steps Recorder)

QPROCESS - відобразити стан процесів

QUERY - опитати стан процесів і сеансів користувачів

QUSER - відобразити інформацію про сеанси користувачів

RASDIAL - управління сеансами віддаленого доступу

RASPHONE - управління сеансами віддаленого доступу

RD - видалення каталогу

REAGENTC - адімністрірованіе стреди відновлення Windows

RECOVER - відновлення файлів на ушкодженому диску

REG - утиліта командного рядка для роботи з реєстром Windows

REGEDIT - імпорт і експорт даних реєстру Windows

REGSVR32 - реєстрація або скасування реєстрації DLL

REGINI - управління доступом до розділів реєстру

REM - коментарі в командних файлах

RENAME (REN) - перейменування файлів

REPLACE - заміна або додавання файлів в Католог

RESET - скидання сеансу віддаленого робочого стола (RDP сесії)

RMDIR - видалення каталогу

ROBOCOPY - утиліта резервного копіювання та синхронізації каталогів (Robust File and Folder Copy)

ROUTE - управління таблицею маршрутизації

RUNAS - запуск програми від імені іншого користувача

RUNDLL32 - запуск DLL в якості додатку

SC - управління службами Windows (Service Control)

SCHTASKS - управління планувальником завдань

SCLIST - відображення списку системних служб

SET - відображення і зміна змінних середовища оточення Windows

SETLOCAL - установка локальних змінних в командному файлі

SETX - утиліта для створення системних змінних

SFC - перевірка і відновлення системних файлів Windows

SHARE - перегляд, створення і видалення поділюваних в локальній мережі ресурсів

SHIFT зрушення вхідних параметрів для командного файлу

SHUTDOWN - виключення або перезавантаження комп'ютера

SLEEP - затримка за часом в пакетному файлі

SLMGR - управління ліцензуванням програмного забезпечення Windows

SORT - сортування рядків у текстовому файлі

START - запуск програми або командного файлу

STORDIAG - діагностика системи зберігання даних в Windows 10

SUBST - призначення (скасування призначення) каталогу букви диска

SxSTrace - діагностичний засіб трасування компонент системи

SYSTEMINFO - відображення інформації про систему

TAKEOWN - зміна власника файлу або каталогу

TAR - архівування даних архіватором tar в Windows 10

TASKKILL - завершення процесів на локальній або віддаленій системі.

TASKLIST - відображення списку виконуються додатків і служб Windows

TIME - відображення і установка системного часу

TELNET - telnet-клієнт Windows

TFTP - **TFTP**-клієнт Windows

TIMEOUT - затримка в пакетних файлах

TITLE - зміна заголовка вікна CMD.EXE

TRACERT - трасування маршруту до віддаленого вузла

TREE - відображення структури каталогу в графічному вигляді

TSCON - підключення до сесії віддалених робочих столів (RDP).

TSDISCON - відключення сесії віддалених робочих столів (RDP).

TSKILL - завершення процесів, адаптоване для середовища сервера терміналів (RDP).

TYPE - вивід на екран вмісту текстового файлу

TypePerf - виведення відомостей про продуктивність на екран або в журнал

TZUTIL - управління часовими поясами в середовищі Windows

VER - відображення версії операційної системи

VERIFY - управління режимом перевірки записуються файлів

VOL - вивід даних мітки тому жорсткого диска.

VSSADMIN - адміністрування служби тіньового копіювання томів.

W32TM - управління службою часу Windows

WAITFOR - організація обміну сигналами між комп'ютерами

WBADMIN - управління резервним копіюванням і відновленням в Windows

WEVTUTIL - управління подіями в Windows

WHERE - визначення місця розташування файлів

WHOAMI - висновок імені поточного користувача

WINDIFF - порівняння вмісту файлів

WinMgmt - обслуговування інструментарію управління Windows (WMI)

WINRM - віддалене управління Windows з командного рядка

WINRS - дистанційна командний рядок (Remote Shell)

WINSAT - засіб перевірки продуктивності Windows

WMIC - виконання команди WMI в командному рядку

WSCollect - отримати CAB-файл з копіями журналів Windows 10 на робочому столі

Wscript - сервер сценаріїв Windows з графічним інтерфейсом

WSL - виконання команд Linux і конфігурація параметрів підсистеми Windows для Linux (WSL) в Windows 10

WSLconfig - конфігурація параметрів підсистеми Windows для Linux (WSL) в Windows 10

XCOPY - копіювання файлів і папок.

ДОДАТОК Б

Рестрація головного сервера конфігурації

$DscHostFQDN =
[System.Net.Dns]::GetHostEntry([string]$env:computername).HostName

```
$DscPullServerURL = "https://$($DscHostFQDN):8080/PSDSCPullserver.svc"

$DscWebConfigChildPath = '\inetpub\psdscpullserver\web.config'

$DscWebConfigPath = Join-Path -Path $env:SystemDrive -ChildPath
$DscWebConfigChildPath

$DscWebConfigXML = [xml](Get-Content $DscWebConfigPath)

$DscRegKeyName = 'RegistrationKeys.txt'

$DscRegKeyXMLNode = "//appSettings/add[@key = 'RegistrationKeyPath']"

$DscRegKeyParentPath =
($DscWebConfigXML.SelectNodes($DscRegKeyXMLNode)).value

$DscRegKeyPath = Join-Path -Path $DscRegKeyParentPath -ChildPath
$DscRegKeyName

$DscRegKey = Get-Content $DscRegKeyPath


[DSCLocalConfigurationManager()]

configuration RegisterOnPull

{

   Node $Node

   {

      Settings

      {

         ConfigurationModeFrequencyMins =   1440

         CertificateID = $Thumbprint

         RefreshMode          ='Pull'

         RefreshFrequencyMins = 1440

         RebootNodeIfNeeded   = $true

         ConfigurationMode ='ApplyAndAutoCorrect'

         AllowModuleOverwrite = $true
```

```
        DebugMode = 'None'
        StatusRetentionTimeInDays = 1

    }

    ConfigurationRepositoryWeb $([string]$env:computername)
    {
        ServerURL =  $DscPullServerURL
        RegistrationKey = $DscRegKey
        CertificateID = $Thumbprint
        ConfigurationNames = @("$hostx")
    }
  }
}
RegisterOnPull -OutputPath $MetaConfigsStorage
Set-DscLocalConfigurationManager -ComputerName $Node  -Path
$MetaConfigsStorage  -Verbose -Force -Credential $LocalAdmin
```

Відправимо першу конфігурацію нашої машині

```
Configuration Rename
{
  param
  (
    [Parameter()]
    [System.String[]]
    $Node,
    $hostname
```

```
    )

    Import-DscResource -ModuleName xComputerManagement
    Import-DscResource –ModuleName PSDesiredStateConfiguration

    Node $Node
    {

        xComputer JoinDomain
        {
            Name       = $hostname
        }
    }
}
Rename -Node $Node -OutputPath $DscConfigPath -hostname $hostname
New-DscChecksum $DscConfigPath -Force
Invoke-Command -ComputerName $Node -ScriptBlock{Update-DscConfiguration -
Verbose -Wait } -Credential $LocalAdmin -Verbose
```

ДОДАТОК В

Створення конфігураційного файлу. Зберігання вхідних данних в SQL та звернення ITSM системи Service NOW до бази данних

```
.\CreateConfiguration.ps1 -SecurityZone trusted -VMDescription "VM for CRM
System" -Requestor "evgeniy.vpro" -OSVersion 2k16 -OSEdition Standard -
```

BuildNewVM -VMEnvironment Prod -VMServiceLevel GOLD -VMSize Medium -Disk0Tier Fast -Disk1Size 50 -Disk1Tier Eco -Disk1Letter D -MSSQLServer -MSSQLInstanceName "Instance1" -SQLCollation Latin1_General_CI_AS -SQLEdition Standard -Disk2Size 35 -Disk3Size 65

У тілі CreateConfiguration .ps1 скрипта:

```
#створюємо PowerShell-объект

$config = [ordered]@{}

#И заповнюємо його вхідними параметрами.

$config.SecurityZone=$SecurityZone
```

В кінці експортуємо наш об'єкт в JSON-файл:

```
$ServerConfig = New-Object –TypeName PSObject  $config

ConvertTo-Json -InputObject $ServerConfig -Depth 100 | Out-File "C:\Configs\TargetNodes\Build\$($Hostname.ToLower()).json" -Force
```

Зразок конфігурації:

```
{
   "Hostname":  "dsctest552",
   "SecurityZone":  "trusted",
   "Domain":  "testdomain",
   "Requestor":  "taras.vpro",
   "VM":  {
            "Size":  "Medium",
            "Environment":  "Prod",
            "SLA":  "GOLD",
            "DbEngine":  "MSSQL",
            "RAM":  8,
            "Storage":  [
                        {
```

```
    "Id": 0,
    "Tier": "Fast",
    "Size": "100",
    "Allocation": 4,
    "Letter": "C"
},
{
    "Id": 1,
    "Tier": "Eco",
    "Size": 50,
    "Label": "Data",
    "Allocation": 64,
    "Letter": "D"
},
{
    "Id": 2,
    "Tier": "Fast",
    "Size": 35,
    "Label": "Data",
    "Allocation": 64,
    "Letter": "E"
},
{
    "Id": 3,
    "Tier": "Fast",
    "Size": 65,
```

```
                "Label": "Data",
                "Allocation": 64,
                "Letter": "F"
            }
        ]
    },
"Network": {
        "MAC": "",
        "IP": "10.230.168.50",
        "Gateway": "10.230.168.1",
        "VLAN": “VLAN168”
    },
"OS": {
    "Version": "2k16",
    "Edition": "Standard",
    "Administrators": [
                "LocaAdmin",
                "testdomain\\ Security-LocalAdmins"
            ]
    },
"OU": "OU=Servers,OU=Staging,DC=testdomain",
"Applications": [
        {
            "Application": "Microsoft SQL Server 2016",
            "InstanceName": "vd",
            "Collation": "Latin1_General_CI_AS",
```

```
                "Edition": "Standard",
                "Features": "SQLENGINE",
                "Folders": {
                        "DataRoot": "E:\\MSSQL",
                        "UserDB": "E:\\MSSQL\\MSSQL11.vd\\MSSQL\\Data",
                        "UserLog": "E:\\MSSQL\\MSSQL11.vd\\MSSQL\\Log",
                        "TempDB":
"D:\\MSSQL\\MSSQL11.vd\\MSSQL\\TempDB",
                        "TempDBLog":
"D:\\MSSQL\\MSSQL11.vd\\MSSQL\\TempDB",
                        "Backup":
"E:\\MSSQL\\MSSQL11.vd\\MSSQL\\Backup"
                    },
                "MaxMemory": 2147483647
            }
        ],
    "Description": "VM for CRM",
    "Certificate": {
            "File": null,
            "Thumbprint": null
        },
    "Version": 0
}
```


ДОДАТОК Г
ПРИЄДНАННЯ СЕРВЕРУ В ДОМЕН

Configuration JoinAD

```
{
    param
    (
        [Parameter()]
        [System.String[]]
        $Node,
        [Parameter(Mandatory = $true)]
        [ValidateNotNullorEmpty()]
        [System.Management.Automation.PSCredential]
        $DomainAdmin,
        $hostname,
        $domain
    )

    Import-DscResource -ModuleName xComputerManagement
    Import-DscResource –ModuleName PSDesiredStateConfiguration

    Node $Node
    {

        xComputer JoinDomain
        {
            Name       = $hostname
            DomainName = $domain
            Credential = $DomainAdmin
            JoinOU = "OU=Servers,OU=Staging,DC=testdomain,DC=eu"
```

```
        }


        GroupSet LocalAdmins
        {
            GroupName = @( 'Administrators')
            Ensure = 'Present'
            MembersToInclude = @( 'testdomain-eu\dscstaging' )
        }
    }
}
$cd = @{
    AllNodes = @(
        @{
            NodeName = $Node
            PSDscAllowPlainTextPassword = $false
            PSDscAllowDomainUser=$true
            Certificatefile = $CertFile
            Thumbprint = $Certificate.ToString()
        }
    )
}


JoinAD -Node $Node -OutputPath $DscConfigPath -DomainAdmin $DomainAdmin -
hostname $hostname -ConfigurationData $cd -domain $domain

New-DscChecksum $DscConfigPath -Force
```

Invoke-Command -ComputerName $Node -ScriptBlock{Update-DscConfiguration -Verbose -Wait } -Credential $LocalAdmin -Verbose

Ось як виглядає наш mof-файл:

instance of MSFT_Credential as $MSFT_Credential1ref

{

Password = "-----BEGIN CMS-----
\nMIIBsgYJKoZIhvcNAQcDoIIBozCCAZ8CAQAxggFKMIIBRgIBADAuMBoxGDAWBgNVBAMMD1dJTi1H\nNFFKTFFQME4xNQIQOQN77pxew75HU6l7GPn99TANBgkqhkiG9w0BAQcwAASCAQAlhFf7Zs2gJbJEnc1DEK2yWbKcO+BEyD2cr6vKHdn\nQ9TrjvbysEOvYjT15o6MccwkMEwGCSqGSIb3DQEHATAdBglghkgBZQMEASoEEEdKJT+GX4IkPezR\nwYncyQiAIAFKxwJocH4ufRsq9L2Ipkp+VQCx2ljlwif6ac4X/PqG\n-----END CMS-----";

UserName = "testdomain.eu\\service_DomainJoin_001";

};

instance of MSFT_xComputer as $MSFT_xComputer1ref

{

ResourceID = "[xComputer]JoinDomain";

Credential = $MSFT_Credential1ref;

DomainName = "testdomain.eu";

SourceInfo = "C:\\Program Files\\WindowsPowerShell\\Scripts\\JoinAD.ps1::34::9::xComputer";

Name = "dsctest51";

JoinOU = "OU=Servers,OU=Staging,DC=testdomain,DC=eu";

ModuleName = "xComputerManagement";

ModuleVersion = "4.1.0.0";

ConfigurationName = "JoinAD"; }

ДОДАТОК Г ANSIBLE OPENSTACK

 # Setup according to Blogpost "Full Stack automation with Ansible and OpenStack".

Execute with "ansible-playbook ansible-openstack-blogpost.yml -c local -vv"# #

```
- name: Execute the Blogpost demo tasks
  hosts: localhost
  tasks:
  - name: Download cirros image
    get_url:
      url: http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
      dest: /tmp/cirros-0.3.4-x86_64-disk.img
  - name: Upload cirros image to openstack
    os_image:
      name: cirros
      container_format: bare
      disk_format: qcow2
      state: present
      filename: /tmp/cirros-0.3.4-x86_64-disk.img
- name: Create new keypair from current user's default SSH key
    os_keypair:
      state: present
      name: ansible_key
      public_key_file: "{{ '~' | expanduser }}/.ssh/id_rsa.pub"

  - name: Create the test network
    os_network:
```

```
    state: present
    name: testnet
    external: False
    shared: False
    #provider_network_type: vlan
    #provider_physical_network: datacentre
  register: testnet_network

- name: Create the test subnet
  os_subnet:
    state: present
    network_name: "{{ testnet_network.id }}"
    name: testnet_sub
    ip_version: 4
    cidr: 192.168.0.0/24
    gateway_ip: 192.168.0.1
    enable_dhcp: yes
    dns_nameservers:
      - 8.8.8.8
  register: testnet_sub

- name: Create the test router
  ignore_errors: yes #for some reasons, re-running this task gives errors
  os_router:
    state: present
    name: testnet_router
    network: nova
    external_fixed_ips:
      - subnet: nova
```

```
    interfaces:
      - testnet_sub
- name: Create a new security group
  os_security_group:
    state: present
    name: secgr
- name: Create a new security group allowing any ICMP
  os_security_group_rule:
    security_group: secgr
    protocol: icmp
    remote_ip_prefix: 0.0.0.0/0
- name: Create a new security group allowing any SSH connection
  os_security_group_rule:
    security_group: secgr
    protocol: tcp
    port_range_min: 22
    port_range_max: 22
    remote_ip_prefix: 0.0.0.0/0
- name: Create server instance
  os_server:
    state: present
    name: testServer
```

```
      image: cirros
      flavor: m1.small
      security_groups: secgr
      key_name: ansible_key
      nics:
        - net-id: "{{ testnet_network.id }}"
    register: testServer
- name: Show Server's IP
    debug: var=testServer.openstack.public_v4
```