OULUN SEUDUN
AMMATTIKORKEAKOULU

Henri Pitkänen

WINDOWS SERVER 2012 -PALVELIMEN HALLINTA POWERSHELL-KOMENNOILLA

WINDOWS SERVER 2012 -PALVELIMEN HALLINTA POWERSHELL-KOMENNOILLA

Henri Pitkänen
Opinnäytetyö
Syksy 2013
Tietojenkäsittelyn koulutusohjelma
Oulun seudun ammattikorkeakoulu

TIIVISTELMÄ

Oulun seudun ammattikorkeakoulu
Tietojenkäsittely

Tekijä: Henri Pitkänen
Opinnäytetyön nimi: Windows Server 2012 -palvelimen hallinta PowerShell-komennoilla
Työn ohjaaja: Jukka Kaisto
Työn valmistumislukukausi ja -vuosi: Syksy 2013 Sivumäärä: 39

Tämän opinnäytetyön tavoitteena oli perehtyä Windows Server 2012 -palvelimen ja aktiivihakemiston hallintaan käyttäen Windows PowerShell -komentotulkkia. PowerShell on Microsoftin kehittämä komentotulkki ja skriptausympäristö, jonka tarkoitus on tehostaa ja automatisoida Windows-käyttöjärjestelmien ja sovellusten hallintaa komentorivin ja skriptien avulla.

Opinnäytetyössä käsiteltiin Windows Server 2012 -käyttöjärjestelmään, aktiivihakemistoon ja Windows PowerShell -komentotulkkiin liittyviä ominaisuuksia. Lisäksi perehdyttiin PowerShell ISE -skriptausympäristöön. Opinnäytetyön lähteinä käytettiin kirjallisia ja sähköisiä teoksia.

Käytännön osuudessa käytiin läpi erilaisia Windows Server 2012 -käyttöjärjestelmään ja aktiivihakemistoon liittyviä hallintatoimenpiteitä. Windows Server 2012 -hallintaa käsiteltiin roolien asentamisen ja poistamisen, tapahtumalokien, palveluiden ja prosessien hallinnan sekä palvelimen sulkemisen ja uudelleenkäynnistämisen osalta. Aktiivihakemiston hallintaa käsiteltiin organisaatioyksiköiden, käyttöoikeusryhmien, tietokonetilien ja käyttäjätilien osalta.

Kaikki työssä tehdyt hallintatoimenpiteet tehtiin virtuaaliympäristössä. Virtuaaliympäristö toteutettiin VMware Player -ohjelmistolla. Virtuaaliympäristö muodostui kahdesta virtuaalikoneesta, joista toinen toimi palvelintietokoneena ja toinen työasemana. Palvelintietokoneelle asennettiin Windows Server 2012 -käyttöjärjestelmä. Lisäksi palvelimeen asennettiin aktiivihakemisto. Työasemalle asennettiin Windows 8 -käyttöjärjestelmä.

Opinnäytetyön tuloksena selvisi, että Windows-palvelimen ja aktiivihakemiston hallinta PowerShell-komennoilla on tehokasta. Todellinen tehokkuus saavutettiin kuitenkin skripteillä, kun hallittavia kohteita oli suuri määrä.

Asiasanat: aktiivihakemisto, hallinta, PowerShell, Windows Server 2012

ABSTRACT

Oulu University of Applied Sciences
Degree Programme in Business Information Systems

Author: Henri Pitkänen
Title of thesis: Managing Windows Server 2012 Using PowerShell
Supervisor: Jukka Kaisto
Term and year when the thesis was submitted: Autumn 2013 Number of pages: 39

The aim of this thesis was to study Windows PowerShell and use it to perform different management tasks on Windows Server 2012 operating system and Active Directory.
PowerShell is a command-line shell and scripting environment developed by Microsoft. The main purpose of PowerShell is to enhance and automate the management capabilities of Windows operating systems and the applications that run on Windows.

The theoretical background of the thesis concentrated on Windows Server 2012 operating system, Active Directory, PowerShell and PowerShell ISE scripting environment. The sources for the thesis were gathered both from literary work and internet sources.

The practical section of the thesis was comprised of different management tasks performed on Windows Server 2012 operating system and Active Directory using PowerShell.
The management tasks performed on Windows Server 2012 included installation and removal of roles, management of event logs, services and processes, and booting and rebooting of a server computer. The management tasks performed on Active Directory dealt with organizational units, security groups, computer accounts and user accounts.

All management tasks in the thesis were performed in a virtual environment. The virtual environment was implemented with VMware Player software. It consisted of two virtual machines, of which one acted as a server computer and the other one as a client computer. Windows Server 2012 operating system and Active Directory were installed on the server computer and Windows 8 operating system was installed on the client computer.

Results of this study revealed that it is efficient to manage Windows server and Active Directory with PowerShell. However, the true efficiency was achieved with scripts when working with large amount of manageable objects.

Keywords: Active Directory, management, PowerShell, Windows Server 2012

SISÄLLYS

1 JOHDANTO

Windows PowerShell on Microsoftin kehittämä komentotulkki ja skriptausympäristö. Se on luotu tehostamaan Windows-käyttöjärjestelmien ja sovellusten hallintaa sekä automatisointia komentorivin ja skriptien avulla. Tässä opinnäytetyössä perehdyttiin Windows Server 2012 -palvelimen ja aktiivihakemiston hallintaan käyttäen Windows PowerShell 3.0 -komentotulkkia. Tarkoituksena oli tehdä erilaisia Windows Server 2012 -palvelimeen ja aktiivihakemistoon liittyviä perus hallintatoimenpiteitä.

Opinnäytetyön teoriaosuudessa käsitellään Windows Server 2012 -käyttöjärjestelmän olennaisimmat uudistukset, saatavilla olevat versiot, Server Core -käyttöjärjestelmän, palvelinroolien ja niihin liittyvien ominaisuuksien tarkoitus. Kolmannessa luvussa käsitellään aktiivihakemiston ja siihen liittyvien komponenttien tarkoitus. Neljännessä luvussa käsitellään Windows PowerShell -komentotulkin käyttötarkoitus, versiohistoria, komentotyypit, tietoturvaan liittyvät seikat, putkitusmenetelmä sekä PowerShell ISE -työkalun käyttö. Käytännön osuudessa tehdään Windows Server 2012 -palvelimeen ja aktiivihakemistoon liittyviä hallintatoimenpiteitä.

Työtä varten toteutettiin virtuaaliympäristö VMware Player-ohjelmistolla. Virtuaaliympäristöön luotiin kaksi virtuaalikonetta, joista toinen toimi palvelintietokoneena ja toinen työasemana. Palvelintietokoneelle asennettiin Windows Server 2012 -käyttöjärjestelmä ja aktiivihakemisto. Työasemalle asennettiin Windows 8 -käyttöjärjestelmä.

2 WINDOWS SERVER 2012

Windows Server 2012 edustaa Windows Server -perheen kuudetta sukupolvea. Windows Server 2012 julkaistiin 1. päivä elokuuta 2012. Windows Server 2012 tuo mukanaan uusia ominaisuuksia ja muutoksia edeltäjäänsä, kuten uuden käyttöliittymän, uusia hallintatyökaluja, uuden version PowerShell-komentotulkista (3.0), uuden IP-osoitteiden hallintaan tarkoitetun roolin (IPAM) ja uuden ReFS-tiedostojärjestelmän.

Windows Server 2012 -käyttöjärjestelmästä julkaistiin R2-versio 18. päivä lokakuuta 2013, joka sisältää virtualisointiin ja levytilan hallintaan liittyviä parannuksia. Lisäksi mukana tulee uusi versio PowerShell-komentotulkista (4.0). (Tulloc 2012, 9; Petri, IT Knowledgebase 2013, hakupäivä 6.9.2013.)

2.1 Versiot

Windows Server 2012 -käyttöjärjestelmästä on saatavilla neljä eri versiota: Datacenter, Standard, Essentials ja Foundation. Jokainen versio on räätälöity tiettyä käyttötarkoitusta varten. Datacenter-versio on tarkoitettu pääosin raskaasti virtualisoituun ympäristöön, eikä virtuaalikoneiden määrää ole rajoitettu. Standard-versio on tarkoitettu vähäisesti virtualisoituun tai kokonaan virtualisoimattomaan ympäristöön. Standard-versiossa virtuaalikoneiden määrä on rajoitettu kahteen. Essentials-versio on tarkoitettu maksimissaan 25 käyttäjän pienyritysympärisöön. Se sisältää yksinkertaisemman käyttöliittymän ja valmiiksi konfiguroidun yhteyden pilvipalveluihin. Foundation-versio on tarkoitettu maksimissaan 15 käyttäjän yleispalvelinkäyttöön. Datacenter ja Standard versiot sisältävät kaikki Windows Server 2012 -käyttöjärjestelmän tarjoamat ominaisuudet, kun puolestaan Essentials ja Foundation versioissa ominaisuuksia on karsittu. (Microsoft 2013, hakupäivä 15.6.2013.)

2.2 Server Core

Server Core on riisuttu asennusvaihtoehto Windows Server 2012 -käyttöjärjestelmälle. Se ei sisällä Full-asennuksesta tuttua graafista käyttöliittymää. Sen sijaan pääkäyttöliitymänä toimii

komentotulkki. Server Core -asennus johtaa palvelinympäristöön, jota on tehokas hallita ja ylläpitää. Server Core -versio tarjoaa kuitenkin rajoitetusti toiminnallisuuksia verrattuna Windows Server -käyttöjärjestelmän Full-asennukseen. Server Core voidaan konfiguroida seuraaville yleisesti tunnetuille palvelinrooleille: File Server, DHCP Server, DNS Server, Media Services ja Active Directory. Server Core -asennuksen tuomia etuja ovat muun muassa vähäisempi huolto- ja hallinnan tarve, pienempi hyökkäyspinta-ala ja pienempi levytilan tarve. (Microsoft 2013, hakupäivä 18.6.2013; Rouse 2008, hakupäivä 15.11.2013.)

Server Core -käyttöjärjestelmän asennuksessa asennetaan ainoastaan asennettavien roolien vaatimat välttämättömät binäärit. Esimerkiksi DNS (Domain Name System) -roolin asennuksessa jätetään Windows Internet Explorer -sovellus kokonaan asentamatta, koska se ei ole välttämätön roolin toimivuuden kannalta.

Server Core -käyttöjärjestelmää voidaan hallita paikallisesti ja etänä. Hallintaan voidaan käyttää muun muassa PowerShell-komentotulkkia, RDP (Remote Desktop Protocol) -yhteyttä tai MMC (Microsoft Management Console) -työkalua. (Microsoft 2013, hakupäivä 18.6.2013.)

2.3 Roolit, roolipalvelut ja ominaisuudet

Windows-palvelinroolilla tarkoitetaan joukkoa sovelluksia (palveluita), jotka mahdollistavat palvelimen tekemään tiettyjä toimintoja verkon käyttäjille ja tietokoneille. Palvelin on tyypillisesti varattu vain tiettyä roolia varten. Pienemmissä kokoonpanoissa palvelin voi olla varattu suorittamaan useampaa roolia samaan aikaa. Palvelimella voi olla yksi tai useampi rooli ja jokaisella roolilla voi olla yksi tai useampi roolipalvelu (role service). (Carpenter 2011, 9.)

Esimerkiksi File Services -rooli Windows Server 2012 -käyttöjärjestelmässä sisältää seuraavia eri palveluita:

- File Server
- BranchCache for Network Files
- Data Deduplication
- DFS Namespaces
- DFS Replication

- File Server Resouce Management
- File Server VSS Agent Service
- iSCSI Target Server
- Server for NFS

(Morimoto, Noel, Yardeni, Droubi, Abbate & Amaris 2012, 1132.)

Palvelinroolien lisäksi on mahdollista lisätä myös ominaisuuksia (features) Windows Server -asennukseen. Ominaisuudet ovat ohjelmistosovelluksia, jotka tukevat tai lisäävät yhden tai useamman palvelinroolin toiminnallisuutta tai yleistä palvelimen toiminnallisuutta, riippumatta siitä, mitä rooleja palvelimelle on asennettu. Esimerkiksi Failover Clustering -ominaisuudella voidaan edistää File Services ja DHCP Server -roolien toiminnallisuutta mahdollistamalla niiden liittymisen palvelinklustereihin. (Carpenter 2011, 9-10; Microsoft 2013, hakupäivä 22.6.2013.)

3 AKTIIVIHAKEMISTO

Aktiivihakemisto (Active Directory) on Microsoftin toteuttama keskitetty ja standardoitu hakemistopalvelu, jonka avulla voidaan mahdollistaa verkkoresurssien hallinta hajautettujen resurssien, käyttäjädatan ja turvallisuuden osalta. Aktiivihakemisto on tarkoitettu erityisesti hajautettuihin verkkoympäristöihin. Aktiivihakemistoon tallennetaan tietoa verkon resursseista, kuten käyttäjistä, käyttäjäryhmistä, työasemista, tulostimista, sovelluksista ja palveluista. (Desmond, Richards, Allen & Lowe-Norris 2013, 1; Rouse 2008, hakupäivä 24.6.2013.)

3.1 Toimialue

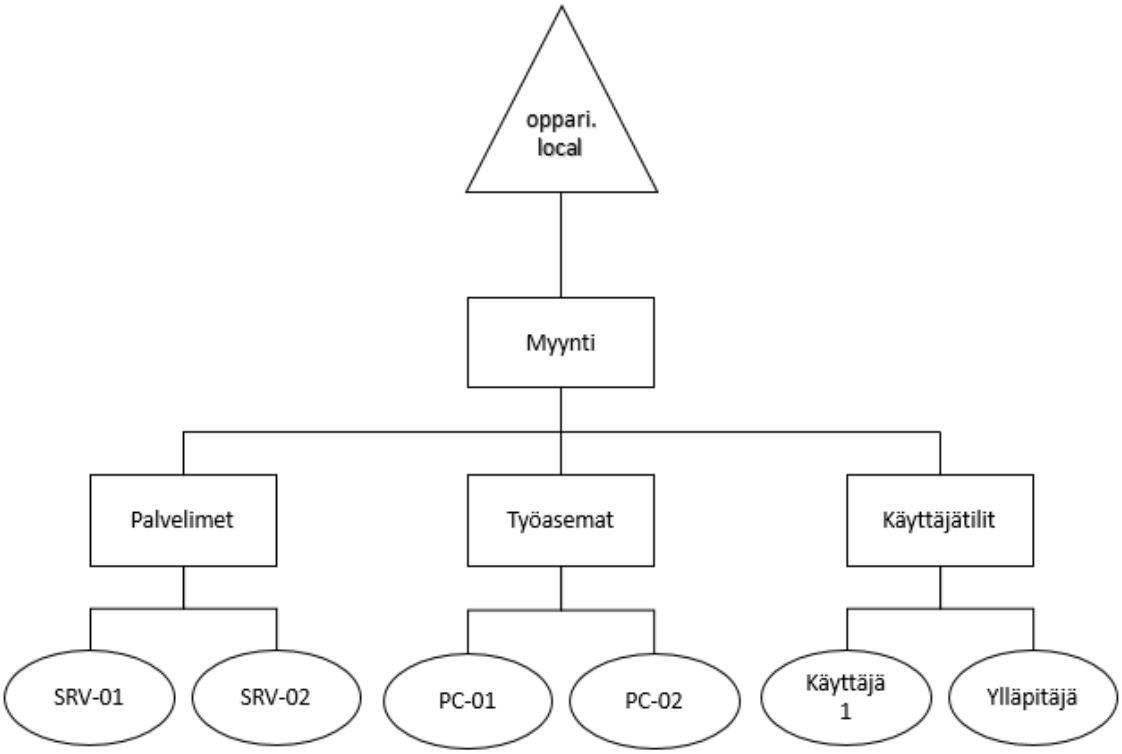
Toimialue (domain) on kokoelma verkkoympäristössä tarvittavia käyttäjätilejä, tietokonetilejä ja muita objekteja, jotka ovat liitetty yhteen. Niitä voidaan hallita keskitetysti yhdeltä tai useammalta Windows-toimialueen ohjauskoneelta (domain controller). Jokaisella toimialueella on ohjauskoneita, jotka pitävät sisällään käyttäjätietokannan ja kaikki käyttäjien todentamiseen liittyvät tiedot. Toimialueen käyttäjillä on rajattu pääsy kaikkiin toimialueen resursseihin, riippuen mitä oikeuksia käyttäjälle on määritetty.

Aktiivihakemiston toimialueet nimetään käyttäen DNS-nimeämiskäytäntöä. Toimialueita voidaan ryhmitellä jos ne kuuluvat samaan DNS-nimiavaruuteen ja samaan toimialuepuuhun. (Rhodes-Ousley 2013, 527; Svidergol & Allen 2013, 11; Posey 2006, hakupäivä 29.8.2013.)

3.2 Organisaatioyksikkö

Toimialueen objekteja voidaan järjestää loogisiin kokonaisuuksiin organisaatioyksiköiden (organizational unit) avulla. Tämä mahdollistaa joustavan hallinnan aktiivihakemiston sisällä ja tarjoaa useita eri mahdollisuuksia käyttöoikeuksien hajautettuun hallinnointiin. Saman toimialueen sisällä voi olla useita eri organisaatioyksiköitä, joille jokaiselle voi olla määritetty oma ylläpitäjä tai ryhmä ylläpitäjiä. (Alpern & Muller 2012, 240-241.)

Kuvion 1 esimerkissä on luotu organisaatioyksikkö ”Myynti” toimialueeseen ”oppari.local”, jonka sisään on luotu kolme organisaatioyksikköä: ”Palvelimet”, ”Työasemat” ja ”Käyttäjätilit”. Palvelimet organisaatioyksikön sisään on luotu kaksi tietokonetiliä: ”SRV-01” ja ”SRV-02”. Organisaatioyksikön ”Työasemat” sisään on luotu tietokonetilit ”PC-01” ja ”PC-02”. Organisaatioyksikön ”Käyttäjätilit” sisään on luotu kaksi käyttäjätiliä: ”Käyttäjä 1” ja ”Ylläpitäjä”.



KUVIO 1. Organisaatioyksiköt järjestetty loogisiin kokonaisuuksiin

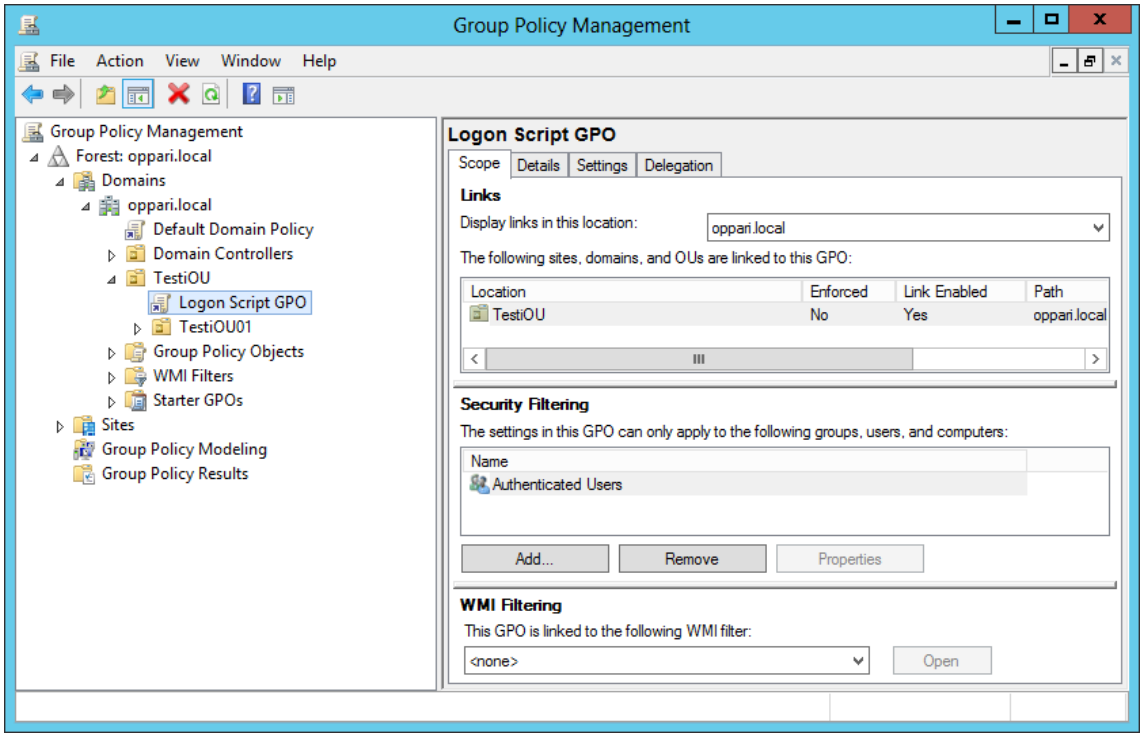
3.3 Ryhmäkäytäntö

Ryhmäkäytäntöjen (group policy) avulla ylläpitäjän on mahdollista määrittää käyttäjiin ja tietokoneisiin liittyvät asetukset toimipaikoille (sites), toimialueille ja organisaatioyksiköille. Ryhmäkäytönnöillä voidaan määrittää, mitä käyttäjät voivat tehdä verkossa, kuten esimerkiksi mihin tiedostoihin, kansioihin tai sovelluksiin heillä on oikeus päästä. (Rouse 2012, hakupäivä 28.7.2013.)

Ylläpitäjän määrittämät asetukset sijaitsevat ryhmäkäytäntöobjekteissa (group policy object). Ryhmäkäytöntöjä voidaan hallita ryhmäkäytäntöhallintakonsolilla (Group Policy Management

Console), paikallisella ryhmäkäytäntöeditorilla (gpedit) tai PowerShell-komennoilla. (Rouse 2012, hakupäivä 28.7.2013.)

Ryhmäkäytäntöhallintakonsoli ja siihen liittyvä PowerShell-moduuli asennetaan aktiivihakemiston asennuksen yhteydessä. Kuviossa 2 on esitetty ryhmäkäytäntöhallintakonsolin käyttöliittymä. (Microsoft 2013, hakupäivä 15.10.2013.)



KUVIO 2. Ryhmäkäytäntöhallintakonsoli

4 WINDOWS POWERSHELL

Windows PowerShell on Microsoftin kehittämä komentotulkki ja skriptausympäristö. Sen tavoitteena on tehdä Windows-pohjaisten järjestelmien ja sovelluksien hallinta mahdollisimman tehokkaaksi sekä paikallisesti että etänä. Windows PowerShell -konsoliin voidaan kirjoittaa komentoja joko yksi kerrallaan tai halutessa voidaan luoda komennoista muodostuva skriptitiedosto. Skriptien avulla voidaan automatisoida hallintaan liittyviä tehtäviä. PowerShell-konsoliin voidaan syöttää yksittäinen komento, esimerkiksi *Create-ADUser*, jonka avulla luodaan uusi käyttäjätili aktiivihakemistoon. Skriptillä voidaan kutsua *Create-ADUser*-komentoa moneen kertaan ja luoda useita käyttäjätilejä samanaikaisesti. Komentojen kirjoittaminen käsin vähenee, joten hallinnasta tulee erityisen tehokasta. (Lee, Mitschke, Schill & Tanasovski 2011, 6-7.)

4.1 Versiohistoria

PowerShell-komentotulkista on julkaistu neljä eri versiota. Ensimmäinen versio (1.0) julkaistiin vuonna 2006 Windows Vista- ja Windows Server 2008 -käyttöjärjestelmien yhteydessä. Se on yhteensopiva Windows XP- ja Windows Server 2003 -käyttöjärjestelmien kanssa. (Microsoft 2013, hakupäivä 16.10.2013.)

Seuraava julkaisu, PowerShell versio 2.0, asennetaan oletuksena Windows 7:n ja Windows Server 2008 R2:n asennuksen yhteydessä. Lisäksi se on saatavilla Windows XP-, Windows Server 2003-, Windows Vista-, ja Windows Server 2008 -käyttöjärjestelmille. PowerShell 2.0 on taaksepäin yhteensopiva PowerShell 1.0:n kanssa. PowerShell 2.0 tuo edeltäjäänsä verrattuna paljon uusia ominaisuuksia ja komentoja kuten, ISE-skriptausympäristön, etähallintaan liittyvän PowerShell Remoting -ominaisuuden ja taustalla ajettavat tehtävät. (Microsoft 2013, hakupäivä 16.10.2013.)

PowerShell 3.0 -versio asennetaan oletuksena Windows 8:n ja Windows Server 2012:n asennuksen yhteydessä ja on lisäksi saatavilla Windows 7 (Service Pack 1)-, Windows Server 2008 (Service Pack 2)-, ja Windows Server 2008 R2 (Service Pack 1) -käyttöjärjestelmille. Se on taaksepäin yhteensopiva PowerShell 2.0:n kanssa, mutta ei ole yhteensopiva Windows XP-,

Windows Server 2003- ja Windows Vista -käyttöjärjestelmien kanssa. PowerShell 3.0 tuo mukanaan muun muassa ajastetut tehtävät ja komentojen automaattisen täydentämisen (IntelliSense). (Microsoft 2013, hakupäivä 16.10.2013.)

Viimeisin versio, PowerShell 4.0 tulee Windows Server 2012 R2- ja Windows 8.1 -käyttöjärjestelmien mukana. PowerShell 4.0 on lisäksi saatavilla Windows 7 (Service Pack 1)-, Windows Server 2008 R2 (Service Pack 1)- ja Windows Server 2012 -käyttöjärjestelmille. PowerShell 4.0 tuo mukanaan uusia cmdlet-komentoja ja ominaisuuksia. Lisäksi oletus Execution Policy -asetus on vaihdettu aikaisemmasta Restricted-tilasta, RemoteSigned-tilaan. (Microsoft 2013, hakupäivä 16.10.2013.)

4.2 Komentotyypit

PowerShell sisältää neljän tyyppisiä komentoja: cmdlet (command-let) -komentoja, shell-funktiokomentoja, skriptikomentoja ja natiivikomentoja. PowerShell-komento koostuu varsinaisesta komennon nimestä, komentoon liittyvistä parametreista ja parametreihin liittyvistä argumenteista. Yksinkertaisen komennon ulkoasu voi näyttää esimerkiksi tältä:

komento -parametri1 -parametri2 argumentti1 argumentti2

Komennon ensimmäinen osa on itse suoritettavan komennon nimi. PowerShell-komentotulkki tulkitsee syötetyn komennon sekä komennon tyypin ja päättää sen perusteella, mitä tehdään. Komennon nimen perään voidaan syöttää parametreja ja/tai argumentteja. Parametrin nimen eteen tulee aina yhdysmerkki. Argumentti puolestaan on arvo, joka on yhteydessä tai sidoksissa tiettyyn parametriin. (Payette 2011, 39.)

Cmdlet-komento on komento, joka suorittaa jonkin tietyn toiminnon. Toiminto voi olla esimerkiksi tiedoston poistaminen, käyttäjän lisääminen aktiivihakemistoon tai rekisterin muokkaaminen. Cmdlet-komennot nimetään verbi-substantiivi-käytännöllä, missä verbillä määritetään suoritettava toiminto ja substantiivilla määritetään, mihin olioon toiminto liittyy. Esimerkiksi cmdlet-komennolla *Get-Process* voidaan tarkastella kaikkia tietokoneessa tällä hetkellä käynnissä olevia prosesseja, kuten kuviossa 3 on esitetty. *Get-Process*-komennolle voidaan määrittää parametri *Name*, kun

halutaan tietoa vain tietystä prosessista. Kuviossa 4 tarkastellaan ”explorer”-prosessin tietoja *Name*-parametrin avulla. (Lee ym. 2011, 8-9.)

```
PS C:\> Get-Process

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    553      73    11600      18456   119     0,23   3076 AsusAudioCenter
    189      12    16448      17520    51            4476 audiodg
    287      18     6472      13508    90     1,37   2872 ComUpdatus
     48       6     1756       4976    60     0,00   2944 conhost
```

KUVIO 3. Get-Process-komennon palauttamat prosessit

```
PS C:\> Get-Process -Name explorer

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    793      55    49300      72364   317    14,12   2584 explorer
```

KUVIO 4. Name-parametrin käyttö Get-Process-komennon yhteydessä

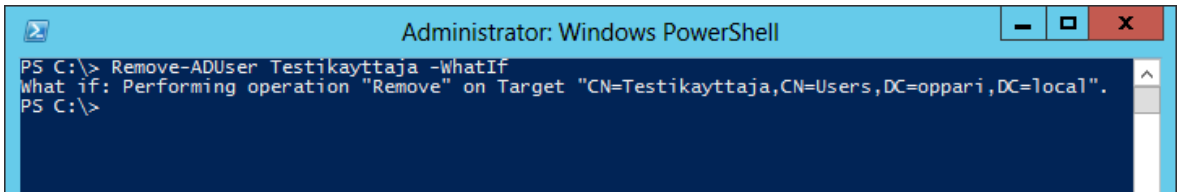
Cmdlet-komennot tukevat aliaksia eli pikakomentoja, jotka muun muassa yksinkertaistavat ja nopeuttavat komentojen kirjoittamista. Esimerkiksi komento *GPS* on pikakomento *Get-Process* komennolle (kuvio 5). PowerShell tuo mukanaan useita valmiita pikakomentoja käytettäväksi, mutta pikakomentoja on mahdollista tehdä myös itse. (Lee ym. 2011, 8-9.)

```
PS C:\> GPS

Handles  NPM(K)    PM(K)      WS(K) VM(M)   CPU(s)     Id ProcessName
-------  ------    -----      ----- -----   ------     -- -----------
    553      73    11600      18456   119     0,23   3076 AsusAudioCenter
    189      12    16448      17520    51            4476 audiodg
    287      18     6472      13508    90     1,37   2872 ComUpdatus
     48       6     1756       4976    60     0,00   2944 conhost
```

KUVIO 5. GPS-komento toimii aliaksena Get-Process-komennolle

Kaikki cmdlet-komennot tukevat niin sanottuja yleisiä parametreja, jotka toimivat kaikkien cmdlet-komentojen kanssa. Voidaan esimerkiksi käyttää *WhatIf*-parametria, jos halutaan testata, mitä toimenpiteitä jokin komento tai komentorivi tekisi, jos se suoritettaisiin. Kuvion 6 esimerkissä käytetään *Remove-ADUser*-komentoa *WhatIf*-parametrin kanssa. Vastauksena saadaan tuloste, joka kertoo, mitä toimenpiteitä komento suorittaisi ja mihin kohteisiin se vaikuttaisi. Komentoa ei kuitenkaan suoriteta loppuun. *(*Microsoft 2013, hakupäivä 2.12.2013.)



KUVIO 6. WhatIf-parametrin palauttama tuloste Remove-ADUser-komennon yhteydessä

Taulukossa 1 esitetään kaikki yleiset parametrit ja niiden tarkoitus. Jokaisen parametrin pikakomento on lisätty parametrin nimen perään sulkuihin.

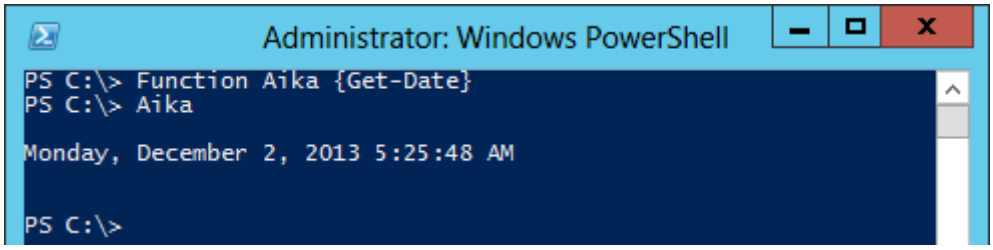
Parametri	Tarkoitus
Verbose (vb)	Näytetään lisätietoja suoritettavasta komennosta
Debug (db)	Näytetään yksityiskohtaista ohjelmoijatason tietoa suoritettavasta komennosta
WarningAction (wa)	Määritetään miten toimitaan, jos toiminnon suorittamisessa ilmenee varoitusviesti
WarningVariable (wv)	Tallentaa mahdolliset komennon suorituksessa ilmenevät varoitusilmoitukset muuttujaan
ErrorAction (ea)	Määritetään miten toimitaan, jos toiminnon suorittamisessa ilmenee virhe
ErrorVariable (ev)	Tallentaa mahdolliset komennon suorituksessa ilmenevät virheilmoitukset muuttujaan
OutVariable (ov)	Tallentaa komennon tulostetiedot muuttujaan
OutBuffer (ob)	Määrittelee kuinka monta oliota puskuroidaan ennen kuin oliot lähetetään putken (pipeline) läpi
WhatIf (wi)	Tulostaa viestin, jossa kerrotaan, mitä tapahtuisi, jos komento suoritettaisiin. Komentoa ei kuitenkaan suoriteta
Confirm (cf)	Kehottaa käyttäjää vahvistamaan suoritettavan komennon ennen komennon lopullista suorittamista

TAULUKKO 1. Cmdlet-komentojen yleiset parametrit

Funktiokomento on käyttäjän määrittämä nimetty koodilohko, jota voidaan kutsua nimen perusteella esimerkiksi skriptin sisällä. Funktion syntaksi koostuu avainsanasta, annetusta funktion nimestä, mahdollisista parametreista ja skriptilohkosta:

Function(avainsana) FunktionNimi(parametrit) {skriptilohko}

Seuraavassa esimerkissä luodaan yksinkertainen funktio, joka kutsuu *Get-Date* cmdlet-komentoa. Kuviossa 7 luodaan *Aika*-niminen funktio ja kutsutaan sitä nimen perusteella.



KUVIO 7. Luodaan Aika-funktio ja kutsutaan sitä nimen perusteella

Funktiot säilyvät muistissa komentotulkin ajon ajan ja pyyhitään muistista, kun komentotulkki suljetaan. Funktion koodi käännetään määrittelyhetkellä. Käännetty koodi säilötään, jolloin sitä ei tarvitse joka käytön yhteydessä kääntää uudelleen. (Payette 2011, 43.)

Skriptikomento on tekstitiedostoon tallennettu osa koodia. Skriptikomennot ladataan ja käännetään joka kerta, kun ne ajetaan, jonka johdosta niiden käynnistäminen on hieman hitaampaa kuin funktiokomennoilla. Funktiot ja skriptit ovat parametreihin liittyvillä ominaisuuksiltaan samanlaisia. Skriptiä kutsutaan sille annetun tiedostonimen perusteella. PowerShell-skriptin tunnistaa .ps1-tiedostopäätteestä. (Payette 2011, 44.)

Kuviossa 8 on kuvattu esimerkkiskripti KotihakemistotOU.ps1, joka luo kotihakemistot kaikille TestiOU-organisaatioyksikön käyttäjätileille, joilta kotihakemisto puuttuu. Jokaiselle kotihakemistolle määritetään lisäksi halutut käyttöoikeudet.

KotihakemistotOU.ps1*

```
$ou ="TestiOU"
$homeDir = "\\SRV-01\kotihakemistot"
$tilit = get-ADUser -filter * -SearchBase  "OU=$TestiOU,DC=oppari,DC=local" -properties samaccountname
foreach ($tili in $tilit)
{
    $sso = $tili.SamAccountName
    # If the folder for the user does not exist, make a new one and set the correct permissions.
    if ( (Test-Path "$homeDir\$sso") -eq $false)
    {
        try
            {
        $NewFolder = New-Item -Path $homeDir -Name $sso -ItemType "Directory"
        $Rights = [System.Security.AccessControl.FileSystemRights]"FullControl,Modify,ReadAndExecute,Li
        $InheritanceFlag = [System.Security.AccessControl.InheritanceFlags]::None
        $PropagationFlag = [System.Security.AccessControl.PropagationFlags]::None
        $objType =[System.Security.AccessControl.AccessControlType]::Allow
        $objUser = New-Object System.Security.Principal.NTAccount "oppari.local\$sso"
        $objACE = New-Object System.Security.AccessControl.FileSystemAccessRule `
                ($objUser, $Rights, $InheritanceFlag, $PropagationFlag, $objType)
        $ACL = get-acl -Path $NewFolder
        $ACL.AddAccessRule($objACE)
        $objReturn = Set-ACL -Path "$homeDir\$sso" -AclObject $ACL
    $objReturn
Get-ADUser -Identity $tili |
Set-ADUser -HomeDirectory "$homedir\$sso" -HomeDrive "k:"

            }
    catch
        {
        $msg = $_
        $msg
     }
    }
}
```

KUVIO 8. KotihakemistotOU.ps1-skriptitiedosto

Natiivikomento on ulkoinen, tyypillisesti .exe tiedostopäätteinen ohjelma, jonka käyttöjärjestelmä voi suorittaa. Natiivikomennot ajetaan aina uudessa prosessissa, joka tekee natiivikomennoista hitaimman komentotyypin suorittaa. Natiivikomento voi olla mikä tahansa ohjelma, mikä on ajettavissa Windows-käyttöjärjestelmällä. Natiivikomentoja käytetään kirjoittamalla suoritettavan ohjelman nimi PowerShell-komentoriville, esimerkiksi notepad.exe. (Payette 2011, 44-45.)

4.3 Tietoturva

PowerShell-komentotulkin päällimmäinen tarkoitus on suorittaa skriptejä, joilla voidaan automatisoida järjestelmänhallintaan liittyviä tehtäviä. Voidaan sanoa, ettei sellaista asiaa kuin luonnostaan turvallinen PowerShell-skripti ole olemassakaan. PowerShell ei tunne niin sanottua ”hiekkalaatikko” käsitettä, jolla tarkoitetaan komentojen suorittamista turvallisessa rajatussa ympäristössä. PowerShell-skriptejä pitää siksi käsitellä kuin ne olisivat suoritettavia sovelluksia. Sen vuoksi PowerShell-komentotulkin asennusvaiheessa tehdään toimenpiteitä, joilla luodaan mahdollisimman turvallinen oletusympäristö. (Payette 2011, 897.)

4.3.1 Execution Policy

PowerShell-skriptien suorittaminen on oletuksena estetty. Skriptien suorittamista hallitaan Execution Policy -asetuksen avulla. On kuitenkin tärkeää ymmärtää, että Execution Policy -asetusta ei ole tarkoitettu estämään PowerShell -komentotulkin käyttöä kokonaisuudessaan, vaan estää tuntemattomien ja mahdollisesti haittaa aiheuttavien skriptien tahaton suorittaminen. PowerShell versio 1.0 sisältää neljä Execution Policy -asetusta: Restricted, AllSigned, RemoteSigned ja Unrestricted. PowerShell versio 2.0 lisäsi kaksi uutta asetusta: Bypass ja Undefined. Execution Policy -asetus asetetaan cmdlet-komennolla *Set-ExecutionPolicy <asetuksen nimi>*. Tällä hetkellä käytössä oleva Execution Policy -asetus saadaan selville komennolla *Get-ExecutionPolicy*. Taulukossa 2 esitetään jokaisen Execution Policy -asetuksen kuvaus. (Payette 2011, 898-900.)

Asetus	Kuvaus
Restricted	Oletusasetus. Kun tämä asetus on käytössä, skriptien suorittaminen on estetty. Tietoturvan kannalta turvallisin asetus, mutta vaikeuttaa huomattavasti PowerShell-automatisointia.
AllSigned	Skriptejä voidaan suorittaa ainoastaan, jos ne ovat Authenticode-allekirjoitettuja. Tietoturvan kannalta turvallinen asetus, mutta vaikeuttaa skriptien luontiprosessia.
RemoteSigned	Paikallisesti luotuja skriptejä voidaan suorittaa vapaasti, mutta ulkopuolisesta lähteestä ladatut skriptit pitää olla Authenticode-allekirjoitettuja.
Unrestricted	Kaikki skriptit ovat sallittuja. Tietoturvan kannalta vähiten turvallinen asetus.
Bypass (2.0+)	Mitään ei ole estetty. Tämä asetus on suunniteltu ympäristöön, missä PowerShell-skriptit ovat osa suurempaa sovellusta.
Undefined (2.0+)	Poistaa voimassa olevan Execution Policy -asetuksen nykyiseltä näkyvyysalueelta (scope).

TAULUKKO 2. Execution Policy -asetusten kuvaukset

4.3.2 Skriptien allekirjoittaminen

Skriptien allekirjoittamisella tarkoitetaan prosessia, missä skriptitiedostoon lisätään tietoa, jolla skriptin julkaisija voidaan tunnistaa turvallisella tavalla. Turvallisella tavalla tarkoitetaan sitä, että voidaan luotettavasti todentaa, että skriptin on varmasti tuottanut luotettava organisaatio ja että skriptin sisältöä ei ole muutettu millään tavalla allekirjoittamisen jälkeen. Allekirjoitettu skripti ei automaattisesti takaa skriptin turvallisuutta. Jos skripti on kuitenkin allekirjoitettu luotettavan tahon

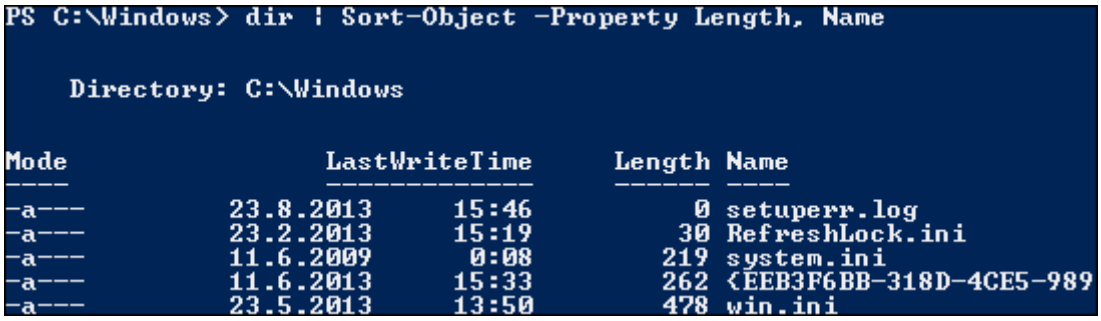
puolesta, voidaan skriptin tekijä selvittää allekirjoituksen perusteella. Jos skripti sisältää esimerkiksi haittaohjelman, voidaan skriptin alkuperäinen tekijä selvittää helposti. (Payette 2011, 904.)

Skriptien allekirjoittamiseen liittyviä sertifikaatteja on kahden tyyppisiä. Sertifikaatteja, jotka on luotu luotettavan tahon puolesta (certificate authority) ja sertifikaatteja, jotka käyttäjä on luonut itse (self-signed certificate). Käyttäjä voi luoda tietokonekohtaisen sertifikaatin itse silloin, kun ei ole tarvetta hankkia maksullista sertifikaattia ulkopuoliselta taholta. Käyttäjän luomalla sertifikaatilla voidaan hallita, mitä skriptejä tietokoneella voidaan suorittaa. Muut tietokoneet eivät kuitenkaan pidä tätä sertifikaattia luotettavana, joten skriptejä, jotka ovat allekirjoitettu tällä sertifikaatilla, ei suoriteta. (Payette 2011, 905-906.)

Skriptien allekirjoittaminen toteutetaan käyttämällä kahta tekniikkaa: julkisen avaimen salausta (public key encryption) ja yksisuuntaisia tiivistefunktioita (one-way hash function). Julkisen avaimen salauksessa käytetään kahta eri avainta: yksityistä avainta (private key) tiedon salaukseen ja julkista avainta (public key) salatun tiedon purkamiseen. Skriptin sisältö ja digitaalinen allekirjoitus salataan allekirjoittajan yksityisellä avaimella (private key), eli sisällöstä luodaan yksisuuntainen tiiviste. Vastaanottaja käyttää allekirjoittajan julkista avainta (public key) tiivisteen purkamiseen. Tämän jälkeen vastaanottaja käyttää samaa salausalgoritmia, jolla alkuperäinen tiiviste luotiin ja laskee saamastaan sisällöstä oman tiivisteen. Vastaanottaja vertaa omaa tiivistettä alkuperäiseen tiivisteeseen. Jos tiivisteet vastaavat toisiaan, voidaan todeta, että sisältö on pysynyt muuttumattomana allekirjoittamisen jälkeen. (Payette 2011, 904.)

4.4 Putkitus

Putkitus (pipelining) on toiminta, jossa Windows PowerShell ottaa tulosteen yhdestä komennosta ja lähettää sen syötteenä toiselle komennolle. Käyttämällä putkijonoa (pipeline), voidaan komentoja ketjuttaa keskenään. Putkituksen avulla voidaan esimerkiksi tulostaa hakemiston tiedostolistaus ja järjestää lista tiedostokoon mukaan käyttämällä hyödyksi kahta erillistä komentoa. Tarvitaan yksi komento tulostamaan tiedostolistaus ja toinen komento, joka järjestää listan halutulla tavalla (kuvio 9). Komentoja ketjutetaan keskenään käyttämällä ”|”-symbolia. (Microsoft 2013, hakupäivä 4.9.2013.)



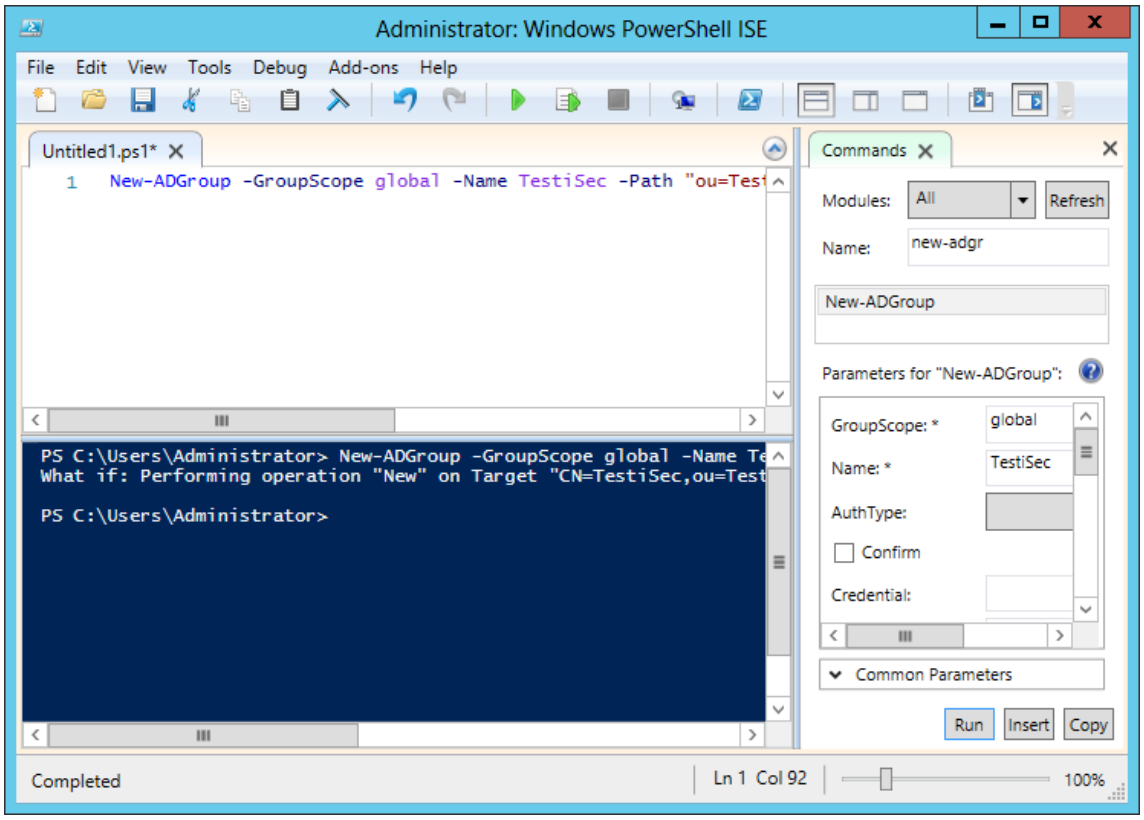
KUVIO 9. Putkituksen käyttö tiedostolistauksen yhteydessä

4.5 PowerShell ISE

PowerShell ISE (Integrated Scripting Environment) on työkalu, joka on luotu pääasiassa helpottamaan PowerShell-skriptausta. Ensimmäinen versio ISE-työkalusta julkaistiin PowerShell version 2.0 mukana. ISE tuo mukanaan samoja ominaisuuksia, joita löytää muistakin nykyaikaisista skriptausympäristöistä, kuten mahdollisuuden rinnakkain ajettaviin istuntoihin, erillisiin välilehtiin ja mahdollisuuden lisätä omia toiminnallisuuksia skriptausympäristöön. (Payette 2011, 607.)

4.5.1 Käyttöliittymä ja pikanäppäimet

ISE:n oletuskäyttöliittymä koostuu neljästä eri osasta: editorista, konsolista, Commands-lisäosasta ja työkalurivistä (kuvio 10). Editorissa luodaan ja/tai muokataan skriptitiedostoja, konsoli sisältää itse PowerShell-komentotulkin ja Commands-lisäosa kaikkien PowerShell-komentojen kuvaukset ja parametrit. Commands-lisäosan avulla voidaan kopioida tai sijoittaa komentoja suoraan editoriin ja konsoliin. Työkalurivi tarjoaa nopean pääsyn yleisiin toimintoihin, kuten kopioimiseen, leikkaamiseen ja liittämiseen. Työkalurivi sisältää myös PowerShell-komentotulkkiin ja ISE:hen liittyviä erityistoimintoja, kuten painikkeet skriptien suorittamiselle, etäyhteysvälilehden luonnille ja PowerShell.exe-isäntäkonsolin käynnistämiselle. Työkalurivi sisältää lisäksi erilliset painikkeet ISE:n ikkuna-asettelun muotoilulle. (Payette 2011, 607-609.)



KUVIO 10. PowerShell ISE 3.0:n oletuskäyttöliittymä

PowerShell ISE:n käyttöliittymä tukee muiden Windows-pohjaisten sovellusten tapaan CUA (Windows Common User Access) -pikanäppäimiä, jolloin esimerkiksi Ctrl+C (kopioi) ja Ctrl+V (liitä) -näppäinyhdistelmien käyttö on mahdollista ISE:n ikkunoiden sisällä. ISE sisältää yleisien pikanäppäimien lisäksi myös ISE-kohtaisia pikanäppäimiä. Nämä pikanäppäimet esitetään taulukossa 3. (Payette 2011, 610-612.)

Pikanäppäin	**Kuvaus**
Ctrl+F	Avaa haku-toiminnon. Aloittaa haun nykyisestä kursorin sijainnista eteenpäin.
Shift+F3	Hakee edellisen hakutuloksen haku-toimintoa käytettäessä.
Ctrl+H	Avaa merkkijonon korvaus -toiminnon.
Ctrl+G	Siirrytään halutulle rivinumerolle nykyisessä tiedostossa.
F5	Suorittaa valittuna olevan ikkunan koko sisällön. Tiedosto tallennetaan ennen suorittamista.
F8	Suorittaa vain erikseen maalatun osan sisällöstä.

Ctrl+Shift+P	Käynnistää PowerShell.exe-isäntäkonsolin
Ctrl+N	Avaa uuden välilehden.
Ctrl+R	Sulkee ja/tai avaa editori-ikkunan
Ctrl+1	Sijoittaa editori-ikkunan konsoli-ikkunan yläpuolelle.
Ctrl+2	Sijoittaa editori-ikkunan oikealle.
Ctrl+3	Suurentaa editori-ikkunan.
Ctrl+T	Avaa uudeen PowerShell-istunto välilehden
Ctrl+W	Sulkee nykyisen PowerShell-istunto välilehden
Ctrl+Shift+R	Avaa uuden PowerShell-etäistunto välilehden
Enter	Luo uuden rivin editori-ikkunassa, suorittaa komennon konsoli-ikkunassa.
Shift+Sarkain	Jos useampi kuin yksi rivi on valittu editorissa, liikuttaa tekstin edelliseen sarkainkohtaan.
Sarkain	Jos mitään ei ole valittu, liikutaan seuraavaan sarkainkohtaan. ISE luo oletuksena neljä välilyöntiä. Jos useampi kuin yksi rivi on valittu, liikuttaa kaikki rivit seuraavaan sarkainkohtaan.

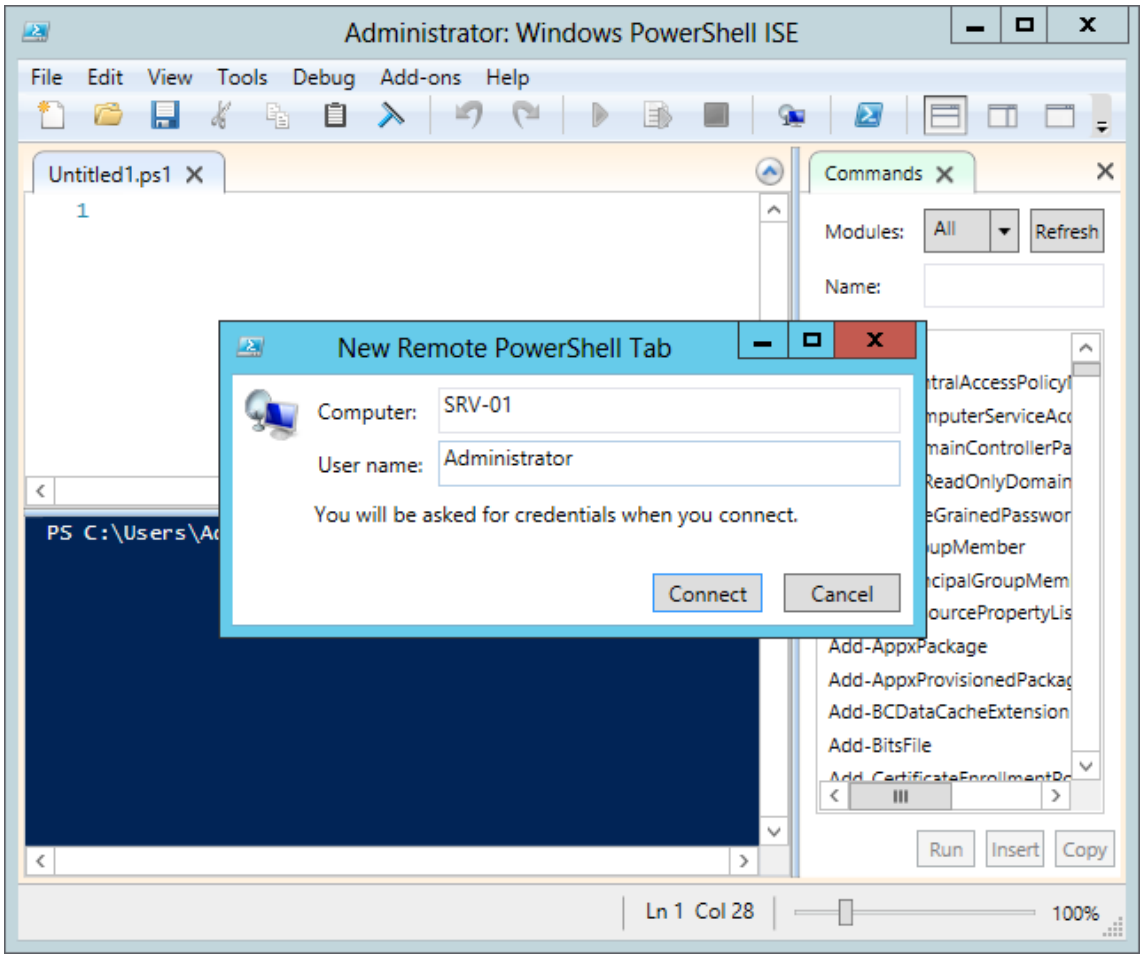
Taulukko 3. ISE-kohtaiset pikanäppäimet

4.5.2 Etäkäyttö

Etäkäytöllä tarkoitetaan toimintaa, jossa otetaan yhteys toiseen tietokoneeseen verkon välityksellä. Etähallinnan avulla voidaan tehdä eri hallintatoimenpiteitä kaikille verkon tietokoneille yhdeltä työpisteeltä.

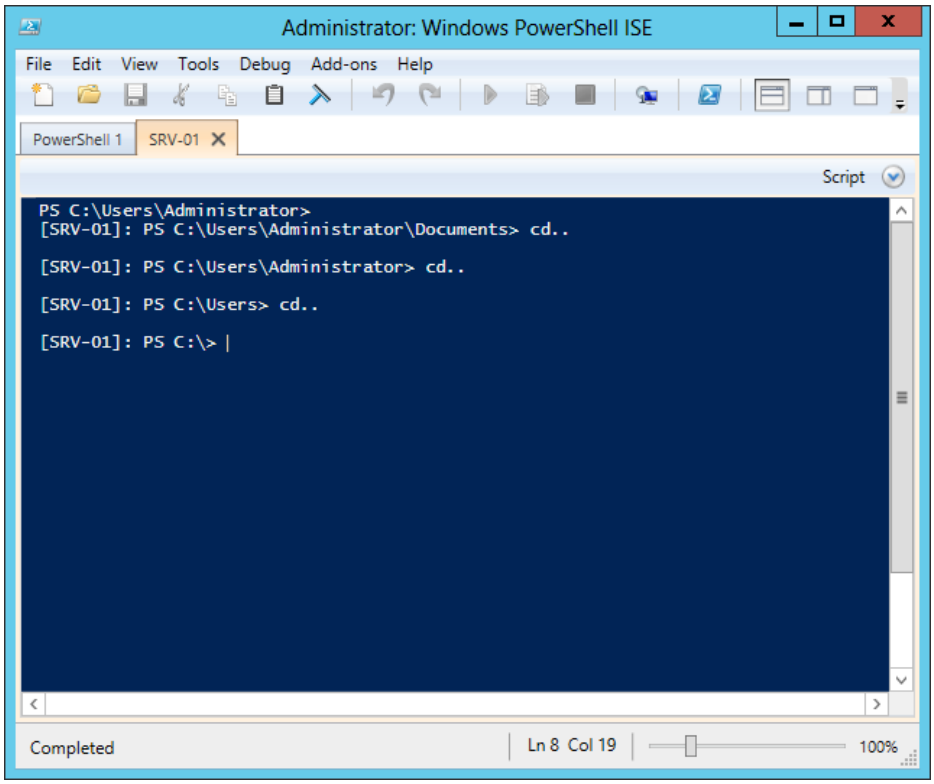
ISE tukee interaktiivista etäkäyttöä välilehtien sisältä. Interaktiivisella etäkäytöllä tarkoitetaan toimintaa, jossa kohdetietokoneeseen avataan suora (one to one) etäyhteys. Kaikki syötetyt komennot suoritetaan suoraan etätietokoneella. Jokainen välilehti on eristetty ja toimii näin omana prosessinaan. ISE:n etäistunto välilehti vastaa samaa toimintoa kuin kutsuisi *Enter-PSSession* cmdlet-komentoa paikallisella istunto välilehdellä, mutta sen sijaan, että avattaisiin ensin paikallinen istunto ja sen jälkeen etäistunto, avataan etäistunto välilehti suoraan. (Payette 2011, 619.)

Etäyhteys avataan joko valitsemalla ”File”-valikosta ”New Remote PowerShell Tab...” tai vaihtoehtoisesti pikänäppäinyhdistelmällä Ctrl+Shift+R. Tämä avaa ponnahdusikkunan, jossa kysytään kohdetietokoneen nimi ja millä käyttäjätilillä tietokoneeseen yhdistetään (kuvio 11).



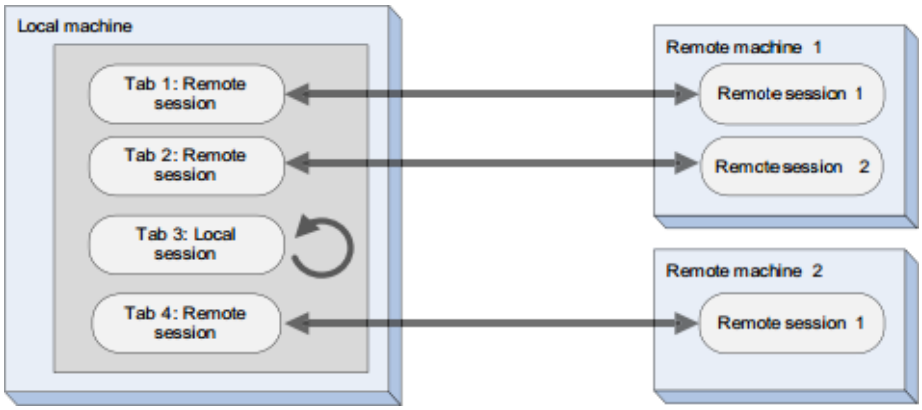
KUVIO 11. Etäyhteyden muodostaminen ISE:llä

Kun pyydetyt tiedot ovat syötetty, klikataan ”Connect”-painiketta ja syötetään käyttäjätilille määritetty salasana. Kun yhteys on luotu onnistuneesti, näkyy kohdetietokoneen nimi uudella välilehdellä (kuvio 12).



KUVIO 12. Etäyhteys muodostettu tietokoneeseen "SRV-01"

Toisin kuin pelkällä PowerShell-konsolilla, jossa interaktiivisia istuntoja voidaan suorittaa vain yksi kerrallaan, mahdollistaa ISE:n graafinen ympäristö usean istunnon yhtäaikaisen suorittamisen. Istunnot voivat olla joko paikallisia tai etäistuntoja, joissa jokainen istunto on omassa välilehdessään. Kuvion 13 esimerkissä ISE-työkalua käytetään paikallisella tietokoneella, jossa on avattu kaksi etäyhteyttä yhdelle tietokoneelle, yksi paikallinen istunto ja yksi etäyhteys toiselle tietokoneelle. (Payette 2011, 622.)



KUVIO 13. Jokaisella välilehdellä on joko paikallinen istunto tai etäistunto

5 WINDOWS SERVER 2012 -PALVELIMEN HALLINTA

Tässä luvussa esitellään Windows Server 2012 -palvelimeen liittyviä hallintatoimenpiteitä PowerShell-komennoilla. Aluksi käsitellään roolien asentaminen ja poistaminen. Tämän jälkeen käydään läpi tapahtumalokien hallinta, palveluiden hallinta, prosessien hallinta ja lopuksi palvelimen sulkeminen ja uudelleenkäynnistäminen etänä.

5.1 Roolien asentaminen ja poistaminen

Cmdlet-komennolla *Get-WindowsFeature* tulostetaan lista paikalliselle tietokoneelle asennetuista ja asennettavissa olevista rooleista ja ominaisuuksista. Jos halutaan tulostaa lista etätietokoneesta, käytetään parametria *ComputerName* määrittämään kohdetietokoneen nimi. Esimerkiksi *Get-WindowsFeature -ComputerName "SRV-01"*. Rooleja ja ominaisuuksia asennetaan cmdlet-komennolla *Install-WindowsFeature -Name <roolin nimi> -ComputerName <tietokoneen nimi> -Restart*, missä *Name*-parametrilla määritetään asennettava rooli tai ominaisuus. *Restart*-parametrilla käynnistetään kohdetietokone uudelleen jos asennettava rooli tai ominaisuus vaatii tietokoneen uudelleenkäynnistämisen.

Seuraavassa esimerkissä asennetaan Active Directory Domain Services -rooli ja Group Policy Management -ominaisuus etätietokoneelle "DC-01". Parametrilla *IncludeManagementTools* otetaan asennukseen mukaan rooliin liittyvät hallintatyökalut.

> *Install-WindowsFeature -Name AD-Domain-Services, GPMC -ComputerName "DC-01" -IncludeManagementTools -Restart*

Rooleja ja ominaisuuksia poistetaan cmdlet-komennolla *Uninstall-WindowsFeature*. Seuraavassa esimerkissä poistetaan edellisessä esimerkissä asennetut roolit ja ominaisuudet sekä niihin liittyvät hallintatyökalut.

> *Uninstall-WindowsFeature -Name AD-Domain-Services, GPMC -ComputerName "DC-01" -IncludeManagementTools -Restart*

5.2 Tapahtumalokien tarkastelu ja hallinta

Tapahtumalokeja tarkastellaan cmdlet-komennolla *Get-EventLog*. Sen perään määritetään tarkasteltavan tapahtuman nimi ja tyyppi. Kuvion 14 esimerkissä tarkastellaan viittä tuoreimpaa virhetapahtumaa tapahtumasta ”system”. *Newest*-parametrin avulla valitaan tuoreimmat viisi tapahtumaa ja *EntryType*-parametrilla määritetään tarkasteltavan tapahtuman tyyppi.

```
PS C:\> Get-EventLog system -Newest 5 -EntryType Error

  Index Time          EntryType   Source                 InstanceID Message
  ----- ----          ---------   ------                 ---------- -------
    613 Sep 10 16:52  Error       EventLog               2147489656 The previous
    365 Sep 10 16:10  Error       Service Control M...   3221232495 The Windows
    363 Sep 10 16:10  Error       Microsoft-Windows...         46 The time ser
    321 Sep 10 16:09  Error       Service Control M...   3221232495 The Windows
    319 Sep 10 16:09  Error       Microsoft-Windows...         46 The time ser
```

KUVIO 14. Tuloste viidestä tuoreimmasta ”system”-tapahtuman virhelokista

Jos halutaan tarkastella tapahtumalokeja etänä, käytetään samaa *Get-EventLog* cmdlet-komentoa, mutta määritetään *ComputerName*-parametrillä sen tietokoneen nimi, josta tapahtumalokeja halutaan tarkastella. Seuraavassa esimerkissä tarkastellaan etänä tietokoneen ”SRV-01” viittä tuoreimpaa virhetapahtumaa tapahtumasta ”system”.

Get-EventLog system -Newest 5 -EntryType Error -ComputerName “SRV-01”

Tapahtumalokien tuloste voidaan muotoilla helpommin luettavampaan muotoon ketjuttamalla edellinen komentosarja *Format-Table* cmdlet-komennon kanssa. Seuraavassa esimerkissä tarkastellaan ”SRV-01” tietokoneen tuoreimpia tapahtumalokeja ja lisäksi käytetään *Format-Table* cmdlet-komennon ominaisuuksia tapahtumalokin tulosteen muotoiluun (kuvio 15).

Get-EventLog system -Newest 5 -EntryType Error -Comp ”SRV-01” | Format-Table TimeWritten,Source, EventID,Message -wrap -auto

```
PS C:\> Get-EventLog system -Newest 5 -EntryType Error -Comp "SRV-01" | Format-Table TimeWritten,Source,EventID,Message
-wrap -auto

TimeWritten          Source                      EventID Message
-----------          ------                      ------- -------
9/10/2013 4:52:16 PM EventLog                       6008 The previous system shutdown at 4:21:06 PM on □9/□10/□2013
                                                         was unexpected.
9/10/2013 4:10:31 PM Service Control Manager        7023 The Windows Time service terminated with the following
                                                         error:
                                                         %%1792
9/10/2013 4:10:31 PM Microsoft-Windows-Time-Service   46 The time service encountered an error and was forced to
                                                         shut down. The error was: 0x80070700: An attempt was made
                                                         to logon, but the network logon service was not started.

9/10/2013 4:09:37 PM Service Control Manager        7023 The Windows Time service terminated with the following
                                                         error:
                                                         %%1792
9/10/2013 4:09:37 PM Microsoft-Windows-Time-Service   46 The time service encountered an error and was forced to
                                                         shut down. The error was: 0x80070700: An attempt was made
                                                         to logon, but the network logon service was not started.
```

KUVIO 15. Tapahtumalokien tuloste muotoiltu luettavampaan muotoon

5.3 Palveluiden hallinta

Tietokoneen palveluita (services) käynnistetään uudelleen cmdlet-komennolla *Restart-Service ”palvelun nimi”*, käynnistetään pysäytetystä tilasta cmdlet-komennolla *Start-Service ”palvelun nimi”* ja pysäytetään cmdlet-komennolla *Stop-Service ”palvelun nimi”*. Palveluita voidaan hallita myös etänä, jos käytössä on Windows PowerShell Remoting -ominaisuus. Seuraavassa esimerkissä ”Spooler”-palvelu käynnistetään uudelleen paikallisesti.

Restart-Service Spooler

Jos Windows PowerShell Remoting -ominaisuus on käytössä ja halutaan käynnistää palvelu etänä tietokoneessa ”SRV-01”, voidaan käyttää seuraavaa komentosarjaa:

Invoke-Command {Start-Service ”Spooler” -passthru} -ComputerName ”SRV-01”.

5.4 Prosessien hallinta

Proseseja lopetetaan cmdlet-komennolla *Stop-Process ”prosessin nimi”* ja käynnistetään cmdlet-komennolla *Start-Process ”prosessin nimi”*. Tietokoneessa käynnissä olevien prosessien tietoja haetaan cmdlet-komennolla *Get-Process ”prosessin nimi”*. Myös prosesseja on mahdollista hallita etänä, jos Windows PowerShell Remoting -ominasuus on käytössä. Ensimmäisessä esimerkissä käynnistetään prosessi ”calc” etänä tietokoneessa ”SRV-01” ja toisessa esimerkissä lopetetaan kyseinen prosessi etänä.

Invoke-WmiMethod -Path win32_process -Name Create -ArgumentList calc.exe -ComputerName “SRV-01”

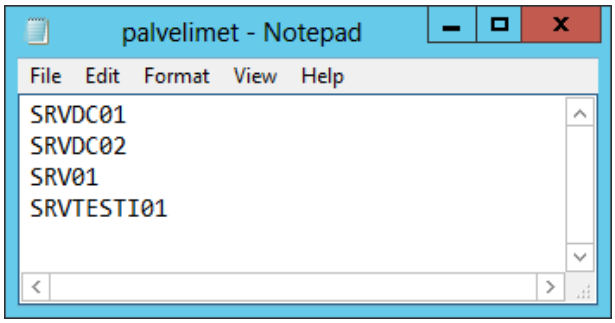
Invoke-Command {Get-Process calc | Stop-Process} -ComputerName “SRV-01”

5.5 Palvelimen sulkeminen ja uudelleenkäynnistäminen

Samassa toimialueessa sijaitsevia palvelimia sammutetaan cmdlet-komennolla *Stop-Computer ”palvelimen nimi”* ja käynnistetään uudelleen cmdlet-komennolla *Restart-Computer ”palvelimen nimi”*. Seuraavassa esimerkissä käynnistetään uudelleen yksittäinen palvelin ”SRV-01”.

Restart-Computer ”SRV-01”

Palvelimia on myös mahdollista sammuttaa tai käynnistää uudelleen suuremmassa mittakaavassa hakemalla sammutettavien tai uudelleen käynnistettävien palvelimien nimet esimerkiksi tekstitiedostosta (kuvio 16).



KUVIO 16. Toimialueen palvelimien nimet lisätty tekstitiedostoon

Seuraavassa esimerkissä käynnistetään uudelleen kaikki ”palvelimet.txt”-tekstitiedossa sijaitsevat palvelimet, jotka vastaavat ping-kutsuun vähintään kolme kertaa.

Get-Content C:\Palvelimet\palvelimet.txt | where {Test-Connection $_ -quiet -count 3} | foreach {Restart-Computer $_ -force}

Komentosarjassa *Get-Content*-komennolla haetaan sisältö määritellystä tekstitiedostosta. Tämän jälkeen putkitetaan saatu sisältö *where*-lauseeseen. *Where*-lauseessa testataan yhteys kuhunkin

palvelimeen *Test-Connection*-komennolla. *Quiet*-parametri palauttaa joko *true* tai *false* arvon ja *count*-parametrillä määritetään kuinka moneen ping-kutsuun kohdepalvelimen täytyy vastata. Jos palvelin vastaa ping-kutsuihin, lisätään se muuttujaan *$_*. *Foreach*-lauseessa käynnistetään ping-kutsuihin vastanneet palvelimet uudelleen ja pakotetaan *force*-parametrilla kaikki käyttäjät kirjautumaan ulos palvelimilta.

6 AKTIIVIHAKEMISTON HALLINTA

Tässä luvussa esitellään aktiivihakemistoon liittyviä hallintatoimenpiteitä PowerShell-komennoilla. Luvussa käsitellään organisaatioyksiköiden, käyttäjätilien, käyttöoikeusryhmien ja tietokonetilien luonti. Lopuksi käsitellään käyttäjätilien hallintaan liittyviä toimenpiteitä.

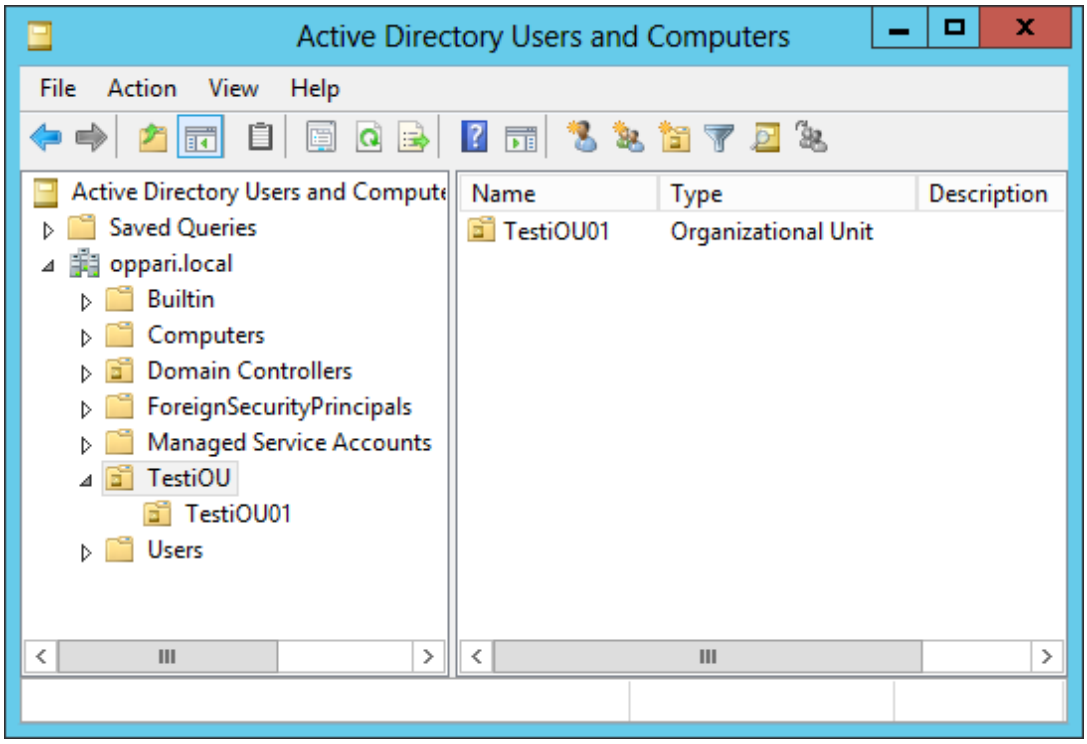
6.1 Organisaatioyksiköiden luonti

Uusi organisaatioyksikkö luodaan cmdlet-komennolla *New-ADOrganizationalUnit*. Seuraavassa esimerkissä luodaan TestiOU-organisaatioyksikkö oppari.local-toimialueeseen, missä *Name*-parametrilla määritetään organisaatioyksikön nimi ja *Path*-parametrilla sen luontisijainti.

New-ADOrganizationalUnit -Name TestiOU -Path ”dc=oppari,dc=local”

Jos halutaan luoda toinen organisaatioyksikkö aiemmin luodun organisaatioyksikön sisään, käytetään samaa cmdlet-komentoa *New-ADOrganizationalUnit*, mutta lisätään *Path*-parametriin sen organisaatioyksikön nimi, jonka sisään toinen organisaatioyksikkö luodaan. Seuraavassa esimerkissä luodaan ”TestiOU01”-organisaatioyksikkö aiemmin luodun ”TestiOU”-organisaatioyksikön sisään (kuvio 17).

New-ADOrganizationalUnit -Name TestiOU01 -Path
”ou=TestiOU,dc=oppari,dc=local”



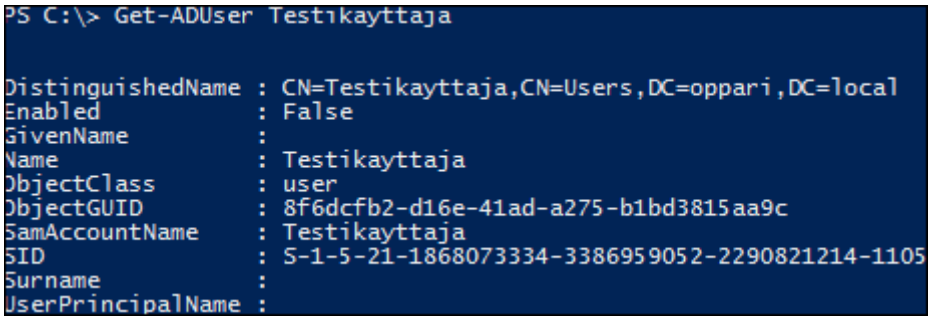
KUVIO 17. TestiOU01-organisaatiyksikkö luotu TestiOU-organisaatioyksikön sisään

6.2 Käyttäjätilien luonti ja poistaminen

Uusi Active Directory -käyttäjätili luodaan cmdlet-komennolla *New-ADUser -Name*, missä *Name*-parametrilla määritetään käyttäjätilin nimi. Seuraavassa esimerkissä luodaan uusi käyttäjätili ”Testikayttaja” organisaatioyksikön ”TestiOU” sisään.

New-ADUser -Name Testikayttaja -Path ”ou=TestiOU,dc=oppari,dc=local”

Jos halutaan varmistaa, että käyttäjätilin luonti onnistui, voidaan käyttää apuna cmdlet-komentoa *Get-ADUser ”käyttäjätilin nimi”*, joka tulostaa halutun käyttäjätilin tiedot Windows PowerShell -konsoliin, kuten kuviossa 18 ilmenee.



KUVIO 18. Get-ADUser-komennon tulostamat tiedot käyttäjätilistä Testikayttaja

Aiemmin luotuja käyttäjätilejä poistetaan cmdlet-komennolla *Remove-ADUser "käyttäjätilin nimi"*. Windows PowerShell -konsoliin tulostuu ilmoitus, joka kysyy halutaanko käyttäjätili varmasti poistaa. Vastaamalla tähän kysymykseen 'Y', käyttäjätili poistetaan. Seuraavassa esimerkissä poistetaan käyttäjätili "Testikayttaja" organisaatioyksiköstä "TestiOU".

Remove-ADUser Testikayttaja -Path "ou=TestiOU,dc=oppari,dc=local"

6.3 Käyttöoikeusryhmien luonti

Aktiivihakemistoon luodaan uusi käyttöoikeusryhmä (security group) cmdlet-komennolla *New-ADGroup*. *New-ADGroup* cmdlet-komento vaatii kolme eri parametriä: *Name*, joka määrittää ryhmän nimen, *Path*, joka määrittää ryhmän luontisijainnin ja *groupScope*, joka voi olla joko *global, universal* tai *domainlocal*. Seuraavassa esimerkissä luodaan uusi käyttöoikeusyhmä "TestiSec" oppari.local-toimialueeseen, sijoitetaan se "TestiOU"-organisaatioyksikköön ja määritetään ryhmä globaaliryhmäksi.

New-ADGroup -Name TestiSec -Path "ou=TestiOU,dc=oppari,dc=local" -groupScope global

6.4 Tietokonetilien luonti

Uusi tietokonetili luodaan cmdlet-komennolla *New-ADComputer*. Tietokonetilin luontisijanti määritetään *Path*-parametrillä ja nimi *Name*-parametrillä. Seuraavassa esimerkissä luodaan uusi tietokonetili "Testikone" aikaisemmin luotuun organisaatioyksikköön "TestiOU01". "TestiOU01"-

organisaatioyksikkö on luotu ”TestiOU”-organisaatioyksikön sisään, joten *Path*-parametriin pitää olla määritetty myös ”TestiOU01”-organisaatioyksikkö.

New-ADComputer -Name Testikone -Path
”ou=TestiOU01,ou=TestiOU,dc=oppari,dc=local”

6.5 Käyttäjätilien hallinta

Tässä osiossa käydään läpi tyypillisimpiä aktiivihakemiston käyttäjätilien hallintaan liittyviä toimenpiteitä. Kaikki komennot ovat annettu PowerShell-komentotulkkiin paikallisesti järjestelmänvalvojan tunnuksilla.

Käyttäjätilin lisääminen käyttöoikeusryhmään

Käyttäjätili lisätään käyttöoikeusryhmään cmdlet-komennolla *Add-ADGroupMember ”ryhmän nimi”*. Seuraavassa esimerkissä lisätään käyttäjätili ”Testikayttaja” aiemmin luodun käyttöoikeusryhmän ”TestiSec” jäseneksi, missä *Members*-parametrilla määritetään lisättävän käyttäjätilin nimi.

Add-ADGroupMember ”TestiSec” -Members Testikayttaja

Cmdlet-komennolla *Get-ADGroupMember “ryhmän nimi”* voidaan tarkistaa, että käyttäjätilin lisääminen käyttöoikeusryhmään varmasti onnistui (kuvio 19). Komento tulostaa kaikki käyttöoikeusryhmään lisätyt käyttäjätilit.

```
PS C:\> Get-ADGroupMember "TestiSec"


distinguishedName : CN=Testikayttaja,CN=Users,DC=oppari,DC=local
name              : Testikayttaja
objectClass       : user
objectGUID        : f98f3dad-8852-43a6-bb55-4238fa229ecb
SamAccountName    : Testikayttaja
SID               : S-1-5-21-1868073334-3386959052-2290821214-1106
```

KUVIO 19. Get-ADGroupMember tuloste käyttöoikeusryhmästä ”TestiSec”

Salasanan uudelleen asettaminen

Käyttäjätilin salasana asetetaan uudelleen cmdlet-komennolla *Set-ADAccountPassword*. Seuraavassa esimerkissä vaihdetaan käyttäjätilin ”Testikayttaja” salasanaksi ”*oB-z(\p_”*. Tietoturvasyistä salasana määritetään *NewPassword*-parametrissä salatuksi tekstijonoksi, joka tallennetaan muistiin Windows PowerShell-istunnon ajaksi (Hicks 2012, hakupäivä 17.8.2013.)

Set-ADAccountPassword Testikayttaja -NewPassword (ConvertTo-SecureString -AsPlainText -String “oB-z(\p_” -force)

Cmdlet-komennolla *Set-ADUser Testikayttaja -ChangePasswordAtLogon $True* pakotetaan käyttäjä vaihtamaan salasana seuraavalla kirjautumiskerralla. Tämä komento asettaa käyttäjätilin *PasswordExpired*-ominaisuuden arvoksi *True*.

Käyttäjätilin aktivointi ja disablointi

Käyttäjätili aktivoidaan cmdlet-komennolla *Enable-ADAccount* ja disabloidaan cmdlet-komennolla *Disable-ADAccount*. Jos halutaan aktivoida tai disabloida useita käyttäjätilejä yhtä aikaa käyttämättä skriptejä, voidaan hyödyntää komentojen putkitusta. Seuraavassa esimerkissä aktivoidaan kaikki ne käyttäjätilit, joille on määritetty osastoksi Myynti.

Get-ADUser -filter ”Department -eq ‘Myynti’” | Activate-ADAccount

Kun halutaan aktivoida tai disabloida vain tietty käyttäjätili, määritetään cmdlet-komennon perään käyttäjätilin nimi. Jos halutaan aktivoida kaikki disabled-tilassa olevat käyttäjätilit, haetaan disabled-tilassa olevat käyttäjätilit komennolla *Get-ADUser -Disabled* ja putkitetaan ne *Enable-ADUser*-komennon kanssa.

Enable-ADAccount Testikayttaja
Disable-ADAccount Testikayttaja
Get-ADUser -Disabled | Enable-ADUser

7 POHDINTA

Opinnäytetyön tavoitteena oli perehtyä Windows Server 2012 -palvelimen ja aktiivihakemiston hallintaan PowerShell-komentotulkkia käyttäen. Varsinaista toimeksiantajaa työllä ei ollut, vaan kaikki työssä suoritetut PowerShell-komennot tehtiin virtuaaliympäristössä. Opinnäytetyössä perehdyttiin Windows Server 2012 -käyttöjärjestelmän tuomiin uudistuksiin yleisesti, sen eri versioihin, Server Core -asennusvaihtoehtoon, palvelinrooleihin ja niihin liittyviin ominaisuuksiin. Tämän jälkeen perehdyttiin itse PowerShell-komentotulkkiin ja siihen liittyviin työkaluihin sekä toiminnallisuuksiin. Käytännön osuudessa tehtiin erilaisia hallintatoimenpiteitä Windows Server 2012 -palvelimelle ja aktiivihakemistoon.

Suurin haaste työtä tehdessä oli rajata teoriaosuus sopivaksi kokonaisuudeksi. Windows Server -palvelinkäyttöjärjestelmät, aktiivihakemisto ja PowerShell ovat kaikki todella laajoja käsitteitä, joista jokaisesta löytyi kattavasti sähköisiä ja kirjallisia lähteitä. Onnistuin mielestäni kuitenkin rajaamaan oleellisimmat asiat tähän työhön. Käytännön osuus koostui sellaisista hallintatoimenpiteistä, joita itse pidin tärkeinä osata, ja joita tuli tuli esille muun muassa oman työharjoittelujakson aikana.

PowerShell-komentotulkin opetteleminen osoittautui haastavaksi, mutta samalla myös mielenkiintoiseksi prosessiksi. Ongelmia esiintyi eniten skriptien luonnissa, mutta kaikki skriptit saatiin kuitenkin toimimaan halutulla tavalla. Kaiken kaikkiaan opin erittäin paljon uutta asiaa PowerShell-komentotulkin käyttöön liittyen. Oppimani taidot ovat tärkeitä jatkon kannalta ja tulenkin varmasti kehittämään PowerShell-taitojani myös jatkossa.

LÄHTEET

Alpern, N., Alpern, J. & Muller, R. 2012. IT Career JumpStart: An Introduction to PC Hardware, Software, and Networking. Sybex.

Carpenter, T. 2011. Microsoft® Windows Server® Administration: Essentials. Sybex.

Desmond, B., Richards, J., Allen, R. & Lowe-Norris, A. 2013. Active Directory, 5th Edition. O'Reilly Media, Inc.

Hicks, J. 2012. Top 10 Active Directory Tasks Solved with PowerShell. Hakupäivä 17.8.2013, http://windowsitpro.com/powershell/top-10-active-directory-tasks-solved-powershell.

Lee, T., Mitschke, K., Schill, M. & Tanasovski, T. 2011. Windows PowerShell® 2.0 Bible. John Wiley & Sons.

Microsoft 2013. About Server Core. Hakupäivä 18.6.2013, http://msdn.microsoft.com/en-us/library/windows/desktop/ee391626%28v=vs.85%29.aspx.

Microsoft 2013. Common Parameter Names. Hakupäivä 2.12.2013, http://msdn.microsoft.com/en-us/library/dd901844%28v=vs.85%29.aspx.

Microsoft 2013. Group Policy Overview. Hakupäivä 15.10.2013, http://technet.microsoft.com/library/hh831791.

Microsoft 2013. Piping and the Pipeline in Windows PowerShell. Hakupäivä 4.9.2013, http://technet.microsoft.com/en-us/library/ee176927.aspx.

Microsoft 2013. Roles, Role Services, and Features. Hakupäivä 22.6.2013, http://technet.microsoft.com/en-us/library/cc754923.aspx.

Microsoft 2013. What's New in Windows PowerShell. Hakupäivä 16.10.2013, http://technet.microsoft.com/en-us/library/hh857339.aspx.

Microsoft 2013. Windows Management Framework. Hakupäivä 16.10.2013, http://support.microsoft.com/kb/968929.

Microsoft 2013. Windows Server 2012 How to Buy. Hakupäivä 15.6.2013, http://www.microsoft.com/en-us/server-cloud/windows-server/buy.aspx.

Morimoto, R., Noel, M., Yardeni, G., Droubi, O., Abbate, A. & Amaris, C. 2012. Windows Server® 2012 Unleashed. Sams.

Payette, B. 2011. Windows PowerShell in Action, Second Edition. Manning Publications.

Posey, B. 2006. Networking Basics: Part 5 - Domain Controllers. Hakupäivä 29.8.2013, http://www.windowsnetworking.com/articles-tutorials/netgeneral/Networking-Basics-Part5.html.

Rhodes-Ousley, M. 2013. Information Security The Complete Reference, Second Edition. McGraw-Hill.

Rouse, M. 2008. Active Directory. Hakupäivä 24.6.2013, http://searchwindowsserver.techtarget.com/definition/Active-Directory.

Rouse, M. 2012. Group Policy. Hakupäivä 28.7.2013, http://searchwindowsserver.techtarget.com/definition/Group-Policy.

Rouse, M. 2008. Server Core. Hakupäivä 15.11.2013, http://searchwindowsserver.techtarget.com/definition/Server-Core.

Svidergol, B. & Allen, R. 2013. Active Directory Cookbook, 4th Edition. O'Reilly Media, Inc.