Windows PowerShellin hyödyntäminen Windows Server- ja AD-ympäristössä

HAMK
HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Ammattikorkeakoulututkinnon opinnäytetyö

Visamäki, Tietojenkäsittelyn koulutusohjelma

Kevät, 2018

Pasi Pahkuri

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Tietojenkäsittelyn koulutusohjelma
Visamäki

Tekijä Pasi Pahkuri **Vuosi** 2018
Työn nimi Windows PowerShellin hyödyntäminen Windows Server- ja AD- ympäristössä
Työn ohjaaja/t Erkki Laine

TIIVISTELMÄ

Tämän opinnäytetyön pääasiallisena tarkoituksena oli selvittää miten PowerShelliä voi hyödyntää Windows Server- ja AD-ympäristössä. Näiden lisäksi pyrittiin selvittämään ja kokeilemaan miten PowerShelliä voi hyödyntää Office 365 -ympäristössä. Työn toimeksiantaja oli tämän työn tekijä itse eli Pasi Pahkuri. Tämä aihe valittiin sillä PowerShell ja sen käyttäminen on ajankohtaista työelämässä.

Työn alussa teoriavaiheessa käydään läpi aluksi mikä on PowerShell, sen ominaisuuksia ja sen historiaa. Teoriaosuudessa käydään lisäksi läpi erilaisia PowerShellin työkaluja jotka helpottavat sen käyttämistä sekä PowerShellillä skriptaamista. PowerShellin lisäksi teoriaosuudessa käydään läpi käytettyjen palvelinalustojen ominaisuuksia ja tietoja. Pääasiallisena alustana työssä oli Windows Server 2012 R2 sekä Windows Server 2016.

Käytännön osuudessa käytiin läpi PowerShellin perusasioita ja komentoja. PowerShellina avulla otettiin käyttöön Aktiivihakemisto ja Office 365 -ympäristö. Näiden lisäksi testattiin erilaisia hallintakomentoja liittyen yleisesti Windows Serverin ja työasemien hallintaan.

Tuloksena työssä huomattiin, että PowerShellistä on erittäin paljon hyötyä ja se helpottaa sekä nopeuttaa työskentelyä paljon. PowerShellin avulla voidaan automatisoida paljon erilaisia palvelimen ja sen ympäristöjen hallintatoimenpiteitä. Kyseessä on erittäin laaja ja monikäyttöinen skriptausympäristö.

Avainsanat PowerShell, Windows Server, Aktiviihakemisto, Office 365

Sivut 35 sivua

HÄMEEN AMMATTIKORKEAKOULU
HÄME UNIVERSITY OF APPLIED SCIENCES

Degree Programme in Business Information Technology
Visamäki

Author Pasi Pahkuri **Year** 2018
Subject Utilization of PowerShell in Windows Server and AD environment.
Supervisors Erkki Laine

ABSTRACT

The main purpose of this thesis was to find out how to manage Windows Server and Active Directory with PowerShell. Also, the aim was to investigate, and test how PowerShell can be used in the Office 365 environment.

The theory part examines what is PowerShell and what are its features and history. Also, the theory section explains different PowerShell tools that can be used with it and in scripting. The Main operating system used in this thesis was Windows Server 2012 R2 and Windows Server 2016. In the theory section these operating systems and their features are also examined.

In the beginning of the practical part PowerShell's basic features and commands were tested. Active Directory and Office 365 environment were set up with PowerShell. In addition, various PowerShell commands were tested as well as how to manage Windows Server and workstations.

As the result of this thesis, it was found out that PowerShell is very useful and powerful tool to manage server and its environments. With PowerShell lots of processes can be automated. PowerShell is very wide and multipurpose scripting environment.

Keywords PowerShell, Windows Server, Active Directory, Office 365

Pages 35 pages

SISÄLLYS

TERMIT JA SANASTOA

AD	Active Directory eli Aktiivihakemisto on Microsoftin käyttäjätietokanta ja hakemistopalvelu.
DSC	Desired State Configuration on hallinta-alusta PowerShellissä jolla voi tehdä erilaisia konfiguraatioita.
ISE	Integrated Scripting Environment on PowerShell skriptaustyökalu ja editori.
O365	Office 365 on Microsoftin toimisto ohjelmistopaketti jota voi käyttää pilvessä sekä paikallisesti.
WMF	Windows Management Framework on hallintatyökalu ja paketti, jonka avulla voi asentaa ja päivittää PowerShellin sekä sen osioita.

1 JOHDANTO

PowerShell on Microsoftin kehittämä ja julkaisema komentorivipohjainen työkalu Windows-käyttöjärjestelmiin. Sen tehtävä on tehostaa käyttöjärjestelmän hallintaa, erityisesti ylläpitäjän näkökulmasta. Opinnäytetyössä tutustutaan siis PowerShelliin, miten sitä käytetään ja mitä ominaisuuksia siinä on.

Tässä työssä pyritään selvittämään, miten PowerShelliä voi hyödyntää Windows Server- ja AD-ympäristön hallinnassa. Niiden lisäksi tutkitaan miten sitä voi hyödyntää Office 365 -ympäristössä ja hallinnassa. Työssä pääasiallinen palvelinkäyttöjärjestelmäalusta on Microsoftin Windows Server 2012 R2 Standard sekä Windows Server 2016 Standard.

Teoria osuudessa tutustutaan PowerShellin lisäksi Windows Serverin 2012 ja 2016 versioihin, Office 365 -ympäristöön sekä Windows Aktiivihakemistoon.

Käytännön osuudessa otetiin käyttöön palvelinkäyttöjärjestelmät Windows Server 2012 ja 2016. Sen asennettiin PowerShell ja PowerShell ISE skriptauseditori, sekä testattiin niitä käytännössä. PowerShell hallintaa kokeiltiin Windows Server-palvelinkäyttöjärjestelmissä, Aktiivihakemistossa ja Office 365 -ympäristössä.

Työn tavoitteena on selvittää mikä on PowerShell ja miten sitä voi hyödyntää Windows Server ja AD -ympäristössä, sekä miten PowerShellillä voi hallita Windows Server –palvelinkäyttöjärjestelmää ja sen komponentteja.

Tutkimuskysymykset tässä työssä ovat: Mikä on Windows PowerShell? Miten PowerShelliä käytetään? Mitä hyötyä siitä on? Miten sitä voi hyödyntää Windows Server 2012 ja 2016 ympäristössä? Miten sitä voi hyödyntää Active Directoryssa?

Työssä tilaaja on työntekijä eli Pasi Pahkuri. Idea ja kiinnostus tähän aiheeseen tulivat omista mielenkiinnoista sekä tarpeesta töissä, PowerShellin osaaminen tehostaisi päivittäistä työskentelyä.

2 POWERSHELL

Windows PowerShell on Microsoftin kehittämä skriptauskieli ja ympäristö. Se on suunnattu erityisesti järjestelmien ylläpitäjille ja tehokäyttäjille. Powershell on komentorivityökalu ja se on rakennettu .NET Frameworkin päälle. (Microsoft n.d.a.)

PowerShellin käyttäminen helpottaa hallintaa ja automaatiota palvelinympäristössä. PowerShellin avulla hallinta onnistuu helposti suuressa mittakaavassa ja reaaliajassa. PowerShellin päätarkoituksena on siis parantaa tehokkuutta. Kaikki PowerShellin versiot ovat ladattavissa Microsoftin sivuilta. (Microsoft 2018a.)

2.1 Versiot

PowerShell Versiot	**Julkaisupäivä**	**Oletus Windows versiot**
PowerShell 1.0	Marraskuu 2006	Windows Server 2008
PowerShell 2.0	Lokakuu 2009	Windows 7, Windows Server 2008 R2
PowerShell 3.0	Syyskuu 2012	Windows 8. Windows Server 2012
PowerShell 4.0	Lokakuu 2013	Windows 8.1, Windows Server 2012 R2
PowerShell 5.0	Helmikuu 2016	Windows 10
PowerShell 5.1	Tammikuu 2017	Windows 10 Anniversary Update, Windows Server 2016
PowerShell 6.0 Core	Marraskuu 2018	Windows Server 2016 ja Windows 10

Taulukko 1. PowerShell versiohistoria.

PowerShellista on julkaistu useita eri versioita eri alustoille, ensimmäinen versio 1.0 julkaistiin marraskuussa 2006. Sen on jälkeen julkaistu versiot 1.0, 2.0, 3.0, 4.0, 5.0, 5.1 ja uusimpana vuonna 2018 PowerShell 6.0 Core. Taulukossa 1 on versiot vielä tarkemmin läpi, julkaisupäivät ja oletuskäyttöjärjestelmä jonka mukana kyseinen versio tulee. (4sysops n.d.)

PowerShellin versio 1.0 julkaistiin vuonna 2006 käyttöjärjestelmiin Windows XP ja Windows Server 2003. Se piti sisällään noin 125 erilaista cmdlet:iä. Ensimmäisestä versioista lähtien PowerShellissä oli tukitoimia muiden skriptauskielten kanssa. Ominaisuuksiltaan tämä versio oli kuitenkin vielä hyvin karsittu verrattuna myöhäisempiin versioihin. (Lee, Mitschke, Schill & Tanasovski 2011, 6.; Computerperformance n.d.a.)

PoweShellin versio 2.0 julkaistiin elokuussa vuonna 2009 Windows Server 2008 R2 ja Windows 7 käyttöjärjestelmien yhteydessä. Windows PowerShell on asennettuna oletuksena näissä käyttöjärjestelmissä, perus PowerShell käyttöliittymän lisäksi mukana on asennettuna kehittyneempi ISE eli Windows PowerShell Integrated Scripting Enviroment. Versio 2.0 on saatavilla aiemmille Windows versiolle erikseen ladattuna.

ISE-käyttöliittymän lisäksi PowerShell 2.0 toi mukanaan suuren määrän uudistuksia. Etähallintaominaisuudet ovat kehittyneemmät. Mukana tuli uutena ominaisuutena Job-komennot, jotka mahdollistavat komentojen ajamisen taustatoimintoina. Tämä on erityisen hyödyllinen toiminto, jos kyseessä on pitkään ajettava skripti. (Lee ym. 2011, 44 - 51.)

Windows PowerShell 2.0 cmdlets:ien määrä kasvoi yli 100:lla eli kokonaisuudessaan niitä on yli 200 kappaletta. Yhteenvetona 2.0 versio toi uusina asioina, etähallinnan, Job tausta-ajot, kehittyneet funktiot, moduulit, eventit, ISE käyttöliittymän, siirrot ja virheiden etsinnän. (Lee ym. 2011, 65 - 68.)

PowerShell 3.0 versio tuli mukana Windows 8 ja Windows Server 2012 käyttöjärjestelmissä, mutta on saatavilla erikseen vanhemmillekin käyttöjärjestelmäversioille. Mukana tuli suuri määrä parannuksia ja uusi toimintoja. (Driscoll 2012, 12.)

ISE-käyttöliittymään tuli PowerShell 3.0 version myötä useita parannuksia ja käyttöliittymä tuli käyttäjäystävällisemmäksi. Mukana tuli paljon uusia funktioita ja cmdlets-komentoja, kokonaisuudessaan satoja uusia. Työskentelyä helpottavana ominaisuutena tuli autotäyttö eli kun aloitat kirjoittamaan cmdlets:iä, tulee listalta valittavaksi vaihtoehtoja alun perusteella. Toinen helpottava ominaisuus on, että mikäli koodissa on virhe, niin se alleviivataan automaattisesti punaisella. PowerShell 3.0 versiossa

on myös mahdollista ryhmittää eri cmdlets-komentoja. Eri cmdlets-komentoja käydään läpi myöhemmin tässä työssä. (Computerperformance n.d.b.)

PowerShell 4.0 versiossa ei ollut kovinkaan suuria muutoksia tai ominaisuuksia aiempiin versioihin verrattuna. Mukana tuli uusia cmdlets-komentoja sekä funktioita. Suurin yksittäinen uusi ominaisuus oli DSC eli Desired State Configuration. Kyseessä on ominaisuus joka helpottaa serverin hallintaa ja sen automaatiota. (Computerperformance n.d.c)

PowerShell 5.0 esittelee uutena ominaisuutena luokat joita voi luoda, samaan tapaan kuin monessa muussakin ohjelmointikielessä. Tässäkin versiossa tulee mukana useita uusia cmdlets-komentoja. Näiden lisäksi ISE käyttöliittymä ja aiemmin mainittu DSC ovat saaneet pieniä parannuksia. PowerShell 5.0 versiossa on korjattu useita eri bugeja, josta johtuen toimivuus on parempaa ja nopeampaa kuin aikaisemmissa versioissa. (Microsoft n.d.c)

PowerShellin viimeisin versio on nimeltään 6.0 Core. Siinä kaikista suurin muutos on, että se on saatavilla Windows käyttöjärjestelmien lisäksi MacOS ja Linux-käyttöjärjestelmille. PowerShell 6.0 Core versiossa on mukana tuki, että sen voi asentaa samaan aikaan muiden versioiden kanssa eli niin sanottu Side-by-side asennus, tämä helpottaa uusien versioiden testausta ja skriptien testausta aiemmassa PowerShell ympäristössä. (Microsoft n.d.d)

2.2 **PowerShell käyttöliittymä**

PowerShell löytyy vähän käyttöjärjestelmästä riippuen hieman eri paikasta, helpoiten sen yleensä saa auki Käynnistä-valikosta hakemalla tai mikäli kyseessä on käyttöjärjestelmältään Windows Server työasema, niin PowerShell löytyy Server Managerin Tools-valikon alta. Käytössä olevan PowerShell version saa helposti selville $PSVersionTable muuttujalla.

Kuva 1. PowerShell -käyttöliittymä.

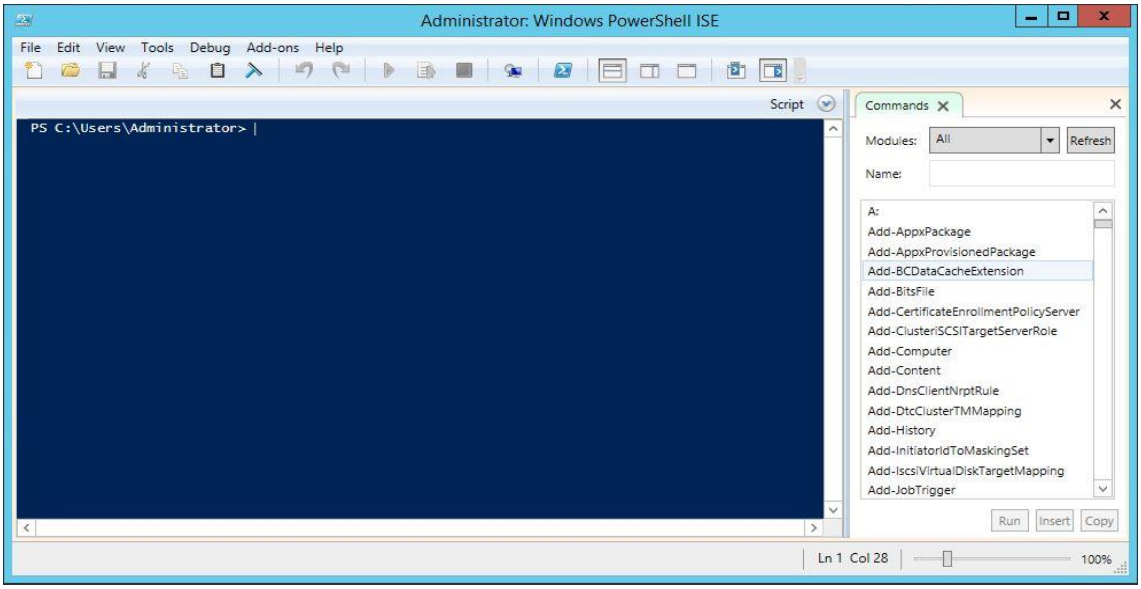
Ulkoasulta PowerShell on hyvin yksinkertainen ja pelkistetty. Ulkoasua ja asetuksia voi säätää klikkaamalla hiiren oikealla painikkeella yläpalkin päällä ja valitsemalla sieltä Asetukset/Properties. Sieltä on mahdollista säätää esimerkiksi fonttien kokoa ja väritystä haluamakseen.

PowerShellin versio riippuu käyttöjärjestelmän versiosta ja kaikki versiot ovat ladattavissa Microsoftin sivuilta. Sieltä voi päivittää PowerShellin uudempaan versioon tai ladata mikäli käyttöjärjestelmässä ei sitä ole mukana. Sivustolta löytyy sen lisäksi ohjeet asennukseen ja tarkemmat tiedot järjestelmävaatimuksista. (Microsoft n.d.f)

2.3 **PowerShell ISE**

PowerShell ISE eli Integrated Scripting Environment on skriptausympäristö ja työkalu, jolla voi kirjoittaa, ajaa sekä testata skriptejä. ISE julkaistiin ensimmäisen kerran PowerShell 2.0 kanssa ja se on saatavilla siihen sekä myöhempiin PowerShell versioihin. Se esiteltiin ensimmäisen kerran Windows Server 2008 R2 ja Windows 7 käyttöjärjestelmissä. Kyseessä on siis normaalia komentokehotetta ja PowerShelliä kehittyneempi ympäristö. Toiminnoiltaan se vastaa hyvin pitkältä muita vastaavia nykyaikaisia skriptaus- ja koodausympäristöjä.

PowerShell ISE on saatavilla kaikille Windows työasemille jotka voivat pyörittää PowerShell 2.0 tai myöhempiä versioita. (Microsoft n.d.e)



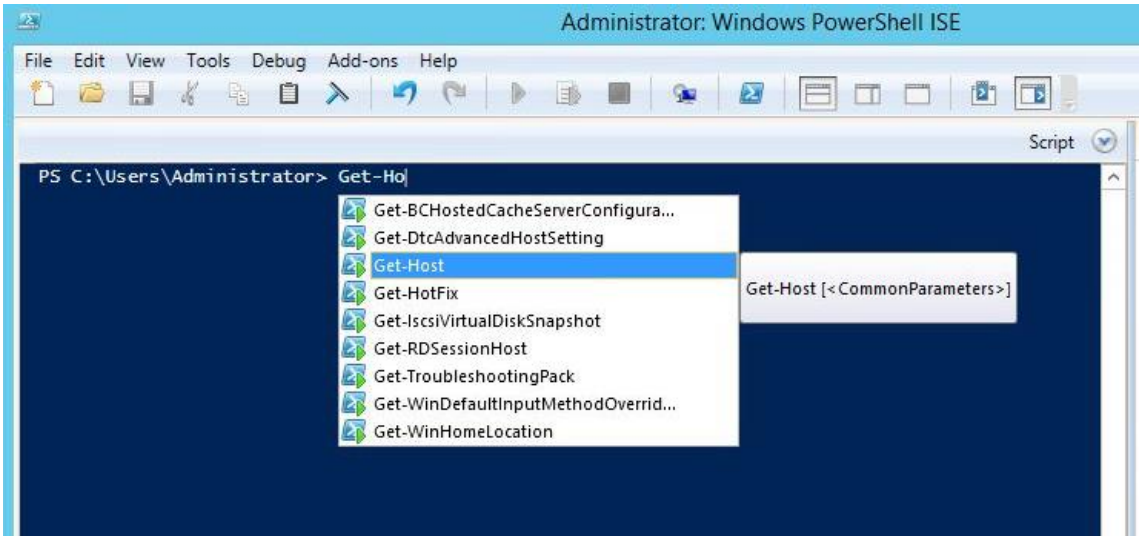
Kuva 2. Windows PowerShell ISE perusnäkymä.

Ulkoasua ja näkymää pystyy muokkaamaan asetuksien kautta haluamakseen. Värien ja fonttien lisäksi säätö mahdollisuuksia on erittäin runsaasti, esimerkiksi eri osioiden asettelua voi säätää. Työskentelyä helpottaa mahdollisuus zoomata ulos ja sisäänpäin, mikä helpottaa, jos koodia on paljon. ISE osaa ilmoittaa virheistä koodista sitä mukaa kun sitä kirjoitetaan, virheelliseen koodiin tulee virheenmerkki ja kun vie hiiren sen päälle, saa tarkempaa tietoa siitä.

ISE:stä löytyy Add-on -valikko. Sen kautta voi hallinnoida mahdollisia lisäosia joita on käytössä ja sitä kautta saa auki Add-on Tools verkkosivun. PowerShell ISE siis tukee lisäosia, joita Microsoft ja yhteisö on julkaissut, näillä saa erilaisia lisäominaisuuksia käyttöön. PowerShell ISE skriptauseditoriin on siis tehty yhteisön toimesta paljon erillisiä lisäosia jotka ovat siis käytännössä koodinpätkiä mitkä lisäävät toiminnallisuuden ISE editoriin tai sen sisällä johonkin toimintoon. Näitä löytyy useita kymmeniä pelkästään Microsoftin sivuilta. Asennus tapahtuu helposti Install-Module komentoa käyttämällä. Lisäosia löytyy muun muassa kielentarkistukseen ja skriptien selaus lisäosa, jolla voit helposti selata tehtyjä skriptejä ja niiden koodia sekä merkitä niitä suosikeiksi. ISE lisäosia voi tehdä myös itse C# kieltä käyttämällä. (4sysops 2016.)

PowerShell ISE:n yksi tärkeimpiä ominaisuuksia on automaattinen täydennys siihen mitä kirjoitat, eli kun lähdetään kirjoittamaan komentoja, saa täydennettyä sen. Esimerkiksi jos kirjoittaa ”Get-H”, niin saa eri ehdotuksia ja voi valita oikean Enteriä tai tabulaattoria painalla. Kuvassa 3 on valittavana Get-Host cmdlet-komento. Tämä helpottaa ja nopeuttaa työskentelyä huomattavasti, kun ei tarvitse tietää ja muistaa komentoja kokonaisuudessaan. Tämän lisäksi komentoja voi selata ja hakea oikealla olevasta Commands-valikosta. Samasta valikosta komentoja voi ajaa, syöttää

ja kopio. Niistä näkyy sen lisäksi mahdolliset parametrit jota komentoon voi antaa.



Kuva 3. Automaattinen tekstin täydennys.

Debug-valikon kautta voi ajaa ja testata koodia, sieltä voi ajaa koodista pätkiä ja asettaa välipisteitä joissa skriptin ajaminen pysähtyy. Näiden avulla voidaan testata ja etsiä virheitä koodista, mikä helpottaa erityisesti pitkien ja monimutkaisten skriptien tekemistä.

PowerShell ISE:ssä voi hyödyntää pikanäppäimiä tehostamaan työskentelyä. Ne löytyvät jokaisen valinnan perästä, eli esimerkiksi tallennus onnistuu Ctrl + S painikkeelle. Nämä ovat hyvin pitkälle samoja kuin muissakin Microsoftin ohjelmissa.

2.4 Muut PowerShellin komponentit ja työkalut

Tässä osiossa käydään läpi muutamia PowerShelliin liittyviä tekniikoita ja komponentteja jotka helpottavat sekä tehostavat PowerShellin käyttämistä. Nämä käydään läpi lyhyesti ja pintapuolisesta. Kyseessä on kuitenkin oleellisia asioita PowerShellin ja skriptauksen osalta.

Sen lisäksi käydään läpi eri työkaluja PowerShelliin liittyen. PowerShelliin on julkaistu useita erilaisia työkaluja ja ohjelmistoja helpottamaan sen käyttämistä. Ohjelmistoja on sekä maksullisia että maksuttomia ohjelmia. Tässä osiossa käydään läpi niistä pari, nämä ovat erikseen asennettavia ohjelmistoja eivätkä tule oletuksena mukana PowerShellissä tai Windows palvelinkäyttöjärjestelmissä. Näistä käydään läpi Windows PowerShell Desired State Configuration, Windows Management Framework, Pester, PowerShell Studio ja PowerShell Tools for Visual Studio 2017.

Windows PowerShell Desired State Configuration eli DSC on alusta/tekniikka konfiguroida ja standardisoida työasemia. Sen avulla saadaan helposti suunniteltua, ajettua ja päivitettyä konfiguraatiota. DSC:n avulla saadaan siis tehostettua entisestään PowerShellilla automatisointia ja suurien massojen hallinnointia. Tämä esiteltiin ensimmäisen kerran Windows Server 2012 R2 –palvelinkäyttöjärjestelmän mukana, mutta on saatavilla sen lisäksi vanhempiinkin versioihin. Tämän avulla saadaan siis ajettua haluttuja konfiguraatioita palvelimiin sekä työasemiin. Identtiset asetukset saadaan laitettua tämän avulla myös useampiin työasemiin kerralla, se voidaan tehdä manuaalisesti tai automaation avulla. DSC:n avulla voidaan esimerkiksi asentaa sekä muokata palvelimen ominaisuuksia ja roolia, muokata rekisteriä, hallita tiedostoja ja kansioita, hallita prosesseja, hallinnoida ryhmiä ja käyttäjätilejä, asentaa ohjelmistoja ja paketteja ja ajaa Windows PowerShell skriptejä.

DSC:ssä on myös mukana monitorointi ja raportointi tekniikka, eli se osaa hälyttää ja kertoa muutoksista ja mikäli tarvetta niin korjata ne automaatiolla. Skripti jaetaan haluamiinsa palikoihin eli ja sinne merkitään konfiguroitavat kohdat Node komennoilla. Halutut muutokset ja säädöt tehdään erilliseen asetustiedostoon. DSC:n käyttöön löytyy omat cmdlets-komennot kuten esimerkiksi Start-DscConfiguration ja Get-DscConfigurationStatus. Näillä voidaan käynnistää tehty konfiguraatio ja tarkistaa menikö se onnistuneesti läpi. DSC konfigurointiin, käyttämiseen ja skripteissä käytettäviin komentoihin löytyy Microsoftin sivuilta erittäin kattavat oppaat. (Microsoft 2017.)

WMF eli Windows Management Framework on asennusalusta ja paketti Windows sekä Windows Server käyttöjärjestelmille. Se helpottaa eri käyttöjärjestelmien ja alustojen välistä käyttämistä. Windows Management Frameworkin kautta onnistuu tiettyjen komponenttien asennus ja päivitys.

WMF paketin asennuksen mukana tulee seuraavat komponentit tai päivitykset riippuen käyttöjärjestelmästä: Windows PowerShell, Desired State Configuration, ISE, Remote Management, Management Instrumentation, Web Services, Inventory Logging, Server Manager CIM Provider.

WMF:n versio riippuu käytössä olevasta käyttöjärjestelmästä. Windows Server 2016 mukana tulee uusin WMF versio eli 5.1, joka on mahdollista ladata jälkikäteen Windows Server 2012 R2 käyttöjärjestelmään. Sen avulla saa siis helposti otettua esimerkiksi uudet Windows Server 2016 käyttöjärjestelmässä esitellyt PowerShellin ominaisuudet käyttöön vanhemmissakin käyttöjärjestelmissä. Asennusohjeet ja paketti löytyvät Microsoftin sivuilta. Asennus onnistuu lisäksi Windowsin asennuskeskuksen kautta. (Microsoft n.d.f.)

Pester on PowerShell skriptausta varten kehitetty testauskehys ja työkalu. Sen avulla pystyy luomaan testejä omille skripteille. Kyseessä on hyvä työkalu, jos kehitys on testivetoista. Idea tässä on, että ennen kuin kirjoitetaan mitään koodia, niin tehdään sitä varten erilaisia testejä. Tässä työssä ei kuitenkaan syvennytä itse testaukseen tai sen menetelmiin tarkemmin.

Pester on ladattavissa ilmaiseksi GitHubista. Asennus on hyvin yksinkertaista, ladataan kyseinen tiedosto, puretaan Moduulit-kansioon, sen jälkeen se pitää ottaa vielä käyttöön PowerShell sessiossa Get-Module komennolla. Tämän jälkeen voi tehdä omia testejään ja ajaa niitä PowerShellissä. Pesteriä kehitetään aktiivisesti yhteisön toimesta ja siihen löytyy paljon käyttäjien tekemiä ohjeistuksia. (Jares 2014.)

PowerShell Studio on editori ja työkalu PowerShell skriptien tekemiseen, kyseessä on todella laaja työkalu. Skriptien tekemisen lisäksi sillä pystyy tekemään ja suunnittelemaan graafisia käyttöliittymiä skripteille. Skriptejä voi myös konvertoida tällä suoraan exe-tiedostoiksi tai msi-asennuspaketeiksi.

PowerShell Studiolla voi sen lisäksi luoda moduuleita suoraan aiemmista luoduista funktioista. Tällä voi sen lisäksi luoda funktioita omalla työkalullaan. Skriptejä tehtäessä käytössä on laajat kirjastot joista voi hakea eri komentoja, funktioita ja objekteja skriptejä varten.

PowerShell Studio on siis hyvin pitkältä samanlainen kuin ISE editori, mutta ominaisuuksiltaan vielä paljon laajempi. Tutkituista ohjelmista tämä oli laajin ja vaikutti selkeäkäyttöiseltä. Kyseessä on maksullinen ohjelma, jonka kokeiluversion voi ladata Saphien Technologiesin kotisivuilta. Kyseessä on siis kolmannen osapuolen julkaisema ohjelma ja työkalu. (Sapien Technologies 2018.)

PowerShell Tools for Visual Studio 2017 ei ole oma ohjelmansa vaan ladattava lisäosa Visual Studioon. Tämä on maksuton lisäosa, joten mikäli Visual Studio löytyy ennestään tätä kannattaa harkita. Lisäosa tuo Visual Studioon paljon lisäominaisuuksia jotka helpottavat PowerShell Skriptien tekemistä, ajamista sekä niiden testausta. Skriptejä on mahdollista ajaa paikallisesti tai etänä. Tämä tukee sen lisäksi myös testausta Pesterin avulla.

Lisäosasta on saatavilla perusversion lisäksi myös maksullinen Pro-versio, joka on ominaisuuksiltaan vielä laajempi. Sen avulla voi suunnitella graafisia käyttöliittymiä, paketoida suoritettavia tiedostoja ja voi kääntää C#

koodia PowerShelliksi. Mikäli on tarvetta tehdä monimutkaisia PowerShell skriptejä, on tämä silloin hyvä valinta. (Driscoll 2018.)

2.5 **Cmdlet-komennot**

Microsoft PowerShellissa käytetään Command-Let komentoja, joista käytetään yleensä nimitystä Cmdlet. Cmdlet-komennot ajetaan komentokehotteella ja ne ajavat aina tietyn toimenpiteen. PowerShellin cmdlet-komennot koostuvat aina verbi ja substantiivi parista, esimerkiksi komento Get-Command. Tuossa komennossa verbi määrittelee sen, mikä toimenpide tehdään. Substantiivi taas määrittelee kohteen, jolle toimenpide tehdään. Get-Command cmdlet siis listaa kaikki käytössä olevat komennot. Näille komennoilla voidaan syöttää erilaisia parametrejä lisätiedoksi. Parametriksi voidaan siis esimerkiksi antaa tietyn työaseman tai käyttäjän nimi.

Cmdletseille ja muille PowerShell komennoille on käytössä erilaisia aliaksia eli lyhenteitä, samaan tapaan kuin komentokehotteessakin vastaavia. Käytössä olevat lyhenteet saa helpoiten selville komennolla Get-Alias. (Microsoft 2018b.)

2.6 **Peruskomentoja**

PowerShellissä on uusimpien versioiden myötä jo satoja erilaisia Cmdlets-komentoja, tässä kappaleessa käydään läpi niistä muutamia hyödyllisiä peruskomentoja.

Get-Help-komento näyttää tietoa PowerShellin komennoista ja osista. Esimerkiksi jos haluaa saada lisätietoa komennosta Get-Process, niin kirjoittamalla Get-Help Get-Process saa auki tiedot siitä. Komennosta näkee sen syntaksin, mitä parametrejä voi syöttää ja missä muodossa. Sen lisäksi kyseisestä komennosta näkyy aliakset eli lyhyemmät komennot jolla sitä voi käyttää. Tämä on yksi hyödyllisimpiä ja käytetyimpiä komentoja. Help-systeemi toimii PowerShellissä erittäin hyvin. Pelkällä Get-Help komennolla näkyy tietoa tarkemmin, miten sitä käytetään.

Get-Command ja Show-Command komennot molemmat näyttävät käytettävissä olevat cmdlets-komennot. Get-Commandilla ne aukeavat listassa ja Show-Command avaa erillisin ikkunan josta ne selattavissa. Samalla tapaa Get-Verb taas näyttää kaikki käytössä olevat verbit listattuna.

Get-Process näyttää prosessit jotka ovat käynnissä ja niistä tiedot, esimerkiksi resurssien käytön. Samalla tapaa taustalla pyörivistä palveluista saa tiedot helposti näkyviin Get-Service-komennolla.

Get-Module näyttää PowerShellissä käytössä olevat moduulit ja niiden kautta näkee mitä cmdlets-komentoja voit käyttää. Osa cmdlets-komennoista tulee tietyn asennettavan moduulin mukana.

Get-Host komento näyttää tarkemmat tiedot käytössä olevasta PowerShellistä tai ISE:stä, tärkeimpänä tietona mikä versio on käytössä. Samat tiedot saa näkyville myös käyttämällä $PSversion muuttuja.

Compare-Object komennon avulla voi verrata kahta eri objektia eli esimerkiksi kahta tekstitiedostoa ja niiden eroja. Tämä on hyödyksi, jos esimerkiksi pitää etsiä eroja kahden aktiivihakemistossa olevasta käyttäjätilistä.

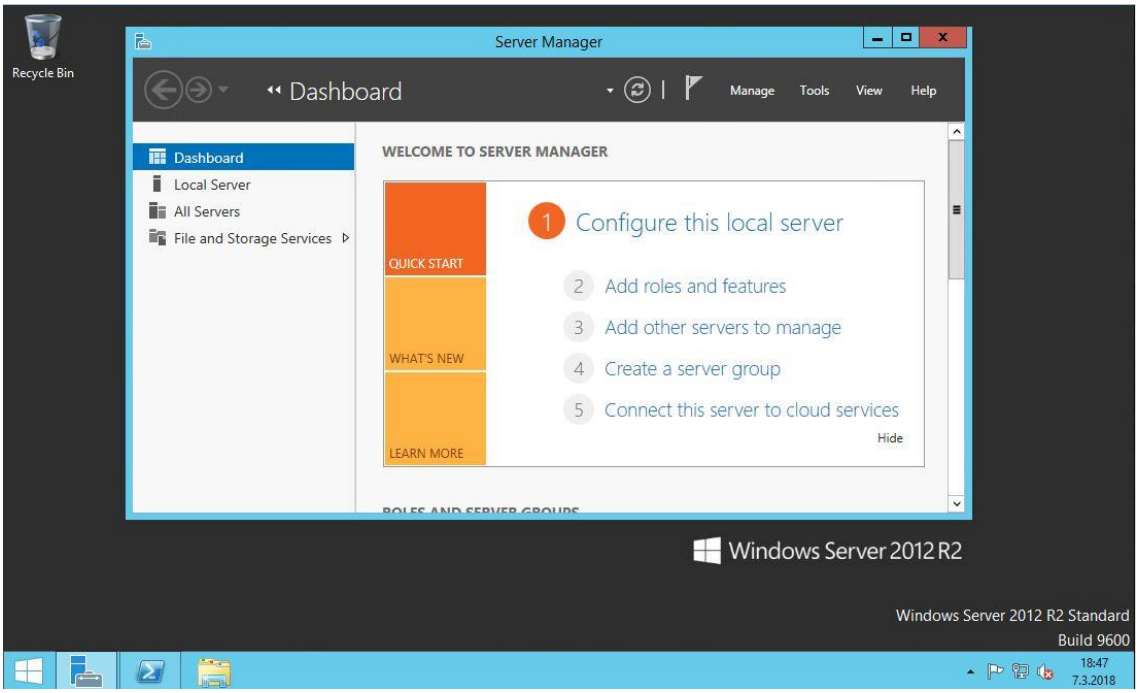
ConvertTo-HTML komennon avulla saa halutut tiedot muutettua HTML-muotoon, samalla tapaa onnistuu myös muut tiedostotyypit. Mikäli ruudulle tulostettavia tietoja on paljon, niin tämä on niissä tilanteissa hyödyllinen. Muitakin vastaavia komentoja on, Export-CSV komennolla saa laitettua tiedot CSV-tiedostoon jonka saa Excelillä auki.

Write-Progress komento näyttää edistymispalkin, eli jos ajaa jotain pidempää skriptiä niin tämän avulla voi näyttää näppärästi sen edistymisen. Kyseiseen komentoon voi valita minkä asian edistymisen se näyttää, esimerkiksi jos ajaa skriptillä jotain päivityksiä niin voi näyttää niiden edistymisen prosentteina.

Whatif- ja Confirm-komennot eivät varsinaisesti ole cmdletsejä, vaan nämä ovat parametreja joita voi käyttää niiden kanssa. Kun syöttää cmdletsin perään -Whatif, niin nimensä mukaisesti se näyttää mitä tapahtuisi, kun ajaa komennon. Tällä siis voi helposti testata ja varmistaa komentojen lopputuloksen. Samalla tavalla syöttämällä komennon perään -Confirm saadaan ilmoitus, että haluatko ajaa komennon. Nämä molemmat paramerit ovat siis hyödyllisiä skriptien testauksessa ja ajamisessa. (Microsoft n.d.g.)

3 WINDOWS SERVER 2012

Tässä opinnäytetyössä pääasiallinen palvelinalusta on Windows Server 2012 ja tarkennuksena sen uudempi versio eli Windows Server 2012 R2. Kyseisessä käyttöjärjestelmässä mukana tulee PowerShellin versio 4.0 ja PowerShell ISE skriptauseditori. Näistä löytyy oletuksena mukana 32- ja 64-bittiset versiot.



Kuva 4. Windows Server 2012 perusnäkymä.

Windows Server 2012 on kymmenes Microsoftin julkaisema palvelinkäyttöjärjestelmä, sen korvasi sitä edeltävän Windows Server 2008 palvelinkäyttöjärjestelmän. Se kehitettiin Windows 8 käyttöjärjestelmän rinnalla. Molemmat ovat ulkoasultaan ja valikoiltaan hyvin samanlaisia. Windows Server 2012 julkaistiin valmistajille elokuussa 2012 ja virallinen julkaisu oli samana vuonna syyskuussa. Ulkoasun ja käyttöliittymän lisäksi merkittäviä uudistuksia oli esimerkiksi uusi versio Server Managerista, mahdollisuus vaihtaa graafisen- ja komentoliittymän välillä, uusi Hyper-V versio jossa paljon uudistuksia virtualisoinnin osalta. Windows Server 2012 otettiin käyttöön uusi tiedostojärjestelmä ReFS(Resilient File System) joka korvasi vanhentuneen NTFS tiedostojärjestelmän. (Tampskins 2015.)

3.1 **Windows Server 2012 R2**

Windows Server R2 on numeroltaan yhdestoista Windows palvelinkäyttöjärjestelmä, joka on käytännössä päivitys sitä edeltäneeseen Windows Server 2012 palvelinkäyttöjärjestelmään. Se julkaistiin lokakuussa 2013 ja sen alustana käytettiin Windows 8.1 käyttöjärjestelmää. Suurimpia uudistuksia oli uusi PowerShell 4.0, Office 365 integraatio, IIS eli Web Serverin 8.5 versio ja kehittyneemmät ryhmäkäytännöt. R2 palvelinkäyttöjärjestelmästä on julkaistu samat neljä versiota kuten edeltäjästään eli Foundation, Essentials, Standard ja Datacenter. (Tampskins 2015)

Foundation on rajoitettu ja karsittu versio, jossa on rajoitettu käyttäjien määrä, maksimi määrä on 15 käyttäjää sekä muistin määrä on maksimissaan 32 GB keskusmuistia. Tämä versio sopii siis hyvin yksittäisten palveluiden ylläpitoon ja tehokkaaseen yleiskäyttöön. Tämä versio ei tue ollenkaan virtualisointia.

Essentials on ominaisuuksiltaan hyvin samankaltainen kuin Foundation, se kuitenkin tukee hieman enemmän käyttäjiä eli 25:tä ja palvelimen maksimimuisti on 64 GB keskusmuistia. Tässä versiossa ei ole myöskään mitään virtualisointi ominaisuuksia. Tämä versio sopii siis parhaiten pienille yrityksille ja vastaaville käyttöryhmille.

Standard-versiossa on virtualisointi tuki rajoitetusti, se tukee maksimissaan kahta virtuaalikonetta. Muuten ominaisuuksiltaan ja palvelinrooleiltaan Standard vastaa hyvin pitkältä Datacenteriä. Standard on siis hyvä valinta, mikäli ympäristö ei ole virtualisoitu tai on virtualisoitu kevyesti.

Datacenter on ominaisuuksiltaan kaikista laajin versio ja virtualisointi omaisuudet ovat kaikista laajimmat, virtuaalikoneiden määrää ei ole rajattu. Tästä syystä Datacenter-versio on paras versio, mikäli ympäristö on laajalti virtualisoitu ja on tarvetta virtuaalikoneille. Laitteistopuolella Datacenterissä on sama maksimimuisti määrä kuin Standardissa eli 64 GB keskusmuistia. (Thomas Krenn 2018.)

Näiden lisäksi Windows Server R2 käyttöjärjestelmästä on versiot Microsoft Hyper-V® Server 2012 R2, joka keskittyy ominaisuuksiltaan virtualisointiin. Tiedostopalveluihin ja verkkotallennukseen keskittyvä Windows Storage Server 2012 R2 Standard ja siitä hieman sitä vastaava Windows Storage Server 2012 R2 Workgroup, joka eroaa lähinnä laitteisto rajoituksiltaan. Nämä kaikki kolme versiota ovat hyvin karsittuja verrattuna Windows Server 2012 R2 –palvelinkäyttöjärjestelmän muihin versioihin ja keskittyvät tiettyyn osa-alueeseen. (Microsoft 2014.)

3.2 **Windows Server 2016**

Windows Server 2016 on Microsoftin uusin palvelinkäyttöjärjestelmä ja se kehitteen yhdessä Windows 10 käyttöjärjestelmän kanssa. Sen edeltäjä oli aiemmin tässäkin työssä esitelty Windows Server 2012 R2. Windows Server 2016 julkaistiin virallisesti lokakuussa 2016.

Ulkoasultaan Windows Server 2016 ei juurikaan eroa edeltäjästään Windows Server 2012:sta. Perusnäkymä on lähes täysin samanlainen. Windows Server 2016 kuitenkin tarjoaa useita uusia ominaisuuksia ja etuja aiempiin Windows Server käyttöjärjestelmiin nähden.

Windows Server 2016 palvelinkäyttöjärjestelmässä on panostettu ja keskitytty entistä enemmän pilvi- ja virtualisointi ominaisuuksiin. Mukana tulee Nano Server ominaisuus, joka on pienikokoinen ja karsittu asennusversio. Nano Serverin kautta voi optimoida palvelimen yksityisiin pilvipalveluihin ja datakeskuksiin, sekä sen avulla voi suorittaa erilaisia webpalveluita kuten vaikkapa DNS-serveriä. Nano Server vie tilaa huomattavasti vähemmin ja on kevyempi toimivuudeltaan. Virtualisoinnin osalta esitellään myös uusi versio Hyper-V:stä.

Uudistuneita merkittäviä osioita on muun muassa Aktiivihakemisto, sen tunnistuspalvelut eli ADFS, Windows Defender ja tiedostopalvelut. PowerShellistä mukana tulee uusi PowerShell 5.0 versio ja sen kautta paljon uusia ominaisuuksia PowerShelliin sekä skriptaukseen.

Windows Server 2016 palvelinkäyttöjärjestelmästä on saatavilla Standard ja Datacenter versiot. Datacenter versio on siis keskittynyt tiedostopalvelin ominaisuuksiin, samalla tapaa kuin Windows Server R2 Datacenter. Näiden lisäksi on pienille yrityksille ja organisaatiolle suunnattu Essentials. Näiden kolmen version lisäksi on julkaistu virtualisointiin keskittyvä Hyper-V Server 2016. (TechRepublic 2017.)

4 PALVELIMEN MUUT OMINAISUUDET

Tässä kappaleessa käydään läpi lyhyesti vielä muut palvelimen ominaisuudet ja käydään niiden osalta teoriaa läpi. Palvelimella otetaan käyttöön aktiivihakemisto, josta alla lyhyt esittely. Aktiivihakemiston lisäksi työssä otetaan käyttöön Office 365 -ympäristö, joten siitäkin alla lisätietoa. Vaikka se ei varsinaisesti ole palvelimen ominaisuus, sen on mahdollista integroida siihen.

Aktiivihakemisto(AD) on Microsoftin kehittämän teknologia ja työkalu käyttäjien, työasemien sekä muiden laitteiden hallintaan. Se on yksi Windows Server-palvelinkäyttöjärjestelmien tärkeimmistä ominaisuuksista. Aktiivihakemiston avulla onnistuu suurenkin organisaation käyttäjien ja työasemien hallinta.

Aktiivihakemiston avulla ylläpitäjät voivat luoda ja hallita toimialueita, käyttäjiä sekä objekteja verkon sisällä. Ylläpitäjä voi siis esimerkiksi luoda käyttäjiä sekä käyttöoikeusryhmiä ja antaa ryhmillä kyseisillä käyttäjillä oikeuksia vaikkapa tiettyyn hakemistoon.

Rakenteeltaan aktiivihakemisto muodostuu kolmesta eri tasosta, toimialueista, puista ja metsistä. Toimialueessa voi olla paljon eri objekteja kuten käyttäjiä ja konetilejä ja puusta voi taas koostua useammista toimialueista. Ylin taso aktiivihakemistossa eli metsä taas koostuu yhdestä tai useammasta puusta.

Aktiivihakemisto tarjoaa useita eri palveluita, kuten palvelun sertifikaattien hallintaan, toimialuepalvelut, kirjautumispalvelut ja käyttöoikeuksien hallinnan. Nämä palvelut helpottavat käyttäjien hallintaa ja ylläpitoa. (TechTerms 2017.)

Office 365 on Microsoftin Office tuoteperheen versio, jossa voit käyttää ohjelmistoja pilvessä verkon kautta. Lisenssejä on useita erilaisia ja eri hintaisia, mutta yleensä siihen sisältyy käyttöoikeus perus Officen ohjelmiin eli Word, Excel, PowerPoint, OneNote ja Outlook. Ohjelmia voi käyttää verkon yli tai ohjelmat ovat asennettavissa omalle työasemalle. (Microsoft n.d.h.)

Office 365:ssä käyttäjien tunnistaminen ja hallinta tapahtuu pilvipohjaisen Azure Active Directoryn kautta. Azure AD ja paikallinen AD on kuitenkin mahdollista integroida, jolloin käyttäjätilit ovat yhteydessä toisiinsa. Tämä helpottaa käyttäjien hallintaa ja esimerkiksi tällöin käyttäjillä on sama salasana paikallisesti ja pilvessä.
(Microsoft 2018c.)

5 WINDOWS SERVER -PALVELIMEN HALLINTA

Työssä käytettiin palvelin alustoina Windows Server 2012 sekä Windows Server 2016 palvelinkäyttöjärjestelmiä. Pääasiallinen alusta opinnäytetyössä oli Windows Server 2012 R2 Standard. Kyseessä oli pilvessä olevia virtuaalikoneita, jotka tilattiin vCommanderin kautta. Palvelimen hallintaa testattiin käytännössä PowerShellillä. PowerShellistä käytettiin pääasiassa versiota 5.0. Käytännöt suoritukset tehtiin pääasiassa ISE-skriptauseditorissa. PowerShell ISE avataan ja käytetään järjestelmänvalvojana, muussa tapauksessa kaikki muutokset eivät mene läpi ja toimi.

Palvelimen hallinta pyrittiin tekemään mahdollisimman paljon PowerShelliä käyttämällä, koska se oli tämän opinnäytetyön tarkoitus. Suurin osa tehdyistä asioista onnistuu toki sen lisäksi palvelimen graafisen käyttöliittymän kautta, mutta niitä ei lähdetä tarkemmin tästä työssä avaamaan.

Palvelimella testattiin Aktiivihakemiston hallintaa PowerShellillä. Ensiksi se otetaan käyttöön palvelimella. Käytännössä testattiin sen lisäksi luoda Aktiivihakemistoon eri objekteja kuten käyttäjätili ja konetili. Aktiivihakemiston lisäksi otettiin käyttöön Office 365 ja kokeiltiin sen hallintaa. Siihen lisättiin käyttäjätili ja hallittiin sitä PowerShellillä.

5.1 Windows Server 2012 ja 2016 asennus sekä käyttöönotto

Windows Server 2012 palvelinkäyttöjärjestelmän asennus tapahtui Hämeenlinnan ammattikorkeakoulun tarjoaman vCommander-sivuston kautta, kyseessä oli VMwaren virtuaalityöasema. Varsinaista käyttöjärjestelmän asennusta ei siis tässä työssä suoritettu.

Käyttöönotto oli yksinkertaista ja helppoa. Ensiksi tehtiin tilaus vCommander-sivuston kautta ja kun tilaus oli valmis, niin käytettävissä oli halutut virtuaalikoneet. Sen kautta kyseinen virtuaalikoneen sai laitettua käyntiin ja pääsi kirjautumaan sinne sisään. Työasemalla oli valmiina luotuna käyttäjätunnus, jolla oli järjestelmänvalvojan oikeudet.

Windows Server 2016 Standard palvelinkäyttöjärjestelmä otettiin samalla tapaa käyttöön kuin Windows Server 2012. Kyseessä oli virtuaalinen työasema, jota käytettiin VMwaren kautta.

5.2 Etäyhteyden muodostaminen PowerShellillä

Tässä kappaleessa käydään läpi etäyhteyden luominen ja ottaminen PowerShell-istunnun avulla. Ensiksi pitää sallia etäyhteyksien ottaminen palvelimeen. Se onnistuu PowerShellissä komennolla Enable-PSRemoting. Kyseinen komento käynnistää WinRM palvelun eli etäyhteyspalvelut ja asettaa ne käynnistymään automaattisesti järjestelmän kanssa. Sekä asettaa palomuurisääntöihin sallituiksi sisään tulevat yhteydet.

Etäyhteyden voi ottaa myös suoraan PowerShell ISE editorissa. Editorista löytyy File-välilehdeltä kohta New Remote PowerShell Tab, johon syötetään halutun palvelimen etäyhteyden nimi sekä kirjautumistunnus. Mikäli oikeudet on kunnossa, niin etäyhteys aukeaa uuteen välilehteen.

Testauksessa käytetyt työasemat olivat samassa toimialueessa, joten muita asetuksia ei tarvitse muuttaa. Mikäli työasemat eivät ole toimialueessa, täytyy erikseen asettaa kyseiset työasemat sallituiksi. Tässä työssä ei kuitenkaan käydä tarkemmin niitä asetuksia läpi.

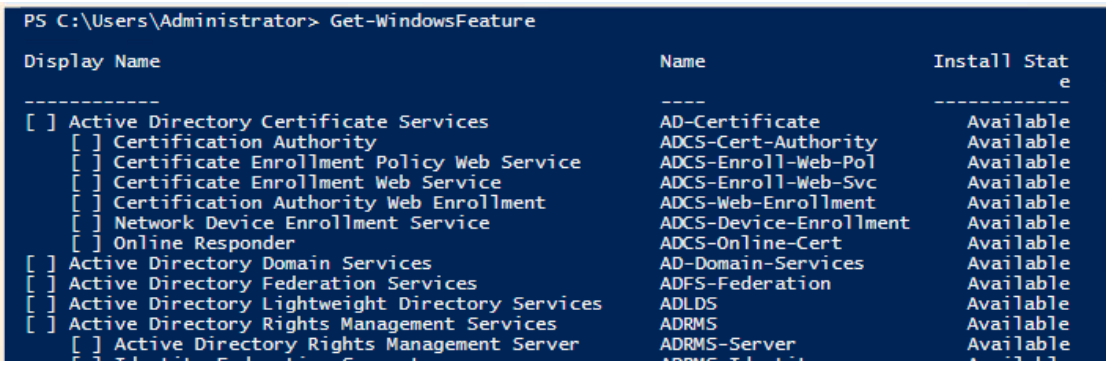
Etäyhteyksien sallimisen jälkeen voidaan ajaa etänä yksittäisiä komentoja Invoke-Command komennon avulla. Työssä testattiin kuitenkin istunnon avaamista, jotta ei täydy ajaa jokaista haluttua komentoa erikseen Invoke-Command komennon avulla.

Ensiksi avataan istunto, se onnistuu komennolla:
Enter-PSSession -Computername Työasema -Credential Käyttäjä.

Komennossa työasema kohtaan tulee siis työaseman nimi tai IP-osoite. Käyttäjä kohtaan samalla lailla käyttäjätilin nimi jolla yhdistetään.

5.3 Aktiivihakemiston käyttöönotto

Ensiksi tarkistettiin, mitä palveluita on käytössä palvelimella, se onnistuu helpoiten Get-WindowsFeature komennolla PowerShellissä. Kyseinen komento näyttää saatavilla olevat ja asennetut palvelimen roolit sekä palvelut. Komennolla nähdään, että aktiivihakemistoon liittyvät palvelut ovat asennettavissa. Aktiivihakemisto-rooli ei siis ole palvelimelle valmiina asennettuna.



Kuva 5. Get-WindowsFeature-komento.

Kyseinen rooli pitää siis ottaa käyttöön. Se onnistuu PowerShellin kautta komennolla:

ADD-windowsfeature AD-Domain-Services.

Tämän avulla saadaan käyttöönotettua halutettu rooli eli Aktiivihakemiston toimialuepalvelu (Active Directory Domain Services). Tämän lisäksi otetaan käyttöön tähän liittyvät moduulit komennolla:

Import-Module ADDSDeployment.

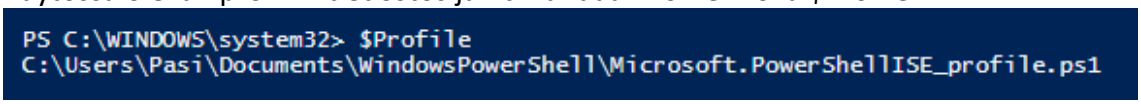
Lopuksi palvelin pitää käynnistää uudelleen, jotta roolin ja moduulin asennus tulee voimaan.

Seuraavaksi palvelimelle luodaan uusi toimialue. Toimialueelle annettiin nimeksi testi.local. Toimialue luodaan PowerShellissä Install-ADDSForest komennolla. Kyseiselle komennolla annettiin parametriksi -DomainName testi.local. Parametrisi voi antaa paljon muitakin tietoja ja tehdä määrityksiä. Tässä kuitenkin mentiin oletusasetuksilla. Komennon syöttämisen jälkeen pitää se vielä vahvistaa syöttämällä järjestelmän valvojan tunnukset. Lopuksi taas palvelin pitää uudelleen käynnistää, jotta muutokset tulevat voimaan. Sen jälkeen kyseinen toimialue on luotu ja palvelin on lisättynä siihen.

Viimeisenä toimenpiteenä asennettiin vielä Active Directory -moduuli, jotta hallinta sen hallinta onnistuu PowerShellillä. Kyseinen moduuli asennettiin komennolla Import-Module ActiveDirectory. Tämän jälkeen päästään hallinnoimaan Aktiivihakemisto PowerShellin kautta. Seuraavana toimenpiteenä luodaan PowerShelliin profiili. Muussa tapauksessa jokaisen session aluksi pitäisi ottaa Aktiivihakemisto -moduuli käyttöön, jos haluaa hallita sitä PowerShellin kautta. Aluksi tarkistetaan PowerShellissa, onko profiilia käytössä, joka onnistuu komennolla Test-Path $Profile. Tässä tilanteessa ei ole profiilia vielä käytössä, joten tuloksena tulee False. Uuden profiilin luonti onnistuu New-Item Cmdletsillä.

Komento on:
New-Item –Path $Profile –Type File –Force.

Käytössä olevan profiilin tiedostosijainti nähdään komennolla $Profile.



Kuva 6. Käytössä oleva Windows PowerShell profiili.

Profiilin luomisen jälkeen profiilin sisältöä voi muokata PowerShellissä komennolla notepad $Profile. Kyseiseen profiilitiedostoon voi lisätä muitakin komentoja, mutta tässä tapauksessa lisättiin Import-Module Active-Directory.

Aktiivihakemiston hallinnointi onnistuu toki sen lisäksi palvelimen graafisenkäyttöliittymän kautta. Joten se otettiin käyttöön. Sitä varten pitää ensiksi ottaa tarvittava Server Manager -moduuli käyttöön, se tapahtuu komennolla Import-Module ServerManager ja sitten asennettaan tarvittava ominaisuus Add-WindowsFeature RSAT-ADDS-Tools. Nyt aktiivihakemistoa voi hallita graafisenkäyttöliittymän kautta, se löytyy Server Managerista Tools-valikon alta.

5.4 Objektin luominen, poistaminen ja muokkaaminen

Tässä kappaleessa luodaan PowerShellin avulla Aktiivihakemistoon käyttäjätili, konetili, ryhmä ja organisaatioyksikkö. Sen lisäksi poistetaan eri objekteja ja muokataan niiden tietoja. PowerShellin lisäksi nämä asiat on mahdollista tehdä palvelimen graafisen käyttöliittymän kautta.

Uuden käyttäjän luominen onnistuu New-ADUser Cmdlets-komennolla. Parametriksi annetaan nimi sekä käyttäjän UPN eli UserPrincipalName, joka on käyttäjän kirjautumisnimi. Käyttäjä luotiin komennolla:

New-ADUser -Name "Pasi" -UserPrincipalName pasi@testi.local.

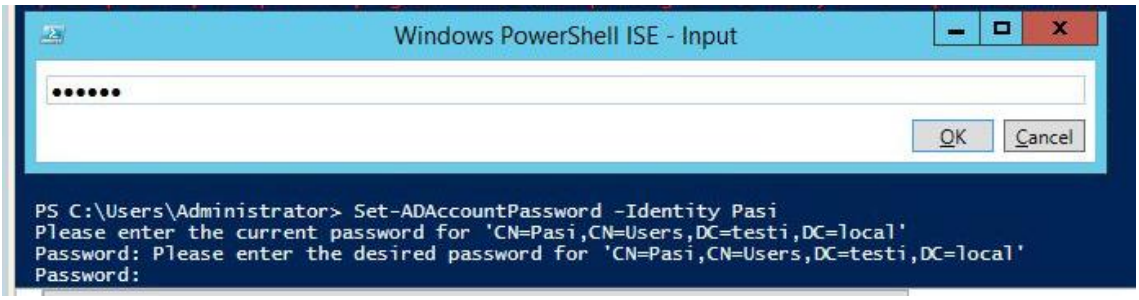
Kun kyseinen komento on ajettu, voidaan tarkistaa, että tili on luotu komennolla:

Get-ADUser Pasi | Select-Object Name.

PowerShellin lisäksi käyttäjätilin luonnin onnistumisen voi tarkistaa aktiivihakemiston graafisesta käyttöliittymästä, josta löytyy nyt juuri luotu

Pasi-niminen käyttäjä. Käyttäjätilille voidaan antaa muitakin tietoja, helpoiten kaikki mahdolliset parametrit saa selville Get-Help-komennolla. Ainoat pakolliset parametrit käyttäjälle ovat nimi sekä UPN.

Kyseisillä komennoilla luodulla käyttäjätilillä ei kuitenkaan ole salasanaa asetettuna eikä tili ole aktiivinen. Salasanan asettaminen käyttäjä tilille onnistuu komennolla: Set -ADAccountPassword -Identity Pasi. Salasana syötetään erilliseen ikkunaan, joka on kuvassa 7.



Kuva 7. Salasanan asettaminen käyttäjätilille.

Salasanan asettamisen jälkeen tili aktivoidaan komennolla Enable-ADAccount -Identity Pasi. Näiden toimenpiteiden jälkeen käyttäjätili on käyttökunnossa. Komentojen läpi menon voi tarkistaa hakemalla käyttäjätilin tiedot Get-ADUser komennolla:

Get-ADUser Pasi -Properties Enabled | Format-Table Name, Enabled

Kyseinen komento hakee annetun käyttäjän tiedot ja näyttää True / False arvon onko tili aktiivinen vai ei. Tämä tieto tulee Enabled sarakkeen mukaan. Format-Table komennolla on mahdollista muuttaa mitä tietoja ja missä muodossa ne tulostuvat. Tässä komennossa tietoja on Name sekä Enabled-kentät.

Kaikki yllä käydyt toimenpiteet voidaan yhdistää yhteen komentoriviin, jotta niitä ei täydy tehdä erikseen jokaisen käyttäjän kohdalla. Tämä onnistuu putkittamalla komennot alla olevanmukaisesti. Komennossa luodaan uusi käyttäjä ja sille annettaan nimeksi Pasi1. Sen jälkeen kysytään erillisellä ikkunalla salasana ja lopuksi aktivoidaan tili.

Komento on kokonaisuudessaan:

New-ADUser -Name Pasi1 -AccountPassword(Read-Host -AsSecureString "Tilin salasana") | Enable-ADAccount

Konetili voidaan luoda aktiivihakemistoon lähes samalla tapaa kuin käyttäjäkin. Se onnistuu New-ADComputer komennolla ja sille annetaan parametriksi haluttu konetilin nimi -Name "Kone1". Kyseinen konetili luodaan

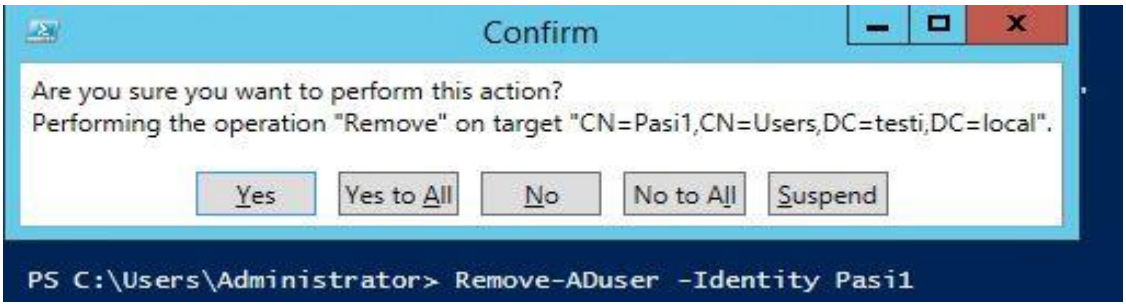
oletuksena Computers-nimisee OU:hun, mutta -path attribuutilla voi määrittää halutessaan eri sijainnin. Sama asia on käyttäjätilien luonnissa, ne menevät oletuksena Users-nimiseen organisaatioyksikköön aktiivihakemistossa. Läpikäydyissä testauksissa sijainti on kuitenkin oletus.

Organisaatioyksikön luonti tapahtuu komennolla New-ADOrganizationalUnit. Sille annettaan pakollisena parametrina nimi. Kokonaisuudessaan komento on siis:

NEW-ADOrganizationalUnit -Name “TestiOU”.

Ryhmän luominen onnistuu komennolla New-ADGroup ja sille parametriksi annettaan nimi sekä pakollinen tieto joka on GroupScope eli miten ryhmää voi käyttää sekä miten se näkyy. Tässä sille määritellään arvoksi Global, eli ryhmä näkyy ja on käytettävissä aktiivihakemisto metsän kaikissa toimialueissa. Muita vaihtoehtoja GroupScopen olisi DomainLocal, jossa ryhmä näkyisi vain kyseissä toimialueessa. Universal GroudScopella ryhmä taas olisi käytettävissä myös muissa mahdollisissa aktiivihakemiston metsissä ja niiden toimialueissa.

Luotujen objektien poistaminen onnistuu Remove-alkuisilla komennoilla, esimerkiksi käyttäjätili poistetaan komennolla Remove-ADUser ja annetaan parametriksi halutun käyttäjätilin nimi. Poistotilanteessa tulee oletuksena kuvassa 8 oleva varmistus ikkuna. Muiden objektien poisto onnistuu vastaavilla menetelmillä.



Kuva 8. Käyttäjätilin poisto.

Aktiivihakemiston objektien siirto tapahtuu Move-ADObject cmdlet-komennolla. Kyseiseen komentoon pitää antaa vähintään parametriksi Identity, jolla määritellään mikä siirretään. Sen lisäksi annettaan polku mistä siirretään ja mihin. TargetPath parametrillä määritellään polku, johon kyseinen objekti siirretään. Tässä tapauksessa esimerkiksi siirretään käyttäjätili organisaatioyksiköstä toiseen.

Testauksessa käytettiin komentoa:
Move-ADObject -Identity Pasi2 -TargetPath "OU=TestiOU ,DC=testi".

Luotuja aktiivihakemiston objekteja voi muokata Set-ADObject komennolla jolla voidaan asettaa halutulle objektille esimerkiksi Description kenttään haluttu teksti. Kuten kuvassa 9 asetettiin käyttäjätilille määritelty teksti. Parametriksi voi antaa minkä tahansa muunkin tiedon, samalla menetelmällä voi siis asettaa esimerkiksi käyttäjän puhelinnumeron.

```
PS C:\Users\Administrator> Set-ADObject 'Cn=Pasi2, CN=Users, DC=testi, DC=local' -Description 'testi kirjoitus'
```

Kuva 9. Käyttäjätilin tietojen muokkaus.

Objekteja voidaan sen lisäksi uudelleen nimetä Rename-ADObject cmdletsillä. Pakollisena parametrina pitää antaa objektin nimi ja uusi haluttu nimi annetaan parametrillä NewName. Esimerkiksi uudelleen nimetään käyttäjätili Pasi2 nimelle Pasi2Uusi komennolla:

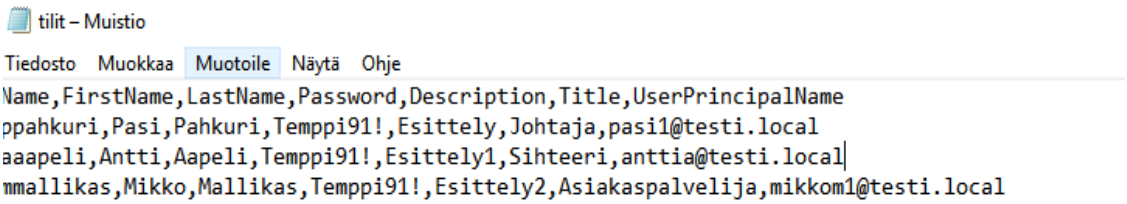
Rename-ADObject ’Cn=Pasi2, CN=Users, DC=testi, DC=local’ -NewName Pasi2Uusi.

5.5 Käyttäjätilin luominen Aktiivihakemistoon

Aiemmassa kappaleessa käytiin läpi jo käyttäjätilin luomista aktiivihakemistoon, mutta tässä kappaleessa testataan luontia vielä kahdella eri menetelmällä ja monimutkaisemmalla PowerShell skriptillä.

Ensimmäisenä luodaan kerralla useita käyttäjätilejä, joiden tiedot ovat CSV-tiedostossa.

Aluksi luodaan tarvittavat tiedosto eli tilit.csv, tähän tiedostoon syötetään tekstimuodossa seuraavaksi luotavien tilien tiedot. Haluttuja tietoja on käyttäjän nimi, etu- ja sukunimi, salasana, esittely, nimike sekä kirjautumisnimi.

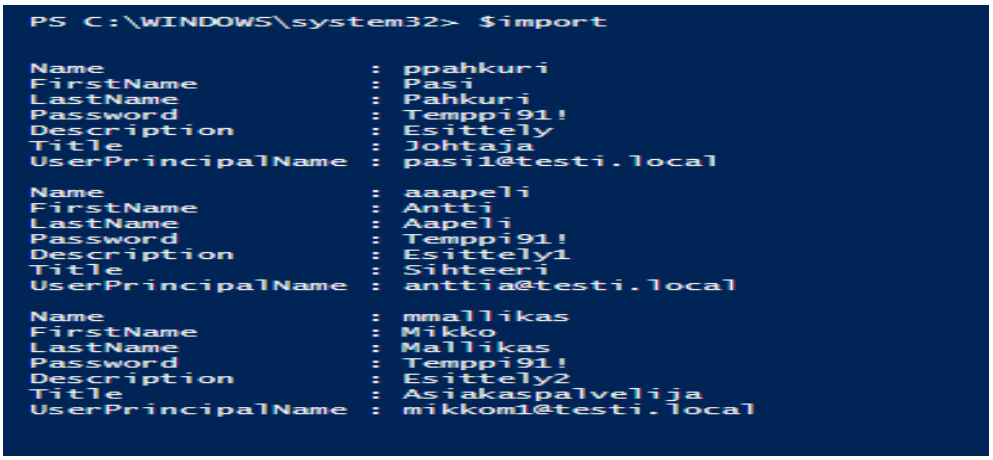


Kuva 10. CSV-tiedosto jonka pohjalta luodaan tilit.

Kyseisen tiedoston tiedot tuodaan seuraavaksi PowerShelliin ja tallennetaan tiedot muuttujaan nimeltä Import. Tiedot tuodaan PowerShellin komennolla:

$import = Import-Csv -Path ”c:\skriptit\tilit.csv”

Kyseisen tiedoston tiedot on tallennettu taulukkoon, jonka tiedot voi tarkistaa syöttämällä muuttujan eli tässä tapauksessa $import.



Kuva 11. CSV-tiedostosta tuotu data.

Tässä tapauksessa kaikille luotaville käyttäjille tulee olemaan sama organisaatioyksikkö aktiivihakemistossa. Joten organisaatioyksiköntiedot tallennetaan muuttujaan nimeltä OU. Organisaatioyksikkö on aiemmin luotu TestiOU ja toimialue testi.

$OU = "OU=TestiOU ,DC=testi".

Seuraavaksi luetaan muuttujasta tiedot käyttäen toistorakennetta eli tässä tapauksessa foreachia. Toistorakenteessa käydään läpi kaikki muuttujan tiedot ja niiden perusteella luodaan aktiivihakemistoon uusi käyttäjä käyttäen cmdlets-komentoa New-ADUser. Muuttujaan hyödyntämällä sille saadaan tarvittavat parametrit. Salasana muutetaan salattuun muotoon. Tietojen lisäksi määritellään käyttäjätileihin pakotettu salasananvaihto ensimmäisellä kirjautumiskerralla.
Toistorakenne menee kokonaisuudessaan:

```
foreach ($tili in $import) {
$password =$tili.password | ConvertTo-SecureString -AsPlainText -Force
New-ADUser -Name $tili.name -GivenName $tili.firstname -Surname
$tili.lastname -path $OU -Account Password $password -OtherAttributes
@{‘Description’=$tili.Description;’title’=$tili.title; ’ UserPrincipal-
Name’=$tili.UserPrincipalName} -ChangePasswordAtLogon $true -Ena-
bled $true
}
```

Lopuksi luodaan PowerShell skriptitiedosto nimeltä tilit.ps1 ja tallennetaan siihen kaikki yllä olevat komennot. Jatkossa käyttäjätilien luominen onnistuu ajamalla kyseinen skriptitiedosto PowerShellissä.

Käyttäjätili voidaan luoda PowerShellissä myöskin käyttäjän syötteiden perusteella, jolloin mitään tietoa ei tuoda erillisestä tiedostosta. Syötetyt tiedot tallennettaan muuttujaan ja sen jälkeen käytetään New-ADUser komentoa tilin luontiin. Tässä menetelmässä ei käytetä toistorakennetta.

Alla olevassa komennossa pyydetään käyttäjänsyöte ja tallennetaan se muuttujaan:
$Etunimi = Read-Host -Prompt 'Input the user name'
$Sukunimi = Read-Host -Prompt 'Input the user name'

Vastaavalla tavalla pyydetään ja tallennetaan muuttujiin kaikki halutut tiedot. Tämän jälkeen luodaan haluttu käyttäjätili aktiivihakemistoon.

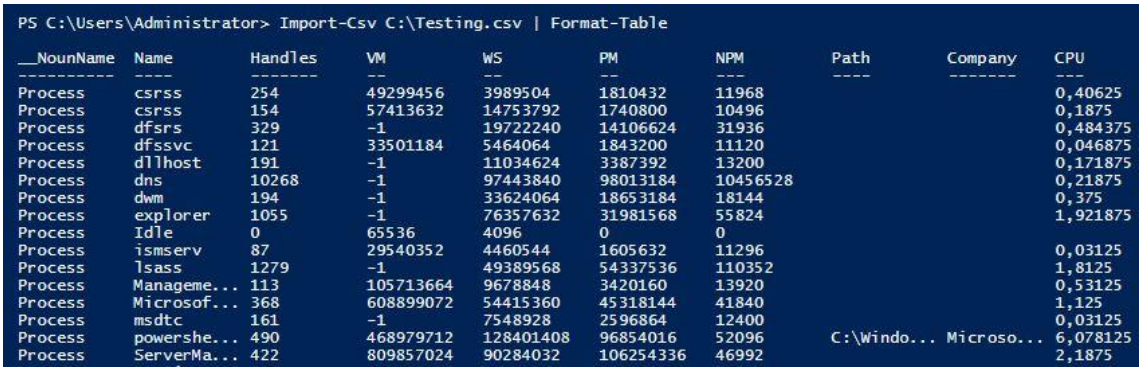
5.6 **Tietojen hakeminen ja tulostus**

Tässä kappaleessa käydään läpi eri tietojen hakemista ja tulostamista. Sen lisäksi niiden tietojen tallennus esimerkiksi CSV-tiedostoksi. Tämä helpottaa hallintaa, jos pitää esimerkiksi hakea useiden käyttäjien tiedot kerralla, niin on helpompi tallentaa ne suoraan tekstitiedostoon. Sama myös toisin päin eli miten lukea esimerkiksi tekstitiedostosta käyttäjien nimet ja luoda niistä käyttäjätilejä. Kaikkea tietoa ei saa graafisesta käyttöliittymästä ulos, joten ne pitää hakea PowerShellin avulla.

Tietoja on mahdollista tuoda ja viedä PowerShellissa useilla eri komennoilla sekä eri tiedostotyypeiksi. CSV-tiedostosta tietojen tuonti ja vienti onnistuvat Import-CSV ja Export-CSV Komennoilla. Sen lisäksi komentoon määritellään tiedoston sijainti komennolla Path. Tässä testatussa komennossa haetaan kaikki prosessit ja ne tallentuvat Testing.csv tiedostoon, komento on:

Get-Process | Export-Csv -Path C:\Testing.csv

Import-CSV komennolla voidaan näyttää tiedoston tiedot PowerShellissä, tässä tapauksessa juuri luodun tiedoston tiedot saadaan näkyviin komennolla Import-CSV C:Testing.csv. Tulostettua tietoa saadaan helposti muotoiltua komennolla Format, tieto saadaan luettavampaan muotoon Format-Table komennolla. Lista muotoon tiedot saa Format-List komennolla.



```
PS C:\Users\Administrator> Import-Csv C:\Testing.csv | Format-Table

__NounName  Name        Handles     VM          WS          PM          NPM         Path        Company     CPU
----------  ----        -------     --          --          --          ---         ----        -------     ---
Process     csrss       254         49299456    3989504     1810432     11968                               0,40625
Process     csrss       154         57413632    14753792    1740800     10496                               0,1875
Process     dfsrs       329         -1          19722240    14106624    31936                               0,484375
Process     dfssvc      121         33501184    5464064     1843200     11120                               0,046875
Process     dllhost     191         -1          11034624    3387392     13200                               0,171875
Process     dns         10268       -1          97443840    98013184    10456528                            0,21875
Process     dwm         194         -1          33624064    18653184    18144                               0,375
Process     explorer    1055        -1          76357632    31981568    55824                               1,921875
Process     Idle        0           65536       4096        0           0
Process     ismserv     87          29540352    4460544     1605632     11296                               0,03125
Process     lsass       1279        -1          49389568    54337536    110352                              1,8125
Process     Manageme... 113         105713664   9678848     3420160     13920                               0,53125
Process     Microsof... 368         608899072   54415360    45318144    41840                               1,125
Process     msdtc       161         -1          7548928     2596864     12400                               0,03125
Process     powershe... 490         468979712   128401408   96854016    52096       C:\Windo... Microso...  6,078125
Process     ServerMa... 422         809857024   90284032    106254336   46992                               2,1875
```

Kuva 12. Tiedot taulukko muodossa.

Näytettävän tulostuksen tietoja voi valita komennolla Property eli esimerkiksi -Property name, niin tulostuksessa näytetään vain prosessien nimet. Näin saadaan näytettävästä tiedosta luettavampaa ja vain ne tiedot jota halutaan, varsinkin mikäli tietoa on paljon.

CSV-tiedostot soveltuvat hyvin PowerShellin kanssa, sillä niitä voi muotoilla hyvin ja helposti, esimerkiksi aktiivihakemistoon käyttäjätilin tietojen lukeminen onnistuu parhaiten CSV-tiedostosta. Muitakin tiedostotyyppejä voi kyllä PowerShellin kanssa käyttää, tässä työssä käytiin käytännössä läpi tekstitiedostoon ja HTML-tiedostoon. Tekstitiedostoon tiedon vienti onnistuu komennolla Out-File, eli Out-File c:\testi.txt. HTML-tiedostoon tiedon vienti taas menee komennolla ConvertTo-Html ja perään halutti sijainti. Molempiin komentoihin aluksi syötetään tiedostoon vietävät tiedot, tässä vietiin työaseman prosessit Get-Process cmdletsillä.

Mikäli ajetaan ja tehdään monimutkaisempaa sekä pidempää skriptiä, niin silloin järkevää voi olla vielä tiedot muuttujaan. Tässä työssä testattiin tallentaa prosessit muuttujaan. Se tapahtuu ja onnistuu komennolla: Get-Process | Tee-Object -Variable Testi.

5.7 **Työaseman hallinta**

Tässä osiossa käydään läpi käytännössä työaseman hallintaa PowerShellillä. Miten voi tehdä erilaisia muutoksia työasemaan ja vaihtaa asetuksia sekä sen tietoja. Näiden lisäksi käydään läpi käytännössä yleistä hallintaa ja hyödyllisiä asioita, miten PowerShelliä voi käyttää hyödyksi. Alla olevia testauksia on kokeiltu suoraan paikallisesti palvelin työasemaan sekä PowerShell etäistunnon kautta toiseen työasemaan.

Työaseman nimen muuttaminen tapahtuu Cmdlets komentoa Rename-Computer käyttämällä. Kyseinen komento vaatii vähintään parametreiksi

työaseman uuden nimen sekä tunnuksen tiedot. Kokonaisuudessaan komento menee siis näin:

Rename-Computer -NewName "Testipalvelin" -DomainCredential testi.local\Administrator -Restart.

Komennon ajamisen jälkeen tulee erillinen kirjautumisikkuna, johon syötetään järjestelmänvalvojan salasana. Lopuksi kyseinen työasema pitää käynnistää uudelleen, jotta muutos tulee voimaan. Tässä komennossa se oli jo mukana, eli käynnistyy automaattisesti uudelleen eikä täydy tehdä manuaalista käynnistämistä. Komennon onnistumisen voi tarkistaa usealla eri tapaa, helpoiten nykyisen työaseman nimen saa selville komennolla hostname. Kyseisellä komennolla nähdään, että työaseman nimi on Testipalvelin.

Mikäli halutaan muuttaa jonkun toisen työaseman nimeä, niin sekin onnistuu Rename-Computer Cmdletsillä. Komentoon pitää parametreiksi lisätä komennolla ComputerName työaseman nykyinen nimi ja paikallisen käyttäjän tunnukset komennolla LocalCredential. Komento kokonaisuudessaan näin:

Rename-Computer -ComputerName "Kone1" -NewName "Kone2" -LocalCredential Kone1\ Administrator -DomainCredential testi.local\ Administrator -Restart.

Seuraavaksi työaseman lisääminen toimialueeseen, se tapahtuu käyttämällä komentoa Add-Computer. Paikallisen työaseman lisääminen ei vaadi parametriksi kuin toimialueen nimen joka annettaan parametrillä DomainName. Toimialueeseen liittäminen tapahtuu siis alla olevalla komennolla. Toimialueeseen liittäminen viimeistellään työaseman uudelleenkäynnistyksellä.

Komento on:

Add-Computer -DomainName "testi.local" -Restart.

PowerShellillä voi helposti konfiguroida työaseman verkkoasetuksia. Työaseman nykyiset nähdään komennolla Get-NetIPConfiguration, kyseinen komento vastaa komentokehotteessa komentoa ipconfig. Verkkoyhteyden katkaisu ja päälle laittaminen onnistuvat komennoilla: Disable-NetAdapter ja Enable-NetAdapter.

```
PS C:\Windows\system32> Get-NetIPConfiguration


InterfaceAlias        : Ethernet0 2
InterfaceIndex        : 12
InterfaceDescription  : vmxnet3 Ethernet Adapter
NetProfile.Name       : testi.local
IPv4Address           : 172.18.170.172
IPv6DefaultGateway    :
IPv4DefaultGateway    : 172.18.168.1
DNSServer             : ::1
                        127.0.0.1
```

Kuva 13. Verkkoasetukset.

Kuvassa 13 näkyy tämän hetkiset verkkoasetukset. Aluksi uudelleenimetään verkkoyhteys, joka tällä hetkellä on nimeltään Ethernet0 2. Se onnistuu komennolla Get-NetAdapter ja siihen parametriksi nykyinen nimi parametrilla Name. Komennon perään putkitetaan uudelleennimeäminen komennolla Rename-NetAdapter ja siihen parametriksi haluttu uusi nimi parametrillä NewName. Verkkohteyden Ethernet0 2 nimeäminen Verkko nimiseksi menee kokonaisuudessaan:

Get-NetAdapter -Name “Ethernet0 2” | Rename-NetAdapter -NewName Verkko.

Seuraavaksi asetetaan työasemaan ip-osoitetiedot ja nimipalvelin eli dns-osoite. Ne voidaan asetettaa komennoilla New-NetIPAddress ja Set-DNSClientServerAddress. Komennot löytyvät kokonaisuudessaan alla olevista kuvista.

```
PS C:\Windows\system32> New-NetIPAddress -InterfaceIndex 12 -IPAddress 192.168.0.14 -PrefixLength 24 -DefaultGateway 192.168.0.1
```

Kuva 14. Verkkoasetuksien asettaminen PowerShellillä.

Komennossa määritellään uusi IP-osoite, sekä haluttu verkkoadapteri komennolla IntefaceIndex. Verkkoasetuksiin määritellään ip-osoitteen lisäksi oletusyhdyskäytävä sekä aliverkonpeite. Näiden asetuksien lisäksi määritetään halutut nimipalvelintiedot komennolla Set-DNSClientServerAddress.

Lopuksi muutettujen tietojen läpi menon voi tarkistaa aiemmin mainituilla komennoilla Get-NetIPConfiguration jolla saadaan kaikki verkkoasetukset näkyville. Verkkoadapterien tarkemmat tiedot saa näkyville Get-NetAdapter. Näin nähdään, että nyt palvelimen verkkoasetukset ovat halutunlaiset. Toisen toimialuessa olevan työasemen verkkoasetuksia voidaan muuttaa PowerShellillä ottamalla etäistunto kyseiseen työasemaan

tai vaihtoehtoisesti ajaa komennot yksittäin käyttämällä Cmdletsiä Invoke-Command.

5.8 **O365 käyttöönotto**

Työtä varten otettiin käyttöön kokeiluversio Office 365:stä. Tämän saa veloituksetta käyttöön Microsoftin sivuilta. Alustana käytettiin tämän lisäksi työpaikan Office 365 -ympäristöä. Versio joka otettiin käyttöön, oli Office 365 Business. Käyttöönotto on hyvin yksinkertainen prosessi, syötetään organisaation tiedot. Sen lisäksi valitaan käytettävä toimialue, johon asetettiin pptesti.onmicrosoft.com ja samalla luotiin testikäyttäjä hallintaa varten pasitesti.

Tämän jälkeen on käytössä O365 alusta sekä hallintaportaali, josta pääsee hallinnoimaan tätä. Tässä työssä kuitenkin on tarkoitus käyttää PowerShelliä, joten hallinnointi tehdään sitä kautta. Aktiivihakemiston ja Office 365:n välille on mahdollista tehdä integraatio, eli käyttäjätilit olisivat toisiinsa yhteydessä, kuten yleensä on esimerkiksi työmaailmassa todellisissa ympäristöissä. Tässä työssä ei kuitenkaan käsitellä tätä tarkemmin eikä testattu asiaa.

Seuraavaksi otetaan käyttöön vielä tarvittava moduuli PowerShellissä. Office 365 -ympäristöön ja pilvipuoleen eli Azure AD:hen yhdistämistä varten pitää ottaa käyttöön moduuli nimeltä MSOnline. Tämän moduulin kautta saadaan käyttöön tarvittavat cmdlets-komennot ja voidaan yhdistää sekä hallinnoida Powershellin avulla Office 365-ympäristöä. Ennen moduulin käyttöönottoa asennettaan Microsoftin sivuilta löytyvä paketti nimeltä Microsoft Online Services Sign-In Assistant for IT Professionals RTW. Paketti tuo Office 365 ja Azure AD PowerShell hallintaan tarvittavia toiminnallisuuksia. Moduuli asennetaan ja otetaan käyttöön komennolla Install-Module MSOnline. Toinen asennettava ja tarvittava moduuli on AzureAD

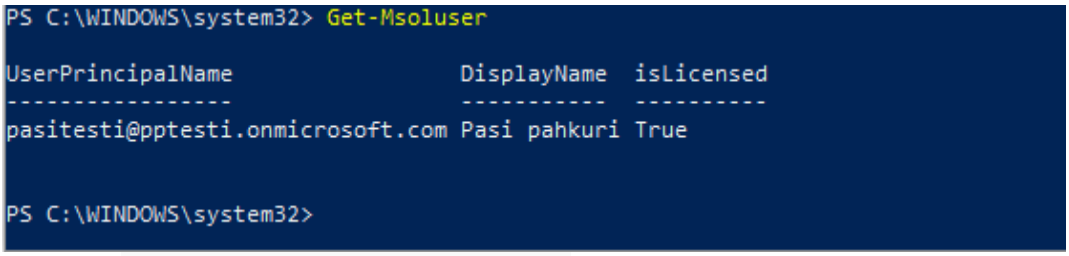
Moduulit pitää ottaa asennuksen jälkeen vielä käyttöön komennolla Import. Moduulin käyttöönoton jälkeen yhdistetään PowerShellillä Office 365 -ympäristöön, jotta hallinnointi onnistuu. Yhdistäminen onnistuu komennolla:

```
$UserCredential = Get-Credential
Connect-MsolService -Credential $UserCredential
```

Yhdistämisen jälkeen aukeaa käyttätunnus ja salasana kysely popup-ikkuna. Tähän syötetään Office 365 -ympäristöä luodessa tehden käyttäjätilin tiedot. Yhdistäminen on mahdollista tehdä myös käyttäen monitasoista tunnistautumista, silloin yhdistetään vain komennolla:

Connect-MsolService

Yhdistämisen onnistumisen voi tarkistaa esimerkiksi komennolla Get-MsolUser, joka näyttää tuloksena ympäristön käyttäjätilit.



Kuva 15. Get-MsolUser komennon tulokset.

Viimeisenä valmisteluna määritellään vielä käytäntö sille mitä skriptejä voidaan ajaa. Skriptien ajamista voidaan rajoittaa, esimerkiksi niin, että vain paikallisesti luodut skriptit voidaan ajaa. Tämä on yksi keino joka lisää tietoturvaa, eli oletuksena ei ajeta netistä ladattuja skriptejä. Otetaan käyttöön käytäntö nimeltä RemoteSigned. Tässä käytännössä voidaan ajaa paikallisesti luodut skriptit rajoituksetta sekä netistä ladatut skriptit jotka on allekirjoitettu ja ladattu turvalliseksi määritellyiltä julkaisijoilta. Tässä työssä suoritetaan vain itse tehtyjä skriptejä, joten sopii tähän tarpeeseen hyvin.

Käytäntö otetaan käyttöön komennolla Set-ExecutionPolicy, komento kokonaisuudessaan:
Set-ExecutionPolicy RemoteSigned

Yllä olevat käyttöönoton toimenpiteet ja yhdistämiset pitää tehdä joka kerta, kun hallinnoidaan PowerShellin avulla O365-ympäristiä. Asiat voidaan kuitenkin lisätä profiiliin, joten luodaan uusi PowerShell-profiili tai voidaan muokata aiemmin luotua profiilia. Profiili löytyy työasemalta PowerShell kansiopolun alta. Sen muokkaus onnistuu aiemmin näytetyllä tavalla tai voidaan käyttää alla olevaa tapaa, jolla saadaan auki nykyisen käyttäjän profiili.

Profiilin muokkaus:
psEdit $PROFILE

Profiiliin lisätään tarvittavat moduulit ja yhdistäminen O365-ympäristöön. Profiilin muokkauksen jälkeen avataan PowerShell ISE-editori järjestelmänvalvojan, jotta kaikki tehtävät muutokset menevät läpi.

5.9 **Käyttäjätilin hallinta O365:ssä**

Tässä kappaleessa käydään läpi käytännössä O365-ympäristön hallintaa PowerShellin avulla. PowerShellin avulla luodaan käyttäjätili ja muokataan sen tietoja. Niiden lisäksi haetaan tietoa ympäristöstä PowerShellin avulla. Kaikkea tietoa ei saa ulos graafisesta käyttöliittymästä, joten PowerShell on näissä tilanteissa erittäin hyödyllinen. Kaikkia muutoksia ei myöskään pysty tekemään normaalista O365-hallintaportaalista.

Ensiksi luodaan ympäristöön käyttäjätili cmdlets-komennolla New-MsolUser. Käyttäjätilille annetaan parametreiksi perustiedot, nimi, kirjautumistunnus, lokaatio sekä lisenssi. Käytössä olevat lisenssityypit ja määrät näkevät cmdlets-komennolla Get-MsolAccountSku. Sen kautta nähdään, että lisenssejä on vapanaa 25 ja tyyppi on O365_Business.

Kokonaisuudessaan käyttäjätilin luonti menee näin:

New-MsolUser -DisplayName "Meikäläinen Matti" -FirstName Matti -LastName Meikälainen -UserPrincipalName mattim@pptesti.onmicrosoft.com -UsageLocation FI -LicenseAssignment pptesti:O365_BUSINESS

Lisenssi voidaan lisätä käyttäjätiliin myös jälkikäteen, jos sitä ei laiteta käyttäjätiliä luodessa. Lisenssi lisääminen tapahtuu käyttämällä cmdlets-komentoa Set-MsolUserLicense.

Komento kokonaisuudessaan:

Set-MsolUserLicense -UserPrincipalName pasip@pptesti.onmicrosoft.com -AddLicenses pptesti: O365_BUSINESS

Suuria määriä käyttäjiä on kuitenkin hidas luoda yksitellen, joten seuraavaksi kokeillaan lukea tiedot Excelillä luodusta CSV-tiedostosta, jossa on useampien käyttäjien tiedot. CSV-tiedoston voi helposti luoda Excelin lisäksi myös notepadilla, koska siihen syötetään vain otsikot ja halutut käyttäjän tiedot tekstimuodossa. Luodun Excel-tiedoston tieto on helppo lukea komennolla Import-CSV ja tähän syötetään vain parametriksi sijainti mistä tiedosto löytyy. Sen jälkeen laitetaan vielä komentoon toistorakenne, jotta saadaan kaikki tunnukset kerralla ajettua. Alla olevaa toistorakennetta voidaan soveltaa muissakin luonneissa, kuten käyttäjien lisäämisessä jakelulistoihin.

```
$tunnukset = Import-Csv C:\Users\Pasi\Desktop\Tilit.CSV

$tunnukset | ForEach-Object {
```

New-MsolUser -DisplayName $_.Nayttonimi - LastName $_.Sukunimi -
FirstName $_.Etunimi -UserPrincipalName $_.UPN }

Kyseinen skripti tallennetaan skripti muotoon, eli .ps1 tiedostotyyppinä. Tässä sille annettiin nimeksi luotili.ps1. Jatkossa voi siis ajaa kyseisen skriptin ja luoda käyttäjät, joten ei täydy kirjoittaa koko komentoa joka kerta.

Ylläolevaan skriptiin voidaan lisätä samaan tapaan muitakin haluttuja tietoja, riippuen mitä käyttäjätileihin halutaan. Komennon onnistuminen voidaan tarkistaa usealla eri tapaa, esimerkiksi aiemmin mainitulla cmdlets-komennolla Get-MsolUser. Tuohon voidaan antaa parametriksi tarkenne SearchString ja hakea vaikkapa kaikki tilit joissa on nimi Matti. Sen kautta nähdään, että juuri luotu Matti Mallikkaan tili löytyy luotuna.

Komento menee kokonaisuudessaan:
Get-MsolUser -SearchString ”Matti”.

Seuraavaksi luodaan muutama jakelulista ja lisätään niihin muutama käyttäjä. Tämän jälkeen kokeillaan hakea PowerShellin avulla hakea tietyn käyttäjän jakelulistat.
Jakelulistan luonti onnistuu käyttämällä komentoa New-DistributionGroup. Siihen annettaan pakollisina parametreina nimi.

Kokonaisuudessaan jakelulistan luonti menee:
New-DistributionGroup -Name "Testilista1"

Parametriksi voidaan antaa pakollisen nimen lisäksi monia muitakin tietoja. Luodaan toinen jakelulista, johon annettaan vielä näyttönimi ja omistaja parametrina.

New-DistributionGroup -Name "Testilista2" -DisplayName "Testilista2" -
ManagedBy mattim

Luotuihin jakelulistoihin voidaan lisätä käyttäjiä komennolla Add-DistributinGroupMember. Jotta käyttäjä saadaan lisättyä samalla kertaa useampaan haluttuun jakelulistaan tai ryhmään, niin käytetään toistorakennetta. Jakelulistojen nimet tallennetaan muuttujaa nimeltä Listat.

Komento menee kokonaisuudessaan:
$Listat = "Testilista1","Testilista2”
$Listat | ForEach-Object {
Add-DistributionGroupMember -Identity $_ –Member "pasip" }

Graafiselta puolelta O365-hallintaa ei välttämättä saa auki helposti kaikkia tietoja, esimerkiksi usein on tarvetta tarkistaa mihin kaikkiin ryhmiin käyttäjä kuuluu ja mihin jaettuihin sähköpostilaatikoihin hänellä on oikeus. Alla olevassa komennossa tarkistetaan käyttäjältä mihin kaikkiin jakelulistoihin käyttäjä kuuluu, tässäkin hyödynnetään toistorakennetta. Aluksi tallennetaan käyttäjänimi Tili nimiseen muuttujaan ja sen avulla tarkistaan mihin jakelulistoihin käyttäjä kuuluu.

Käyttäjänjakelulistojen haku PowerShellillä:

```
$Tili = "pasip@pptesti.onmicrosoft.com"

$Listat= Get-DistributionGroup | where { (Get-DistributionGroupMember
$_.Nimi | foreach {$_.PrimarySmtpAddress}) -contains "$Tili"}
```

Käyttätilin ryhmäjäsenyydet saa näkyviin myöskin alla olevalla komennolla. Komennossa haetaan kaikki kyseisen käyttäjätilin ryhmät. Harjoitus mielessä testattiin tätäkin tapaa. Komennossa tallennetaan tulokset ja tiedot muuttujaan nimeltä ryhmat. Ryhmät haetaan käyttämällä cmdlets-komentoa get-group. Tuloksina ne joissa ne joissa on kyseinen käyttäjänimi. Tämänkin toteuttamiseen löytyy siis useita eri tapoja.

Komento menee kokonaisuudessaan:

```
$ryhmat = get-group -resultsize unlimited
$ryhmat | where{$_.members -like "*pasip"}
```

Yhtenä tärkeänä osa O365:ssa on jaetut sähköpostilaatikot. Joten seuraavaksi luodaan PowerShellillä jaettu sähköpostilaatikko ja lisätään siihen oikeudet käyttäjälle. Jaetun sähköpostilaatikon luonti onnistuu komennolla New-Mailbox.

Luodaan jaettu sähköpostilaatikko:

```
New-Mailbox -Name Testilaatikko1 -PrimarySmtpAddress
testijaettu@pptesti.onmicrosoft.com -Shared
```

Yllä olevassa komennossa luodaan jaettu sähköpostilaatikko ja annetaan sille haluttu primääriosoite. Komentoon määritetään perään parametri Shared, jonka kautta tiedetään, että luodaan jaettua sähköpostilaatikkoa eikä käyttäjälaatikkoa.

Kyseisessä komennossa ei vielä annettu oikeuksia siihen kenelläkään käyttäjälle. Oikeudet voidaan antaa käyttämällä cmdlets-komentoa Add-MailboxPermission.

Add-MailboxPermission “Testilaatikko1” -User pasitesti -AccessRights FullAccess -InheritanceType all

Tässä annetaan pasitesti nimisellä käyttäjätilille täydet oikeudet luotuun jaettuun sähköpostilaatikkoon nimeltä Testilaatikko1. Viimeisellä parametrillä määritetään, että käyttäjä saa oikeudet laatikon mahdollisiin alikansioihin. Komennon läpi meno voidaan tarkistaa käyttämällä komentoa Get-MailboxPermission. Tällä komennolla voi tarkistaa sähköpostilaatikon oikeudet, parametrina käytetään Identity ja siihen haluttu käyttäjätili.

Tähän asti työn käytännön osuudessa on pääasiassa ajettu ja tehty lyhyitä komennon pätkiä. Joten tässä viimeisessä käytännön osiossa luodaan laajempi skripti, jolla luodaan O365-ympäristöön jaettu sähköpostilaatikko. Skriptiin luodaan kyselyt niin, että käyttäjältä kysytään tarvittavat tiedot eli mikä laatikko luodaan.

Luodaan työasemalla Skriptin kansioon tiedosto joka tallennettaan .ps1 muodossa eli PowerShell skriptinä. Kyseinen tiedosto voidaan tällöin ajaa suoraan PowerShellin kautta.

Write-Host-komennolla tulostetaan tekstiä ruutuun ja ilmoitetaan tässä tapauksessa mitä ollaan tekemässä. Tämän avulla kuitataan lopuksi, että haluttu jaettu laatikko nyt luotu. Jaetun sähköpostilaatikon luotiin käytetään cmdlets-komentoa New-Mailbox. Siihen annettaan parametreiksi näyttönimi sekä sen osoite.

Kokonaisuudessaan jaetun sähköpostilaatikon skripti menee näin:

```
Write-Host "Tämä skripti luo uuden jaetun sähköpostilaatikon" -foregroundcolor "yellow"

$jaettu_nimi = Read-Host "Jaetun sähköpostilaatikon nimi (esim. jaettu@pptesti.onmicrosoft.com)"
 $Jaettu_display = Read-Host "Syötä jaetun sähköpostilaatikon näyttönimi (esim. Jaettuloota)"

Write-Host "Luodaan jaettusähköpostilaatikkoa..." -foregroundcolor "yellow"

New-Mailbox -Name $jaettu_nimi  -Shared -PrimarySMTPAddress $jaettu_nimi

Write-Host "Odota hetki, laatikkoa luodaan -foregroundcolor "yellow"
```

```
Write-Host "Jaettu sähköpostilaatikko on luotu: $jaettu_nimi  " -fo-
regroundcolor "green"
```

Yllä olevassa skriptissä luodaan pelkästään haluttu jaettusähköpostilaatikko eikä lisätä oikeuksia siihen kenellekään. Joten kysytään syötteellä haluttu käyttäjänimi ja tallennetaan se muuttujaan. Tämä jälkeen annettaan täydet oikeudet halutulle käyttäjällä luotuun jaettuun sähköpostilaatikkoon.

```
$tili_oikeudet = Read-Host "Anna tilin nimi jolle annetaan oikeudet”

Add-MailboxPermission $Jaettu_display -User $tili_oikeudet -Ac-
cessRights FullAccess
```

Yhdistetään kaikki ylläolevat samaan luotuun skriptitiedostoon, jolloin se ajamalla saadaan luotua syötteiden mukainen jaettu sähköpostilaatikko ja siihen halutulle käyttäjälle täydet oikeudet.

Testailun perusteella tämä luotu skipti ei ole kovinkaan käytännöllinen todellisessa käytössä, vaan toimivampi ratkaisu on tehdä luonti excel-tiedoston tai vastaavan pohjalta. Tätä voidaan kuitenkin soveltaa eri käyttötarkoituksiin.

6 YHTEENVETO

Työn tavoitteisiin ja opinnäytetyön tutkimiskysymyksiin saatiin kaikkiin vastaukset, joten niiden osalta työ oli onnistunut. Työssä esiteltiin kattavasti mikä on PowerShell ja miten sitä käytetään. Sen lisäksi käytännön osuudessa havaittiin, että PowerShelliä voi hyödyntää palvelin ja AD-ympäristössä erittäin laajasti.

Työssä käsiteltiin pääasiassa käyttöjärjestelmänä Windows Server 2012 palvelinympäristöä, joka näin jälkikäteen ajateltuna oli virhe. Olisi pitänyt valita työhön pelkästään Windows Server 2016 palvelinkäyttöjärjestelmä, niin olisi saanut enemmän uuttaa näkökulmaa aiheeseen. Lisäksi työn olisi pitänyt olla rajatumpi.

PowerShelliä ja sen ominaisuuksia kehitetään jatkuvasti. Sen rooli palvelimienhallinnassa ja erityisesti järjestelmäasiantuntijan hommissa tulee varmasti kasvamaan koko ajan. PowerShell tehostaa työskentelyä erittäin paljon ja erityisesti kaikenlaiset massa sekä rutiinityöt onnistuvat, sillä tehokkaasti. PowerShellin avulla voidaan automatisoida paljon eri asioita ja sen kautta yritykset voivat säästää kuluissa.

Työtä tehdessä kuitenkin opin erittäin paljon liittyen PowerShelliin ja sen käyttöön. Käytännön vaihe oli myös erittäin opettavainen ja näistä opeista tulee olemaan hyötyä työelämässä. PowerShell on kuitenkin työkalu, jota käytän lähes päivittäin nykyisessä työssä ja varmasti myös entistä laajemmin tulevissa työtehtävissäni. Tulen siis varmastikin jatkossa perehtymään ja syventymään aihealueeseen enemmän. Tämä työ oli siihen erittäin hyvä aloitus. Tärkeimpänä oppina varmasti tästä työstä tuli itselle, että tämän kaltaiset työt vaativat rajatumman aiheen sekä selkeämmän suunnitelman.

LÄHTEET

4sysops (2016). My favorite Windows PowerShell ISE add-ons. Haettu 15.3.2018 osoitteesta https://4sysops.com/archives/my-favorite-windows-powershell-ise-add-ons/

4sysops (n.d). Differences between PowerShell versions. Haettu 12.2.2018 osoitteesta https://4sysops.com/wiki/differences-between-powershell-versions/

Computerperformance (n.d.a). What's New In Version PowerShell 2.0. Haettu 18.2.2018 osoitteesta http://www.computerperformance.co.uk/powershell/index_v2.htm

Computerperformance (n.d.b). New Features in Windows PowerShell 3.0. Haettu 18.2.2018 osoitteesta http://www.computerperformance.co.uk/powershell/powershell3-whats-new.htm

Computerperformance (n.d.c). Windows PowerShell 4.0. Haettu 18.2.2018 osoitteesta http://www.computerperformance.co.uk/powershell/index_v4.htm

Driscoll, A. (2012). *Microsoft Windows PowerShell 3.0 First Look.* Birmingham: Pack Publishing Ltd.

Driscoll, A. (2018). PowerShell integration for Visual Studio 2015 and 2017. Haettu 18.3.2018 osoitteesta https://github.com/adamdriscoll/poshtools

Jares, J. (2014). Get Started with Pester. Haettu 15.3.2018 osoitteesta http://www.powershellmagazine.com/2014/03/12/get-started-with-pester-powershell-unit-testing-framework/

Lee, T., Mitschke, K., Schill, M. & Tanasovski T. (2011). *Windows Powershell 2.0 Bible*. Indianapolis: John Wiley & Sons.

Microsoft (2014). Windows Server 2012 R2 Products and Editions Comparison. Haettu 14.3.2018 osoitteesta https://www.microsoft.com/en-us/download/details.aspx?id=41703&WT.mc_id=rss_alldownloads_all

Microsoft (2017). Desired State Configuration Quick Start. Haettu 15.3.2018 osoitteesta https://docs.microsoft.com/en-us/powershell/dsc/quickstart

Microsoft (2018a). Getting Started with Microsoft PowerShell. Haettu 18.2.2018 osoitteesta https://mva.microsoft.com/en-US/training-courses/getting-started-with-microsoft-powershell-8276?l=r54IrOWy_2304984382

Microsoft (2018b). Cmdlet Overview. Haettu 15.3.2018 osoitteesta https://msdn.microsoft.com/en-us/library/ms714395(v=vs.85).aspx

Microsoft (2018c). Sep up directory synchronization for Office 365. Haettu 7.4.2018 osoitteesta https://support.office.com/en-us/article/set-up-directory-synchronization-for-office-365-1b3b5318-6977-42ed-b5c7-96fa74b08846

Microsoft (n.d.a). PowerShell. Overview. Haettu 11.2.2018 osoitteesta https://docs.microsoft.com/fi-fi/powershell/scripting/powershell-scripting?view=powershell-6

Microsoft (n.d.c). What's New in Windows PowerShell 5.0. Haettu 18.2.2018 osoitteesta https://docs.microsoft.com/en-us/powershell/scripting/whats-new/what-s-new-in-windows-powershell-50?view=powershell-6

Microsoft (n.d.d). What's New in PowerShell Core 6.0. Haettu 18.2.2018 osoitteesta https://docs.microsoft.com/en-us/powershell/scripting/whats-new/what-s-new-in-powershell-core-60?view=powershell-6

Microsoft (n.d.e). Windows PowerShell Integrated Scripting Environment (ISE). Haettu 11.2.2018 osoitteesta https://docs.microsoft.com/fi-fi/powershell/scripting/getting-started/fundamental/windows-powershell-integrated-scripting-environment--ise-?view=powershell-6

Microsoft (n.d.f). Windows Management Framework. Haettu 15.3.2018 osoitteesta https://docs.microsoft.com/en-us/powershell/wmf/readme

Microsoft (n.d.g). Overview of Cmdlets Available in Windows PowerShell. Haettu 18.3.2018 osoitteesta https://technet.microsoft.com/en-us/library/ff714569.aspx

Microsoft (n.d.h). Office 365 for business. Haettu 7.4.2018 osoitteesta https://products.office.com/en-us/business/microsoft-office-365-frequently-asked-questions

Sapien Technologies, Inc (2018). PowerShell Studios 2018. Haettu 18.3.2018 osoitteesta https://www.sapien.com/software/powershell_studio

Tampkins, L. (2015). 20 Years of Windows Server Product History. Haettu 14.3.2018 osoitteesta http://www.veritlabs.com/20-years-of-windows-server-product-history/

TechRepublic (2017). Windows Server 2016: The smart person's guide. Haettu 25.3.2018 osoitteesta https://www.techrepublic.com/article/windows-server-2016-the-smart-persons-guide/

TechTerms (2017). Active Directory. Haettu 3.4.2018 osoitteesta https://techterms.com/definition/active_directory

Thomas Krenn (2017). Windows Server 2012 Editions Comparison. Haettu 14.3.2018 osoitteesta https://www.thomas-krenn.com/en/wiki/Windows_Server_2012_Editions_comparison