Marko Kautiainen

IT-JÄRJESTELMIEN TAUSTAPROSESSIEN AUTOMATISOINTI

Opinnäytetyö

Liiketalouden ammattikorkeakoulututkinto

Tietojenkäsittely

2021

Kaakkois-Suomen ammattikorkeakoulu

Kaakkois-Suomen ammattikorkeakoulu

Tutkintonimike	Tradenomi (AMK)
Tekijä/Tekijät	Marko Kautiainen
Työn nimi	IT-järjestelmien taustaprosessien automatisointi
Toimeksiantaja	Jyväskylän ammattikorkeakoulu, ICT-Palvelut
Vuosi	2021
Sivut	63 sivua, liitteitä 2 sivua
Työn ohjaaja(t)	Heli Manninen

TIIVISTELMÄ

Tässä opinnäytetyössä kuvataan toimeksiantajan tietojärjestelmäinfrastruktuurin selkärankana toimivat järjestelmät ja kartoitetaan menetelmiä niiden välillä tarvittavien automatisointien toteuttamiseksi.

Useat järjestelmät edellyttivät taustalla tapahtuvaa tiedonvaihtoa muiden järjestelmien kanssa. Osa tietolähteistä/kohteista oli sisäisiä ja osa ulkoisia. Osassa oli käytettävissä valmiita rajapintoja ja osassa ei. Kartoituksen jälkeen automatisointeja toteutettiin tärkeysjärjestyksessä niin kattavasti kuin se oli tässä yhteydessä mahdollista.

Toteutuksessa on huomioitu automaatioiden yhteensopivuus järjestelmien välillä, niiden ajastukset, tietoturva, virheentarkistukset, varmuuskopioinnit, dokumentointi ja tuotettujen toimintojen ylläpito jatkossa.

Toteutus onnistui hyvin ja erilaisia automatisointeja tuotettiin kartoituksen jälkeen useisiin järjestelmiin. Pääasiassa käytettiin Microsoft PowerShelliä, jolla toteutettuja skriptejä ajetaan eri palvelimilla. Työssä hyödynnettiin myös Power Automatea. Automaatiot toteuttavat joko ajastetusti tai reaaliaikaisia hakemistopalvelutapahtumia monitoroiden tietojärjestelmien vaatimia toimenpiteitä, mm. tiedostojen siirtoja ja varmuuskopiointeja, Active Directory (AD) -objektien ja niiden attribuuttien hallintaa sekä Azure-objektien hallintaa paikallisen AD-ryhmän perusteella. Osa skripteistä jätettiin manuaalisesti käynnistettäväksi.

Toteutetut automatisoinnit nopeuttavat prosessien toimintaa, vähentävät manuaalista työtä ja inhimillisten virheiden mahdollisuutta sekä parantavat tietoturvaa. Yleisesti ottaen taustalla tapahtuvat automatisoinnit parantavat henkilöstön ja järjestelmien kokonaistehokkuutta.

Asiasanat: automatisointi, PowerShell, Power Automate, ohjelmistorobotiikka, skriptaus, ohjelmointi, tietojenkäsittely

South-Eastern Finland University of Applied Sciences

Degree	Bachelor of Business Administration
Author (authors)	Marko Kautiainen
Thesis title	Automating ICT system background processes
Commissioned by	Jyväskylä University of Applied Sciences, ICT services
Time	2021
Pages	63 pages, 2 pages of appendices
Supervisor	Heli Manninen

ABSTRACT

The objective of this thesis was to describe the systems that serve as the backbone of the client's information system infrastructure and to find methods to implement necessary automations between them. Many systems under study required background information exchange with other systems. Some data sources were internal and some external. Some had built-in application protocol interfaces, and some did not. After the survey phase, the automations were carried out in the order of priority as comprehensively as possible in this context.

The implementation took into account the compatibility of automations between the systems, their timings, information security, error checking, backups, documentation and the future maintenance of automations. Implementation was successful and various automations were produced for several systems. Microsoft PowerShell was the main tool used to run scripts on different servers. Power Automate was also used. Automations carried out functions that were either timed or launched automatically by monitoring real-time directory service events. Automated tasks included, for example, file transfers and backups, managing Active Directory (AD) objects with their attributes, and managing Azure AD objects based on the local AD group. Some of the scripts were intentionally left to be started manually.

The results of this thesis provided new functionality to the client's infrastructure. The automations speed up the operation of processes, reduce manual work, reduce the possibility of human error, and improve data security. In general, underlying automations improve the overall efficiency of staff and IT systems.

Keywords: automation, PowerShell, Power Automate, robotic process automation, scripting, programming, information technology

SISÄLLYS

1 JOHDANTO

Työn otsikkona oleva taustaprosessien automatisointi on laaja ja hieman hankalasti rajattava käsite. Tässä yhteydessä taustaprosesseilla kuitenkin tarkoitetaan niitä toimintoja, joita eri tietojärjestelmät toteuttavat taustalla halutun lopputuloksen saavuttamiseksi omatoimisesti, tai yhdessä muiden tietojärjestelmien kanssa. Erilaiset automatisointitarpeet vaihtelevat organisaatioittain ja järjestelmittäin, mutta useimmissa organisaatiossa automatisaatioita toteutetaan jollakin menetelmällä, tai ainakin sitä on suunniteltu. Tällaisten prosessien suunnittelu, kehitys ja hallinta kuuluu yhtenä osana työhöni, ja se sisältää monia suoria yhteyksiä opinnäytetyön myötä päättyvään tietojenkäsittelyn koulutukseen. Tältä pohjalta ajateltuna oli luontevaa valita se myös opinnäytetyön aiheeksi. Minulla on entuudestaan pitkä työkokemus alalta, mutta halusin syventää osaamistani erityisesti ohjelmoinnin osalta. Työn toteutuksen aikana pääsin hyödyntämään kattavasti aiemman osaamisen lisäksi koulussa oppimaani tietoa, sekä myös soveltamaan uutta.

Opinnäytetyön toimeksiantajana on Jyväskylän ammattikorkeakoulun ICT-Palvelut, ja tarkoituksena on kartoittaa toimeksiantajan tietojärjestelmien taustaprosesseissa sellaisia tarpeita, jotka on järkevää ja mahdollista automatisoida. Kartoituksen jälkeen selvitetään eri menetelmiä automatisoinnille. Tavoitteena on toteuttaa havaituista tarpeista olennaisimmat kuhunkin prosessiin tarkoituksenmukaisimmaksi todettavalla menetelmällä.

Automatisoinnin tarve toimeksiantajalla muodostuu siitä, että useat käytetyt tietojärjestelmät edellyttävät taustalla tapahtuvaa tiedonvaihtoa muiden järjestelmien kanssa. Osa tietolähteistä/kohteista on sisäisiä ja osa ulkoisia. Osassa on käytettävissä valmiita rajapintoja näiden toteuttamiseksi ja osassa ei. Toteutettavien automatisointien on tarkoitus nopeuttaa prosessien toimintaa, vähentää manuaalista työtä ja virheiden määrää sekä parantaa sitä kautta tietoturvaa. Yleisesti ottaen taustalla ilman ihmisen toimia tapahtuvat automatisoinnit parantavat henkilöstön ja järjestelmien kokonaistehokkuutta. Työssä ei käsitellä loppukäyttäjän sovellusten käyttöä automatisoivaa ohjelmistorobotiikkaa kuin pintapuolisesti, vaan keskitytään nimenomaan taustalla tapahtuviin toimintoihin.

Toteutuksessa tulee huomioida automaatioiden yhteensopivuus, ajastukset, tietoturva, virheentarkistukset, varmuuskopioinnit ja skriptien ylläpito jatkossa. Suurin osa huomioitavista seikoista on sellaisia, joita ei ole voinut oppia koulussa, eli toteutus vaatii kokemuksen hyödyntämisen lisäksi monin paikoin myös uuden opettelemista. Työn toteutuksessa hyödynnän koulussa omaksuttujen taitojen lisäksi pitkää alan työkokemusta. Aihe on monimuotoisuudessaan haastava, mutta realistinen.

Opinnäytetyön luvussa 2 käyn läpi järjestelmäprosessien automatisointia ja kokonaisuuteen liittyviä osa-alueita yleisesti. Luvussa 3 esittelen toimeksiantajan toimintaympäristön ja listaan toteutettavaksi tunnistettuja tarpeita. Siinä myös luokitellaan toteutettavat automaatiot, jonka jälkeen luvussa 4 siirryn itse toteutuksen kuvaamiseen. Viimeisessä luvussa pohdin toteutusta ja sen lopputuloksia, sekä esitän muutamia kehitysehdotuksia jatkoa ajatellen. Osa automatisoinneissa käytetyistä ohjelmakoodeista on mukana erillisinä liitteinä.

2 JÄRJESTELMÄPROSESSIEN AUTOMATISOINTI

Tässä luvussa on kuvattu automatisoinnin historian lisäksi taustalla toimivien, muiden sovellusten ja näiden tunnus/pääsynhallinnan mahdollistavien tietojärjestelmien toimintaa, sekä myös itse automatisointiin liittyviä käsitteitä.

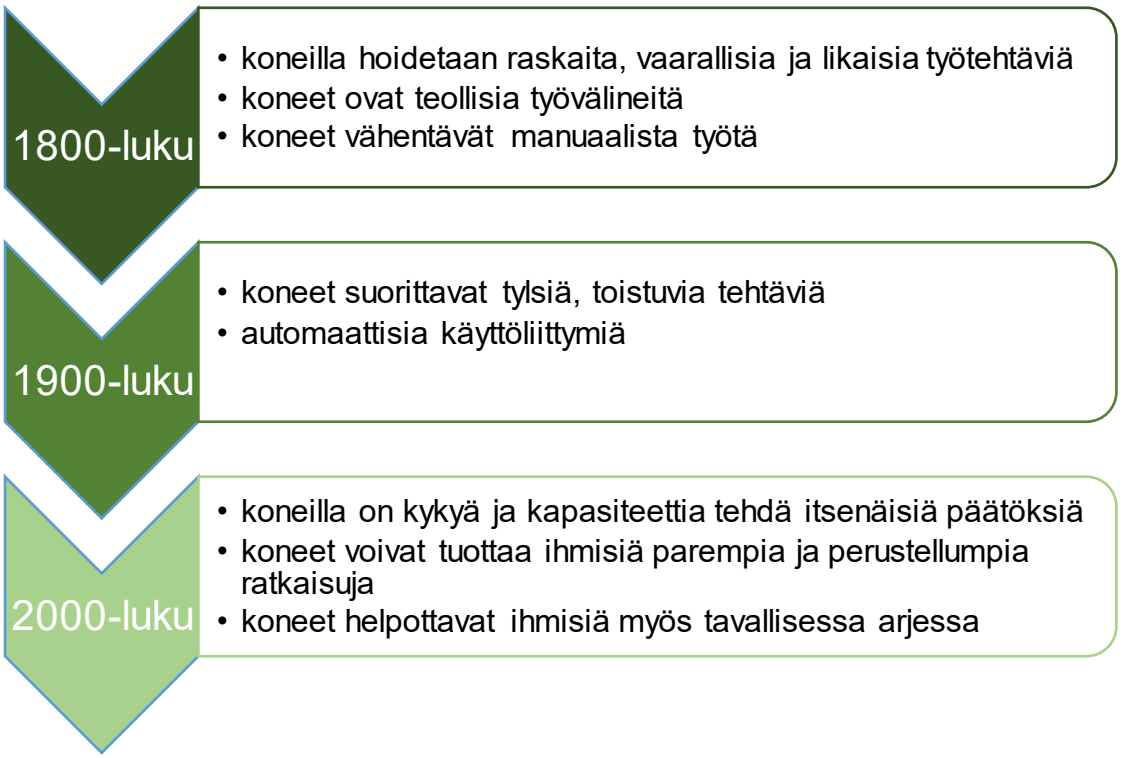
2.1 Automatisoinnin historia

Automatisointi itsessään on jo vanha keksintö. Yritykset ja organisaatiot ovat pyrkineet automatisoimaan toistuvia toimintoja ja vapauttamaan henkilöstöä tuottavampiin tehtäviin jo 1700-luvun teollisesta vallankumouksesta lähtien. Ensin tämä koski teollisuuslaitteita, ja sittemmin se on siirtynyt monilta osin yritysten IT-yksiköiden vastuulle. Vaikka automatisoinnin edut ovat kiistattomat, on silti hieman erikoista, kuinka monissa yrityksissä automatisaatiota ei edelleenkään hyödynnetä läheskään kaikissa toiminnoissa, jos lainkaan. (Mann 2018.)

Automatisoinnin hyötyjä ovat suoritusnopeuden kasvu, koneiden väsymättömyys, kulujen vähentäminen, parempi käyttökokemus loppukäyttäjälle, inhimillisten virheiden vähentäminen, joustavuus, muokattavuus ja uusien ominaisuuksien lisääminen. Automatisointia ei ole pakko nähdä aina jonkin vanhan

korvaajana, vaan sen avulla voidaan tuottaa jopa kokonaan uusia toimintamalleja ja työskentelytapoja. (Mann 2018.)

Davenportin ja Kirbyn mukaan (2015) automaation käyttöönotto yrityksissä jakautuu kolmeen aikakauteen, jotka on esitetty kuvassa 1.



Kuva 1. Automaation kolme aikakautta (mukaillen Davenport & Kirby 2015)

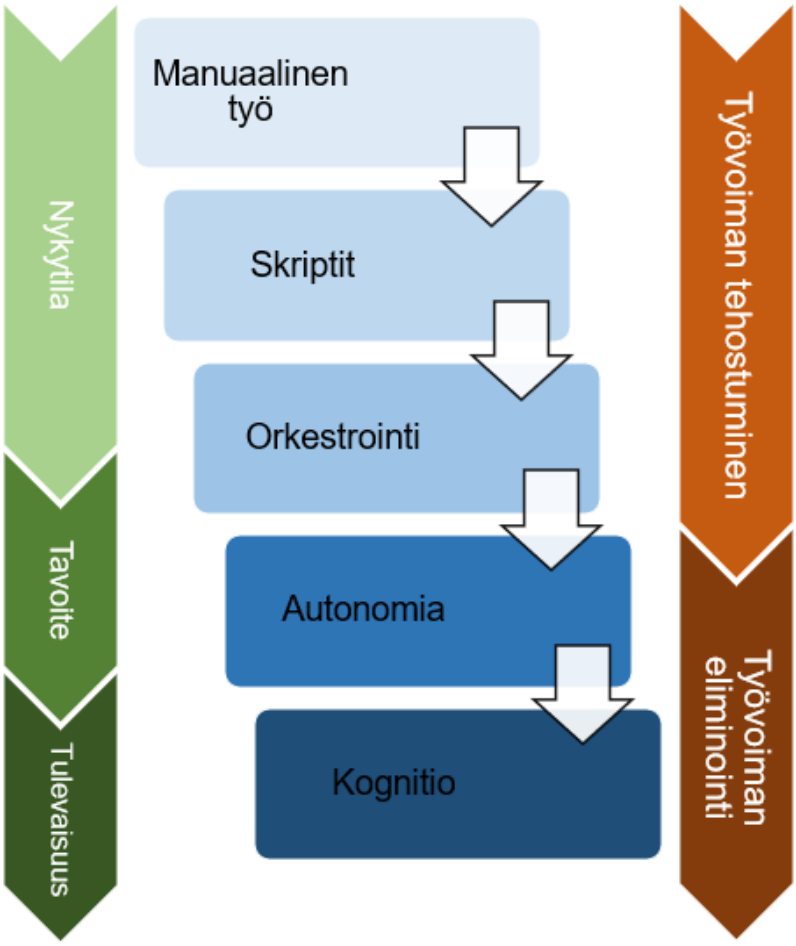
Kone jaksaa tehdä kaikenlaisia tehtäviä pyyteettömästi ja ihmistä virheettömämmin. Teollistumisen alussa 1800-luvulla erilaisia teollisuuskoneita valjastettiin aiemmin käsin tehtyihin vaarallisiin ja likaisiin töihin. Seuraavalla vuosisadalla koneet siirtyivät vähitellen tekemään myös tavallisempia, toistuvia tehtäviä ja niissä alkoi näkymään erilaisia automaattisia käyttöliittymiä. 2000-luvulla koneiden prosessointitehon edelleen kasvaessa niiden harteille on pystytty jättämään myös päätöksentekovastuuta, ja ne ovat monesti siinä myös ihmistä parempia ja tehokkaampia. 2000-luvulla koneet ovat läsnä jokapäiväisissä toiminnoissa helpottamassa ihmisten elämää enemmän kuin koskaan aiemmin. (Davenport & Kirby 2015.)

Automatisaatio herättää monissa ihmisissä myös pelkoa. Tämä on luonnollinen reaktio siihen, että pelätään oman työn ja työpaikan olevan uhattuna. Automatisaatio onkin aiheuttanut myös työttömyyttä ja niin tulee tapahtumaan

jatkossakin, mutta toisaalta sen avulla luodaan tilalle huomattavissa määrin myös uusia työpaikkoja. (Autor 2015.)

Uuden teknologian käyttöönotto ja varsinkaan siirtymävaihe ei ole täysin ongelmatonta, eikä yksittäisiltä inhimillisiltä tragedioiltakaan voida aina välttyä. Kaiken automatisaation perimmäinen tarkoitus on kuitenkin tehdä tavallisen ihmisen arjesta kaikille kokonaisuutena parempaa, joustavampaa, turvallisempaa ja vapaampaa, ei aiheuttaa tarpeetonta kärsimystä.

Institute for Robotic Process Automation (2015) kuvaa automatisaation alkavan manuaalisesta vaiheesta, joka käsittää lähinnä käsityötä ja ei-toistettavia tehtäviä. Seuraavalla tasolla hyödynnetään skriptejä avustamaan toistuvissa vakiotehtävissä. Kolmannessa vaiheessa automatisaatio on edennyt orkestrointivaiheeseen, jossa automatisaation kautta toteutetaan laajempia aktiviteetteja useiden toisiaan tukevien skriptien ja työnkulkujen toimesta. Tähän vaiheeseen saakka automatisaatio toimii lähinnä parantaen työvoiman tuotavuutta. Seuraava vaihe on tavoitetilaan kuuluva autonomia, jossa koneäly suorittaa kokonaisia epästandardeja prosesseja ymmärtäen jollakin tasolla myös niiden kontekstin. Tulevaisuudessa on vuorossa tietoinen, kognitiivinen vaihe, joka käsittää itsetietoisen, ennakoivan ja itse itseään kehittävän automatisaation. Kahdessa viimeisessä vaiheessa on kyse työvoiman tehokkuuden parantamisen sijaan työvoiman eliminoinnista.



Kuva 2. Automatisaation tasot (mukaillen Institute for Robotic Process Automation 2015)

Kuvassa 2 on esitetty tarkemmin, kuinka automatisaation eri vaiheet jakautuvat siirryttäessä nykytilasta kohti tulevaisuutta ja kuinka ne vaikuttavat työn tekemiseen ja työvoiman tarpeeseen.

2.2 Pääsynhallinta, identiteetinhallinta ja hakemistopalvelut

Automatisoinnin osalta identiteetinhallinta ja pääsynhallinta ovat merkittävässä asemassa, jotta automatisointiprosessissa käytettävä tunnus ei pääse tekemään vääriä toimenpiteitä väärissä paikoissa.

Identiteetinhallinta ja pääsynhallinta ovat laajoja käsitteitä. Lukuun ottamatta kenen tahansa käytettävissä olevia verkkosivuja, kirjaudutaan verkkopalveluihin ja tietojärjestelmiin yleensä henkilökohtaisella tunnuksella ja siihen liitetyllä salasanalla, tai käyttäen jotain muuta ennalta sovittua autentikointimenetelmää. Samalla tavalla toimivat myös eri taustajärjestelmät aloittaessaan kom-

munikoinnin keskenään. Mikäli pääsyä pyytävä tunnus löytyy palvelun käyttämästä autentikointilähteestä ja salasana tai sertifikaatti täsmää tarkistuksessa, päästetään pyytäjä etenemään palveluun sille myönnettyjen oikeuksien rajoissa. Tätä prosessia kutsutaan pääsynhallinnaksi (*engl. access management*). (Linden 2015, 11.)

Tunnusta oikeuksineen kutsutaan myös identiteetiksi. Yhdellä henkilöllä tai palvelulla voi olla samaan aikaan useampia sähköisiä identiteettejä. Yleensä sähköinen identiteetti sisältää muitakin tietoja, kuten nimi, puhelinnumero, osoite, sähköpostiosoite, titteli, esimies, sijainti ja tieto oikeuksista johonkin toiseen palveluun, eli valtuudet. Näitä kaikkia kutsutaan attribuuteiksi ja niiden kokoelmaksi (*engl. attribute collection*). (Linden 2015, 10.)

Luonnollisten henkilöiden lisäksi verkossa voi olla monia muitakin objekteja, jotka tarvitsevat identiteetin. Tällaisia voivat olla mm. huoltotunnukset (*engl. service account*), palvelinlaitteet, tulostimet ja tietokoneet. Myös näillä objekteilla on attribuutteja, mutta ne ovat usein erilaisia kuin luonnollisten henkilöiden tunnuksissa. Esimerkiksi verkossa muiden laitteiden kanssa kommunikoivan tietokoneobjektin attribuutteja voisivat olla IP-osoite, toimialue ja julkinen avain. (Linden 2015, 10.)

Verkon objekteja, eli identiteettejä on syytä hallita jollakin tavalla, jotta tunnuksilla on pääsy pelkästään niihin paikkoihin minne pitääkin, mutta ei yhtään laajemmin. Identiteetinhallinta on tavallaan prosessi, jonka avulla organisaation eri toimijoiden sähköisiä identiteettejä ja niiden rooleja hallitaan. (Linden 2015, 10–11.)

Identiteetinhallinnassa olennaista on, että jokaisella objektilla/identiteetillä on attribuutti, jolla siihen voi yksiselitteisesti viitata. Vaikka palvelussa olisi useita saman nimisiä käyttäjiä, jokaisella pitää aina olla yksilöivä tunniste (*engl. unique identifier, UID*), jolla on aina jokin arvo (Linden 2015, 12–13). Sukunimi on hyvä esimerkki attribuutista, joka voi muuttua ja sitä kautta myös käyttäjän sähköpostiosoite voi vaihtua vastaamaan uutta nimeä. Unique identifier sen sijaan viittaa aina tiettyyn objektiin sen sisältämien muiden attribuuttien muutoksista riippumatta. (Baier ym. 2011, 36.)

LDAP

Erilaisten verkkoidentiteettien/tunnusten attribuutteja on syytä säilyttää ja hallita keskitetysti. LDAP-hakemisto (*engl. Light-weight Directory Access Protocol*) on laajasti käytetty hakemistopalvelu ja sen yhteysprotokolla juuri tähän tarkoitukseen (Linden 2015, 47; Wilson s.a.).

LDAP-hakemistoon tukeutuvat myös yleisesti käytössä olevat Microsoft Active Directory ja NetIQ eDirectory -hakemistopalvelut. Myös tämän työn toimeksiantajalla on käytössä useampia eri hakemistoja eri tarkoituksia varten.

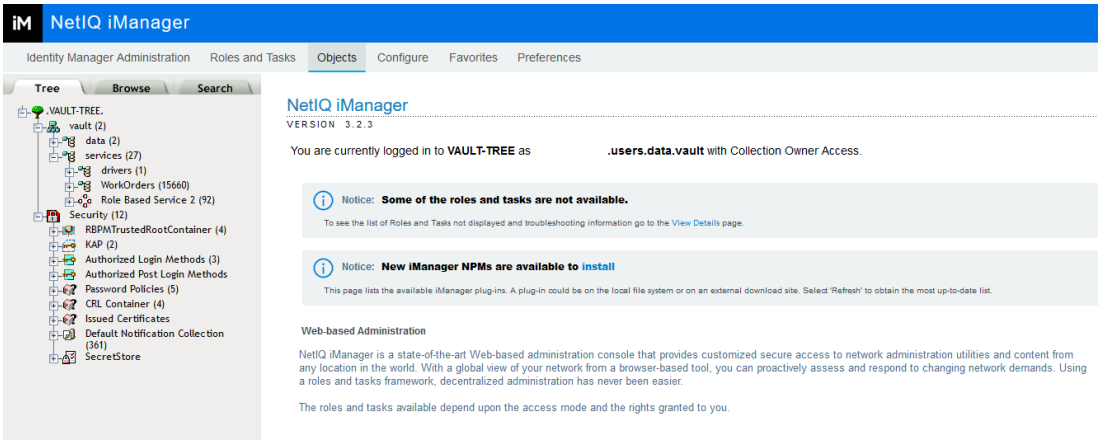
Active Directory ja Azure AD

Aktiivihakemisto (*engl. Active Directory, AD*) on Microsoftin kehittämä ja hyvin laajasti yrityksissä käytetty hakemistopalvelu, jossa säilytetään ja hallitaan käyttäjätietoja ja muita perustietoja eri objekteista verkossa. AD-toimialueet koostuvat siihen liitetyistä laitteista, joita hallitaan toimialueen ohjauspalvelimilla (*engl. domain controller*). AD on keskeinen osa Microsoftin palvelinarkkitehtuuria, ja se tukee parhaiten heidän omien tuotteiden toimintaa, mutta Microsoft ei ole halunnut rajoittua pelkästään omiin tuotteisiinsa, joten siinä on mukana myös natiivi LDAP-tuki. Tavallaan voisi sanoa, että Active Directory on Microsoftin näkemys LDAP-hakemistosta. (Microsoft 2017.)

Azure AD taas on Microsoftin pilvipalveluiden taustalla oleva kehittyneempi versio organisaatioiden omassa hallinnassa olevasta ns. on-premise Active Directorystä. Azure AD on pohjana myös kaikkien Office 365 -tuotteiden käyttäjätiedoille. (Microsoft 2020.)

eDirectory ja NetIQ Identity Manager

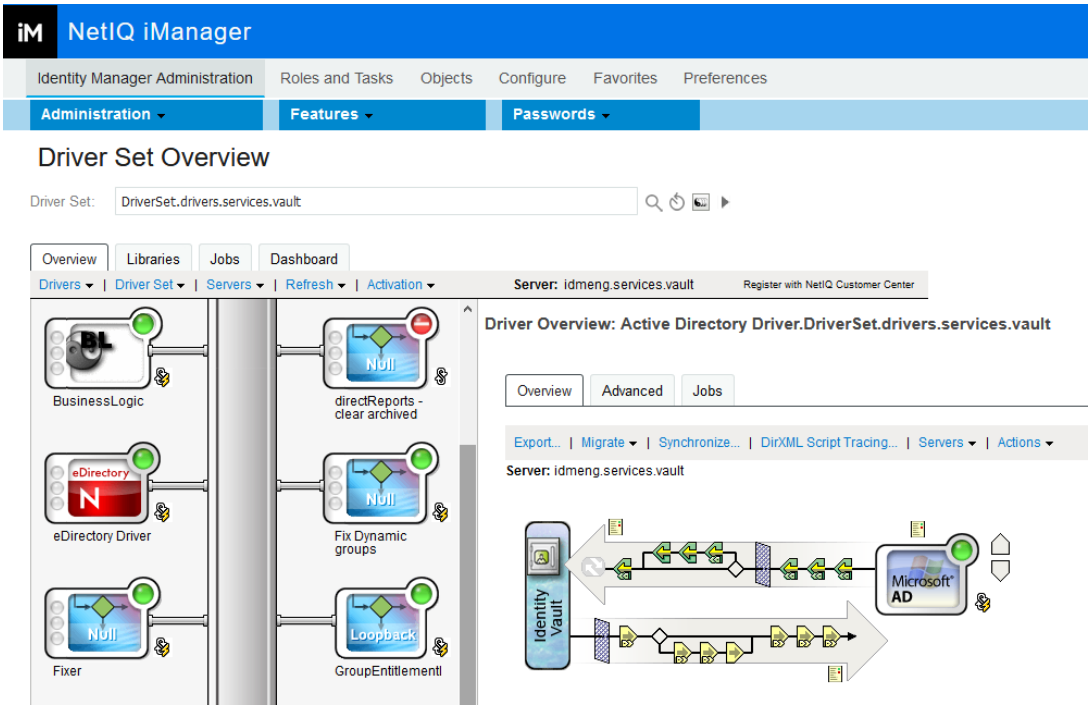
NetIQ eDirectory on Microsoftin aktiivihakemiston kaltainen LDAP v3 -protokollan mukainen hakemistopalvelu. Se koostuu erilaisista objektityypeistä, joita ovat mm. käyttäjät, tulostimet, palvelimet, sertifikaatit ja sovellukset. eDirectory on suunniteltu skaalautumaan jopa miljoonien objektien keskitetyksi hallintapisteeksi. eDirectoryn puumaisen hakemistorakenteen hallinta tapahtuu kuvassa 3 näkyvällä selainkäyttöisellä iManagerilla. (eDirectory 9.2 s.a.)



Kuva 3. eDirectoryn selainkäyttöinen iManager-hallintaliittymä

NetIQ Identity Manager on eDirectoryyn tukeutuva kokonaisvaltainen identiteetinhallintajärjestelmä (*engl. identity and access management system, IAM*), jonka avulla voidaan välittää objekteja ja niiden attribuutteja automaattisesti järjestelmästä toiseen ja tarvittaessa myös muokata niitä välissä. Järjestelmän toiminta perustuu hakemistopalveluiden tapahtumien sekä tiedostotasolla hakemistojen sisällön tarkkailuun ajureiden toimesta (*engl. driver/driverset*), joissa olevilla säännöillä ja suodatuksilla määritellään eri vaiheissa tehtävät toimenpiteet ja niihin yhdistetyt järjestelmät. (Identity Manager 4.8 s.a.)

Identity Managerin ajureiden hallinta tapahtuu iManagerilla, jolla voidaan tarkastella ja muokata kaikkien ajureiden kokonaiskuvaa tai yksittäistä ajuria kuten kuvassa 4. Hallinta on mahdollista myös työasemaan asennettavalla erillisellä Identity Manager Designerilla. (Identity Manager 4.8 s.a.)



Kuva 4. iManagerin näkymä Identity Managerin ajureista

Preimesberger (2021) listaa muiksi suosituiksi kaupallisiksi identiteetin- ja pääsynhallintaohjelmistoiksi mm. seuraavia:

- Okta
- Auth0
- Ping Identity
- Microsoft Azure Identity Management
- Oracle Identity Management

Tuotteita on markkinoilla enemmänkin, eikä valinta niiden välillä ei ole aina välttämättä kovinkaan suoraviivainen prosessi, sillä ohjelmistojen ominaisuudet ja organisaatioiden tarpeet vaihtelevat suuresti.

2.3 Automatisointimenetelmiä

Automatisointeja on mahdollista toteuttaa monella eri menetelmällä ja eri ohjelmointikielillä. Automatisoitava prosessi ja käytössä oleva palvelinalusta määrittää toteuttajan osaamisen lisäksi sen, millä työkalulla on parasta edetä. Tässä luvussa käydään läpi muutamia erilaisia toteutustapoja ja yleisiä vaihtoehtoja.

PowerShell-skriptit ja skriptaus yleisesti

Kopczynski (2007, 10) toteaa skriptauksen merkittäväksi apuvälineeksi tietojärjestelmien ylläpidossa ja kehityksessä, koska sen avulla ylläpitäjä pystyy erilaisten skriptien ja skriptikielten avulla rakentamaan omatoimisesti tarvitsemansa työkalun kuhunkin tarpeeseen ja myös automatisoimaan niiden avulla henkilökohtaisia ja taustajärjestelmissä toistuvia tehtäviä. Aivan kaikki eivät luokittele skriptausta varsinaiseksi ohjelmoinniksi, mutta se ei pidä varsinkaan nykypäivän tehokkaiden ja monipuolisten skriptauskielten osalta paikkaansa muilta osin, kuin että niillä kirjoitettua ohjelmakoodia ei yleensä tarvitse ajaa erikseen kääntäjän läpi.

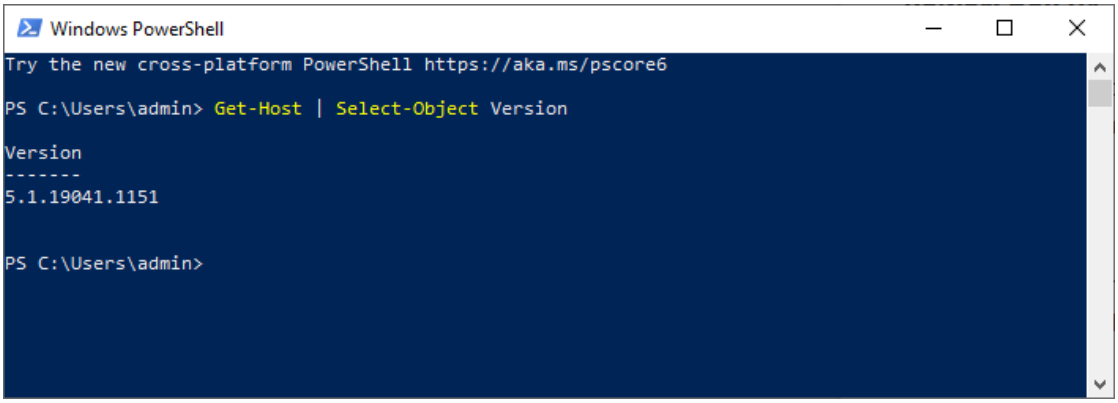
PowerShell on Microsoftin monialustainen järjestelmien hallinnan automatisoinnissa käytetty viitekehys (*engl. framework*), joka koostuu objektipohjaisesta skriptauskielestä ja komentorivikäyttöliittymästä. PowerShell on alun perin ollut vain Windows-alustojen työkalu, mutta sen uusin versio toimii Windows, Linux -ja macOS -alustoilla. (Microsoft 2021.)

Monista muista pelkästään tekstiä käsittelevistä ja palauttavista kilpailijoista poiketen, PowerShell hyväksyy ja palauttaa suorituksen aikana .NET-objekteja. PowerShelliä käytetään yleisesti järjestelmien hallinnan automatisointiin varsinkin Windows-alustoilla. Sitä käytetään myös kokonaisten sovellusten rakentamiseen ja testaamiseen. Koska PowerShell palauttaa aina objekteja, ei arvoja ole tarvetta parsia tietojen poimimiseksi. PowerShell-skriptejä on mahdollista suorittaa ajastettuina tehtävinä palvelimilla ja tarpeeksi pitkälle vietynä nämä mahdollistavat monipuolisen automatisoinnin, jossa on mahdollista huomioida myös tapahtumien lokitietojen tallentaminen ja virheentarkistus. (Microsoft 2021.)

PowerShell-skriptien kehityksessä on mahdollista hyödyntää ilmaista Windows PowerShell ISE (Integrated Scripting Environment) työkalua, joka tarjoaa käyttäjäystävällisen ja helppokäyttöisen ympäristön, jonka ulkoasua voi muokata pelkkää komentorivinäkymää vapaammin. ISE sisältää myös IntelliSense-ominaisuuden, joka tarjoaa käyttäjälle vinkkejä komentojen eri vaihtoehdoista ja parametreista sitä mukaa kun niitä kirjoitetaan. (Hassell 2017, 2–3.)

Myös Azure AD:n toimintojen ohjaaminen onnistuu PowerShellillä. Tällä hetkellä se tapahtuu AzureAD-moduulin avulla (moduulin koko nimi on Azure Active Directory PowerShell for Graph), mutta Microsoft on ohjaamassa käyttäjiä aktiivisemmin Graph Apin (koko nimeltään Microsoft Graph PowerShell SDK) käyttöön, joka tulee korvaamaan AzureAD-moduulin. (Simmons 2020.)

Windows 10:n ja Windows Server 2019:sta mukana tulee kuvassa 5 näkyvä PowerShellin versio 5.1. Kuvassa näkyy myös, kuinka komennolla *Get-Host | Select-Object Version* voi tarkistaa käytössä olevan PowerShellin version.



Kuva 5. PowerShell-konsoli

Tätä kirjoitettaessa uusin PowerShellin versio on maaliskuussa 2020 julkaistu PowerShell 7.

Ohjelmistorobotiikka

Ohjelmistorobotiikka (*engl. robotic process automation, RPA*) ei ole uusi keksintö, mutta sitä on alettu hyödyntämään uudentyyppisten työkalujen myötä tehokkaammin vasta viime vuosien aikana. Ohjelmistorobotiikalla tarkoitetaan ohjelmistokokonaisuutta, jonka ideana on automatisoida käyttäjän normaalisti tietokoneella suorittamia yksinkertaisia tehtäviä ohjelmallisesti (Sinikallio 2018, 15). Toisin sanoen ohjelmistorobotiikalla toteutetaan tietojärjestelmien toistuvia rutiininomaisia toimenpiteitä samaan tapaan kuin ihminen, mutta ohjelmallisesti. Tietokone kuitenkin kykenee suorittamaan samat toimenpiteet nopeammin ja luotettavammin ja samalla se vapauttaa työntekijän resursseja vaati-

vampaan työhön. Ohjelmistorobotiikkaa on otettu hyötykäyttöön varsinkin julkisella sektorilla ja sen käytön odotetaan lisääntyvän lähivuosien aikana huomattavasti. (Kääriäinen ym. 2018, 2.)

Kääriäinen ym. (2018, 7) toteavat että ”Ohjelmistorobotiikan ja tekoälyn (koneoppisen) työkalut ja menetelmät ovat yleiskäyttöisiä ja näin toimialariippumattomia, mutta sovellusten toteutuminen näyttäytyy usein käyttötapaus- ja organisaatiospesifisenä”. Jokainen yritys/organisaatio siis koodaa itse omia tietojärjestelmiä ja omia tarpeita vastaavan automatiikan.

Ohjelmistorobotiikan yhteydessä puhutaan usein myös tekoälystä (*engl. artificial Intellegence*). Nämä ovat erilaisia tekniikoita, joilla on mahdollista toisiaan tukien ratkaista erilaisia haasteita. Ohjelmistorobotiikka keskittyy lähinnä aiemmin mainittujen rutiininomaisten tehtävien ratkaisemiseen, kun taas tekoälyä voidaan hyödyntää päättelyä vaativien ongelmien ratkaisussa. (Kääriäinen ym. 2018, 2.)

Taustalla tapahtuvien prosessien automatisoinnista ohjelmistorobotiikan erottaa varsinkin sen ajatusmalli toimia ensisijaisesti esityskerroksessa kommunikoimatta suoraan minkään järjestelmän ohjelmointirajapinnan kautta (Sinikallio 2018, 15.)

Lacity ja Willcocks (2016) toteavat, että tulevaisuudessa yrityksillä on kasvava tarve automatisoida epämääräisiä tehtäviä ja ymmärtää tekstiä, esimerkiksi sähköpostiviestejä tai chat-viestintää. Tällaisia toiminnallisuuksia on jo nähty joidenkin yritysten sähköisissä asiakaspalvelutilanteissa, joissa automatiikka voi käydä taustalla läpi valmiiden vastausten tietokantaa ja osaa poimia, parsia ja tarjota sieltä asiakkaalle tai käsittelijälle keskustelun edetessä valmiita vastauksia.

Power Automate

Power Automate on aiemmin Microsoft Flow -tuotenimellä tunnettu Microsoftin Office 365 -pilvipalvelukokonaisuuden kautta käytettävä työkalu prosessien ja työnkulkujen rakentamiseen. Mainittakoon, että Office 365 (myöhemmin O365) on tässä yhteydessä sama asia kuin Microsoft 365, vaikka näillä onkin

pieniä eroja lisensointimielessä ja osittain myös palveluiden sisällössä sopimustasosta riippuen.

Microsoft yhtenäisti automatisointituotteiden nimiä vuonna 2019, jolloin myös Flow nimettiin uudelleen ja siitä tuli Power Automate. Power Automaten idea on mahdollistaa varsinkin loppukäyttäjälle erilaisten henkilökohtaisten prosessien automatisointi, esimerkiksi kaikkien saapuvien sähköpostien liitetiedostot voisi tallentaa sen avulla haluttuun hakemistoon ilman käyttäjän muita toimenpiteitä. Power Automate sisältää yli 300 valmista yhdistintä (*engl. connector*), joiden avulla työnkulkuja ei tarvitse välttämättä luoda täysin tyhjästä. Merkittävänä erona taustalla tapahtuvien prosessien automatisointiin Power Automate ei välttämättä edellytä järjestelmänvalvojan oikeuksia. Myös joitakin organisaatiotason toimenpiteitä on silti mahdollista toteuttaa, mutta melko rajallisesti. (Nevalainen 2020.)

Yksi esimerkki Power Automaten hyödyntämisestä taustaprosessien automatisoinnissa voisi olla tilanne, jossa loppukäyttäjä vastaa O365:n Forms-palvelun lomakkeella kyselyyn ja Power Automate suorittaa taustalla vaadittavia toimintoja käyttäjän antamien vastausten perusteella. Tästä käytännön esimerkki myöhemmin.

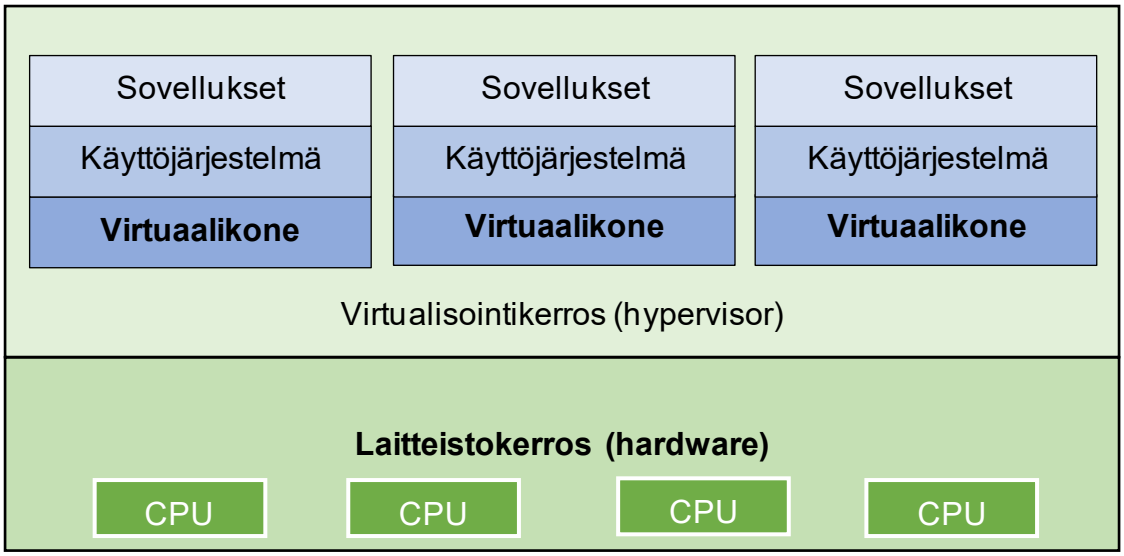
2.4 Palvelimet ja virtualisointi

Palvelinresurssit ovat rajallisia ja organisaatioiden ICT-yksiköt joutuvat jatkuvasti pohtimaan järjestelyitä resurssien tehokkuutta, skaalautuvuutta, joustavuutta, mukautuvuutta ja säästöjä tavoitellen. Tarpeet kasvavat ja muutoksiin täytyy pystyä reagoimaan nopeammin, mutta rahaa ei välttämättä kuitenkaan ole käytettävissä samassa suhteessa. Fyysinen palvelin hyödyntää resursseja vain murto-osan laitteiston tarjoamasta kapasiteetista. Virtualisoinnin avulla hyötysuhdetta on mahdollista kasvattaa jopa 70–80 prosenttiin ja vähentää samalla fyysisen laitteiston määrää. Virtualisointiteknologia ei ole uusi keksintö, mutta yhä enenevissä määrin se on osoittautunut hyödylliseksi apukeinoksi resurssien kasvattamiseen hallintakuorman ja kustannusten ylettömästi kasvamatta. (Golden 2009, 3–7 & 10–11.)

Virtualisointi liittyy automatisointiin siinä mielessä, että kaikki palvelut tarvitsevat aina jonkin alustan toimiakseen ja automatiikan avulla on mahdollista sekä jakaa kuormaa, että jopa muodostaa uusia virtuaalipalvelimia kuormahuippujen niin vaatiessa. Automatiikka voi siis seurata järjestelmän ja palveluiden kuormitusta ja tehdä ennakoivia toimenpiteitä tarpeen mukaan.

Palvelinvirtualisointi

Tietotekniikassa virtualisoinnilla tarkoitetaan menetelmää, jossa fyysisen palvelinjärjestelmän tarjoamia resursseja jaetaan useammaksi lohkoksi ja sitä kautta yksittäisiksi loogisiksi resursseiksi. Fyysiset laitteet eivät siten näy suoraan niitä käyttäville järjestelmille ilman erityisjärjestelyitä. Vastaavalla tavalla useita fyysisiä resursseja on mahdollista yhdistää virtualisoinnin avulla yksittäiseksi loogiseksi resurssiksi. (Golden 2009, 3.)



Kuva 6. Palvelinvirtualisoinnin kerrokset (mukaillen Golden 2009, 14)

Kuvassa 6 on havainnollistettu palvelinvirtualisoinnin kerrokset. Käytännössä yhdelle fyysiselle palvelinlaitteelle asennetaan virtualisointikerroksena toimiva hypervisor, jonka päälle asennetaan yksi tai useampia virtuaalisia palvelinlaitteita käyttöjärjestelmineen ja sovelluksineen jakamaan laitteen fyysisiä resursseja.

Sovellusvirtualisointi

Palvelinkäyttöjärjestelmien virtualisoinnin lisäksi myös yksittäisiä sovelluksia on mahdollista virtualisoida. Samalla tavalla kuin palvelinvirtualisoinnissa, on tähän tarpeeseen olemassa erilaisia ratkaisuja. Sovellusvirtualisointia toteutetaan pääasiassa siksi, koska se mahdollistaa sovellusten keskitetyn jakelun ja päivitettävyyden. On mahdollista, että organisaatiossa on satoja tai jopa tuhansia työasemia, joilla kaikilla on pääosin samat sovellukset. Sovellusvirtualisoinnin avulla ne kaikki voidaan päivittää keskitetysti yhdellä kerralla, eli palvelimella tehty muutos vaikuttaa suoraan kaikkiin koneisiin. Lisäksi sovellusvirtualisointi mahdollistaa eri alustalle ohjelmoidun sovelluksen toiminnan toisella alustalla ja jopa useita samaan aikaan käynnistettäviä versioita samasta sovelluksesta. (Vmware s.a.)

2.5 Tietoturva

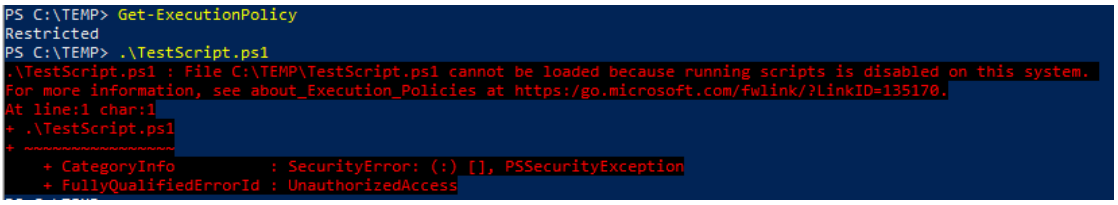
Koska automatisointeja suorittavilla palvelimilla ja niihin liitetyillä tunnuksilla on oltava paikoin melko laajojakin oikeuksia muille palvelimille tai palveluihin, on niiden tietoturvapäivitysten syytä tapahtua automaattisesti ja niiden toimintaa on syytä seurata jopa normaaliakin aktiivisemmin. Kaikissa toiminnoissa on syytä käyttää vähimpien tarvittavien oikeuksien mallia (*engl. least privilege model*), joka tarkoittaa, että palvelimeen ja kaikkiin siellä suoritettaviin toimintoihin liittyvillä tunnuksilla, prosesseilla ja sovelluksilla on vain ne minimioikeudet, joilla toiminnon saa suoritettua. (Kopczynski 2007, 118.)

Myös verkkoliikennettä sekä eri oikeusryhmiin kuuluvien tunnusten pääsyä automatisointipalvelimille on syytä rajoittaa ja seurata mahdollisimman kattavasti.

PowerShell-skriptien suorituksesta Kopczynski (2007, 86) toteaa, että aiemmin hyvin laajasti käytetyn Windows Scripting Host -rajapinnan skriptien tietoturvan lähes täydellistä puuttumista on kritisoitu ja siksi Microsoft kehitti PowerShelliin suojauskäytännöt (*engl. execution policy*). Suojauskäytäntöjä ovat seuraavat:

- *Restricted*. Toimii vain interaktiivisessa tilassa suoraan konsolilta ja estää skriptien suorituksen kokonaan. Tämä on oletusasetus palvelimen asennuksen jälkeen.

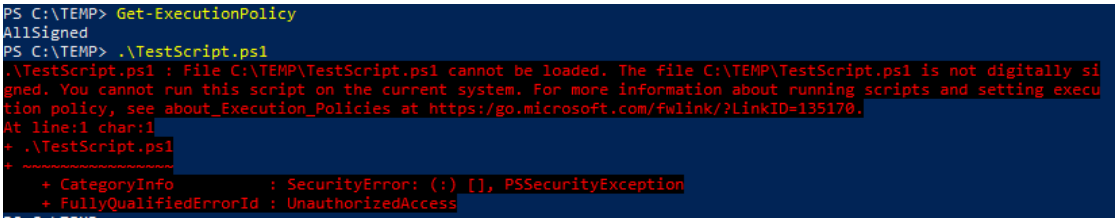
  Kuvassa 7 on esitetty PowerShell-konsoli, jossa suojauskäytännön taso on *Restricted*, jolloin yritys suorittaa skriptiä päättyy virheeseen.



Kuva 7. Restricted-tason suojauskäytäntö

- *AllSigned*. Turvallisin vaihtoehto skriptien suoritukselle, sallii vain digitaalisesti allekirjoitetut skriptit. Jos skriptiä muokataan, se ei toimi ilman uutta varmennusta.

  Kuvassa 8 on esitetty PowerShell-konsoli, jossa suojauskäytännön taso on *AllSigned*, jolloin yritys suorittaa allekirjoittamatonta skriptiä päättyy virheeseen.



Kuva 8. AllSigned-tason suojauskäytäntö

- *RemoteSigned.* Sallii samassa koneessa luodut skriptit, mutta muualta ladattujen skriptien tulee olla digitaalisesti allekirjoitettuja.

  Kuvassa 9 on esitetty PowerShell-konsoli, jossa suojauskäytännön taso on *RemoteSigned*, jolloin paikallisesti samassa koneessa luodun skriptin suoritus onnistuu. Jos skripti olisi peräisin jostakin muualta, sitä ei suoritettaisi.

Kuva 9. RemoteSigned-tason suojauskäytäntö

- *Unrestricted.* Täysin rajoittamaton. Kaikki skriptit suoritetaan, mutta ulkoisista lähteistä ladattujen skriptien kohdalla kysytään käyttäjältä erillistä kuittausta. Tätä vaihtoehtoa ei suositella kuin testikäytössä.
- *Bypass*. Ohittaa tarkistuksen suojauksen kokonaan. Tätä ei suositella kuin testikäyttöön, tai jos käytössä on joku toinen turvamekanismi.
- *Undefined*. Poistaa määrityksen. Käytännössä tällä palautetaan aiemmin asetettu suojauskäytäntö takaisin oletukseksi.

Skriptien allekirjoituksella ei voi taata tekijän tuottaman ohjelmakoodin tasoa, mutta sillä voi suojata koodit muutoksia vastaan ja varmenteesta riippuen myös varmistua koodin tekijästä. Skriptien allekirjoitus edellyttää code-signing-varmennetta (*engl. code-signing-certificate*). Sellaisen voi tehdä itsekin käyttäen esimerkiksi organisaation omaa PKI-järjestelmää, mutta luotettavuuden lisäämiseksi voi olla syytä harkita jonkin luotettavan ulkoisen varmennepalvelun käyttämistä. Tällaisia ovat esimerkiksi Digicert, GeoTrust, Thawte ja Comodo. (Kopczynski 2007, 86.)

Kun varmenne on noudettu ja talletettu varmennesäilöön (*engl. certificate store*), tapahtuu koodin allekirjoitus komennolla *Set-AuthenticodeSignature*, jolle annetaan parametrina allekirjoitettavan skriptin tiedostopolku, sekä allekirjoitusvarmenne.

Esimerkki komennosta:

Set-AuthenticodeSignature -filePath script.ps1 -certificate @(Get-ChildItem cert:\CurrentUser\My -codeSigningCert) [0] -IncludeChain "All"

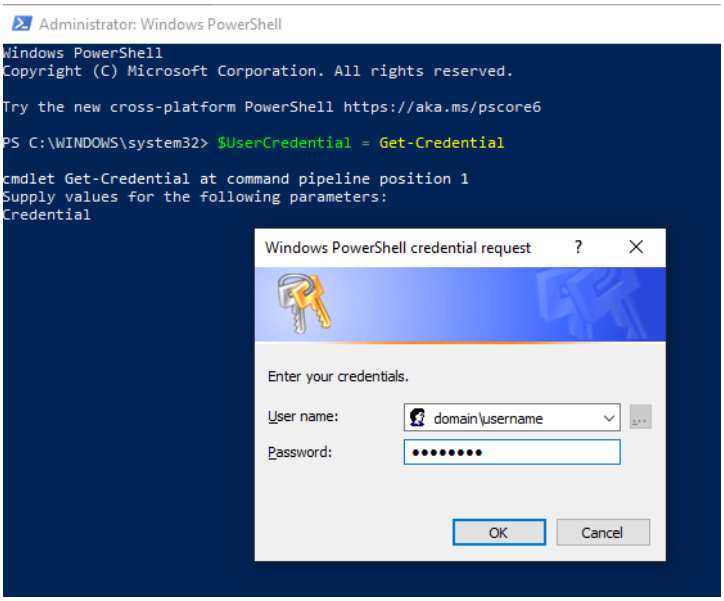
Edellisen esimerkin komennolla allekirjoitettaisiin script.ps1-niminen tiedosto käytetyn tunnuksen omasta varmennesäilöstä löytyvällä code-signing-varmenteella ja lisäksi varmistettaisiin, että mukana on koko varmenneketju (*engl. certificate chain*. Komennon onnistuneen suorituksen jälkeen tiedoston loppuun lisätään allekirjoituslohko, jossa allekirjoitusvarmenne on "*SIG # Begin signature block*" ja "*SIG # End signature block*" –merkintöjen välissä. Jos tuo lohko poistetaan tiedostosta, poistuu myös sen allekirjoitus. (Kopczynski 2007, 94–105.)

Valitun suojauskäytännön voi kohdentaa kolmelle eri kohteelle (*engl. scope*). Näitä ovat Process, CurrentUser ja LocalMachine. Suojauskäytäntöä voi muuttaa vain koneen järjestelmänvalvojan oikeuksin ja tämän vuoksi PowerShell-komentokehote pitää käynnistää tunnuksesta riippumatta korotetuilla oikeuksilla (*engl. elevated rights*). Voimassa olevan asetuksen näkee konsolilla annettavalla komennolla *Get-ExecutionPolicy* ja vastaavasti sitä muutetaan komennolla *Set-ExecutionPolicy*, jonka perään annetaan parametreinä

haluttu suojauskäytäntötaso, sekä vaihtoehtoisesti myös kohde, eli scope. (Kopczynski 2007, 88.)

Brown (2018) toteaa, että suojaustasojen lisäksi skriptien sisältämien komentojen onnistuminen ja toiminta riippuu myös sen tunnuksen oikeuksista, jolla skriptejä suoritetaan. PowerShell-skriptien suorituksen aikana on mahdollista käyttää joihinkin toimintoihin tarvittaessa myös eri tunnusta, kuin millä itse konsoli on käynnistetty.

Interaktiivisessa sessiossa käyttäjältä voidaan kysyä salasana ja tallettaa se turvallisesti muuttujaan myöhempää käyttöä varten komennolla *$UserCredential = Get-Credential*, jolloin käyttöjärjestelmä kysyy tunnusta omalla dialogillaan kuten kuvassa 10 on esitetty. Tunnustiedot sisältävää muuttujaa voi hyödyntää myöhemmin skriptin suorituksen aikana.



Kuva 10. Tunnuksen tallettaminen muuttujaan interaktiivisessa tilassa

Kuvan 10 esimerkissä tunnus ja salasana talletetaan muuttujaan *$UserCredential*, mutta se voisi olla nimetty miksi tahansa muuksikin, eli muuttujan nimi ei ole olennainen.

Skriptiä automaattisesti suoritettaessa ei kuitenkaan ole interaktiivisesta konsolista poiketen käyttäjää, jolta voisi tunnusta kysyä, joten tunnuksen tallentaminen ja sen noutaminen skriptissä hyödynnettäväksi pitää hoitaa eri menetelmällä. Tähän on olemassa eri vaihtoehtoja.

Helpoin vaihtoehto on tallettaa tunnus suojattuna skriptin käynnistävän ajastetun tehtävän asetuksiin. Tällöin skriptit toimivat tunnuksen oikeuksilla suoraan ja jos skriptissä tarvitsee avata yhteyksiä muihin palvelimiin, voidaan skriptissä tarvittaessa käyttää optiota *NegotiateWithImplicitCredential*, joka kertoo PowerShellille, että halutaan käyttää oletuksena tapahtuvan Kerberos-tunnistuksen sijaan tunnuksen sisältävää objektia, joka on juuri se ajastetun tehtävän asetuksissa määritelty tunnus. (Brown 2018.)

Esimerkki komennosta:

$Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri http://<server-FQDN>/PowerShell/ -Authentication NegotiateWithImplicitCredential

Tämän menetelmän etuna on se, ettei tunnusta tarvitse tallettaa skriptin sisälle lainkaan. Miinuspuolet ovat siinä, että tunnuksia voi olla tällä tavalla määriteltynä vain yksi kappale jokaista skriptiä varten ja jos käytetyn tunnuksen salasana vaihtuu, se pitää muuttaa myös kaikkien sitä käyttävien tehtävien asetuksissa. Menetelmä on lisäksi ongelmallinen Office365-yhteyksien kanssa. (Brown 2018.)

Toinen vaihtoehto on luoda tunnusobjekti PowerShell-konsolilla ja tallettaa se salatuksi tekstitiedostoksi. Tämä voi kuulostaa turvattomalta, mutta tunnus ei silti paljastu koskaan, sillä sen sisältöä käsitellään Windows Data Protection -rajapinnan kautta, joka tarkoittaa, ettei tiedosta voi lukea selväkielisenä missään vaiheessa ja lisäksi salaus aukeaa vain ja ainoastaan sillä koneella ja sen käyttäjän tunnuksella, jolla tiedosto on luotu. (Brown 2018.)

3 TOIMINTAYMPÄRISTÖ JA TARVEKARTOITUS

Tässä luvussa esittelen tarkemmin toimeksiantajan toimintaympäristön ja automatisointiprosessien tarvekartoituksen. Toimintaympäristön kuvaus perustuu toimeksiantajan verkkosivuilla olevien tietojen lisäksi organisaation sisäisiin markkinointimateriaaleihin ja haastatteluihin. Myös tarvekartoitus perustuu pääasiassa eri toimijoiden haastatteluihin.

Toimintaympäristö

Jyväskylän ammattikorkeakoulu (JAMK) on vuonna 1994 perustettu Keski-Suomessa sijaitseva monialainen ja arvostettu korkeakoulu.

JAMK tarjoaa tällä hetkellä 40 erilaista tutkintoa 7 eri alalta. Päätoimisia opiskelijoita koulussa on noin 8500 yli 70 maasta. (Jyväskylän ammattikorkeakoulu 2021.) Henkilökuntaa organisaatiossa on noin 800. Hallintoyksikköön kuuluvalla ICT-Palvelut -tulosalueella työskentelee ICT-päällikön lisäksi 19 henkilöä eri rooleissa IT-tukihenkilöistä järjestelmien teknisiin pääkäyttäjiin, projektipäälliköihin ja järjestelmäsuunnittelijoihin.

Karkeasti yleistäen ICT-palveluiden vastuulla on oppilaitoksen tietojärjestelmien perusinfrastruktuurin ja näihin tukeutuvien muiden palveluiden suunnittelu, kehitys ja ylläpito. Nykypäivänä automatisaatio auttaa kohdistamaan ylläpidon työaikaa nimenomaan erilaisten digitaalisten palveluiden kehitykseen ja järjestelmien perinteinen ylläpito on hyvin pitkälle automatisoitua.

Toimeksiantajan IT-infrastruktuuri koostuu tyypilliseen tapaan palvelinlaitteista, tallennusjärjestelmistä, verkkolaitteista, näiden kaikkien välisistä verkkoyhteyksistä, sekä palvelukerroksen tietojärjestelmistä. Palvelinalustojen käyttöjärjestelmänä on pääosin Windows Server 2016 tai 2019. Myös Linux-palvelimia on käytössä.

Palvelinlaitteita on hallinnassa yhteensä noin 250 kappaletta, joista vain pieni osa on fyysisiä, eli suurin osa palvelimista on virtuaalisia. Erilaisia henkilöstön ja opiskelijoiden käyttämiä tietojärjestelmiä palvelinlaitteissa on toiminnassa kymmeniä, ellei peräti satoja.

Hakemistopalveluina on käytössä Microsoft Active Directory, sekä NetIQ eDirectory. Käyttäjätunnuksia eri hakemistopalveluissa on yhteensä noin 25 000. Tässä luvussa on mukana henkilöstön ja päätoimisten opiskelijoiden lisäksi myös avoimen ammattikorkeakoulun opiskelijat sekä lukitut, poistoa odottavat tunnukset. Tunnusten hallinta tapahtuu henkilöstön osalta HR-järjestelmässä ja opiskelijoiden osalta opiskelijahallintojärjestelmässä. Tunnukset ja niiden

attribuutit siirtyvät järjestelmien välillä pääasiassa NetIQ Identity Managerin avulla.

Virtualisointialustana toimeksiantaja käyttää sekä Microsoft Hyper-V:tä, että VMware vSphereä. Näistä VMware vSphere on pääasiallinen alusta tuotantojärjestelmille ja Hyper-V on käytössä lähinnä testiympäristöjä varten. VMwaren ESXi-hypervisoria ja sen sisältämiä virtuaalikoneita ajetaan kuuden noodin HPe Simplivity -klusterissa, joka on vikasietoisuuden vuoksi kahdennettu kahteen eri konesaliin. Simplivity on HCI-ratkaisu (engl. hyper-converged infrastructure), jossa palvelinkapasiteetti, levyjärjestelmä, deduplikoitu tallennus ja järjestelmän sisäiset varmistukset on keskitetty yhteen kokonaisuuteen.

Palvelinjärjestelmät ja muut kriittiset verkkopalvelut on eriytetty omiin virtuaalilähiverkkoihin (*engl. virtual local area network, VLAN*) ja erillisiin aliverkkoihin. Verkkoliikenne näiden ja muiden verkkojen välillä on sallittu whitelist-periaatteella vain tarpeen mukaan.

Organisaatiossa on käytössä myös Microsoft Office365 -pilvipalvelu ja sen rinnalla Azure AD, sekä useita muita Microsoft Azuren tarjoamia pilvipalveluita. Käyttäjäobjektit ja koneobjektit synkronoidaan paikallisesta on-premise aktiivihakemistosta edelleen Microsoftin pilvipalveluun Azure AD Connect -sovelluksella. Joitakin palveluita on lisäksi hankittu SaaS-palveluina (*Software as a Service*) muilta ulkoisilta yhteistyökumppaneilta. SaaS-palveluihin kirjaudutaan joko LDAP-protokollan, ADFS-palvelun, tai sitä vastaavan Azure-palvelun avulla. Lisäksi toimeksiantajalla on oppilaitosympäristönä ollut joidenkin palveluiden kohdalla mahdollista hyödyntää HAKA-federaation Shibboleth-pohjaista käyttäjätunnistusjärjestelmää, joka tarjoaa käyttäjille mahdollisuuden kirjautua yli 300:aan eri palveluun globaalisti (Haka-käyttäjätunnistusjärjestelmä s.a.).

Tarvekartoitus

Työn alkuvaiheessa kävin useita keskusteluita toimeksiantajan ICT-palveluiden henkilöstön, sekä muutamien eri tulosyksiköiden satunnaisten työntekijöiden kanssa. Näissä keskusteluissa kävi selväksi, että automatisoinnille löytyy hyvin monenlaisia tarpeita. Niitä löytyy sekä ICT-palveluiden omista lähtökoh-

dista, eli organisaation ydintoimintoja tukevien palvelinsovellusten taustatoiminnallisuuksista, että myös eri tukipalveluyksiköiden päivittäistoiminnoista. Jälkimmäiset eivät kuitenkaan ole taustapalveluita, vaan nimenomaan käyttäjien manuaalisesti jossakin tietyssä sovelluksessa suorittamia toimintoja, joihin ohjelmistorobotiikka soveltuu paremmin.

ICT-palvelun järjestelmäsuunnittelijoiden kanssa käytyjen keskusteluiden perusteella tunnistetut tarpeet luokiteltiin kolmeen kategoriaan: kriittiset, vähemmän kriittiset ja manuaalisesti käynnistettävät. Arviointi perustui niiden aiheuttamaan työkuormaan, kyseisen aktiviteetin merkitykseen loppukäyttäjille ja luonnollisesti myös itse toimenpiteen luonteeseen.

Kriittisimmät tunnistetut tarpeet automatisoinnille olivat seuraavat:

- Office365:n lisenssien määrittely käyttäjille
- jaettujen Office365-sähköpostilaatikoiden luonti ja oikeushallinta
- jaettujen verkkohakemistojen luonti ja oikeushallinta
- tiedostosiirtoja taloushallinnon palvelinsovellusten ja toimittajien järjestelmien välillä.

Vähemmän kriittisiksi toimenpiteiksi luokiteltiin

- lupalomake käyttäjän kuvan julkaisulle
- orpojen Office365-ryhmien etsintä ja poisto
- web-palvelinten lokitiedostojen siivoustoimet
- sähköpostin jakelulistojen siivous (lukittujen ja poistettavien tunnusten poisto jakelusta).

Manuaalisesti käynnistettäviksi toimenpiteiksi luokiteltiin mm.

- tiettyihin ryhmiin kuuluvien käyttäjien lähettäminen sähköpostiraporttina edelleen
- halutun aliverkon vapaiden IP-osoitteiden listaaminen
- erilaisten käyttäjätietoraporttien muodostaminen
- palvelinjärjestelmän status/huoltoraporttien muodostaminen ja lähettäminen sähköpostitse asianosaisille
- käyttäjätunnusten tiettyjen attribuuttien muokkaus tai nollaus.

Manuaalisesti käynnistettävät toimenpiteet sisältävät kymmeniä erilaisia aktiviteetteja. Nämä ovat kaikki sellaisia, joissa valmis skripti nopeuttaa ja helpottaa työn tekemisessä, mutta joiden ei kuitenkaan tarvitse olla kokonaan automati-

soituja. Osa skripteistä käynnistetään vain tarvittaessa ja ne suorittavat toimenpiteensä taustalla. Osa toimii interaktiivisesti, kysyen suorituksen aikana käyttäjältä lisätietoja, joiden perusteella toimenpiteet suoritetaan.

4 AUTOMATISOINNIN KEHITTÄMISPROSESSI

Tässä luvussa kerron ensin toteutuksessa käytettävistä menetelmistä ja avaan taustoja, joiden vuoksi niihin on päädytty. Sen jälkeen kerron muista komponenteista, jotka eivät itsessään välttämättä toteuta automatisointia, mutta joita kuitenkin tarvitaan kokonaisuudessa. Lopuksi esittelen muutaman työn toteutuksessa tuotetun varsinaisen automatisoinnin.

4.1 Toteutusmenetelmien valinta

Erilaisia toteutusvaihtoehtoja oli useita, mutta koska palvelinalustat olivat pääosin Windows-pohjaisia ja toteutettavat automaatiot liittyvät nimenomaan järjestelmien taustalla tapahtuviin, paikoin hyvinkin laajoja oikeuksia vaativiin toimintoihin, todettiin PowerShell useimpiin tarpeeseen soveltuvimmaksi työkaluksi. PowerShell itsessään ei ole automatisaatiomenetelmä, mutta sillä luotavat skriptit voivat sellaisia olla. Käytännössä toteutus tapahtui niin, että luotiin useita PowerShell-skriptejä, jotka suorittavat ajastetusti haluttuja toimenpiteitä eri palvelimilla.

Joissakin tapauksissa kyse oli käyttäjän aloittamasta toimenpiteestä, jonka piti käynnistää toimenpiteitä taustalla. Näitä varten olisi voitu hyödyntää jo olemassa olevan Identity Managerin itsepalveluliittymää, mutta koska tarve koski vain Azure-palveluita, oli niiden kohdalla järkevintä toteuttaa automatisointi Microsoft Power Automaten avulla. Tässä valinnassa oli taustalla myös tarve kerätä lisää tietoa Power Automaten toiminnasta ja sen mahdollisesta laajemmasta hyödyntämisestä jatkossa.

Paikallisesti suoritettavien skriptien rinnalla päädyttiin joissakin tapauksissa käyttämään Azuren automaatiotunnuksia (*engl. automation account*) ja niiden Runbook-toimintoa, jotta pelkästään Azuressa tapahtuvat toiminnot saatiin automatisoitua ilman Azuren ulkopuolelta tulevia herätteitä. Käytännössä runbookit ovat Azuressa toimivia ajastettuja PowerShell-skriptejä.

Mikäli olisi ollut tarvetta automatisoida toimenpiteitä, joissa pyritään jäljittelemään käyttäjän valintoja ja painikkeiden paineluita jossakin kolmannen osapuolen sovelluksen käyttöliittymässä, olisi todennäköisesti ollut syytä hyödyntää ohjelmistorobotiikkaa. Tällaisiakin tarpeita kartoituksessa tunnistettiin, mutta ne eivät olleet ajankohtaisia toteutushetkellä ja lisäksi ne olivat tämän työn rajauksen ulkopuolella. Tästä tarkemmin päätännössä.

4.2 Virtuaalipalvelinten luonti

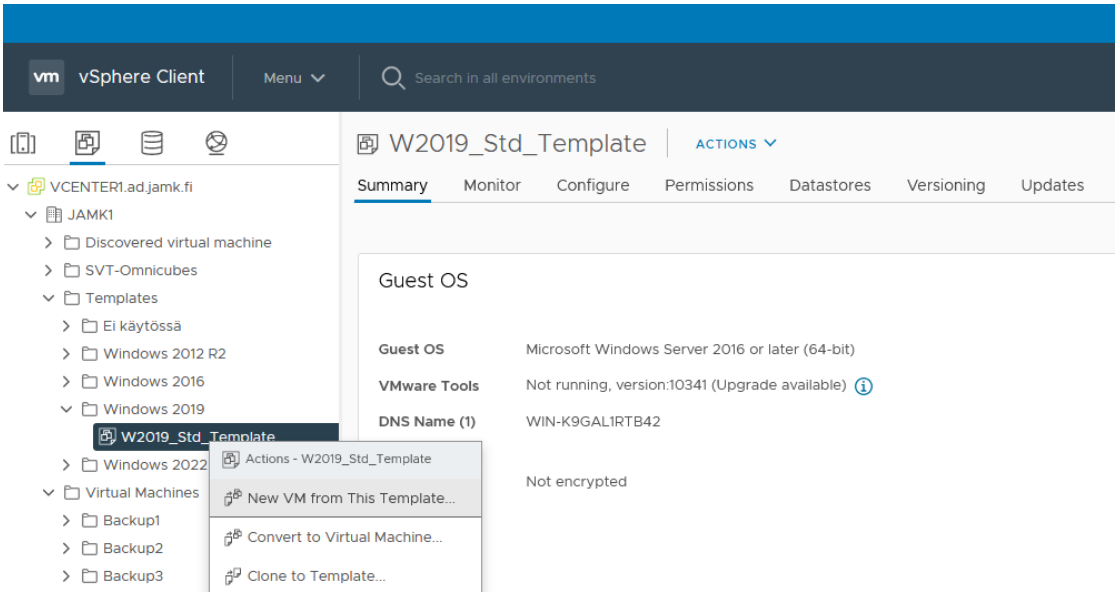
PowerShell-skriptejä on mahdollista suorittaa monella eri tavalla ja monella palvelimella. Hallittavuuden, tietoturvan, selkeyden ja lokitietojen helpomman keräämisen vuoksi oli järkevää asentaa kaksi erillistä Windows 2019 -virtuaalipalvelinta, joissa pääosa skripteistä suoritetaan.

Yksikin virtuaalipalvelin olisi tarpeeseen riittänyt, mutta tässä tapauksessa taloushallinnon sovellusten tarvitsemat toiminnot olivat siinä määrin muista poikkeavia, että niitä varten oli syytä luoda kokonaan oma palvelimensa. Tätä päätöstä tuki sekin, että osa taloushallinnon sovelluksista on hankittu ulkoiselta toimittajalta SaaS-palveluna ja sinne meneviä siirtoja varten tarvittiin joka tapauksessa erillistä palvelinta, jossa ajetaan toimittajan agenttisovellusta, joka hoitaa siirtoja edelleen toimittajalle ja sieltä takaisin. Osa tämän työn tekemisen aikana tuotetuista skripteistä vie ja noutaa tiedostoja eri palveluista tuon agentin käsiteltäväksi.

Muutamia skriptejä oli edellisten lisäksi tietoturvan ja selkeyden vuoksi järkevämpää ajaa suoraan niillä palvelimilla, joissa toimenpide tapahtuu.

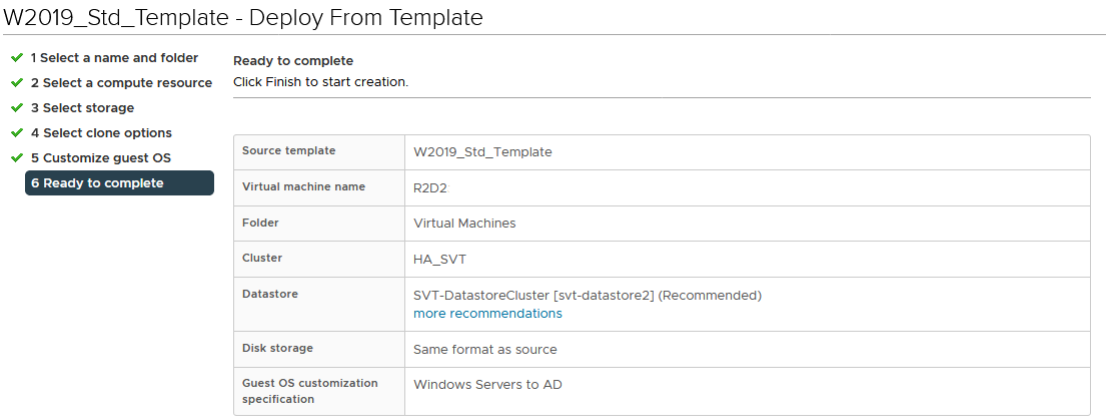
Virtuaalisia Windows-palvelimia ei ollut tarvetta asentaa täysin nollasta, sillä järjestelmässä oli jo valmiiksi luotuna organisaation omien käytäntöjen mukainen mallipalvelin (*engl. virtual machine template*), josta pystyi nopeasti ja helposti monistamaan uusia virtuaalipalvelimia.

Kuvassa 11 on esitetty, kuinka aloitetaan virtuaalikoneen luonti aiemmin luodusta mallipalvelimesta vCenterin selainkäyttöisessä hallintaliittymässä (vSphere Client).



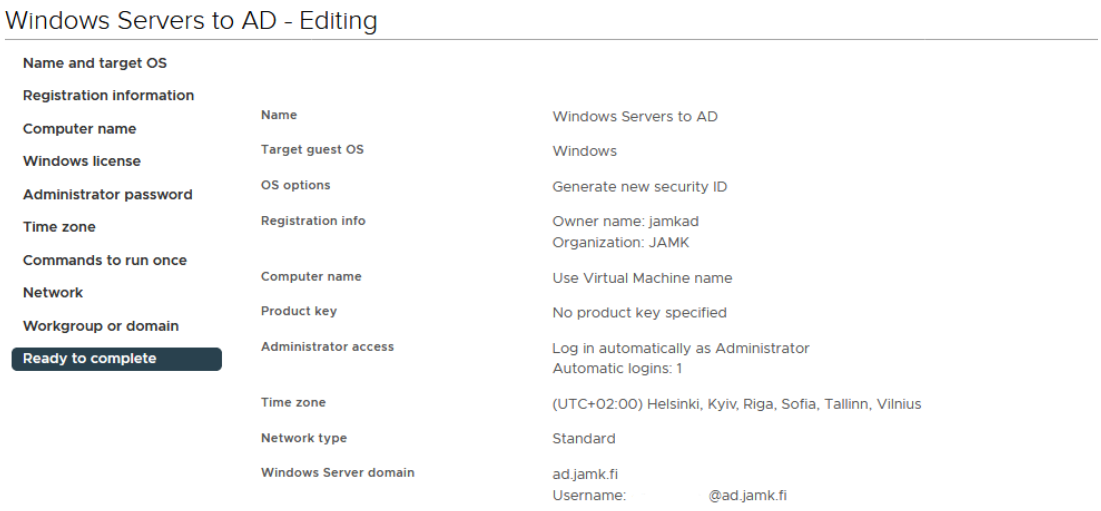
Kuva 11. Virtuaalikoneen luonnin käynnistys mallipalvelinta käyttäen

Prosessi jatkuu tästä edelleen valitsemalla vaihe kerrallaan eri optiot, joita uuden palvelimen alustaminen edellyttää. Kuvassa 12 on menossa viimeinen tiivistelmäsivu mallipalvelimesta luotavan uuden palvelimen luontiprosessissa.



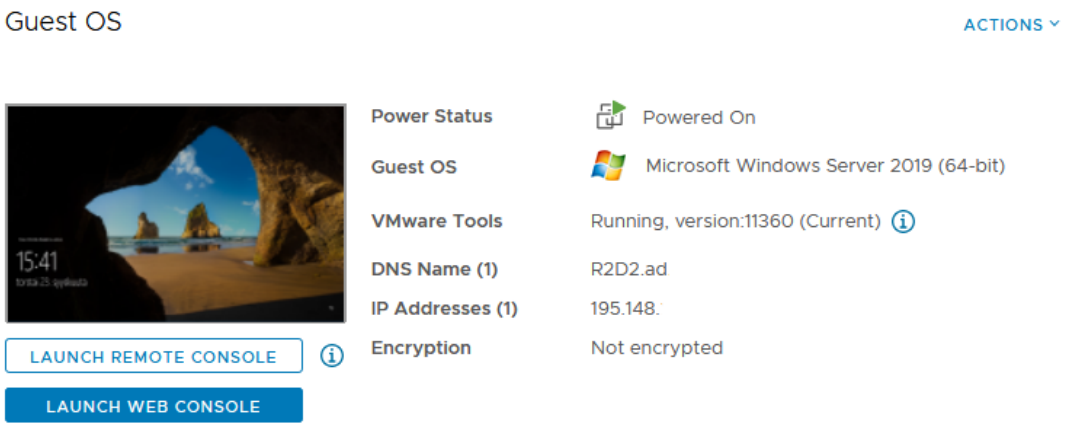
Kuva 12. Virtuaalikoneen luonti mallipalvelimesta

Esimerkissä on käytetty tämän prosessivaiheen automatisoimiseksi luotua määritystiedostoa (*engl. guest OS customization specification*), jonka avulla uusi palvelin liitetään automaattisesti organisaation AD-hakemistoon. Määritystiedostossa on kerrottu myös millä menetelmällä uusi palvelin asennetaan, millä tunnuksella, mihin verkkoon ja niin edelleen. Itse määritystiedoston luontivaiheen viimeinen koontisivu on esitetty kuvassa 13.



Kuva 13. VMwaren mallikoneen määritystiedoston luonti

Palvelimen asennuksen jälkeen sitä voi hallita etänä samaan tapaan kuin mitä tahansa muutakin palvelinta. Mikäli verkkoyhteys ei ole käytettävissä tai palvelin edellyttää jotain muita lisätoimenpiteitä ennen etähallintayhteyden avaamista, voi palvelimen konsolille kirjautua VMware vSphere -ympäristön hallinnassa käytettävän vCenterin selainliittymän kautta (kuva 14).



Kuva 14. vCenter-hallinnan näkymä virtuaalikoneesta

Kun palvelin luodaan tällä menetelmällä, se saadaan liittymään AD-toimialueeseen automaattisesti, minkä jälkeen olennainen osa sen käytettävyyteen ja tietoturvaan liittyvistä asetuksista päivittyy toimialueen ohjauskoneelta ryhmäkäytäntöjen avulla. Näiden toimenpiteiden jälkeen palvelin on päivitetty ja identtinen muiden järjestelmään kuuluvien palvelinten kanssa, eikä se edellytä tietoturvan osalta erityistä koventamista enää erikseen. Ryhmäkäytäntöjen

määrittely on osa kokonaisuuden automatisointia, mutta se on erillinen osa-alueensa, joka kuuluu tämän työn rajauksen ulkopuolelle.

Asetukset palvelimella

Tietoturvasyistä PowerShell-skriptien suoritus on käyttöjärjestelmän asennuksen jälkeen oletuksena estetty. Koska nyt luoduilla palvelimilla ajetaan PowerShell-skriptejä käsin ja ajastetusti, pitää niiden suoritus sallia.

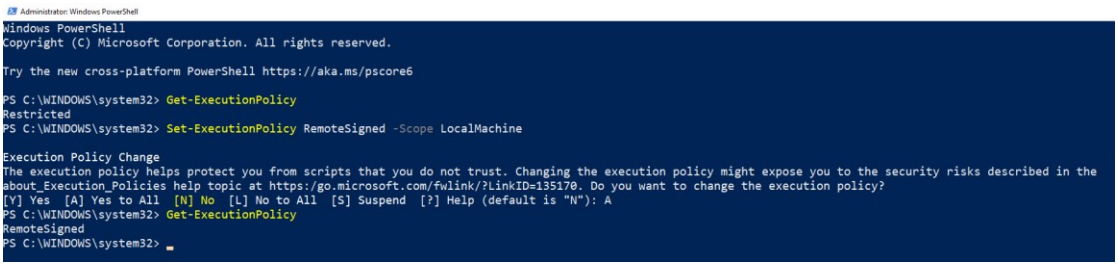
Suojauskäytännön voi kohdentaa kolmelle eri kohteelle (*engl. scope*). Näitä ovat Process, CurrentUser ja LocalMachine. Tässä tapauksessa määrityksen halutaan vaikuttavan sillä palvelimella, jossa skriptejä ajetaan, joten LocalMachine on sopiva valinta.

Suojauskäytäntöä voi muuttaa vain koneen järjestelmänvalvojan oikeuksin ja tämän vuoksi PowerShell-komentokehote pitää käynnistää korotetuilla oikeuksilla (*engl. elevated rights*).

Komento kokonaisuudessaan:

Set-ExecutionPolicy RemoteSigned -Scope LocalMachine

Komennon antamisen jälkeen muutos pitää vielä vahvistaa vastaamalla esitettyyn kysymykseen ”A”. Kuvassa 15 näkyy, kuinka ensin tarkistetaan sillä hetkellä aktiivisena oleva suojauskäytäntö, sen jälkeen vaihdetaan se halutuksi ja vahvistetaan valinta. Lisäksi tarkistetaan uudelleen, että muutos on voimassa.



Kuva 15. PowerShellin suojauskäytännön asettaminen

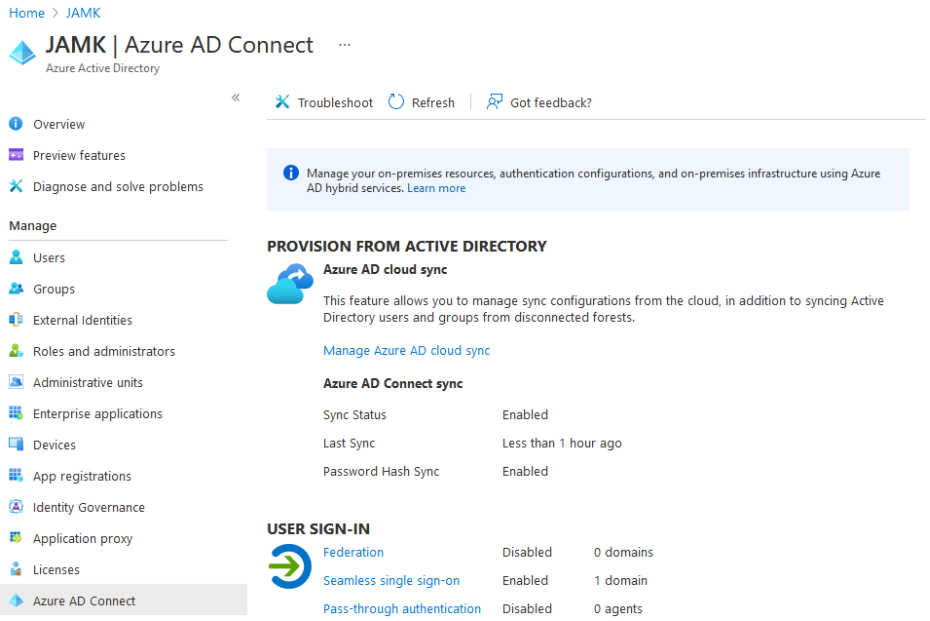
Näiden toimenpiteiden jälkeen palvelin on valmis toimimaan skriptien suoritusalustana. Tässä yhteydessä skriptejä ei allekirjoiteta, vaan palvelimen ja sen

sisällön suojaus toteutetaan muilla menetelmillä. Vaikka koodi olisi allekirjoitettu, se ei takaisi sisällön turvallisuutta, eli allekirjoituksella on mahdollista varmentaa vain skriptin tekijä ja suojata se muutoksilta, mutta tekijän luotettavuuteen se ei ota kantaa. (Holmes 2013, 515–519.)

Skriptien luomisesta ja niiden ajastamisesta lisää myöhemmin.

4.3 Office365:n lisenssien määrittely

Office365 tarjoaa useita palveluita, mutta paikallisen Active Directoryn käyttäjätunnuksia ei voi käyttää siellä suoraan, vaan ne pitää ensin synkronoida Office365:n taustalla toimivaan Azure AD -hakemistoon ja sen jälkeen lisätä niille myös tarvittavat lisenssit. Paikallisten AD-objektien synkronointi onnistuu Microsoftin tarjoaman Azure AD Connect -sovelluksen avulla, jonka määrittelyn jälkeinen tilanäkymä Azure-hallinnan kautta katsottuna on esitetty kuvassa 16.



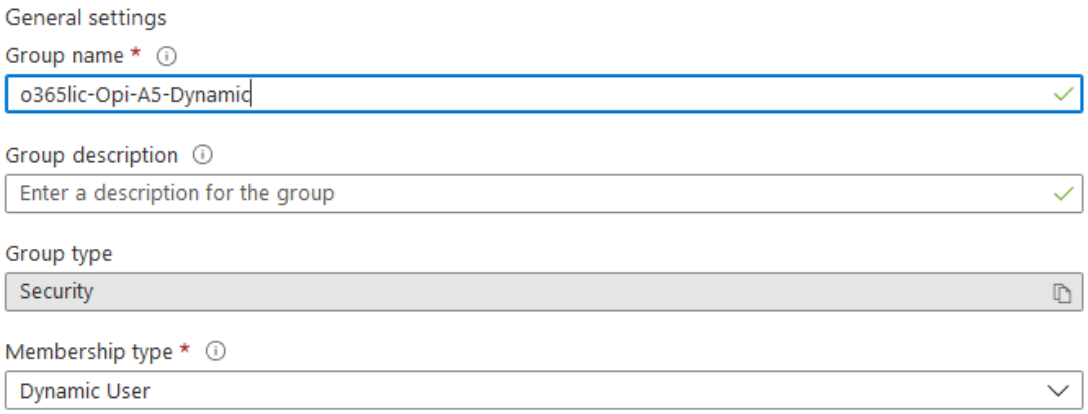
Kuva 16. Azure AD Connect Azuren hallintaliittymässä

Tämä ei sellaisenaan mahdollista vielä Office365-palveluiden käyttöä, koska tunnuksille pitää määritellä myös sopivat lisenssit. Oppilaitosorganisaatiossa tunnuksia tulee ja menee päivittäin kymmeniä ja lukuvuosien alussa jopa tuhansia päivässä, joten työmäärä on manuaalisena sietämätön. Tässä avuksi astuu automatiikka.

Aiemmin lisenssit piti määritellä jokaiselle tunnukselle yksitellen joko manuaalisesti tai skriptattuna, mutta nykyään on mahdollista hyödyntää Azuren ryhmälisensointia (*engl. group based licensing*), jonka avulla lisenssit saadaan määriteltyä tietylle käyttäjäryhmälle automaattisesti.

Tämä edellyttää ensin Azureen luotavaa ryhmäobjektia ja käyttäjien lisäämistä siihen. Käyttäjiä ei ymmärrettävästi haluta lisätä ryhmään käsityönä, joten ryhmästä tehdään dynaaminen ja määritellään sen asetuksissa, että kaikki tietyt ehdot täyttävät käyttäjätunnukset lisätään siihen automaattisesti.

Kuvassa 17 näkyy kuinka ryhmän asetuksissa jäsenyystyyppi (*engl. membership type*) on määritelty dynaamiseksi (engl. dynamic user).

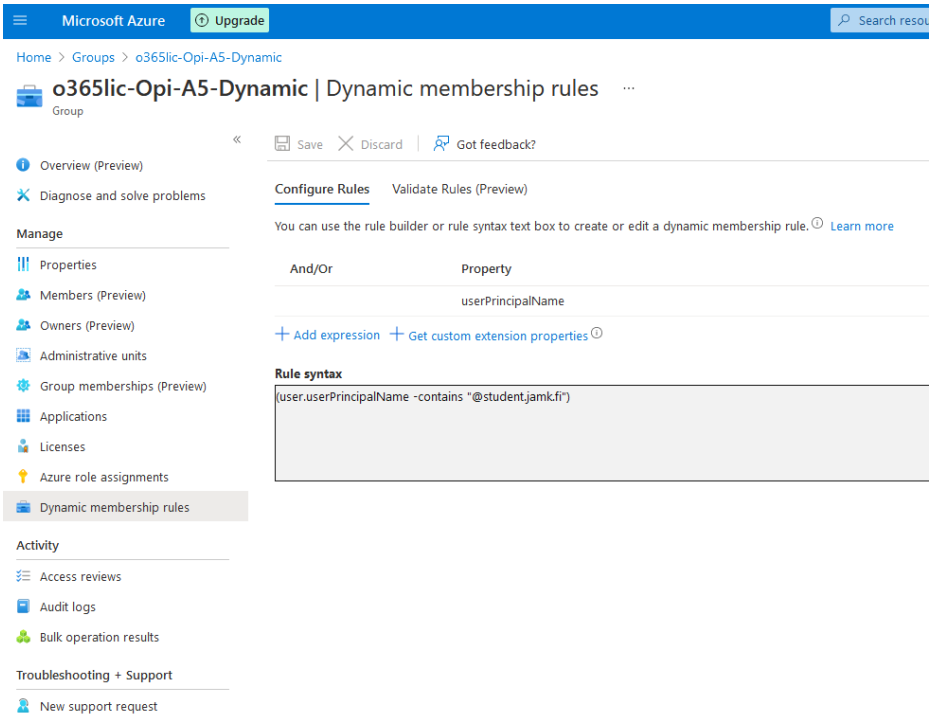


Kuva 17. Azuren dynaaminen ryhmä

Tässä esimerkissä on selkeyden vuoksi mainittu ryhmän tyyppi myös ryhmän nimessä (engl. group name), mutta sillä ei kuitenkaan ole vaikutusta sen tekniseen toimintaan. Vain ryhmän tyyppi ja sen sisältämä sääntörakenne ratkaisee.

Kuvassa 18 on esitetty aiemmin luotu dynaaminen ryhmä ja siihen vaikuttava sääntö:

(user.userPrincipalName -contains "@student.jamk.fi")

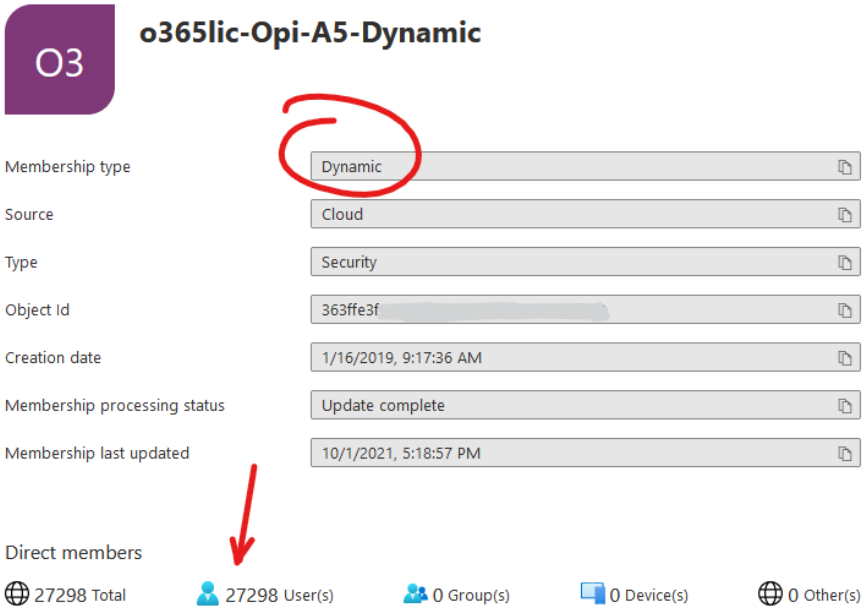


Kuva 18. Dynaamisen ryhmän määrittely

Esimerkissä ryhmään lisätään kaikki käyttäjäobjektit, joilla on tietyllä tavalla muotoiltu UPN-attribuutti (*userPrincipalName*). Kyse on tässä tapauksessa opiskelijoiden lisenssejä hallitsevasta ryhmästä, joten sääntö lisää ryhmään opiskelijatunnukset UPN:n sisältämän sähköpostiosoitteen domain-osan perusteella, tässä tapauksessa siis vaikuttava osa on ”@student.jamk.fi”. Henkilökunnan tunnuksissa käytetään eri lisenssejä, joten niitä hallitaan erillisen ryhmän kautta, mutta idea on sama.

Säännöllä toteutettu erottelu on tässä tapauksessa helppoa, koska henkilökunnalla ja opiskelijoilla on tunnuksissa käytössä eri domain, eli toimialue. Tarvittaessa säännöt voisivat olla paljon monimutkaisempiakin.

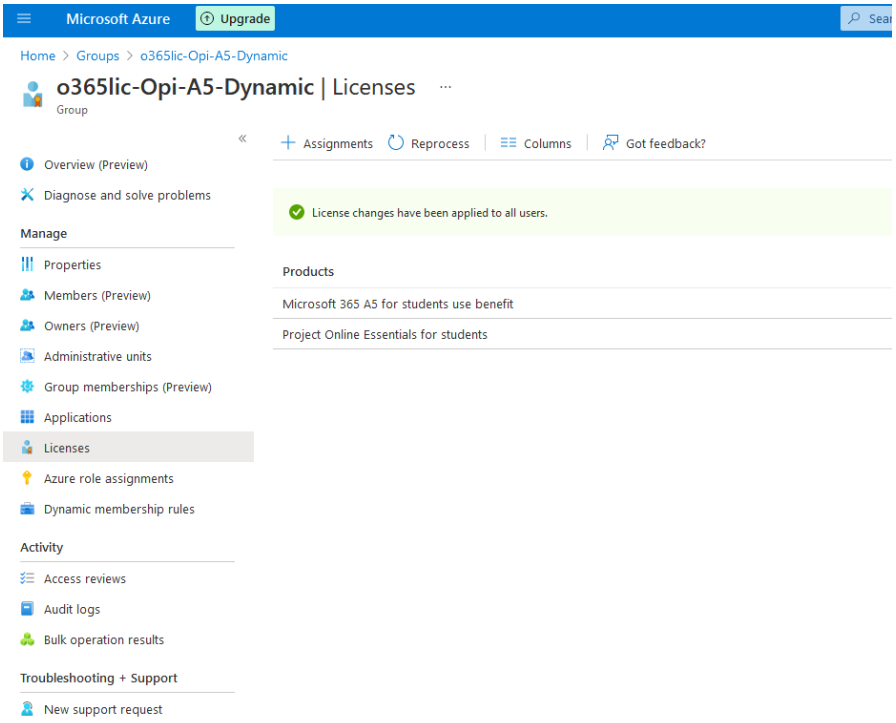
Kuvassa 19 näkyy Azure Active Directoryyn luodun dynaamisen ryhmäobjektin tiivistelmäsivu ja myös automaattisesti siihen lisättyjen jäsenten kokonaismäärä.



Kuva 19. Ryhmän tiivistelmäsivu Azuressa

Tiivistelmäsivu sisältää myös muita tietoja, joita on mahdollista hyödyntää vianselvityksessä. Tällaisia tietoja ovat mm. ryhmäjäsenyyden prosessoinnin vaihe (engl. membership processing status), sekä ryhmän viimeisimmän päivityksen aikaleima (engl. membership last updated).

Kuvassa 20 on esitetty seuraava vaihe, eli dynaamisen ryhmän asetuksissa sen sisältämiin tunnuksiin vaikuttavien lisenssien määrittely (engl. assignments). Lisenssejä on tässä tapauksessa lisätty kaksi.



Kuva 20. Lisenssien määrittely

Näiden toimenpiteiden jälkeen opiskelijatunnukset synkronoituvat paikallisesta aktiivihakemistosta Azureen ja saavat siellä käyttöönsä tarvittavat lisenssit.

Lopputuloksena henkilöstön ei tarvitse huolehtia lisenssien asettamisesta lainkaan, kun automatiikka hoitaa sen. Prosessi on siis siirtynyt Institute for Robotic Process Automationin (2015) kuvaamissa automaatiotasoissa manuaalisesta tasosta skriptitasolle. Tavallaan ollaan jo orkestrointitasolla, sillä tunnukset siirtyvät automaattisesti opiskelijahallintojärjestelmästä identiteetinhallintajärjestelmän kautta aktiivihakemistoon ja sieltä edelleen Azure AD Connectin avulla Azureen, jossa varsinainen lisenssien asettaminen tapahtuu.

4.4 Jaettujen postilaatikoiden luonti ja oikeushallinta

Organisaatiossa on jo entuudestaan käytössä NetIQ Identity Manager, jonka itsepalveluliittymän kautta käyttäjät voivat suorittaa omatoimisesti joitakin työnkulkuja, jotka sitten taas käynnistävät taustalla muita toimenpiteitä. Jo aiemmin käyttäjille on tarjottu mahdollisuus anoa sen kautta jaetun postilaatikon luontia tai oikeuksia aiemmin luotuun postilaatikkoon. Tästä eteenpäin prosessi on kuitenkin ollut manuaalinen ja sen vuoksi hidas ja epäluotettava.

Nyt toteutettu automaatio toimii seuraavalla periaatteella:

1.) Identity Manager (myöhemmin IDM) tuottaa eDirectoryyn postilaatikkoa vastaavan ryhmäobjektin ja lisää sinne ne käyttäjät, joilla tulee olla oikeus kuhunkin postilaatikkoon. IDM pitää myös huolen, että käyttäjät poistuvat ryhmästä, kun tarve lakkaa. Kullakin postilaatikolla on yksi tai useampia omistajia, jotka voivat tehdä muutoksia jäseniin IDM:n itsepalveluliittymän kautta.
2.) IDM luo ja synkronoi vastaavan ryhmän Active Directoryyn, josta se synkronoidaan edelleen Azure Active Directoryyn.
3.) Automatisointipalvelimella toimiva skripti käy läpi hakemistopalvelun tietyssä organisaatioyksikössä olevat ryhmäobjektit ja vertaa niitä jo luotuihin postilaatikoihin.
    a. Jos postilaatikko löytyy, skripti tarkistaa ryhmän jäsenet, vertaa postilaatikossa entuudestaan olevia oikeuksia ryhmään ja tekee tarvittavat oikeusmuutokset.
    b. Jos ryhmää vastaavaa postilaatikkoa ei löydy, skripti luo sen ja asettaa oikeudet jäsenille.
4.) IDM ilmoittaa postilaatikon omistajalle, kun postilaatikon käyttöaika on umpeutumassa ja jos omistaja ei jatka aikaa, merkitään postilaatikko lukittavaksi ja poistettavaksi. Poiston ei haluttu tapahtuvan automaattisesti, joten siitä lähetetään tässä vaiheessa pelkästään viesti ylläpidolle.

Skripti pyörii automatisointipalvelimella ajastetusti ja sen tunnustiedot on talletettu erilliseen tiedostoon salattuna. PowerShell-skriptien ajastamisesta Windowsin tehtävien ajastuksen (*engl. task scheduler*) avulla on kerrottu tarkemmin luvussa 4.7.

Skripti kirjoittaa vianselvitystä varten myös yksinkertaista lokitiedostoa palvelimelle. Skriptin ohjelmakoodi on olennaisilta osin mukana liitteessä 1. Liitteestä on poistettu tietoturvasyistä kaikki tunnustiedot, mutta muuten ohjelmakoodin toimintalogiikkaan ei ole koskettu.

Lähtötilanteessa jaettujen postilaatikoiden luonti oli käyttäjältä tulleen syötteen jälkeen täysin manuaalinen. Institute for Robotic Process Automationin (2015) kuvaamissa automaatiotasoissa tämä prosessi sijoittuu toteutuksen jälkeen orkestrointitasolle (anomus, hyväksyntä, postilaatikon luonti, oikeuksien asettaminen, kuittausviesti).

4.5 Jaettujen hakemistojen luonti ja oikeushallinta

Microsoft Teams ja monet muut nykyaikaiset pilvipohjaiset ryhmätyöskentelyvälineet ovat vähentäneet jaettujen verkkolevyjen tarvetta, mutta palvelimelta käyttäjille tarjottaville perinteisille verkkolevyille tuntuu olevan edelleen kysyntää. Toimeksiantajalla näiden hakemistojen luonti ja niiden oikeusmäärittelyt ovat hoituneet erikseen helpdesk-järjestelmän kautta pyydettäessä, mutta niiden toteutus on vaatinut manuaalista työtä, eli ensin on pitänyt luoda oikeusryhmä ja lisätä sinne käyttäjät. Sen jälkeen on pitänyt luoda hakemisto verkkolevylle ja liittää aiemmin luotu oikeusryhmä siihen.

Automatisoinnin avulla tätä prosessia muutettiin niin, että käyttäjä lähettää pyynnön hakemiston luomisesta ja sen hyväksynnän jälkeen riittää, kun paikalliseen aktiivihakemistoon luodaan haluttu oikeusryhmä. Tarkalleen ottaen ryhmiä luodaan kaksi, joista toinen on pelkkiä lukuoikeuksia varten ja toinen myös kirjoitusoikeuksille. Jos tarvetta on vain toiselle ryhmälle, ei toista ole pakko luoda. Institute for Robotic Process Automationin (2015) kuvaamissa automaatiotasoissa prosessi sijoittuu toteutuksen jälkeen skriptitasolle, mutta se on myöhemmin melko helppo jalostaa edelleen orkestrointitasolle lisäämällä ensimmäiseen vaiheeseen työnkulku samaan tapaan kuin jaettujen postilaatikoiden tapauksessa.

Automatiikka seuraa hakemistopalvelussa määritellyn organisaatioyksikön tapahtumia ja havaitessaan siellä aikaleiman perusteella uuden ryhmäobjektin tai muutoksia aiemmin käsiteltyyn ryhmään, se luo sitä vastaavan hakemiston verkkolevylle ja asettaa oikeudet kohdalleen. Hakemistolle asetetaan samalla myös kokorajoitus (*engl. quota*) sen tiedostopalvelimen resurssienhallintapalvelun (*engl. File Server Resource Manager*) avulla, joka kyseisen tallennustilan tarjoaa. Kokorajoituksen voisi asettaa myös PowerShell-komennoilla, mutta testauksen jälkeen resurssienhallintapalvelu osoittautui tässä tarpeessa niin käteväksi jatkossa yksittäisten hakemistojen kokorajoituksiin tehtäviä muutoksia ajatellen, että se jäi tuotantoon.

Automatiikka olisi ollut mahdollista rakentaa myös niin, että käyttäjän lähettämän lomakkeen kautta pelkkä hyväksymistoimenpide riittäisi ryhmäobjektin automaattiseen luomiseen, mutta tässä vaiheessa se ei ollut tarpeen.

Skriptien ajastuksesta on kerrottu tarkemmin luvussa 4.7. Kuvassa 21 näkyy skriptin varsinainen ohjelmakoodi.

```
#---------------------
#
# Muuttujat
#
#---------------------

# Ryhmän muutosten "tarkkailuaika"
$When = ((Get-Date).AddMinutes(-5).AddSeconds(-15))

# Ryhmät
$groups = Get-ADGroup -Filter { whenChanged -ge $When } -Properties whenChanged -SearchBase "OU=FoldersHEN,OU=Groups,DC=ad,DC=jamk,DC=fi" | select -ExpandProperty name

# Login sijainti
$Logfile = "log\$(get-date -Format yyyy-MM-dd).log"

#---------------------
#
# Funktiot
#
#---------------------

# Login kirjoitus
Function LogWrite
{
    Param ([string]$logstring)
    $Logfiletime = get-date -Format HH:mm
    $content = $Logfiletime + " " + $logstring
    Add-content $Logfile -value $content
}

#---------------------
#
# Käsittely
#
#---------------------


foreach ($group in $groups)
{

    # Lopetetaan, jos ryhmän nimessä ei ole oikeaa alkua tai päätettä
    if($group -notlike "FOLDER_*") { LogWrite "Ryhmän nimestä puuttuu FOLDER_! $group";continue}
    if($group -notlike "*-RW" -and $group -notlike "*-RO") { LogWrite "Ryhmän nimestä puuttuu -RW tai -RO! $group";continue}

        # Haetaan ryhmän nimi poistamalla siitä alku- ja loppuliite
        $groupshort = $group -replace "FOLDER_", "" -replace "-RW", "" -replace "-RO", ""
        $dir = "\\ad\fs\shared1\HEN\$groupshort"

        # Jos hakemistoa ei ole, luodaan se ja määritetään rajoitukseksi 2 GB
        if (!(Test-Path -Path $dir))
        {
                New-Item -ItemType directory -Path $dir
                sleep -Seconds 5
                #Invoke-Command -ComputerName fileshare2 -ScriptBlock { new-FsrmQuota -Path "D:\shared1\hen\$using:groupshort" -Template "2 GB Limit" }
                #Set-FsrmQuota -Path $dir -Description "limit usage to 2 GB" -Size 2GB
                LogWrite "Luotu hakemisto $groupshort"
        }

        # Ryhmän nimessä -RW, annetaan ryhmälle read/write oikeudet hakemistoon. Hakemiston poisto/siirto-oikeuksia ei anneta
        if ($group -like "*-RW")
        {
                $acl = Get-Acl $dir
                $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("AD\$group", "ReadAndExecute,Write,DeleteSubdirectoriesAndFiles", "ContainerInherit,ObjectInherit", "None", "Allow")
                #$AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("AD\$group", "Modify", "Allow")
                $acl.SetAccessRule($AccessRule)
                $acl | Set-Acl $dir
                LogWrite "Annettu RW oikeudet ryhmälle $group"
        }

        # Ryhmän nimessä -RO, annetaan ryhmälle read-only oikeudet
        if ($group -like "*-RO")
        {
                $acl = Get-Acl $dir
                $AccessRule = New-Object System.Security.AccessControl.FileSystemAccessRule("AD\$group", "ReadAndExecute", "ContainerInherit,ObjectInherit", "None", "Allow")
                $acl.SetAccessRule($AccessRule)
                $acl | Set-Acl $dir
                LogWrite "Annettu RO oikeudet ryhmälle $group"
        }
        $dir = ""
        $groupshort = ""
}
```

Kuva 21. Ryhmähakemistojen luonti PowerShellillä

Kuvan 21 skriptin ohjelmakoodista on poistettu kaikki tunnustiedot tietoturvasyistä, mutta ohjelman muu toimintalogiikka on koskematon.

4.6 Tiedostosiirtoja palvelinsovellusten kesken

Toimeksiantajan taloushallinnon järjestelmistä osa on hankittu ulkoiselta kumppanilta ja osa toimii organisaation omissa datakeskuksissa. Kaikkien näiden välillä on tarvetta siirtää tiedostoja, joita toiset järjestelmät tuottavat ja vastaavasti toiset lukevat sisään omilla sisäisillä prosesseillaan. Näitä siirtoja on useita ja vaikka niissä onkin jokaisessa omat vaatimuksensa ja erikoisuutensa, ei niitä kaikkia ole mielekästä käydä läpi tässä dokumentissa. Valitsin siksi tähän esimerkiksi yhden skriptin, jossa tehdään useampia asioita.

Tässä esimerkkitapauksessa on tarve noutaa kolme kertaa vuorokaudessa, kaksi etukäteen tiedossa olevaa ja aina samalla tavalla nimettyä csv-tiedostoa toimittajan palvelimelta sftp-protokollaa käyttäen. Noudon jälkeen toinen tiedostoista jaetaan kahteen osaan sisällön perusteella ja käsittelyn jälkeen kaikki kolme tiedostoa talletetaan kohdehakemistoon. Välissä tiedostoista otetaan varmuuskopiot ja lopuksi lopetetaan avattu sessio palvelinten välillä ja siivotaan vanhat varmuuskopiot. Ilman toteutettua automatisointia kaikki työvaiheet olisivat täysin manuaalisia ja veisivät paljon työntekijöiden aikaa. Institute for Robotic Process Automationin (2015) kuvaamissa automaatiotasoissa prosessi sijoittuu toteutuksen jälkeen skriptitasolle, eikä se edellytä työntekijöiltä tai ylläpidolta muita toimenpiteitä.

Skriptin ohjelmakoodi löytyy liitteestä 2. Siitä on tietoturvasyistä poistettu tunnustiedot, tiedostojen nimet ja palvelinten osoitteet, mutta muu toimintalogiikka on mukana. Skriptien ajastuksesta on kerrottu tarkemmin luvussa 4.7.

4.7 Valokuvan julkaisulupa

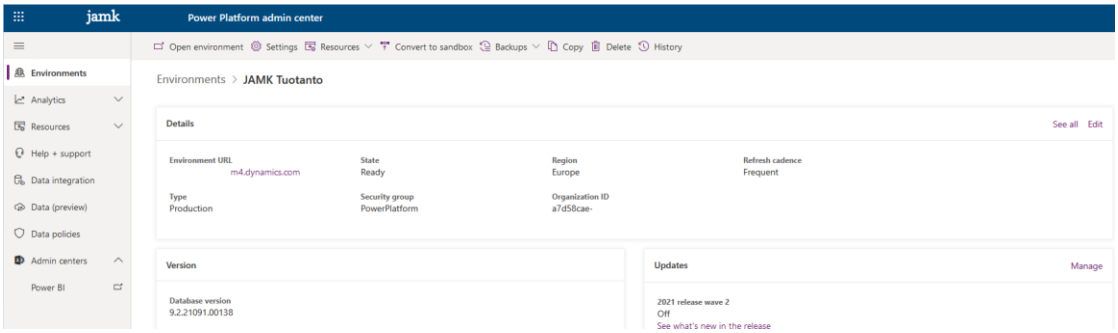
Tämän prosessin automatisoinnissa vaatimuksena oli esittää käyttäjälle lomake, jonka avulla voi antaa suostumuksen kuvan julkaisulle organisaation julkisissa palveluissa.

Toteutustavaksi valittiin Office365:n Forms, jolla voidaan tuottaa tarvittava lomake, sekä sen taustalle Power Automate, joka saa herätteen lomakkeelta ja toteuttaa varsinaisen työnkulun. Vastauksen jälkeen käyttäjän tunnus liitetään taustalla Azure AD -ryhmään, jonka perusteella julkaisujärjestelmä tietää, että kuvalla on julkaisulupa, eli jos käyttäjä ei kuulu määriteltyyn ryhmään, ei kuvaa saa julkaista ja päinvastoin.

Koska tällaisia keskitettyjä toiminnallisuuksia ei ole järkevää henkilöidä kenenkään työntekijän henkilökohtaiseen tunnukseen, luotiin ensimmäisenä tätä varten erillinen Azure AD -tunnus ja sille tarvittavat lisenssit ja oikeudet. Tällä tunnuksella muodostettiin Forms-lomake ja työnkulku Power Automateen.

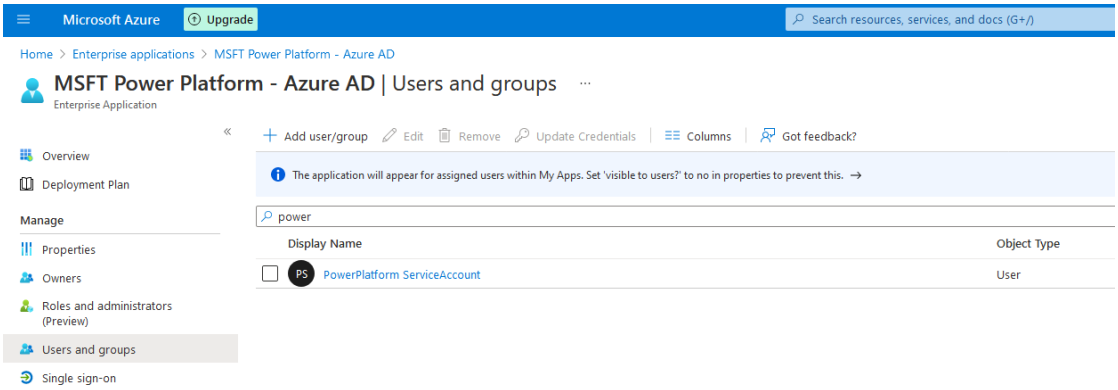
Azure AD:ssa voi luoda ympäristöjä (*engl. environment*), joiden avulla on mahdollista tallettaa, hallita ja jakaa yrityksen bisneslogiikkaan liittyviä palveluita. Niiden avulla myös oikeuksien hallinta ja käytön seuranta on näissä palveluissa hieman helpompaa. Tyypillisesti esimerkiksi testaamista ja tuotantoa varten luodaan omat ympäristöt. (Microsoft 2021a.)

Kuvassa 22 on esitetty Azuren Power Platform -palveluun luotavia automatisointiprosesseja varten perustettu tuotantoympäristö, jonne varsinainen työnkulku liitetään.



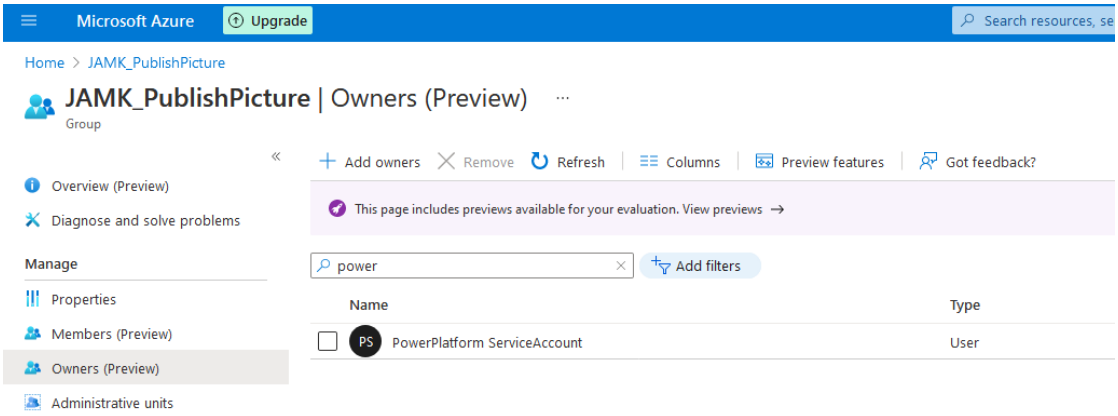
Kuva 22. Power Platformin tuotantoympäristön koontisivu

Tunnuksen luomisen jälkeen se pitää lisätä käyttäjäksi luotuun ympäristöön. Lisäksi tunnus pitää lisätä Azuren ”MSFT Power Platform – Azure AD Enterprise Applicationiin”, kuten kuvassa 23 on tehty.



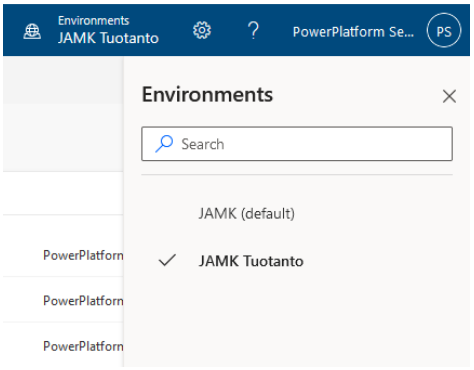
Kuva 23. Käyttäjän lisääminen Power Platform -käyttäjäksi

Tunnukselle pitää myös antaa oikeudet muokata ryhmäobjektia, jonne lomakkeen kautta kuvansa julkaisuluvan anteiden tunnukset tallennetaan. Kuvassa 24 näkyy, että tässä tapauksessa oikeus ryhmäobjektiin on annettu lisäämällä huoltotunnus ryhmän omistajaksi (*engl. owner*).



Kuva 24. Huoltotunnuksen lisääminen Azure-ryhmän omistajaksi.

Kun työnkulkua aletaan luomaan, pitää muistaa valita se ympäristö, jonne työnkulku halutaan liittää. Kuvassa 25 näkyy, että on valittu nimenomaan aiemmin tätä tarvetta varten luotu ympäristö.



Kuva 25. Power Automaten työnkulun ympäristön valinta.

Työnkulun muodostaminen Power Automatessa tapahtuu graafisen käyttöliittymän kautta. Työnkulkuun lisätään eri yhdistimiä ja niiden aktiviteetteja. Aktiviteetit voivat olla parhaimmillaan hyvinkin valmiita käyttöön sellaisenaan, kun taas osa niistä edellyttää manuaalista ohjelmointia. Edistyneemmissä toiminnoissa saatetaan tarvita ohjelmointitaitoja ja taustapalveluiden toiminnan ymmärtämistä melko kattavastikin.

Varsinaisen työnkulun vaiheet Power Automatessa ovat seuraavat:

1. When a new response is submitted
    - Asetetaan työnkulun käynnistysehdoksi uusi vastaus valitulta lomakkeelta
2. Get response details

- Noudetaan käyttäjän antamat vastaukset lomakkeelta

3. Get user profile
    - Noudetaan lomakkeen lähettäneen käyttäjän profiilitiedot myöhempää käyttöä varten
4. Check for group membership
    - Tarkistus, kuuluuko käyttäjä määriteltyyn ryhmään
5. Initialize variable
    - Talletetaan käyttäjän valinta *varUserResponse*-muuttujaan myöhempää käyttöä varten
6. Response condition check

    - *If varUserResponse = true*

        - Käyttäjä on valinnut lomakkeella ”Allow”
        - Tarkistetaan aiemmin tehdystä aktiviteetista, kuuluiko käyttäjä ryhmään entuudestaan. Tarkistus kohdistetaan aiemmin kohdassa 4 luotuun aktiviteettiin komennolla

          length(outputs('Check_group_membership_(V2)')?['body/value'])

          Jos komennon palauttama arvo on 0, käyttäjä ei kuulunut ryhmään, joten lisätään käyttäjä sinne ja lähetetään infoviesti, muuten lähetetään pelkästään info ilman lisätoimenpiteitä

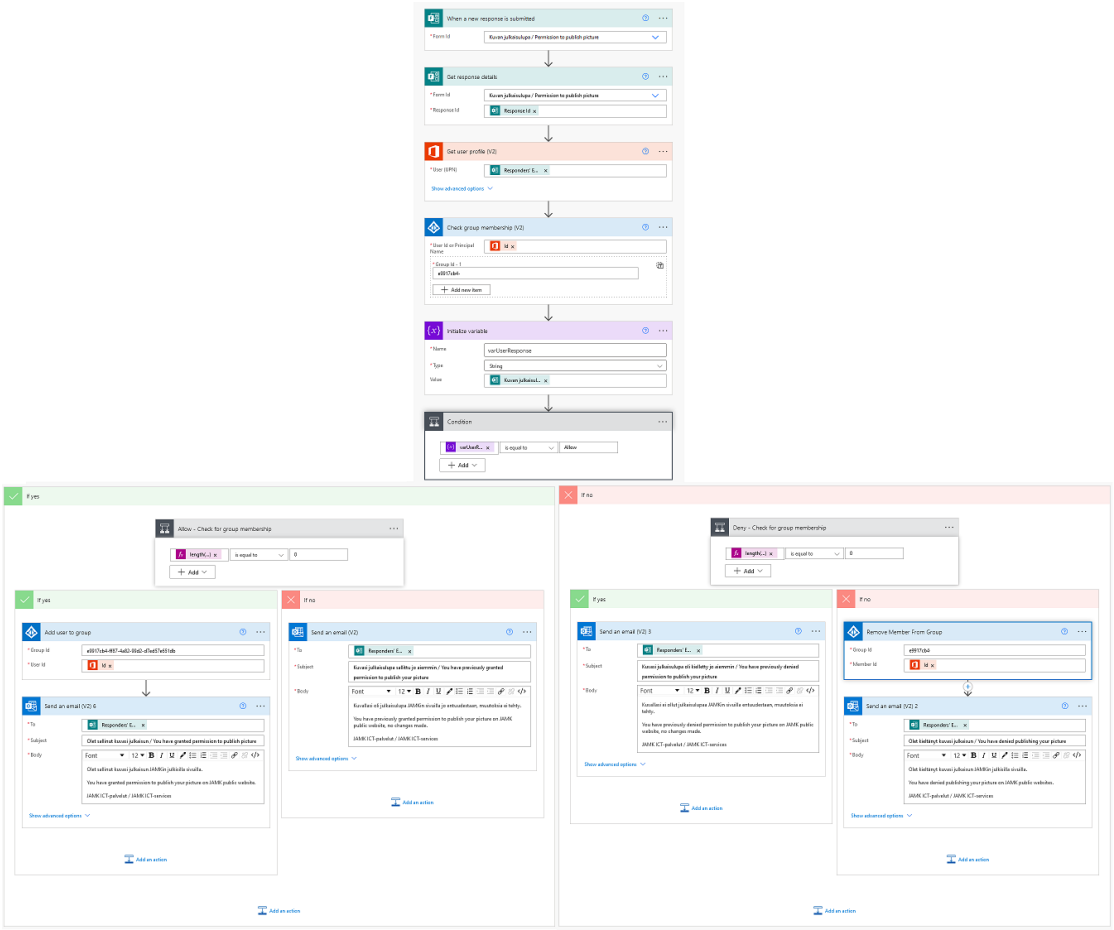
    - *If varUserResponse = false*

        - Käyttäjä on valinnut lomakkeella jotain muuta kuin ”Allow”, tässä tapauksessa siis valinta on ollut ”Deny”
        - Tarkistetaan tehdystä aktiviteetista, kuuluiko käyttäjä ryhmään entuudestaan. Tarkistus kohdistetaan aiemmin kohdassa 4 luotuun aktiviteettiin komennolla

          length(outputs('Check_group_membership_(V2)')?['body/value'])

          Jos komennon palauttama arvo on 0, käyttäjä ei kuulu ryhmään, joten lähetetään pelkkä infoviesti ilman muita toimenpiteitä, muuten poistetaan lisäksi käyttäjä ryhmästä.

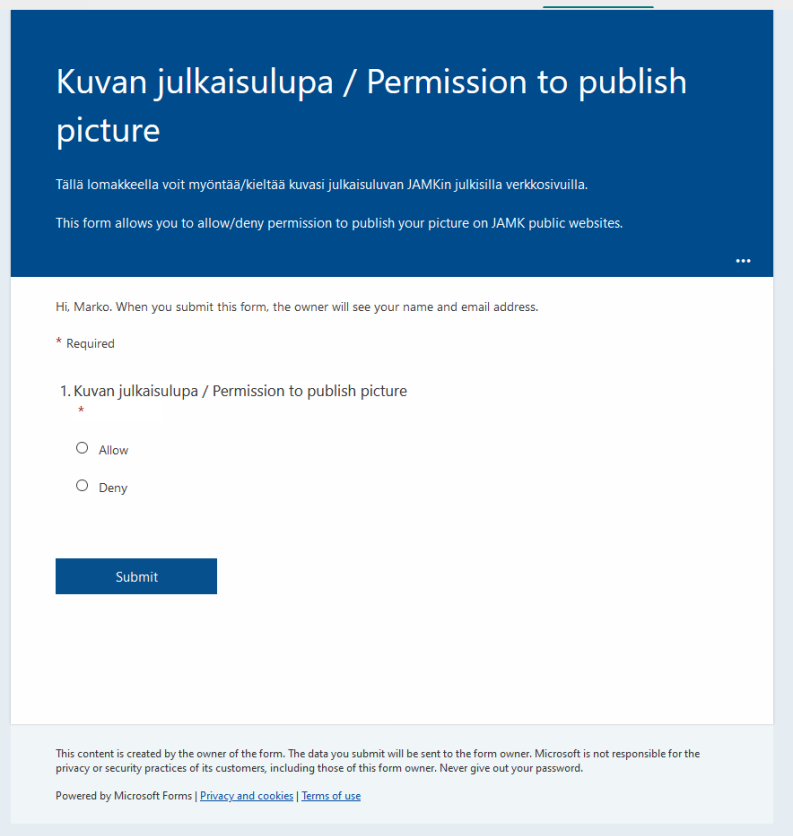
Kuvassa 26 on esitetty tämän prosessin automatisoinnin yhteydessä luotu työnkulku Power Automaten graafisessa hallintaliittymässä.



Kuva 26. Kuvan julkaisuluvan työnkulku Power Automatessa

Kuvassa 27 näkyvä loppukäyttäjän lomake on selkeä, eli siinä kysytään kirjautumisen jälkeen pelkästään julkaisulupaa kahdella vaihtoehdolla. Jos käyttäjä hyväksyy, lisätään tunnus taustalla haluttuun Azure AD -ryhmään ja päinvastoin.

Lomakkeen lähettäjä saa sähköpostitse myös vahvistusviestin toimenpiteestä. Tarvittaessa työnkulkuun voisi lisätä myös ”teknisen osoitteen”, jonne vahvistus lähetetään, jolloin sieltä olisi helposti tarkistettavissa missä vaiheessa käyttäjä on antanut tai kieltänyt luvan. Tässä yhteydessä sitä ei kuitenkaan tarvittu.



Kuva 27. Loppukäyttäjän näkymä lomakkeella

Tässä kuvattu prosessi kuvan julkaisuluvalle on hyvin yksinkertainen, mutta se vastaa tarvetta. Toiminnallisuutta on mahdollista laajentaa ja hyödyntää myöhemmin myös muissa vastaavissa prosesseissa.

Toimenpide on hoidettu aiemmin manuaalisesti kysymällä käyttäjiltä lupa erikseen sähköpostitse tai muulla vastaavalla tavalla. Institute for Robotic Process Automationin (2015) kuvaamissa automaatiotasoissa prosessi sijoittuu toteutuksen jälkeen orkestrointitasolle.

4.8 Skriptien ajastaminen

PowerShell-skriptien suoritus on mahdollista ajastaa Windows-alustalla käyttöjärjestelmän sisäänrakennetulla ajastetut tehtävät (*engl. task scheduler*) -ominaisuudella. Tarvittaessa ajastettujen tehtävien luominen ja muokkaaminen onnistuu myös suoraan PowerShellin kautta, mutta lopputulos on sama, kuin jos tehtävä luodaan graafisen käyttöliittymän kautta. (Melnick 2021.)

PowerShellin kautta toimittaessa tehtävien ajastaminen etenee vaiheittain. Seuraavassa luodaan esimerkin vuoksi PowerShellillä tehtävä nimeltä ”calculator_example”, joka käynnistää calc.exe -sovelluksen joka päivä kello 12:00.

Ensimmäisenä on selkeintä luoda muuttujat tehtävän nimelle, toiminnolle, laukaisijalle ja kuvaukselle. Kuvaus ei ole pakollinen tieto. Muuttujien nimillä ei ole toiminnallista merkitystä, mutta selkeyden vuoksi ne kannattaa nimetä kuvaavasti, tässä niiden niminä ovat ”taskName”, ”taskAction”, ”taskTrigger” ja ”taskDescription”.

Tehtävän nimi asetetaan taskName-muuttujaan tekstinä.

```
$taskName = “calculator_example”
```

Toiminnoksi taskAction-muuttujaan asetetaan calc.exe-sovelluksen käynnistys.

```
$taskAction = New-ScheduledTaskAction -Execute "calc.exe"
```

Laukaisin asetetaan taskTrigger-muuttujaan. Tässä esimerkissä laukaisin aktivoituu joka päivä klo 12:00. Ajastetuissa tehtävissä käytettäviä muita mahdollisia laukaisimia ovat:

- vain kerran (Once)
- viikoittain (Weekly)
- koneen käynnistyksessä (AtStartup)
- kirjauduttaessa (AtLogOn)

```
$taskTrigger = New-ScheduledTaskTrigger -Daily -At 12:00
```

Tehtävälle annetaan joku kuvausteksti. Tämä ei ole pakollinen parametri.

```
$taskDescription = “This example task opens calc.exe”
```

Koska Windows-palvelimilla on valmiina useita järjestelmän sisäisiä ajastettuja tehtäviä, on joskus selkeyden vuoksi järkevää sijoittaa omat tehtävät erilliseen kansioon. Tämä tapahtuu antamalla tehtävän rekisteröinnin yhteydessä parametri -TaskPath ja sille halutun kansion nimi.

```
$taskPath = “Test”
```

Lopuksi luodaan varsinainen tehtävä rekisteröimällä se edellisissä vaiheissa luotuja muuttujia apuna käyttäen.

```
Register-ScheduledTask `
   -TaskName $taskName `
   -Action $taskAction `
   -Trigger $taskTrigger `
   -Description $taskDescription `
   -TaskPath $taskPath
```

Kuvassa 28 näkyy PowerShell-konsoli, jossa on luotu uusi tehtävä edellisiä komentoja käyttäen.

```
PS C:\> $taskName = "calculator_example"
PS C:\> $taskAction = New-ScheduledTaskAction -Execute "calc.exe"
PS C:\> $taskTrigger = New-ScheduledTaskTrigger -Daily -At 12:00
PS C:\> $taskDescription = "This example task opens calc.exe"
PS C:\> $taskPath = "Test"
PS C:\> Register-ScheduledTask `
>>      -TaskName $taskName `
>>      -Action $taskAction `
>>      -Trigger $taskTrigger `
>>      -Description $taskDescription `
>>      -TaskPath $taskPath

TaskPath                                 TaskName                       State
--------                                 --------                       -----
\Test\                                   calculator_example             Ready
```

Kuva 28. Ajastetun tehtävän luonti PowerShellillä

Ajastettuja tehtäviä voi tarkastella PowerShellin kautta komennolla *Get-ScheduledTask*, joka palauttaa kaikki järjestelmässä olevat ajastetut tehtävät. Listauksen voi myös kohdistaa vain johonkin tiettyyn kansioon TaskPath-parametrilla.

```
Get-ScheduledTask -TaskPath *test
```

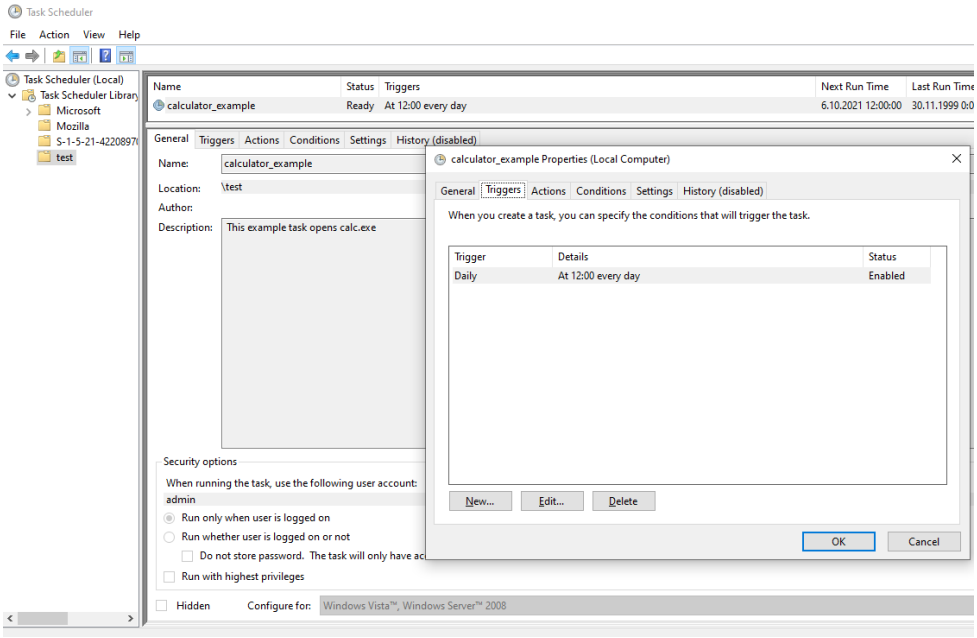
Yksittäisen tehtävän voi noutaa komennolla *Get-ScheduledTask*, jonka TaskName-parametrille annetaan halutun tehtävän nimi, tai sen osa (kuva 29).

*Get-ScheduledTask -TaskName calc**

```
PS C:\> Get-ScheduledTask -TaskName calc*

TaskPath                                 TaskName                 State
--------                                 --------                 -----
\test\                                   calculator_example       Ready
```

Kuva 29. Ajastetun tehtävän noutaminen PowerShellillä

Kuvassa 30 näkyy luotu tehtävä Windowsin graafisen käyttöliittymän kautta katsottuna.



Kuva 30. Ajastettu tehtävä graafisessa käyttöliittymässä

Kuten kuvasta näkyy, tällä tavalla toimittaessa tehtävä luodaan sillä käyttäjätunnuksella, jolla komento on annettu. Esimerkin tapauksessa tunnuksen nimi on ollut ”admin”. Jos tehtävän suoritus halutaan tehdä jollakin toisella tunnuksella, sen voi määritellä näin:

$taskPrincipal = New-ScheduledTaskPrincipal -UserId "tunnus" -LogonType ServiceAccount

Ja vastaavasti luotu muuttuja pitää lisätä mukaan komennon parametriksi.

-Principal $taskPrincipal

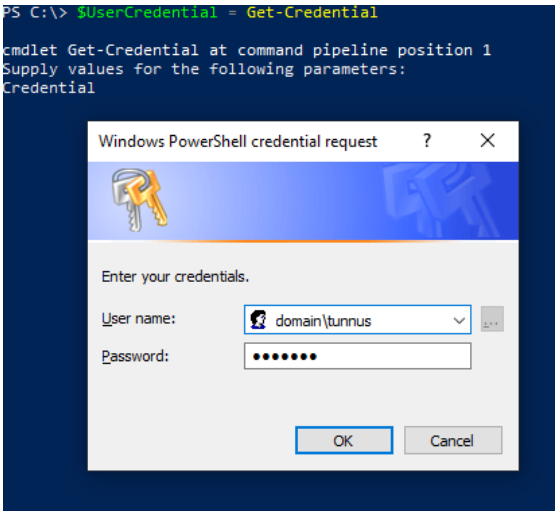
Tarpeeton tehtävä poistetaan komennolla *Unregister-ScheduledTask* ja sille annetaan parametriksi poistettavan tehtävän nimi.

Unregister-ScheduledTask calculator_example

Seuraavassa käydään läpi ajastetun tehtävän luominen PowerShell-esimerkkiskriptille, jota suoritetaan 15 minuutin välein. Tehtävä on määritelty käynnistymään admin-nimisellä tunnuksella, mutta skriptin sisällä käytetään erilliseen tiedostoon salattuna tallennettua tunnusta. Tässä esimerkissä molemmat tunnukset ovat samoja, mutta ne voisivat olla myös eri tunnuksia. Olennaista on luoda salattu salasanatiedosto sillä tunnuksella ja siinä koneessa, jolla ajastettu tehtävä käynnistetään.

Aloitetaan tehtävän luonti tallentamalla skriptissä käytettävä tunnus/salasanapari PowerShellissä System.Security.SecureString -tyyppiseksi objektiksi. Tässä vaiheessa käytettävän tunnuksen pitää olla sama kuin millä salasanatiedostoa on tarkoitus lukea myöhemmin. Kuvassa 31 näkyy komento suoritettuna konsolilla.

$UserCredential = Get-Credential

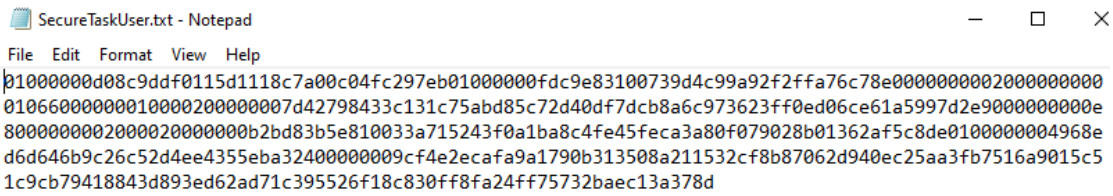


Kuva 31. Tunnuksen tallentaminen muuttujaan

Tämän jälkeen salasana talletetaan salattuna erilliseen tiedostoon.

$UserCredential.Password | ConvertFrom-SecureString | Out-File C:\Temp\SecureTaskUser.txt

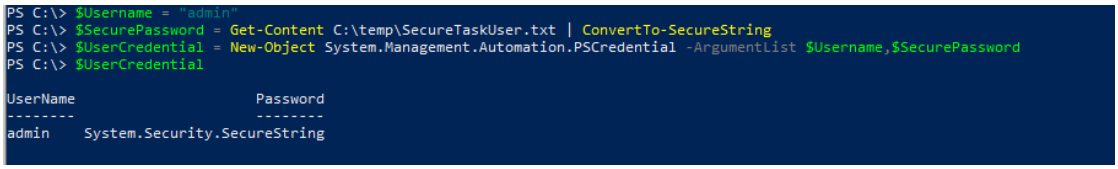
Kuvassa 32 tiedosto on esimerkin vuoksi avattu tavallisella tekstieditorilla ja kuten nähdään, tiedoston sisältö on tässä vaiheessa salattu.



Kuva 32. Salattu salasanatiedosto

Skriptissä salasanaa hyödynnetään lukemalla se muuttujaan SecureString-tyyppisenä ja muodostamalla sen avulla uusi PSCredential-objekti. Kuvassa 33 näkyy edellisessä vaiheessa luotu PSCredential-objekti konsolille tulostettuna ja kuten nähdään, salasana on SecureString-tyyppinen, eikä sitä voi lukea selväkielisenä.

$Username = "domain\tunnus"
$SecurePassword = Get-Content C:\Temp\SecureTaskUser.txt | ConvertTo-SecureString
$UserCredential = New-Object System.Management.Automation.PSCredential -ArgumentList $Username, $SecurePassword



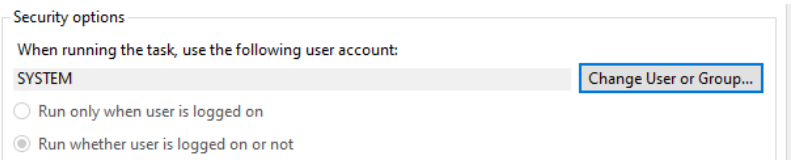
Kuva 33. Salatun tunnuksen lukeminen tiedostosta

Vaihtoehtoisesti skriptissä käytettävä tunnus voisi olla määriteltynä sen sisällä suoraan selväkielisenä tähän tapaan:

$Username = "domain\username"
$Password = "salasana123!" | ConvertTo-SecureString -AsPlainText -Force

$UserCredential = New-Object System.Management.Automation.PSCredential -ArgumentList $Username, $Password

Suoraan skriptiin talletettu salasana on kuitenkin tietoturvamielessä huono ratkaisu, eli salasana on aina syytä salata, mikäli vain mahdollista. Kolmas vaihtoehto olisi jättää salasana kokonaan pois ja käyttää palvelimen systeemitunnusta tehtävän käynnistämisessä, kuten kuvassa 34, mutta sitten palvelinobjektilla pitää olla oikeudet kaikkiin palveluihin mitä skriptissä käytetään. Tietoturvamielessä myös systeemioikeuksien käyttäminen voi olla ajastetuissa tehtävissä vaarallinen tapa, eli sellaisessa toteutuksessa tulee kiinnittää erityistä huomiota tietoturvaan kokonaisuutena, huomioiden palvelimen päivitykset ja käyttöoikeudet, sekä mahdollisesti myös skriptien digitaalinen allekirjoitus. Kokemuksen mukaan skriptejä suoritetaan organisaatioissa kaikilla mainituilla tavoilla ja niiden sekoituksina.



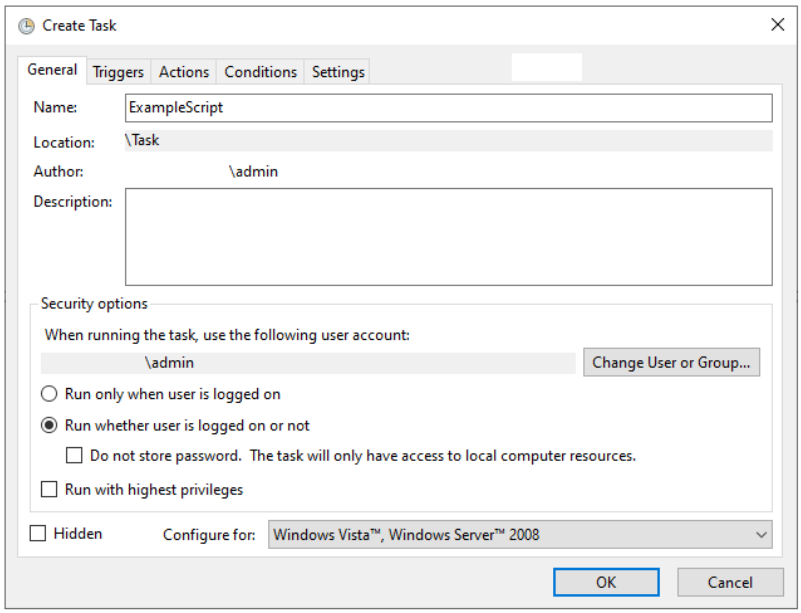
Kuva 34. Systeemitunnuksen käyttäminen ajastetussa tehtävässä

Kuvassa 35 näkyy tässä esimerkissä käytettävä skripti, joka lukee salasanan aiemmin luodusta tiedostosta. Skripti kirjoittaa lokitiedostoon kellonajan, tunnuksen ja salasanan uudelle riville joka suorituksella.

```
#----------------------------------------------------
#
# This is just an example Powershell script that reads secure
# password from file.
#
# Version: 0.1
#
#----------------------------------------------------

#-----------------
#
# Username
#
#-----------------
$Username = "admin"
$SecurePassword = Get-Content C:\Temp\SecureTaskUser.txt | ConvertTo-SecureString
$UserCredential = New-Object System.Management.Automation.PSCredential -ArgumentList $Username, $SecurePassword

#Log location and format
$Logfile = "exampleLog-$(get-date -Format yyyy-MM-dd).log"

#-----------------
#
# Functions
#
#-----------------

Function LogWrite
{
    Param ([string]$logstring)
    $Logfiletime = get-date -Format HH:mm
    $content = $Logfiletime + " " + $logstring
    Add-content $Logfile -value $content
}

#Write log
LogWrite "$Username was here using password $SecurePassword"
```

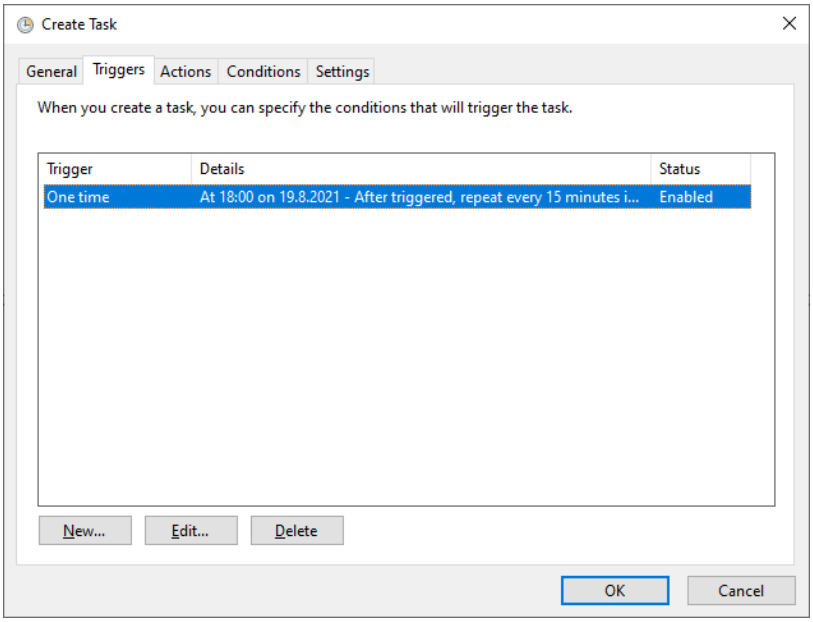
Kuva 35. Esimerkkiskripti salasanan lukemisen demoamiseksi

Luodaan seuraavaksi ajastettu tehtävä graafisen käyttöliittymän kautta. General-välilehdellä annetaan tehtävälle nimeksi *ExampleScript* ja määritellään se käynnistymään tunnuksella *admin*. Valitaan myös, että tehtävää ajetaan riippumatta siitä, onko käyttäjä kirjautuneena vai ei. General-välilehden asetukset on esitetty kuvassa 36.



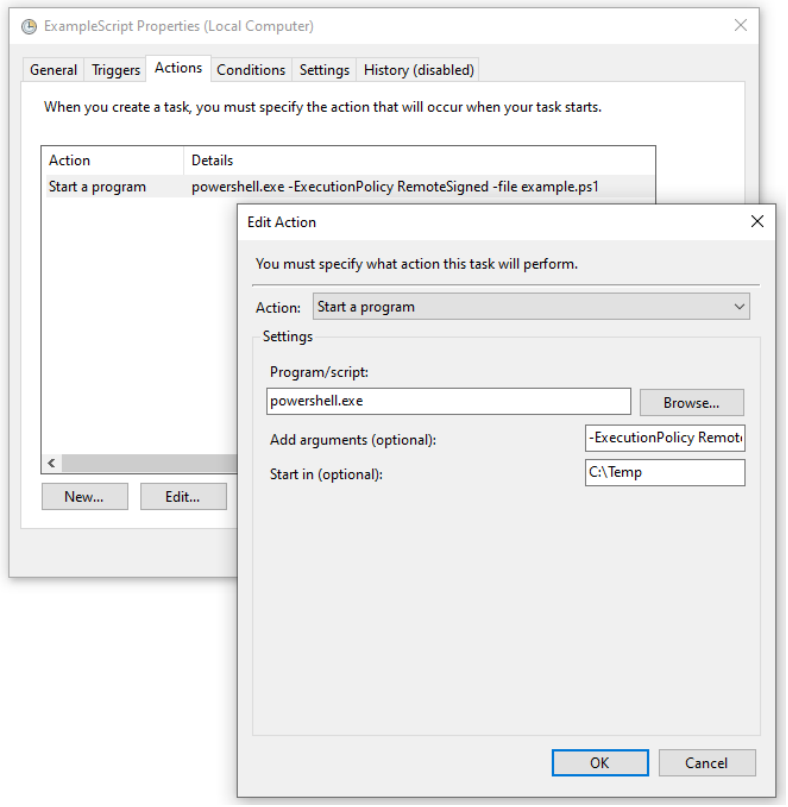
Kuva 36. Esimerkkitehtävän general-välilehti

Kuvassa 37 näkyy triggers-välilehti, jossa määritellään halutut laukaisimet tehtävälle. Tässä tapauksessa tehtävää halutaan suorittaa 15 minuutin välein, eli valitaan aloitushetki ja toistoväliksi ”repeat every 15 minutes”.



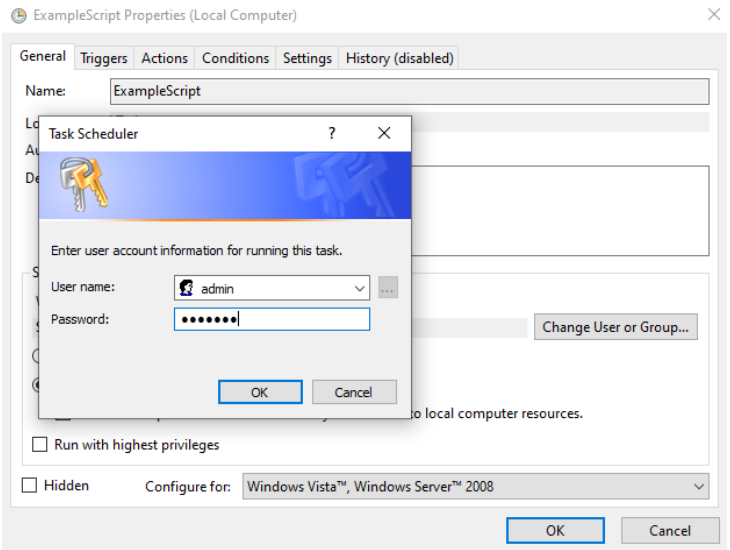
Kuva 37. Esimerkkitehtävän laukaisimen määritys

Kuvassa 38 näkyy tehtävän actions-välilehti ja sinne määriteltävät toiminnot. Toimintoja voisi olla useampiakin, mutta tässä tapauksessa halutaan käynnistää vain yksi toiminto, jossa suoritettava sovellus on ”powershell.exe”, arguments-kentässä parametreina ”-ExecutionPolicy RemoteSigned -file example.ps1” ja start in -kentässä se hakemisto, jossa skripti sijaitsee (ns. työhakemisto).



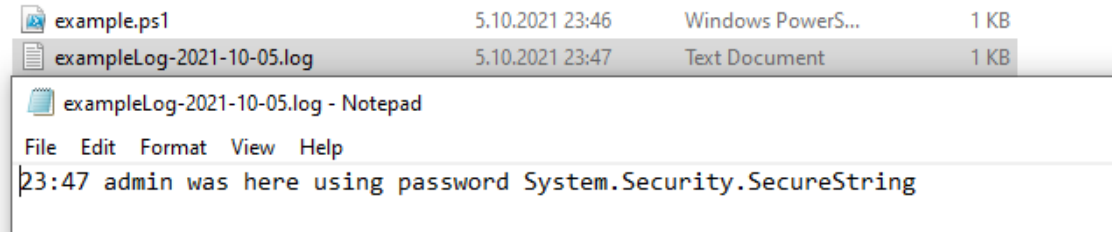
Kuva 38. Esimerkkitehtävän toimintojen määritys

Muita lisäasetuksia ei tässä vaiheessa tarvita. Viimeisenä vaiheena tehtävä tallennetaan, jolloin Windows kysyy sen tunnuksen salasanaa, joka on määritelty tehtävän asetuksissa. Tunnus tallennetaan salattuna myös tässä vaiheessa. Tämä vaihe on esitetty kuvassa 39.



Kuva 39. Esimerkkitehtävän tallentaminen

Kun tehtävä suoritetaan, se käynnistää skriptin, joka puolestaan lukee salasanan erillisestä tiedostosta ja kirjoittaa sen tunnuksen kanssa lokitiedostoon, kuten kuvassa 40 näkyy.



Kuva 40. Esimerkkitehtävän suoritus ja lokitiedosto

Skriptin suoritus onnistui, eli myös salasanan lukeminen ja sen käyttäminen onnistui. Tiedoston sisällöstä kuitenkin nähdään, että salasanan sisältö on edelleen salattu, jos sitä yritetään tulostaa.

5 PÄÄTÄNTÖ

PowerShellistä puhuttaessa on hieman harmaalla alueella, kutsutaanko sitä omaksi ohjelmointikieleksi vai pelkästään skriptauskieleksi, mutta joka tapauksessa se on äärimmäisen monipuolinen ja tehokas työkalu moneen tarpeeseen ja se osoittautui tässäkin yhteydessä toimivaksi valinnaksi skriptien toteutuksessa.

Toteutus onnistui hyvin ja automatisointeja tuotettiin kartoituksen jälkeen onnistuneesti useisiin järjestelmiin. Ne myös ne jäivät sellaisenaan tuotantoon. Toteutuksen aikana piti soveltaa vanhaa osaamista ja opetella uutta. Tavoitteet saavutettiin sekä opinnäytetyön osalta, että myös toimeksiantajan kanssa ennen toteutusta asetettujen tavoitteiden osalta.

Työssä toteutettuja PowerShell-skriptejä ajetaan ajastetusti eri palvelimilla ja ne toteuttavat vaadittuja toimenpiteitä. Osa skripteistä toteutettiin niin, että ne ottavat varmuuskopiot käsittelemistään aineistoista ja kirjoittavat myös lokia kaikista tehdyistä toimenpiteistä. Tämän todettiin helpottavan varsinkin ongelmatilanteiden vianselvityksessä.

Osa tuotetuista skripteistä oli järkevämpää jättää niiden luonteesta johtuen ylläpitäjien manuaalisesti käynnistettäväksi. Tällaisia toimenpiteitä ovat esimerkiksi sellaiset, joiden suoritus vaatii erityistä valvontaa tai joiden tarve on vain satunnaista, eikä niiden käynnistämiseksi välttämättä saada herätettä muualta. Nämä eivät olleet aina edes kovin monimutkaisia tai laajoja toteuttaa, mutta kun niitä tuotettiin valmiiksi kymmeniä kappaleita ja määriteltiin niin, että ne ovat aina ja helposti käytettävissä, niin ne tehostavat ylläpitäjien päivittäistä työskentelyä.

Myös Office365-palvelukokonaisuuden Power Automatea hyödynnettiin ja sen käytöstä taustaprosessien toteuttamisessa saatiin arvokasta lisätietoa. Havaittiin, että Power Automate soveltuu joissakin tapauksissa hienosti myös tähän tarpeeseen, vaikka se onkin ensisijaisesti lähinnä henkilökohtaisten työnkulkujen automatisointiväline. Power Automaten työnkulkujen muodostaminen tapahtuu graafisen käyttöliittymän kautta, mutta ohjelmointiosaaminen helpottaa niiden muodostamista varsinkin monimutkaisemmissa työnkuluissa. Voisi sanoa, että Power Automaten aktiviteettien monipuolisessa hyödyntämisessä ja vianselvityksessä on ohjelmointiosaaminen välttämätöntä ja se koskee erityisesti sillä toteutettavien taustaprosessien automatisointia.

Azure-pilvessä toimivan Power Automaten yksi miinus on ainakin tällä hetkellä siinä, ettei sillä voi suoraan tuottaa sellaista toiminnallisuutta, jolla voisi ohjata organisaation paikallista aktiivihakemistoa, koska siinä ei ole sopivaa yhdistintä tähän tarpeeseen. Tähän on olemassa erilaisia kiertoteitä ja kolmannen osapuolen maksullisia sovelluksia, mutta ei Microsoftin virallisesti tukemaa ratkaisua suoraan Power Automaten yhdistimillä. (Korth 2020.)

Toteutettujen automaatioiden lisäksi toimeksiantaja sai dokumentaation järjestelmien välille luoduista yhteyksistä ja automatiikan toiminnasta. Tuotetut automaatiot on kommentoitu myös skriptien sisällä ja siten ne toimivat tavallaan itse itsensä dokumentaationa. Monessa kohdassa havaittiin myös, että PowerShellin helppolukuinen syntaksi auttaa vianselvityksessä, vaikka kommentointia tai dokumentaatiota ei olisikaan sillä hetkellä käytettävissä.

Sen lisäksi, että työn toteutuksen aikana havaittiin uusia tarpeita ja kehitysmahdollisuuksia taustajärjestelmien automatisaatiolle, tunnistettiin organisaatiossa jo kartoitusvaiheessa useampiakin sellaisia tarpeita, joita voisi jatkossa pyrkiä automatisoimaan myös ohjelmistorobotiikkaa hyväksi käyttäen. Mm. henkilöstö- ja talouspalveluissa on monia sellaisia, jopa useita kertoja päivässä toistuvia toimintoja, joita henkilöstö suorittaa perinteisten taulukkolaskentasovellusten lisäksi omissa erikoissovelluksissaan. Tällaisten automatisointi vapauttaisi henkilöstön aikaa muuhun tuottavaan työhön ja samalla se vähentäisi inhimillisten virheiden määrää. Toimeksiantajan taloushallinnossa olisi myös syytä selvittää tekoälypohjaisen automatisaation hyödyntämistä esimerkiksi laskujen tiliöinnissä ja muissa vastaavissa tapahtumissa.

Osa taustajärjestelmissä tapahtuvista toiminnoista on sellaisia, joihin saadaan syöte käyttäjän tekemistä valinnoista jonkin itsepalvelukäyttöliittymän kautta. Tällä hetkellä asiakkaat joutuvat asioimaan usean eri itsepalveluliittymän kautta, eivätkä ne ole kaikki erityisen responsiivisia tai tietoturvasyistä edes käytettävissä muualta kuin organisaation sisäverkosta. Tämä voi olla joillekin käyttäjille sekavaa ja se on osaltaan myös taakka kehittäjille ja ylläpitäjille. Kehitysehdotus tähän olisi tutkia mahdollisuutta keskitetylle itsepalveluliittymälle, joka toimii responsiivisesti ja turvallisesti eri alustoilla, myös mobiililaitteilla, ja tietoturva huomioiden mahdollisesti myös muualta kuin organisaation sisäverkosta. Lisähuomiona edellinen kommentti koskee myös muita organisaation eri käyttäjäryhmille tarjoamia palveluita.

Organisaation käyttämät palvelimet ja verkkolaitteet ovat nykyaikaisia ja ohjelmistojen sekä tietoturvamääritysten suhteen keskitetysti ja nykyaikaisin menetelmin kattavasti hallittuja, joten näiltä osin ei löytynyt kehitysehdotuksia. PowerShell-skriptien osalta voisi olla järkevää pohtia niiden digitaalista allekirjoittamista ja yksittäisten skriptien kokoamista moduuleiksi. Lisäturva allekirjoituksesta muodostuisi siitä, että skriptien muokkaaminen estyisi ilman uutta allekirjoitusta, ja toisaalta myös siitä, että palvelimella voitaisiin estää allekirjoittamattomien skriptien suoritus kokonaan. Moduulien käyttäminen helpottaisi ja selkeyttäisi yksittäisinä luotujen skriptien käyttämistä ylläpitäjien kesken.

Monesti automatisaation laajentamisen esteenä ovat rahoitus, henkilöresurssit, eri tietojärjestelmien oma historia ja mahdollisesti näiden kaikkien yhdessä

aiheuttama uskalluksen puute lähteä toteuttamaan kovin merkittäviä, kokonaisvaltaisia uudistuksia. Automatisoinnin hyödyt ovat kiistattomia, mutta varsinkin alkuvaiheessa hyppy tuntemattomaan saattaa kustannusten ja projektin kokonaisvaltaisten vaikutusten myötä tuntua ylitsepääsemättömältä. Näin ei kuitenkaan tarvitsisi olla, sillä kaikkea ei tarvitse tehdä kerralla, mutta maaliin johtavan tien olisi hyvä olla suunniteltuna ja kaikkien osapuolten tiedossa.

Järjestelmien muuttuessa ja kehittyessä on muistettava huomioida myös aiemmin luotujen automatisointien toiminnallisuus ja niiden kehitys. Monesti kuulee, että yrityksissä on lähdetty toteuttamaan automatisointia vanhalle prosessille, vaikka olisi saattanut olla järkevämpää uudistaa samassa yhteydessä koko prosessi. Mikäli vanhalle toimintamallille ei ole muuta perustetta kuin tunnesyyt, ei sellaista kannata roikottaa mukana painolastina. Tämä ei ole useinkaan tekninen haaste, vaan pikemminkin yrityksen sisäistä politiikkaa ja toimintakulttuurin johtamista.

Vanhan sanonnan mukaan vain muutos on pysyvää ja se pitää paikkaansa IT-maailmassa erityisen hyvin. Tähän liittyen Microsoft on ohjeistanut Office365:n/Azuren käyttäjiä siirtymään Graph API:n käyttöön ja vaikka muutos ei hetkessä tapahdukaan, on vähitellen syytä varautua päivittämään toteutetut automatisoinnit sen kautta. Käytetty kieli on edelleen sama PowerShell, mutta rajapinta vaihtuu ja siten monia Azurea vasten toimivia skriptejä pitää muokata jonkin verran. Power Automate käyttää suoraan Graph Apia, joten sitä tämä ei koske.

Tämän raportin tuottamisen yhteydessä tehty työ tarjoaa hyvän pohjan toimeksiantajan tietojärjestelmien automatisoinnin jatkokehitykselle järjestelmien ja tarpeiden muuttuessa.

LÄHTEET

Autor, D. 2015. Why Are There Still So Many Jobs? The History and Future of Workplace Automation. The Journal of Economic Perspectives; Nashville. Vol. 20(3), pp. 3–30. WWW-dokumentti. Saatavissa: https://www.aeaweb.org/articles/pdf/doi/10.1257/jep.29.3.3 [viitattu 19.8.2021].

Baier, D., Bertocci, V., Brown, K., Densmore, S., Pace, E. & Woloski, M. 2011. A Guide to Claims-Based Identity and Access Control, Second Edition. E-kirja. Redmond, WA, USA: Microsoft patterns & practices. Saatavissa: https://www.microsoft.com/en-us/download/details.aspx?id=28362 [viitattu 19.8.2021].

Brown, J. 2018. Scheduling PowerShell scripts with usernames and encrypted password. Blogi. Saatavissa: https://4sysops.com/archives/scheduling-powershell-scripts-with-usernames-and-encrypted-passwords [viitattu 19.8.2021].

Davenport, T., Kirby, J. 2015. Beyond Automation. Harvard Business Review, 6/2015. Saatavissa: https://hbr.org/2015/06/beyond-automation [viitattu 19.8.2021].

eDirectory 9.2 s.a. Micro Focus. WWW-dokumentti. Saatavissa: https://www.netiq.com/documentation/edirectory-92 [viitattu 19.8.2021].

Golden, B. 2009. Virtualization for Dummies. 2. HP erikoispainos. Indianapolis, USA: Wiley Publishing, Inc.

Haka-käyttäjätunnistusjärjestelmä s.a. CSC – Tieteen tietotekniikan keskus Oy. WWW-dokumentti. Saatavissa: https://www.csc.fi/haka-kayttajatunnistusjarjestelma viitattu 19.8.2021].

Hassell, J. 2017. Learning PowerShell. E-kirja. Boston, USA: DEG Press. Saatavissa: https://kaakkuri.finna.fi [viitattu 19.8.2021].

Holmes, L. 2013. Windows PowerShell Cookbook: The Complete Guide to Scripting Microsoft's Command Shell. 3 painos. O'Reilly Media.

Identity Manager 4.8 s.a. Micro Focus. WWW-dokumentti. Saatavissa: https://www.netiq.com/documentation/identity-manager-48 [viitattu 19.8.2021].

Institute for Robotic Process Automation. 2015. Introduction to Robotic Process Automation. Saatavissa: https://irpaai.com/wp-content/uploads/2015/05/Robotic-Process-Automation-June2015.pdf [viitattu 19.8.2021]

Jyväskylän ammattikorkeakoulu. 2021. Miksi JAMKiin? WWW-dokumentti. Saatavissa: https://www.jamk.fi/fi/Koulutus/Miksi-JAMKiin [viitattu 19.8.2021].

Kopczynski, T. 2007. Windows PowerShell Unleashed. E-kirja. Indianapolis: Sams Publishing. Saatavissa: https://bbooks.info/b/w/b98dac8632af7204085f132c08de8fd1644e336e/windows-powershell-unleashed.pdf [viitattu 19.8.2021].

Korth, E. 2020. Use Power Automate for Azure AD and Active Directory Tasks – Part Two. Blogi. Saatavissa: https://identitypro.blog/use-power-automate-for-azure-ad-and-active-directory-tasks-part-two [viitattu 19.8.2021].

Kääriäinen, J. (toim.), Aihkisalo, T., Halén, M., Holmström, H., Jurmu, P., Matinmikko, T., Seppälä, T., Tihinen, M. & Tirronen, J. 2018. Ohjelmistorobotiikka ja tekoäly – soveltamisen askelmerkkejä? E-kirja. Valtioneuvoston kanslia. Saatavissa: http://urn.fi/URN:ISBN:978-952-287-616-4 [viitattu 19.8.2021].

Lacity, M., Willcocks, L. 2016. A New Approach to Automating Services. MITSloan Management Review. WWW-dokumentti. Saatavissa: https://sloanreview.mit.edu/article/a-new-approach-to-automating-services [viitattu 19.8.2021].

Linden, M. 2015. Identiteetin- ja pääsynhallinta. Tampereen teknillinen yliopisto. Tietotekniikan laitos. Raportti 6. Saatavissa: http://URN.fi/URN:ISBN:978-952-15-3568-0 [viitattu 19.8.2021].

Mann, R. 2018. 10 Automation and AI Tips for Working Smarter on the IT Service Desk. WWW-dokumentti. Saatavissa: https://itsm.tools/10-automation-ai-tips-for-working-smarter-on-the-it-service-desk [viitattu 19.8.2021].

Melnick, J. 2021. How to Automate PowerShell Scripts with Task Scheduler. Blogi. Saatavissa: https://blog.netwrix.com/2018/07/03/how-to-automate-powershell-scripts-with-task-scheduler [viitattu 19.8.2021].

Microsoft. 2017. Active Directory Domain Services Overview. WWW-dokumentti. Saatavissa: https://docs.microsoft.com/en-us/windows-server/identity/ad-ds/get-started/virtual-dc/active-directory-domain-services-overview [viitattu 19.8.2021].

Microsoft. 2020. What is Azure Active Directory? WWW-dokumentti. Saatavissa: https://docs.microsoft.com/en-us/azure/active-directory/fundamentals/active-directory-whatis [viitattu 19.8.2021].

Microsoft. 2021. What is Powershell? WWW-dokumentti. Saatavissa: https://docs.microsoft.com/en-us/powershell/scripting/overview?view=powershell-7.1 [viitattu 19.8.2021].

Microsoft. 2021a. Environments overview. WWW-dokumentti. Saatavissa: https://docs.microsoft.com/en-us/power-platform/admin/environments-overview [viitattu 19.8.2021].

Nevalainen, J. 2020. Automatisoi rutiinit Microsoft Power Automate -työkalun avulla. Blogi. Saatavissa: https://www.tietokeskus.fi/blogi/automatisoi-rutiinit-microsoft-power-automate-tyokalun-avulla [viitattu 19.8.2021].

Preimesberger, C. 2021. Best identity access management software 2021. WWW-artikkeli. Saatavissa: https://www.zdnet.com/article/best-enterprise-identity-access-management-software [viitattu 19.8.2021].

Simmons, A. 2020. Update your applications to use Microsoft Authentication Library and Microsoft Graph API. Blogi. Saatavissa: https://techcommunity.microsoft.com/t5/azure-active-directory-identity/update-your-applications-to-use-microsoft-authentication-library/ba-p/1257363 [viitattu 19.8.2021].

Sinikallio, M. 2018. Ohjelmistorobotiikka asiakaspalvelun kehittämisessä. Diplomityö. Tampereen teknillinen yliopisto. Saatavissa: http://urn.fi/URN:NBN:fi:tty-201805251865 [viitattu 19.8.2021].

Vmware s.a. Application virtualization? WWW-dokumentti. Saatavissa: https://www.vmware.com/topics/glossary/content/application-virtualization [viitattu 19.8.2021].

Wilson, N s.a. Why choose LDAP? WWW-dokumentti. Saatavissa: https://ldap.com/why-choose-ldap [viitattu 19.8.2021].

KUVALUETTELO

Jaettujen postilaatikoiden luonti ja oikeushallinta, ohjelmakoodi

```
<#
        .NOTES
        ===========================================================================
         Created with:  PowerShell Studio
         Created on:    16.8.2021
         Created by:    M. Kautiainen
         Organization:  JAMK
         Filename:
        ===========================================================================
        .DESCRIPTION
                Creates shared mailboxes to all "smb-groups" and syncs rights to all previously created mailboxes based on smb-group members.
#>

#Log location and format
$Logfile = "log\SharedMailbox-$(get-date -Format yyyy-MM-dd).log"

Function LogWrite
{
    Param ([string]$logstring)
    $Logfiletime = get-date -Format HH:mm
    $content = $Logfiletime + " " + $logstring
    Add-content $Logfile -value $content
}

#Import modules
import-module ActiveDirectory

# -------------------
#
# O365 User
#
# -------------------

$O365CREDS = ###REMOVED###
Import-Module MSOnline
Connect-MSOLService -Credential $O365CREDS
$O365Session = New-PSSession -ConfigurationName Microsoft.Exchange -ConnectionUri "https://ps.outlook.com/powershell/" -Credential $O365CREDS -Authentication Basic -AllowRedirection
Import-PSSession $O365session

#------------------------

#Fetch all groups from OU
#Write-Host "Searching for smb-groups..."
$groups = Get-ADGroup -filter * -Searchbase "OU=smb,OU=Groups,DC=ad,DC=jamk,DC=fi"


$count = $groups.count

Write-Host "-------------------------------------"
Write-Host ""
Write-Host "Found total of $count groups to process..." -foregroundcolor yellow
Write-Host ""

$counter = 0

# ---------------------------------------------
#
# Process all groups
#
# ---------------------------------------------

foreach ($group in $groups){
        #Increment counter inside loop
        $counter++

        #Variables
        $name = $group.name
        $samname = $group.samaccountname
        $alias = $name
        $domain = "@jamk.fi"
        $email = $name+$domain

        Write-host "Now processing group ($counter/$count): $name" -foregroundcolor yellow

        #If samaccountname does not match match cn. This is needed to process rights correctly later
        if ($group.name -notlike $group.samaccountname){
                Write-host "Group cn does not match samaccountname, setting exceptionbit: $name / $samname" -foregroundcolor red
                $nameMatchError = 1

                #Uncomment next line if there is a need to skip group process for current object
                #continue
        }

        # ---------------------------------------------
        #
        # Mailbox already exists --> Check and set ACL's
        #
        # ---------------------------------------------


        if (Get-mailbox $name -Erroraction SilentlyContinue){
                Write-host "$name --> Mailbox found!" -foregroundcolor green

                #Read current rights
                Write-host "Reading current ACL's from mailbox..." -foregroundcolor yellow
                $mailboxrights = Get-MailboxPermission $name | where { ($_.IsInherited -eq $False) -and -not ($_.User -like "NT AUTHORITY\SELF") } | select -ExpandProperty user
                Write-host "Reading current ACL's from group..." -foregroundcolor yellow
                #$usersInGroup = Get-ADGroupMember -Identity $group -Recursive | Get-ADUser -Properties Mail | Select -ExpandProperty  Mail
                |
                #use different attribute for group name based on $nameMatchError variable
                if ($nameMatchError){
                        #$users = Get-ADGroupMember -identity $samname
                        $usersInGroup = Get-ADGroupMember -Identity $samname -Recursive | Get-ADUser -Properties Mail | Select -ExpandProperty  Mail
                }
                else{
                        #$users = Get-ADGroupMember -identity $name
                        $usersInGroup = Get-ADGroupMember -Identity $name -Recursive | Get-ADUser -Properties Mail | Select -ExpandProperty  Mail
                }

                #Compare group members to mailbox
                Foreach ($user in $usersInGroup) {
                        if ($mailboxrights -contains $user){
                                write-host "User in AD group has rights to mailbox --> bypass" -foregroundcolor green
                                #continue
                        }
                        if ($mailboxrights -notcontains $user){
                                write-host "User in AD group does not have rights to mailbox --> add rights" -foregroundcolor yellow
                                #$username = Get-ADUser $user
                                Add-MailboxPermission -Identity $email -User $user -AccessRights FullAccess -AutoMapping:$true -InheritanceType all -Confirm:$false
                                Add-RecipientPermission -Identity $email -Trustee $user -AccessRights SendAs -Confirm:$false
                                Set-Mailbox -Identity $email -GrantSendOnBehalfTo @{Add=$user}
                        }
                }

                #Compare current mailbox users to group object
                Foreach ($user in $mailboxrights) {
                        if ($usersInGroup -contains $user){
                                write-host "Mailbox user is a member of AD group --> bypass" -foregroundcolor green
                                #continue
                        }
                        if ($usersInGroup -notcontains $user){
                                write-host "Mailbox user is not a member of AD group --> remove rights" -foregroundcolor yellow
                                Remove-RecipientPermission $email -AccessRights SendAs -Trustee $user -Confirm:$false
                                Remove-MailboxPermission -Identity $email -User $user -AccessRights FullAccess -Confirm:$false
                                Set-Mailbox -Identity $email -GrantSendOnBehalfTo @{Remove = $user}
                        }
                }
        }
        # ---------------------------------------------
        #
        # Mailbox not found --> create mailbox and set rights
        #
        # ---------------------------------------------
        else {
                write-host "$name --> Mailbox not found, creating it..." -foregroundcolor yellow

                #Create shared mailbox

        try {
                New-Mailbox -Shared -Name $name -Alias $alias -PrimarySmtpAddress $email -ErrorAction Stop
                }
                catch {
                LogWrite "Mailbox luonti epäonnistui $name"
                LogWrite "$_"
                Continue
                }

                Set-Mailbox $name -MessageCopyForSentAsEnabled $True
                Set-Mailbox $name -MessageCopyForSendOnBehalfEnabled $True

                #Set rights to mailbox
                Write-host "Reading group members..." -foregroundcolor yellow

                #use different attribute for group name based on $nameMatchError variable
                if ($nameMatchError){
                        $users = Get-ADGroupMember -identity $samname
                }
                else{
                        $users = Get-ADGroupMember -identity $name
                }

                #Skip if group is empty
                if ($users.count -lt 1){
                        write-host "No users in group..." -foregroundcolor yellow
                }
                else{
                        Write-host "Adding rights to mailbox..." -foregroundcolor yellow
                        Foreach ($user in $users){
                                $username = Get-ADUser $user
                                Add-MailboxPermission -Identity $email -User $username.name -AccessRights FullAccess -AutoMapping:$true -InheritanceType all -Confirm:$false
                                Add-RecipientPermission -Identity $email -Trustee $username.name -AccessRights SendAs -Confirm:$false
                                Set-Mailbox -Identity $email -GrantSendOnBehalfTo @{Add=$user.name}
                        }
                }
        }
}
Write-host ""
Write-host "Processed $counter groups --> Done!" -foregroundcolor green
Write-host ""
```

Tiedostosiirtoja, ohjelmakoodi

```
<#
        .NOTES
        ===========================================================================
         Created with:  PowerShell Studio
         Created on:    19.8.2021
         Organization:  JAMK
        ===========================================================================
        .DESCRIPTION
                Noutaa tiedostoja palvelimelta, jakaa ne osiin ja tallentaa kohdehakemistoihin. Suoritus ajastettuna kolmesti päivässä.

                SFTP-yhteys:
                https://github.com/darkoperator/Posh-SSH/

#>
#---------------------
#
#Muuttujat
#
#---------------------
$scriptPath = ###removed###
$backupPath = "$scriptPath\Backup"
$tempPath = "$scriptPath\Temp"
$targetPath = ###removed###
$MerkkiKoodaus = [System.Text.Encoding]::GetEncoding(1252)

#---------------------
#
#Tunnukset
#
#---------------------
$sftpServer = "###removed###"
$ServerPort = "###removed###"
$credentials = New-Object System.Management.Automation.PSCredential $sftpUsername, $sftpPassword

try
{
        #Alustetaan aikaleima
        $aikaleima = (get-date).AddDays(0).ToString("yyMMdd-HH.mm.ss")

        #Muodostetaan yhteys/sessio ja kopioidaan tiedosto(t) sftp-palvelimelta
        $session = New-SFTPSession -ComputerName $sftpServer -Port $ServerPort -Credential $Credentials -AcceptKey
        Get-SFTPFile $session.SessionId -RemoteFile /###removed###/###removed###1.csv -LocalPath $tempPath -Overwrite
        Get-SFTPFile $session.SessionId -RemoteFile /###removed###/###removed###2.csv -LocalPath $targetPath -Overwrite


        #-------------------------
        #Tiedoston jako osiin
        #-------------------------

        $jaettava_tiedosto = "###removed###1.csv"

        $path = "$tempPath\"

        $fullpath = "$path$jaettava_tiedosto"
        $tiedostotyyppi = ".csv"


        #Määritellään käsiteltävät vuodet
        $kuluva_vuosi = (Get-date).ToString('yy')
        $edellinen_vuosi = (Get-Date).AddYears(-1).ToString('yy')

        #Määritellään tulostiedostot edellisen ja kuluvan vuoden tiedoille
        $kv_tiedosto =  -join((Get-date).ToString('yyyy'),$tiedostotyyppi)
        $ev_tiedosto =  -join((Get-Date).AddYears(-1).ToString('yyyy'),$tiedostotyyppi)

        #Testataan ensin, onko jaettava tiedosto siirtynyt hakemistoon
        if ( Test-Path $fullpath)
        {
                #Listataan "vuositiedostojen" nimet
                $vuosittaiset_tiedostot = $ev_tiedosto,$kv_tiedosto

                #Avataan lukustriimi jaettavalle tiedostolle
                #$lukustriimi = New-Object IO.StreamReader -Arg $fullpath
                $lukustriimi = New-Object IO.StreamReader($fullpath, $MerkkiKoodaus)

                #Avataan kaksi kirjoitus-striimiä. Toinen edelliselle vuodelle ja toinen kuluvalle vuodelle
                $kirjoitus_striimi_ev = New-Object IO.StreamWriter("$path$ev_tiedosto", $false, $MerkkiKoodaus)
                $kirjoitus_striimi_kv = New-Object IO.StreamWriter("$path$kv_tiedosto", $false, $MerkkiKoodaus)

                #Käsitellään lukustriimi yksi rivi kerrallaan. Kunkin rivin neljännessä sarakkeessa/kentässä on tositepäivämäärä.
                #Kaksi ensimmäistä merkkiä kertoo vuoden. Luupin sisällä if-lauseissa splitataan rivi kenttiin puolipisteen
                #perusteella ja testataan vuosi.
                while($rivi = $lukustriimi.ReadLine()){
                        #Testataan, onko kyseessä kuluva vuosi
                        if($rivi.Split(';')[3].Substring(0,2) -eq $kuluva_vuosi)
                        {
                                #Jos on, niin kirjoitetaan se kuluvan vuoden tiedostoon
                                $kirjoitus_striimi_kv.WriteLine($rivi);
                        }
                        #Testataan, onko kyseessä edellinen vuosi
                        elseif ($rivi.Split(';')[3].Substring(0,2) -eq $edellinen_vuosi)
                        {
                                #Jos on, niin kirjoitetaan se edellisen vuoden tiedostoon
                                $kirjoitus_striimi_ev.WriteLine($rivi);
                        }
                }

                #Suljetaan striimit lopuksi
                $lukustriimi.Close(); $lukustriimi.Dispose()
                $kirjoitus_striimi_kv.Close(); $kirjoitus_striimi_kv.Dispose()
                $kirjoitus_striimi_ev.Close(); $kirjoitus_striimi_ev.Dispose()

                #Siirretään luodut tiedostot kohdehakemistoon
                foreach ($tiedosto in $vuosittaiset_tiedostot) {
                        #Määritellään yksittäinen tiedosto polkuineen
                        $siirrettava = "$path$tiedosto"
                        Copy-Item -path $siirrettava -destination $backupPath\$aikaleima-$tiedosto -ErrorAction Ignore -Force
                        Move-Item -path $siirrettava -destination $targetPath\Tiedostot -ErrorAction Ignore -Force
                }

                #Siirretään osiin jaettu siirtotiedosto lopuksi varmuuskopiohakemistoon
                Move-Item -path $fullpath -destination $backupPath\$aikaleima-$jaettava_tiedosto -ErrorAction Ignore -Force
        }
        #-------------------------
}
catch [Exception]
{
    Write-Host $_.Exception.Message
    Exit 1
}
finally
{
        #Siivotaan sessio
        if ($session = Get-SFTPSession -SessionId $session.SessionId) {
                $session.Disconnect()
        }
        $null = Remove-SftpSession -SftpSession $session

        # Siivotaan vanhat tiedostot
        $Days = 10
        get-childitem -Path $backupPath -recurse |
        where-object {$_.lastwritetime -lt (get-date).addDays(-$Days)} |
        Foreach-Object { del $_.FullName }
        Exit 0
}
```