Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Øyvind Lotten Westrum

Azure Automation State Configuration for administrasjon av Windows-servere

Bacheloroppgave i informatikk, drift av datasystemer
Veileder: Jostein Lund
Mai 2019

NTNU
Kunnskap for en bedre verden

Øyvind Lotten Westrum

Azure Automation State Configuration for administrasjon av Windows-servere

Bacheloroppgave i informatikk, drift av datasystemer
Veileder: Jostein Lund
Mai 2019

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU
Kunnskap for en bedre verden

Sammendrag

Windows-gruppen ved NTNU IT, Seksjon for IT-drift ønsket å se på muligheten for å benytte Azure Automation State Configuration til å konfigurere og administrere Windows-servere. De ønsket et sluttprodukt som automatisk installerte og konfigurerte Windows-servere og sikret at konfigurasjonen ikke utilsiktet ble endret over tid. Prosjektet ble gjennomført som en driftsoppgave hvor det ble utviklet og tatt i bruk et produkt for å tilegne seg relevant erfaring og kompetanse. Prosjektet har satt opp løsningen, onboardet maskiner og utviklet konfigurasjonsdokumenter og Desired State Configuration-ressurser for de installasjonene og tilpasningene som gjøres manuelt i dag. De forskjellige funksjonene har blitt nøye testet og prosjektet kan konkludere med at selv om det er enkelte forhold man må ta stilling til, som behov for to-faktor-autentisering, rutiner for tilbakekalling av konfigurasjon og valg av konfigurasjonsmodus er løsningen ellers godt egnet til å fungere som konfigurasjonsstyringssystem.

Abstract

The Windows team at NTNU IT, IT Operation wanted to investigate if Azure Automation State Configuration could be used as a configuration management system for Windows servers. They wanted a Proof of Concept that automatically installed and configured Windows servers and ensured that the configuration did not accidentally change over time. The project was carried out as an operational task were the system is installed and used to gain experience and knowledge. The project installed the solution, onboarded servers and developed configuration and Desired State Configuration resources for installations and adaptations manually configured today. The various features were carefully tested and the project can conclude that although there are certain conditions to be considered, such as the need of two-factor authentication, routines to recall configurations and the selection of configuration mode, the solution is otherwise suitable for functioning as a configuration management system.

Forord

Jeg vil takke Brynjulf Mauring og Jon Arne Reitan ved NTNU IT, Seksjon for IT-Drift for at jeg fikk muligheten til å jobbe med denne oppgaven og for de ideer og støtte som ble gitt underveis. Videre vil jeg også takke veileder Jostein Lund for gode tilbakemeldinger og veiledning gjennom prosessen.

– Øyvind Lotten Westrum

Innholdsfortegnelse

1 Introduksjon

NTNU IT ønsker å se på muligheten til å bruke Azure Automation State Configuration (Azure DSC) til å automatisere installasjon, oppsett og konfigurasjon av servere på Windows-plattformen. Azure DSC er Microsoft sin skytjeneste for konfigurasjonsstyring og gir muligheten til å automatisk konfigurere servere og sikre at konfigurasjonen ikke endres over tid. Oppdragsgiver ønsker en løsning med så mye automatisering som mulig. Dette ønsket er et resultat av at man vil å bruke mer tid på andre og mer interessante arbeidsoppgaver. Men det er også for å sikre mot menneskelige feil, vanvare og muligheten til å reversere uautoriserte endringer i konfigurasjonen automatisk.

Dagens systemer og rutiner baserer seg på bruk av godt innarbeidede tjenester og verktøy som delvis automatiserer installasjon og konfigurasjon. Nesten all installasjon og konfigurasjon blir utført via PowerShell-Script som kjøres sekvensielt i henhold til dokumentert rutinebeskrivelse, mens noe konfigurasjon blir utført via Group Policy (GPO). Ved installasjon og konfigurasjon av en "Session Host" benyttes Microsoft Deployment Toolkit (MDT) for automatisk utrulling og installasjon av operativsystemet på serveren. Videre blir serveren tilpasset av Group Policy (GPO) etter hvor serveren er plassert i Active Directory (AD). Til slutt utføres manuelle tilpasninger via PowerShell-script før den legges inn i Session Collection.

1.1 Dokumentets hensikt

Dokumentet er sluttrapporten i faget IDRI3001 Bacheloroppgave i drift av datasystemer. Rapportens hensikt er å fungere som en oppsummering av prosjektgjennomføringen ved å gi en innføring i problemområdet, beskrive prosjektgjennomføringens metode og resultater, svare på oppdragsgivers forskningsspørsmål og komme med en konklusjon og anbefaling for videre arbeid. Rapporten skal svare på om de mål, rammebetingelser og kritiske suksessfaktorer som ble definert i forstudierapporten er oppnådd.

1.2 Problemstilling

Windows-gruppa ved seksjon for IT-Drift ønsker et sluttprodukt (Proof of Concept) i form av en ferdig konfigurert løsning for konfigurasjon av Session Hosts i et terminalservermiljø. Løsningen skal automatisk konfigurere Session Hosts uten manuelle steg og sikre at konfigurasjonen utilsiktet ikke blir endret over tid. Oppgaven skal løses med skytjenesten Azure Automation State Configuration og man ønsker å kunne konfigurere servere både On-Premises og i skyen. Sluttproduktet skal inkludere komplett dokumentasjon av implementering, konfigurering og driftsrutiner.

I tillegg til sluttproduktet ønskes det svar på følgende forskningsspørsmål:

- Azure Automation State Configuration vs. andre metoder for konfigurasjonsstyring - Hva er fordeler og ulemper, og hvordan kan det eventuelt fungere sammen?
- Vurdering av sikkerhet, stabilitet og sporbarhet.
- Hvilke krav til kompetanse kreves for å implementere og drifte løsningen?
- Skaleringsmuligheter også til andre områder enn "Session Hosts".

- Hvilke utfordringer finnes i forhold til driftsrutiner ved bruk av Azure Automation State Configuration?
- Egne betraktninger rundt teknologien.

Spørsmålene er besvart løpende i teksten, hovedsakelig i diskusjonskapittelet.

1.3 Prosjektmål

Dette kapittelet beskriver mål som ble definert for prosjektet i forstudierapporten. Disse målene har lagt grunnlaget for gjennomføringen og resultatet av prosjektet.

1.3.1 Effektmål

Den ønskede effekten av prosjektet er definert i følgende effektmål:

- Øke seksjonens grad av automatisering ved å benytte Azure Automation for installasjon, konfigurasjon og drift av servertjenester på Windows-plattformen.
- Bedre kvaliteten på tjenestene ved å unngå avvik og konfigurasjonsdrift.

1.3.2 Resultatmål

For å konkretisere det ønskede sluttprodukt og for å definere når prosjektet er fullført, settes følgende resultatmål:

- Levere et sluttprodukt (Proof of Concept) der man automatisk installerer, konfigurerer og setter opp "Session Hosts" i et terminalservermiljø ved hjelp av Azure Automation State Configuration.
- Besvare forskningsspørsmål som er presentert i kapittel 1.2.

Resultatmålene har en tidsramme på 500 timer og skal leveres innen 20. Mai 2019.

1.3.3 Prosessmål

Prosjektgruppen har satt følgende prosessmål for prosjektgjennomføringen:

- Gjennomføre et større prosjekt forankret i en reell problemstilling fra arbeidslivet.
- Øke kompetansen innen automatisering av driftsoppgaver ved bruk av PowerShell.

1.4 Terminologi

Session Host (RDSH) En rolle i Remote Desktop Services (RDS) som gir brukere mulighet til å koble seg til eksternt skrivebord og kjøre Windows-programvare eksternt.

Remote Desktop Services (RDS) Samlebetegnelse for en Windows Server tjeneste som gir mulighet for fjernkjøring av programmer.

Session Collection Samling av Session Hosts som gir mulighet for lastbalansering.

Terminalservermiljø Generelt begrep for et system tilsvarende Remote Desktop Services.

Azure Microsoft sin nettskyplattform.

Azure AD Microsofts skytjeneste for identitet- og tilgangsstyring.

Azure Automation Automasjonstjeneste i nettskyen for prosessautomatisering og konfigurasjonsstyring i Azure.

Azure Automation State Configuration Tjeneste som gir mulighet til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC) konfigurasjoner, importere DSC-ressurser og tilordne konfigurasjoner til noder.

Desired State Configuration (DSC) Styringsplattform i PowerShell som gjør at man kan installere, styre og konfigurere IT-systemene med kode.

Pull Server Tjeneste som tilgjengeliggjør konfigurasjoner, lar noder kontrollere den ønskede tilstand og rapportere tilbake om de er i ønsket tilstand.

DSC Configuration PowerShell script som definerer konfigurasjon, node, ressurs og hvilke attributter ressursen skal konfigureres med.

DSC Node Configuration En kompilert DSC konfigurasjon. Fungerer som en rolle som kan tildeles en server.

DSC Node En maskin/enhet hvor konfigurasjonen er administrert av DSC

DSC Resource Det man bruker for å definere DSC konfigurasjonen (ressursene)

Compilation Job Å kompilere en konfigurasjon til MOF-fil og gjøre den tilgjengelig på pull-serveren.

Local Configuration Manager (LCM) Maskineriet i DSC. Funksjonaliteten som henter, leser og eksekverer konfigurerer på noden.

Managed Object Format (MOF) Format som brukes til å beskrive en konfigurasjon. Brukes av DSC som konfigurerer en node.

Konfigurasjonsstyring Styring av innhold, endringer og status på en konfigurasjon.

On-Premises (Computing) Servere som er eid, kontrollert og plassert lokalt.

Nettsky / Sky (Computing) Infrastruktur, plattform eller programvare som leies, administreres og er plassert hos en tredjepart.

PowerShell Et verktøy og scriptspråk som gir mulighet til å automatisere og styre et operativsystem fra kommandolinjen.

PowerShell Script Samling av PowerShell-kommandoer som automatiserer en oppgave.

PowerShell Cmdlets Et PowerShell-script som utfører en funksjon.

2 Azure Automation State Configuration

Azure Automation State Configuration (Azure DSC) gir muligheten til å administrere Windows-servere med prinsippene til programmerbar infrastruktur, «Infrastructure as Code» [6]. Dette gir muligheten til å definere infrastrukturen som kode og arbeide med lik praksis slik man gjør det innen programvareutvikling. Dette sikrer endringsprosessen, gjør dokumentasjonen enklere og lar deg jobbe med andre oppgaver enn å installere servere [5].

Azure DSC er en tjeneste i Azure som gir muligheten til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC)-konfigurasjoner, importere DSC ressurser og tilordne konfigurasjoner til noder. Det overordnede målet med Azure DSC er å automatisere installasjon og konfigurasjon av servere og sikre at konfigurasjonen ikke endres over tid. Konfigurasjonene skal videre fungere som "en enkeltkilde til sannhet" og en levende dokumentasjon av infrastrukturen. Det betyr at i stedet for å klikke og konfigurere seg igjennom en rekke menyer, veivisere, register-innstillinger og konfigurasjonsfiler har man ett konfigurasjonsdokument som definerer alt det man ønsker å konfigurere.

Konfigurasjonsdokumentet vil fungere som dokumentasjon siden alt som installeres og konfigureres på maskinene som administrerer blir definert ett sted. Konfigurasjonen er definert deklarativt i PowerShell-kode som gjør det mye enklere å lese og feilsøke ved eventuelle problemer siden antall mulige feilkilder blir redusert [2].

Videre blir alle endringer i programvare, konfigurasjon, register, roller og funksjoner gjort i konfigurasjonsfila. Dette resulterer i en enklere og tryggere endringshåndtering siden endringen utføres ett sted og det gir større mulighet for testing. Man kan for eksempel med få tastetrykk kjøre ut konfigurasjonen til et testmiljø, utføre og teste endringen, før man igjen tar ned testmiljøet for å utføre endringen på produksjonsmiljøet. Dette gjør konfigurasjonene til en levende dokumentasjon siden dokumentasjonen endres i takt med endringene i infrastrukturen uten ekstra arbeid [3].

Til slutt er det verdt å nevne at Azure DSC gir større mulighet til å skalere infrastrukturen. Det vil si at man får muligheten til å kjøre opp og ned servere uten mye arbeid ettersom installasjon- og konfigurasjonsprosessen er automatisert. Dette egner seg godt i dynamiske miljøer der behovet for datakraft er variabelt i forhold til pågangen [3]. Dette kan for eksempel passe i henhold til problemstillingen der man i perioder har behov for økt datakraft og antall Session Hosts i et terminalservermiljø.

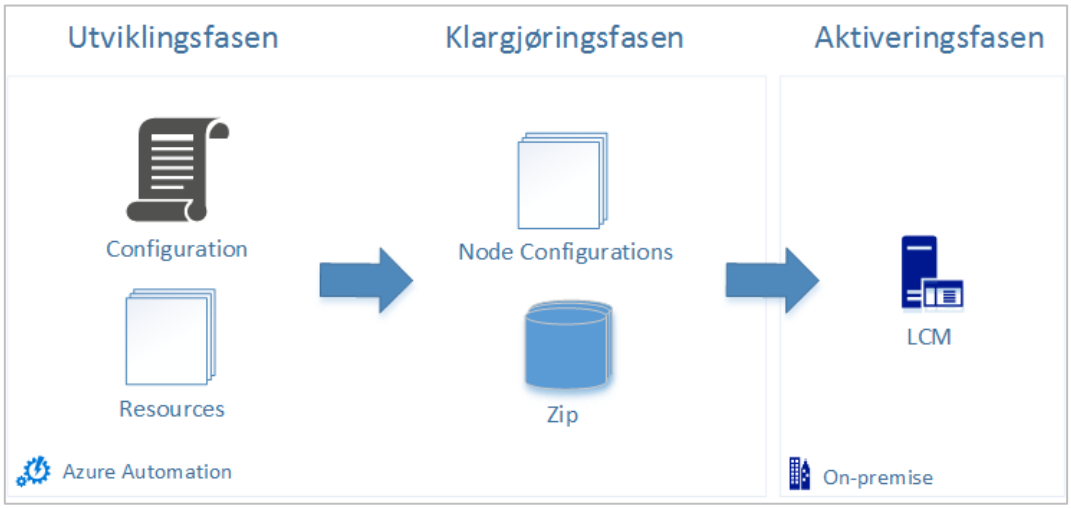
2.1 Komponenter og faser

DSC er som tidligere nevnt deklarativ. Deklarativ betyr at man beskriver kun det som skal være gjeldene og at koden som utfører handlingene er abstrahert vekk fra brukerens betraktning. Dette resulterer i lettlest kode som er enkel å administrere. Man trenger for eksempel kun å definere at en bestemt Windows Feature skal være tilstede, og det er det hele. Koden som utfører selve operasjonen ligger i den underliggende infrastrukturen og i ressursene man inkluderer sammen med konfigurasjonen. Koden inneholder logging, skriving av eventuelle feilmelding og logikk som sikrer korrekt kjøring av PowerShell-kommandoene.

DSC er også Idempotent. Det betyr at man kan kjøre den samme konfigurasjonen mange ganger og fortsatt oppnå samme resultat. Det vil si at hvis man definerer en konfigurasjon der en Windows Feature skal være tilstede, vil den kun installere den hvis den ikke er der. Den vil ikke reinstallere eller installere den flere ganger hvis den allerede er installert [4].

Det er verdt å nevne at Azure DSC kun vil kontrollere tilstanden på det man definerer i konfigurasjonen. Hvis man administrerer en maskin med DSC vil ikke DSC bry seg om programvare, filer og annet som ikke er definert i konfigurasjonen.

Prosessen i Azure DSC består av tre faser som vist på Figur 1. I tillegg til disse fasene vil noden rapportere tilbake til Azure Automation med tilstandsstatus på konfigurasjonen dersom dette valget er tatt. Det er ressursene man inkluderer i konfigurasjonen som tar seg av kontroll av tilstanden.



Figur 1 - Faser i Azure DSC [4]

Utviklingsfasen (Authoring) består i å utvikle konfigurasjonsdokumenter og inkludere egen- eller allerede utviklede ressurser i PowerShell for å konfigurere de rollene man ønsker å ha i sin infrastruktur. Det kan for eksempel være en rolle for en "Session Host" i Collection Adminfarm eller en rolle for "Connection-Broker". Konfigurasjonsdokumentene er bygget opp av en "Configuration"-blokk, en "Node"-blokk og en eller flere ressursblokker. Konfigurasjonsblokka fungerer på samme måte som en PowerShell-funksjon hvor man kan importere eksterne ressurser, parametere, løkker og if-else logikk. Nodeblokka spesifiserer konfigurasjonens rolle. Se Kodeblokk 1 for eksempel på en konfigurasjon.

```
Configuration ConfigurationExample
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration

    Node SessionHost
    {
        WindowsFeature RDS
        {
            Ensure = "Present"
            Name = "RDS-RD-Server"
        }
    }
}
```

Kodeblokk 1 – Konfigurasjonseksempel

Når konfigurasjonen er klar, kompileres den først til en fil med Managed Object Format

(MOF) og gjøres tilgjengelig på Pull-serveren. Filformatet fungerer som mellomformat og gir et standard oppsett som lar DSC utføre det man definerer i konfigurasjonen. DSC er basert på Common Information Model (CIM) som er en standard for hvordan man definerer struktur og oppførsel til administrerte nettverksressurser. MOF-dokument benyttes til å beskrive CIM-klassene som DSC benytter [16] [17].

I klargjøringsfasen (Stageing) vil den kompilerte MOF-fila og de tilhørende ressursene gjøres tilgjengelig på Pull-serveren i Azure Automation. Ressursene komprimeres for økt ytelse [4].

Siste fase er aktiveringsfasen (Configuration Enacting). Her vil Local Configuration Manager (LCM) lese MOF-fila og konfigurere klienten i henhold til hva som er definert i konfigurasjonsfila i fase én [4]. Den siste fasen består også av rapportering av konfigurasjonsstatus til Azure DSC.

2.1.1 Local Configuration Manager (LCM)

LCM er selve motoren / agenten til DSC som kjører på alle nodene som skal administreres. Den er ansvarlig for å hente og motta konfigurasjonene fra Pull-serveren, dekode og utføre konfigurasjonene på noden, kontrollere om noden er i den ønskede tilstanden og rapportere tilbake til Azure DSC med nodens konfigurasjonsstatus.

LCM har en rekke konfigurasjonsparametere som defineres i en konfigurasjonsfil med navn "MetaConfig.mof". Denne konfigureres når man onboarder en node til Azure. Minimal konfigurasjon er visst i Tabell 1 [10].

Innstilling	**Forklaring**
NodeConfigurationName	Hvilken DSC Node Configuration som skal konfigureres.
RefreshFrequency	Tiden for hvor ofte LCM skal sjekke Pull Server om det har kommet ny konfigurasjon.
ConfigurationMode	Definerer hvordan LCM bruker konfigurasjonen. ApplyOnly ApplyAndMonitor ApplyAndAutoCorrect
ConfigurationModeFrequency	Tiden for hvor ofte LCM kontrollerer om konfigurasjonen har blitt endret.
RebootNodeifNeeded	Definere om LCM kan utføre restart av noden hvis det er behov i f.eks. en installasjonsprosess.
ActionAfterReboot	Definere hvilken handling LCM skal utføre etter en restart som er utløst av DSC. Man kan velge mellom å stoppe eller fortsette installasjon.

Tabell 1 - LCM konfigurasjon

Innstillingen ConfigurationMode inneholder tre valg:

- **ApplyOnly** betyr at LCM bare konfigurerer noden
- **ApplyAndMonitor** betyr at i tillegg til å konfigurere noden skal også LCM melde tilbake hvis konfigurasjonen drifter fra den ønskede tilstanden.
- **ApplyAndAutoCorrect** betyr at i tillegg til å konfigurere noden skal også LCM rekonfigurere noden hvis konfigurasjonen drifter fra den ønskede tilstanden.

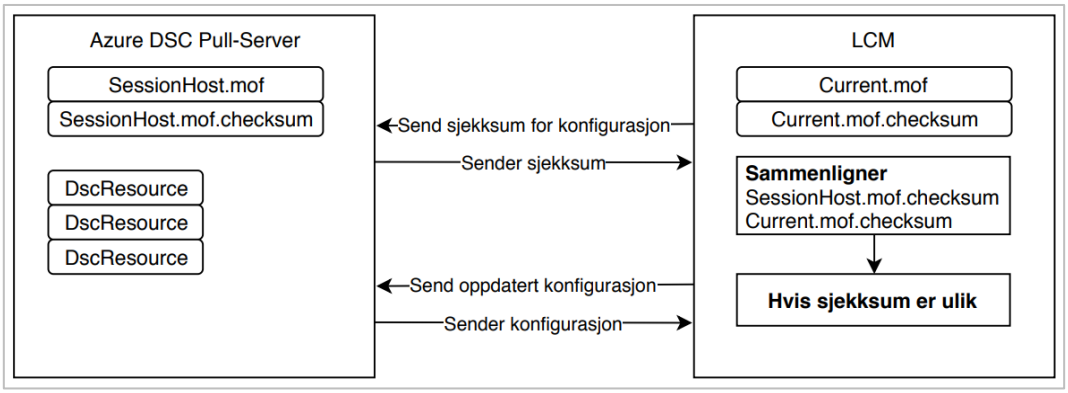
Videre har LCM tre konfigurasjonsfiler [6:68-69] som har følgende funksjonalitet:

- **Current.mof** er konfigurasjonen som maskinen har i dette øyeblikket
- **Pending.mof** er konfigurasjonen som maskinen har mottatt, men ikke enda aktivert
- **Previous.mof** er konfigurasjonen som maskinen har hatt tidligere

2.2 Henting av ny konfigurasjon

LCM vil regelmessig kontrollere Pull-serveren for oppdateringer. Det er egenskapen "RefreshFrequencyMins" som bestemmer hvor ofte den vil sjekke. Denne konfigurasjonen er satt til hvert 30. minutt som standard, men det er også mulig å fremprovosere oppdatering med PowerShell-kommandoer [6:69-70].

Som vist på figur 2 gjennomfører LCM kontroll av Pull-serveren om det har kommet en ny konfigurasjon. Dette gjør den ved å kjøre PowerShell-kommandoen "Get-DscAction". Da sammenlignes sjekksummen av den nåværende konfigurasjonen opp mot sjekksummen av konfigurasjonen på Pull-server som den har blitt tilsendt. Stemmer sjekksummene overens vil ikke LCM foreta seg noe. Er det forskjell på sjekksummene vil LCM laste ned den oppdaterte konfigurasjonsfila og eventuelle nødvendige DSC-ressurser og implementere disse på noden [6:286-287].



Figur 2 - Kontroll av ny konfigurasjon

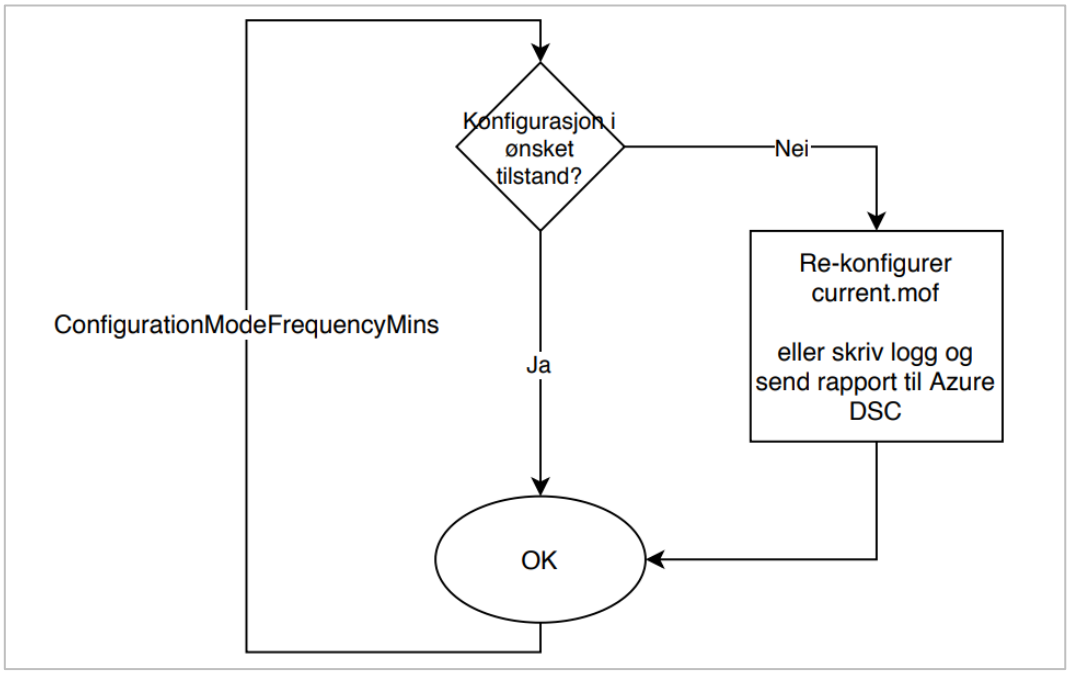
DSC benytter PowerShell-kommandoen "Get-DscDocument" for nedlastning av konfigurasjonsdokumentet og kommandoen "Get-DscModule" for nedlastning av eventuelle DSC-ressurser som er inkludert i konfigurasjonen. Disse kommandoene er ikke noe man behøver å kjøre manuelt, men de er relevante i for eksempel en feilsøkingsituasjon [6:312-313].

Med Azure DSC er det ikke behov for å opprette sjekksum av konfigurasjonsdokumentene manuelt. Håndtering og opprettelse av sjekksum til konfigurasjonsdokumentene er inkludert som en tjeneste i Azure DSC.

2.3 Monitorere konfigurasjonsdrift

LCM kan kontrollere om noden er i ønsket tilstand eller ikke. Det er den konfigurerbare egenskapen "ConfigurationModeFrequencyMins" som spesifiserer hvor ofte LCM kan kontrollerer tilstanden til konfigurasjonen. Standardkonfigurasjonen er hvert 15. minutt og resultatet av hva LCM gjør kommer an på valget man tok for ConfigurationMode da LCM ble konfigurert.

Som visst på figur 3 vil LCM vil utføre en «Dsc Consistency Check» regelmessig. Den sjekker om konfigurasjonen er i ønsket tilstand. Hvis noden ikke er det vil LCM enten re-konfigurere «current.mof» eller skrive en EventLog og rapportere til Azure DSC at den er «Not Compliant». Dette er avgjørende i hensyn til hvilken konfigurasjonsmodus som er konfigurert i LCM. Når det har gått så lang tid som "ConfigurationModeFrequencyMins" er konfigurert til vil operasjonen starte på nytt [6:316-317].



Figur 3 – Tilstandskontroll

Det er DSC-ressursene som innehar funksjonaliteten til å kontrollere om konfigurasjonen er i ønsket tilstand eller ikke. Ressursene er bygget opp etter en standard og alle ressursene har tre metoder tilgjengelig: Get, Test og Set [12].

- **Get-metoden** kontrollerer hvilken verdi egenskapene til ressursen har hos noden.
- **Test-metoden** kontrollerer om noden er i den konfigurerte ønskede tilstand.
- **Set-metoden** vil konfigurere noden slik at den settes i konfigurasjonens ønskede tilstand.

Av de forskjellige metodene er det er test-metoden som benyttes til å bedømme om konfigurasjonen er i ønsket tilstand. Dette er en funksjon som returnerer en boolsk verdi om noden er korrekt konfigurert. Mer om DSC-ressurser i kapittel 2.4

2.4 DSC-ressurser

DSC ressursene er de spesifikke funksjonene som kjøres når Azure DSC iverksetter konfigurasjon og kontroll mot nodene som skal administreres. En ressurs kan for eksempel være “WindowsFeature”, som kan brukes til å sikre at en Windows Feature enten er tilstede eller fjernet fra en definert node. Som vist på Kodeblokk 2 kalles ressursen med ressursnavnet “WindowsFeature”. Denne konfigurasjonen sier at rollen “RDS-RD-Server" skal være tilstede med verdien “Present”.

```
WindowsFeature ResourceExample
{
    Ensure = "Present"
    Name = "RDS-RD-Server"
}
```

Kodeblokk 2 - Eksempel på bruk av en DSC Resource

Det er en rekke inkluderte ressurser i PowerShell, men Microsoft legger opp til at brukerne selv kan utvikle sine egne ressurser og dele de med andre [11]. Ressursene er Open Source og publiseres på for eksempel The PowerShell Gallery og kan importeres direkte inn i Azure Automation. Finner man ikke ressurser som dekker behovet kan man enten lage egendefinerte ressurser eller endre på eksisterende ressurser.

Kompositttressurser

Det er også mulig å utvikle kompositttressurser (DSC Composite Resource). En kompositttressurs er en DSC-ressurs bestående av en annen konfigurasjon. Kompositttressurser egner seg godt som baseline-konfigurasjon. Ved å gjøre det på denne måten vil man håndtere konfigurasjonen ett sted og ikke én gang for hver konfigurasjon man har. Dette sikrer for eksempel en endringsprosess mot vanvare og forglemmelse.

Andre bruksområder enn baseline konfigurasjon kan være ved veldig avanserte konfigurasjonsblokker. Det kan da lages standardkonfigurasjon for de forskjellige tjenestene man har behov for som så refereres inn i konfigurasjonene til de nodene som har behov for en viss type funksjonalitet. Da vil man i stedet for å bygge opp en ny konfigurasjon hver gang kunne benytte allerede eksisterende konfigurasjon. Dette kan for eksempel være brannmurregler eller et bestemt oppsett av Web-servere.

3 Metode

Dette kapittelet beskriver hvordan arbeidsprosessen har vært igjennom prosjektperioden. Det tar for seg informasjonsinnhenting, arbeidsmetodikk, de valgene som er tatt og bakgrunnen for disse.

For innhenting av relevant informasjon og kunnskap til prosjektet er Microsofts dokumentasjonssider en mye brukt ressurs. Dokumentasjonen blir oppdatert kontinuerlig og er den sikreste kilden til "beste praksis" for hvordan forskjellige oppgaver skal løses. For bakgrunnsinformasjon og teori om teknologiens underliggende funksjonalitet er det benyttet faglitteratur [6] og videokurs hos Microsoft Virtual Academy Live [7][8] og Pluralsight [9].

Prosjektet er en driftsoppgave der det skal utvikles et "Proof of Concept" som demonstrerer funksjonalitet og gjennomførbarhet ved å ta i bruk et produkt. Prosjektgjennomføringen er basert på fossefallsmetoden. Det er en lineær og sekvensiell metode hvor hver fase gjennomføres etter hverandre med et avsluttende beslutningspunkt der man bestemmer om man skal gå videre eller ikke. For hver fase i prosjektet ble det utformet en rapport som både oppdragsgiver og prosjektgruppa har måttet godta før neste fase kunne påbegynnes. Fossefallsmetoden egnet seg godt for prosjektet fordi det var en definert start- og sluttdato og de forskjellige oppgavene som skal løses er godt dokumentert av oppdragsgiver slik at risikoen for overraskelser underveis var minimal.

Prosjektet besto av fire faser: forprosjekt, planleggingsfase, implementeringsfase og avslutningsfase. Forprosjektet definerte på et overordnet nivå hva som skulle gjøres, rammebetingelser prosjektet var underlagt og hva sluttproduktet skulle være. Det skulle fungere som en avtale mellom prosjektgruppa og oppdragsgiver. I planleggingsfasen ble det foretatt grundige undersøkelser, tester og innhenting av relevant teori. Her ble sluttproduktet konkretisert. Implementeringsfasen besto av realisering av planene, testing av funksjonalitet og eventuelle revideringer. Den siste fasen var avslutningsfasen. Her ble det utformet en sluttrapport som beskrev løsningen, inkludert en besvarelse av forskningsspørsmålene som beskrevet i problemstillingen.

Det er benyttet et Kanban-brett, som er en smidig metode, i gjennomføringsfasen som bistand til fossefallsmetoden. Brettet ble laget med følgende arbeidsflyt: To do, Doing, Waiting, Done og Dokumentert.

- **To do** inneholder alle oppgaver som ikke ennå er påbegynt.
- **Doing** er alle oppgaver som er under arbeid. Det er satt "Work in Progress Limit" til to for å begrense arbeidsmengde.
- **Waiting** inneholder alle oppgaver som er på vent av forskjellige grunner.
- **Done** inneholder alle oppgaver som er fullført.
- **Dokumentert** inneholder alle oppgaver som er dokumentert i driftsrapporten.

Kanban-brettet har bidratt med å gi oversikt over hva som må gjøres, hva som gjenstår og å visualisere arbeidsflyten for hver oppgave og begrense hvor mye arbeid som kan gjøres samtidig. Dette er med på å sikre fullt fokus på de oppgavene som er planlagt at skal løses og hindre at man starter for mange oppgaver samtidig.

Alle oppgaver som ble planlagt i planleggingsfasen ble beskrevet på Kanban-kort og kategorisert etter type oppgave. De forskjellige kategoriene er DSC-konfigurasjon, Azure, On-premise, Automatisering og Dokumentasjon.

Arbeidsoppgavene ble så prioritert etter i hvilken rekkefølge de burde løses. De første prioriteringene som ble gjort er visst i Tabell 2.

Prioritering	Kategori	Forklaring
1.	Azure	Konfigurasjon av funksjonalitet i Azure Automation.
2.	On-Premise	Konfigurasjon og on-boarding av lokale maskiner.
3.	DSC-konfigurasjon	Utvikling av konfigurasjonsblokker
4.	Automatisere	Utvikle script som automatiserer oppgaver.
5.	Dokumentasjon	Arbeid som skal dokumenteres i driftsdokumentasjon.

Tabell 2 - Prioritering av kategori

Videre ble de forskjellige konfigurasjonsblokkene som skulle utvikles prioritert etter sannsynligheten for hvor lett de kunne løses. Noen konfigurasjonsblokker var enkle å løse da det ikke trengtes samarbeid og avklaringer fra andre eller utvikling av egenkomponerte DSC-ressurser. Derfor ble det valgt å starte med de konfigurasjonsblokkene som var enkle. Denne prioriteringen ble tatt med tanke på å øke motivasjonen og få "Quick Wins" da kunnskapen og kompetansen i utgangspunktet var lav. De mer avanserte konfigurasjonsblokkene ble utført senere da prosjektgruppa hadde mer erfaring og kunnskap om hvordan teknologien fungerte.

I den praktiske gjennomføringen ble det arbeidet etter prinsippene om kontinuerlig forbedring. Det innebar å jobbe etter følgende metode: planlegge > Gjennomføre > Måle > Korrigere. Man planlegger hva som skal gjøres, gjennomfører planene man har lagt, måler og tester man hvordan det gikk, før man til slutt korrigerer og forbedrer. Dette gjentas helt til man er fornøyd.

4 Resultater

Dette kapittelet beskriver resultatene til arbeidet som er gjennomført i prosjektperioden.

Det er utviklet et sluttprodukt som benytter Azure DSC til å automatisk konfigurere Windows-servere og sikre at konfigurasjonen utilsiktet ikke blir endret over tid. Resultatene består hovedsakelig av tre deler:

- Erfaringer og kunnskap og med bruk av tjenesten Azure DSC.
- Egen- og videreutviklede DSC-ressurser som behøves for å dekke enkelte behov.
- Konfigurasjonsdokumentet som definerer konfigurasjonen for Session Hosts.

4.1 Bruk av Azure DSC

Tjenesten Azure DSC er tatt i bruk og konfigurert i henhold til tilgjengelig dokumentasjon og oppdragsgivers ønske. Det er testet funksjonalitet og gjennomførbarhet på en rekke av tjenestens funksjoner. De forskjellige punktene er beskrevet med status i Tabell 3.

Hva	**Status**
Opplastning og kompilering av konfigurasjonsdokument med PowerShell og i Azure Portalen. Både med og uten bruk av parametere.	OK
Håndtere konfigurasjonsdrift ved å automatisk rekonfigurere maskinen.	OK
Håndtere konfigurasjonsdrift ved å varsle i Azure Portalen.	OK
Opprettelse av ekstern legitimasjon med PowersShell og i Azure Portalen. Det er testet både lokal brukerkonto og domenelegitimasjon.	OK
Bruk av Run As Account som legitimasjon i konfigurasjon	Ikke testet
Opplastning av Moduler og DSC-ressurser i Azure Portalen.	OK
Oppdatere Azure Automation moduler i Azure Portalen.	OK
Onboarding av on-premises maskiner i Azure DSC.	OK
Tilpasset onboarding av on-premises maskiner i Azure DSC med bruk av konfigurasjonsdata.	OK
Onboarding av Azure-maskiner i Azure DSC.	Ikke testet
Testet funksjonalitet i utvidet logging.	Begrenset
Feilsøking og monitorering av konfigurasjonsstatus.	OK
Utvikle egne DSC-ressurser (Se kapittel 4.2 Egen- og videreutviklede DSC-ressurser).	OK
Utvikle kompositttressurser (Se kapittel 4.2 Egen- og videreutviklede DSC-ressurser).	OK
Utvikle Partial Configurations.	Ikke testet
Automatisering med Azure Automation Runbooks og versjonkontrollsystem	Ikke testet

Tabell 3 - Testet funksjonalitet i Azure DSC

Fremgangsmåte for installasjon og bruk av tjenesten, veiledning og driftsdokumentasjon er tilgjengelig i dokumentet «Driftsrapport».

4.2 Egen- og videreutviklede DSC-ressurser

Som vist i Tabell 4 er det utviklet og videreutviklet totalt fire egendefinerte DSC-ressurser og en kompositttressurs for å dekke oppdragsgivers behov for konfigurasjon av Session Hosts. Dette var et behov da det ikke fantes eksisterende ressurser med tilstrekkelig funksjonalitet.

Verktøyet xDSCResourceDesigner ble benyttet som bistand i utviklingen av DSC-ressurser. I tillegg til å sette opp fil- og mappestruktur inkluderer det også testverktøy som kontrollerer om ressursene er bygget opp korrekt i henhold til Microsofts definerte regler. Enkelte av ressursene er bygget videre på andres arbeid. Dette ble gjort der det for eksempel fantes en eksisterende ressurs som har manglende funksjonalitet og det er enkelt å legge inn den funksjonalitet oppdragsgiver har behov for. Der det er lånt kode er det spesifisert både i kildekode og i driftsdokumentasjon.

Navn	Funksjon	Status
CopyAcl	Kopierer Access Controll List fra et objekt til en annen.	OK
IPv6Address	Konfigurerer IPv6-Site-Local-adresse.	OK
PrinterDriver	Installerer og konfigurerer driver for skrivere.	OK
SessionHost	Konfigurerer en server til en Session Host og melder den inn i et Session Collection.	IKKE TESTET
SessionHostBaseline	Composite Resource som inneholder Baselinekonfigurasjon for Session Hosts	OK

Tabell 4 - Utviklede DSC-ressurser

Alle DSC-ressursene er testet OK, unntatt SessionHost. Denne ressursen benytter Windows Remote Management (WinRm)[1] for å sende PowerShell-kommandoer til Connection Broker for å gjøre seg selv til en Session Host og melde seg selv inn i et Session Collection. WinRm-trafikk er blokkert internt mellom forskjellige servere som resulterer i at funksjonaliteten i DSC-ressursen ikke fungerer.

Det er utviklet en kompositttressurs "SessionHostBaseline" som inneholder grunnkonfigurasjonen som er lik for alle Session Hosts. Denne ressursen ble laget for å teste funksjonalitet med bruk av kompositttressurser.

4.3 Konfigurasjonsdokumentet

Det er utviklet et dynamisk konfigurasjonsdokument som inneholder en rekke av funksjonaliteten Azure DSC tilbyr. Konfigurasjonsdokumentet brukes til å kompilere ulike konfigurasjoner i henhold til hvilken input-data som spesifiseres. Konfigurasjonen er tilpasset den funksjonaliteten som ble beskrevet i designdokumentet og NTNUITs interne dokumentasjon for terminalservertjenesten. Hvis det i fremtiden kommer nye operativsystem eller Session Collection som behøver spesialinnstillinger må dette tilpasses i konfigurasjonsdokumentet.

[1] https://docs.microsoft.com/en-us/windows/desktop/winrm/portal

Konfigurasjonsdokumentet har konfigurasjonsblokker som visst på Tabell 5. Tabellen inkluderer status og hvilken PowerShell-modul med hvilken ressurs som er benyttet til å løse problemet. Som man ser på tabellen er det tre konfigurasjonsblokker som ikke er OK. Installasjon av sertifikat og konfigurasjon av brannmur er ikke påbegynt etter avtale med oppdragsgiver. Administrere Session Host er ikke fullført på grunn av samme årsak som beskrevet i Kapittel 4.2 der WinRm-trafikk er blokkert internt mellom forskjellige servere slik at funksjonaliteten i ressursen ikke fungerer.

Konfigurasjonsblokk	**Løst av DSC-ressurs**	**Status**
Konfigurasjon av pagingfil	ComputerManagementDsc (VirtualMemory)	OK
Legge til IPv6-adresse	AzureDSC_Modules (IPv6Address)	OK
Installasjon av Windows Features	PSDesiredStateConfiguration (WindowsFeature)	OK
Deaktivere Windows Search	PSDesiredStateConfiguration (Search)	OK
Installere skriver	PrinterManagement (Printer)	OK
Installere skriverdriver	AzureDSC_Modules (PrinterDriver)	OK
Installere sertifikat	-	IKKE OK
Potensielt sikkerhetshull	AzureDSC_Modules (CopyAcl)	OK
Administrere Session Host	AzureDSC_Modules (SessionHost)	IKKE TESTET
Konfigurere brannmur	-	IKKE OK
Kopiere Matlab-script	(File)	OK

Tabell 5 - Konfigurasjonsblokker

5 Diskusjon

I dette kapittelet blir prosjektets resultater diskutert. Her gjennomgås prosjektgruppas erfaringer med Azure DSC, hvilke styrker, svakheter og eventuelle problemområder. Kapittelet er delt opp i fire underkapitler:

- Erfaringer ved bruk av Azure DSC
- Prosjektets implementering av Azure DSC
- Egen- og videreutviklede DSC-ressurser
- Konfigurasjonsdokumentet

5.1 Bruk av Azure DSC

Gjennom prosjektarbeidet er det erfart at Azure DSC fungerer veldig godt som konfigurasjonsstyringsverktøy. Inntrykket er at det holder det det lover ved at det gir en enkel og oversiktlig endringsprosess, det sikrer konfigurasjonsdrift og forenkler og automatiserer installasjonsprosessen. Konfigurasjonsdokumentet er selvdokumenterende og kan lett overføres og benyttes til nye problemstillinger på andre maskiner i fremtiden. Når et konfigurasjonsdokument er opprettet er det enkelt å bruke det som mal til å lage nye konfigurasjonsdokument. Portalen gir god oversikt over status på maskinene som gjør det lett å feilsøke hvis noe går galt. Det kan være at en maskin er ute av ønsket tilstand eller at en konfigurasjon feilet ved implementasjon. Ved bruk av PowerShell-kommandoer kan opplastning av konfigurasjon, testing, kompilering og tildeling av konfigurasjon på noder enkelt automatiseres ytterligere.

5.1.1 Krav til kompetanse

For å implementere og drifte løsningen er det behov for kompetanse og kunnskap på en rekke områder. Man bør ha kunnskap og forståelse om bruk av Azure Automation, Windows-operativsystemet og inngående erfaring med PowerShell. Dette er spesielt viktig når det kommer til implementasjon da dette krever mer kompetanse enn daglig drift.

Azure Portalen gjør implementasjon og drift av Azure DSC veldig enkelt. Det eneste man behøver å gjøre er å opprette en "Automation Account" så er man klar til å lage konfigurasjoner og administrere maskiner. Det er ikke behov for installasjon og konfigurasjon av Pull-server og sikring og / eller opprettelse av sjekksum av MOF-filene. Dette er alt sammen inkludert i tjenesten Azure DSC. Det er også veldig positivt som bruker å kunne velge mellom å utføre arbeidsoppgavene i et grafisk grensnitt eller i et kommandolinjeverktøy. Azure Portalen er enkel å bruke og logisk oppbygd. Den gir hjelpetekst, tips og link til dokumentasjon der det er behov.

Videre er det viktig med god kompetanse med PowerShell og hvordan operativsystemet fungerer for de som skal lage konfigurasjoner. Det å lage og idriftsette en DSC-konfigurasjon trenger ikke være veldig avansert, men det er viktig at man har kunnskap nok til å gå inn i kildekoden til den DSC-ressurs man benytter. Man må kunne se hva ressursen faktisk gjør og

forstå hvordan operativsystemet vil reagere, før den tas i bruk på produksjonsservere. Dette er spesielt viktig om man benytter seg av ressurser som er utviklet av andre enn Microsoft (Community Resources). De aller fleste DSC-ressursene er godt dokumentert og følger de reglene som Microsoft har definert, men man kan ikke være sikker før man har sett på kildekoden og forstår eksakt hva en ressurs gjør. Dette er også viktig hvis man skal tilbakestille en konfigurasjon, som rapporten kommer mer inn på i kapittel 5.1.3.

For å utvikle egenkomponerte DSC-ressurser behøves samme kompetanse som beskrevet i avsnittet over, men man må også i tillegg kunne bruke kompetansen til å utvikle nye scripts. Dette krever ofte mer enn om man bare skal forstå hvordan et script fungerer. Modulen "xDscResourceDesigner" bistår i å forenkle utviklingsoppgaven ved at den oppretter fil- og mappestrukturen rundt ressursen slik at man kun kan fokusere på selve utviklingen.

5.1.2 Hvor passer Azure DSC inn?

Azure DSC kan fungere bra sammen med men ikke som erstatning for dagens system for konfigurasjonsstyring. Azure DSC, Group Policy og Microsoft MDT løser forskjellige oppgaver og så lenge det ikke er konflikter der forskjellige system håndterer samme objekt. Azure DSC kan for eksempel ikke brukes til å installere operativsystem, lage et image eller fjerne funksjonalitet i brukergrensesnittet, men det funker bra for oppgaver som vanligvis kan utføres med PowerShell-kommandoer og script. Det kan godt hende at det er deler av det Group Policy og Microsoft MDT løser i dag som også kan bli utført av Azure DSC. Prosjektet har kun tatt for seg det som i dag har blitt utført manuelt.

Generelt sett fungerer Azure DSC bra til å bygge server-tjenester og baseline-konfigurasjon i dynamiske miljø med mye endringer der det er behov for å overvåke og sikre mot konfigurasjonsdrift. Et annet positivt aspekt med Azure DSC er at det ikke er behov for at maskinene er innmeldt i et domene for at de skal administreres som kan forenkle enkelte problemstillinger [14].

5.1.3 Driftsrutiner og sikkerhet

Dette kapittelet tar for seg erfaringene som er knyttet til driftsrutiner og sikkerhet som er opparbeidet gjennom prosjektperioden.

Endringshåndtering

Azure DSC sikrer effektiv og trygg endringshåndtering ved at alle endringene utføres ett sted og kan enkelt testes før implementering.

En ulempe er at det kan være lett å miste oversikten over hva slags konfigurasjon en maskin har hatt tidligere. En maskin tilbakestilles ikke til slik den var før konfigurasjon hvis konfigurasjonen fjernes. Hvis konfigurasjonen i Kodeblokk 2 er gjeldende og Windows Feature «RDS-RD-Server» ikke lenger skal være en del av konfigurasjonen er det ikke bare å fjerne linjene fra konfigurasjonsfila. Man må også manuelt avinstallere «RDS-RD-Server».

Eventuelt kan man endre egenskapen «Ensure» til «Absent» og implementere endringen på nytt.

```
Configuration ConfigurationExample
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration -name
WindowsFeature

    Node SessionHost
    {
        WindowsFeature RDS
        {
            Ensure = "Present" # Denne endres til "Absent"
            Name = "RDS-RD-Server"
        }
    }
}
```

Kodeblokk 3 - Endring av konfigurasjon

Eksempelet tar for seg et ganske simpelt senario ved installasjon / avinstallasjon av en Windows Feature, men det finnes mer avanserte eksempler der det kan være flytting av filer, setting av ACL, endringer i registry, opprettelse av filshare, oppgaver som utføres på domenet osv. Derfor er det veldig viktig at man går inn og ser på kildekoden og forstår fullstendig hva en DSC-ressurs faktisk gjør før den settes i produksjon.

Det eksisterer funksjonalitet i Azure portalen for å hente ut tidligere konfigurasjon på en maskin. Konfigurasjonsstatusen for hver konfigurasjonsblokk blir logget hver gang LCM initierer en «Dsc Consistency Check», men denne historikken blir raskt uoversiktlig da sjekken vanligvis gjøres hvert 30. minutt. Å finne den konfigurasjonsblokka man er ute etter kan være utfordrende.

Sikre konfigurasjonsdrift

En av de store fordelene til Azure DSC er at den automatisk kan kontrollere tilstanden til nodene og konfigurere den tilbake til ønsket tilstand. Dette har fungert veldig godt i praksis, men det er noen momenter man må være klar over. Er det en konfigurasjon som krever omstart av maskinen når den rekonfigureres av LCM vil maskinen restarte seg selv uten forvarsel. Dette er ofte ikke ønskelig for en server i produksjon. Problemstillingen kan løses med å benytte konfigurasjonsmodus (ConfigurationMode) «ApplyAndMonitor». Da vil LCM varsle Azure DSC om at konfigurasjonen ikke er i ønsket tilstand slik at man senere i et vedlikeholdsvindu kan gå inn å rekonfigurere maskinen tilbake i ønsket tilstand. En annen mulighet er eventuelt å benytte konfigurasjonsmodus (ConfigurationMode) «ApplyAndAutoCorrect» og sette «RebootNodeIfNeeded» til «False». Da vil LCM konfigurere serveren tilbake til ønsket tilstand, men avvente en eventuell restart av maskinen slik at dette kan gjøres manuelt.

Videre bør man være klar over at hvis maskinen har konfigurasjonsmodus (ConfigurationMode) «ApplyAndAutoCorrect», vil det ikke skrives logg når tilstanden blir rekonfigurert til ønsket tilstand. Det vil si at hvis man har problemer med uautoriserte endringer vil dette ikke bli oppdaget.

Konfigurasjon av Local Configuration Manager (LCM)

Generelt sett har konfigurasjon og drift av maskinene i prosjektet fungert tilfredsstillende. Men det er eksempler der konfigurasjonsmodus (ConfigurationMode) er endret fra "ApplyAndAutoCorrect" til "ApplyAndMonitor" der LCM fortsatt re-konfigurerer maskinen til ønsket tilstand i stedet for å kun rapportere om at den ikke er i ønsket tilstand. Maskinens LCM innstillinger viser at den er konfigurert med "ApplyAndMonitor", men altså fortsatt re-konfigurere hvis maskinen settes ut av ønsket tilstand.

Det meste er prøvd for å tvinge noden over i rett konfigurasjonsmodus, men det eneste som har lykkes er å avregistrere og nullstille maskinen.

Sikkerhet

Siden Azure DSC er en skytjeneste og tilgjengelig eksternt er det viktig at tilgang til tjenesten sikres tilstrekkelig. Dette er spesielt viktig hvis man skal administrere vital infrastruktur. Uten ekstra tilpasning er det kun brukernavn og passord som benyttes som autentisering i Azure. Det vil si at hvis en administrator får brukerkontoen sin kompromittert vil altså dette være nok til at en potensiell angriper kan kjøre ny konfigurasjon til alle servere.

For å sikre Azure DSC bør det minimum være to-faktor-autentisering for pålogging i Azure og det bør være restriksjoner i hvem som får tilgang og i hvilken grad de får tilgang. Azure Automation har et rollebasert tilgangssystem[2] som setter restriksjoner i hvilke oppgaver brukere får tilgang til å utføre. Beste praksis er ifølge Microsoft å gi restriktive tilganger til brukere slik at de bare akkurat får nok tilgang til å utføre sitt arbeid [14].

Siden Azure DSC er et kraftig verktøy som kan distribuere en konfigurasjon til alle maskiner samtidig, er det viktig at de som administrerer, vedlikeholder og bruker det er nøyaktige og kompetente nok til å gjøre dette på en trygg måte. Distribueres det en konfigurasjon som inneholder problemer vil ikke dette oppdages før alle maskinene er ferdig konfigurert og alle har feilen. Det er ikke funnet noen funksjonalitet i Azure DSC for å spore hvem som har lastet opp, kompilert og tildelt konfigurasjoner til noder. Dette kan medføre forlenget rettetid hvis man ikke finner ut hvem som har gjennomført endringen. Hvis det er ønskelig med sporing av hvem som utfører endringer i konfigurasjonsdokumentene anbefales det å benytte et versjonskontrollsystem. Da er det mulig å spore endringer i konfigurasjonsdokumentene der de er lagret lokalt. Slik gis muligheten til å se hvem som har utført endring, med en forklarende tekst og det gir mulighet til å enkelt reversere tilbake til tidligere versjoner av konfigurasjonsdokumentet.

5.1.4 Alternativer til Azure DSC

Hvilke alternativer til Azure DSC finnes og hva er forskjellen på disse? Vi har DSC Pull Service som er en del av PowerShell DSC. Det er en on-premises løsning som installeres på Windows Server. Den store forskjellen på Azure DSC og PowerShell DSC er at Azure DSC er et ferdig produkt, mens PowerShell DSC er en plattform. Det vil si at med Azure DSC vil man

[2] https://docs.microsoft.com/en-us/azure/automation/automation-role-based-access-control

få en ferdig fungerende løsning der det er lite eller ingen installasjon-, konfigurasjon- og vedlikeholdskostnader. Med PowerShell DSC får man kun byggeklossene slik at man enten må enten lage sine egne produkt, benytte andres verktøy eller gjøre en del av oppgavene manuelt. Et eksempel på dette er kompilering, flytting av filer og opprettelse av sjekksum for å tilgjengeliggjøre en konfigurasjon på Pull-serveren, som enkelt kan gjøres i én operasjon i Azure DSC. I tillegg må man selv installere, drifte og vedlikeholde DSC Pull Service-tjenesten som er plassert on-premises. Videre er det verdt og nevnte at Microsoft har anbefalt brukere overgang til Azure DSC siden den lokale Pull serveren ikke lengre vil videreutvikles med ny funksjonalitet av Microsoft [1].

For andre alternativer må man da se på produkter som er utviklet av andre enn Microsoft. Det er en rekke åpen kildekodeprosjekter som utvikler sin egen Pull- og Reporting-server-tjeneste som fungerer som et alternativ til skytjenesten Azure Automation. Dette er et resultat av at PowerShell DSC er lansert som åpen kildekode slik at utviklere kan lage sine egne løsninger. Et alternativ er Tug[3] som blir vedlikeholdt av brukermiljøet hos PowerShell.org.

Til slutt er det verdt å nevne at siden PowerShell DSC er en åpen plattform gir det muligheten for leverandører av andre konfigurasjonsstyringssystemer til å ta i bruk teknologien. De fleste leverandørene av konfigurasjonsstyringssystemer som tradisjonelt sett har blitt brukt til å konfigurere Linux-operativsystem og nettverksinfrastruktur har support for og gir muligheten til å administrere Windows-maskiner i sitt grensesnitt. De mest kjente alternativene her er Ansible, Puppet og Chef.

Prosjektet har ikke testet funksjonalitet på alternative systemer for konfigurasjonsstyring.

5.2 Implementering av Azure DSC

5.2.1 Bruk av egendefinerte DSC- og kompositttressurser

I implementeringsfasen er det testet det meste av den funksjonaliteten som Azure DSC tilbyr, for å løse oppdragsgivers behov. På noen punkter er det gjort mer arbeid enn hva det reelle behovet faktisk som et forsøk på å teste ut avansert funksjonalitet og vise oppdragsgiver hvilke muligheter Azure DSC gir. Det er for eksempel ikke nødvendig å utvikle egendefinerte DSC- og kompositttressurser i konfigurasjonen fordi bruk av Script-DSC-ressursen vil ha tilsvarende funksjon, men det er gir en del fordeler når det kommer til endringshåndtering og det skalerer godt når infrastrukturen blir større og mer avansert. Fordelene er at man får en sikrere endringshåndtering siden man administrerer koden ett sted, i selve ressursen, i stedet for at man håndterer flere ulike versjoner i ulike konfigurasjonsdokument [13]. Ulempene kan være noe høyere administrasjonskostnad siden man implementerer et ekstra element som må administreres og vedlikeholdes. Totalt sett må bruk av egendefinerte DSC- og kompositttressurser vurderes etter behov, kompetanse og forholdene på infrastrukturen som skal administreres.

[3] https://github.com/PowerShellOrg/tug

5.2.2 Utvidet logging

Prosjektet burde i større grad vurdert og sett på bruk av logging ved bruk av Azure Log Analytics Workspace. Det er mye spennende funksjonalitet her som burde vært utforsket som kunne kommet til nytte for oppdragsgiver. Eksempler på dette er automatisk varsling per epost hvis en node ikke er i ønsket tilstand med mer. Dette ble ikke prioritert i prosjektgjennomføringen til fordel for utvikling av konfigurasjonsdokumentet og DSC-ressurser.

5.2.3 Løsning for høyere grad av automatisering

I designrapporten kapittel 4.3 ble det beskrevet en løsning for høyere grad av automatisering. Den gikk ut på at konfigurasjonen ble lagret i et versjonshåndteringssystem som ble overvåket av Azure Automation Runbooks. En Runbook skulle overvåke og se etter endringer i konfigurasjonen, utføre tester av konfigurasjonen og automatisk oppdatere konfigurasjonen i Azure DSC. Denne løsningen ble ikke realisert etter avtale med oppdragsgiver siden det gikk utover prosjektets definerte rammer. Forslaget til løsning blir fortsatt stående i designrapporten slik at man er klar over mulighetene for en eventuell implementering i fremtiden.

5.3 Egen- og videreutviklede DSC-ressurser

Dette underkapittelet tar for seg diskusjon rundt resultatene for de egen- og videreutviklede DSC-ressursene.

Som beskrevet i kapittel 4.2 er enkelte av ressursene bygget videre på andres arbeid. I tillegg til å bygge videre er det også lånt script fra forskjellige steder for å lage en fullstendig operativ løsning. Dette ble gjort med begrunnelse i at DSC er åpen kildekode og det lønner seg å bygge videre på andres veldokumenterte og velfungerende arbeid i stedet for å finne opp hjulet på nytt hver gang. Microsoft anbefaler denne fremgangsmåten og forteller at DSC har blomstret etter at de lanserte DSC som Åpen kildekode [15].

5.4 Konfigurasjonsdokumentet

5.4.1 Dynamisk konfigurasjonsdokument

Konfigurasjonsdokumentet er utviklet med tanke på økt fleksibilitet siden det kan kompilere flere forskjellige kompilerte konfigurasjoner i henhold til de parameterne som spesifiseres. Det er gjort slik for å vise mulighetene som finnes i Azure DSC. En svakhet med dette er at det vil redusere oversikten og øke behovet for kompetanse for leseren, siden dette medfører at konfigurasjonsdokumentet blir større og inneholder elementer som ikke er relevant for det leseren er interessert i. Et alternativ til dynamiske konfigurasjonsdokument er å ha flate konfigurasjonsdokument for hver konfigurasjonsrolle. Dette vil nok lønne seg hvis konfigurasjonsdokumentene ved en reell implementering blir større og mer avanserte enn hva prosjektgruppa har utviklet. Flate konfigurasjonsdokument vil være en mer oversiktlig løsning,

men det vil resultere i flere filer som må vedlikeholdes. Ved en slik løsning vil det nok være hensiktsmessig å benytte et versjonskontrollsystem til å administrere konfigurasjonsdokumentene.

5.4.2 Automatisk kompilering og konfigurasjonsdata

Det er forsøkt utviklet et konfigurasjonsdokument som tar inn konfigurasjonsdata på samme måte som det tilpassede scriptet for onboarding av noder, som forklart i Driftsrapporten kapittel 2.3.4. Hvis dette hadde blitt fullført kunne man kompilert de mulige konfigurasjonene som var definert i konfigurasjonsdataen automatisk. Hvis man med dagens løsning legger inn en ny node med en kombinasjon av Session Collection og operativsystem som ikke er kompilert tidligere, vil ikke noden ha noen konfigurasjon å benytte og resulterer dermed i feilmelding. Man må altså kompilere konfigurasjonen manuelt før noden tar i bruk konfigurasjonen.

Dette ble ikke fullført på grunn av tidsnød i gjennomføringsfasen. Løsningen ble forkastet da den ikke var velfungerende ved gjennomføringsfasens slutt. Et alternativ vil være å utvide onboarding-scriptets funksjonalitet med å legge inn logikk som kontrollerer om den kompilerte konfigurasjonen eksisterer på Pull-serveren for så automatisk starte en kompileringsjobb med nye parametere hvis den ikke eksisterer.

6 Konklusjon

Gjennom arbeidet med prosjektet er det testet de forskjellige funksjonene som Azure DSC tilbyr, utviklet forskjellige DSC-ressurser og laget et konfigurasjonsdokument som automatisk installerer og implementerer forskjellige konfigurasjonselementer som er definert i NTNUs dokumentasjonssystem. Selv om ikke all funksjonalitet i Azure er testet og ikke all manuell konfigurasjon er automatisert, kan man si at prosjektgruppa har testet og tilegnet seg tilstrekkelig erfaring med tjenesten for at det skal komme NTNUIT til gode for videre utvikling og bruk av løsningen. Sluttproduktet viser hvordan en full implementering av Azure DSC vil fungere i praksis.

De viktigste erfaringene fra arbeidet er at Azure DSC egner seg godt som konfigurasjonsstyringssystem for de oppgavene som blir manuelt utført i dag. Men for å ta ut maksimalt utbytte av løsningen bør det gjøres en vurdering av det som blir tilpasset med andre konfigurasjonsstyringssystem i dag for å se om det er flere elementer som kan konfigureres av Azure DSC. Styrkene til Azure DSC er brukervennligheten i Azure Portalen med de monitorerings- og overvåkingsmulighetene som finnes, i tillegg til mulighetene for å reversere uautoriserte konfigurasjonsendringer automatisk.

6.1 Videre arbeid

Ved full implementering av Azure DSC i produksjon må det tas hensyn til de utfordringene som er belyst under driftsrutiner og sikkerhet i kapittel 5.1.3. Her er noen forslag for hvordan man bør gå frem.

Sikre Azure Portalen med to-faktor-autentisering. Azure DSC skal benyttes til å konfigurere kritisk infrastruktur og det er da særdeles viktig at tjenesten sikres tilfredsstillende.

Lag rutiner for fjerning av konfigurasjon fra konfigurasjonsdokumentet. Dette for å sikre at konfigurasjonen faktisk blir reversert ved endring.

Benytt konfigurasjonsmodus (ConfigurationMode) "ApplyAndMonitor" i starten. Da sikrer man mot uforutsette omstarter hvis maskinen blir automatisk re-konfigurert. Man kan eventuelt på et senere tidspunkt gå over til "ApplyAndAutoCorrect" når man får mer erfaring med bruk av tjenesten.

Start migreringen til Azure DSC med gradvis overgang. Session Hosts passer spesielt bra som utgangspunkt for videre testing og utprøving av teknologien. Dette skyldes at Session Hosts i praksis er like i konfigurasjonen og vil administreres med få konfigurasjonsdokument. I tillegg har Session Hosts tilgang til Internett og med det direkte tilgang til Azure DSC uten at man må åpne brannmur mot Azure. For migrering kan man starte med en Session Collection for senere å utvide når man ser at det fungerer tilfredsstillende.

Benytt et versjonshåndteringssystem for administrasjon av konfigurasjonsfilene og eventuelle egenlagde DSC-ressurser. Da ligger filene lagret i et sentralt oppbevaringssted som synkroniseres ned til de som har tilgang. Dette gjør det mulig å samarbeide om utvikling av dokumentene på en trygg måte. Alle endringer som lagres til oppbevaringsstedet blir logget slik at man kan følge endringene over tid og se hvem som har utført endringene.

6.2 Evaluering av arbeidet

Måloppnåelse i henhold til kapittel 1.3 Prosjektmål er bra. Ikke alle punktene som ble definert i designrapporten er fullført, men det er levert et sluttprodukt som viser løsningens funksjonalitet. Det kunne vært større grad av automatisering i løsningen som var et av de suksesskriteriene som ble kartlagt i interessentanalysen.

Læringsutbytte for prosjektgruppen har vært meget bra. Prosjektgruppa har tilegnet seg høyere kompetanse om konfigurasjonsstyringssystemer, prosjektarbeid og automatisering av Windows-servere med PowerShell.

Prosessen gjennom prosjektperioden har gått fint. I noen perioder kunne det vært fint å hatt en partner for faglige diskusjoner, men det har gått bra på grunn av god hjelp fra veileder og oppdragsgiver.

Organiseringen av prosjektet har fungert bra. Planene som ble lagt i forstudien og i planleggingsfasen er fulgt opp og det har ikke vært store revideringer. Timeføringen kunne vært mer detaljert da timene ikke ble ført opp mot aktivitet fra starten. Dette resulterte i en stor jobb før innlevering der timelista manuelt måtte gjennomgås for å opprette regnskapet som skulle leveres. Ellers har timeføring, statusrapporter, veiledningsmøter med innkalling og referat fungert akseptabelt. Arbeidsinnsatsen gjennom hele prosjektperioden har vært jevn og god med et snitt på 23 timers arbeid per uke.

Referanser

[1] "Desired State Configuration Pull Service" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/pull-server/pullServer, Apr. 11, 2018 [Feb. 3, 2019].

[2] "Desired State Configuration Overview for Decision Makers" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/overview/decisionmaker, Jun. 12, 2017 [Feb. 20, 2019].

[3] "Desired State Configuration Overview for Engineers" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/overview/dscforengineers, okt. 13, 2017 [Feb. 23, 2019].

[4] M. Gray & J. Levy. Microsoft Ignite 2015, Topic: "Heterogeneous Configuration Management Using Microsoft Azure Automation" McCormick Place, Chicago, IL., May 2015. Internett: https://azure.microsoft.com/en-in/resources/videos/microsoft-ignite-2015-heterogeneous-configuration-management-using-microsoft-azure-automation/, [Feb. 23, 2019]

[5] Morris, Kief, Brian Anderson, Jasmine Kwityn, Karen Montgomery, Rong Tang, and Rebecca Demarest. Infrastructure as Code: Managing Servers in the Cloud. First ed. Sebastopol, California, 2016.

[6] Chaganti, R. Pro PowerShell Desired State Configuration: An In-Depth Guide to Windows PowerShell DSC 2nd ed., Berkeley, CA. 2018,

[7] J. Helmic & J. Snover. Online, Topic: "Advanced PowerShell Desired State Configuration (DSC) and Custom Resources", mars 2015. Internett: https://mva.microsoft.com/en-us/training-courses/advanced-powershell-desired-state-configuration-dsc-and-custom-resources-8702

[8] J. Helmic & J. Snover. Online, Topic: "Getting Started With PowerShell Desired State Configuration (DSC)", mars 2015. Internett: https://mva.microsoft.com/en-us/training-courses/getting-started-with-powershell-desired-state-configuration-dsc-8672

[9] R. Smith Online, Topic: "Managing Configurations with Azure Automation DSC", Okt 2016. Internett: https://www.pluralsight.com/courses/azure-automation-dsc-managing-configurations

[10] "Configuring the Local Configuration Manager" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/managing-nodes/metaConfig, Des. 12, 2018 [Feb. 16, 2019].

[11] "DSC Resources" Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/resources, Des. 12, 2018 [Feb. 13, 2019].

[12] "Get-Set-Test» Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/get-test-set, Des. 12, 2018 [Feb. 13, 2019].

[13] “Composing DSC Configurations in Azure Automation State Configuration (DSC) using Composite Resources” Internett: https://docs.microsoft.com/en-us/azure/automation/compose-configurationwithcompositeresources. Aug. 21, 2018 [May. 7, 2019]

[14] “Compare group policy gpo and PowerShell Desired state configuration” Internett: https://docs.microsoft.com/en-us/azure/role-based-access-control/overview. Mar. 13, 2019 [May. 16, 2019]

[15] "DSC Resource-Kit Flourishes as open source": https://devblogs.microsoft.com/powershell/dsc-resource-kit-flourishes-as-open-source/, Jun. 23, 2015 [May, 08, 2019]

[16] " Managed Object Format (MOF): https://docs.microsoft.com/en-us/windows/desktop/wmisdk/managed-object-format--mof-, May. 31, 2018 [May, 18, 2019]

[17] "Glossary C": https://docs.microsoft.com/en-us/windows/desktop/wmisdk/gloss-c, May. 31, 2018 [May, 18, 2019]

Vedlegg A - Forstudierapport

Forstudierapport

Azure Automation State Configuration for administrasjon av Windows-servere

Bacheloroppgave i informatikk, drift av datasystemer

Institutt for datateknologi og informatikk

Norges teknisk-naturvitenskapelige universitet (NTNU)

Innlevering

3. Februar 2019

Øyvind Lotten Westrum

Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
3. Februar 2019	1.0	Første utkast	OLW
7. Februar	2.0	Endringer etter tilbakemelding på veiledningsmøte	OLW

Innholdsfortegnelse

1 Introduksjon

Dokumentet er forstudierapporten i faget IDRI3001 Bacheloroppgave i drift av datasystemer ved NTNU. Rapportens hensikt er å skape en felles forståelse mellom prosjektgruppen og oppdragsgiver og skal fungere som en kontrakt for leveransen. Den inneholder alle nødvendige opplysninger som skal kunne bidra med å ta en avgjørelse om prosjektet skal gjennomføres. Hovedpunktene er bakgrunn for prosjektet, prosjektmål, rammebetingelser som prosjektet er underlagt, hvilke suksessfaktorer og hvilke risikoer man må vurdere.

1.1 Terminologi

Session Host (RDSH) En rolle i Remote Desktop Services (RDS) som gir brukere mulighet til å koble seg til eksternt skrivebord og kjøre Windows-programvare eksternt.

Remote Desktop Services (RDS) Samlebetegnelse for en Windows Server tjeneste som gir mulighet for fjernkjøring av programmer.

Session Collection Samling av Session Hosts som gir mulighet for lastbalansering.

Terminalservermiljø Generelt begrep for et system tilsvarende Remote Desktop Services.

Azure Microsoft sin nettskyplattform.

Azure Automation Automasjonstjeneste i nettskyen for prosessautomatisering og konfigurasjonsstyring i Azure.

Azure Automation State Configuration Tjeneste som gir mulighet til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC) konfigurasjoner, importere DSC-ressurser og tilordne konfigurasjoner til noder.

Desired State Configuration (DSC) Styringsplattform i PowerShell som gjør at man kan installere, styre og konfigurere IT-systemene med kode.

Pull Server Tjeneste som tilgjengeliggjør konfigurasjoner, lar noder kontrollere den ønskede tilstand og rapportere tilbake om de er i ønsket tilstand.

Konfigurasjonsstyring Styring av innhold, endringer og status på en konfigurasjon.

On-Premise (Computing) Servere som er eid, kontrollert og plassert lokalt.

Nettsky / Sky (Computing) Infrastruktur, plattform eller programvare som leies, administreres og er plassert hos en tredjepart.

Active Directory (AD) Katalogtjeneste fra Microsoft.

Group Policy (GP) Verktøy for å implementere konfigurasjoner og innstillinger for brukere og datamaskiner.

Microsoft Deployment Toolkit (MDT) En samling av verktøy og prosesser som bistår i å automatisere installasjon av et operativsystem.

PowerShell Et verktøy og scriptspråk som gir mulighet til å automatisere og styre et operativsystem fra kommandolinjen.

2 Bakgrunn for prosjektet

Bakgrunnen for prosjektet er å se på mulighetene til å benytte Azure Automation State Configuration (Azure DSC) til å administrere Windows-servere. Det er en tjeneste i Azure som gir deg muligheten til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC) konfigurasjoner, importere DSC ressurser og tilordne konfigurasjoner til noder. Forskjellen mellom tradisjonell DSC og Azure DSC er at Pull serveren er flyttet inn i Azure Automation, slik at man ikke behøver å administrere Pull Serveren lokalt [1].

Microsoft har anbefalt brukere å starte overgangen til Azure DSC siden den lokale Pull serveren ikke lengre vil videreutvikles med ny funksjonalitet av Microsoft [2].

2.1 Beskrivelse av behov

Oppgaven er gitt av seksjon for IT-Drift ved NTNU IT, som er den sentrale IT-avdelingen ved Norges teknisk-naturvitenskapelige universitet (NTNU). De tjener over 40 000 studenter og nesten 8 000 ansatte med et vidt spekter av IT-tjenester. NTNU IT drifter cirka 500 Windows-servere og 500 Linux-servere, hvorav mesteparten er virtuelle.

Windows-gruppa ved seksjon for IT-Drift ønsker et sluttprodukt (Proof of Concept) i form av en ferdig konfigurert løsning for konfigurasjon av "Session Hosts" i et terminalservermiljø. Løsningen skal automatisk konfigurere en "Session Hosts" uten manuelle steg og sikre at konfigurasjonen utilsiktet ikke blir endret over tid. Oppgaven skal løses med Azure Automation State Configuration og man ønsker å kunne konfigurere servere både On-Premise og i skyen. Sluttproduktet skal inkludere komplett dokumentasjon av implementering, konfigurering og driftsrutiner. I tillegg til sluttproduktet ønskes det svar på følgende forskningsspørsmål:

- Azure Automation State Configuration vs. andre metoder for konfigurasjonsstyring - Hva er fordeler og ulemper, og hvordan kan det eventuelt fungere sammen?
- Vurdering av sikkerhet, stabilitet og sporbarhet.
- Hvilke krav til kompetanse kreves for å implementere og drifte løsningen?
- Skaleringsmuligheter også til andre områder enn "Session Hosts".
- Hvilke utfordringer finnes i forhold til driftsrutiner ved bruk av Azure Automation State Configuration?
- Egne betraktninger rundt teknologien.

2.2 Kort om dagens systemer og rutiner

Dagens systemer og rutiner baserer seg på bruk av godt innarbeidede tjenester og verktøy som delvis automatiserer installasjon og konfigurasjon. Nesten all installasjon og konfigurasjon blir utført via PowerShell-Script som kjøres sekvensielt i henhold til dokumentert rutinebeskrivelse, mens noe konfigurasjon blir utført via Group Policy (GPO).

Ved installasjon og konfigurasjon av en "Session Host" benyttes Microsoft Deployment Toolkit (MDT) for automatisk utrulling og installasjon av operativsystemet på serveren. Videre blir serveren tilpasset av Group Policy (GPO) etter hvor serveren er plassert i Active Directory (AD). Til slutt utføres manuelle tilpasninger via PowerShell-script før den legges inn i Session Collection.

3 Prosjektmål

Dette kapittelet beskriver mål som legger grunnlaget for gjennomføringen og resultatet av prosjektet.

3.1 Effektmål

Den ønskede effekten av prosjektet er definert i følgende effektmål:

- Øke seksjonens grad av automatisering ved å benytte Azure Automation for installasjon, konfigurasjon og drift av servertjenester på Windows-plattformen.
- Bedre kvaliteten på tjenestene ved å unngå avvik og konfigurasjonsdrift.

3.2 Resultatmål

For å konkretisere det ønskede sluttprodukt og for å vite når prosjektet er fullført, settes følgende resultatmål:

- Levere et sluttprodukt (Proof of Concept) der man automatisk installerer, konfigurerer og setter opp "Session Hosts" i et terminalservermiljø ved hjelp av Azure Automation State Configuration.
- Besvare forskningsspørsmål som er presentert i kapittel 2.1 Beskrivelse av behov.

Resultatmålene har en tidsramme på 500 timer og har frist til å leveres den 20. Mai 2019.

3.3 Prosessmål

Prosjektgruppen har satt følgende prosessmål for prosjektgjennomføringen:

- Gjennomføre et større prosjekt forankret i en reell problemstilling fra arbeidslivet.
- Øke kompetansen innen automatisering av driftsoppgaver ved bruk av PowerShell.

3.4 Prosjektets omfang

Prosjektet gjennomføres som en test av ny teknologi for å undersøke hvordan den kan benyttes for å oppnå de effektmål som er definert i kapittel 3.1 Effektmål. Sluttproduktet (Proof of Concept) sammen med besvarelse på forskningsspørsmål skal bistå oppdragsgiver i å ta avgjørelse om dette er teknologi som det er verdt å investere tid og energi på for å undersøke nærmere.

- Sluttproduktet skal kjøres på et eget avgrenset testcollection med to "Session Hosts" i produksjonsmiljøet.
- Servere i testcollection overleveres med OS installert. All videre konfigurasjon for at serveren skal bli en "Session Host" skal løses av prosjektet.
- Det er hovedsakelig konfigurasjon under "Session Host - Manuelle tilpasninger, installasjon og oppsett" fra NTNU ITs dokumentasjon som det skal fokuseres på.
- Se på konfigurasjon av brannmur hvis det blir tid.

4 Prosjektets milepæler og hovedaktiviteter

Som vist på figur 1 – Gantt-diagram er prosjektperioden delt inn i fire faser.

4.1 Forprosjekt

Prosjektperioden starter med et forprosjekt som skal bidra til at prosjektgruppen og oppdragsgiver skal få en felles oppfatning om hva prosjektet skal produsere. Den skal på et overordnet nivå definere hva som skal gjøres, hva som skal være resultatet og hvilke rammebetingelser prosjektet er underlagt. Fasen avsluttes ved at både prosjektgruppen og oppdragsgiver godkjenner forprosjektrapporten.

4.2 Planleggingsfase

I neste fase, planleggingsfasen foretas det undersøkelser og tester, planlegging av detaljene til løsningen og dokumentasjon for hva som skal implementeres. Fasen avsluttes med en designrapport som skal legge grunnlaget for implementeringsfasen. Det er naturlig at prosjektplanen endres og blir mer detaljert etter denne fasen.

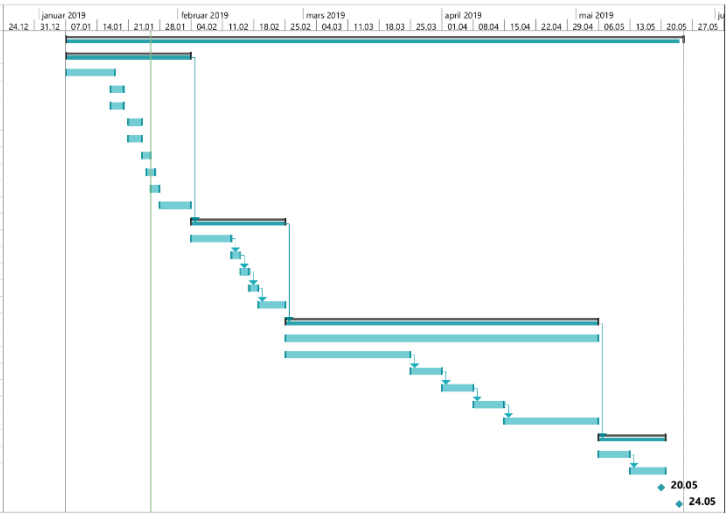
4.3 Implementeringsfase

Implementeringsfasen består av realisering av planene, testing og eventuell revidering og sluttføring. Denne fasen består i å dokumentere implementeringen i en driftsrapport som viser hvordan løsningen implementeres.

4.4 Avslutningsfase

Den siste fasen er avslutningsfasen. Her skal det skrives sluttrapport som beskriver løsningen overordnet inkludert besvarelse på forskningsspørsmålene som er beskrevet i problembeskrivelsen. Det er denne rapporten som skal leveres. All tidligere produsert dokumentasjon, forstudierapport, designrapport og driftsrapport skal ligge som vedlegg til besvarelsen.

ID	Aktivitetsnavn	Varighet	Start	Slutt
1	**Prosjektperiode**	**138 dager**	**ma 07.01.19**	**fr 24.05.19**
2	**Forstudierapport**	**28 dager**	**ma 07.01.19**	**sø 03.02.19**
3	Informasjonsinnhenting	11 dager	ma 07.01.19	to 17.01.19
4	Problemstilling	3 dager	to 17.01.19	lø 19.01.19
5	Risikoanalyse	3 dager	to 17.01.19	lø 19.01.19
6	Retningsliner og standarder	3 dager	ma 21.01.19	on 23.01.19
7	Prosjektets organisering	3 dager	ma 21.01.19	on 23.01.19
8	Prosjektmål	2 dager	to 24.01.19	fr 25.01.19
9	Interessenter og rammebetingeler	2 dager	fr 25.01.19	lø 26.01.19
10	Kritiske suksessfaktorer	2 dager	lø 26.01.19	sø 27.01.19
11	Ferdigstille forstudierapport	7 dager	ma 28.01.19	sø 03.02.19
12	**Designrapport (Planleggingsfase)**	**21 dager**	**ma 04.02.19**	**sø 24.02.19**
13	Undersøkelser og tester	9 dager	ma 04.02.19	ti 12.02.19
14	Bestemme løsning	2 dager	on 13.02.19	to 14.02.19
15	Planlegge løsning	2 dager	fr 15.02.19	lø 16.02.19
16	Revidere løsning	2 dager	sø 17.02.19	ma 18.02.19
17	Dokumentasjon av løsning	6 dager	ti 19.02.19	sø 24.02.19
18	**Driftsrapport (Implementeringsfase)**	**70 dager**	**ma 25.02.19**	**sø 05.05.19**
19	Dokumentasjon	70 dager	ma 25.02.19	sø 05.05.19
20	Implementering	28 dager	ma 25.02.19	sø 24.03.19
21	Testing	7 dager	ma 25.03.19	sø 31.03.19
22	Revidering	7 dager	ma 01.04.19	sø 07.04.19
23	Sluttføring	7 dager	ma 08.04.19	sø 14.04.19
24	Besvare oppgavens forskningsspørsmål	21 dager	ma 15.04.19	sø 05.05.19
25	**Hovedrapport (Avsluttningsfase)**	**15 dager**	**ma 06.05.19**	**ma 20.05.19**
26	Sluttrapport	7 dager	ma 06.05.19	sø 12.05.19
27	Ferdigstille dokumenter	8 dager	ma 13.05.19	ma 20.05.19
28	Innlevering	0 dager	ma 20.05.19	ma 20.05.19
29	Presentasjon	0 dager	fr 24.05.19	fr 24.05.19



Figur 4 - Gantt-diagram

5 Interessentanalyse

Interessentanalyse benyttes for å identifisere hvilke behov de forskjellige interessentene har, hva som skal til for at de skal bli tilfreds med prosjektresultatet og hva de bidrar med i prosjektarbeidet.

5.1 Identifisering av interessenter

Oppdragsgiver er Windows-gruppen ved Seksjon for IT-Drift på NTNU. De har ansvar for terminalservertjenester, støttesystemer, utskriftstjenester og lisensservere som kjører på Windows-plattformen hos NTNU.

Prosjektgruppen består av Øyvind Lotten Westrum som er tredjeårsstudent ved informatikk: spesialisering i drift av datasystemer på NTNU.

Veileder er Jostein Lund. Han er universitetslektor ved Institutt for datateknologi og informatikk og underviser i mange fag ved instituttet og veileder en rekke bachelorprosjekt hvert år.

5.2 Interessentanalyse

I dette underkapittelet defineres de suksesskriteriene og de bidrag interessentene har til prosjektet. Suksesskriterier er hva som skal til for at interessenten skal bli fornøyd med sluttresultatet av prosjektet.

Interessent	**Suksesskriterier**	**Bidrag til prosjektet**
	Eksterne	
Oppdragsgiver	• Ferdig sluttprodukt som viser konseptet i praksis. • God faglig besvarelse på forskningsspørsmål. • Dokumentasjon med høy kvalitet og med stor grad av kunnskapsoverføring • Ønsket ideal: så mye automatisering som mulig	• Prosjektoppgave • Testmiljø / Lab • Kunnskap og erfaring
	Interne	
Prosjektgruppen	• Spennende og faglig utfordrende prosjektperiode. • Arbeide jevnt igjennom hele prosjektperioden	Gjennomføre selve arbeidet Planlegge → Utføre → Teste → Justere Besvare problemstillingen
Veileder	• Godt gjennomført prosjektarbeid • Jevn oppfølging	Kunnskap, erfaring og veiledning innen prosjektgjennomføring

Tabell 6 - Interessentanalyse

6 Rammebetingelser

Rammebetingelser defineres for å konkretisere de absolutte krav som stilles til prosjektgjennomføringen og sluttresultatet.

6.1 Økonomiske rammer

Det er ikke forbundet noen kostnader ved prosjektet og gjennomføringen og har derfor ingen økonomiske rammer.

6.2 Tidsmessige rammer

- Prosjektet har krav om ferdigstillelse 20. Mai 2019. Dette inkluderer sluttproduktet og de rapporter som skal inkluderes i sluttrapporten.
- Det er krav om 500 timers arbeid igjennom prosjektperioden.
- Det skal holdes presentasjon av prosjektet senest én uke etter leveringsdato.

6.3 Kravspesifikke rammer

- "Session Hosts" skal konfigureres med Azure Automation State Configuration i henhold til dokumentert rutinebeskrivelse.
- Det skal foreligge komplett dokumentasjon av implementering, konfigurering og driftsrutiner for sluttproduktet.

7 Kritiske suksessfaktorer

Suksessfaktorer og informasjonsbehov bidrar med å sikre at prosjektet gjennomføres med suksess og at alle interessentene informeres med relevant informasjon.

7.1 Suksessfaktorer

For å rette fokus på de viktige forholdene i prosjektgjennomføringen defineres det suksessfaktorer. Dette er forhold som enten er relevante for oppdragsgiver, for prosjektgruppa eller begge parter. Følgende kritiske suksessfaktorer er definert:

- Prosjektet skal bidra til lave transaksjonskostnader for NTNU IT ved en senere eventuell full implementering av Azure Automation State Configuration på Session Hosts.
- Mulighet for skalering til andre problemstillinger enn "Session Hosts"
- Forsøke å følge "The Unofficial PowerShell Best Practices and Style Guide" for å skrive oversiktlig og forståelig script / kode slik at den enkelt kan videreutvikles senere.

7.2 Informasjonsbehov

Det vil være ukentlig arbeid som utføres i NTNU ITs lokaler på Sluppenveien 14. Derfor blir det en naturlig uformell informasjonsformidling i møte med oppdragsgiver på disse tidspunktene. Formelle møter innkalles og tas ved behov. Det er planlagt veiledningsmøter med Veileder / Faglærer hver 14. Dag. Timelister og ukesrapporter legges direkte ut på SharePoint Online. I tillegg til dette er det definert følgende regler for informasjonsformidling som utløses av gitte situasjoner:

Formål / Utløsende faktor	**Mottakere**	**Form**	**Tidspunkt**
Forsinkelse i fase	Alle	Ukesrapport	Førstkommende søndag.
Stor endring i sluttprodukt	Alle	E-post	Iht. Kapittel 9.4 Endringshåndtering.
Tidsbrist	Veileder / Faglærer	E-post / telefon	Så raskt som mulig.

Tabell 7 - Informasjonsbehov

8 Risikoanalyse

Risikoanalyse skal identifisere eventuelle farer, sårbarheter og risiko knyttet til prosjektgjennomføringen. Det skal fungere som et hjelpemiddel for å avdekke uforutsette og uønskede hendelser slik at man kan forhindre at de forekommer og ha en plan for hva man skal gjøres hvis de faktisk oppstår. Det skal også være med på å ta en avgjørelse om man skal gå videre med prosjektet.

8.1 Risikomatrise

Risikomatrisen er representert i figur 2 – Risikomatrise. Risiko (R) er produktet av sannsynligheten (S) for at en hendelse inntreffer og hva konsekvensene (K) er hvis det skjer. Akseptkriteriet er satt til fire som betyr at man aksepterer risikoen og ingen tiltak er nødvendig. Hendelser med risikoverdi mellom fem og ni vurderer man eventuelle tiltak og hendelser med risikoverdi over ti bør innføres.

Svært stor 5 *Daglig*	5	10	15	20	25
Stor 4 *Ukentlig*	4	8	12	16	20
Middels 3 *Månedlig*	3	6	9	12	15
Liten 2 *Semester*	2	4	6	8	10
Svært liten 1 *Årlig*	1	2	3	4	5
	1 Ikke alvorlig *Lite merarbeid*	2 Mindre alvorlig *Merarbeid*	3 Alvorlig *Mye merarbeid*	4 Kritisk *Vil ikke komme i mål*	5 Meget kritisk *Kan ikke fortsette*

Figur 5 - Risikomatrise

8.2 Uønskede hendelser

De ulike uønskede hendelsene blir vurdert med utgangspunkt i risikomatrisen.

Nr.	Uønsket hendelse	S	K	R	Forklaring
1	Uforutsett Korttidsfravær	2	2	4	Reduksjon i produksjon i en kortere periode. Vil kunne jobbe, men med lavere produktivitet.
2	Uforutsett Langtidsfravær	1	4	4	Langtidssykdom eller familiære situasjoner. Kritisk konsekvens siden det bare er ett gruppemedlem.
3	Nedsatt arbeidskapasitet	2	4	8	Lite overskudd til prosjektarbeid pga. 100% jobb ved siden av studier.
4	Tilgangsproblemer	2	5	10	Ingen tilgang til Microsoft Azure pga. Problemer hos Microsoft (Basert på tidligere erfaringer)
5	For lite arbeidstid og samarbeid hos NTNU IT	2	4	8	For lite arbeidstid og samarbeid kan føre til at det blir for dårlig innblikk i problemstilling og hva ønsket sluttprodukt skal bestå av.
6	Utilgjengelig veileder ved NTNU IT	1	4	4	Dårlig innblikk i problemstilling og hva sluttproduktet skal bestå av og hvilken kompetanse som behøves
7	Utilgjengelig veileder / faglærer	1	3	3	Dårligere innblikk i hvordan prosjektgjennomføring skal utføres. Ingen tilbakemelding på skrevet tekst.
8	Korrupte datafiler	3	5	15	Oppgavefiler blir korrupt før innlevering. Forstudierapport, designrapport, driftsrapport og sluttrapport. Script / kode som er utviklet
9	Tap av datafiler	3	5	15	Oppgavefiler går tapt før innlevering. Forstudierapport, designrapport, driftsrapport og sluttrapport Script / kode som er utviklet
10	Dårlig fremdrift / motivasjon	3	3	9	Dårlig fremdrift iht. Prosjektplanen og dårlig kvalitet på produsert materiell
11	Problemer med utstyr / lab / systemer	2	4	8	Uforutsette problemer med PC, labutstyr, Azure og / eller SharePoint Online
12	Dårlig planlegging	2	4	8	For dårlig planlegging som resulterer i sluttproduktet ikke gjennomføres eller at forskningsspørsmål ikke blir besvart.

Tabell 8 - Uønskede hendelser

8.3 Tiltak

For uønskede hendelser med risiko over 4 utarbeides det tiltak for å redusere risikoen.

Nr.	R	Tiltak
3	8	Søke og få godkjent 20 % permisjon fra jobb. Lage gode arbeidsplaner som legger opp til kvelds- og helgearbeid. Eventuelt ta ut feriedager for å jobbe med prosjektet. Ikke ta på seg ekstra arbeidsoppgaver på jobb i prosjektperioden.
4	10	Svare opp raskt og gi eventuell ønsket dokumentasjon til Microsofts supportmedarbeidere. Henvise til tidligere sak.
5	8	Arbeide i kontorlandskapet til Seksjon for IT-Drift på tilgjengelig arbeidsplass de dagene jeg skal ha permisjon fra jobb.
8	15	Benytte skytjeneste (SharePoint Online) med versjonskontroll. Benytte versjonskontrollsystem (git) for utvikling av script / kode. I tillegg ta ukentlig sikkerhetskopi som lagres hos alternativ skytjeneste.
9	15	Benytte skytjeneste (SharePoint Online) med versjonskontroll. Benytte versjonskontrollsystem (git) for utvikling av script / kode. I tillegg ta ukentlig sikkerhetskopi som lagres hos alternativ skytjeneste.
10	9	Gi tidlig beskjed til Veileder / faglærer. Sette opp milepæler i prosjektplanleggingen for å motivere for fremgang. Ta fri når det er behov. Gjøre andre ting enn jobb / prosjektarbeid når man har mulighet.
11	8	Ha klar alternativ PC. Endre tidlig til Google Drive hvis SharePoint Online ikke holder mål.
12	8	Bruke god tid i starten til å sette seg inn i problemstilling / tema. Revidere og oppdatere planen med status fortløpende. Skrive ukentlige statusrapporter og føre timer kontinuerlig.

Tabell 9 - Tiltak

9 Retningslinjer og standarder

Her blir de retningslinjer og standarder prosjektet må forholde seg til presentert. Dette er krav til dokumentasjon, krav til kvalitetsgjennomgang, krav til standarder og metoder og endringshåndtering.

9.1 Krav til dokumentasjon

Dette er den dokumentasjonen som skal produseres i løpet av prosjektperioden.

Krav til dokumentasjon	
Timeføring	Daglig timeføring i timeføring på SharePoint Online rom
Statusrapport	Ukentlig statusrapport på SharePoint Online rom
Prosjektplanlegging	Gantt diagram med ukentlige revisjoner
Dokumentasjon / Rapporter	Forstudierapport (3. Februar) Designrapport (24. Februar) Driftsrapport (5. Mai) Sluttrapport (20. Mai) Prosjekthåndbok (20. Mai)
Godkjenning og revisjon	Godkjenning sendes på e-post. Revisjon gjøres fortløpende og dokumenters i revisjonstabell.
Innlevering	Inspera Assessment (https://ntnu.inspera.no/) 1. Oppgaverapporten skal leveres som ett enkelt dokument med klikkbar innholdsfortegnelse. 2. Vedlegg (Presentasjon, kildekode og lignende) skal lastes opp som én zip-fil. Innleveringsfrist: **20. Mai**

Tabell 10 - Krav til dokumentasjon

9.2 Krav til kvalitetsgjennomgang

Her beskrives rutiner for kvalitetsgjennomgang for å sikre at prosjektgruppa og oppdragsgiver har samme forståelse av hva prosjektet skal levere igjennom hele prosjektet.

Krav til kvalitetsgjennomgang	
Revidering	Det er planlagt revideringsaktivitet i både planlegging og implementeringsfasen.
Kontinuerlig forbedring	Arbeide med kontinuerlig forbedring som metode: Planlegge → Gjennomføre → Måle → Korrigere
Samarbeid med oppdragsgiver	Ukentlige uformelle møter for kontroll med oppdragsgiver for å kvalitetssikre sluttproduktet.
Veiledningsmøte med faglærer	Holde veiledningsmøter med status hver 14. Dag.

Tabell 11 - Krav til kvalitetsgjennomgang

9.3 Krav til standarder og metoder

Her beskrives de konkrete standardene, metodene og verktøyene som skal benyttes i prosjektperioden.

Krav til standarder og metoder	
Dokumentmal	Tittelside Prosjekthåndbok Forstudierapport Designrapport Driftsrapport Sluttrapport Ukesrapport Timelister Møtereferat og møteinnkalling
Programvare	Microsoft SharePoint Online Microsoft Word Microsoft Outlook Microsoft Project Microsoft Planner Online Microsoft PowerShell ISE Microsoft Code
Metode	Følge NTNU ITs rutiner for dokumentasjon Følge NTNU ITs rutiner for scripting med PowerShell Kanban Board til arbeidsorganisering

Tabell 12 - Krav til standarder og metoder

9.4 Endringshåndtering

For å sikre at endringsønsker håndteres på en kvalitetssikker måte, er rutine for endringshåndtering definert i tabell 8 - Endringshåndtering.

Endringshåndtering	
Endringsbestiller	Oppdragsgiver, veileder eller prosjektgruppe
Fremgangsmåte	1. Dokumentere endringens innhold 2. Analysere konsekvensene for prosjektet 3. Godkjennelse og aksept 4. Logge endringen 5. Justere planleggingen 6. Informere interessentene 7. Gjennomføre endringen

Tabell 13 - Endringshåndtering

10 Prosjektorganisering

De involverte personene i prosjektet er visst i tabell 9 – Prosjektorganisering.

Involverte personer	
Prosjektgruppe / Student	Øyvind Lotten Westrum
Veileder / Faglærer	Jostein Lund
Oppdragsgiver	NTNU IT, Seksjon for IT-Drift
Veiledere ved NTNU IT	Brynjulf Mauring Jon Arne Reitan

Tabell 14 - Prosjektorganisering

11 Anbefaling for videre arbeid

Ut ifra de forhold som er avdekket i forstudien anbefales videre arbeid med prosjektet. Prosjektet kan fortsette med de planer og de rammebetingelser som er presentert i rapporten.

Problemstillingen er aktuell for både oppdragsgivers definerte effektmål og prosjektgruppens prosessmål. I tillegg er de tidsmessige rammene på 500 timers arbeid passende med prosjektets omfang. Det er også mulighet å utvide prosjektets omfang hvis det blir tid til det. Risikoanalysen har avdekket en rekke uønskede hendelser, men ved å innføre og følge opp de tiltak som er definert kan prosjektet gjennomføres med lav risiko.

Referanser

[1] "Azure Automation State Configuration Overview" Internet: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-overview, Jun. 11, 2018 [Feb. 3, 2019].

[2] "Desired State Configuration Pull Service" Internet: https://docs.microsoft.com/nb-no/powershell/dsc/pull-server/pullServer, Apr. 11, 2018 [Feb. 3, 2019].

Vedlegg B – Designrapport

Designrapport

Azure Automation State Configuration for administrasjon av Windows-servere

Bacheloroppgave i informatikk, drift av datasystemer

Institutt for datateknologi og informatikk

Norges teknisk-naturvitenskapelige universitet (NTNU)

Innlevering

24. Februar 2019

Øyvind Lotten Westrum

Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
24. februar 2019	1.0	Første utkast	OLW
28. februar 2019	1.1	Revisjon etter tilbakemelding fra oppdragsgiver	OLW
11. mai 2019	2.0	Klargjøring ifm. innlevering.	OLW

Innholdsfortegnelse

1 Introduksjon

NTNU IT ønsker å se på muligheten til å bruke Azure Automation State Configuration (Azure DSC) til å automatisere installasjon, oppsett og konfigurasjon av «Session Hosts» i et terminalservermiljø. Det skal utvikles et sluttprodukt (Proof of Concept) i form av en ferdig konfigurert løsning. Azure DSC er Microsoft sin skytjeneste for konfigurasjonsstyring og gir muligheten til å automatisk konfigurere servere og sikre at konfigurasjonen ikke endres over tid.

1.1 Dokumentets hensikt

Dokumentet er designrapporten i faget IDRI3001 Bacheloroppgave i drift av datasystemer. Rapportens hensikt er å vise planleggingen og de tekniske detaljene rundt det som skal gjennomføres i implementeringsfasen. Målet ved designrapporten er at den skal bidra med at prosjektgruppen og oppdragsgiver har samme opplysninger om hva som skal gjennomføres. Oppdragsgiver får her muligheten til å ta en vurdering om dette er hva de ønsker og komme med eventuelle tilbakemeldinger om endringsønsker. Dette er en del av kvalitetsgjennomgangen som er beskrevet i forstudierapporten.

1.2 Avgrensning

Det skal benyttes Azure Automation State Configuration, men annen konfigurasjonsstyring kan benyttes hvis det er hensiktsmessig. Prosjektet er avgrenset til å kun se på konfigurasjon av Session Hosts i et terminalservermiljø. Det vil si at annen infrastruktur i terminalservermiljøet ikke vil bli vurdert.

Tilgjengelige servere i det lokale testmiljøet blir levert med operativsystem installert. All videre konfigurasjon for at serveren skal bli en “Session Host” skal løses av prosjektet.

1.3 Terminologi

Session Host (RDSH) En rolle i Remote Desktop Services (RDS) som gir brukere mulighet til å koble seg til eksternt skrivebord og kjøre Windows-programvare eksternt.

Remote Desktop Services (RDS) Samlebetegnelse for en Windows Server tjeneste som gir mulighet for fjernkjøring av programmer.

Session Collection Samling av Session Hosts som gir mulighet for lastbalansering.

Terminalservermiljø Generelt begrep for et system tilsvarende Remote Desktop Services.

Azure Microsoft sin nettskyplattform.

Azure AD Microsofts skytjeneste for identitet- og tilgangsstyring.

Azure Automation Automasjonstjeneste i nettskyen for prosessautomatisering og konfigurasjonsstyring i Azure.

Azure Automation State Configuration Tjeneste som gir mulighet til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC) konfigurasjoner, importere DSC-ressurser og tilordne konfigurasjoner til noder.

Desired State Configuration (DSC) Styringsplattform i PowerShell som gjør at man kan installere, styre og konfigurere IT-systemene med kode.

Pull Server Tjeneste som tilgjengeliggjør konfigurasjoner, lar noder kontrollere den ønskede tilstand og rapportere tilbake om de er i ønsket tilstand.

DSC Configuration PowerShell script som definerer konfigurasjon, node, ressurs og hvilke attributter ressursen skal konfigureres med.

DSC Node Configuration En kompilert DSC konfigurasjon. Fungerer som en rolle som kan tildeles en server.

DSC Node En maskin/enhet hvor konfigurasjonen er administrert av DSC

DSC Resource Det man bruker for å definere DSC konfigurasjonen (ressursene)

Compilation Job Å kompilere en konfigurasjon til MOF-fil og gjøre den tilgjengelig på pull-serveren.

Local Configuration Manager (LCM) Maskineriet i DSC. Funksjonaliteten som henter, leser og eksekverer konfigurerer på noden.

Managed Object Format (MOF) Format som brukes til å beskrive en konfigurasjon. Brukes av DCS som konfigurerer en node.

Konfigurasjonsstyring Styring av innhold, endringer og status på en konfigurasjon.

On-Premise (Computing) Servere som er eid, kontrollert og plassert lokalt.

Nettsky / Sky (Computing) Infrastruktur, plattform eller programvare som leies, administreres og er plassert hos en tredjepart.

Active Directory (AD) Katalogtjeneste fra Microsoft.

Group Policy (GP) Verktøy for å implementere konfigurasjoner og innstillinger for brukere og datamaskiner.

Microsoft Deployment Toolkit (MDT) En samling av verktøy og prosesser som bistår i å automatisere installasjon av et operativsystem.

PowerShell Et verktøy og scriptspråk som gir mulighet til å automatisere og styre et operativsystem fra kommandolinjen.

PowerShell Script Samling av PowerShell-kommandoer som automatiserer en oppgave.

PowerShell Cmdlets Et PowerShell-script som utfører en funksjon.

2 Kort om oppdragsgiver og behov

Oppdragsgiver er seksjon for IT-Drift ved NTNU IT, som er den sentrale IT-avdelingen ved Norges teknisk-naturvitenskapelige universitet (NTNU). De tjener over 40 000 studenter og nesten 8 000 ansatte med et vidt spekter av IT-tjenester. NTNU IT drifter cirka 500 Windows-servere og 500 Linux-servere, hvorav mesteparten er virtuelle.

Windows-gruppa ved seksjon for IT-Drift ønsker et sluttprodukt (Proof of Concept) i form av en ferdig konfigurert løsning for konfigurasjon av "Session Hosts" i et terminalservermiljø. Løsningen skal automatisk konfigurere en "Session Hosts" uten manuelle steg og sikre at konfigurasjonen utilsiktet ikke blir endret over tid. Oppgaven skal løses med Azure Automation State Configuration (Azure DSC) og man ønsker å kunne konfigurere servere både On-Premise og i skyen. Sluttproduktet skal inkludere komplett dokumentasjon av implementering, konfigurering og driftsrutiner.

Oppdragsgiver ønsker et sluttprodukt med så mye automatisering som mulig. Dette ønsket er et resultat av at man vil å bruke mer tid på andre og mer interessante arbeidsoppgaver. Men det er også for å sikre mot menneskelige feil, vanvare og muligheten til å reversere uautoriserte endringer i konfigurasjonen automatisk.

2.1 Dagens løsning

Dagens systemer og rutiner baserer seg på bruk av godt innarbeidede tjenester og verktøy som delvis automatiserer installasjon og konfigurasjon. Nesten all installasjon og konfigurasjon blir utført via PowerShell-Script som kjøres sekvensielt i henhold til dokumentert rutinebeskrivelse, mens noe konfigurasjon blir utført via Group Policy (GPO). Dokumentasjon og rutinebeskrivelse for installasjon, konfigurasjon og drift av løsningen ligger lagret i oppdragsgivers Wiki-løsning som er tilgjengelig for alle IT-ansatte i NTNU.

Ved installasjon og konfigurasjon av en "Session Host" benyttes Microsoft Deployment Toolkit (MDT) for automatisk utrulling og installasjon av operativsystemet på serveren. Videre blir serveren tilpasset av Group Policy (GPO) etter hvor serveren er plassert i Active Directory (AD). Til slutt utføres manuelle tilpasninger via PowerShell-script før den legges inn i Session Collection.

3 Azure Automation State Configuration

Bakgrunnen for prosjektet er å se på mulighetene til å benytte Azure DSC til å administrere Windows-servere. Det er en tjeneste i Azure som gir deg muligheten til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC)-konfigurasjoner, importere DSC ressurser og tilordne konfigurasjoner til noder. Forskjellen mellom tradisjonell DSC og Azure DSC er at Pull serveren er flyttet inn i Azure Automation, slik at man ikke behøver å administrere Pull Serveren lokalt. I tillegg er det en del utvidet funksjonalitet [1]. Microsoft har anbefalt brukere å starte overgangen til Azure DSC siden den lokale Pull serveren ikke lengre vil videreutvikles med ny funksjonalitet av Microsoft [2].

Det overordnede målet med Azure DSC er å automatisere installasjon og konfigurasjon av servere og sikre at konfigurasjonen ikke endres over tid. Konfigurasjonene skal videre fungere som "en enkeltkilde til sannhet" og en levende dokumentasjon av infrastrukturen. Det betyr at i stedet for å klikke og konfigurere seg igjennom en rekke menyer, veivisere, register-innstillinger og konfigurasjonsfiler har man én konfigurasjonsfil. Konfigurasjonen er definert deklarativt i PowerShell-kode som gjør det mye enklere å lese og feilsøke ved eventuelle problemer siden antall mulige feilkilder blir redusert [3].

Videre blir alle endringer i programvare, konfigurasjon, register, roller og funksjoner gjort i konfigurasjonsfila. Dette resulterer i en enklere og tryggere endringshåndtering siden endringen utføres ett sted og det gir større mulighet for testing. Man kan for eksempel med få tastetrykk, kjøre ut konfigurasjonen til et testmiljø. Utføre og teste endringen, før man igjen tar ned testmiljøet for å utføre endringen på produksjonsmiljøet. Dette gjør konfigurasjonene til en levende dokumentasjon siden dokumentasjonen endres i takt med endringene i infrastrukturen [4].

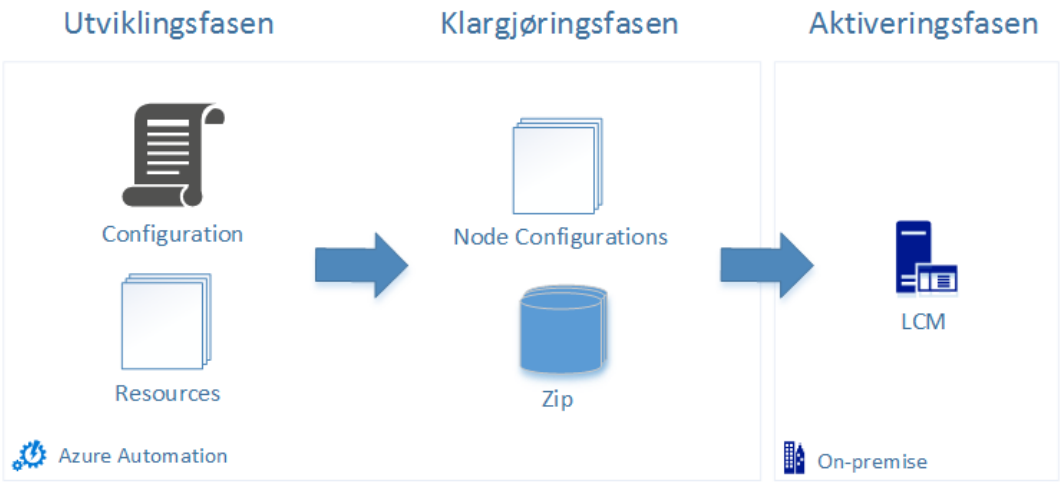
Til slutt er det verdt å nevne at Azure DSC gir større mulighet til å ha en skalerbar infrastruktur. Det vil si at man får muligheten til å kjøre opp og ned servere uten mye arbeid siden prosessen er automatisert. Dette egner seg godt i dynamiske miljøer der behovet for datakraft er variabelt i henhold til pågangen [4]. Dette kan for eksempel passe i henhold til problemstillingen der man i perioder har behov for økt datakraft og antall Session Hosts i et terminalservermiljø.

3.1 Komponenter og faser

Desired State Configuration er som sagt deklarativ, men det er også idempotent. Deklarativ betyr at man beskriver kun det som skal være gjeldene og at koden som utfører handlingene er abstraktert vekk fra brukerens betraktning. Dette resulterer i lettlest kode som er enkel å administrere. Man trenger for eksempel kun å definere at en bestemt Windows Feature skal være tilstede. Koden som utfører selve operasjonen ligger i den underliggende infrastrukturen og i ressursene man inkluderer sammen med konfigurasjonen.

Idempotent betyr at man kan kjøre den samme konfigurasjonen så mange ganger som mulig og fortsatt oppnå samme resultat. Det vil si at hvis man definerer at en Windows Feature skal være tilstedte vil den kun installere den hvis den ikke er tilstedte og den vil ikke installeres flere ganger oppå hverandre hvis den er tilstede [5].

Prosessen i Azure DSC består av tre faser som visst på Figur 1 – Faser i Azure Desired State Configuration. I tillegg til disse fasene vil klienten rapportere tilbake til Azure Automation med tilstandsstatus på konfigurasjonen. Det er ressursene man inkluderer i konfigurasjonen som tar seg av kontroll av tilstanden. Alle ressurser har en testmetode som benyttes til å sjekke tilstanden av det som blir konfigurert.



Figur 6 - Faser i Azure Desired State Configuration

Utviklingsfasen (Authoring) består i å utvikle konfigurasjoner og inkludere egenutviklede eller allerede utviklede ressurser i PowerShell for å konfigurere de rollene man ønsker å ha i sin infrastruktur. Det kan for eksempel være en rolle for en "Session Host". Når konfigurasjonen er klar vil denne kompileres til en MOF-fil. MOF-fila og ressursene gjøres tilgjengelig på pullserveren i Azure Automation. Ressursene komprimeres for økt ytelse [5].

I klargjøringsfasen (Stageing) vil klientene gjennomføre en "pull" mot den konfigurerte pullserveren og laste ned MOF-fila og eventuelle ressurser som den trenger for å fullføre konfigurasjonen [5].

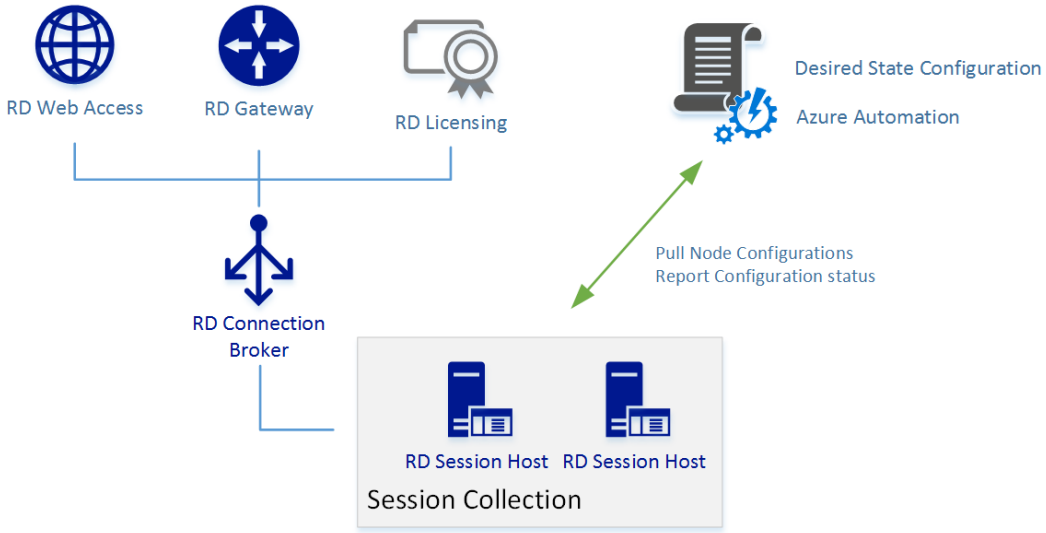
Siste fase er aktiveringsfasen (Make it so). Her vil Local Configuration Manager (LCM) lese MOF-fila og konfigurere klienten i henhold til hva som er definert i konfigurasjonsfila i fase én [5].

Til slutt er det verdt å nevne at Azure DSC kun vil kontrollere tilstanden på det man definerer i konfigurasjonen. Hvis man administrerer en maskin med DSC vil ikke DSC til å bry seg om programvare, filer og konfigurasjon som ikke er definert i konfigurasjonen.

4 Tekniske løsninger

Sluttproduktet består hovedsakelig av tre deler. Den første er konfigurasjon og tilpasning av Desired State Configuration i Azure Automation, den andre er utvikling av konfigurasjoner for installasjon og tilpasning av en "Session Hosts" og det tredje er det å sette det hele sammen med hensikt om at det skal bli så automatisert som mulig.

Den overordnede topologien for sluttproduktet er visst i Figur 2 – Topologi. Her vises terminalservermiljøet til venstre med all nødvendig infrastruktur og Desired State Configuration i Azure Automation til høyre. Pilen mellom RD Session Hosts og Azure Automation indikerer at RD Session Hosts henter konfigurasjoner fra pullserveren og rapporterer tilbake med sin konfigurasjonsstatus.



Figur 7 – Topologi

4.1 Azure Automation

For konfigurasjon og tilpasning av Azure DSC benyttes Azure Portalen og eller PowerShell Azure Cmdlets. Her kommer en beskrivelse av de forskjellige tjenestene, ressursene og funksjonaliteten man må benytte og konfigurere for å få løst problemstillingen.

4.1.1 Resource Group

For å samle og gi oversikt over relaterte ressurser i Azure brukes Resource Groups. Det gir oversikt over gruppens samlede kostnader, tilgang til monitorering og alarmer i tillegg til tilgangskontroll og andre administrative handlinger. Lokasjon defineres for å bestemme hvor metadataen om gruppens ressurser skal lagres [6].

Alle ressurser tilhørende prosjektet vil samles i samme ressursgruppe. Den er tildelt av NTNU IT, har navn "NTNU02-M-northeu-DSC-rg" og er plassert i "North Europe".

4.1.2 Automation Account

For å få tilgang til Azure Automations tjenester må det opprettes en Azure Automation konto. Azure Automation består av prosessautomasjon, oppdateringsbehandling og konfigurasjonsstyring. Prosjektet skal benytte seg av tjenesten konfigurasjonsstyring (Configuration Management) [7].

Tjenesten prises per administrerte node. Det koster 6 USD pr måned for on-premise-noder, men det er gratis for Azure-noder. Det er inkludert 5 stykk on-premise noder for bruk til testing og utvikling som prosjektet skal benytte. Kostnad for logging kommer i tillegg.

4.1.3 Azure Run As Account

For å sikre autentisering mot Azure når man bruker Azure PowerShell cmdlets kan man opprette en Azure Run As Account. Da får man muligheten til å autentisere sikkert med et sertifikat og et "Service Principal Names" i stedet for å benytte brukernavn og passord. Opprettelse av en Run As Account gjør følgende:

- Oppretter en Azure AD applikasjon, med et selvsignert sertifikat, en "Service Principal Account" for applikasjonen i Azure AD som tildeles "Contributor"-rollen.
- Oppretter sertifikatet "AzureRunAsCertificate" for Automation-kontoen.
- Oppretter en "Automation Connection Asset" med navn "AzureRunAsConnection".

Tilgangen til "Run As Service Principal" kan begrenses til "Contribrutor" for Ressursgruppen i etterkant av opprettelse [8].

4.1.4 Onboarding

De maskinene som er plassert on-premise vil kunne administreres fra skyen av Azure DSC. Denne prosessen kalles "onboarding", som vil si at nodene legges inn i Azure DSC. Maskinene må ha utgående tilgang til Azure som beskrevet i Kapittel 5.2 Brannmur og Windows Management Framework 5.1 installert. Onboarding kan gjøres via Azure Automation cmdlets eller via "WMF 5 DSC Registration Protocol". Sistnevnte gir mulighet for å utføre en sikker registrering der det opprettes unike sertifikat for hver node, som benyttes til autentisering av maskinen mot Azure [9].

Når maskinene onboardes må det konfigureres en rekke innstillinger vedrørende oppdateringsfrekvens, konfigurasjonsfrekvens og hvilken modus noden skal ha. Dette er innstillinger som lagres i Local Configuration manager og er nærmere forklart i Kapittel 4.1.5 Local Configuration Manager (LCM).

4.1.5 Local Configuration Manager (LCM)

LCM er selve motoren til DSC som kjører på alle nodene som skal administreres. Den er ansvarlig for å hente og motta konfigurasjonene fra Pull-serveren, dekode og utføre

konfigurasjonene på noden og kontrollere om noden er i den ønskede tilstanden. Er ikke noden i ønsket tilstand kan man for eksempel konfigurere LCM slik at den skal rekonfigurere noden tilbake til ønsket tilstand.

Når en node onboardes til Azure blir nodens LCM konfigurert og man må minimum definere følgende innstillinger i konfigurasjonen som visst i Tabell 1 – Innstillinger i LCM [10].

Innstilling	**Forklaring**
NodeConfigurationName	Hvilken DSC Node Configuration som skal konfigureres.
RefreshFrequency	Tiden for hvor ofte LCM skal sjekke Pull Server om det har kommet ny konfigurasjon.
ConfigurationMode	Definerer hvordan LCM bruker konfigurasjonen. ApplyOnly ApplyAndMonitor ApplyAndAutoCorrect
ConfigurationModeFrequency	Tiden for hvor ofte LCM kontrollerer om konfigurasjonen har blitt endret.
RebootNodeifNeeded	Definere om LCM kan utføre restart av noden hvis det er behov i f.eks. en installasjonsprosess.
ActionAfterReboot	Definere hvilken handling LCM skal utføre etter en restart som er utløst av DSC. Man kan velge mellom å stoppe eller fortsette installasjon.

Tabell 15 - Innstillinger i LCM

Innstillingen ConfigurationMode inneholder tre valg:

- **ApplyOnly** betyr at LCM bare konfigurerer noden
- **ApplyAndMonitor** betyr at i tillegg til å konfigurere noden skal også LCM melde tilbake hvis konfigurasjonen drifter fra den ønskede tilstanden.
- **ApplyAndAutoCorrect** betyr at i tillegg til å konfigurere noden skal også LCM rekonfigurere noden hvis konfigurasjonen drifter fra den ønskede tilstanden.

4.1.6 DSC Resource

DSC ressursene er de spesifikke funksjonene som kjøres når Azure DSC iverksetter konfigurasjon og kontroll mot nodene som skal administreres. En ressurs kan for eksempel være "WindowsFeature", som kan brukes til å sikre at en Windows Feature enten er tilstede eller fjernet fra en definert node. Som visst på Kodeblokk 1 – Eksempel på bruk av en DSC Resource, kalles ressursen med ressursnavnet "WindowsFeature". Denne konfigurasjonen sier at rollen "RDS-RD-Server" skal være tilstedte, med verdien "Present".

```
WindowsFeature ResourceExample
{
    Ensure = "Present"
    Name = "RDS-RD-Server"
}
```

Kodeblokk 1 - Eksempel på bruk av en DSC Resource

Det er inkludert en rekke ressurser med PowerShell, men Microsoft legger opp til at brukerne selv kan utvikle sine egne ressurser og dele de med andre [11]. Ressursene er Open Source og publiseres på for eksempel The PowerShell Gallery og kan importeres direkte inn i Azure Automation. Finner man ikke ressurser som dekker behovet kan man enten lage egendefinerte ressurser eller endre på eksisterende ressurser. Disse må bygges opp etter en viss standard og alle ressursene må ha følgende metoder tilgjengelig: "Get", "Test" og "Set".

- Get-metoden kontrollerer hvilken status egenskapene til ressursen har hos noden.
- Test-metoden kontrollerer om noden er i den konfigurerte ønskede tilstand.
- Set-metoden vil forsøke å konfigurere noden slik at den kommer i ressursens ønskede tilstand.

Forskjellen på Get og Test er at Get returnerer selve statusen for ressursen, mens Test returnerer en boolsk verdi, altså $true eller $false [12].

For å utvikle egne ressurser bør man benytte Resource Designer Tool. Dette er et verktøy setter opp rammene for script-fila, MOF-skjemaet, modulen, og mappestrukturen. Dette gjør at man heller kan fokusere på utviklingen av PowerShell kode i stedet for å tenke på detaljene rundt ressursen [13].

Kompositttressurser

Det er også mulig å utvikle kompositttressurser (DSC Composite Resource). En kompositttressurs er en DSC-ressurs bestående av en annen konfigurasjon. Ved å gjøre det på denne måten vil man håndtere konfigurasjonen ett sted og ikke en gang for hver forskjellig konfigurasjon man har. Dette sikrer for eksempel en endringsprosess mot vanvare og forglemmelse.

Andre bruksområder enn baseline konfigurasjon kan være hvis man har veldig avanserte konfigurasjonsblokker. Det kan da lages standardkonfigurasjon for de forskjellige tjenestene man har behov for som så refereres inn i konfigurasjonene til de nodene som har behov for en viss type funksjonalitet. Da vil man i stedet for å bygge opp en ny konfigurasjon hver gang kun benytte allerede eksisterende konfigurasjon. Dette kan for eksempel være brannmurregler eller et bestemt oppsett av Web-servere.

4.1.7 DSC Configuration

DSC konfigurasjonen er et PowerShell-script hvor det defineres en node, hvilke ressurser som skal konfigureres og hvilken tilstand ressursene skal ha. Som visst på Kodeblokk 2 – Eksempel på DSC Configuration ser man at konfigurasjonen består av tre blokker. Configurations-blokken fungerer på samme måte som en PowerShell-funksjon, Node-blokken definerer de maskinene som skal konfigureres, og til slutt er det en eller flere ressursblokker hvor man definerer de egenskapene noden skal ha [14].

```
Configuration ConfigurationExample
{
    Node IsSessionHost
    {
        WindowsFeature RDS
        {
            Ensure = "Present"
            Name = "RDS-RD-Server"
        }
    }
}
```

Kodeblokk 2 - Eksempel på DSC Configuration

Før nodene kan konfigureres av DSC konfigurasjonen må den kompilereres. Dette blir nærmere forklart i Kapittel 4.1.8 DSC Node Configurations.

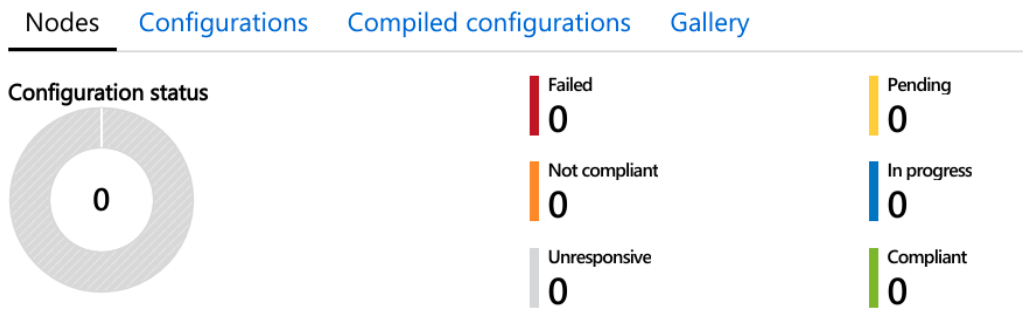
4.1.8 DSC Node Configurations

Kompilering av DSC konfigurasjoner kan enten gjøres i Azure portalen eller med PowerShell-kommando. Kompileringen resulterer i en MOF-fil, som vil få følgende navn: "Configuration.Node" altså "ConfigurationExample.IsSessionHost" som visst i eksempelet i Kodeblokk 2 - Eksempel på DSC Configuration. MOF-filen er bygget opp etter en standard, slik at DSC kan lese den og konfigurere noden. Når konfigurasjonen kompileres blir MOF-filene lagret og gjort tilgjengelig for nodene på pullserveren. Den kompilerte konfigurasjonen er i praksis en rolle som kan tildeles de nodene man ønsker å konfigurere der rollenavnet er det samme som node-navnet.

I PowerShell DSC er det viktig at MOF-filen blir kryptert og sikret siden den inneholder detaljert informasjon om konfigurasjonen. I Azure DSC behøver man ikke ta stilling til dette siden det blir tatt hånd om av tjenesten. Nodene får et automatisk sertifikat som benyttes i krypteringen av MOF-filene [15].

4.1.9 Monitorering og logging

Azure Automation gir mulighet for monitorering og logging. Som visst på Figur 3 – Configuration Status vises et bilde fra selve portalen. Her får man oversikt over nodene og i hvilken status de står i, for øyeblikket.



Figur 8 - Configuration Status (Bilde fra Azure)

Det er mulig å trykke nodene slik at man får mer informasjon og detaljer. Forklaring for de ulike statusbeskjedene kan ses i Tabell 2 – Statusrapport for administrert node [16]. Samme informasjon kan hentes ut ved bruk av Azure Automation Cmdlets (AzureRm).

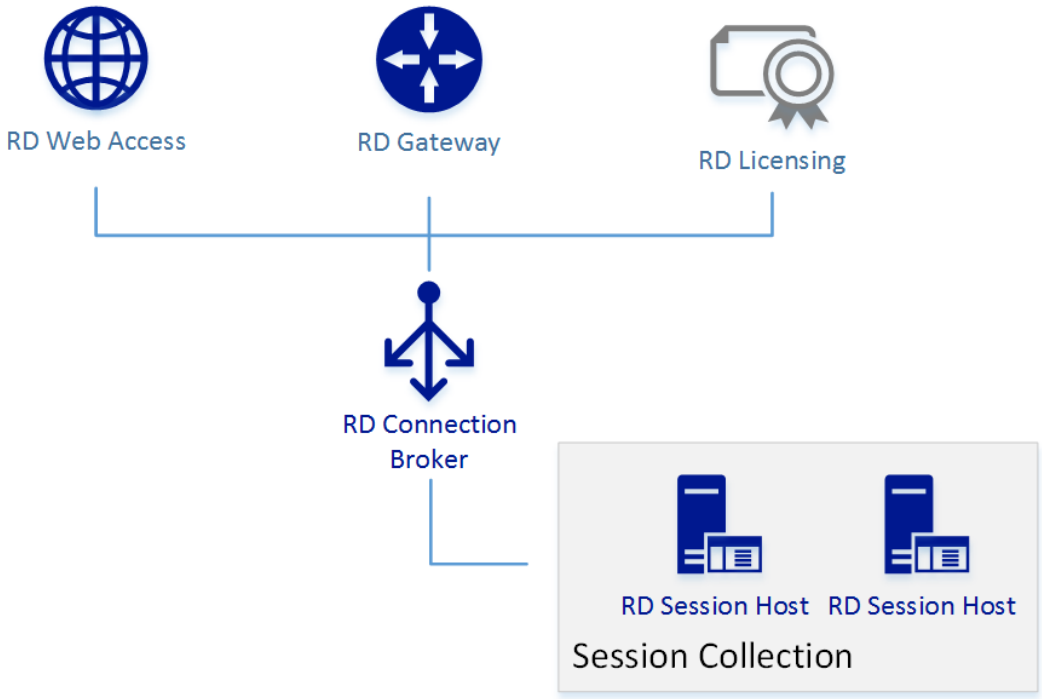
Status	**Forklaring**
Report status	Viser hvilken status noden står i.
Report time	Tidspunkt rapporten blir hentet
Start time	Tidspunkt for statussjekk
Total runtime	Hvor lang tid statussjekken benyttet
Type	Hvilken type sjekk (Consistency)
Error	Eventuelle errorkoder og errorbeskjeder
Resources	Eventuelle DSC ressurser som er benyttet i konfigurasjonen
Node navn	Maskinenes (nodens) enhetsnavn
IP adresse	Maskinens (nodens IP-adresse

Tabell 16 - Statusrapport for administrert node

Azure DSC kan sende statistikk og logger til Azure Log Analytics. Dette gir muligheten til å hente ut en mer avansert oversikt over node-status, kompatibilitet, konfigurere e-postvarsling og visualisere status for noder over tid. Dette gir mulighet til å lettere feilsøke og se sammenhenger mellom forskjellige systemer [17].

4.2 Terminalservermiljø

NTNU ITs terminalservermiljø er bygget på Microsoft Remote Desktop Services. Det fullstendige terminalservertmiljøet er visst på Figur 4 - Terminalservermiljø. De forskjellige serverrollene blir forklart forløpende.



Figur 9 – Terminalservermiljø

4.2.1 Remote Desktop Session Hosts

RD Session Hosts driver selve skrivebordet og applikasjonene brukere kobler seg til via terminalservertjenesten. Det er altså disse maskinene man kobler seg til via for eksempel Eksternt Skrivebord (Remote Desktop) i Windows. De fungerer i praksis som en klientmaskin der brukere logger seg på.

RD Session Hosts organiseres i "Session Collections" (session-farm) der man samler en eller flere servere som utfører samme oppgave. Dette gjøres for å øke tilgjengeligheten og ytelsen ved at man benytter Remote Desktop Connection Broker til å lastbalansere brukerne over flere RD Session Hosts [18].

4.2.2 Remote Desktop Connection Broker

RD Connection Broker har ansvar for å håndtere innkommende Remote Desktop tilkoblinger til et RD Session Host server-farm. Den vil balansere tilkoblingene utover de forskjellige RD Session Hosts som finnes i en Session Collection. Ved å lastbalansere brukere utover flere maskiner unngår man at alle brukere havner på én RD Session Hosts slik at ytelsen blir bedre

og mer fordelt utover. Samtidig vil tilgjengeligheten øke siden en eller flere RD Session Hosts kan gå ned, men så lenge det er minst én som er oppe vil terminalservertjenesten være operativ [18].

4.2.3 Remote Desktop Gateway

RD Gateway gir eksterne (off-campus) brukere muligheten til å benytte terminalservertjenesten over internett. Kommunikasjonen krypteres mellom klienten og serveren for å sikre konfidensialitet og integritet [18].

4.2.4 Remote Desktop Web Access

RD Web Access gir brukere mulighet til å koble til eksterne skrivebord og programvare via en webportal. Webportalen tilgjengeliggjør skrivebordet og applikasjonene for brukerne, også benyttes operativsystemets programvare for eksternt skrivebord til å koble mot skrivebordet eller applikasjonen. I praksis lastes det ned en RDP-fil som kjøres i klientprogramvaren [18].

4.2.5 Remote Desktop Licensing

RD Licensing er lisensieringstjenesten til Remote Desktop Services. Det er mulig å lisensiere for hver bruker eller for hver maskin i terminalservertjenesten [18].

4.3 Automatisering

Et av suksesskriteriene til oppdragsgiver som ble avdekket i interessentanalysen i forstudien, var et ønsket ideal om så mye automatisering som mulig. Dette suksesskriteriet er noe som også stemmer godt overens med prosjektgruppens prosessmål. Det er gode muligheter for stor grad av automatisering ved kun bruk av Azure DSC, men dette kan tas videre ved å ta i bruk et versjonshåndteringssystem og Azure Automation Runbooks.

Det er mulig å lage en Runbook som kontinuerlig kontrollerer om en konfigurasjon lagret på et versjonshåndteringssystem er endret. Hvis noen gjør en endring kan Runbooken kjøre de Azure DSC Cmdlets som skal til for at konfigurasjonen skal kompileres slik at nodene mottar den nye konfigurasjonen automatisk. Med denne løsningen vil man også dra nytte av versjonshåndtering på konfigurasjonene og man kan også utføre automatiserte tester. Dette gjør det enklere å tilbakekalle endringer ved eventuelle feil og det blir mer oversiktlig over hvem som har skrevet hvilken kode.

5 Detaljerte løsningsbeskrivelser

Dette kapittelet tar for seg de detaljerte løsningsbeskrivelsene for hva som skal gjøres i prosjektet.

5.1 Onboarding i Azure DSC

For å legge til maskiner i Azure DSC må man ha forutsetningene som er tidligere nevnt Kapittel 4.1.4 Onboarding. Videre må man lage en DSC metakonfigurasjon som konfigurerer serverens LCM til å benytte Azure Automation som pull-server. Microsoft har utviklet et script[4] som tar seg av selve konfigurasjonen. Scriptet har en rekke parametere som må tilpasses i henhold til ønsket konfigurasjon. Parameterne vil bli fylt ut med verdier som visst i Tabell 3 – Parametere i script som konfigurerer LCM.

Verdi	**Konfigurasjon**
RegistrationUrl	Hentes fra Azure Automation Account
RegistrationKey	Hentes fra Azure Automation Account
ComputerName	it-azdscfarm01, it-azdscfarm02
NodeConfigurationName	RemoteDesktopServices.SessionHost
RefreshFrequencyMins	30
ConfigurationModeFrequencyMins	15
RebootNodeIfNeeded	$True
AllowModuleOverwrite	$False
ConfigurationMode	ApplyAndAutoCorrect
ActionAfterReboot	ContinueConfiguration
ReportOnly	$False

Tabell 17 - Parametere i script som konfigurerer LCM

5.2 Brannmur

Maskiner som skal administreres av Azure DSC behøver utgående internettaksess på TCP port 433. Dette er ikke noe problem da "Session Hosts" i prosjektet har full utgående internettaksess med offentlig IP-adresse.

Hvis man er på et lukket nettverk eller på et nettverk med private IP-adresser må følgende verdier konfigureres:

- Port: TCP 443 utgående
- Global URL: *.azure-automation.net

[4] https://docs.microsoft.com/nb-no/azure/automation/automation-dsc-onboarding#generating-dsc-metaconfigurations

- Agent Service: http://ne-agentservice-prod-1.azure-automation.net (North Europe)

Det slippes i tillegg ukentlig en XML-fil med oversikt over alle IP-adressene i Azure sitt datasenter som kan benyttes til å oppdatere lokal brannmur.

5.3 Lokalt testmiljø

Som nevnt i forprosjektet skal prosjektetgruppen benytte et testcollection og to servere som skal fungere som "Session Hosts". Serverne er virtuelle og det kan tas Snapshot av maskinene i forkant av prosjektarbeidet slik at man enkelt kan føre maskinene tilbake til tilstanden før konfigurasjon. Dette gir mulighet til å enkelt teste forskjellig funksjonalitet uten at man skal bruke lang tid på installasjon av operativsystem. Det er Azure DSC og selve konfigurasjonene som er produktet i prosjektet, så at de lokale serverne tilbakestilles medfører ingen problemer for gjennomføringen.

Navnsetting på testmiljø skal følge NTNU IT sin navnestandard på terminalservermiljøet og vil få navn som visst i Tabell 4 – Navnsetting for testmiljø.

Hva	**Ønsket navn**
Session Collection	azdscfarm.it.ntnu.no
Server 1	it-azdscfarm01
Server 2	it-azdscfarm02

Tabell 18 - Navnsetting for testmiljø

5.4 Konfigurere en Session Host

Resten av kapittelet vil gå med på beskrivelse av hvordan NTNU IT løser konfigurasjonen i dag og hvilke ressurser som kan benyttes til å la Azure DSC håndtere konfigurasjonen. Gjennomgangen vil ikke si noe om akkurat hvilke ressurser som er best egnet for å løse problemstillingen. Det kan godt hende at de må tilpasses eller at det må utvikles egne ressurser. Dette er kun en grov planlegging av arbeidet for implementeringen.

5.4.1 Konfigurasjon av disk og plassering av pagingfile

Dette punktet omhandler konfigurasjon av pagingfil for harddisk. NTNU IT har laget et script som utfører operasjonen. Følgende script utfører operasjonen.

```
Import-Module "\\win.ntnu.no\shares\powershell\Prod-
signed\AdjustVirtualMemoryPagingFileSize\AdjustVirtualMemoryPagingFile
Size.psm1"
Set-OSCVirtualMemory  -InitialSize 100000  -MaximumSize 200000  -
DriveLetter "C:"
```

Kodeblokk 3 - Konfigurasjon av disk og plassering pagingfile

Mulige DSC Ressurser: ComputerManagementDsc (VirtualMemory)

5.4.2 Legge til IPv6-adresse

For at Direct Access skal fungere med terminalservermiljøet trenger hver "Session Hosts" en IPv6 site-local adresse. Denne adressen er bygget opp av IPv4 adressen. NTNU IT benytter følgende script for å konfigurere adressen.

```
$NAT64daccess02 = "fdd2:2868:bf02:7777::81f1:e"
$serverip = (Resolve-DnsName -Name $env:Computername | where {$_.Type
-like "A"} | select IPAddress).IPAddress
$interfaceindex = (Get-NetIPAddress | where {$_.IPAddress -like
$serverip} | select InterfaceIndex).InterFaceindex
$serveriplast = [int]$serverip.Substring(11)
$serveriplast ="{0:X}" -f $serveriplast
$ServerNAT64da02 = $NAT64daccess02+$serveriplast
New-NetIPAddress -AddressFamily IPv6 -InterFaceIndex $interfaceindex -
IPAddress $ServerNAT64da02 -PrefixLength 64
```

Kodeblokk 4 - Konfigurasjon av IPv6-adresse

Mulige DSC Ressurser: NetworkingDsc (IPAddress), utvikle egen

5.4.3 Installasjon av rollen Remote Desktop Session Host

For installasjon av Remote Desktop Session Host rollen kjøres følgende kommando.

```
Install-WindowsFeature RDS-RD-Server
```

Kodeblokk 5 - Installasjon av Remote Desktop Session Host rolle

Mulige DSC Ressurser: WindowsFeature

5.4.4 Installasjon av XPS Viewer

For installasjon av XPS Viewer kjøres følgende kommando.

```
Install-WindowsFeature -Name XPS-Viewer
```

Kodeblokk 6 - Installasjon av XPS-Viewer

Mulige DSC Ressurser: WindowsFeature

5.4.5 Installasjon av Windows Identity Foundation 3.5

For installasjon av Windows Identity Foundation kjøres følgende kommando.

```
Install-WindowsFeature -Name Windows-Identity-Foundation
```

Kodeblokk 7 - Installasjon av Windows-Identity-Foundation

Mulige DSC Ressurser: WindowsFeature

5.4.6 Installasjon av Active Directory modul PowerShell

For installasjon av Active Directory modul PowerShell kjøres følgende kommando.

```
Install-WindowsFeature RSAT-AD-Powershell
```

Kodeblokk 8 - Installasjon av RSAT-AD-PowerShell

Mulige DSC Ressurser: WindowsFeature

5.4.7 Installere og deaktivere Windows Search

For installasjon og deaktivering av Windows Search følgende kommandoer. Dette gjøres for å fjerne en feilmelding som kommer opp i Outlook.

```
Install-WindowsFeature -Name Search-Service
Stop-Service wsearch
Set-Service wsearch -StartupType disabled
```

Kodeblokk 9 - Installasjon og deaktivering av Windows Search

Mulige DSC Ressurser: WindowsFeature, Service

5.4.8 Avinstallere Windows Defender

For avinstallasjon av Windows Defender kjøres følgende kommando.

```
Uninstall-WindowsFeature -Name Windows-Defender
```

Kodeblokk 10 - Avinstallasjon av Windows-Defender

Mulige DSC Ressurser: WindowsFeature

5.4.9 Installasjon av .NET Framework 3.5

For installasjon av .NET Framework 3.5 kjøres følgende kommando.

```
DISM /Online /Enable-Feature /FeatureName:NetFx3 /All
```

Kodeblokk 11 - Installasjon av .Net Framework 3.5

Mulige DSC Ressurser: WindowsFeature

5.4.10 Installere Xerox Generic PS printerdriver

For installasjon av Xerox Generic PS printerdriver gjøres følgende tilpasninger.

> For at utskrift skal fungere over alt kreves det noen tilpasninger. Programpakken "OI-ITAVD_SRV Legg til FS-skriver" oppretter en skriverkø med navnet Sybase DataWindow PS. Driveren til denne må endres til Xerox Global Print Driver PS etter at programpakken er installert.
>
> Åpne Enheter og Skrivere (Devices and Printers, **control /name Microsoft.DevicesAndPrinters**) på Sessionhost-server, og åpne deretter Sybase DataWindow PS. Velg Åpne som administrator. Velg Skriver - Egenskaper - Fanen Avansert - Ny driver... - Har disk...
> Velg \\win.ntnu.no\shares\installshare\pakker\dagens\Legg til skriver\Xerox Generic PS\X-GPD_5.273.23.2_PS_x64_Driver.inf.
>
> Etter installasjonen skal det stå Driver: Xerox Global Print Driver PS

Figur 10 - Installere Xerox Generic PS printerdriver

Mulige DSC Ressurser: PrinterManagement, utvikle egen

5.4.11 Installere sertifikat

Rutine for installasjon av sertifikat for langt til å inkluderes i Designrapporten.

Mulige DSC Ressurser: CertificateDsc

5.4.12 Fikse potensielt sikkerhetshull på Windows Server 2016

Det er et sikkerhetshull i Windows Server 2016 og følgende script forhindrer at alle brukere kan opprette filer under c:\users\default.

```
<#
.SYNOPSIS
Temporary fix for vulnerable permissions on default user profile for
Windows 10/Server 2017 build 1607
.DESCRIPTION
This script sets ACL on the default user profile the same as other
folders in C:\Users as a temporary fix to a vulnerability Introduced
by Microsoft in Windows 10, and Windows server 2016 build 1607
.NOTES
File Name : setacl.ps1
Author : Lasse B. Gustavsen <lasse.b.gustavsen@ntnu.no>
#>
$newACL = get-acl -path c:\users\
get-childitem -path
'C:\users\Default\Appdata\Roaming\Microsoft\Windows\Start Menu\' -
Recurse -force | set-acl -Aclobject $newACL
set-acl -path c:\users\default\Desktop\ -ACLobject $newACL
set-acl -path c:\users\default\Documents\ -ACLobject $newACL
set-acl -path c:\users\default\Downloads\ -ACLobject $newACL
set-acl -path c:\users\default\Favorites\ -ACLobject $newACL
set-acl -path c:\users\default\Links\ -ACLobject $newACL
set-acl -path c:\users\default\Music\ -ACLobject $newACL
set-acl -path c:\users\default\Pictures\ -ACLobject $newACL
set-acl -path 'c:\users\default\Saved Games\' -ACLobject $newACL
set-acl -path c:\users\default\Videos\ -ACLobject $newACL
if (!(Test-Path "C:\Windows\aclfix.txt"))
{
New-Item -path C:\Windows -name aclfix.txt -type "file" -value "yes"
}
else
{
Add-Content -path C:\Windows\aclfix.txt -value "yes"
}
```

Kodeblokk 12 - Fikse potensielt sikkerhetshull

Mulige DSC Ressurser: cNtfsAccessControl (Community-based), utvikle egen

5.4.13 Legge session host i riktig collection

Følgende script kjøres for å legge en "Session Hosts" i en collection.

```
$SessionHost = "it-officefarm03.win.ntnu.no"
$Collection = "Officefarm"
$ConnectionBroker = "it-rdconnect01.win.ntnu.no"
# Legg til server i "connections"
Add-RDServer -ConnectionBroker $ConnectionBroker -Server $SessionHost
-Role "RDS-RD-SERVER"
# Legg til session host i "collection"
Add-RDSessionHost -CollectionName $Collection -SessionHost
$SessionHost -ConnectionBroker $ConnectionBroker
# Stopp tilkoblinger til session host
Set-RDSessionHost -ConnectionBroker $ConnectionBroker -
NewConnectionAllowed No -SessionHost $SessionHost
```

Kodeblokk 13 - Legge Session Host i rett collecton

Mulige DSC Ressurser: xRemoteDesktopSessionHost, utvikle egen

5.4.14 Åpne brannmur på løsninger

Brannmurregler administreres via Group Policy.

Mulige DSC Ressurser: NetworkingDsc

Referanser

[1] "Azure Automation State Configuration Overview" Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-overview, Jun. 11, 2018 [Feb. 3, 2019].

[2] "Desired State Configuration Pull Service" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/pull-server/pullServer, Apr. 11, 2018 [Feb. 3, 2019].

[3] "Desired State Configuration Overview for Decision Makers" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/overview/decisionmaker, Jun. 12, 2017 [Feb. 20, 2019].

[4] "Desired State Configuration Overview for Engineers" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/overview/dscforengineers, okt. 13, 2017 [Feb. 23, 2019].

[5] M. Gray & J. Levy. Microsoft Ignite 2015, Topic: "Heterogeneous Configuration Management Using Microsoft Azure Automation" McCormick Place, Chicago, IL., May 2015. Internett: https://azure.microsoft.com/en-in/resources/videos/microsoft-ignite-2015-heterogeneous-configuration-management-using-microsoft-azure-automation/ [Feb. 23, 2019]

[6] "Azure Resource Manager overview" Internett: https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview, Jan. 02, 2019 [Feb. 9, 2019].

[7] "An introduction to Azure Automation" Internett: https://docs.microsoft.com/en-us/azure/automation/automation-intro, Okt. 10, 2019 [Feb. 1, 2019].

[8] "Manage Azure Automation Run As accounts" Internett: https://docs.microsoft.com/nb-no/azure/automation/manage-runas-account, Sep. 12, 2018 [Feb. 11, 2019].

[9] "Onboarding machines for management by Azure Automation State Configuration" Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding, Aug. 8, 2018 [Feb. 11, 2019].

[10] "Configuring the Local Configuration Manager" Internett: https://docs.microsoft.com/nb-no/powershell/dsc/managing-nodes/metaConfig, Des. 12, 2018 [Feb. 16, 2019].

[11] "DSC Resources" Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/resources, Des. 12, 2018 [Feb. 13, 2019].

[12] "Get-Set-Test Internett": https://docs.microsoft.com/en-us/powershell/dsc/resources/get-test-set, Des. 12, 2018 [Feb. 13, 2019].

[13] "Using the Resource Designer tool" Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/authoringresourcemofdesigner, Des. 06, 2017 [Feb. 16, 2019].

[14] “DSC Configuration” Internett: https://docs.microsoft.com/en-us/powershell/dsc/configurations/configurations, Des. 12, 2018 [Feb. 14, 2019].

[15] “Securing the MOF File” Internett: https://docs.microsoft.com/en-us/powershell/dsc/pull-server/securemof, Okt. 31, 2017 [Feb. 24, 2019].

[16] “Getting started with Azure Automation State Configuration” Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-getting-started, Aug. 08, 2018 [Feb. 17, 2019].

[17] “Forward Azure Automation State Configuration reporting data to Log Analytics” Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-diagnostics, Jun. 11, 2018 [Feb. 17, 2019].

[18] “Remote Desktop Services roles” Internett: https://docs.microsoft.com/en-us/windows-server/remote/remote-desktop-services/rds-roles, Jun. 07, 2018 [Feb. 23, 2019].

Vedlegg C – Driftsrapport

Driftsrapport
Azure Automation State Configuration for administrasjon av Windows-servere

Bacheloroppgave i informatikk, drift av datasystemer

Institutt for datateknologi og informatikk

Norges teknisk-naturvitenskapelige universitet (NTNU)

Innlevering

20. Mai 2019

Øyvind Lotten Westrum

Revisjonshistorikk

Dato	Versjon	Beskrivelse	Forfatter
20. mai 2019	1.0	Innlevering	OLW

Innholdsfortegnelse

1 Introduksjon

NTNU IT ønsker å se på muligheten til å bruke Azure Automation State Configuration (Azure DSC) til å automatisere installasjon, oppsett og konfigurasjon av «Session Hosts» i et terminalservermiljø. Det er utviklet et sluttprodukt (Proof of Concept) i form av en ferdig konfigurert løsning. Azure DSC er Microsoft sin skytjeneste for konfigurasjonsstyring og gir muligheten til å automatisk konfigurere servere og sikre at konfigurasjonen ikke endres over tid.

1.1 Dokumentets hensikt

Dokumentet er driftsrapporten i faget IDRI3001 Bacheloroppgave i drift av datasystemer. Rapportens hensikt er å fungere som komplett dokumentasjon av implementering, konfigurering og driftsrutiner for de tekniske detaljene rundt sluttproduktet. Målet ved driftsrapporten er å bidra til kunnskapsoverføring og fungere som oppslagsverk for videre drift og utvikling av løsningen. Oppdragsgiver får her detaljkunnskap om de valg som er tatt og hvordan forskjellige problemstillinger ble løst.

1.2 Hvordan lese rapporten

Rapporten kan leses fra start til slutt, den er bygget opp slik implementeringen ble gjennomført i prosjektet. Men det er mer hensiktsmessig å bruke rapporten som et oppslagsverk der man slår opp på det temaet som man ønsker å vite mer om.

For å treffe bredest mulig inneholder driftsrapporten både instruksjon for hvordan man utfører forskjellige oppgaver i Azure portalen og med PowerShell-kommandoer. Rapporten har en hel del kodeeksempler som viser PowerShell-Script- og kommandoer. PowerShell-kommandoer blir ofte lange og går over flere linjer i Word hvis det er spesifisert mange parametere. For å øke leseopplevelsen er det benyttet PowerShell-Splatting[5] av parameterdata i kodeeksemplene slik at enkeltkommandoer ikke går over flere linjer.

Videre følger en kort introduksjon for hvert kapittel.

Kapittel 1 Introduksjon

Gir en introduksjon til rapporten.

Kapittel 2 Oppsett og konfigurasjon i Azure

Tar for seg all funksjonalitet, oppsett og konfigurasjon i Azure Automation.

Kapittel 3 Oppsett og konfigurasjon i testmiljø

Tar for seg funksjonalitet, oppsett og konfigurasjon lokalt på maskinene. Den gir en kort oversikt over nyttige PowerShell-kommandoer og hvordan man kan feilsøke problemer med Azure DSC på en maskin.

[5] https://docs.microsoft.com/en-us/powershell/module/microsoft.powershell.core/about/about_splatting?view=powershell-6

Kapittel 4 Egen- og videreutviklede DSC-ressurser

Fungerer som dokumentasjon for egen- og videreutviklede DSC-ressurser som har blitt laget og tilpasset under prosjektperioden. Viser syntaks, egenskaper og et eksempel samtidig som det gir en kort forklaring for valg som er tatt i utviklingen.

Kapittel 5 Konfigurasjon

Gir en inngående innføring i konfigurasjonen som automatisk konfigurerer en Session Host. Kapittelet tar for seg alle konfigurasjonsblokkene og bruk av DSC-ressurser.

Kapittel 6 Az PowerShell modul

Gir en innføring i bruk av de mest nyttige PowerShell-Az-kommandoene.

Kapittel 7 Driftsrutiner for forskjellige oppgaver

Tar for seg driftsrutiner for forskjellige oppgaver.

Kapittel 8 Kjente problemer og begrensninger

Gir en kort innføring og forklarer hvor man finner informasjon om kjente feil og begrensninger.

1.3 Terminologi

Session Host (RDSH) En rolle i Remote Desktop Services (RDS) som gir brukere mulighet til å koble seg til eksternt skrivebord og kjøre Windows-programvare eksternt.

Remote Desktop Services (RDS) Samlebetegnelse for en Windows Server tjeneste som gir mulighet for fjernkjøring av programmer.

Session Collection Samling av Session Hosts som gir mulighet for lastbalansering.

Terminalservermiljø Generelt begrep for et system tilsvarende Remote Desktop Services.

Azure Microsoft sin nettskyplattform.

Azure AD Microsofts skytjeneste for identitet- og tilgangsstyring.

Azure Automation Automasjonstjeneste i nettskyen for prosessautomatisering og konfigurasjonsstyring i Azure.

Azure Automation State Configuration Tjeneste som gir mulighet til å skrive, administrere og kompilere PowerShell Desired State Configuration (DSC) konfigurasjoner, importere DSC-ressurser og tilordne konfigurasjoner til noder.

Desired State Configuration (DSC) Styringsplattform i PowerShell som gjør at man kan installere, styre og konfigurere IT-systemene med kode.

Pull Server Tjeneste som tilgjengeliggjør konfigurasjoner, lar noder kontrollere den ønskede tilstand og rapportere tilbake om de er i ønsket tilstand.

DSC Configuration / Konfigurasjonsdokument PowerShell script som definerer konfigurasjon, node, ressurs og hvilke attributter ressursen skal konfigureres med.

DSC Node Configuration / Kompilert konfigurasjon En kompilert DSC konfigurasjon. Fungerer som en rolle som kan tildeles en server.

DSC Node En maskin/enhet hvor konfigurasjonen er administrert av DSC

DSC Resource Det man bruker for å definere DSC konfigurasjonen (ressursene)

Compilation Job Å kompilere en konfigurasjon til MOF-fil og gjøre den tilgjengelig på pull-serveren.

Local Configuration Manager (LCM) Maskineriet i DSC. Funksjonaliteten som henter, leser og eksekverer konfigurerer på noden.

Managed Object Format (MOF) Format som brukes til å beskrive en konfigurasjon. Brukes av DSC som konfigurerer en node.

Konfigurasjonsstyring Styring av innhold, endringer og status på en konfigurasjon.

On-Premise (Computing) Servere som er eid, kontrollert og plassert lokalt.

Nettsky / Sky (Computing) Infrastruktur, plattform eller programvare som leies, administreres og er plassert hos en tredjepart.

PowerShell Et verktøy og scriptspråk som gir mulighet til å automatisere og styre et operativsystem fra kommandolinjen.

PowerShell Script Samling av PowerShell-kommandoer som automatiserer en oppgave.

PowerShell Cmdlets Et PowerShell-script som utfører en funksjon.

PowerShell Splatting Sende en samling av parameterverdier til en PowerShell-kommando.

2 Oppsett og konfigurasjon i Azure

Dette kapittelet tar for seg hvordan oppsett og konfigurasjon gjøres i Azure for at løsningen skal fungere. Dette består av oppsett av Azure Resource Group, Automation Account, onboarding av noder, inkludere legitimasjon og DSC-ressurser. I tillegg gås det igjennom administrasjon av konfigurasjoner, kompilerte konfigurasjoner og feilsøking, logging og overvåking av noder.

Kapittelet sier for eksempel ikke noe om eksakt hvilke DSC-ressurser og hvilken legitimasjon som må inkluderes. Dette blir gått igjennom i kapittel 5 – konfigurasjon.

2.1 Opprette Azure Resource Group

For å samle og gi oversikt over relaterte ressurser i Azure brukes Resource Groups. Det gir oversikt over gruppens samlede kostnader, tilgang til monitorering og alarmer i tillegg til tilgangskontroll og andre administrative handlinger. Lokasjon defineres for å bestemme hvor metadataen om gruppens ressurser skal lagres [1].

Alle ressurser tilhørende prosjektet vil samles i samme ressursgruppe. Den er tildelt av NTNU IT, har navn “NTNU02-M-northeu-DSC-rg" og er plassert i “North Europe”.

2.1.1 Opprette Azure Resource Group med PowerShell

Azure Resource Group opprettes med PowerShell kommando som visst på kodeblokk 1.

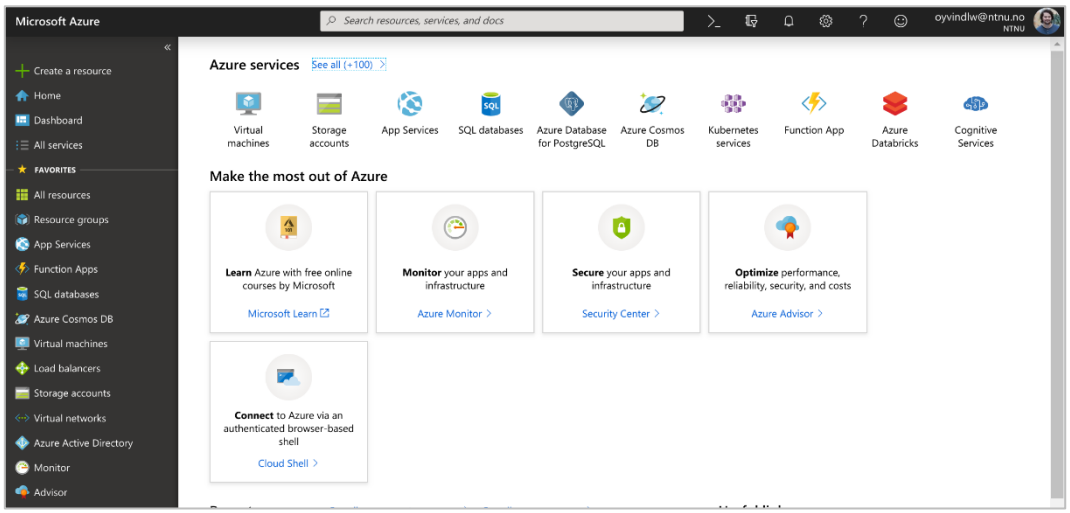
```
Connect-AzAccount
$Params = @{
    Name = "NTNU02-M-northeu-DSC-rg";
    Location = "North Europe";
}
New-AzResourceGroup @Params
```

Kodeblokk 4 - Opprette Azure Resource Group

2.1.2 Opprette Azure Resource Group i portalen

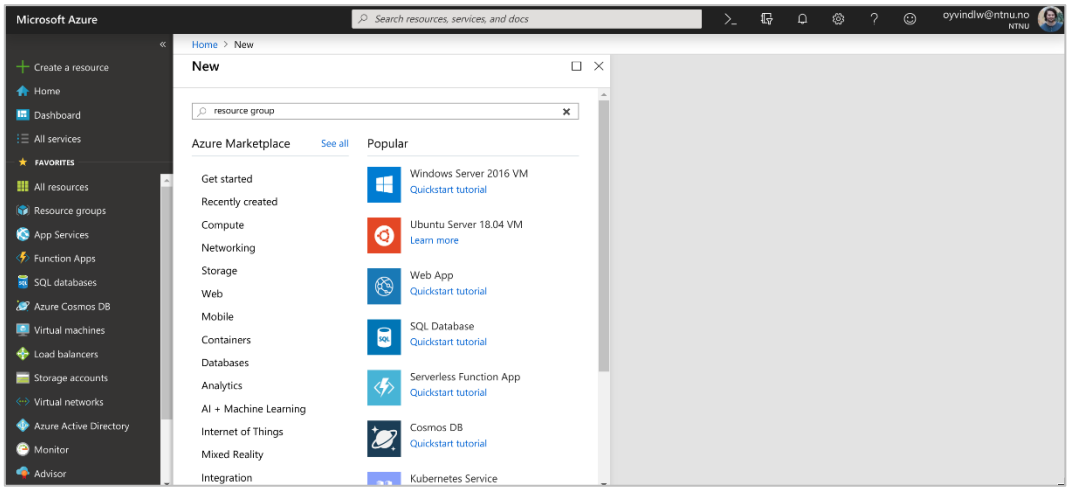
For å opprette en Azure Automation Account i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg “Create a resource”.



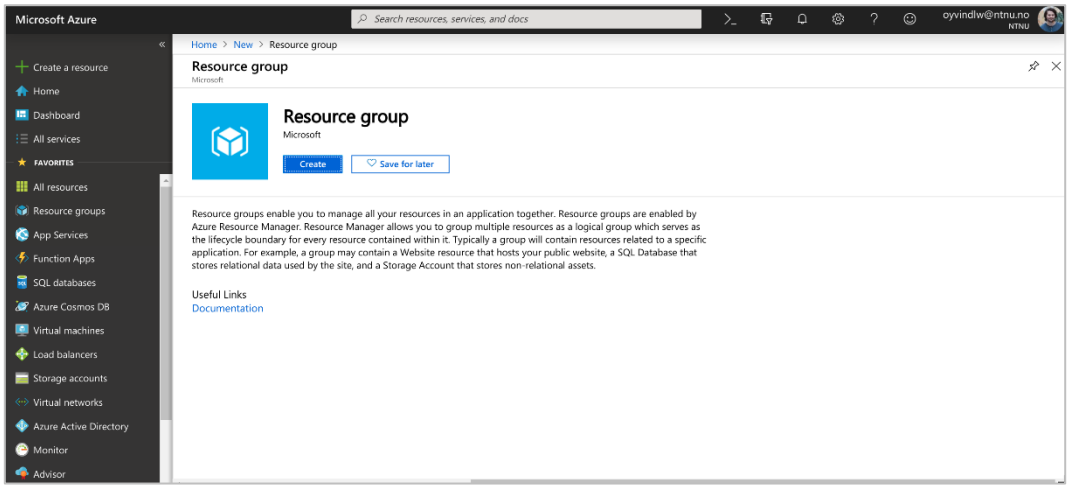
Figur 11 - Opprette Azure Resource Group

3. Skriv inn “resource group” i søkefeltet og trykk Enter.



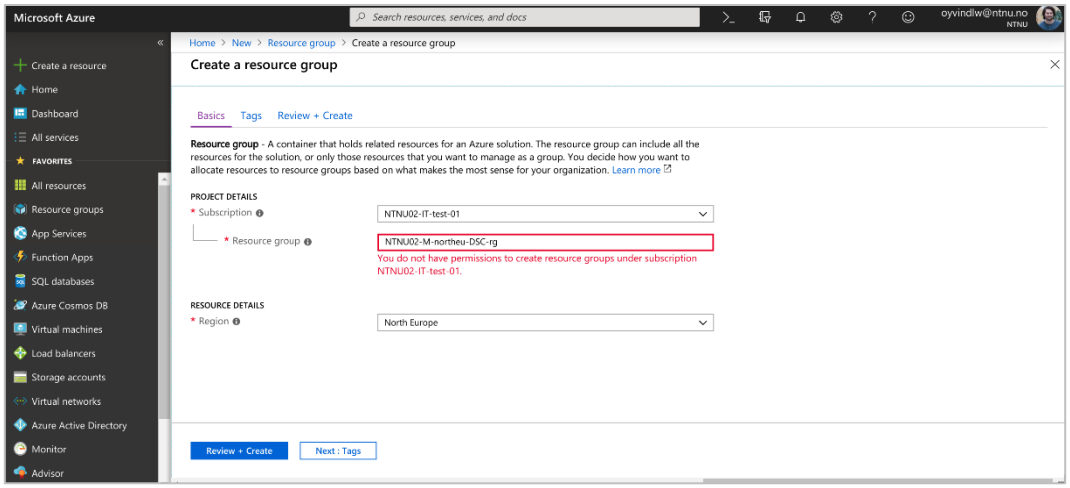
Figur 12 - Opprette Azure Resource Group

4. Velg “Create”.



Figur 13 - Opprette Azure Resource Group

5. Fyll ut feltene med ønsket informasjon og velg Create.
    a. Name: azure-dsc-aa
    b. Subscription: NTNU02-IT-test-01
    c. Resource group: NTNU02-M-northeu-DSC-rg
    d. Location: North Europe



Figur 14 - Opprette Azure Resource Group

2.2 Opprette Azure Automation Account

For å få tilgang til Azure Automations tjenester må det opprettes en Azure Automation konto. Azure Automation består av prosessautomasjon, oppdateringsbehandling og konfigurasjonsstyring. Prosjektet skal benytte seg av tjenesten konfigurasjonsstyring (Configuration Management) [2].

Tjenesten prises per administrerte node. Det koster 6 USD pr måned for on-premise-noder, men det er gratis for Azure-noder. Det er inkludert 5 stykk on-premise noder for bruk til testing og utvikling som prosjektet skal benytte. Kostnad for logging kommer i tillegg.

2.2.1 Opprette Azure Automation Account med PowerShell

Azure Automation Account opprettes med PowerShell kommando som visst på kodeblokk 2.

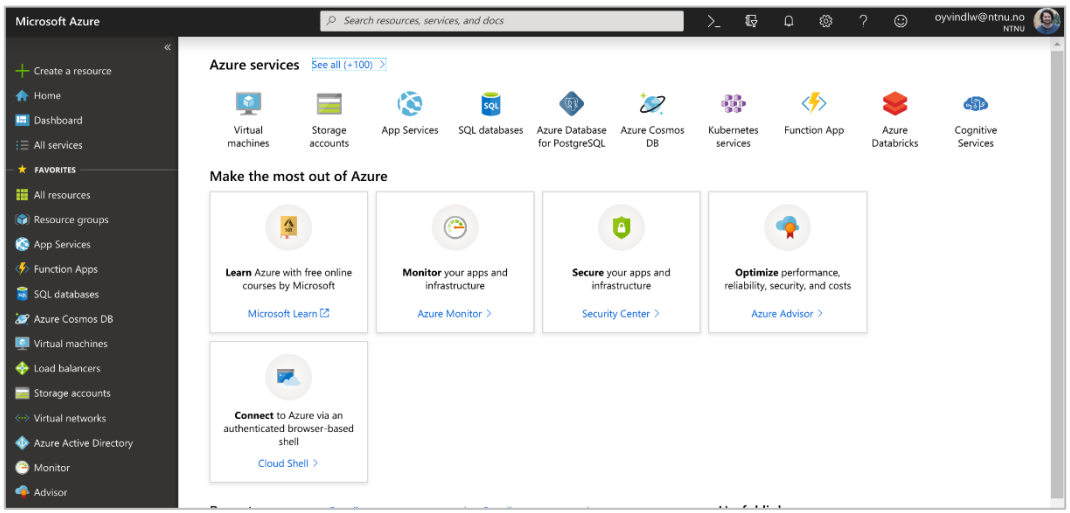
```
Connect-AzAccount
$Params = @{
    Name = "azure-dsc-aa";
    Location = "North Europe";
    ResourceGroupName = "NTNU02-M-northeu-DSC-rg";
}
New-AzAutomationAccount @Params
```

Kodeblokk 5 - Opprette Azure Automation Account

2.2.2 Opprette Azure Automation Account i portalen

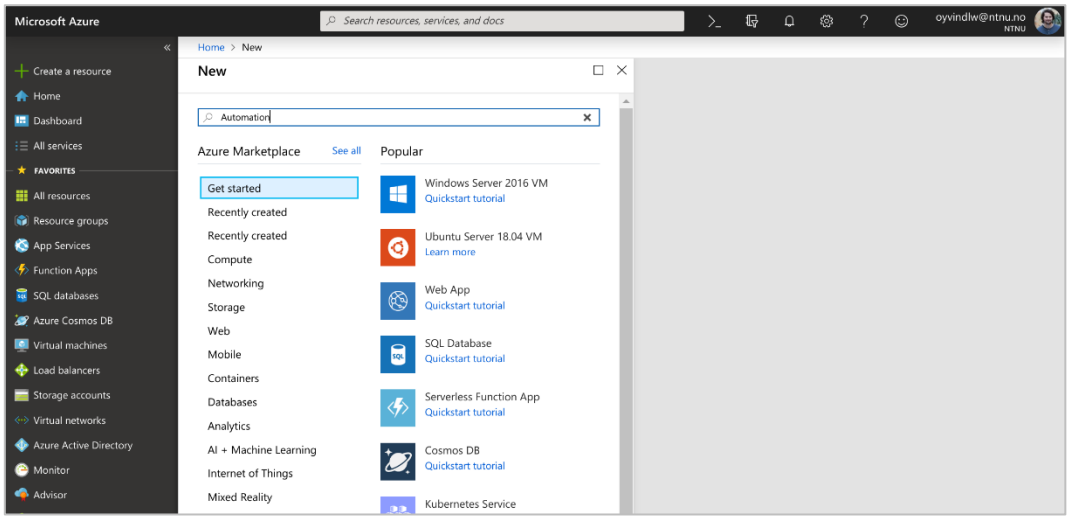
For å opprette en Azure Automation Account i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg "Create a resource".



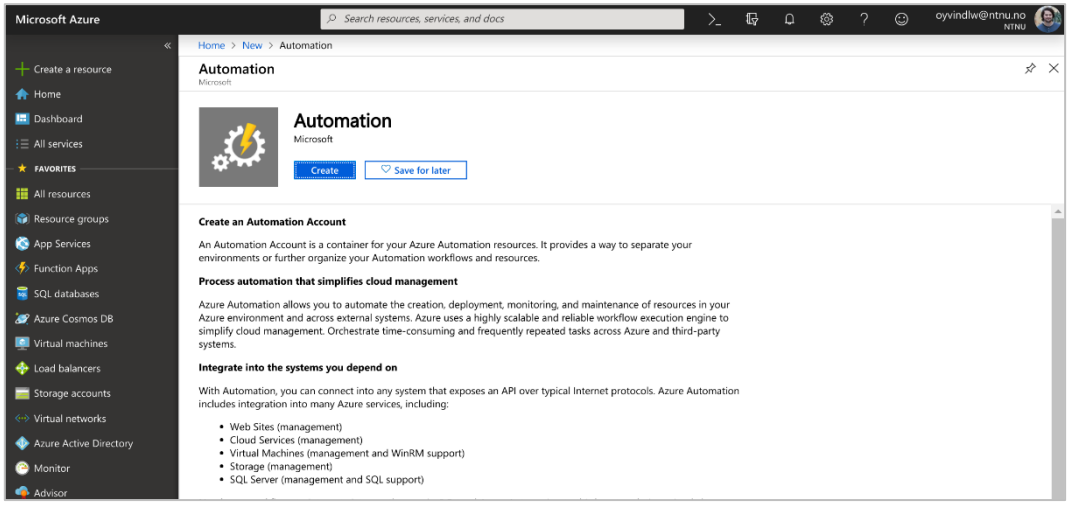
Figur 15 - Opprette Azure Automation Account

3. Skriv inn “Automation” i søkefeltet og trykk Enter.



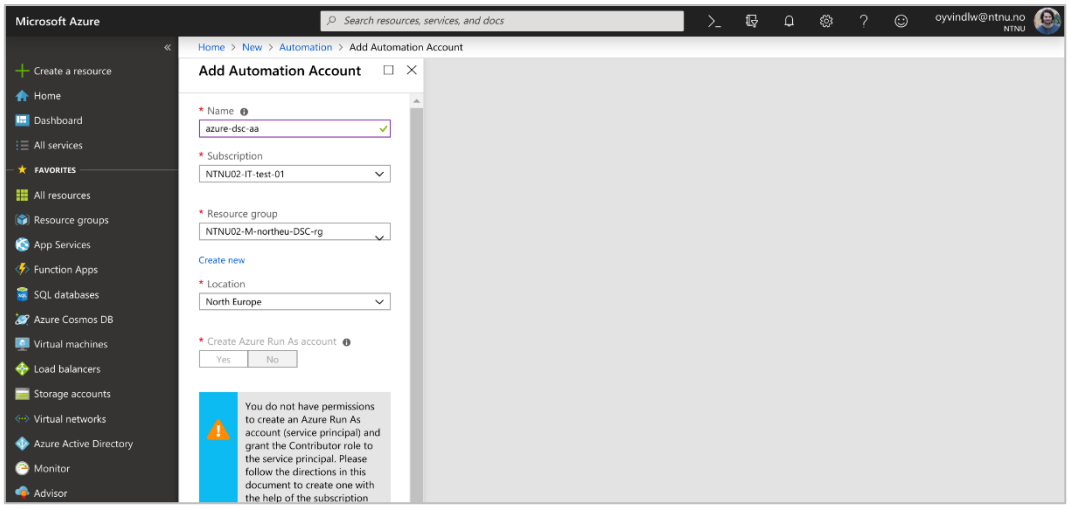
Figur 16 - Opprette Azure Automation Account

4. Velg Create.



Figur 17 - Opprette Azure Automation Account

5. Fyll ut feltene med ønsket informasjon og velg Create.
    a. Name: azure-dsc-aa
    b. Subscription: NTNU02-IT-test-01
    c. Resource group: NTNU02-M-northeu-DSC-rg
    d. Location: North Europe



Figur 18 - Opprette Azure Automation Account

2.3 Onboarding av node

Maskinene som er plassert on-premise skal administreres fra skyen av Azure DSC. Prosessen ved å legge maskinene inn i Azure DSC kalles onboarding. Maskinene må ha utgående internettilgang til Azure og Windows Management Framework 5.1 installert. Onboarding kan gjøres via Azure Automation CmdLets eller via "WMF 5 DSC Registration Protocol". Sistnevnte gir mulighet for å utføre en sikker registrering der det opprettes unike sertifikat for hver node, som benyttes til autentisering av maskinen mot Azure [4].

Når maskinene onboardes konfigureres det en rekke innstillinger vedrørende oppdateringsfrekvens, konfigurasjonsfrekvens og hvilken modus noden skal ha. Dette er innstillinger som lagres i Local Configuration Manager (LCM).

Innstilling	**Forklaring**
NodeConfigurationName	Hvilken DSC Node Configuration som skal konfigureres.
RefreshFrequency	Tiden for hvor ofte LCM skal sjekke Pull Server om det har kommet ny konfigurasjon.
ConfigurationMode	Definerer hvordan LCM bruker konfigurasjonen. ApplyOnly ApplyAndMonitor ApplyAndAutoCorrect
ConfigurationModeFrequency	Tiden for hvor ofte LCM kontrollerer om konfigurasjonen har blitt endret.
RebootNodeifNeeded	Definere om LCM kan utføre restart av noden hvis det er behov i f.eks. en installasjonsprosess.
ActionAfterReboot	Definere hvilken handling LCM skal utføre etter en restart som er utløst av DSC. Man kan velge mellom å stoppe eller fortsette installasjon.

Tabell 19 - Konfigurasjonsmuligheter i LCM

Innstillingen ConfigurationMode inneholder tre valg:

- **ApplyOnly** betyr at LCM bare konfigurerer noden
- **ApplyAndMonitor** betyr at i tillegg til å konfigurere noden skal også LCM melde tilbake hvis konfigurasjonen drifter fra den ønskede tilstanden.
- **ApplyAndAutoCorrect** betyr at i tillegg til å konfigurere noden skal også LCM rekonfigurere noden hvis konfigurasjonen drifter fra den ønskede tilstanden.

2.3.1 Onboarding ved bruk av DSC-konfigurasjon

Den ene metoden for å onboarde maskiner er å lage en DSC-konfigurasjon som konfigurerer Local Configuration Manager (LCM). Denne konfigurasjonen defineres ved å sette inn "[DscLocalConfigurationManager()]" i starten av scriptet. Scriptet er tilgjengelig for nedlastning på dokumentasjonssidene til Azure Automation[6]. For å benytte DSC-konfigurasjon må man ha "URL endpoint" og "Primary access key" til Azure Automation kontoen. Dette kan enten hentes via PowerShell eller via portalen.

Hente URL Endpoint og Primary Access Key med PowerShell

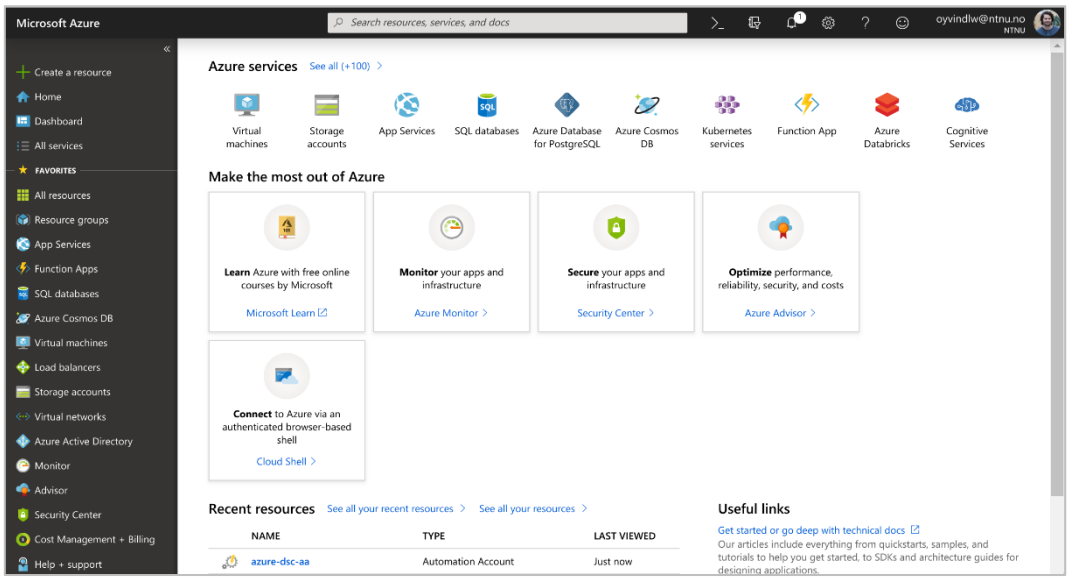
For å hente ut "URL-Endpoint" og "Primary access key" for Azure Automation Account med PowerShell kjøres PowerShell-kode som vist i kodeblokk 3.

```
Connect-AzAccount
$Params = @{
    ResourceGroupName = "NTNU02-M-northeu-DSC-rg";
    AutomationAccountName = "azure-dsc-aa";
}
Get-AzAutomationRegistrationInfo @Params
```

Kodeblokk 6- Hente URL Endpoint og Primary Access Key

Hente URL Endpoint og Primary Access Key i Azure Portalen

1. For å hente ut "URL-Endpont" og "Primary access key" for Azure Automation Account i Azure portalen må følgende fremgangsmåte følges:Velg den Azure Automation Account du ønsker at nodene skal onboardes til.

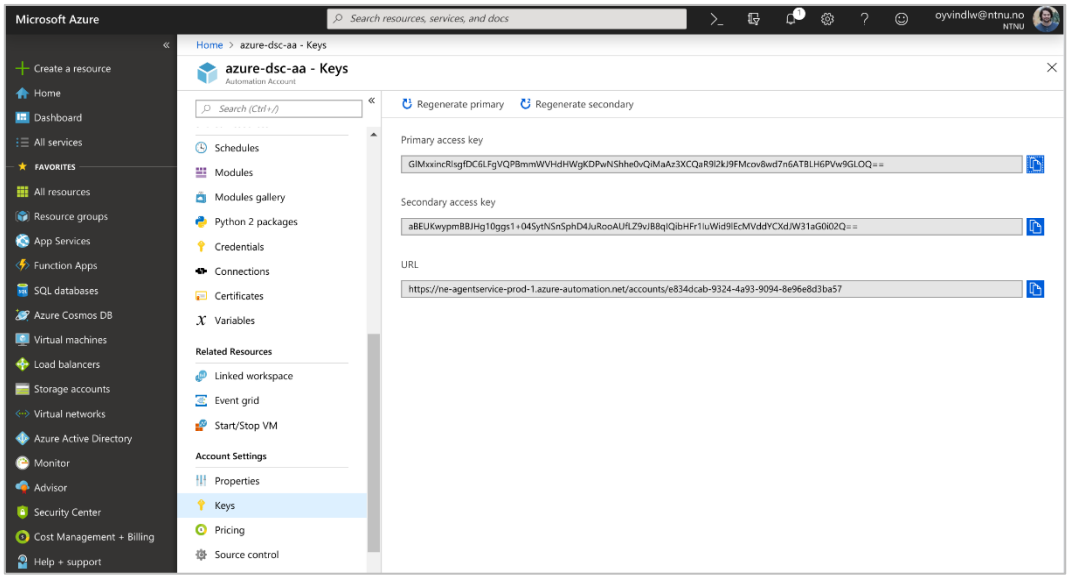


Figur 19- Hente URL Endpoint og Primary Access Key

[6] https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding#generating-dsc-metaconfigurations

2. Velg “Keys” under “Account Settings” i sidemenyen til venstre.
3. Kopier URL og Primary access key.

Access keys er regenerert i etterkant av skjermbilde.

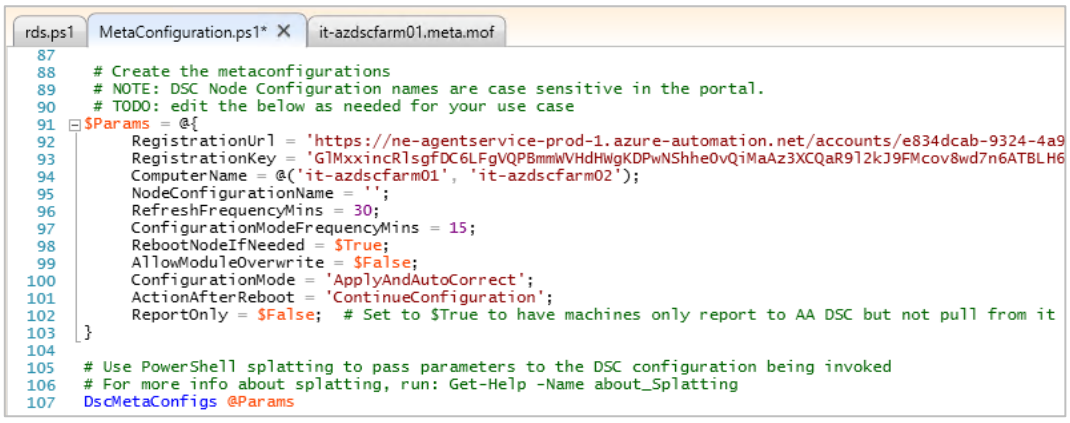


Figur 20- Hente URL Endpoint og Primary Access Key

Kompilere MetaConfiguration

Før Meta-konfigurasjonen kan kjøres på Local Configuration Manager (LCM) må konfigurasjonen kompileres. Dette utføres på følgende måte:

1. Lim inn URL og Primary access key.
2. Skriv inn navn for nodene som skal onboardes.
3. Konfigurer de forskjellige parameterne etter behov.
4. Kjør script for å kompilere MetaConfiguration.



Figur 21 - Kompilere MetaConfiguration

Scriptet oppretter en mappe som inneholder en Meta-MOF-fil for hver node som er definert i konfigurasjonen.

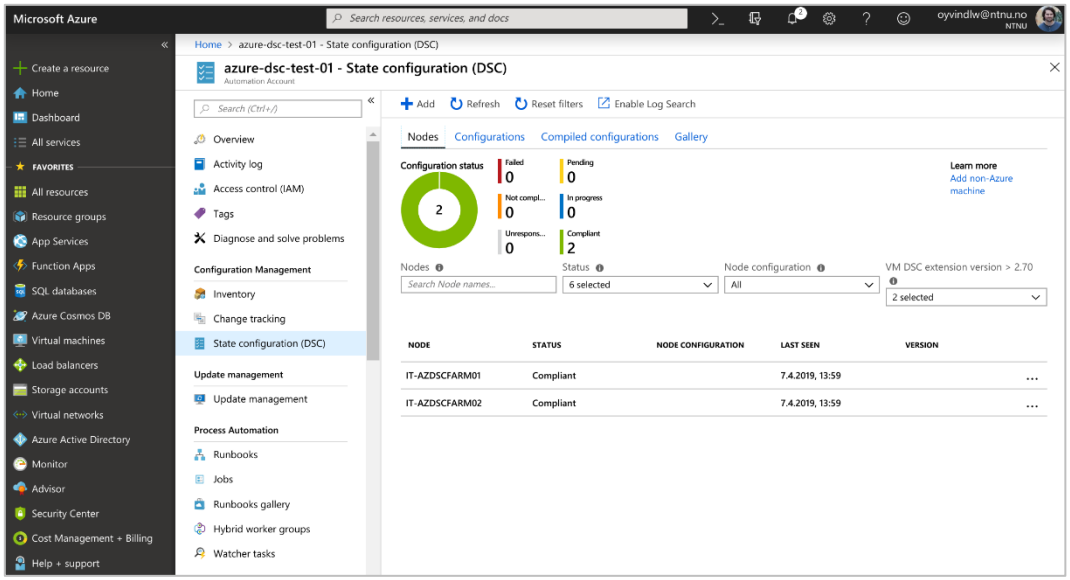
Konfigurere Local Configuration Manager (LCM)

Det er på dette punktet at maskinen legges inn i Azure DSC. For å konfigurere Local Configuration Manager (LCM) benyttes følgende PowerShell-kommando visst i kodeblokk 4. Parametere "Path" defineres for å vise hvilken mappe konfigurasjonen man ønsker å benytte ligger lagret. Ønsker man kun å konfigurere én maskin kan man benytte parametere "ComputerName" for å filtrere ut dette.

```
Set-DscLocalConfigurationManager -Path .\DscMetaConfigs -Verbose
```

Kodeblokk 7 - Konfigurere LCM

Nodene vil nå dukke opp under Nodes på Azure Automation Account som visst på figur 12.



Figur 22 - Nodene er onboardet

2.3.2 Onboarding med bruk av Az CmdLets

Onboarding med bruk av Az CmdLets er enklere, men man får ikke mulighet til å velge noen innstillinger for noden. Disse innstillingene kan kun settes ved Onboarding ved bruk av DSC-konfigurasjon eller manuell endring i MetaMof-filen.

URL Endpoint og Primary access key hentes fra Azure Automation og konfigurasjonen kompileres med bruk av CmdLet-en "Get-AzureAutomationDscOnboardingMetaConfig". Local Configuration Manager (LCM) må konfigureres med CmdLet "Set-DscLocalConfigurationManager" for at noden skal ta i bruk konfigurasjonen.

```
Connect-AzAccount
# Define the parameters
$Params = @{
    ResourceGroupName = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName = 'azure-dsc-test-01';
    ComputerName = @('it-azdscfarm01', 'it-azdscfarm02');
    OutputFolder = "$env:UserProfile\Desktop\";
}
# Use PowerShell splatting to pass parameters to the Azure Automation
cmdlet being invoked
Get-AzAutomationDscOnboardingMetaconfig @Params

Set-DscLocalConfigurationManager -Path
$env:UserProfile\Desktop\DscMetaConfigs
```

Kodeblokk 8 - Script for å onboarde node

2.3.3 Re-onboarding av node

Det vil være behov for å re-registrere nodene til Azure DSC fra en tid til annen. Dette skyldes enten at man ønsker å endre innstillingene i LCM eller at autentiseringssertifikatet for når man første gangen onboarded maskinen er utløpt. Sertifikatet har en varighet på 12 måneder. Når maskinens autentiseringssertifikat utløper vil maskinen få status som "Unresponsive" i Azure DSC. Per dags dato er det ikke mulig å endre på LCM eller fornye sertifikatet uten å re-onboarde nodene [9].

2.3.4 Tilpasset onboarding ved bruk av konfigurasjonsdata

Etter tilbakemelding fra oppdragsgiver var det behov for et lokalt system som lagret informasjon om nodene. Denne informasjonen kunne være operativsystem, Session Collection og konfigurasjon. Ønsket kom siden portalen raskt blir uoversiktlig når er over 10-20 administrerte noder. Det blir også et problem å huske hvilken konfigurasjon en node har, hvis en node skal re-onboardes eller oppdateres. Dette fører til mye manuelt klipp-lim-arbeid og stort rom for menneskelige feil.

Løsningen er å utvide funksjonaliteten i onboarding-scriptet som Microsoft har gjort tilgjengelig med konfigurasjonsdata. Konfigurasjonsdataen består av data lagret i et hierarki av matrise og hash-tabeller, hvor man for eksempel kan definere informasjon om en node. Variablene NodeName, Collection, OS, Configuration og ConfigurationMode er nå

tilgjengelig pr node. Siden de kompilerte konfigurasjonene er navngitt av "$Collection.$OS" er det nå mulig å bygge "NodeConfigurationName" opp fra tilgjengelig informasjon definert i konfigurasjonsdataen. Det er også inkludert "ConfigurationMode" i konfigurasjonsdataen, siden det da gir mulighet til å enkelt ha forskjellig konfigurasjonsmodus på forskjellige noder.

Med denne løsningen blir det enklere å holde oversikten over nodene som er administrert i Azure DSC. Det gir muligheten til å enkelt re-onboarde nodene med tidligere egenskaper siden dataen er lagret uavhengig av Azure DSC. I tillegg blir det enklere hvis nodene skal migreres over på alternative løsninger eller andre Automation Accounts i fremtiden. Konfigurasjonsdataen åpner også for en del forenkling når det kommer til kjøring av script mot Azure DSC siden man allerede har dataen tilgjengelig i lokale variabler. Se kapittel 6.3.1 for eksempel for bruk av konfigurasjonsdata i script.

Løsningen skaper også mer administrasjon siden man nå må håndtere data to steder. Det er viktig at konfigurasjonsdataen vedlikeholdes fortløpende hvis for eksempel operativsystemet oppgraderes til en nyere versjon.

Fullstendig kildekode til onboarding-scriptet ligger vedlagt.

```
$ConfigData =
@{
    AllNodes =
    @(
        @{
            NodeName            = "it-azdscfarm01"
            Collection          = "adminfarm"
            OS                  = "srv2016"
            Configuration       = "SessionHost"
            ConfigurationMode   = "ApplyAndMonitor"
        },
        @{
            NodeName            = "it-azdscfarm02"
            Collection          = "calcfarm"
            OS                  = "srv2019"
            Configuration       = "SessionHost"
            ConfigurationMode   = "ApplyAndAutoCorrect"
        }
    );
}
```

Kodeblokk 9 - Bruk av konfigurasjonsdata i onboarding

2.4 Funksjonalitet i Azure Portalen

Azureportalen er fullspekket med funksjonalitet når det kommer til Azure DSC. Dette underkapittelet vil gå over og gi informasjon om den viktigste funksjonaliteten.

2.4.1 Feilsøking og overvåking av noder

Portalen gir god oversikt over de administrerte nodene. Det er en grafisk oversikt med kakediagram med de forskjellige konfigurasjonsstatusene en node kan inneha. I tillegg til konfigurasjonsstatus vil man se hvilken konfigurasjon nodene er tildelt. Oversikten inneholder en søkefunksjon der det er mulig å søke på nodenavn, filtrere etter konfigurasjonsstatus eller tildelt konfigurasjon.

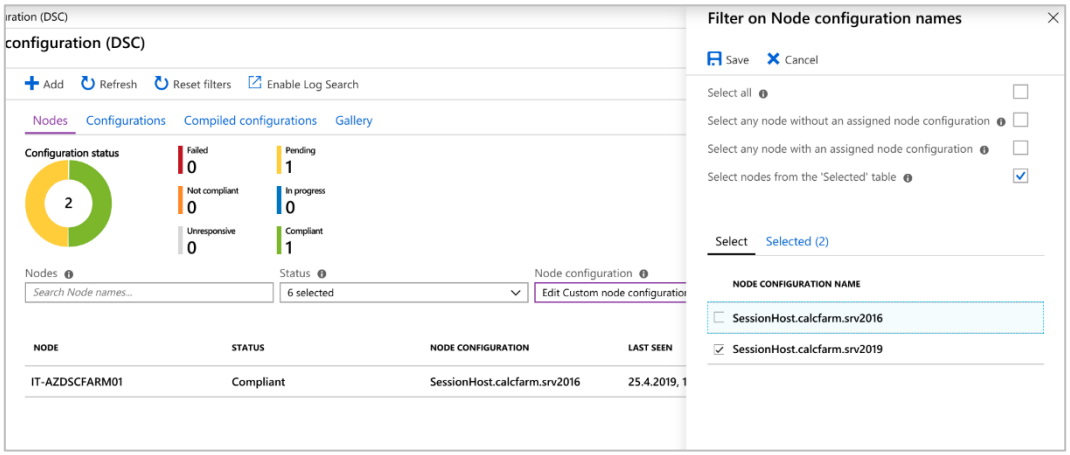
Søke og filtrere ut noder etter tildelt konfigurasjon

Søkefunksjonen gjør det enkelt å filtrere ut noder med samme konfigurasjon. Dette er visst på figur 13. Her ser man alle nodene som er konfigurert med konfigurasjonen "SessionHost.calcfarm.srv2019". Dette er da en Session Host i Calcfarm som kjører operativsystemet Windows Server 2019. Ifølge navnestandarden som er forklart i kapittel 5.4.1.

Søket blir utført på følgende måte:

1. Velger "Custom node configuration" i rullgardinmenyen "Node configuration".
2. Haker på "Select nodes from the Selected table"
3. Haker på ønsket konfigurasjon.
4. Velger "Save"

Ønsker man å endre filtreringen velger man "Edit custom node configuration" i rullgardinmenyen "Node configuration". Søket resettes ved å velge "Reset filters".



Figur 23 - Filtrering av nodekonfigurasjon

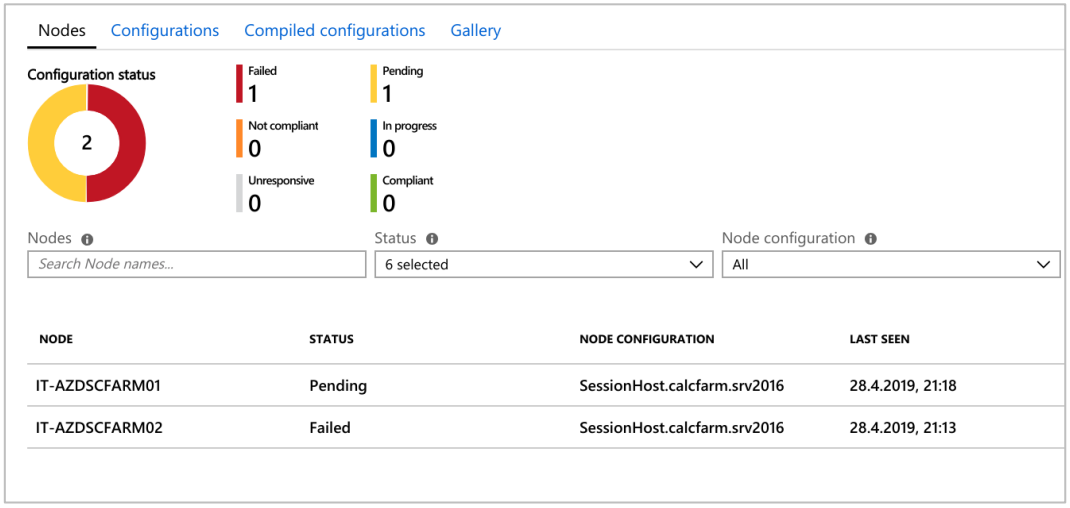
Konfigurasjonsstatus

Portalen viser nodens konfigurasjonsstatus. De forskjellige kan ses i tabell 2. Statusene representerer hvilken tilstand nodene er i for øyeblikket. Ikke alle statusene vil være tilgjengelig for enhver Local Configuration Manager (LCM) konfigurasjon. For eksempel hvis nodene er konfigurert med konfigurasjonsmodusen "ApplyAndAutoCorrect", vil aldri konfigurasjonsstatusen "Not Compliant" være i bruk siden Local Configuration Manager vil sette noden tilbake i den ønskede status før konfigurasjonsstatusen blir satt.

Konfigurasjonsstatus	**Forklaring**
Compliant	Noden er i den ønskede tilstand.
Not compliant	Noden er IKKE i den ønskede tilstand.
In progress	Noden er i ferd med å bli konfigurert.
Pending	Noden har registrert en ny konfigurasjon, men venter på at LCM skal initiere rekonfigurasjon.
Failed	Konfigurasjon av en node har feilet.
Unresponsive	Noden har ikke tilknytning til Pull-serveren

Tabell 20 – Konfigurasjonsstatus

Portalen med noder som har forskjellig konfigurasjonsstatus vises på figur 14. Her ser man at det er en node som er "Compliant" og en node som er "Pending". Etter kort tid vil IT-AZDSCFARM01 gå over i "In Progress". Da vil konfigurasjonen, konfigureres på noden. Når konfigurasjonen er ferdig vil den få status "Compliant".



Figur 24 - Konfigurasjonsstatus i Azure

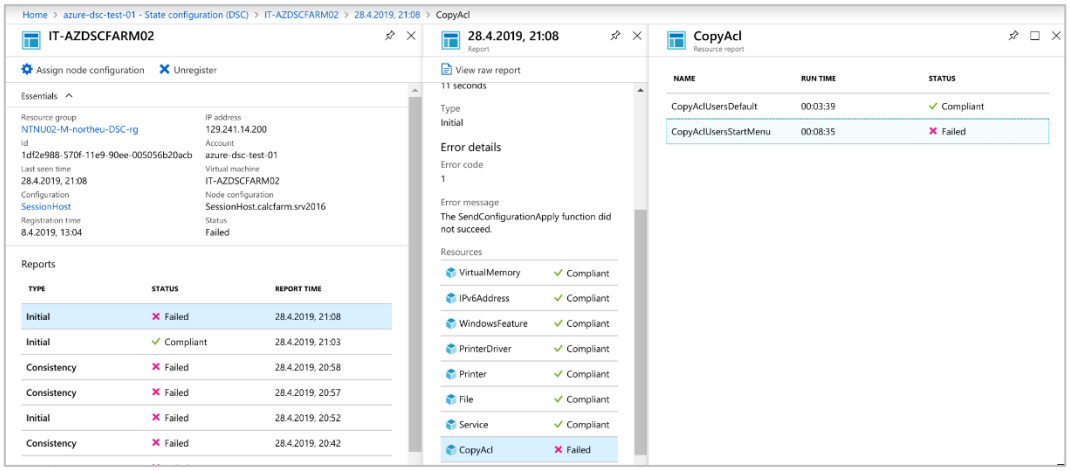
Feilsøke node

Hvis en node har konfigurasjonsstatus "Failed" kan man via portalen undersøke og feilsøke hva som har gått galt. Ved å trykke seg lengre ned i hierarkiet kan man se detaljerte feilmeldinger som gir en god beskrivelse for hva som er galt. I dette tilfellet er det

konfigurasjonsblokka “CopyAcl CopyAclUsersStartMenu” som ikke har tilgang til å sette Aksess Control List for et objekt.

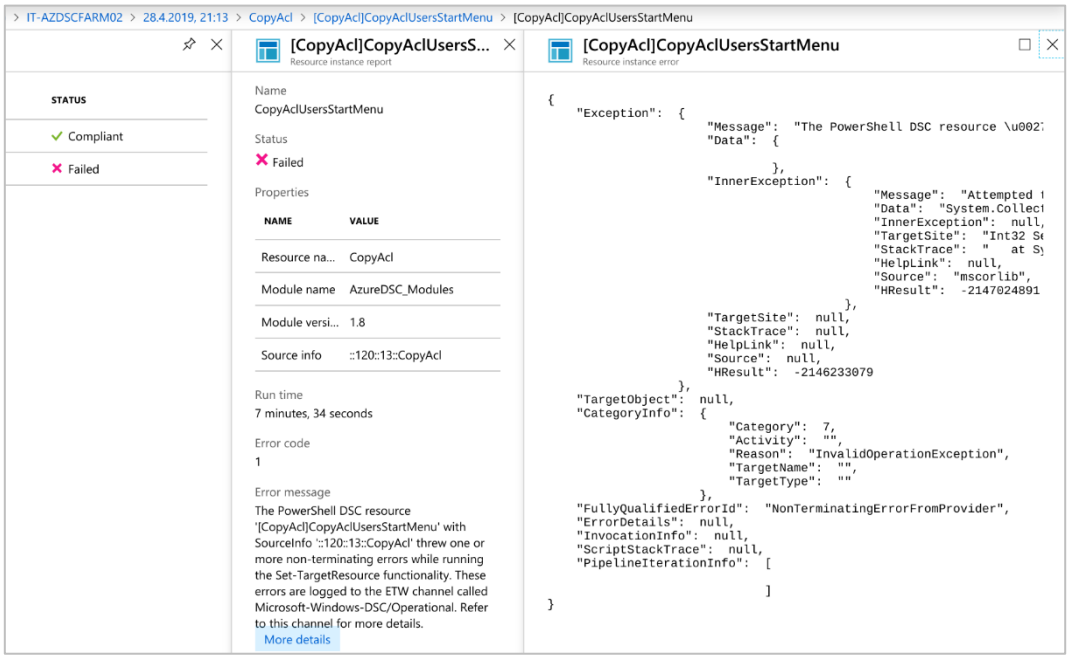
For å feilsøke en node i Azure Automation må følgende fremgangsmåte følges:

1. Velg den noden som har status “Failed”. Se figur 14.
2. Velg den rapporten som har status “Failed”. Se figur 15.
3. Velg den DSC-ressursen som har status “Failed”. Se figur 15.
4. Velg den konfigurasjonsblokka som har status “Failed”. Se figur 15.



Figur 25 - Feilsøke node

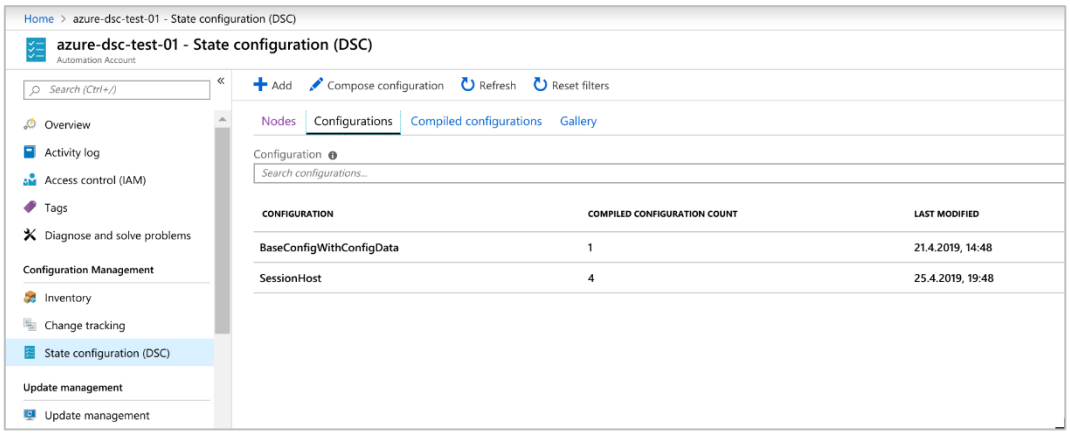
5. Her vil man se:
    a. DSC-Ressurs
    b. Hvilken modul DSC-ressurser er en del av inklusive versjonsnummer
    c. Error-kode
    d. Overordnet error-melding
6. Velg “More details” for ytterligere detaljert feilmelding.



Figur 26 - Feilsøke node

2.4.2 Opplastning og administrasjon av konfigurasjonsfiler

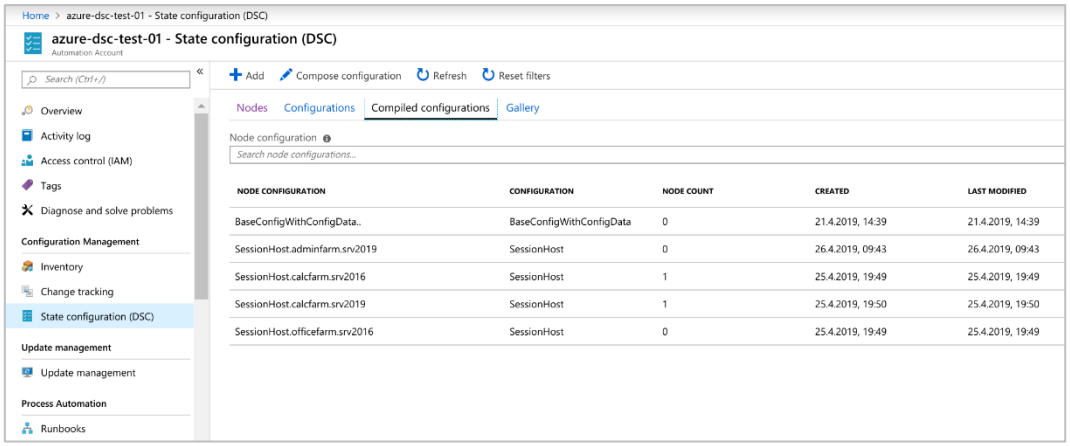
Konfigurasjonsfanen i State Configuration (DSC) i Azure Automation som presentert på figur 17 viser de opplastede konfigurasjonsfilene. Dette er PowerShell (*.ps1) filene som inneholder hva som skal gjøres, ikke de kompilerte MOF-dokumentene. Oversikten viser også hvor mange kompilerte konfigurasjoner hver konfigurasjonsfil har.



Figur 27 – Konfigurasjonsdokument

2.4.3 Kompilerte konfigurasjoner

Kompilerte konfigurasjoner i State Configuration (DSC) i Azure Automation som presentert på figur 18 viser de kompilerte konfigurasjonene (*.MOF) som er tilgjengelig på Pull-serveren. Oversikten viser også hvor mange noder hver konfigurasjon har.



Figur 28 - Kompilerte konfigurasjonsdokument

2.4.4 Legitimasjon

Man kan møte problemstillinger der systemkontoen "NT AUTHORITY\SYSTEM" ikke har tilgang til å utføre operasjoner som er definert i DSC-konfigurasjonen. Dette kan for eksempel være hvis systemkontoen ikke har tilgang til et filområde eller hvis det er en oppgave som behøver domeneadministratorrettigheter.

For å løse dette er det mulig å legge inn legitimasjon i Azure Automation. Legitimasjonen kan hentes inn konfigurasjonen med PowerShell-kommandoen "Get-AutomationPSCredential" som lagrer legitimasjonen i et PSCredential-objekt som videre kan brukes for å kjøre konfigurasjonsblokker med DSC-ressur-egenskapen "PsDscRunAsCredential". Et eksempel på dette kan ses i kapittel 4.2.1.

Når man behandler legitimasjon er det viktig å ta hensyn til sikkerheten. Azure DSC sikrer legitimasjon, overføring av data og MOF-filene som standard. Det betyr at i Azure Automation og når MOF-filene ligger lagret på nodene blir legitimasjonene kryptert med krypteringsnøkkelen som ligger i Azure Key Vault og i transitt mellom Azure DSC og noden er kommunikasjonen alltid HTTPS [10].

Microsoft anbefaler ikke å lagre domenelegitimasjon i Azure Automation. Beste praksis er å så langt det lar seg gjøre å benytte en lokal brukerkonto med forhøyede rettigheter [11].

Azure Run As Account

For å sikre autentisering mot Azure når man bruker Azure PowerShell-CmdLets kan man benytte en Azure Run As Account. Da får man muligheten til å autentisere sikkert med et sertifikat og et "Service Principal Names" i stedet for å benytte brukernavn og passord. Opprettelse av en Run As Account gjør følgende:

- Oppretter en Azure AD applikasjon, med et selvsignert sertifikat, en "Service Principal Account" for applikasjonen i Azure AD som tildeles "Contributor"-rollen.
- Oppretter sertifikatet "AzureRunAsCertificate" for Automation-kontoen.
- Oppretter en "Automation Connection Asset" med navn "AzureRunAsConnection".

Tilgangen til "Run As Service Principal" kan begrenses til "Contribrutor" for Ressursgruppen i etterkant av opprettelse [3].

Azure Run As Account er ikke testet ut i prosjektet.

Legge inn legitimasjon med PowerShell

Legitimasjon opprettes med PowerShell kommandoer som visst på kodeblokk 7.

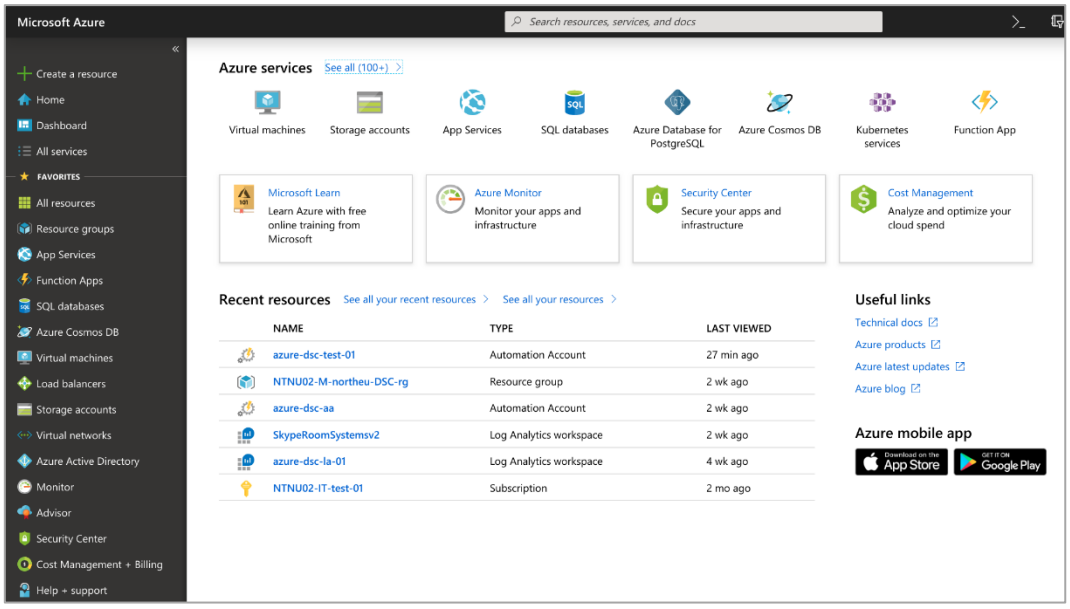
```
$User = "win-ntnu-no\testbruker"
$Password = ConvertTo-SecureString "Password" -AsPlainText -Force
$CredParams = @{
    TypeName = System.Management.Automation.PSCredential;
    ArgumentList = $User, $Password;
}
$Credential = New-Object $CredParams

Connect-AzAccount
$Params = @{
    AutomationAccountName = "azure-dsc-aa";
    Name = "Testbruker";
    Value = $Credential;
    ResourceGroupName = "NTNU02-M-northeu-DSC-rg";
}
New-AzAutomationCredential @Params
```

Kodeblokk 10 - Legge inn legitimasjon i Azure Automation

Legge inn legitimasjon i portalen

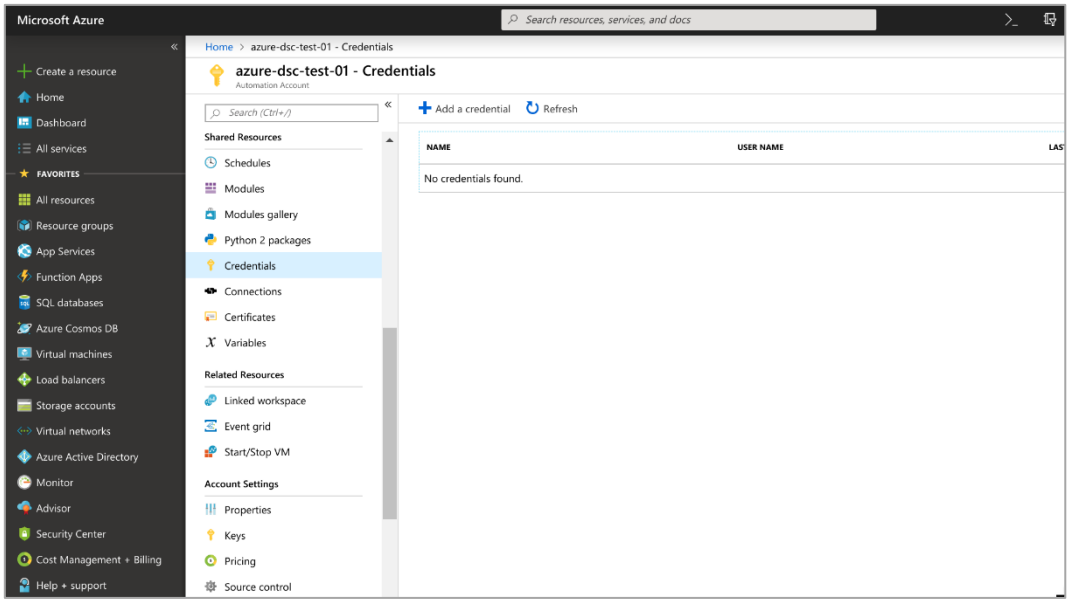
For å legge til en legitimasjon i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg Automation Account



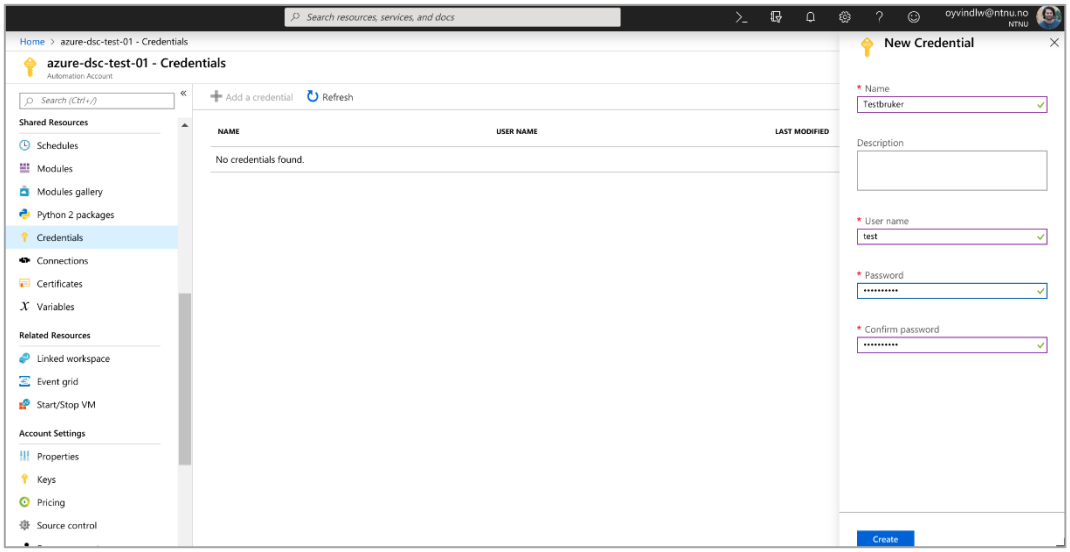
Figur 29 - Legge inn legitimasjon

3. Velg “Credentials” under “Shared Resources” i menyen til venstre
4. Velg “Add a credentials”



Figur 30 - Legge inn legitimasjon

5. Fyll ut feltene med ønsket informasjon og velg Create.
    a. Name: testbruker
    b. User Name: test
        i. Ønsker man å legge inn en domenebruker gjøres dette ved å skrive “domene\brukernavn”
    c. Passord: Velg et passord
    d. Comfirm password: Gjenta passord



Figur 31 - Legge inn legitimasjon

2.4.5 Moduler og DSC-ressurser

De DSC-ressurser som skal benyttes i konfigurasjonen må lastes inn i Azure Automation for at konfigurasjonen skal fungere. Det er allerede en rekke DSC-ressurser inkludert som kan benyttes uten å foreta seg noe.

Se tilgjengelige moduler med PowerShell

For å se tilgjengelige moduler i Azure Automation kan PowerShell kommando som visst på kodeblokk 8 kjøres.

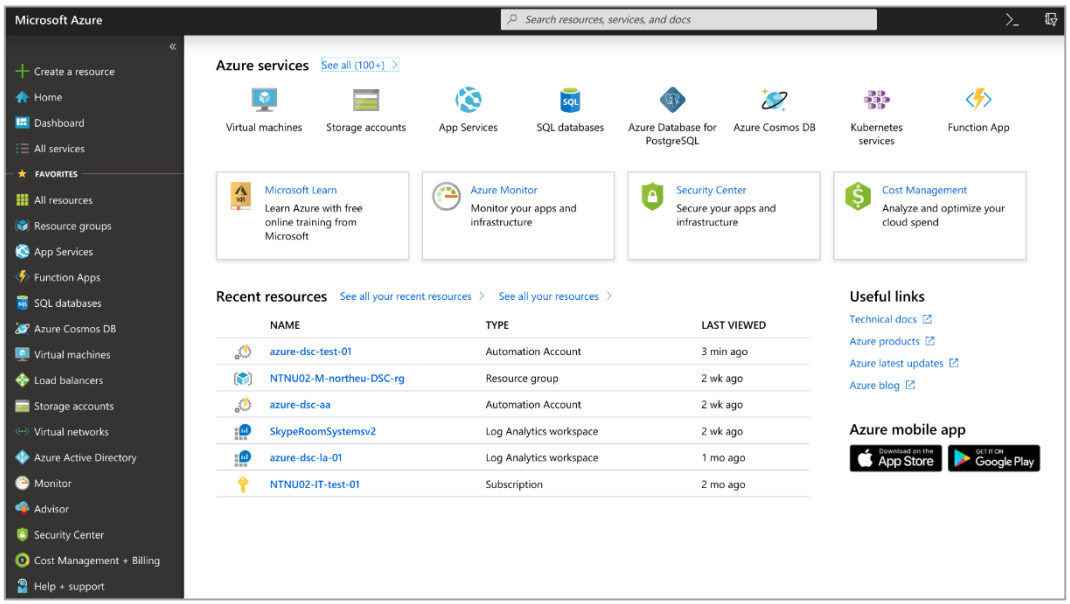
```
Connect-AzAccount
$Params = @{
    AutomationAccountName = "azure-dsc-aa";
    ResourceGroupName = "NTNU02-M-northeu-DSC-rg";
}
Get-AzAutomationModule @Params
```

Kodeblokk 11 - Se tilgjengelige moduler

Se tilgjengelige moduler og DSC-ressurser i Portalen

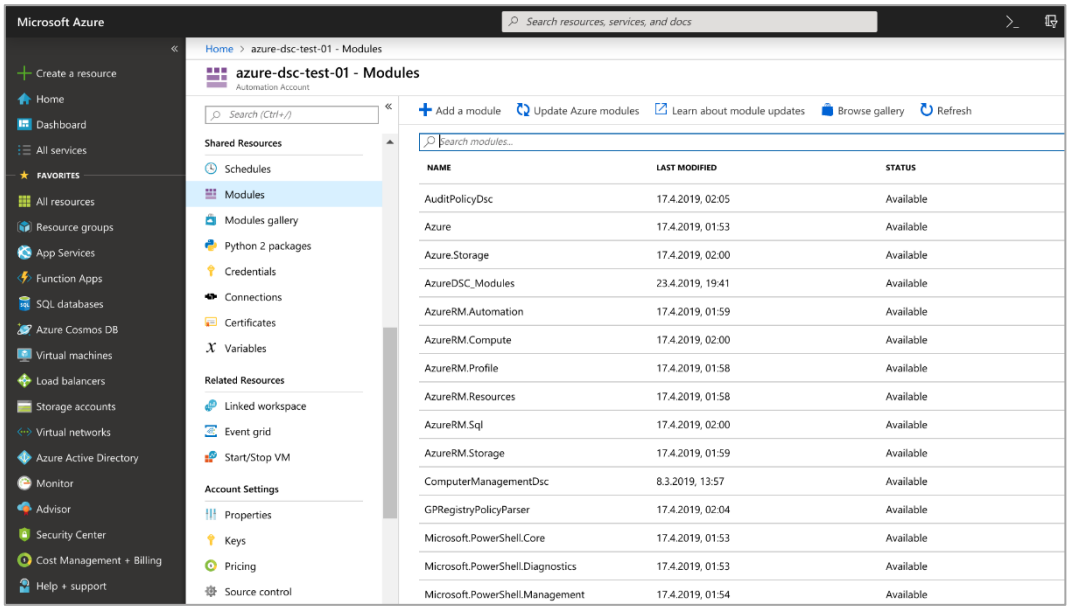
For å se tilgjengelige moduler og DSC-ressurser i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg ønsket Automation Account



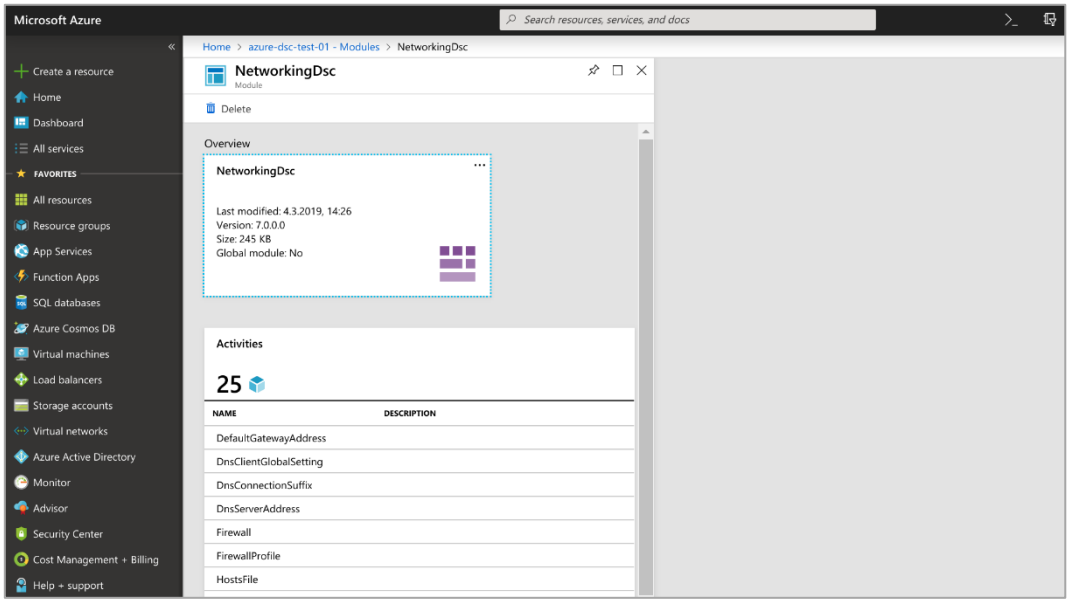
Figur 32 - Se tilgjengelige moduler

3. Velg “Module” under “Shared Resources” i menyen til venstre



Figur 33 - Se tilgjengelige moduler

4. Velg ønsket modul for å se hvilke DSC-ressurser som er tilgjengelig.

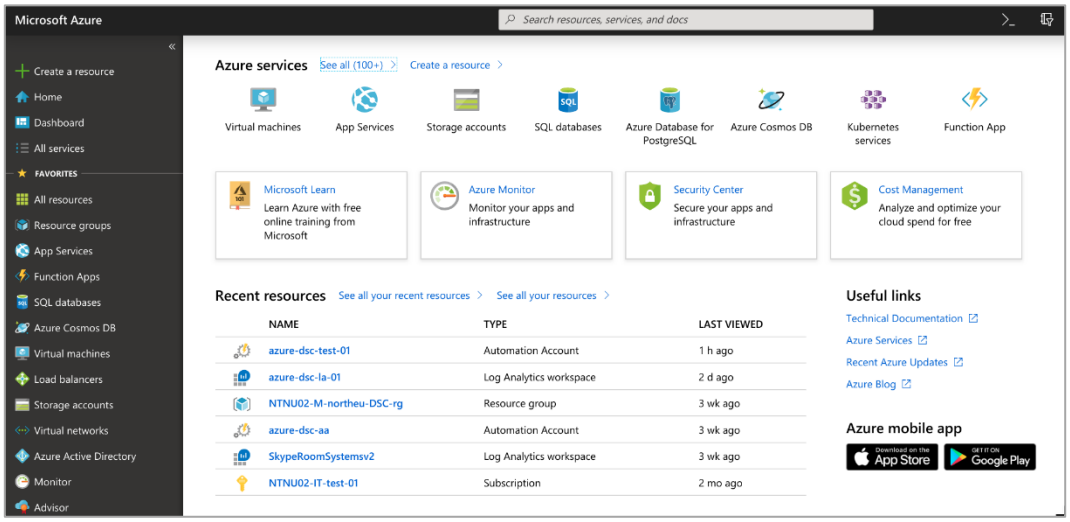


Figur 34 - Se tilgjengelige moduler

Laste opp Moduler og DSC-ressurser i Portalen

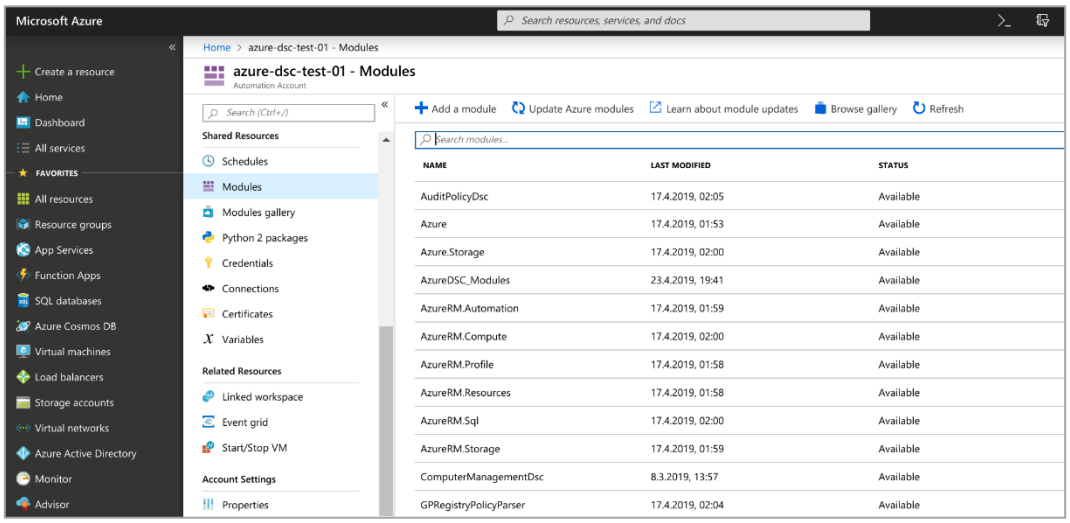
For å legge til en modul i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg Automation Account



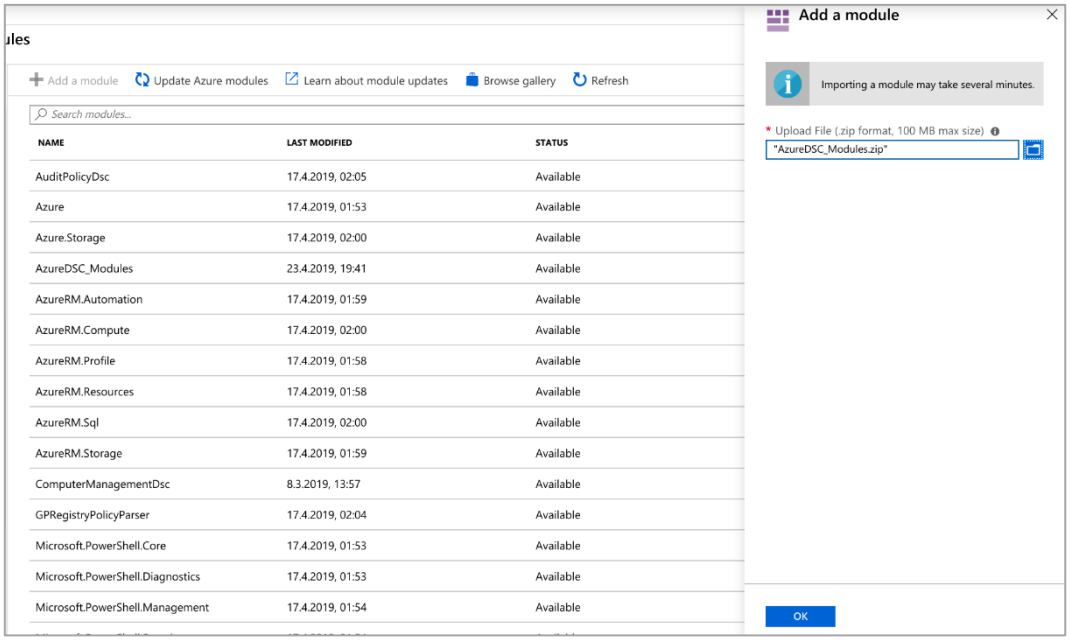
Figur 35 - Legge inn moduler

3. Velg “Module” under “Shared Resources” i menyen til venstre
4. Velg “Add a module”



Figur 36 - Legge inn moduler

5. Velg modulen som skal lastes opp. Må være en komprimert fil (*.zip) og avslutt med OK.



Figur 37 - Legge inn moduler

2.4.6 Utvidet Logging

Azure DSC kan kobles til Azure Log Analytics Workspace for å få utvidet loggmuligheter. Det er store muligheter for utvidede funksjonalitet ved å benytte seg av allerede utviklede Azure applikasjoner, som kan installeres i Portalen. Det er for eksempel mulig å sette opp automatisk varsling hvis nodene endrer konfigurasjonsstatus og lignende. Prosjektet har dessverre har ikke fått testet denne funksjonaliteten fullt ut. Det har ikke blitt prioritert og det ser ut til at det har vært noe problemer med tilgang til å bestille applikasjoner i Azure [8].

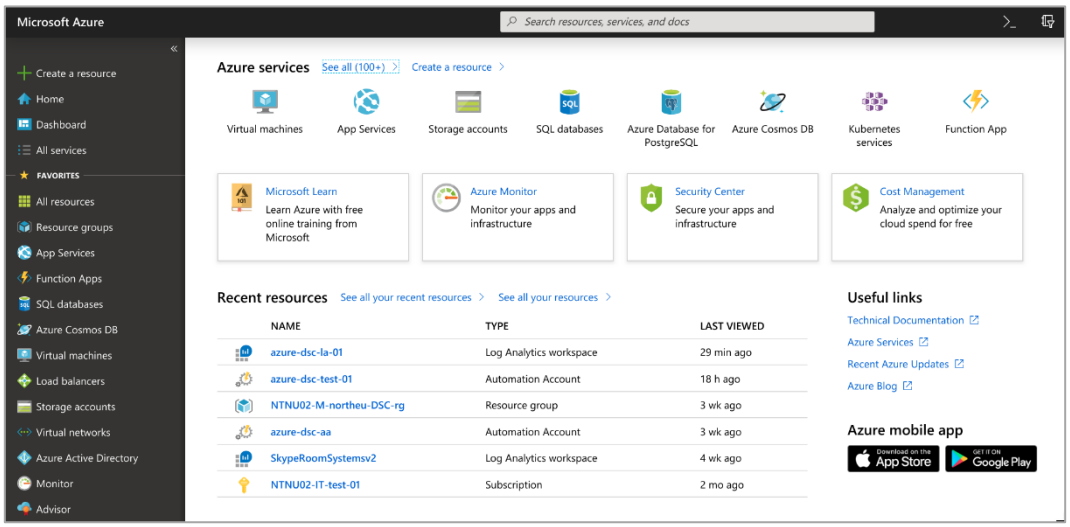
For å sette opp integrasjon mellom Azure Automation og Azure Log Analytics Workspace må PowerShell-kode i kodeblokk 9 kjøres. Parametere Navn for Automation Account og Log Analytics Workspace må tilpasses situasjonen.

```
$Login = Connect-AzAccount
# Find the ResourceId for the Automation Account
$Params = @{
    ResourceType = 'Microsoft.Automation/automationAccounts';
    Name         = 'azure-dsc-test-01'; # Tilpass denne til ønsket AA
}
$aaResrouceId = Get-AzResource @Params

# Find the ResourceId for the Log Analytics workspace (LAW)
$Params = @{
    ResourceType = 'Microsoft.OperationalInsights/workspaces';
    Name         = 'azure-dsc-la-01'; # Tilpass denne til ønsket LAW
}
$laResourceId = Get-AzResource @Params

# Enabe logging
$Params = @{
    ResourceId   = $aaResrouceId.ResourceId;
    WorkspaceId  = $laResourceId.ResourceId;
    Enabled      = $true;
    Category     = 'DscNodeStatus';
}
Set-AzDiagnosticSetting @Params
```

Kodeblokk 12 - Aktivere utvidet logging

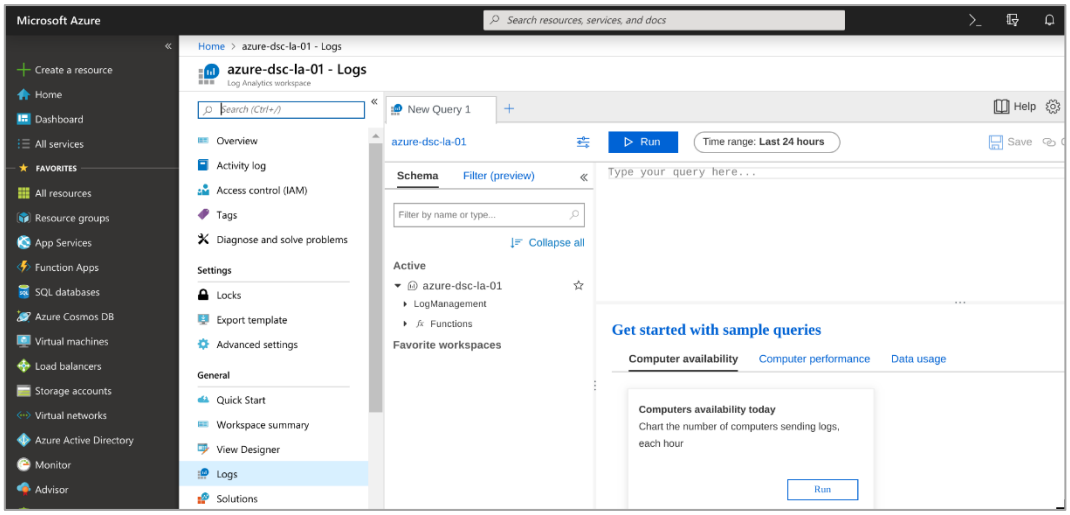
Dette vil gjøre at Azure Automation nå sender logg med kategorien «DscNodeStatus» til den spesifiserte Log Analytic Workspace. Dette er konfigurasjonsstatusen til nodene. Man kan gjøre et søk i Azure Log Analytics Workspace på følgende måte:

1. Åpne og logg inn på portal.azure.com.
2. Velg “Log Analytics Workspace”.



Figur 38 - Utvidet logging

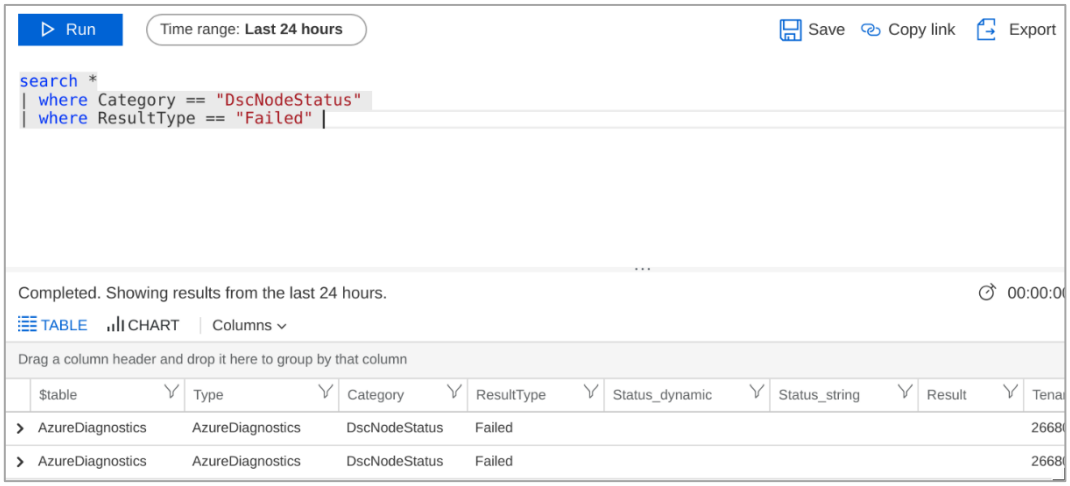
3. Velg “Logs” under “General” i menyen til venstre.
4. Velg “Add a credentials”.



Figur 39 - Utvidet logging

5. Skriv inn spørring som visst under.

Hvis det finnes noder med ResultType “Failed” vil dette dukke opp i søket. Ønsker man detaljer kan man trykka på pila helt til venstre for flere detaljer.



Figur 40 - Utvidet logging

3 Oppsett og konfigurasjon i testmiljø

Dette kapittelet tar for seg informasjon om det lokale testmiljøet. I tillegg gis det informasjon om nyttige PowerShell-kommandoer og feilsøkingsverktøy som er anbefalt å benytte seg av.

3.1 Testmiljø

Det lokale testmiljøet består av to virtuelle maskiner med operativsystemet Windows Server 2016. Maskinene er gjort tilgjengelig, installert og konfigurert av NTNUIT. Operativsystem ble valgt siden det er dette som hovedsakelig i dag benyttes til Session Host. I tillegg er det operativsystemspesifikke konfigurasjonsblokker for Windows Server 2016, slik at det kunne testes dynamisk konfigurasjon i henhold til operativsystem. Se kapittel 5.10 for mer informasjon.

Navnsetting på testmiljø skal følge NTNU IT sin navnestandard på terminalservermiljøet og vil få navn som visst i Tabell 3.

Hva	**Ønsket navn**
Session Collection	azdscfarm.it.ntnu.no
Server 1	it-azdscfarm01
Server 2	it-azdscfarm02

Tabell 21 – Testmiljø

3.2 Nyttige PowerShell-kommandoer i bruk lokalt

Det er en rekke PowerShell-kommandoer som egner seg godt når man skal administrere og drifte Azure DSC lokalt på maskinene.

3.2.1 Get-DscConfiguration

Denne kommandoen returnerer den aktive konfigurasjonen på noden. Kommandoen kjører og gir resultatene til DSC-ressursens metode "Get-TargetResource". Det vil si at man får tilbakemelding om alle egenskapene til DSC-ressursen.

```
Get-DscConfiguration
```

Kodeblokk 13 - Returnere den aktive konfigurasjonen

3.2.2 Test-DscConfiguration

Denne kommandoen returnerer om noden er i den ønskede tilstand eller ikke. Kommandoen kjører og gir resultatene til DSC-ressursens metode "Test-TargetResource". Den vil enten være "True" eller "False".

```
Test-DscConfiguration
```

Kodeblokk 14 - Teste om konfigurasjonen er i ønsket tilstand

3.2.3 Start-DscConfiguration

Denne kommandoen resetter konfigurasjonen på en node og får den tilbake i ønsket tilstand. Denne er relevant å bruke hvis noden er konfigurert med "ApplyAndMonitor" og den står med konfigurasjonsstatus "Not Compliant". Det vil si at noden ikke er i den ønskede tilstanden. Ved kjøring av kommandoen vil LCM rekonfigurere noden i henhold til den konfigurasjonen som står i "Staging = Current", altså nåværende konfigurasjon.

```
Start-DscConfiguration -UseExisting -Wait -Verbose
```

Kodeblokk 15 - Oppdatere konfigurasjon

3.2.4 Update-DscConfiguration

Denne kommandoen kontrollerer Pull-serveren for ny konfigurasjon og implementerer denne hvis det er en eventuell forskjell fra nodens lokale konfigurasjon og den på Pull-serveren. Denne er nyttig da man kan fremprovosere implementasjon av en ny konfigurasjon med en gang i stedet for å vente på at LCM skal oppdatere, noe som kan ta litt tid ettersom dette er en konfigurert tidsfrekvens.

For å få oversikt over hva kommandoen gjør og se at alt går som det skal er det lurt å gjøre kommandoen med flaggene "-Wait" og "-Verbose". Da vil resultatet av kommandoen bli presentert direkte i konsollet i stedet for en jobb, inklusive Verbose-strømmen.

```
Update-DscConfiguration -Wait -Verbose
```

Kodeblokk 16 - Oppdatere konfigurasjon

3.2.5 Remove-DscConfigurationDocument

Denne kommandoen sletter konfigurasjonsdokumentet (*.MOF) fra noden. Denne egner godt i testing og feilsøking da man raskt må nullstille noden før man henter ny konfigurasjon fra Pull-serveren med "Update-DscConfiguration" kommandoen.

```
Remove-DscConfigurationDocument -Stage Current
```

Kodeblokk 17 - Fjerne konfigurasjonsdokumentet fra noden

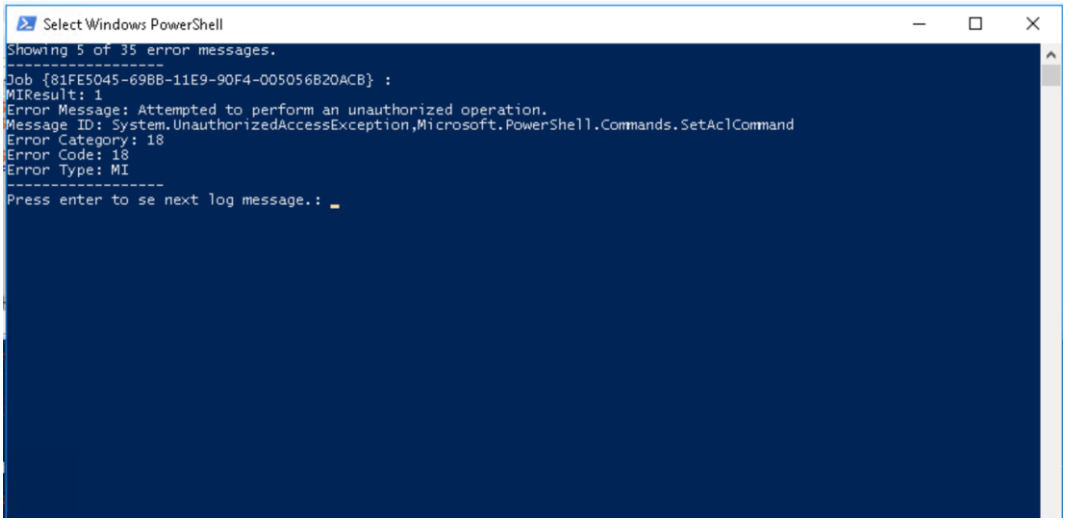
3.3 Feilsøking på node

I tillegg til PowerShell-kommandoene som beskrevet i forrige underkapittel er bruk av hendelsesloggen et godt verktøy for bruk i feilsøking. DSC skriver til tre logger som ligger under "Applications and Services Logs/Microsoft/Windows/Desired State Configuration". Det er bare én av tre som er påskrudd som standard [12.

- **Operational** som er påskrudd som standard. Den inneholder alle Error-meldinger som kan brukes til å identifisere et problem.
- **Analytic** inneholder et høyere antall hendelser og kan brukes til å identifisere hvor feilen skjedde. Denne loggen inneholder også "Verbose" meldinger.
- **Debug** inneholder hendelser som kan bistå en utvikler forstå hvordan feilen skjedde.

3.3.1 Hente ut hendelseslogg med PowerShell

Uthenting av hendelsesloggen med PowerShell er visst i figur 31. Her vises en hendelse for en DSC-ressurs som feiler på grunn av manglende rettigheter.



Figur 41 - Hente ut hendelseslogg med PowerShell

Scriptet som kjøres er visst i kodeblokk 15.

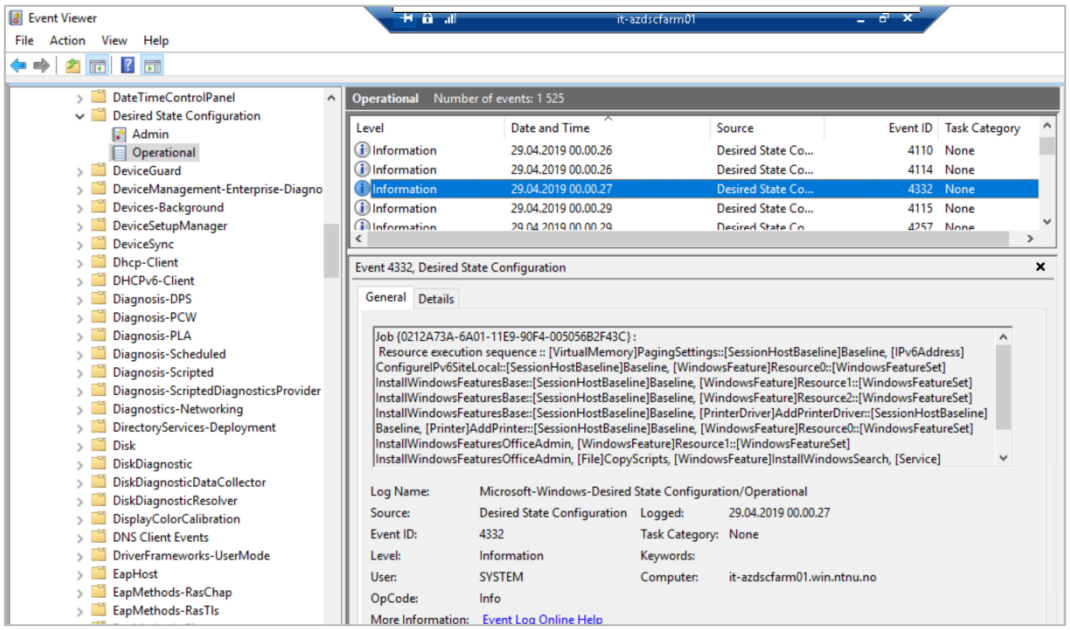
```
$eventlog = Get-WinEvent -LogName “Microsoft-Windows-Dsc/Operational”
$EventError = $eventlog | Where-Object LevelDisplayName -eq "Error"

$n = 1
foreach ($message in $EventError) {
    cls
    Write-Output "Showing $n of $($EventError.Count) error messages."
    Write-Output "------------------"
    Write-Output $message.Message
    Write-Output "------------------"
    Read-Host "Press enter to se next log message."
    $n = $n + 1
}
```

Kodeblokk 18 - Script til å hente ut hendelseslogg

3.3.2 Hente ut hendelseslogg med Event Viewer

Uthenting av hendelsesloggen med Event Viewer er visst i figur 16.

1. Logg på maskinen.
2. Start “Event Viewer”.
3. Naviger til “Applications and Services Logs/Microsoft/Windows/Desired State Configuration\Operational”



Kodeblokk 19 - Hendelseslogg med Event viewer

4 Egen- og videreutviklede DSC-ressurser

Det er inkludert en rekke ressurser med PowerShell som dekker mange behov, men ikke alle. Microsoft legger opp til at brukerne selv kan utvikle egne ressurser og dele de med andre [5]. Ressursene er Open Source og publiseres på for eksempel The PowerShell Gallery og kan importeres direkte inn i Azure Automation. Finner man ikke ressurser som dekker behovet kan man enten lage egendefinerte ressurser eller endre på eksisterende ressurser for å tilpasse dem til den problemstillingen man ønsker.

En DSC-ressurs er en del av en Windows PowerShell modul. Den består hovedsakelig av et skjema (MOF-fil), som definerer egenskapene til ressursen og implementeringen (psm1-fil), som er kildekoden som faktisk utfører det man ønsker å gjøre. Mappestrukturen for DSC-ressursen CopyAcl er vist på figur 32.

```
$env:ProgramFiles\WindowsPowerShell\Modules (folder)
    |- AzDSC_Modules (folder)
        |- DSCResources (folder)
            |- CopyAcl (folder)
                |- CopyAcl.psm1 (file, required)
                |- CopyAcl.schema.mof (file, required)
```

Figur 42 - Mappestruktur i DSC-ressurs

DSC-ressurser må bygges opp etter en fast oppsett og alle ressursene må ha følgende metoder tilgjengelig: "Get-TargetResource", "Test-TargetResource" og "Set-TargetResource".

- Get-metoden kontrollerer hvilken status egenskapene til ressursen har hos noden.
- Test-metoden kontrollerer om noden er i den konfigurerte ønskede tilstand.
- Set-metoden vil forsøke å konfigurere noden slik at den kommer i ressursens ønskede tilstand.

Forskjellen på Get og Test er at Get returnerer selve statusen for ressursen, mens Test returnerer en boolsk verdi, altså $true eller $false [6]. Alle metodene må ha hver egenskap som er definert i MOF-skjemaet er tilgjengelige parametere.

4.1 xDscResourceDesigner

For å utvikle egne ressurser bør det benyttes Resource Designer Tool. Dette er et verktøy setter opp rammene for script-fila, det vil si MOF-skjemaet, modulen, og mappestrukturen. Dette gjør at man heller kan fokusere på utviklingen av PowerShell kode i stedet for å tenke på detaljene rundt ressursen [7].

Bruk av xDSCResourceDesigner vises i kapittel 7.4.

4.2 Ressurser

Det er utviklet fire DSC-ressurser i prosjektperioden som er med på å løse oppdragsgivers behov for å automatisk konfigurere en Session Host. Disse DSC-ressursene ble utviklet da eksisterende ressurser ikke løste problemstillingene tilstrekkelig. Det ble vurdert å benytte

Script-ressursen i noen tilfeller, men for å skape en mer oversiktlig konfigurasjonsfil ble det i stedet utviklet DSC-ressurser. Dette gjør også at det i fremtiden vil være enklere å bruke og vedlikeholde DSC-ressursene i andre konfigurasjoner. Kildekoden ligger ett sted og man trenger kun inkludere modulen og skrive inn konfigurasjonsblokken for å benytte seg av den. Skal det gjøres endringer eller oppdateringer på DSC-ressursen gjøres dette i modulen og endringene blir distribuert ut til alle konfigurasjonene. DSC-ressursene er laget så enkle som mulig for å optimalisere for gjenbruk i alternative problemstillinger.

DSC-ressursene er hovedsakelig videreutviklet av tidligere PowerShell-script og / eller andre ressurser som manglet spesifikk funksjonalitet. Der det er gjort er det spesifisert og kreditert i dokumentasjonen og i kildekoden. Microsoft har lansert alle DSC-ressursene som åpen kildekode og anbefaler brukere om å bidra med å utvikle ressurser. [13]

Dokumentasjonens utforming er inspirert av PowerShell sin dokumentasjon. Dette for at det skal være enkelt og oversiktlig og sette seg inn i hvordan de fungerer og hva som ble tenkt da ressursene ble utviklet. Dette kommer godt med hvis man for eksempel skal gjøre endringer, feilsøke eller bygge videre på ressursene i fremtiden.

Dokumentasjonen starter først med navnet på ressursen og en kort beskrivelse av hva DSC-ressursen gjør. Så vises syntaksen for hvordan ressursen skal defineres i konfigurasjonsblokken. Videre gis det en utvidet forklaring på hva DSC-ressursen gjør, overordnet hvordan den fungerer og informasjon om ressursen er videreutviklet eller bygget på andre sitt arbeid. Til slutt presenteres ressursen egenskaper med forklaring før det vises et eksempel på hvordan man benytter DSC-ressursen.

4.2.1 CopyAcl

Kopierer Access Control List fra et objekt til et annet

Syntaks

```
CopyAcl [string] #ResourceName
{
    SourcePath = [string]
    DestinationPath = [string]
    Recurse = [bool]
}
```

Kodeblokk 20 - Syntaks CopyAcl

Forklaring

CopyAcl er en videreutvikling av scriptet som tetter det potensielle sikkerhetshullet på Windows Server 2016. Scriptets kode er forenklet og skrevet om til en DSC-ressurs for å få en oversiktlig og lettlest konfigurasjon.

DSC-ressursen er såpass forenklet at man kun kan definere én kopiering per konfigurasjon. Det vil si at for at DSC-ressursen skal gjøre det samme som scriptet hentet fra dokumentasjonen til NTNUIT må man ha to konfigurasjonsblokker som spesifiserer de forskjellige mappene. Dette er visst nærmere i kapittel 5 konfigurasjon. Dette valget ble tatt for å gjøre DSC-ressursen mer gjenbrukbar og entydig slik at man i fremtiden kan benytte den til andre problemstillinger.

Ønsker man å forenkle dette ytterligere er det mulig å lage en Composite Resource[7] lik WindowsFeatureSet der man definerer flere DestinationPath per konfigurasjon.

Egenskaper

- **SourcePath** spesifiserer hvor Access Controll List skal kopieres fra
- **DestinationPath** spesifiserer hvor Access Controll List skal kopieres til.
- **Recurse** spesifiserer om Access Controll List skal gjelde rekursivt for alt innhold nedover i mappehierarkiet.

Eksempel

```
CopyAcl CopyAclUsersDefault
{
    SourcePath              = "c:\users"
    DestinationPath         = "c:\users\default\"
    Recurse                 = $false
}
```

Kodeblokk 21 - Eksempel CopyAcl

[7] En DSC-ressurs som består av én eller flere DSC-ressurser. Dette er nærmere forklart i kapittel 5.5.

4.2.2 IPv6Address

Konfigurerer en IPv6-adresse

Syntaks

```
IPv6Address [string] #ResourceName
{
    NetIPv6Address = [string]
}
```

Kodeblokk 22 - Syntaks IPv6Address

Forklaring

Valget med å utvikle en egendefinert ressurs for å konfigurere IPv6-adresse ble tatt siden de eksisterende ressursene ikke løste det fulle og hele behovet til NTNUIT. IPv6-adressen skal bestå av en fast forhåndsdefinert nett-del og en dynamisk node-del. Node-delen er IPv4-adressens konvertert heksadesimalt. Det ble forsøkt å bruke Script-DSC-ressursen i konfigurasjonen, men dette ble gått bort ifra da det skapte en stor og uoversiktlig konfigurasjonsfil.

DSC-ressursen er en videreutviklet og omskrevet versjon av scriptet NTNUIT benytter i dag. I stedet for å kun konvertere siste del av IPv4-adressen automatisk, vil scriptet konvertere hele IPv4 adressen. Dette gjør scriptet mer dynamisk i tilfelle man skal gjøre større endringer på IPv4-adressen i fremtiden. Noe kode er lånt fra NetworkingDsc[8].

Egenskaper

- **NetIPv6Address** spesifiserer nettverksdelen av en IPv6-adresse

Eksempel

```
IPv6Address ConfigureIPv6SiteLocal
{
    NetIPv6Address          = "fdd2:2868:bf02:7777::"
}
```

Kodeblokk 23 - Eksempel IPv6Address

[8] https://github.com/PowerShell/NetworkingDsc

4.2.3 PrinterDriver

Installer og konfigurerer en printerdriver.

Syntaks

```
PrinterDriver [string] #ResourceName
{
    DriverName = [string]
    [InfPath = [string]]
    [TPInfPath = [string]]
    [Environment = [string]] { x86 | x64 }
    [Ensure = [string]] { Present | Absent }
}
```

Kodeblokk 24 - Syntaks PrinterDriver

Forklaring

DSC-ressursen er en videreutvikling av VirtualEngine sin DSC-ressurs, VE_PrinterDriver[9]. Den utvidede funksjonalisten er TPInfPath, som står for "Third Party Inf Path". Denne egenskapen tar imot en sti til en tredjeparts skriverdriver, den henter driveren og installerer den.

Det fantes ingen gode og overstigelige DSC-ressurser med mulighet til å legge inn tredjeparts drivere, som det er behov for i problemstillingen til NTNUIT. Denne funksjonaliteten måtte da utvikles. Noe kode er lånt og skrevet om fra Kevin Marquettes, DSC-ressurs KevMar_TcpPrinter[10] som har lignende funksjonalitet.

Egenskaper

- **DriverName** spesfiserer printernavnet.
- **InfPath** spesifiserer stien til printer driver INF-filen i Windows Driver Store.
- **TPInfPath** spesifiserer stien til tredjeparts driver INF-fil. Denne kan være lokalisert lokalt eller på et nettverksområde.
- **Enviroment** Spesifiserer drivermiljøet driveren skal installeres på.
    - Mulige verdier er "x64" og "x86". Standardvalget er "x64".
- **Ensure** sikrer om driveren er installert eller ikke.

Eksempel

```
PrinterDriver AddPrinterDriver
{
    DriverName     = "Xerox Global Print Driver PS"
    Ensure         = "Present"
    TPInfPath      = "\\win.ntnu.no\share\...\Xerox\x2UNIVP.inf"
}
```

Kodeblokk 25 - Eksempel PrinterDriver

[9] https://github.com/VirtualEngine/PrinterManagement
[10] https://github.com/KevinMarquette/PowershellWorkspace

4.2.4 SesionHost

Installerer og konfigurerer en Session Host.

Syntaks

```
SessionHost [string] #ResourceName
{
    ConnectionBroker = [string]
    Collection = [string]
    NewConnectionAllowed = [string] { yes | no }
}
```

Kodeblokk 26 - Syntaks SessionHost

Forklaring

DSC-ressursen er en videreutvikling og forenkling av Microsoft sin DSC-Ressurs, MSFT_xRDServer som er inkludert i modulen xRemoteDesktopSessionHost[11]. Valget med å lage en ny ressurs ble tatt fordi MSFT_xRDServer er laget for installasjon og konfigurasjon av en fullstendig Remote Desktop Services-løsning. I tillegg var det behov for å definere FQDN for hver Session Host i konfigurasjonsfilen, og jeg forsøkte å unngå denne dobbeltregistreringen av informasjon siden denne informasjonen er allerede tilgjengelig i Azure Automation.

DSC-ressursen har ikke blitt testet på testmiljøet fordi dagens løsning blokkerer WinRM trafikk mellom Session Host og Connection broker. Denne DSC-ressursen kan ses på som administrasjon og ikke installasjon da PowerShell-kommandoer i ressursen ikke kjører mot seg selv, men mot en annen maskin, Connection Broker.

Egenskaper

- **ConnectionBroker** spesifiserer FQDN for Connection Broker i RDS installasjonen.
- **Collection** spesifiserer Session Collection
- **NewConnectionAllowed** spesifiserer om Session Hosten skal akseptere nye tilkoblinger.
- **PSDscRunAsCredential** spesifiserer hvilken legitimasjon DSC-ressursen skal kjøre under.

Eksempel

```
SessionHost Test
{
    ConnectionBroker            = "it-rdconnect01.win.ntnu.no"
    Collection                  = "Azdscfarm"
    NewConnectionAllowed        = "No"
    PsDscRunAsCredential        = $Cred
}
```

Kodeblokk 27 - Eksempel SessionHost

[11] https://github.com/PowerShell/xRemoteDesktopSessionHost

5 Konfigurasjon

Dette kapittelet tar for seg oppbyggingen av konfigurasjonsfila som er utviklet i prosjektperioden. Den går igjennom alle de forskjellige momentene og gir en forklaring på hva, hvor og hvordan. Kapittelet er bygget opp slik at det går igjennom konfigurasjonsfila fra start til slutt. Konfigurasjonen består av bruk av moduler, bruk av legitimasjon, håndtering av avhengigheter, dynamisk konfigurasjon og de forskjellige konfigurasjonsblokkene.

5.1 Bruk av moduler

De DSC-ressursene som skal benyttes i konfigurasjonen må importeres. Disse kommer i moduler med én eller flere DSC-ressurser. Tabell 4 viser hvilke moduler konfigurasjonen benytter. Disse må importeres i Azure Automation som visst i kapittel 2.5.4.

Modulnavn	**DSC-ressurs**	**Forklaring**
ComputerManagementDsc	VirtualMemory	Konfigurere paging-fil
AzureDSC_Modules	IPv6Address	Konfigurer IPv6-adresse
	PrinterDriver	Installere Printerdriver
	Printer	Installere Printer
	SessionHost	Installere og konfigurere Session Host
	CopyAcl	Kopiere Access Controll List
PSDesiredStateConfiguration	WindowsFeatureSet	Installere et sett av WindowsFeatures
	WindowsFeature	Installere én WindowsFeature
	Service	Konfigurere / styre en tjeneste
-	File	Kopiere fil / mappe

Tabell 22 - Brukte DSC-moduler

For å benytte en DSC-ressurs i konfigurasjonen må de importeres med bruk av PowerShell-kommandoen "Import-DscResource" som visst på kodeblokk 25 og 26. Alle ressurser som ligger lagret under $PSHOME[12] på maskinen vil bli importert automatisk, men det er fortsatt beste praksis å importere ressursene uansett slik at det er visst hvilke ressurser som benyttes. Dette sikrer også at man benytter seg av korrekt DSC-ressurs i tilfelle det finnes flere ressurser med samme navn på systemet.

Når man importerer DSC-ressurser er det beste praksis å definere både navn og fra hvilken modul DSC-ressursen kommer fra. Dette reduserer søkeomfanget og sikrer at man importerer rett DSC-ressurs. Hvis man kun benytter Name-parametere vil maskinen bruke tid og ressurser på å søke etter en DSC-ressurs med det spesifiserte navnet. Jo flere installerte moduler på maskinen, jo mer tid og ressurser vil søket kreve. Det er også mulig å importere feil DSC-ressurs hvis det finnes flere DSC-ressurser med samme navn. PowerShell vil importere den første ressursen den finner i søket.

[12] C:\WINDOWS\System32\WindowsPowerShell\v1.0\

Det er også mulig å kun definere modulnavn slik at man importerer hele modulen. Dette vil fungere, men da importerer man en hel del ressurser man ikke har behov for som skaper unødvendig mye overhead. I tillegg viser man heller ikke hvilke DSC-ressurser som er i bruk i de forskjellige modulene.

I konfigurasjonsfila er det anbefalt at hver importering legges på egen linje. Det er mulig å importere flere DSC-ressurser på én line, men dette gjør det uoversiktlig for leseren og hvis man benytter et versjonskontrollsystem for håndtering av endringer er det lettere å se endringer som utføres på hele linjer i stedet for endringer på en linje. Dette gjør endringsprosessen sikrere siden det er enkelt å se i historikken hva som er endret [14].

5.1.1 Import av DSC-ressurser i Composite Resource

DSC-ressurser som er brukt i kompositttressurs SessionHostBaseLine er visst i kodeblokk 25.

```
Import-DscResource -ModuleName ComputerManagementDsc -name
VirtualMemory
Import-DscResource -ModuleName PSDesiredStateConfiguration -name
WindowsFeatureSet
Import-DscResource -ModuleName AzureDSC_Modules -name IPv6Address
Import-DscResource -ModuleName AzureDSC_Modules -name PrinterDriver
Import-DscResource -ModuleName AzureDSC_Modules -name Printer
#Import-DscResource -ModuleName AzureDSC_Modules -name SessionHost
```

Kodeblokk 28 - Inkluderte DSC-ressurser i kompositttressurs

5.1.2 Import av DSC-ressurser i konfigurasjon

DSC-ressurser som er brukt i konfigurasjonen er visst i kodeblokk 26.

```
Import-DscResource -ModuleName SessionHostComposite -name
SessionHostBaseline
Import-DscResource -ModuleName PSDesiredStateConfiguration -name
WindowsFeatureSet
Import-DscResource -ModuleName PSDesiredStateConfiguration -name
WindowsFeature
Import-DscResource -ModuleName PSDesiredStateConfiguration -name
Service
Import-DscResource -ModuleName AzureDSC_Modules -name CopyAcl
```

Kodeblokk 29 - Inkluderte DSC-ressurser i konfigurasjon

5.2 Legitimasjon

Noen ganger er det behov å kjøre DSC-konfigurasjonen som andre brukerkontoer enn systemkontoen "NT AUTHORITY\SYSTEM". Dette kan skyldes at det er oppgaver systemkontoen ikke har tilgang til å utføre. Som visst i kapittel 2.2.4 kan legitimasjonen lagres i Azure Automation. For å benytte disse må man benytte PowerShell-kommando i konfigurasjonen som visst i kodeblokk 27. Legitimasjonen lagres som et PSCredential-objekt som senere benyttes i konfigurasjonen ved å kalle egenskapen "PsDscRunAsCredential = $Cred" i definisjonen av DSC-ressursen. Dette er en automatisk egenskap som er tilgjengelig for alle DSC-ressurser.

Navnet "AccessUser" referer til "Name" for legitimasjonen som er opprettet i Azure Automation.

```
$Cred = Get-AutomationPSCredential "AccessUser"
```

Kodeblokk 30 - Legitimasjon

Ved å bruke dot-notasjon er det for også for eksempel mulig å opprette lokale brukerkontoer med den legitimasjonen man har lagt inn i Azure Automation. Dette er visst i kodeblokk 28. Denne konfigurasjonen vil opprette en bruker med brukernavn og passord lik det som er spesifisert under Credentials for "AccessUser" i Azure Automation.

```
User InstallUser
{
    Ensure              = "Present"
    UserName            = $Cred.UserName
    Password            = $Cred
    Description         = "Test"
}
```

Kodeblokk 31 - Eksempel på bruk av legitimasjon

5.3 Håndtering av avhengigheter

Konfigurasjonen kan etter hvert bli stor og vanskelig å håndtere. Noen ganger kan det oppstå avhengigheter mellom ulike konfigurasjonsblokker. For å håndtere dette er det mulig å benytte egenskapen "DependsOn" som er tilgjengelig for alle DSC-ressurser. Et eksempel på en avhengighet i konfigurasjonen er installasjon av Windows Search Service og styring av tjenesten beskrevet i kapittel 5.9. Her skal først Windows Search Service installeres før tjenesten videre skal deaktiveres. Hadde Azure DSC først forsøkt å deaktivere tjenesten før den var installert, hadde konfigurasjonen feilet [15].

DependsOn vil også sikre at resten av avhengighetene ikke blir forsøkt utført hvis DSC-ressursen den er avhengig av feiler. På denne måten utelukker man følgefeil i konfigurasjonen [15].

5.4 Dynamisk konfigurasjon

For å gjøre konfigurasjonen mer fleksibel og mer dynamisk er det lagt inn parameter og kontrollstrukturer som gir forskjellig konfigurasjon etter hva slags inndata som gis. Dette gjør at man kan lage flere forskjellige kompilerte konfigurasjoner (MOF-filer) som kan tildeles nodene ut ifra samme konfigurasjon. Konfigurasjonen har parametere som visst på kodeblokk 29 som viser de tilgjengelige parameterne.

- Parametere OS definerer hvilket operativsystem den kompilerte konfigurasjonen skal gjelde for.
- Parametere Collection definerer hvilken Session Collection den kompilerte konfigurasjonen skal gjelde for.

```
param(
    [Parameter(Mandatory=$true)]
    [ValidateSet("srv2019", "srv2016", "srv2012r2")]
    [string] $OS,

    [Parameter(Mandatory=$true)]
    [ValidateSet("adminfarm", "officefarm", "calcfarm", "Azdscfarm")]
    [string] $Collection
)
```

Kodeblokk 32 - Parametere i konfigurasjon

De forskjellige betingelsene i konfigurasjonsfila er visst i kodeblokk 30. En konfigurasjon kan få konfigurasjon fra flere av disse betingelsene. Som man ser er "Officefarm" definert i to forskjellige betingelser.

```
# Hvis $Collection er satt til officefarm eller adminfarm
if ($Collection -eq "officefarm" -or "adminfarm")
{...}

# Hvis $Collection er satt til calcfarm
if ($Collection -eq "calcfarm")
{...}

# Hvis $Collection er satt til officefarm eller adminfarm eller
azdscfarm
if ($Collection -eq "officefarm" -or "adminfarm" -or "azdscfarm")
{...}

# Hvis $OS er satt til "srv2016"
if ($OS -eq "srv2016")
{...}
```

Kodeblokk 33 - Betingelser i konfigurasjon

5.4.1 Navnestandard

Den kompilerte konfigurasjonen vil få navn etter parameterne "$Collection.$OS". Kompilerer man for eksempel konfigurasjonen med parameterne OS satt til "srv2016" og Collection til

"Calcfarm" vil navnet på konfigurasjonen bli "Calcfarm.srv2016". Konfigurasjonen vil få den konfigurasjonen som er definert under Calcfarm og de operativspesifikke konfigurasjonsblokkene for Windows Server 2016.

5.5 Composite Resource - Baselinekonfigurasjon

Den konfigurasjonen som gjelder for alle Session Hostene uavhengig av hvilken Session Collection, Session Hosten er medlem av, er tatt ut av konfigurasjonsfila og satt inn i en Composite Resource (kompositttressurs) kalt SessionHostBaseline. Dette blir gjort for å forenkle konfigurasjonsfila og vise konseptet ved å bruke kompositttressurrser i Azure DSC.

Baselinekonfigurasjonen refereres så til i konfigurasjonen som visst i kodeblokk 31.

```
SessionHostBaseline Baseline
{
        # Baseline configuration from Module SessionHostComposite
}
```

Kodeblokk 34 - Eksempel på bruk av kompositttressurs SessionHostBaseline

En kompositttressurs er en DSC-ressurs bestående av en annen konfigurasjon. Det lages en PowerShell-modul med én eller flere ressurser. Den består av en konfigurasjonsfil og et PowerShell-modul-manifeste. Kompositttressursen bør benyttes når en node skal konfigureres av flere forskjellige konfigurasjoner eller hvis man har en konfigurasjon som skal brukes på flere forskjellige roller. Kompositttressursen består av fil- og mappestrukturen visst på figur 33. Det er hovedsakelig to filer vi må konfigurere:

- **CompositeResource.psd1** som er en PowerShell-modul-manifeste.
- **CompositeResource.schema.mof** som består av DSC-konfigurasjonen.

```
$env:ProgramFiles\WindowsPowerShell\Modules (folder)
    |- AzDSC_Modules (folder)
        |- DSCResources (folder)
            |- SessionHostBaseline (folder)
                |- SessionHostBaseline.psd1 (file, required)
                |- SessionHostBaseline.schema.mof (file, required)
```

Figur 43 - Mappestruktur i kompositttressurs

Ved å gjøre det på denne måten vil man håndtere konfigurasjonen ett sted og ikke en gang for hver forskjellig konfigurasjon man har. Dette sikrer for eksempel en endringsprosess mot vanvare og forglemmelse.

Andre bruksområder enn baseline konfigurasjon kan være hvis man har veldig avanserte konfigurasjonsblokker. Det kan da lages standardkonfigurasjon for de forskjellige tjenestene man har behov for som så refereres inn i konfigurasjonene til de nodene som har behov for en viss type funksjonalitet. Da vil man i stedet for å bygge opp en ny konfigurasjon hver gang kun

benytte allerede eksisterende konfigurasjon. Dette kan for eksempel være brannmurregler eller et bestemt oppsett av Web-servere.

Baseline konfigurasjonen består av DSC-Ressursene VirtualMemory, IPv6Address, WindowsFeatureSet, PrinterDriver, Printer og Session Host. Dette blir videre forklart nærmere.

5.6 Konfigurasjonsblokker

Dokumentasjonens utforming er inspirert av PowerShell sin dokumentasjon. Dette for at det skal være enkelt og oversiktlig og sette seg inn i hvordan noden er konfigurert, hvordan konfigurasjonen fungerer og hva som ble tenkt da valg av DSC-ressurs ble gjort. Hver konfigurasjonsblokk blir tatt hver for seg. Eventuelle avhengiheter blir spesifisert.

Dokumentasjonen starter først med navnet på konfigurasjonen, navnet på konfigurasjonsblokka og en kort beskrivelse av hva konfigurasjonen gjør. Så vises syntaksen for hvordan ressursen skal defineres i konfigurasjonsblokken. Videre gis det en utvidet forklaring på hva konfigurasjonen gjør, hvor den er hentet fra og om den må lastes inn i Azure Automation. Til slutt presenteres ressursen egenskaper med forklaring før selve konfigurasjonsblokka presenteres slik oppgaven er løst.

5.6.1 VirtualMemory PagingSettings

Konfigurerer virtuelt minne.

Syntaks

```
VirtualMemory [String] #ResourceName
{
    Drive = [string]
    Type = [string]{ AutoManagePagingFile | CustomSize | NoPagingFile
| SystemManagedSize }
    [InitialSize = [Int64]]
    [MaximumSize = [Int64]]
}
```

Kodeblokk 35 - Syntaks VirtualMemory

Forklaring

Konfigurasjonen benytter DSC-ressursen VirtualMemory som er inkludert i modulen ComputerManagementDsc[13]. Den benyttes for å tilfredsstille behovet for å konfigurere systemets paging-fil. Paging-filen benyttes til å utvide maskinens fysiske minne. Ressursen er utviklet av Microsoft og er tilgjengelig for direkte distribusjon inn i Azure Automation fra PowerShell Gallery.

Egenskaper

- **Type** defineres etter hva slags behov man har. Man har 4 valgmuligheter:
  - AutoManagePagingFile bestemmer om man skal la operativsystemet lagre paging-fil til alle drev. Hvis dette alternativet velges vil Drive parametere bli ignorert.
  - CustomSize gir muligheten til å definere egendefinerte innstillinger for størrelsen på paging-filen.
  - SystemManagedSize lar operativsystemet selv avgjøre paging-filens størrelse.
  - NoPagingFile slår av muligheten for virtuelt minne.
- **Drive** definerer hvilket drev som skal konfigureres.
- **InitialSize** bestemmer paging-filens minimumsstørrelse i megabyte
- **MaximumSize** bestemmer paging-filens maksimumstørrelse i megabyte.

Konfigurasjon

```
VirtualMemory PagingSettings
{
    Type            = "CustomSize"
    Drive           = "C"
    InitialSize     = "10000"
    MaximumSize     = "20000"
}
```

Kodeblokk 36 - Konfigurasjon VirtualMemory

[13] https://github.com/PowerShell/ComputerManagementDsc

5.6.2 IPv6Address ConfigureIPv6SiteLocal

Konfigurerer en IPv6-site-local-adresse basert på spesifisert nettverksdel og IPv4-adressen konvertert til heksadesimal.

Syntaks

```
IPv6Address [string] #ResourceName
{
    NetIPv6Address = [string]
}
```

Kodeblokk 37 - Syntaks IPv6Address

Forklaring

Konfigurasjonen benytter DSC-ressursen IPv6Address som er inkludert i modulen AzureDSC_Modules. Denne ressursen benyttes til å konfigurere en IPv6-Site-Local-adresse som behøves for at Direct Access skal fungere på Session Hosts. Ressursen er utviklet i forbindelse med bachelorprosjektet og er må lastes opp manuelt i Azure Automation.

IPv6-adressen som konfigureres er basert på den spesifiserte adressen i parametere og IPv4-adressen konvertert til heksadesimal.

Egenskaper

- **NetIPv6Address** spesifiserer nettverksdelen av en IPv6-adresse.

Konfigurasjon

```
IPv6Address ConfigureIPv6SiteLocal
{
    NetIPv6Address = "fdd2:2868:bf02:7777::"
}
```

Kodeblokk 38 - Konfigurasjon IPv6Address

5.6.3 WindowsFeatureSet InstallWindowsFeaturesBase

Installerer ett sett med Windows Features.

Syntaks

```
WindowsFeatureSet [string] #ResourceName
{
    Name = [string[]]
    [Ensure = [string] { Absent | Present }]
    [Source = [string]]
    [IncludeAllSubFeature = [bool]]
    [Credential = [PSCredential]]
    [LogPath = [string]]

}
```

Kodeblokk 39 - Syntaks WindowsFeatureSet

Forklaring

Konfigurasjonen benytter DSC-ressursen WindowsFeatureSet som er inkludert i modulen PSDesiredStateConfiguration. Det er en kompositttressurs som kjører DSC-ressursen WindowsFeature for hvert enkelte spesifisert Windows-funksjon. Det gir en mer oversiktlig konfigurasjon siden alle funksjoner som skal være tilstede blir definert ett sted i stedet for en gang per funksjon. Dette vil spare mange linjer i konfigurasjonen som vil lette administrasjon og feilsøking. DSC-Ressursen eksisterer allerede i Azure Automation og må ikke inkluderes.

Konfigurasjonen sikrer at rollene “RDS-RD-Server", “XPS-Viewer” og “NET-Framework-Core" er tilstede.

Egenskaper

- **Name** spesifiserer navnet på rollene og funksjonene som skal håndteres av konfigurasjonen.
- **Ensure** bestemmer om rollen eller funksjonene skal være tilstede eller ikke.

Konfigurasjon

```
WindowsFeatureSet InstallWindowsFeaturesBase
{
    Name      = @("RDS-RD-Server", "XPS-Viewer", "NET-Framework-Core")
    Ensure    = "Present"
}
```

Kodeblokk 40 - Konfigurasjon WindowsFeatureSet

5.6.4 PrinterDriver AddPrinterDriver

Installerer tredjeparts skriverdriver.

Syntaks

```
PrinterDriver [string] #ResourceName
{
    DriverName = [string]
    [InfPath = [string]]
    [TPInfPath = [string]]
    [Environment = [string]] { x86 | x64 }
    [Ensure = [string]] { Present | Absent }
}
```

Kodeblokk 41 - Syntaks PrinterDriver

Forklaring

Konfigurasjonen benytter DSC-ressursen PrinterDriver som er inkludert i modulen AzureDSC_Modules. Denne ressursen benyttes til å installere en printerdriver. Ressursen er videreutviklet fra den eksisterende DSC-ressursen PrinterManagement[14]. DSC-ressursen må lastes opp manuelt i Azure Automation.

Egenskaper

- **DriverName** spesifiserer navnet til printerdriveren.
- **Ensure** spesifiserer om printerdriveren skal være tilstede eller ikke.
- **TPInfPath** spesifiserer stien til tredjeparts drivere som skal bli installert inn i Windows sitt driverlager.

Konfigurasjon

```
PrinterDriver AddPrinterDriver
{
    DriverName      = "Xerox Global Print Driver PS"
    Ensure          = "Present"
    TPInfPath       = "\\share\printdriver.inf"
}
```

Kodeblokk 42 - Konfigurasjon PrinterDriver

[14] https://github.com/VirtualEngine/PrinterManagement

5.6.5 Printer AddPrinter

Installerer skriver.

Syntaks

```
Printer [String] #ResourceName
{
    DriverName = [string]
    Name = [string]
    PortName = [string]
    [Comment = [string]]
    [Ensure = [string]{ Absent | Present }]
    [Location = [string]]
    [Published = [bool]]
    [ShareName = [string]]
}
```

Kodeblokk 43 - Syntaks Printer

Forklaring

Konfigurasjonen benytter DSC-ressursen Printer som er inkludert i modulen PrinterManagement. Denne ressursen benyttes til å installere en skriver. Ressursen er utviklet av VirtualEngine[15] og er lastet ned og inkludert i modulen AzureDSC_Modules og må lastes opp manuelt i Azure Automation.

Egenskaper

- **Name** spesifiserer navnet på skriveren.
- **DriverName** spesifiserer hvilken driver skriveren skal benytte.
- **PortName** spesifiserer hvilken skriverport skriveren skal benytte.
- **Ensure** spesifiserer om skriveren skal være tilstede eller ikke.
- **DependsOn** spesifiserer en avhengighet til DSC-ressursen PrinterDriver. Det vil si at skriveren ikke vil bli installert før skriverdriveren er på plass.

Konfigurasjon

```
Printer AddPrinter
{
    Name            = "Xerox Global Print Driver PS"
    DriverName      = "Xerox Global Print Driver PS"
    PortName        = "portprompt:"
    Ensure          = "Present"
    DependsOn       = "[PrinterDriver]AddPrinterDriver"
}
```

Kodeblokk 44 - Konfigurasjon Printer

[15] https://github.com/VirtualEngine/PrinterManagement

5.6.6 SessionHost MakeSessionHost

Installerer og konfigurerer en Session Host.

Syntaks

```
SessionHost [string] #ResourceName
{
    ConnectionBroker = [string]
    Collection = [string]
    NewConnectionAllowed = [string] { yes | no }
}
```

Kodeblokk 45 - Syntaks SessionHost

Forklaring

Konfigurasjonen benytter DSC-ressursen SessionHost som er inkludert i modulen AzureDSC_Modules. Denne ressursen benyttes til å konfigurere maskinen til en Session Host og legge den inn i en Session Collection. Ressursen er utviklet i forbindelse med bachelorprosjektet og er må lastes opp manuelt i Azure Automation.

Egenskaper

- **ConnectionBroker** spesifiserer full-domenenavn for Connection Broker.
- **Collection** spesifiserer hvilket Session Collection Session Hosten skal meldes inn i.
- **NewConnectionAllowed** spesifiserer om Session Hosten skal akseptere nye tilkoblinger.

Konfigurasjon

```
SessionHost MakeSessionHost
{
    ConnectionBroker     = "it-rdconnect01.win.ntnu.no"
    Collection           = "Azdscfarm"
    NewConnectionAllowed = "No"
}
```

Kodeblokk 46 - Konfigurasjon SessionHost

5.7 Officefarm og Adminfarm

For maskiner i Session Collection Officefarm og Adminfarm gjelder konfigurasjon som forklart i dette underkapittelet.

5.7.1 WindowsFeatureSet InstallWindowsFeatureOfficeAdmin

Installerer ett sett med Windows Features.

Syntaks

```
WindowsFeatureSet [string] #ResourceName
{
    Name = [string[]]
    [Ensure = [string] { Absent | Present }]
    [Source = [string]]
    [IncludeAllSubFeature = [bool]]
    [Credential = [PSCredential]]
    [LogPath = [string]]

}
```

Kodeblokk 47 - Syntaks WindowsFeatureSet

Forklaring

Konfigurasjonen benytter DSC-ressursen WindowsFeatureSet som er inkludert i modulen PSDesiredStateConfiguration. Det er en kompositttressurs som kjører DSC-ressursen WindowsFeature for hvert enkelte spesifisert Windows-funksjon. Det gir en mer oversiktlig konfigurasjon siden alle funksjoner som skal være tilstede blir definert ett sted i stedet for en gang per funksjon. Dette vil spare mange linjer i konfigurasjonen som vil lette administrasjon og feilsøking.

Konfigurasjonen sikrer at rollene “Windows-Identity-Foudation" og “RSAT-AD-Powershell" er tilstede.

Egenskaper

- **Name** spesifiserer navnet på rollene og funksjonene som skal håndteres av konfigurasjonen.
- **Ensure** bestemmer om rollen eller funksjonene skal være tilstede eller ikke.

Konfigurasjon

```
WindowsFeatureSet InstallWindowsFeaturesOfficeAdmin
{
    Name      = @("Windows-Identity-Foundation", "RSAT-AD-Powershell")
    Ensure    = "Present"
}
```

Kodeblokk 48 - Konfigurasjon WindowsFeatureSet

5.8 Calcfarm

Maskiner som ligger i Session Collection calcfarm har én spesifikk konfigurasjon.

5.8.1 File CopyScripts

Kopiere en mappe fra et nettverksområde til c:\scritpts

Syntaks

```
File [String] #ResourceName
{
    DestinationPath = [string]
    [Attributes = [string[]]{ Archive | Hidden | ReadOnly | System }]
    [Checksum = [string]{ CreatedDate | ModifiedDate | SHA-1 | SHA-256
| SHA-512 }]
    [Contents = [string]]
    [Credential = [PSCredential]]
    [Ensure = [string]{ Absent | Present }]
    [Force = [bool]]
    [MatchSource = [bool]]
    [PsDscRunAsCredential = [PSCredential]]
    [Recurse = [bool]]
    [SourcePath = [string]]
    [Type = [string]{ Directory | File }]
}
```

Kodeblokk 49 - Syntaks File

Forklaring

Konfigurasjonen benytter DSC-ressursen File som er inkludert i Azure Automation. Den benyttes til å kopiere en mappe med scripts fra et nettverksområde til lokal lagring. Definerer man en mappe som ikke finnes vil mappen opprettet. DSC kjører med systemkontoen "NT AUTHORITY\SYSTEM" så det er viktig at denne brukeren har tilgang til det eksterne nettverksområdet. Eventuelt er det mulig å lagre legitimasjon til en brukerkonto i Azure Automation.

Egenskaper

- **Ensure** bestemmer om mappen skal være tilstede eller ikke.
- **Type** spesifiserer om det er en mappe eller en fil som skal kopieres
- **DestinationPath** spesifiserer destinasjonen for hvor mappen skal kopieres
- **SourcePath** spesifiserer hvor mappen skal kopieres fra.

Konfigurasjon

```
File CopyScripts
{
    Ensure          = "Present"
    Type            = "Directory"
    DestinationPath = "C:\scripts"
    SourcePath      = "\\win.ntnu.no\shares\...\Scripts"
}
```

Kodeblokk 50 - Konfigurasjon File

5.9 Øvrige farms

Maskiner som ligger i Session Collection Adminfarm, Officefarm, Kavlifarm, Edgefarm, Ielfarm to spesifikke konfigurasjonsblokker.

5.9.1 WindowsFeature InstallWindowsSearch

Installere én Windows Feature

Syntaks

```
WindowsFeature [string] #ResourceName
{
    Name = [string]
    [Credential = [PSCredential]]
    [Ensure = [string] { Absent | Present }]
    [IncludeAllSubFeature = [bool]]
    [LogPath = [string]]
    [Source = [string]]
}
```

Kodeblokk 51 - Syntaks WindowsFeature

Forklaring

Konfigurasjonen benytter DSC-ressursen WindowsFeature som er inkludert i modulen PSDesiredStateConfiguration. Ressursen benyttes til å installere og sikre at rollen "Search-Service" er tilstedte. Denne rollen skal installeres og senere stoppes og deaktiveres (beskrevet i 4.5.2 Service DisableWindowsSearch). Hvis dette ikke blir utført vil brukerne man få opp en feilmelding når man starter Outlook.

Egenskaper

- **Name** spesifiserer navnet på rollen eller funksjonen som skal håndteres av konfigurasjonen.
- **Ensure** bestemmer om rollen eller funksjonen skal være tilstede eller ikke.

Konfigurasjon

```
WindowsFeature InstallWindowsSearch
{
    Name            = "Search-Service"
    Ensure          = "Present"
}
```

Kodeblokk 52 - Konfigurasjon WindowsFeature

5.9.2 Service DisableWindowsSearch

Deaktivere en tjeneste

Syntaks

```
Service [String] #ResourceName
{
    Name = [string]
    [BuiltInAccount = [string]{ LocalService | LocalSystem |
NetworkService }]
    [Credential = [PSCredential]]
    [Dependencies = [string[]]]
    [DependsOn = [string[]]]
    [Description = [string]]
    [DisplayName = [string]]
    [Ensure = [string]{ Absent | Present }]
    [Path = [string]]
    [PsDscRunAsCredential = [PSCredential]]
    [StartupType = [string]{ Automatic | Disabled | Manual }]
    [State = [string]{ Running | Stopped }]
}
```

Kodeblokk 53 - Syntaks Service

Forklaring

Konfigurasjonen benytter DSC-ressursen Service som er inkludert i modulen PSDesiredStateConfiguration. Ressursen benyttes til å administrere tjenester i operativsystemet. I dette tilfellet sikre at tjenesten er stoppet og deaktivert slik at brukere ikke får opp en feilmelding da de starter Outlook.

Egenskaper

- **Name** spesifiserer navnet på tjenesten.
- **StartupType** spesifiserer hvilken tilstand tjenesten skal ha når systemet starter opp.
- **State** spesifiserer hva slags tilstand tjenesten skal ha.
- **DependsOn** spesifiserer en avhengighet til DSC-ressursen InstallWindowsSearchService. Det vil si at tjenesten ikke vil bli konfigurert før Search-Service er installert og tilgjengelig på systemet.

Konfigurasjon

```
Service DisableWindowsSearch
{
    Name            = "wsearch"
    StartupType     = "Disabled"
    State           = "Stopped"
    DependsOn       = "[WindowsFeature]InstallWindowsSearch"
}
```

Kodeblokk 54 - Konfigurasjon Service

5.10 Operativsystem

For maskiner med operativsystem Windows Server 2016 har tre spesifikke konfigurasjonsblokker.

5.10.1 WindowsFeature RemoveWindowsFeatures

Avinstallere én Windows Feature.

Syntaks

```
WindowsFeature [string] #ResourceName
{
    Name = [string]
    [Credential = [PSCredential]]
    [Ensure = [string] { Absent | Present }]
    [IncludeAllSubFeature = [bool]]
    [LogPath = [string]]
    [Source = [string]]
}
```

Kodeblokk 55 - Syntaks WindowsFeature

Forklaring

Konfigurasjonen benytter DSC-ressursen WindowsFeature som er inkludert i modulen PSDesiredStateConfiguration. Ressursen benyttes til å avinstallere og sikre at rollen "Windows-Defender" ikke er tilstedte. For Windows Server 2016 skal Windows Defender avinstalleres.

Egenskaper

- **Name** spesifiserer navnet på rollen eller funksjonen som skal håndteres av konfigurasjonen.
- **Ensure** bestemmer om rollen eller funksjonen skal være tilstede eller ikke.

Konfigurasjon

```
WindowsFeature RemoveWindowsFeatures
{
    Name            = "Windows-Defender"
    Ensure          = "Absent"
}
```

Kodeblokk 56 - Konfigurasjon WindowsFeature

5.10.2 Copy CopyAclUsersDefault

Kopierer Access Controll List (ACL) fra et objekt til et annet.

Syntaks

```
CopyAcl [string] #ResourceName
{
    SourcePath = [string]
    DestinationPath = [string]
    Recurse = [bool]
}
```

Kodeblokk 57 - Syntaks CopyAcl

Forklaring

Konfigurasjonen benytter DSC-ressursen CopyAcl som er inkludert i modulen AzureDSC_Modules. Denne ressursen benyttes til å kopiere Access Controll List (ACL) fra en mappe og sette dem på en annen. Dette behøves i konfigurasjonen for å rette opp i en sikkerhetsfeil som gjør at brukere får opprettet filer under "C:\users\default". Ressursen er utviklet i forbindelse med bachelorprosjektet og er må lastes opp manuelt i Azure Automation.

Egenskaper

- **SourcePath** spesifiserer hvor ACL skal kopieres fra
- **DestinationPath** spesifiserer hvor ACL skal kopieres til
- **Recurse** definerer om ACL skal settes rekursiv på alt innhold i DestinationPath

Konfigurasjon

```
CopyAcl CopyAclUsersDefault
{
    SourcePath              = "c:\users"
    DestinationPath         = "c:\users\default\"
    Recurse                 = $false
}
```

Kodeblokk 58 - Konfigurasjon CopyAcl

5.10.3 Copy CopyAclStartMenu

Kopierer Access Controll List (ACL) fra et objekt til et annet.

Syntaks

```
CopyAcl [string] #ResourceName
{
    SourcePath = [string]
    DestinationPath = [string]
    Recurse = [bool]
}
```

Kodeblokk 59 - Syntaks CopyAcl

Forklaring

Konfigurasjonen benytter DSC-ressursen CopyAcl som er inkludert i modulen AzureDSC_Modules. Denne ressursen benyttes til å kopiere Access Controll List (ACL) fra en mappe og sette dem på en annen. Dette behøves i konfigurasjonen for å rette opp i en sikkerhetsfeil som gjør at brukere får opprettet filer under "C:\users\default". Ressursen er utviklet i forbindelse med bachelorprosjektet og er må lastes opp manuelt i Azure Automation.

Egenskaper

- **SourcePath** spesifiserer hvor ACL skal kopieres fra
- **DestinationPath** spesifiserer hvor ACL skal kopieres til
- **Recurse** spesifiserer om ACL skal settes rekursiv på alt innhold i DestinationPath
- **PsDscRunasCredential** spesifiserer en legitimasjon som har rettigheter til å sette ACL på innholdet som er definert i DestinationPath. Dette må defineres i denne konfigurasjonen siden systemkontoen "NT AUTHORITY\SYSTEM" som DSC kjører med ikke har tilgang til en mappe under spesifiserte DestinationPath.

Konfigurasjon

```
CopyAcl CopyAclStartMenu
{
    SourcePath           = "c:\users"
    DestinationPath      = "C:\users\Default\...\Start Menu\"
    Recurse              = $true
    PsDscRunAsCredential = $Cred
}
```

Kodeblokk 60 - Konfigurasjon CopyAcl

6 Az PowerShell modul

Az er den nye PowerShell-modulen som bør benyttes til å administrere Azure. Det tar over for AzureRm, som fortsatt har support frem til 2020. PowerShell Az tilbyr fortløpende oppdatering med ny funskjonalitet, kortere kommandoer og kryssfunksjonalitet, som betyr at det fungerer uavhengig av operativsystem. Man kan altså kjøre de samme PowerShell-kommandoene på Windows, MacOS og Linux.

6.1 Installasjon av Az

For å installere Az PowerShell-modul på Windows må PowerShell-versjon 5.1 eller høyere og NET Framework 4.7.2 være tilgjengelig på maskinen.

Man installerer Az Powershell-modul med PowerShell-kommando visst i kodeblokk 58.

```
Install-Module -Name Az
```

Kodeblokk 61 - Installere Az PowerShell Module

6.2 Nyttige kommandoer

Dette underkapittelet viser de mest interessante og nyttige PowerShell-kommandoene i Az-modulen. Disse forenkler administrasjon av konfigurasjoner, kompilerte konfigurasjoner og noder.

6.2.1 Import-AzAutomationDscConfiguration

Denne kommandoen laster opp en konfigurasjonsfil (*.ps1) til Azure Automation. Man definerer ressursgruppe og Automation Account og stien til konfigurasjonsfila. Det er veldig viktig at konfigurasjonsfila har samme navn som konfigurasjonen. Det betyr at hvis det er definert "Configuration SessionHost {…}" i fila skal fila hete "SessionHost.ps1".

```
$Login = Connect-AzAccount
$Params = @{
    ResourceGroupName      = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName = 'azure-dsc-test-01';
    SourcePath             = 'SessionHost.ps1';
    Published              = $true;
    Force                  = $true;
    Verbose                = $true;

}
$ConfigFile = Import-AzAutomationDscConfiguration @Params
```

Kodeblokk 62 - Importere konfigurasjonsdokument

6.2.2 Start-AzAutomationDscCompilationJob

Denne kommandoen initierer kompileringsjobben av konfigurasjonen som ble lastet opp i forrige underkapittel. Flagget "IncrementNodeConfigurationBuild" spesifiseres for å bevare tidligere konfigurasjon og gi den nye konfigurasjonen et inkrementelt byggenummer. Ved å spesifisere egenskapen "Parameters" kan man kjøre kompileringsjobben inkludert parametere "OS" og "Collection". Parametere må inkluderes som en hash-tabell.

```
$Login = Connect-AzAccount
$ConfigParams = @{
    "OS"="srv2016";
    "Collection"="officefarm";
}
$Params = @{
    ResourceGroupName                = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName            = 'azure-dsc-test-01';
    ConfigurationName                = $configFile.Name;
    Parameters                       = $ConfigParams
    IncrementNodeConfigurationBuild = $true;
    Verbose                          = $true;
}
$CompilateConfig = Start-AzAutomationDscCompilationJob @Params
```

Kodeblokk 63 - Kompilere konfigurasjon med paramtere definert

6.2.3 Set-AzAutomationDscNode

Denne kommandoen tildeler en node en konfigurasjon. Som visst i kodeblokk 61 får noden "IT-AZDSCFARM02" konfigurasjonen som ble kompilert i forrige eksempel. For å tildele en node en konfigurasjon må man ha nodens ID i Azure. Det er slik siden det kan være flere noder med samme navn i Azure og det er ID-en som er den unike identifikatoren for en administrert maskin.

```
$Login = Connect-AzAccount
$Node = Get-AzAutomationDscNode @P -Name "IT-AZDSCFARM02"
$Params = @{
    ResourceGroupName        = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName    = 'azure-dsc-test-01';
    NodeConfigurationName    = $CompilateConfig.Name;
    Id                       = $Node.id;
    Force                    = $true;
    Verbose                  = $true;
}
$SetNodeConfig = Set-AzAutomationDscNode @Params
```

Kodeblokk 64 - Konfigurere node med konfigurasjon

6.2.4 Get-AzAutomationDscNodeConfiguration

Denne kommandoen presenterer de kompilerte konfigurasjonene som er tilgjengelig på Pull-serveren.

```
$Login = Connect-AzAccount
$Params = @{
    ResourceGroupName       = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName   = 'azure-dsc-test-01';
}
Get-AzAutomationDscNodeConfiguration @Params
```

Kodeblokk 65 - Hente DSC-konfigurasjoner

6.2.5 Get-AzAutomationDscNode

Denne kommandoen presenterer alle onboardede noder i Azure Automation.

```
$Login = Connect-AzAccount
$Params = @{
    ResourceGroupName       = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName   = 'azure-dsc-test-01';
}
Get-AzAutomationDscNode @Params
```

Kodeblokk 66 - Henter alle onboarde noder

6.3 Nyttige scripts

Dette underkapittelet inneholder noen få PowerShell-scripts som utfører administrativt arbeid med Azure DSC.

6.3.1 Endre nodekonfigurasjon

Dette scriptet tar konfigurasjonsdata som definert i kapittel 2.3.4 og oppdaterer nodene med gjeldende konfigurasjon. Skal det utføres endringer i nodene eller konfigurasjon skal dette først gjøres i konfigurasjonsdataen før dette scriptet kjøres.

```
$Login = Connect-AzAccount

foreach ($NodeObject in $ConfigData.AllNodes) {
    $NodeParams = @{
        ResourceGroupName     = "NTNU02-M-northeu-DSC-rg";
        AutomationAccountName = "azure-dsc-test-01";
        Name                  = "$NodeObject.NodeName";
    }
    $Node = Get-AzAutomationDscNode @NodeParams

    $ConfigParams = @{
        ResourceGroupName     = "NTNU02-M-northeu-DSC-rg";
        AutomationAccountName = "azure-dsc-test-01";
        NodeConfigurationName =
"$($NodeObject.Configuration).$($NodeObject.Collection).$($NodeObject.
OS)";
    }
    $Config = Get-AzAutomationDscNodeConfiguration @Params

    $SetParams ={
        ResourceGroupName     = "NTNU02-M-northeu-DSC-rg";
        AutomationAccountName = "azure-dsc-test-01";
        NodeConfigurationName = $($Config.Name);
        Id                    = $Node.Id
        Force                 = $true;
        Verbose               = $true;
    }
    try {
        $SetNodeConfig = Set-AzAutomationDscNode @Params
    }
    catch {
        Write-Output $_.Exception.Message
    }
}
```

Kodeblokk 67 - Oppdatere nodekonfigurasjon

6.3.2 Slette ubrukte nodekonfigurasjoner

Dette scriptet rydder opp og sletter ubrukte nodekonfigurasjoner i Azure DSC.

```
$Login = Connect-AzAccount
$Params = @{
    ResourceGroupName     = "NTNU02-M-northeu-DSC-rg";
    AutomationAccountName = "azure-dsc-test-01";
}

# Delete all configs that does not have nodes attatched
$ActiveConfigs = (Get-AzAutomationDscNode
@Params).NodeConfigurationName
foreach ($ActiveConfig in $ActiveConfigs) {
    Get-AzAutomationDscNodeConfiguration @Params | Where-Object -
Property Name -NE $ActiveConfig | Remove-
AzAutomationDscNodeConfiguration @P -Force -Verbose
}
```

Kodeblokk 68 - Slette ubrukte nodekonfigurasjoner

7 Driftsrutiner for forskjellige oppgaver

Dette kapittelet tar for seg forskjellige driftsrutiner for hvordan man utfører forskjellig arbeid. Hvert underkapittel inkluderer en liste over senario der rutinebeskrivelsen kan passe inn. Oppgavene kan være rutinemessige oppgaver eller utviklingsoppgaver.

7.1 Laste opp en ny konfigurasjon

Fremgangsmåten visst her kan benyttes i følgende senario:

- Det er laget ny konfigurasjonsfil som man ønsker å distribuere til Pull-serveren.
- Endring på nåværende konfigurasjonsfil som man ønsker å distribuere til Pull-serveren.

7.1.1 Laste opp og kompilere en ny konfigurasjon med PowerShell

For å laste opp og kompilere en ny konfigurasjon med PowerShell må PowerShell-script visst i kodeblokk 34 kjøres. Parameteret «SourcePath» refererer til PowerShell-fila der konfigurasjonen ligger lagret. De forskjellige PowerShell-kommandoene er beskrevet i kapittel 6.2.

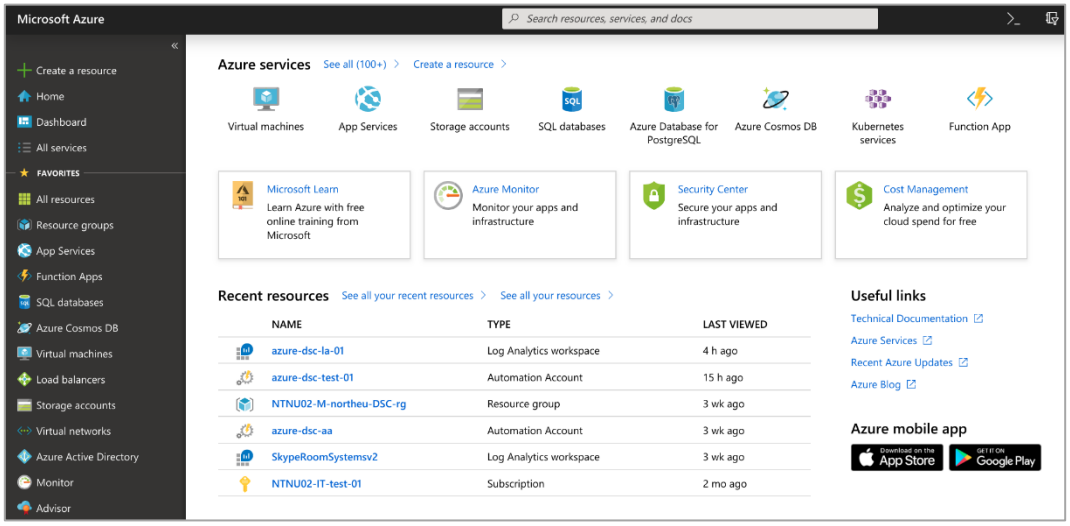
```
$Login = Connect-AzAccount

# Laster opp konfigurasjon
$ImportParams = @{
    ResourceGroupName      = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName = 'azure-dsc-test-01';
    SourcePath             = 'SessionHost.ps1';
    Published              = $true;
    Force                  = $true;
    Verbose                = $true;

}
$ConfigFile = Import-AzAutomationDscConfiguration @ImportParams

# Kompilerer konfigurasjon
$ConfigParams = @{
    "OS"="srv2016";
    "Collection"="officefarm";
}
$CompileParams = @{
    ResourceGroupName               = 'NTNU02-M-northeu-DSC-rg';
    AutomationAccountName           = 'azure-dsc-test-01';
    ConfigurationName               = $configFile.Name;
    Parameters                      = $ConfigParams
    IncrementNodeConfigurationBuild = $true;
    Verbose                         = $true;
}
$CompilateConfig = Start-AzAutomationDscCompilationJob @CompileParams
```

Figur 44 - Laste opp og kompilere ny konfigurasjon

7.1.2 Laste opp og kompilere en ny konfigurasjon i portalen

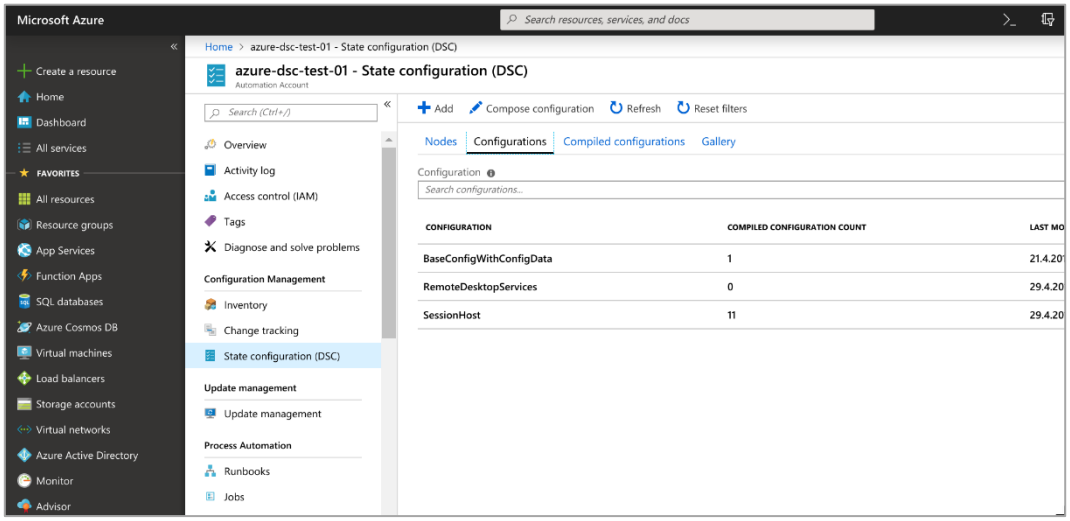
For å laste opp og kompilere en ny konfigurasjon må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg ønsket Automation Account



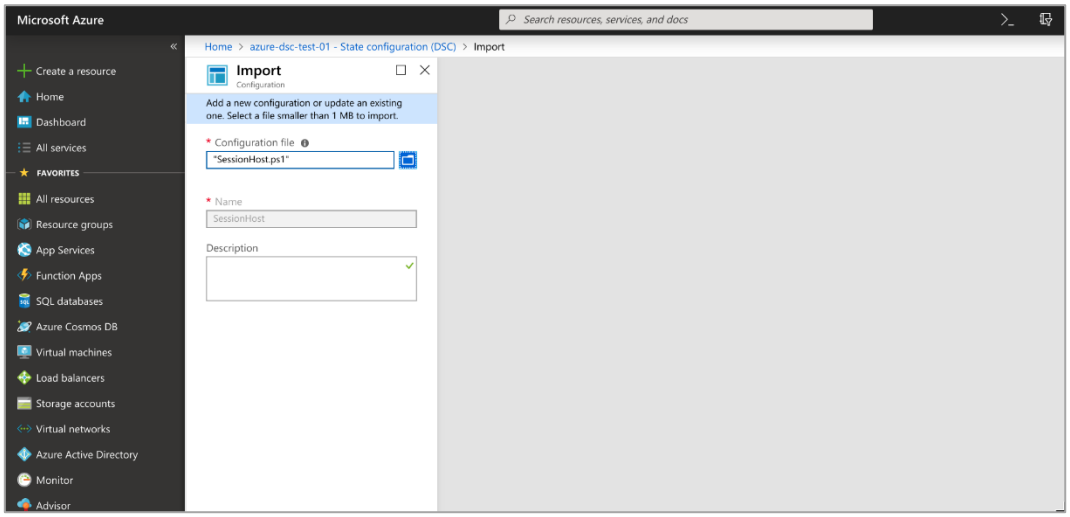
Figur 45 - Laste opp og kompilere ny konfigurasjon

3. Velg "State Configuration (DSC)" under "Configuration Management" i menyen til venstre
4. Velg "Configurations" i fanemenyen
5. Velg "Add" for å legge til ny konfigurasjon.



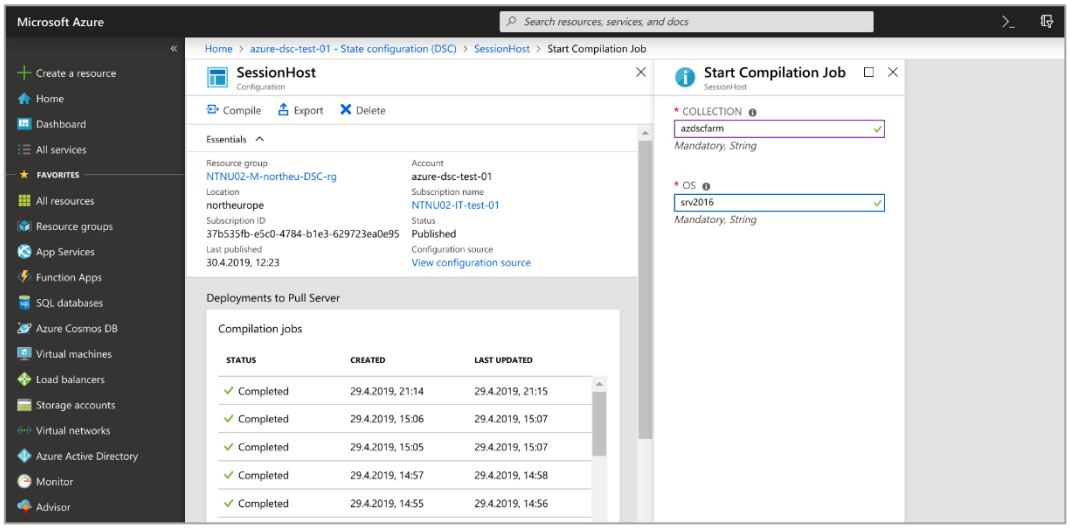
Figur 46 - Laste opp og kompilere ny konfigurasjon

6. Velg ønsket konfigurasjonsfil og avslutt med “OK”.



Figur 47 - Laste opp og kompilere ny konfigurasjon

7. Velg “Compile”
8. Skriv inn parametere og avslutte med “OK”:
    a. Collection - “azdscfarm”
    b. OS - “srv2016”

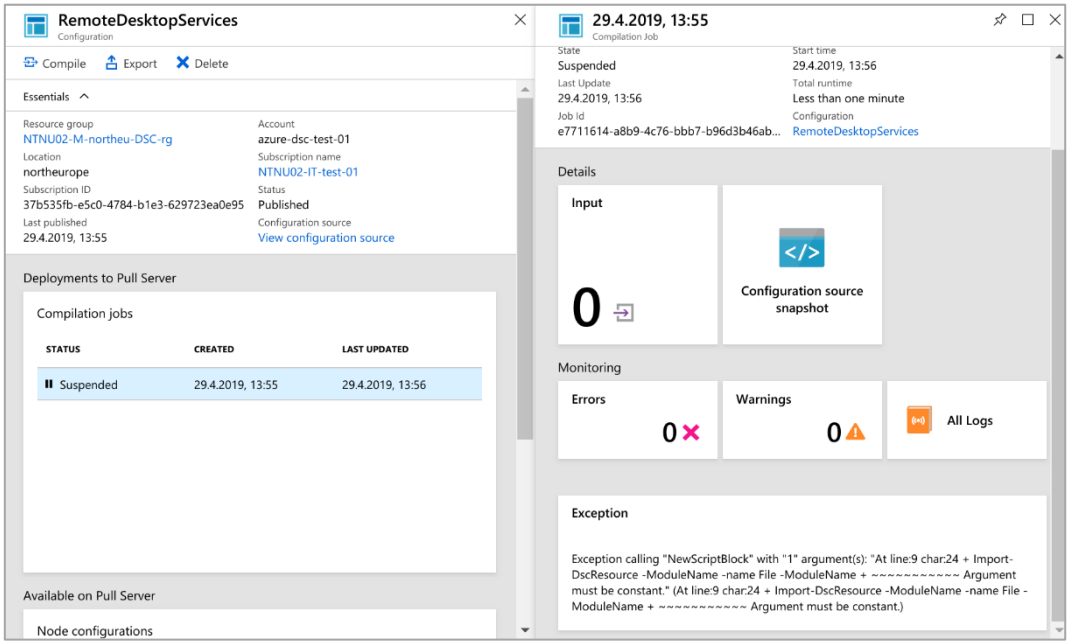


Figur 48 - Laste opp og kompilere ny konfigurasjon

Kompileringsjobben starter opp og den vil få navn “SessionHost.azdscfarm.srv2016” etter navnestandarden definert i konfigurasjonsfila. Kompileringsjobben vil gå igjennom fire faser før den er klar til bruk:

- **Queued** som betyr at jobben ligger i kø for å bli kompilert.
- **Starting** som betyr at kompileringsjobben starter opp.
- **Running** som betyr at kompileringsjobben kjører
- **Completed** som betyr at kompileringsjobben kjørte uten problemer.

Hvis kompileringsjobben feiler får den status “Suspended”. Ved å trykke på kompileringsjobben som visst på figur 39 får man opp informasjon, logger og feilmeldinger om jobben som kan benyttes til å feilsøke eventuelle problemer.



Figur 49 - Laste opp og kompilere ny konfigurasjon

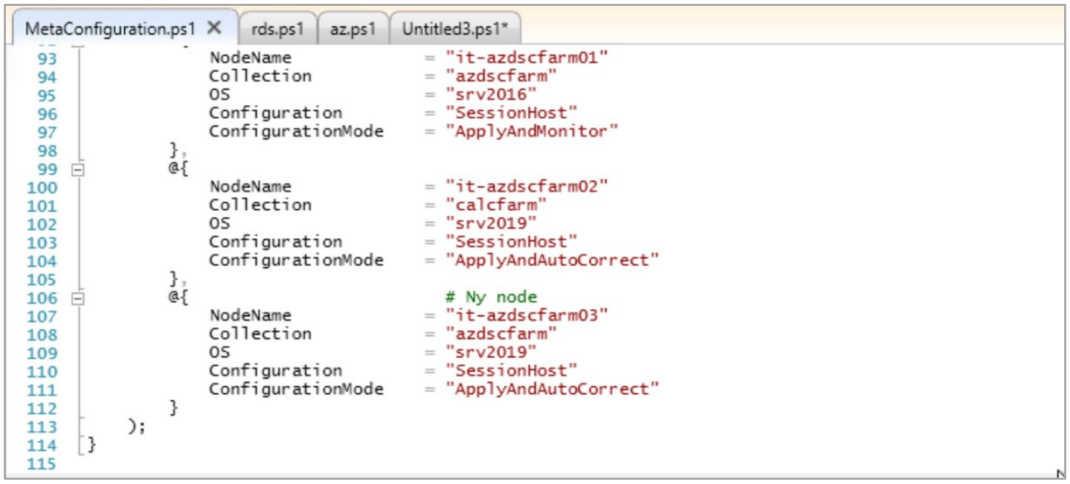
7.2 Onboarding av node

Som forklart i kapittel 2.3.3 vil det være behov for å re-onboarde eksisterende noder og onboarde nye noder jevnlig. Fremgangsmåten visst her kan benyttes i følgende senario:

- Onboarding av ny node
- Re-onboarding av node på grunn av utløpt sertifikat / reinstallasjon eller lignende.
- Endring av innstillinger i nodens Local Configuration Manager (LCM).

For å onboarde / re-onboarde en node må fremgangsmåte følges:

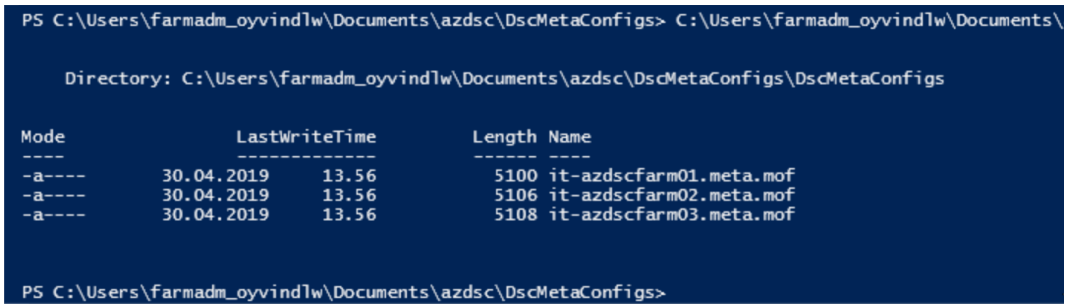
1. Åpne det tilpassede onboarding-scriptet.
2. Før opp en ny node / endre på eksisterende node i konfigurasjonsdataen som visst på figur 40.
    a. NodeName
    b. Collection
    c. OS
    d. Configuration
    e. ConfigurationMode



Figur 50 - Tilpasning av konfigurasjonsdata

3. Slett mappen "DscMetaConfigs" for å eventuelt fjerne gamle metakonfigurasjoner som ikke skal implementeres.
4. Kjør scriptet.

Som visst på figur 41 under ser man at det har kommet en ny Meta-mof for den nye maskinen.



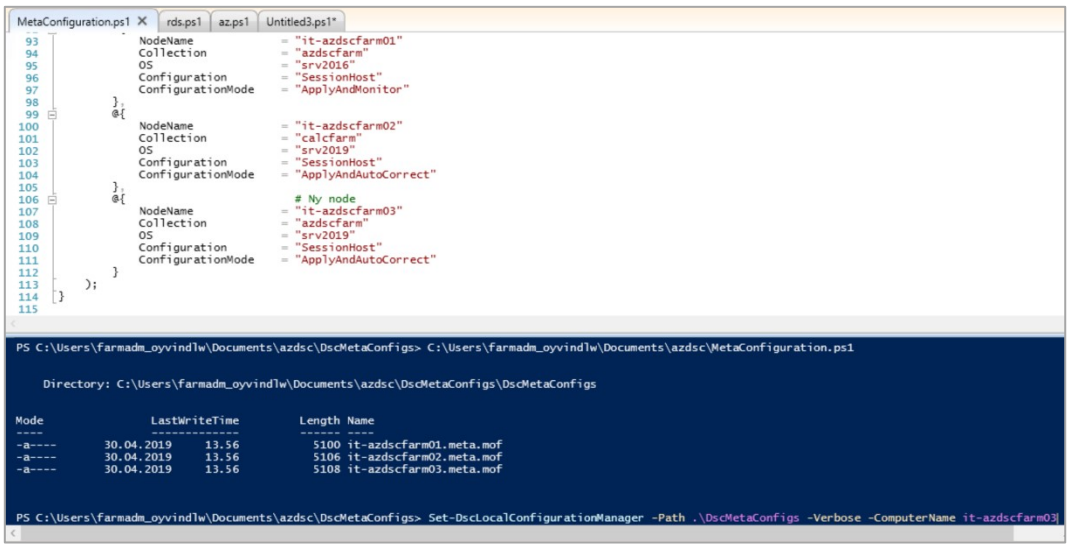
Figur 51 - Kompilering av Meta-konfigurasjon

5. Kjør PowerShell-kommando som visst i kodeblokk 66.

```
$Params = @{
    Path         = ".\DscMetaConfigs";
    ComputerName = "it-azdscfarm03";
    Verbose      = $true;
}
Set-DscLocalConfigurationManager @Params
```

Kodeblokk 69 - Konfigurere LCM

Det kan være lurt å definere datamaskinnavn i kommandoen slik at LCM ikke blir re-konfigurert for de eksisterende nodene. Med det tilpassede scriptet er ikke dette et stort problem konfigurasjonen alltid vil være lik om konfigurasjonsdataen er uendret. Da forblir meta-mof-en uendret.



Figur 52 - Konfigurasjon av LCM

7.3 Tildele konfigurasjon på node

Fremgangsmåten som visst her kan benyttes i følgende senario:

- Ønske om å endre konfigurasjon på en node utenfor konfigurasjonsdata.

For endring av konfigurasjon i konfigurasjonsdata se kapittel 6.3.1.

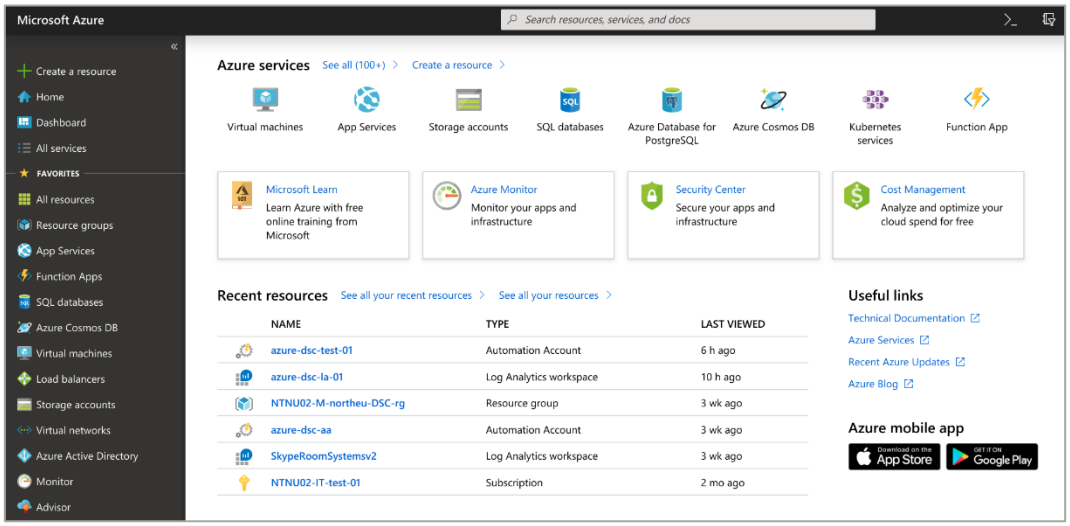
7.3.1 Tildele konfigurasjon på node med PowerShell

Dette vises i kapittel 6.2.5.

7.3.2 Tildele konfigurasjon på node i Portalen

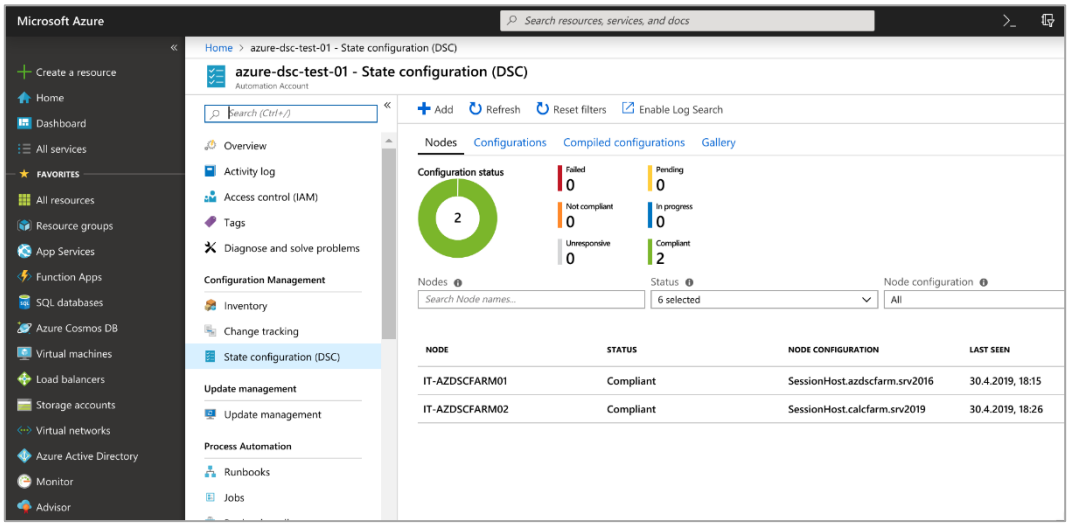
For å tildele og endre konfigurasjon på en node i portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg ønsket Automation Account



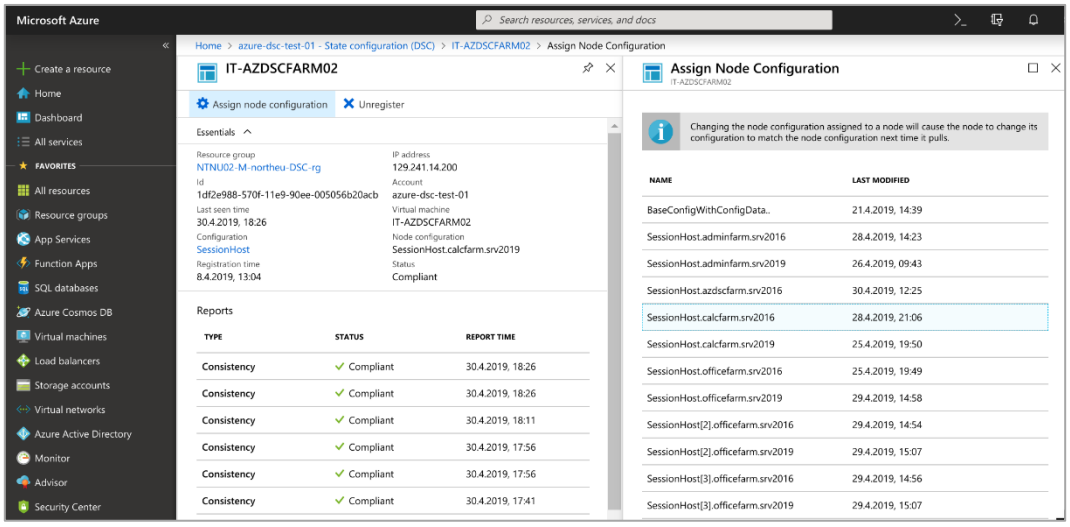
Figur 53 - Tildele konfigurasjon

3. Velg “State Configuration (DSC)” under “Configuration Management” i menyen til venstre
4. Velg node – IT-AZDSCFARM02



Figur 54 - Tildele konfigurasjon

5. Velg "Assign node configuration”
6. Velg ønsket konfigurasjon og avslutt med “OK”.



Figur 55 - Tildele konfigurasjon

7.4 Opprette ny egenkomponert DSC-ressurs

Fremgangsmåten som visst her kan benyttes i følgende senario:

- Opprette ny egenkomponert DSC-ressurs
- Endre på eksisterende DSC-ressurser

Se kapittel 4 Egen- og videreutviklede DSC-ressurser for bakgrunnsinformasjon om DSC-ressurser.

Underkapittelet bruker DSC-ressursen CopyAcl som eksempel på bruk av xDscResourceDesigner.

7.4.1 Installere xDscResourceDesigner

xDscResourceDesigner kan lastes ned fra PowerShell Gallery[16] eller installeres via PowerShellGet med kommando visst i kodeblokk 67.

```
Install-Module -Name xDSCResourceDesigner -RequiredVersion 1.10.0.0
```

Kodeblokk 70 - Installere xDscResourceDesigner

7.4.2 Opprette ressursegenskaper

Det første som må gjøres er å opprette DSC-ressursens egenskaper. Dette spesifiseres i variabler med PowerShell-kommandoen "New-DscResourceProperty".

SourcePath og DestinationPath settes til "Key" da dette sammen fungerer som en unik nøkkel som identifiserer DSC-konfigurasjonen. Det vil si at det ikke er mulig å ha flere konfigurasjonsblokker med CopyAcl ressursen som spesifiserer samme SourcePath og DestinationPath. Egenskapen Recurse settes til "Boolean" da dette spesifiserer om kopieringen skal kjøres rekursivt.

```
$SourcePath = New-xDscResourceProperty –Name SourcePath -Type String -
Attribute Key
$DestinationPath = New-xDscResourceProperty –Name DestinationPath -
Type String -Attribute Key
$Recurse = New-xDscResourceProperty –Name Recurce -Type Boolean -
Attribute Write
```

Kodeblokk 71 - Definere egenskaper til DSC-ressursen

[16] https://www.powershellgallery.com/packages/xDSCResourceDesigner/1.10.0.0

7.4.3 Opprette DSC-ressursen

Når egenskapene er valgt kjører man PowerShell-kommandoen "New-xDscResource" som visst i kodeblokk 69. Her spesifiserer man de forskjellige egenskapene man tidligere har lagd, navn på DSC-ressursen, navn på modulen og hvor den skal lagres. Ved kjøring vil xDscResourceDesigner opprette fil- og mappestrukturen som visst i Kapittel 4.

```
$Params = @{
    Name          = "CopyAcl"
    Property      = $SourcePath, $DestinationPath, $Recurse
    Path          = "C:\Program Files\WindowsPowerShell\Modules"
    ModuleName    = "AzureDsc_Modules"
}
New-xDscResource @Params
```

Kodeblokk 72 - Opprette DSC-ressursen

Er det behov for å endre parametere i etterkant kan man definere disse på samme måte, men bytte ut PowerShell-kommandoen "New-xDscResource" med "Update-xDscResource".

7.4.4 Programmere ressursen

Siden xDscResourceDesigner oppretter alle filer og mapper som behøves står det kun igjen å programmere selve funksjonaliteten til ressursen. PowerShell-koden skal ligge i filen "AzureDsc_Modules\DSCResources\CopyAcl\CopyAcl.ps1". Filen opprettes automatisk og har allerede get-, set-, og testmetodene klare. Egenskapene som vi tidligere definerte er satt in som parametere og er klare til bruk. Det eneste som står igjen er å programmere funksjonaliteten inn i strukturen.

De viktigste reglene man må følge når man utvikler DSC-ressurser er:

- At Test-TargetResource returnerer en boolsk verdi i henhold til om ressursen er i ønsket tilstand og
- At Get-TargetResource returnerer en hash-tabell med ressursens egenskaper.

7.4.5 Klargjøre og publisere til Azure Automation

Når ressursen er ferdig bør ressursen testes grundig. Den enkleste måten å utvikle og teste ressursens funksjonalitet var å kjøre get-, set- og testfunksjonene lokalt på en Session Host. Da behøver man ikke vente på opplastning, kompilering, og oppdatering av konfigurasjon. Dette kan testes senere. I tillegg inkluderer modulen xDscResourceDesigner to test-kommandoer som kan benyttes til å gi en indikasjon på at ressursen har korrekt funksjonalittet.

- **Test-xDscResource** kontrollerer om PowerShell-koden (CopyAcl.ps1) følger reglene til DSC.
- **Test-xDscSchema** kontrollerer om skjemafilen (CopyAcl.Schema.mof) følger reglene til DSC.

Når DSC-ressursen er ferdig programmert, testet og er klar til publisering må Powershell-modulen komprimeres til en ZIP-fil. Dette kan gjøres i Windows ved å høyreklikke på modulmappa og velge komprimer eller ved å kjøre PowerShell-kode som visst på kodeblokk 70.

```
Compress-Archive ./AzureDSC_Modules ./AzureDSC_Modules.zip
```

Kodeblokk 73 - Komprimere PowerShell-modulen

Opplastning av Modul til Azure Automation kan ses i kapittel 2.4.5.

7.5 Opprette ny Composite Resource (Kompositttressurs)

Fremgangsmåten som visst her kan benyttes i følgende senario:

- Opprette ny kompositttressurs
- Endre på eksisterende kompositttressurs

Se kapittel 4.5 Composite Resource - Baselinekonfigurasjon for bakgrunnsinformasjon om Composite Resource.

Det finnes ingen kjente verktøy for å automatisk opprette strukturen til kompositttressurser lik xDscResourceDesigner for DSC-ressurser. Derfor må fil- og mappestrukturen opprettes manuelt.

7.5.1 Opprette fil- og mappestruktur

1. Opprett fil- og mappestrukturen som visst på figur 46. Ikke opprett SessionHostBaseline.psd1.

```
$env:ProgramFiles\WindowsPowerShell\Modules (folder)
    |- AzDSC_Modules (folder)
        |- DSCResources (folder)
            |- SessionHostBaseline (folder)
                |- SessionHostBaseline.psd1 (file, required)
                |- SessionHostBaseline.schema.mof (file, required)
```

Figur 56 - Mappestruktur kompositttressurs

2. Opprett PowerShell-Manifeste med kommando som visst på kodeblokk 47.

```
$Params = @{
    Path          = "SessionHostBaseline.psd1"
    ModuleVersion = "1.0"
    Author        = "Ola Nordmann"
}
New-ModuleManifest @Params
```

Figur 57 - Opprette PowerShell-Manifeste

7.5.2 Lag konfigurasjonen

Legg ønsket konfigurasjon inn i "SessionHostBaseline.schema.mof". Som visst på figur 48 ser man at det kun er "Configuration"- og konfigurasjonsblokka som er definert. Nodeblokka er ikke nødvendig og skal ikke defineres i en kompositttressurs.

Denne ressursen og "SessionHostBaseline" som er utviklet i prosjektet har ingen definerte parametere. Det er mulig å legge inn parametere som sendes videre til konfigurasjonsblokkene i kompositttressursen hvis det er ønskelig. Dette gjøres på samme måte som visst i kapittel 5.4.

SessionHostbaseline.schema.mof

```
Configuration SessionHostBaseline
{
    Import-DscResource -ModuleName PSDesiredStateConfiguration -name WindowsFeatureSet

    WindowsFeatureSet InstallWindowsFeaturesBase
    {
        Name                    = @("RDS-RD-Server", "XPS-Viewer", "NET-Framework-Core")
        Ensure                  = "Present"
    }
}
```

Figur 58 - Mof-fila som skal inneholde selve konfigurasjonen

7.5.3 Klargjøre og publisere kompositttressurs

Når kompositttressursen er ferdig konfigurert, testet og er klar til publisering må Powershell-modulen komprimeres til en ZIP-fil. Dette kan gjøres i Windows ved å høyreklikke på modulmappa og velge komprimer eller ved å kjøre PowerShell-kode som visst på kodeblokk 49.

```
Compress-Archive ./AzureDSC_Modules ./AzureDSC_Modules.zip
```

Figur 59 - Komprimere PowerShell-modulen

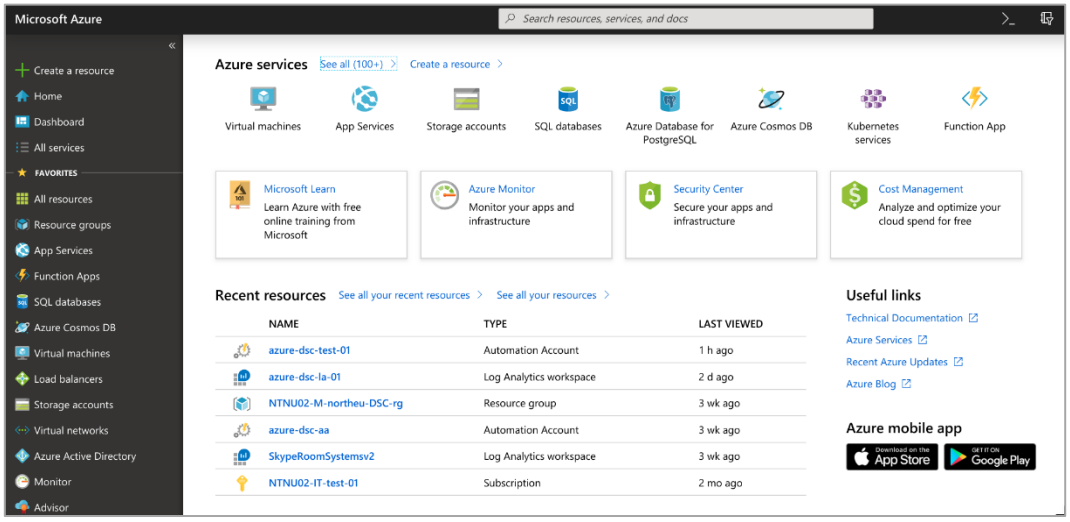
Opplastning av Modul til Azure Automation vises i kapittel 2.4.5.

7.6 Oppdatere Azure Automation moduler

Standardmodulene som kommer med Azure Automation blir ikke automatisk oppdatert. Dette må tidvis gjøres for å få inn oppdateringer og eventuelle feilrettinger.

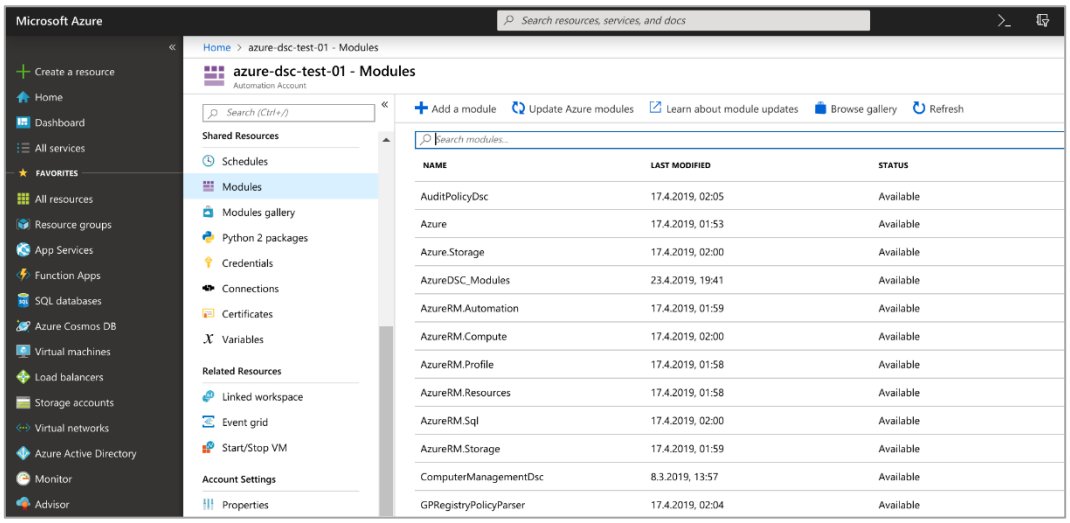
For å oppdatere importerte moduler i Azure portalen må følgende fremgangsmåte følges:

1. Åpne og logg inn på portal.azure.com.
2. Velg Automation Account



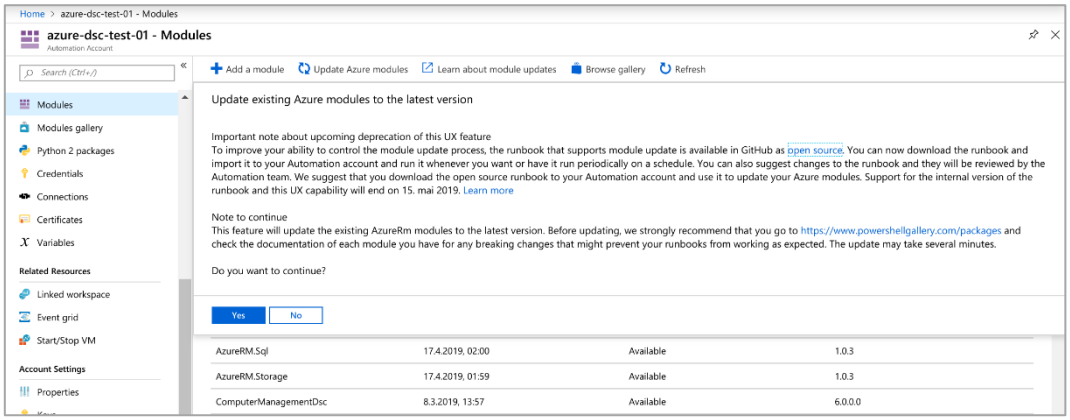
Figur 60 - Oppdatere Azure Automation Moduler

3. Velg “Module” under “Shared Resources” i menyen til venstre
4. Velg “Update Azure modules”



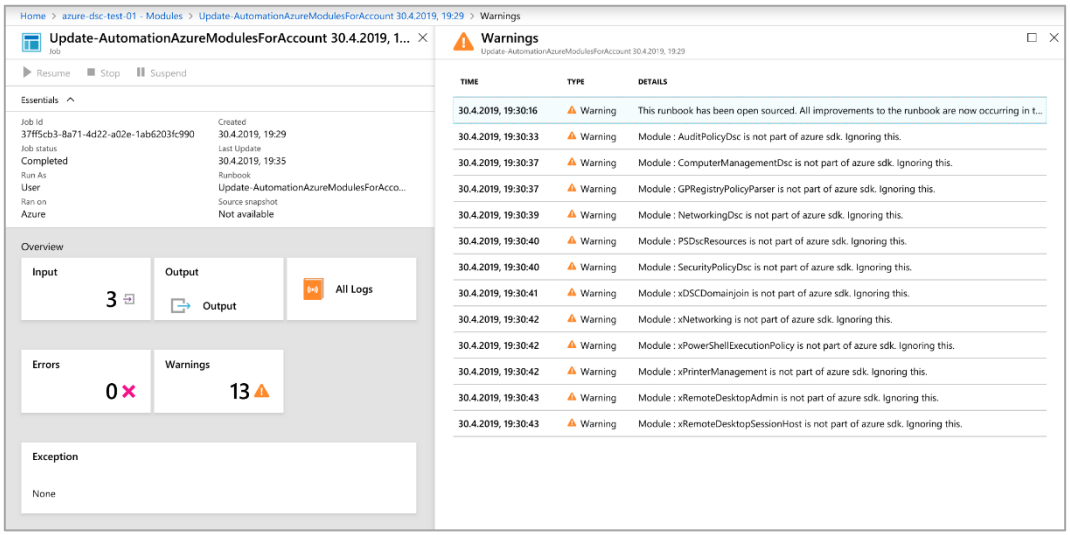
Figur 61 - Oppdatere Azure Automation Moduler

5. Les over informasjonsmelding og velg “Yes”.



Figur 62 - Oppdatere Azure Automation Moduler

6. Oppdateringen starter en Runbook der man kan se advarsler, logger og eventuelle feilmeldinger.



Figur 63 - Oppdatere Azure Automation Moduler

8 Kjente problemer og begrensninger

De forskjellige systemene og teknologien benyttet i prosjektet er helt ferskt og er under kontinuerlig utvikling. Som følge av det vil man kunne oppleve ustabilitet, tekniske problemer og at dokumentasjonen oppdateres kontinuerlig. For hvert system er som regel en dokumentasjon på kjente og vanlige feil eller oversikt over feil som jobbes med. Dette blir lenket fortløpende i dette kapittelet.

Azure Automation har en egen dokumentasjonsside[17] med "vanlige feil" som oppdateres fortløpende.

Azure DSC har en egen dokumentasjonsside[18] med "vanlige feil" som oppdateres fortløpende.

DSC-ressurser som har navn som starter på "x" må man regne med at det kan bli endringer i fremtiden. X-en står for at ressursen er eksperimentell og Microsoft garanterer ikke at funksjonaliteten i dag vil være lik i morgen. Derfor er det viktig at man tester funksjonaliteten nøye før man setter konfigurasjonen i drift. Man må også teste ressursene nøye før man oppdaterer DSC-ressursene i Azure Automation som forklart i kapittel 7.6.

Ellers for DSC-ressurser kan det være lurt å følge med på GitHub[19] der kildekoden til mange ressurser ligger lagret. Det er en egen fane med "Issues" der feil og mangler blir meldt inn og dokumentert. Opplever man også problemer er dette stedet man skal melde inn problemer.

Az CmdLet har en side på GitHub[20] der det er en egen fane med "Issues".

[17] https://docs.microsoft.com/nb-no/azure/automation/troubleshoot/desired-state-configuration
[18] https://docs.microsoft.com/en-us/powershell/wmf/5.0/limitation_dsc
[19] Eksempel på DSC-ressurs: https://github.com/PowerShell/ComputerManagementDsc/issues
[20] https://github.com/Azure/azure-powershell

Referanser

[1] “Azure Resource Manager overview” Internett: https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-overview, Jan. 02, 2019 [Feb. 9, 2019].

[2] “An introduction to Azure Automation” Internett: https://docs.microsoft.com/en-us/azure/automation/automation-intro, Okt. 10, 2019 [Feb. 1, 2019].

[3] “Manage Azure Automation Run As accounts” Internett: https://docs.microsoft.com/nb-no/azure/automation/manage-runas-account, Sep. 12, 2018 [Feb. 11, 2019].

[4] “Onboarding machines for management by Azure Automation State Configuration” Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding, Aug. 8, 2018 [Feb. 11, 2019].

[5] “DSC Resources” Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/resources, Des. 12, 2018 [Feb. 13, 2019].

[6] “Get-Set-Test Internett”: https://docs.microsoft.com/en-us/powershell/dsc/resources/get-test-set, Des. 12, 2018 [Feb. 13, 2019].

[7] “Using the Resource Designer tool” Internett: https://docs.microsoft.com/en-us/powershell/dsc/resources/authoringresourcemofdesigner, Des. 06, 2017 [Feb. 16, 2019].

[8] “Forward Azure Automation State Configuration reporting data to Log Analytics” Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-diagnostics, Jun. 11, 2018 [Feb. 17, 2019].

[9] “Certificate expiration and reregistration Internett: https://docs.microsoft.com/en-us/azure/automation/automation-dsc-onboarding#certificate-expiration-and-reregistration, Jul. 08, 2018 [May. 07, 2019].

[10] "Credential asset" Internett: https://docs.microsoft.com/en-us/azure/automation/shared-resources/credentials, Apr. 12, 2019 [May, 07, 2019]

[11] "Domain Credentials: https://docs.microsoft.com/en-us/powershell/dsc/configurations/configdatacredentials#domain-credentials, Jun. 12, 2019 [May, 07, 2019]

[12] "Using Event Logs to diagnose errors in desired state configuration": https://devblogs.microsoft.com/powershell/using-event-logs-to-diagnose-errors-in-desired-state-configuration/, Jan. 03, 2014 [May, 07, 2019]

[13] "DSC Resource-Kit Flourishes as open source": https://devblogs.microsoft.com/powershell/dsc-resource-kit-flourishes-as-open-source/, Jun. 23, 2015 [May, 08, 2019]

[14] "Import-DscResource": https://docs.microsoft.com/en-us/powershell/dsc/configurations/import-dscresource, Des. 12, 2018 [May, 08, 2019]

[15] "Depends On": https://docs.microsoft.com/en-us/powershell/dsc/configurations/resource-depends-on, Des. 12, 2018 [May, 08, 2019]

Øyvind Lotten Westrum

Azure Automation State Configuration for administrasjon av Windows-servere

NTNU
Kunnskap for en bedre verden