Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for informasjonssikkerhet og kommunikasjonsteknologi

Petter Jacob Brautaset, Vilja Lauritsen Langseth, Olav Andreas Strandjord og Edward Cornelius Haukø Svihus

Automatisering av konfigurasjon og testoppsett i nettverkslaboratoriet ved NTNU i Gjøvik

Bacheloroppgave i Digital infrastruktur og cybersikkerhet
Veileder: Ernst Gunnar Gran
Mai 2022

NTNU
Kunnskap for ei betre verd

Automatisering av konfigurasjon og testoppsett i nettverkslaboratoriet ved NTNU i Gjøvik

Petter Jacob Brautaset
Vilja Lauritsen Langseth
Olav Andreas Strandjord
Edward Cornelius Haukø Svihus

2022/05/20

ABSTRACT

Title	Automation of the Configuration and Test Setup in the Network Laboratory at NTNU in Gjøvik	**Date :** 20/05/2022
Participants	Petter Jacob Brautaset Vilja Lauritsen Langseth Olav Andreas Strandjord Edward Cornelius Haukø Svihus	
Supervisor(s)	Ernst Gunnar Gran, Associate Professor	
Employer	Eigil Obrestad, Assistant Professor, Senior Engineer	
Keywords	Automation, Cisco-laboratory, Programming, Network-administration, Cisco Systems	

Number of pages/words : 72	**Number of appendix** : 11	**Availability :** Yes

The Cisco-laboratory in the Amethyst-building at Norwegian University of Science and Technology (NTNU) in Gjøvik is designed for teaching courses in network equipment, more specifically Cisco-routers and switches. Much time is used in preparation work before each lesson. Eigil Obrestad, who is responsible for the laboratory, has discovered that parts of the laboratory lessons can be automated and streamlined.

This project will address the different solutions which have the potential to increase efficiency by reducing the preparation time used prior to a laboratory lesson. After looking at different solutions for automation, the team discovered that the workflow could be more or less automated. Some of the solutions discovered require manual labour, but others could be automated to a larger degree. The team has decided to use a solution which is fully automated, using tools including Cisco Autoinstall, Python and Ansible.

The solution involves a program written in Python to discover the network devices in the Cisco-laboratory, and the use of Ansible to facilitate the transfer of customized configuration to the respective devices through a computer with the operating system Ubuntu installed. The solution will enable the administrators to automate the setup of all routers and switches in the Cisco-laboratory.

This allows the lecturers to configure the devices in a well-tested and identical approach each time, which will result in fewer errors in the configuration, while also saving preparation time. As a consequence, the lecturers can focus more on the content of their lectures, rather than the preparation work related to each laboratory session.

SAMMENDRAG

Tittel	Automatisering av konfigurasjon og testoppsett i nettverkslaboratoriet ved NTNU i Gjøvik	**Dato :** 20/05/2022
Deltaker(e)	Petter Jacob Brautaset Vilja Lauritsen Langseth Olav Andreas Strandjord Edward Cornelius Haukø Svihus	
Veileder(e)	Ernst Gunnar Gran, Førsteamanuensis	
Oppdragsgiver	Eigil Obrestad, Universitetslektor, Senioringeniør	
Stikkord/nøkkelord (3-5 stk)	Automatisering, Cisco-laboratorie, Programmering, Nettverksadministrering, Cisco Systems	

Antall sider/ord : 72	**Antall vedlegg :** 11	**Publiseringsavtale inngått :** Ja

Cisco-laboratoriet i Ametyst-bygget ved Norges teknisk-naturvitenskapelige universitet (NTNU) i Gjøvik er benyttet i undervisningssammenheng for opplæring i bruk av nettverksutstyr, deriblant Cisco-rutere og svitsjer. Mye tid går til med å forberede Cisco-laben i forbindelse med undervisning. Laboratorieansvarlig Eigil Obrestad har oppdaget at deler av forberedelsene til laboratorieøvelsene kan automatiseres og effektiviseres.

Oppgaven vil ta for seg de ulike løsningene som vil øke effektiviteten og redusere tidsbruk i forbindelse med forberedelse av nettverksutstyr før en eventuell laboratorieøvelse. Ved å utforske ulike løsninger for automasjon, vil gruppen oppdage at arbeidsflyten i større eller mindre grad kan automatiseres. Noen av løsningene krever mer manuelt arbeid, mens andre løsninger kan i større grad automatiseres.

For å skape en optimalisert løsning som i større grad automatiserer manuelt arbeid, ble det utformet en løsning som benytter seg av Cisco Autoinstall, Python-skripting og konfigurasjonstyringsverktøyet Ansible. Programmering kan bli brukt som et verktøy for å oppdage nettverksutstyr i Cisco-laben og opprette en forbindelse til de respektive enhetene. Ansible vil gjøre det mulig å overføre tilpasset konfigurasjon til samtlige enheter via en datamaskin med den åpne linuxdistribusjonen Ubuntu installert som operativsystem.

Denne løsningen vil gjøre det mulig for administratorene å automatisere oppsettet av alle rutere og svitsjer i Cisco-laboratoriet. Dette tillater foreleserne å konfigurere enhetene identisk hver gang, noe som vil resultere i mindre feil i konfigurasjonen, samt redusere forberedelsestid. Som en bonus kan foreleserne fokusere mer på innholdet i forelesningene, istedenfor forberedelsesarbeidet relatert til hver laboratorieøvelse.

Forord

Forfatterne av denne bacheloroppgaven; Petter Jacob Brautaset, Vilja Lauritsen Langseth, Olav Andreas Strandjord og Edward Cornelius Haukø Svihus vil gjerne takke veileder Ernst Gunnar Gran for god veiledning og et godt samarbeid i løpet av prosjektet.

Vi vil også gjerne takke oppdragsgiveren Eigil Obrestad ved Norges teknisk-naturvitenskapelige universitet (NTNU) for et spennende og engasjerende prosjekt. Vi setter pris på tilgjengeligheten og muligheten både oppdragsgiver og veileder har gitt oss. De har vært tilgjengelig når nødvendig, og har stilt opp med god veiledning og anbefalinger i løpet av prosjektet. Vi vil takke dem for tilliten i forbindelse med oppgaven, og håper at løsningen står til forventningene.

Innhold

Figurer

Tabeller

Kodelister

Akronymer

AUX Auxiliary. xxi

BOOTP Bootstrap Protocol. 12

DHCP Dynamic Host Configuration Protocol. xiii, xxii, 11, 14, 15

DNS Domain Name System. 14

EEM Embedded Event Manager. xiii, xxii, 11, 15, 16

GRE Generic Routing Encapsulation. xxii

LAN Local Area Network. xxiv

MAC Media Access Control. xxiv, 39–41, 44

NTNU Norges teknisk-naturvitenskapelige universitet. vii, 1, 5–8, 65

PnP Plug And Play. xiii, 11, 22

SLARP Serial Line Address Resolution Protocol. 12

SSH Secure Shell. xiii, xxv, 11, 14, 26, 27

TCL Tool Command Language. xxv, 11, 27

TFTP Trivial File Transfer Protocol. xxvi, 11, 12, 28

USB Universal Serial Bus. xiii, 11, 28, 29

Ordliste

administrert node Nettverksutstyr som administreres med Ansible. xiii, xxv, 16–20, 50

Ansible Et open-source programvareverktøy for klargjøring og konfigurering av programvare. xiii, xxi, xxiii–xxv, 16–18, 20, 21, 32, 33, 35, 37, 41, 43, 49–53, 58, 60, 62

Ansible Galaxy Ansible Galaxy er en del av Ansible programverktøyet, og er nettsideområdet der brukere kan dele roller og kommandolinjen for å installere, lage og administrere roller. 51

ARP En protokoll eller en prosedyre som kobler en flyktig IP-adresse til en bestemt fysisk maskinvareadresse (MAC) i et lokalnett (LAN). 32, 39

AUX-port Auxiliary-porten på en ruter er en port som gjerne brukes for alternativ tilkoblingsmetode hvis den vanlige fjernstyringsmetoden blir utilgjengelig. AUX-porten bruker vanligvis rollover-kabel. 24, 25

Bash Bourne Again Shell (Bash) er et fritt og åpent UNIX-kommandolinjeskall og kommandospråk. 33

BIN-fil En binær fil der innholdet kan bli tolket og forstått av et program eller en maskinvareprosessor som forstår formateringen. 53, 55

Bolt Et automatisk konfigureringsverktøy med åpen kildekode. 19, 20

broadcast Å overføre et program eller informasjon ved hjelp av et trådløst signal. 39

Chef Et konfigurasjonsstyringsverktøy som er basert på en klient/tjener teknologi avsnitt 2.5.2. xiii, 16, 18–20

Cisco Teknologiselskap som spesialiserer seg på nettverksprodukter og telekommunikasjon. 5, 12, 18, 22, 25, 38, 40, 50, 51

Cisco 2901 ISR En administrert ruter-modell lansert av Cisco 13.oktober 2009 [1]. xxiv, 2, 22–24, 28, 56

Cisco 4221 ISR En administrert ruter-modell lansert av Cisco 30.november 2016 [2]. xxiv, 2, 28, 56, 59, 61, 62

Cisco AutoInstall Cisco AutoInstall er et program som tillater nettverksutstyret å hente konfigurasjoner fra en TFTP-server. xi, 13, 27, 28, 31–33, 35, 36, 59–62

Cisco Catalyst 2960 En administrert lag-2 Cisco svitsj-modell lansert av Cisco 18.september 2005 [3]. xxiv, 2, 16, 29, 56

Cisco Catalyst 2960 Plus En administrert lag-2 Cisco svitsj-modell lansert av Cisco 03. juni 2013 [3]. 2, 16, 29, 56, 59–61

Cisco Catalyst 3650 En administrert lag-3 Cisco svitsj-modell lansert av Cisco 1.september 2013 [4]. xi, xvii, xxiv, 2, 20, 28, 53, 55, 56, 66, 87

Cisco-lab Laboratoriet med Cisco-utstyr på NTNU i Gjøvik. xiii, xxiv, 1–3, 5–7, 12, 14, 15, 18, 20–25, 27, 28, 32, 35, 36, 40, 44, 49, 52, 59–63, 65, 66

DHCP Dynamic Host Configuration Protocol (DHCP) er en protokoll som er laget for å tildele IP-adresseer og andre relaterte nettverkskonfigurasjoner. 12, 14, 21, 32, 35, 53

EEM Embedded Event Manager (EEM) er en programvarekomponent som finnes i mange Cisco-svitsjer og rutere. Den kan konfigureres slik at den reagerer hvis gitte hendelser oppstår på maskinen og utfører prekonfigurerte kommandoer på bestemte tidspunkter. 27, 28

Ethernet-kabel En Ethernet-kabel er en kabel som brukes til å skape kommunikasjon mellom nettverksutstyr, som benytter RJ45-kontakten og følger IEEE 802.3 standarden. 2, 65

flashminne Flashminnet er en lagringsbrikke der IOS-oppstartsfilen og annen data er lagret på Cisco-utstyr. 14, 51, 59

grafisk brukergrensesnitt Kontaktflaten mellom bruker og maskinens operativsystem og programmer [5]. Det grafiske aspektet innebærer at bruker kan klikke på ikoner for å styre programmer. 64, 66

GRE-tunnel Generic Routing Encapsulation (GRE)-tunnel er en protokoll som pakker datapakker med en type rutingsprotokoll inn i en annen rutingsprotokoll for å forenkle traversering av pakken over flere nettverk. 65

grensesnitt Kontaktflaten mellom ulike enheter eller delsystemer som skal virke sammen og må derfor kunne kommunisere med hverandre. xxiv, 24

internett-svitsj Svitsjene som er montert øverst på hvert rack i alle podene. Brukes til å koble opp nettverksutstyr til Internett. 2, 14, 44, 65

inventar En liste av administrerte noder, kan inneholde IP-adresser og grupper. 17, 49–52, 60

IOS Internetwork Operating System, operativsystemet Cisco-nettverksutstyr har installert. 12, 27, 37, 51–53, 55

IOS-image En fil som inneholder operativsystemet til nettverksutstyret. 22, 25, 51, 59, 61, 66

IP-adresse En unik identifikator som blir brukt i lokale og globale nettverk for nettverksutstyr. xxii, xxiii, 12, 14, 15, 18, 24, 26, 32, 33, 35, 37–41, 43, 44, 52, 53

Knife Kommandolinjeverktøy som tilfører et grensesnitt mellom en lokal Chef-mappestruktur og Chef-Infra Server. Knife er nyttig for å administrere noder, kokebøker, roller etc.. 19

kokebok En kolleksjon med instruksjoner som definerer jobboppgaver avsnitt 2.5.2. 18, 19

kolleksjon Distribusjonsformat for Ansible-innhold. 17, 18, 50, 51, 53

Kommandolinjeverktøy Er et verktøy som tar i bruk kommandolinjen (også kjent som en CLI). 19

konfigurasjonsfil En fil som er skrevet i IOS-syntaks som bestemmer hvordan nettverksutstyret oppfører seg. xxv, xxvi, 1, 2, 6, 12–14, 21–24, 28, 32, 33, 35, 63

konfigurasjonsmodus En kommandomodus på nettverksutstyret hvor det blir konfigurert [6]. 28

konsollkabel En konsollkabel er en type null-modem kabel som kan kobles fra en dataterminal til en konsollport på et nettverksutstyr. 21

kontrollnode Maskin med Ansible installert som kan kjøre tilhørende kommandoer og Playbooker. Maskiner med Windows installert støtter ikke kontrollnode-modus. 17, 20, 49

kringkastingsadresse En IP-adresse som brukes til å nå alt nettverksutstyr som er på det samme lokalnettverket. 37

kronjob Kronjob er et verktøy som brukes til å automatisk kjøre skripts eller IOS-kommandoer etter planlagte tidspunkter. 28, 37

kryptografisk nøkkel En nøkkel med et stykke informasjon som er nødvendig for å løse en kryptografisk algoritme. 27, 37

Linux Et operativsystem utgitt i åpen-kildekode basert på UNIX. 6, 20, 25

lokalnettverk Local Area Network (LAN), eller lokalnett, er som definert av Store Norske Leksikon "et datakommunikasjonssystem som knytter sammen flere datamaskiner eller datamaskin-systemer innenfor et begrenset geografisk område (for eksempel en bygning) og lar brukerne dele ressurser og tjenester som skrivere, lagringsenheter, programvare og Internett-tilkobling."[7] . xxiii, 25, 28, 31, 32, 36, 60, 61

loopback-grensesnitt Et grensesnitt som adresserer seg selv. 24

MAC-adresse Media Access Control er en unik identifikator gitt til fysisk utstyr for identifisering av utstyr. Benyttes til identifisering av nettverksutstyr. En standard som er gitt av produsenten og skal ikke endres. 26, 32, 33, 40

maskinvare De fysiske komponentene i et datasystem, det vil si skjerm, tastatur, harddisk, prosessor og kretskort, samt tilkoblet utstyret som skriver, skanner, mus etc.. 1, 25

modul Hver modul har sin bruksmåte, en task kan bruke en modul, kan ha flere moduler i en Playbook. 51–53, 56

nettverksenhet En enhet med innebygd nettverkskort som gjør at enheten kan kommunisere med andre nettverksenheter på nettverket. 40

nettverksområde Et definert område innenfor et lokalnett eller nettverk som skiller seg fra andre deler av nettverket. 39, 40

nettverksutstyr Ruterene og svitsjene som blir benyttet av studentene i Cisco-lab, disse er Cisco 2901 ISR, Cisco 4221 ISR, Cisco Catalyst 3650 og Cisco Catalyst 2960. xi, xiii, xvii, xviii, xxi–xxiii, xxv, xxvi, 1, 2, 5–8, 11–14, 16–22, 24–28, 31–33, 35–38, 40, 41, 43–46, 49–53, 56, 57, 59–66, 91, 102, 114, 121

open-source Open-source er det engelske ordet for åpen kildekode, og er data-programvare som er utgitt under en lisens der opphavsrettsinnehaveren gir brukerne rettigheter til å bruke, studere, endre og distribuere programvaren og kildekoden til alle og for ethvert formål. 51

operativsystem Grunnleggende programvare for å få et datasystem til å fungere. xxii, xxiii, xxvi, 17, 25, 33, 59

Playbook En organisert struktur med tasks som definerer konfigurasjondata med automatiseringsverktøyet Ansible. xi, xiii, xvii, xxiii, xxiv, 17, 18, 49, 51–53, 56–62, 64, 87

PnP Er en offisiell standard som gjør det mulig å oppdage en maskinvarekomponent i et system uten en fysisk enhetskonfigurasjon og brukerinteraksjon. 22, 31

pod En serverrack med nettverksutstyr. xxii, 1, 2, 44, 46, 48, 59–61

protokoll Fremgangsmåte og formater som gir regler for kommunikasjon mellom IT-utstyr [8]. xxii, xxvi, 28

Puppet Et konfigurasjonsstyringsverktøy som tar i bruk agenter på administrerte nodene avsnitt 2.5.3. xiii, 16, 19, 20

Python Et objekt-orientert programmeringsspråk beregnet på utvikling av enkle verktøy og applikasjoner. xiii, 17, 23–25, 27, 32, 33, 35, 40, 58

rollover-kabel En type nullmodem kabel[9]. Den er relativt lik en vanlig ethernet kabel, der en av forskjellene er at endene er speilvendt. Den brukes til å kommunisere mellom nettverksutstyr. xxi, 24

running-config En konfigurasjonsfil som er lagret i minnet til nettverksutstyret. Den oppbevarer konfigurasjonen som blir brukt av ruteren ellet svitsjen når den er skrudd på. *running-config* blir forkastet hvis enheten mister strøm eller blir restartet. 13, 14, 37, 57

Scapy Et pakkemanipuleringsverktøy, kan benyttes til å forfalske, dekode pakker, oppdage pakker og matche forespørsler og sende de videre. Verktøyet er kapabelt til å skanne nettverk, traceroute, sondering, enhetstesting, angrep og til nettverksoppdagelse. xiii, xvii, 6, 25, 26, 31–33, 38–41, 43

Scrum Et enkelt rammeverk for å optimalisere produktutvikling. Rammeverket er basert på iterativt arbeid med en fast lengde. 6, 7, 66

serverrack Et engelsk ord for en samling servere stablet oppå hverandre i et stativ. xxv, 1

skript Et program som består av en serie instruksjoner til et brukerprogram (applikasjon) eller til et hjelpeprogram. 27, 28, 38, 64

SSH Secure Shell (SSH) er en nettverksprotokoll som sørger for en kryptert og sikker kommunikasjon mellom enheter når man skal administrere nettverksutstyr [10]. xxvi, 14, 16, 18, 20, 32, 33, 35, 37, 49, 50

startup-config En konfigurasjonsfil som er lagret i minnet til nettverksutstyret, Den er den standard konfigurasjonsfilen som rutere og svitsjer bruker ved oppstart av enheten. 12–14, 16, 25, 27, 31, 32, 57, 60

task En handlingsenhet i Ansible. xvii, xxiv, 17, 49–53, 55, 57

TCL Tool Command Language (TCL) Et programmeringsspråk som kan kjøres direkte på nettverksutstyret. 16, 27, 28, 32, 33, 37

Telnet En protokoll på applikasjonslaget som tillater kommunikasjon mellom enheter når man skal administrere nettverksutstyr. Det sies å være den mindre sikre forgjengeren til SSH[11]. 24, 26

terminalserver Kobler enheter med en seriell port til et lokalnett. 31

TFTP Trivial File Transfer Protocol (TFTP) er en protokoll som er laget for overføring av filer. Protokollen er designet for å være så enkel som mulig så den bør ikke brukes til å overføre konfigurasjonsfilerer som inneholder sensitiv informasjon. xi, xxii, 12, 13, 21, 28, 33, 36, 37, 60–62

Ubuntu En linuxdistribusjon som er både fritt og åpent. Basert på Debian, men har fokus på brukervennlighet, jevnlige utgivelser og en god inngangsopplevelse for førstegangsbrukere. 58, 59

URI Uniform Resource Identifier, en unik streng som peker på en ressurs som blir brukt av nettverksutstyr. 37

VM Betegnelsen på en emulering av et gitt operativsystem. 65

Kapittel 1

Introduksjon

1.1 Bakgrunn

Norges teknisk-naturvitenskapelige universitet (NTNU) i Gjøvik har en Cisco-lab, der studenter i nettverksadministreringsemner har muligheten til å øve på konfigurering av maskinvare som rutere, lag-2-svitsjer og lag-3-svitsjer. Dette utstyret er plassert i *serverracks* som er referert til som en pod.

Nettverksutstyrets innstillinger og konfigurasjoner skal være nullstilt etter hver laboratorieøkt. I dag gjøres dette manuelt, da det er komplisert å automatisere oppdateringsprosessen av operativsystemet på disse maskinene. Dette er en meget tidskrevende prosess. I tillegg bruker forelesere og studenter mye tid på å sette opp testmiljøene for gjennomføring av laboratorieøvelser. Dette inkluderer å enten laste opp ferdige konfigurasjonsfiler til hver laboratorieøvelse, eller å koble seg opp mot nettverksutstyret og manuelt legge til den nødvendige konfigurasjonen.

NTNU ønsker nå å utforske løsninger for å forbedre disse arbeidsoppgavene. Ambisjonen med prosjektet er å finne en løsning som både kan spare forelesere og studenter tid ved forberedelse av laboratorieøvelser.

1.2 Oppgavedefinisjon

Dette prosjektet skal utforske ulike løsninger for å automatisere oppgraderinger og klargjøring av nettverksutstyr i Cisco-laben ved NTNU i Gjøvik. For øyeblikket blir forberedelse og konfigurering av nettverksutstyr gjennomført manuelt av faglærere før eller etter undervisning. En helhetlig løsning som automatiserer oppgavene både før/etter hver laboratoriesesjon er ønskelig for oppdragsgiver. Ved å utforske mulige løsninger vil det være mulig å finne en løsning som er både effektiv og intuitiv, og dersom implementert, kan øke effektiviteten og redusere tidsbruk i forbindelse med forberedelse av Cisco-laben for forelesere og faglærere.

Hovedmålene som ble gitt av oppdragsgiver, som vist i vedlegg I:

- Se på hvilke løsninger som finnes for automatisering av oppsett/konfigurasjon av rutere, svitsjer etc. i et lab-miljø som Cisco-laben ved IIK, NTNU, Gjøvik.
- Vurdere fordeler og ulemper knyttet til de identifiserte løsningene.
- Velge den løsningen som synes å være best egnet, og implementere en prototype.
- Lage en manual som beskriver bruk av prototype for lett implementering i Cisco-laben.

Utifra samtaler med oppdragsgiver ble det bestemt at arbeidet som utføres innebærer å oppgradere nettverksutstyret ved Cisco-laben og sende konfigurasjonsfiler for labarbeid slik at studenter som skal utføre læringsarbeid kan implementere konfigurasjon selv etter behov.

Oppgaven skal utføres på de ni podene i Cisco-laben. Nettverksutstyret har mer eller mindre det samme oppsettet (for full oversikt over podene *se vedlegg* A):

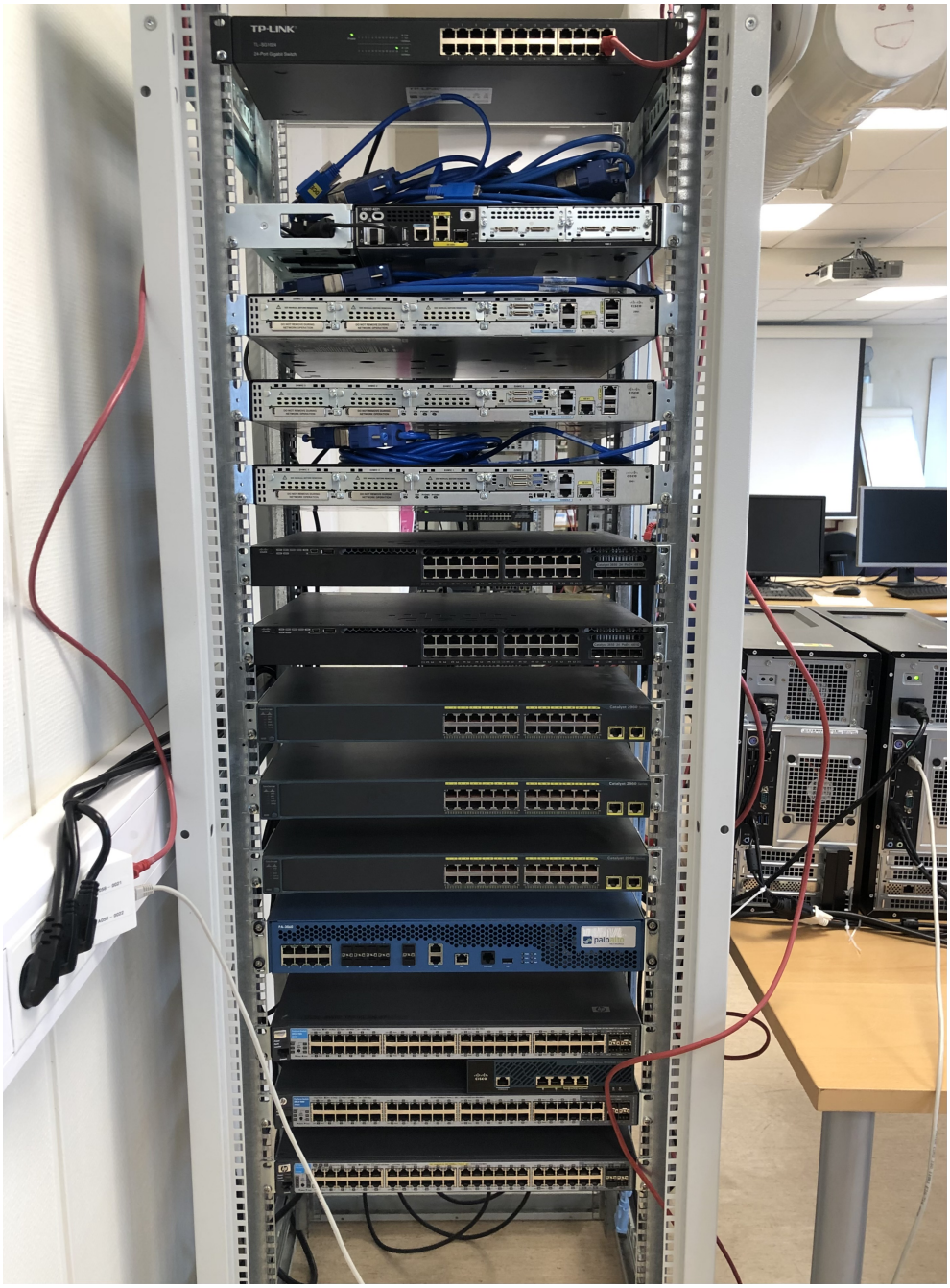
- Fire rutere, enten fire Cisco 2901 ISR eller en Cisco 4221 ISR og tre Cisco 2901 ISR.
- To lag-3-svitsjer, alle Cisco Catalyst 3650.
- Tre ordinære lag-2-svitsjer, enten Cisco Catalyst 2960 eller Cisco Catalyst 2960 Plus.
- En lag-2-svitsj som resten av nettverksutstyret i poden kan koble seg til med Ethernet-kabel for å få tilgang til internettet. Disse refereres til som en internett-svitsj i denne oppgaven.

1.3 Avgrensninger

Arbeidet skal utføres på Cisco-laben ved NTNU i Gjøvik som vist på figur 1.1. Arbeidet skal utføres på ni poder avbildet i figur 1.2. Hovedoppgaven til prosjektet er å kartlegge mulige metoder for å effektivt oppdatere og sende konfigurasjonsfiler til nettverksutstyret på Cisco-laben. Nettverksutstyret består av ruterene Cisco 4221 ISR og Cisco 2901 ISR, lag-3-svitsjmodellen Cisco Catalyst 3650 og lag-2-svitsjmodellene Cisco Catalyst 2960 og Cisco Catalyst 2960 Plus. Oppgaven er avgrenset til å oppdatere og sende konfigurasjonsfiler til nevnte modellvarianter. Av den grunn er fokuset i prosjektet å finne den beste løsningen for akkurat disse modellene. Det skal også tas høyde for at nettverksutstyret kan bli byttet ut i fremtiden.

I tillegg finnes det internett-svitsjene TLS-SG1024, som vil bli brukt i oppgaven for å koble resten av nettverksutstyret til Internett. Denne lag-2-svitsjen er ikke administrert og vil ikke være mulig å modifisere innstillingene på, og dermed ligger internett-svitsjen utenfor omfanget av dette prosjektet.

Figur 1.1: Cisco-laben i A-bygget på NTNU i Gjøvik



Figur 1.2: Testmiljø: Eksempel på en pod

1.4 Sikkerhetskrav

Testing og automatisering av konfigurasjonen av nettverksutstyret er tiltenkt å foregå internt i Cisco-laben. Vi har lagt til grunn at Cisco-laboratoriet er et avgrenset romområde med streng tilgangskontroll. Det er derfor ikke implementert ytterlige sikkerhetsprotokoller da dette anses å være unødvendig for gjennomføring av oppgaven.

Gruppen har konkludert med at gjeldende sikkerhetsprotokoller er tilstrekkelige siden;

- Cisco-laben har tilgangskontroll.
- Cisco-nettverksutstyr blir resatt når nødvendig.
- Nettverksutstyr benyttes som testmiljø i undervisningformål.

Risikoen for at nettverksutstyr blir kompromittert vil alltid være tilstede, men følgerisikoen blir ansett like lav som før oppgavestart. Følgene vil gjelde et lite antall enheter, som gjør risikoen ubetydelig. Det er derfor ikke spesifisert sikkerhetskrav i denne oppgaven. I og med at nettverksutstyret blir kontinuerlig nullstilt, er det heller ikke spesifisert sikkerhetskrav til utstyret. Av den grunn kan det benyttes protokoller med lav sikkerhet. Dersom forutsetningene hadde vært annerledes kunne strengere sikkerhetsprotokoller vært nødvendig.

1.5 Målgruppe

Målgruppen for denne oppgaven er hovedsakelig forelesere som anvender Cisco-laben på NTNU i Gjøvik som undervisningslokale. Implementering av prototypen er skreddersydd for anvendelse i Cisco-laben ved NTNU i Gjøvik. Andre aktører som ikke har mulighet til å oppbevare en fast konfigurasjon av nettverksutstyr, kan dra nytte av denne oppgaven som inspirasjon for implementering av egen løsning.

Forventede forkunnskaper er generell forståelse av Linux-økosystemet. Det er samtidig forventet at målgruppen har grunnleggende kunnskap om versjonskontroll med Git, og om nødvendige kommandoer for opplasting av filer til mappestrukturen. Kunnskap om nettverk og programmering er et kriterium, da det er her oppgavens fokus skal ligge. En grunnleggende forståelse av Python som programmeringsspråk er forventet.

1.6 Prosjektmål

Prosjektmålene består av tre underkategorier: *effektmål*, *resultatmål* og *læringsmål*. *Effektmål* er det oppdragsgiver har som mål å få ut av oppgaven. *Resultatmål* handler om hva gruppen ønsker å levere. Gruppens ambisjoner om tilegning av kunnskap anses som *læringsmål*.

1.6.1 Effektmål

Effektmålet for denne oppgaven er å automatisere oppdateringen og overføringen av konfigurasjonsfiler til nettverksutstyr i Cisco-laben. Automatisering vil føre til mer effektiv tidsbruk for forelesere og studenter i Cisco-laben, dersom denne løsningen blir utviklet på en hensiktsmessig og intuitiv måte.

1.6.2 Resultatmål

Resultatmålet for oppgaven er å utvikle en prototype av løsningen som oppdragsgiver kan implementere. I tillegg skal gruppen skrive en manual for denne prototypen. Målet er at resultatet skal være godt nok til å bli tatt i bruk av oppdragsgiver, og samtidig gi et solid utgangspunkt for å bygge videre på.

1.6.3 Læringsmål

Læringsmålet er å få en større forståelse for arbeid med nettverksutstyr, øke kompetansen rundt automatisering av arbeidsoppgaver og få praktisk erfaring med nettverksinfrastruktur.

1.7 Gruppens forkunnskaper

Oppgaven er gjennomført av en gruppe på fire NTNU-studenter som går på linjen Digital Infrastruktur og Cybersikkerhet ved NTNU i Gjøvik [12]. Gruppen deler interesse for nettverksadministrering og automatisering, som er svært relevant for denne oppgaven. I tillegg har gruppen relevante forkunnskaper og erfaringer fra tidligere emner i studiet (se tabell 1.1). Dette inkluderer temaer som nettverksadministrering, systemutvikling, programmering, Linux, Scapy og Scrum.

Tabell 1.1: Relevant kompetanse

Emnekode	**Navn på emnet**
DCSG1001	Infrastruktur: grunnleggende ferdigheter
DCSG1002	Cybersikkerhet og teamarbeid
PROG1001	Grunnleggende programmering
DCSG1005	Infrastruktur: sikre grunntjenester
DCSG1006	Datakommunikasjon og nettverk
PROG1003	Objektorientert programmering
PROG1004	Programvareutvikling
DCSG2001	Sammenkoblede nettverk og nettverkssikkerhet
DCSG2003	Robuste og skalerbare tjenester
IMT4116	Reverse Engineering and Malware Analysis
IIKG3021	Campusnettverk og Internettarkitektur
IIKG3005	Infrastructure as Code

1.8 Rammer

Det er bestemt at rapporten skal skrives på norsk ettersom publikum og oppdragsgiver er norske, og for at oppgavens innhold og sammenheng skal være lettere å forstå. Språklige nyanser kan spille en rolle i forståelsen av innholdet. Bacheloroppgaven skal bli skrevet i LaTeX på den skybaserte tjenesten Overleaf, etter en mal gitt av Norges teknisk-naturvitenskapelige universitet (NTNU). Referanselisten benytter stilen IEEE som følger med LaTeX malen for bacheloroppgaver på NTNU. Diagrammer er laget på diagrams.net og i Cisco Packet Tracer.

Det er ønsket at løsningen skal implementeres i Cisco-laben i Gjøvik. Løsningen skal bestå av et helhetlig program som skal eksekveres i forkant av undervisning for å konfigurere nettverksutstyret. All testing og utvikling vil foregå av en datamaskin og nettverksutstyr som er tilgjengelig i Cisco-laben. Oppdragsgiver stiller seg åpen til at andre universiteter og privatpersoner skal ha mulighet til å få tilgang til kildekoden, men det er forventet at de må påberope seg tilpasning av kildekoden for å få det til å fungere som tiltenkt.

Prosjektet ledes gjennom iterasjoner med Scrum. Scrum er et rammeverk for å utvikle, levere og opprettholde produkter i et komplekst miljø. I en liten gruppe på fire gruppedeltakere vil rammeverket være til hjelp for å opprettholde sekvensene og bryte ned arbeidet i små mål som skal gjennomføres innenfor tidsbokser som kalles en sprint. Fremgangen i teamet vil bli vurdert kontinuerlig i form av daglige møter på 15 minutter i begynnelsen av arbeidsøkten. Hver uke vil det bli gjennomført et Scrum-møte for å få et overblikk over progresjonen, og for å undersøke i hvilken grad de fastsatte målene er blitt nådd.

Bacheloroppgaven skal leveres 20.05.2022. Underveis lages iterasjoner av prosjektrapporten for å definere klare mål underveis i gjennomføringen av prosjektet. Etter oppgaven er levert skal det holdes en presentasjon av sluttresultatet.

Ferdig forprosjekt

Dato: 31.01.2022

Iterasjon en av prosjektrapport

Dato: 08.04.2022

Iterasjon to av prosjektrapport

Dato: 02.05.2022

Siste iterasjon av prosjektrapport

Dato: 20.05.2022

Ferdig prototype

Dato: 02.05.2022

Presentasjon

Presentasjonen av oppgaven vil bli holdt den 7. og 8. juni 2022.

1.9 Øvrige roller

NTNU er oppdragsgiver. Universitetslektor og senioringeniør Eigil Obrestad fra Institutt for informasjonssikkerhet og kommunikasjonsteknologi (IIK) ved NTNU er kontaktperson. Akademisk veileder er førsteamanuensis Ernst Gunnar Gran fra IIK ved NTNU.

1.10 Rapportens struktur

Innledningsvis vil teknologier som er relevante for gjennomføring av oppgaven bli avdekket i kapittel 2. Det vil bli presentert en kort innføring av teknologien, bruksområde for teknologien, i hvilken grad den vil være behjelpelig med å løse oppgaven, og avslutningsvis en kort oppsummering av teknologien og om nytteverdien den vil eventuelt ha videre i prosjektet.

I kapittel 3 vil mulige løsninger på oppgaven bli presentert, inklusiv nytteverdi av teknologiene og hvordan de eventuelt kan bli implementert. Første avsnitt vil omhandle det å etablere kommunikasjon til utstyret. Andre avsnittet vil gå gjennom hva som er nødvendig for å overføre ønsket konfigurasjon til nettverksutstyr. Tredje avsnitt skal gå nærmere inn på hva som er nødvendig for å få på plass en bekreftelse på at konfigurasjonen har blitt gjennomført og satt i funksjon. Videre skal det avdekkes om teknologiene kan kombineres. Til slutt oppsummeres de valgte teknologiene, og med begrunnelse for hvorfor de er valgt, og hvordan de kan implementeres.

I kapittel 4 vil prototypen bli presentert. Dette kapittelet vil beskrive prosessen med å utvikle prototypen, og anvendelsen av teknologiene i praksis. Kapittelet vil først gå gjennom grunnkonfigurasjonen, og forutsetningene for at prototypen skal kunne gjennomføres. Videre gjennomgås hva som skal til for å identifisere nettverksutstyr, hvordan kommunikasjonen skal opprettes med nettverksutstyret samt relevante kommandoer for gjennomføring av prototypen. Til slutt vil det bli oppsummert hvordan teknologiene er blitt kombinert for å skape en helhetlig løsning med testdata.

I kapittel 5 vil testmiljøet, testmetodikken og testresultatene bli gjenomgått. I kapittel 6 vil det lages en konklusjon av arbeidet, og i hvilken grad prototypen inneholder de elementene gruppen hadde planlagt å implementere. Videre vil det vurderes om resultatmålene ble oppnådd, og nevne eventuelt videre arbeid som

er relevant for å forbedre prototypen. Til slutt vil gruppen reflektere over arbeidet som er gjort og i hvilken grad målene for oppgaven er realisert.

Kapittel 2

Teknologier

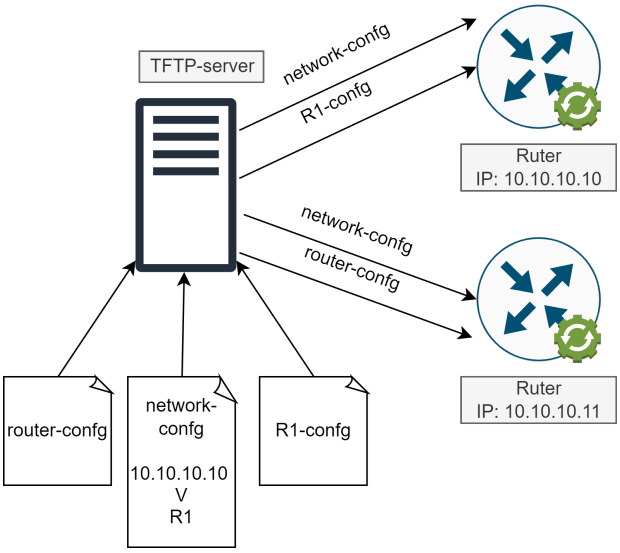
Dette kapittelet vil ta for seg de teknologiene som er blitt utforsket på under prosjektet. Teknologiene i kapittelet vil bli vurdert utifra hvor stor nytteverdi de vil ha i den endelige løsningen. Det ble bestemt å kaste et vidt nett for å utforske ulike løsninger. De nevnte teknologiene vil dekke et bredt spekter, alt fra maskinvare, eventuelle programmer og innebygde funksjonaliterer i nettverksutstyret etc..

Liste over teknologier:

- 2.1 Cisco Autoinstall
- 2.2 CONFIG_FILE
- 2.3 Dynamic Host Configuration Protocol (DHCP)
- 2.4 Embedded Event Manager (EEM)
- 2.5 Konfigurasjonsstyringsverktøy
- 2.6 Konsollserver
- 2.7 Plug And Play (PnP)
- 2.8 Python
- 2.9 Reverse Telnet
- 2.10 Scapy
- 2.11 Secure Shell (SSH)
- 2.12 Tool Command Language (TCL)
- 2.13 Trivial File Transfer Protocol (TFTP)
- 2.14 Universal Serial Bus (USB)

2.1 Cisco AutoInstall

Cisco AutoInstall er et verktøy utviklet av Cisco og er inkludert i IOS-versjon 12.4 (1) og nyere [13]. AutoInstall tillater nettverksutstyret å automatisk hente en konfigurasjonsfil fra en TFTP-server hvis den ikke har en *startup-config*. AutoInstall har den fordelen at den kan benyttes til å gi alt nettverksutstyret en felles konfigurasjon, alternativt er det mulig å sende individuelle konfigurasjonsfiler til nettverksutstyret.
Som vist i figur 2.1 begynner prosessen med at nettverksutstyret laster ned *network-confg*-filen fra en TFTP-serveren, i dette eksempelet er IP-adresseen `10.10.10.10` koblet til R1 og dermed vil ruteren med IP-adressen `10.10.10.10` laste ned *R1-confg*. Den andre ruteren har ikke et navn koblet til IP-adressen sin og vil dermed laste ned *router-confg* som inneholder IOS-konfigurasjonskommandoer.



Figur 2.1: Autoinstall: AutoInstall i praksis

2.1.1 Avhengigheter og synergier

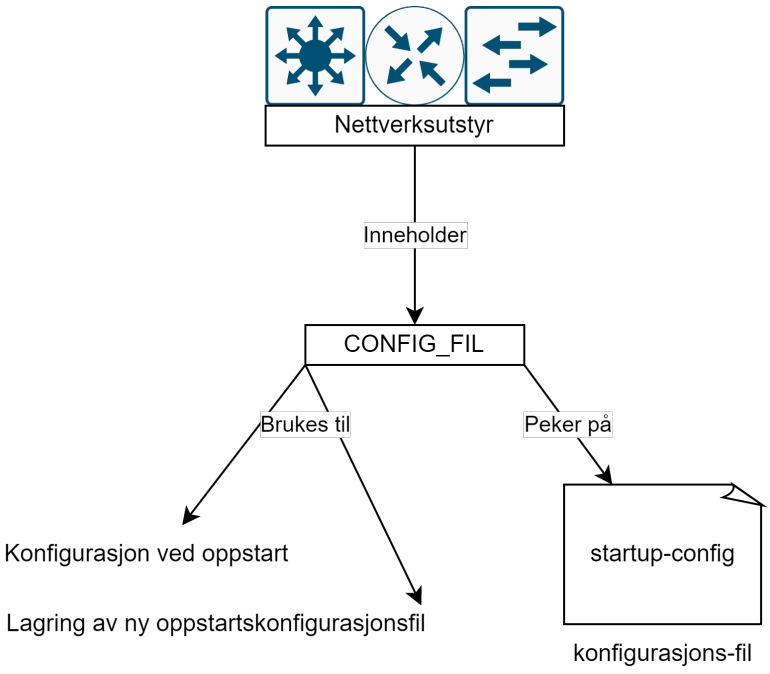
For at AutoInstall skal fungere må nettverksutstyret få tildelt IP-adresser. I Cisco-laben er det allerede satt opp en DHCP-server som kan gjøre dette, men andre alternativer er Bootstrap Protocol (BOOTP) og Serial Line Address Resolution Protocol (SLARP) [13]. Siden AutoInstall kan sende en konfigurasjonsfil automatisk til nettverksutstyret kan den kombineres med mer tradisjonelle automatiseringsverktøy som ofte krever en grunnleggende konfigurasjon på nettverksutstyret. I denne rapporten blir tre relevante teknologier nevnt; konfigurasjonsstyringsverktøy (avsnitt 2.5), TCL (avsnitt 2.12) og Python (avsnitt 2.8).

2.1.2 Vurdering av teknologi

Cisco AutoInstall er et veldig kraftig verktøy som kan anvendes for å få nettverksutstyret til å automatisk hente konfigurasjonsfiler fra en TFTP-server. Siden Cisco AutoInstall kjører når det ikke er noen konfigurasjonsfiler på nettverksutstyret, er det en veldig fin løsning for å laste inn en grunnleggende konfigurasjon som andre teknologier kan bruke som et utgangspunkt. Cisco AutoInstall har imidlertid noen utfordringer. Ett problem er at den bare kjører når det ikke finnes konfigurasjonsfiler på nettverksutstyret, som betyr at *startup-configen* alltid blir slettet etter bruk for at AutoInstall skal kunne kjøre problemfritt neste gang. Basert på forelesers tidligere erfaring glemmer studenter ofte å slette *startup-configen*.

2.2 CONFIG_FILE

Rutere og svitsjer har en *konfigurasjonsfil* som kjøres ved oppstart av nettverksutstyret, *startup-config*. CONFIG_FILE er variabelen som bestemmer hvilken *konfigurasjonsfil* som skal brukes ved oppstart [14]. Filen CONFIG_FILE peker på blir også overskrevet ved lagring av nåværende konfigurasjon fra running-config til startup-config. Dette er illustrert i figur 2.2.



Figur 2.2: CONFIG_FIL: CONFIG_FIL forteller maskinen både hvilken fil å bruke ved oppstart, og fil overskrives ved lagring

2.2.1 Avhengigheter og synergier

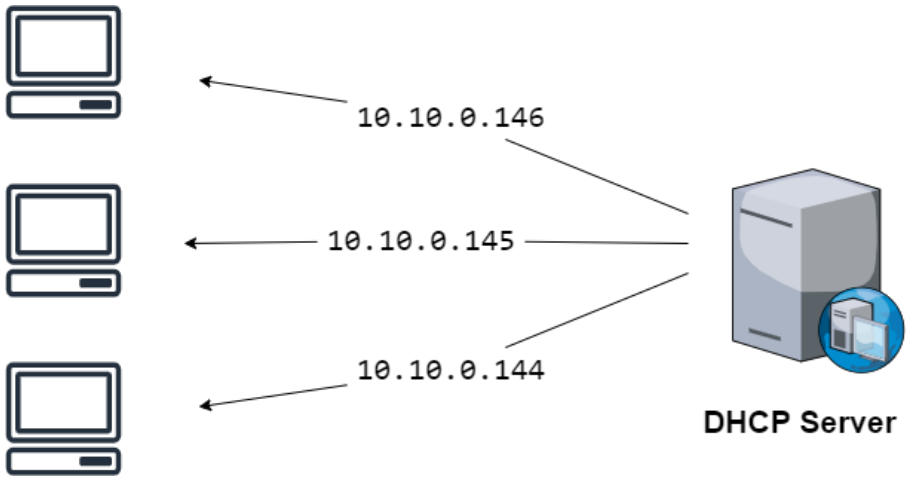
CONFIG_FILE er fabrikkinstallert på alt nettverksutstyr på Cisco-laben. Nettverksutstyret er derfor ikke avhengig av andre teknologier for å modifisere denne variabelen.

2.2.2 Vurdering av teknologi

En løsning som ble utforsket var å endre CONFIG_FILE-variabelen fra å peke på startup-config til en ny egendefinert konfigurasjonsfil i flashminnet. Tanken var at denne konfigurasjonsfilen skulle inneholde et script som opprettet SSH-konfigurasjonen, og deretter fikk ruteren til å sette opp alt som eksisterte på *startup-config*. Studenter vil da ha ubegrenset adgang til å slette og endre på startup-config, samtidig som SSH-oppsettet ikke blir berørt. Det viste seg at CONFIG_FILE-variabelen også brukes til å fortelle ruteren hvilken fil som skal overskrives når man lagrer running-config i startup-config. Dermed vil den egendefinerte konfigurasjonsfilen i flashminnet som oppbevarer SSH-oppsettet bli overskrevet eller endret når running-config blir lagret. Derfor nytter det ikke å bruke CONFIG_FILE i med den tiltenkte løsningen.

2.3 DHCP

Dynamic Host Configuration Protocol (DHCP) er en tjeneste som er satt opp i de fleste nettverk. Som vises i figur 2.3 tildeler DHCP en IP-adresse automatisk til en enhet, uten at denne trenger å være forhåndskonfigurert med IP-adresse, subnettmaske, gateaway og en DNS-adresse. Tildelingen av IP-adressene ved benyttelse av denne tjenesten kalles derfor for dynamisk IP-adresse. I Cisco-laben vil nettverksutstyret automatisk få tildelt en IP-adresse ved tilkobling til internett-svitsj øverst i hver pod fra en DHCP-server.



Figur 2.3: DHCP: Tildeling av IP-adresser med en DHCP-server

2.3.1 Avhengigheter og synergier

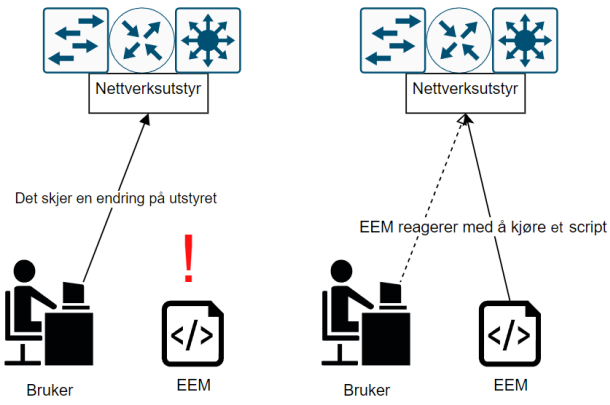
For at DHCP skal fungere trenger den en ruter eller en dedikert server til å kjøres på. Den trenger også en gruppe med IP-adresser å dele ut.

2.3.2 Vurdering av teknologi

DHCP er en nyttig teknologi for å automatisk tildele IP-adresser. Det finnes allerede en dedikert DHCP-server på Cisco-laben, der nettverksidentifikatorer (IP-adresse og MAC-adresse) vil ligge oppdatert til enhver tid, noe som er en fordel for utviklingen av prototypen.

2.4 Embedded Event Manager

Embedded Event Manager (EEM) er en programvarekomponent som er installert på en rekke Cisco-svitsjer og rutere [15]. Den vil bli konfigurert på den måten at den reagerer på fastsatte hendelser som oppstår på maskinen, og utfører automatiserte prekonfigurerte kommandoer når dette oppstår. Den kan for eksempel benyttes til å forsikre at et grensesnitt ikke kan bli manuelt avskrudd av en bruker. Skrur de den av registrerer EEM hendelsen og kjører kommandoene som er nødvendige for å få den opp igjen. En illustrasjon av dette finnes i 2.4.



Figur 2.4: EEM: EEM gjennomfører en samling kode når det den er satt til å reagere på skjer

2.4.1 Avhengigheter og synergier

Det er mulig å lage enkle EEM konfigurasjoner med den innebygde funksjonaliteten *applets*. Hvis det er ønsket å benytte et større skript er det mer funksjonelt

å benytte programmeringspråket Tool Command Language (TCL) [16] og koble det sammen med EEM for å eksekvere denne når en gitt hendelse oppstår.

2.4.2 Vurdering av teknologi

EEM kan være nyttig for å forsikre at maskinene har en ferdig konfigurert Secure Shell (SSH)-bruker for fjernstyring til enhver tid. Dessverre har EEM to ulemper. For det første blir EEM-konfigurasjonsdata oppbevart i startup-config. EEM vil dermed bli slettet med en gang `startup-configen` blir fjernet eller overskrevet. For det andre har ikke lag-2-svitsjene Cisco Catalyst 2960 eller Cisco Catalyst 2960 Plus EEM programvaren innebygd, og kan derfor ikke bruke EEM for SSH-konfigurering. Derfor er ikke EEM et verktøy som passer som en mulig løsning for prototypen.

2.5 Konfigurasjonsstyringsverktøy

Konfigurasjonsstyringsverktøy [17] hjelper å administrere nettverksutstyr. Det finnes mange alternativer, hvorav Ansible [18], Chef [19] og Puppet utforsket i denne oppgaven. Chef og Puppet er begge agentbasert som vil si at det kjøres et program direkte på nettverksutstyret, mens Ansible er agentløst, som vil si ett program som eksekveres utenfor nettverksutstyret.

2.5.1 Ansible

Ordet Ansible ble først brukt i science fiction-litteratur i 1966 om en fiktiv gjenstand som kunne sende og motta beskjeder fortere enn lysets hastighet [20]. Ansible programvaren ble lansert i *Februar 2012* og ble beskrevet slik av dens skaper:

"*Ansible is an extra-simple Python API for doing 'remote things' over SSH. As Func, which I co-wrote, aspired to avoid using SSH and have it's own daemon infrastructure, Ansible aspires to be quite different and more minimal, but still able to grow more modularly over time.*" [21]

Siden den gang har Ansible fått over 35,000 commits fra over 3000 bidragsytere. Ansible benytter seg av følgende designprinsipper [21]:

- **Agentløs:** Ansible trenger ikke å være installert på de administrerte nodene.
- **Deskriptiv:** Skal beskrive infrastrukturen på en slik måte at den kan lett forstås av mennesker og maskiner.
- **Enkel:** Oppsettet og lærekurven for å benytte seg av Ansible skal være enkel og intuitiv.
- **Enkelt å bruke:** Det bør være det enkleste IT-automasjonsystemet å bruke.

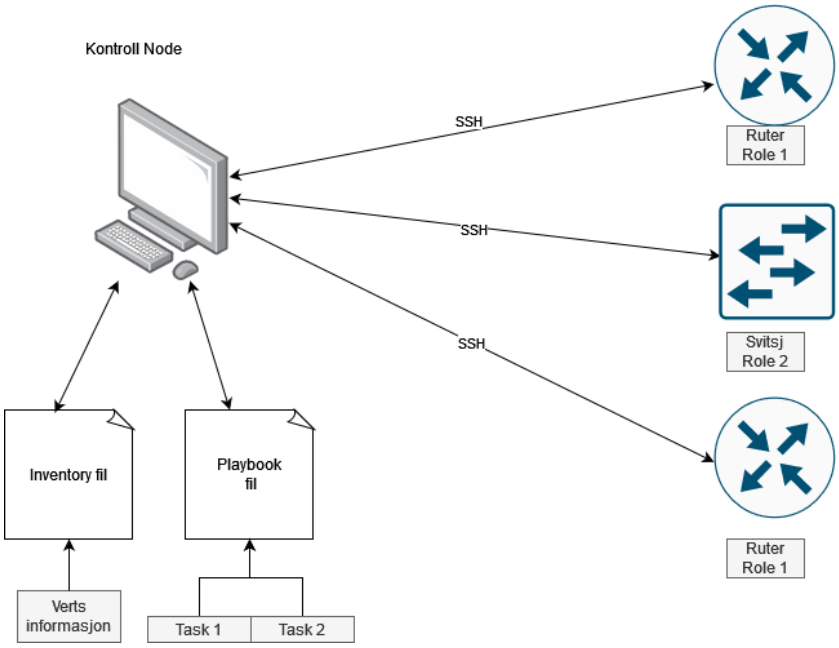
Avhengigheter og synergier

Ansible er designet for å være minimalistisk slik at man ikke er avhengig av å installere ekstra pakker på de administrerte nodene som man har mål om å konfigurere. På kontrollnoden er man avhengig av at Python er installert med nødvendige pakker. Som kontrollnode kan man ikke benytte seg av et Windows operativsystem, men er avhenging å bruke et UNIX-basert operativsystem.

Ansible benytter seg av kolleksjoner, som muliggjør anvendelse av programvareutvidelser på de forskjellige systemene så at man fort kan komme i gang med automasjonsprosessen. For å kunne komme fort i gang med konfigurasjon av nettverksutstyret kan man ta i bruk en Cisco IOS-kolleksjon [22].
Noen av nøkkelkomponentene i bruk av Ansible er som følger [23]:

- **Inventar:** Konfigurasjonsfil som benyttes for å definere informasjon om de administrerte nodene som skal benyttes for å kjøre Ansible.
- **Playbook:** Playbook innholder instruksjonene som Ansible skal benytte seg av for å konfigurere, rulle ut og administrere noder som er lokalisert i inventaret.
- **Tasks:** Tasks er spesifikke handlinger som blir utført i en Playbook.
- **Variabler:** Verdier som kan variere basert på hvilken vert som en task skal utføres på.
- **Roles:** Roller er hierarisk bestemt slik at man kan ha subtasks basert på forskjellige verdier, som type utstyr, lokasjon av utstyret m.m.

Ansible-programvaren er installert på kontrollnoden som gjør det mulig å ta i bruk Playbooken og inventarfilen for å gjøre konfigurasjonsendringer på administrerte nodene. Figur 2.5 viser hvordan kontrollnoden benytter seg av informasjon om de administrerte nodene den får av inventarfilen til å kjøre en Playbook med tasks som konfigurerer administrerte noder.



Figur 2.5: Ansible: Kontrollnode som bruker inventarfilen og en Playbook på to rutere og en svitsj

Vurdering av teknologi

Ansible trenger bare SSH konfigurert på de administrert nodene for å raskt opprette kommunikasjon med nettverksutstyret i Cisco-laben. Ansible er nyttig som et verktøy da den er kompatibel med Cisco-nettverksutstyr. Ved bruk av kolleksjoner vil man få satt opp konfigurasjonene som er i dette prosjektet på en oversiktlig måte. Man kan lett overføre filer, og sende kommandoer til Cisco-systemer så lenge man har IP-adresser og SSH konfigurert.

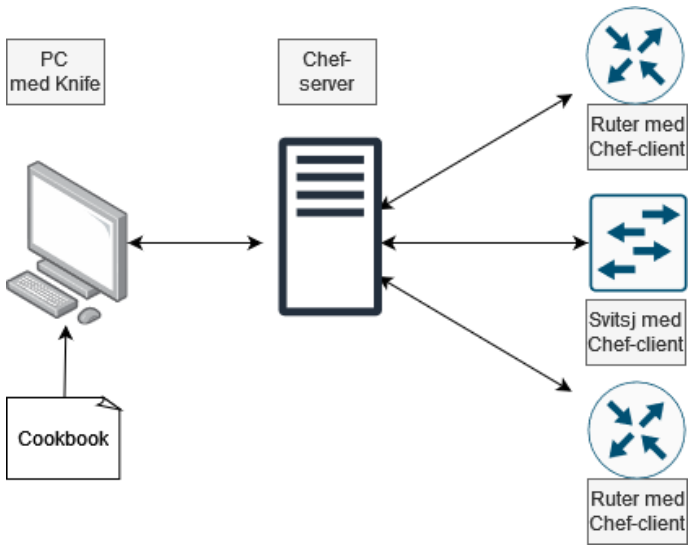
2.5.2 Chef

Chef er et konfigurasjonsstyringsverktøy som gjør det mulig med automatisering av konfigurasjon og utrulling av infrastruktur [23]. Chef er klient/tjener-basert programvare som sørger for at konfigurasjonen blir administrert fra en server, der klienter kan utføre tjenester ved å hente arbeidsoppgaver fra serveren ved hjelp av en agent installert på nettverksutstyret.

Avhengigheter og synergier

Chef er avhengig av at man har en agent som kjører på de administrerte nodene, kalt Chef-klient, som tar imot kommandoer fra en Chef-server [23]. Kommunikasjonen mellom Chef-klient og Chef-server er avhengig av kryptonøkler for å fungere. Chef benytter *cookbooks* eller kokebøker som er en kolleksjon med kom-

ponenter som brukes på en server [19]. For å bruke Chef er man avhenging av at man setter opp en Chef-server som brukes som et sentralt register som redistribuerer kokebøkene til Chef-klientene, se 2.6. For å kommunisere med Chef-serveren bruker man Kommandolinjeverktøyet `Knife`. For at nodene skal kunne ta imot kokebøker fra Chef-serveren må Chef-klientagenten være tilstede. Figur 2.6 viser hvordan en PC sender en kokebok til de administrerte nodene til nettverksutstyret med Chef-klient installert.



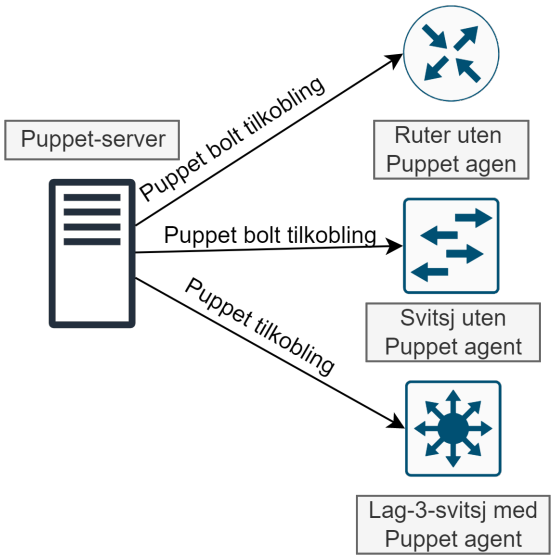
Figur 2.6: Chef: Chef-server som kommuniserer med administrerte noder og tar imot kokebok

Vurdering av teknologi

Chef er ikke passende for denne oppgaven fordi den har behov for en agent på nettverksutstyret for å fungere. Konfigurasjonene på nettverksutstyret blir kontinuerlig slettet og rekonfigurert, så denne agenten vil bli slettet fortløpende.

2.5.3 Puppet

Puppet er et konfigurasjonsstyringsverktøy som har mange fellestrekk med Chef. Den har en sentral server som lagrer konfigurasjoner [24]. Puppet-agenter som er installert på nettverksutstyret vil periodisk laste ned konfigurasjonene fra en sentral Puppet-server slik som Chef [25]. Puppet har også en agentløs versjon som kalles Bolt [26]. I figur 2.7 vises hvordan Puppet kommuniserer med nettverksutstyret.



Figur 2.7: Puppet: Samhandling mellom Puppet-server og nettverksutstyr

Avhengigheter og synergier

En vesentlig ulempe er at Puppet-agenten kjører bare på nettverksutstyr som benytter Linux [27]. Nettverksutstyret i Cisco-laben er ikke Linux-basert med unntak av Cisco Catalyst 3650 [28], men Puppet har en mulig løsning på dette. Denne løsningen heter Puppet Bolt som har likhetstrekk med Ansible, da den *dytter* konfigurasjoner fra den sentrale Puppet-serveren. Hvis det allerede er en Puppet-basert infrastruktur går det an å utvide den til å håndtere nettverksutstyret, dersom dette ikke er tilfellet er det mest anvendelig å bruke Ansible som er designet for en push-basert konfigurasjon.

Vurdering av teknologi

Siden Puppet primært er er et agentbasert verktøy betyr det at mange av fordelene med å bruke *agents* forsvinner når det bare er mulig å bruke Puppet Bolt. Siden Puppet ikke kan bli fullstendig utnyttet av nettverksutstyret, vil ikke teknologien kunne anvendes som en del av løsningen i oppgaven.

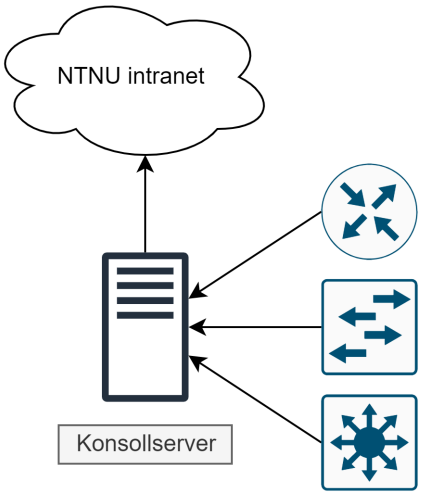
2.5.4 Sammenligning av Ansible, Chef og Puppet

Både Puppet og Chef er avhengig av å starte med installasjon av agenter for å kjøres på de administrerte nodene [23]. Både Puppet og Chef benytter seg av at nodene må hente konfigurasjon fra en Chef/Puppet-server. Med Ansible dyttes konfigurasjon til de administrerte nodene fra en kontrollnode. Siden man ikke trenger å installere noe annet enn SSH på nettverksutstyret, resulterer det i at

Ansible vil være det mest praktiske alternativet for denne oppgaven, særlig fordi den er agentløs.

2.6 Konsollserver

En konsollserver tillater brukere å koble konsollportene på nettverksutstyret til en server, som er koblet opp til internett eller et lokalt nettverk. Dette gjør det mulig å konfigurere alt nettverksutstyret direkte fra ett punkt [29], som vist på figur 2.8.



Figur 2.8: Konsollserver: Kommunikasjon med Konsollserver

2.6.1 Avhengigheter og synergier

For at denne løsningen skal fungere må det legges opp konsollkabel til alle de 81 rutere og svitsjer i Cisco-laben. Konsollserveren er ikke avhengig av andre teknologier eller protokoller for å kommunisere med nettverksutstyret, i motsetning til Cisco AutoInstall (se avsnitt 2.1), som krever DHCP og en TFTP-server. Men det må lages en løsning for å håndtere den faktiske konfigurasjonen av nettverksutstyret siden konsollserveren bare fungerer som en tilkobling til nettverksutstyret.

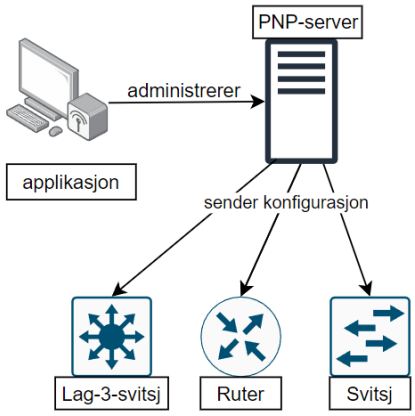
2.6.2 Vurdering av teknologi

Den største fordelen med en konsollserver er at den gir en veldig pålitelig tilkobling til nettverksutstyret. Dette kan være behjelpelig i de tilfeller konfigurasjonsfilene ikke blir slettet. Løsningen vil også ta i bruk den ene konsollporten til nettverksutstyr. En ulempe med teknologien er at studenter ikke kan konfigurere nettverksutstyr når konsollserveren er tilkoblet. I tillegg er det viktig å poengtere

at konsollservere er kostbare, spesielt siden dedikerte konsollkabler må legges til nettverksutstyr i Cisco-laben.

2.7 PnP

Plug And Play (PnP) er en innebygd programvarekomponent i Cisco-rutere og svitsjer. Den kan være til nytte for å automatisere utrulling av nytt nettverksutstyr [30]. Den tilbyr en alternativ måte å konfigurere nettverksutstyr som ikke har en eksisterende konfigurasjonsfil [31]. For å få den til å fungere trenger man en applikasjon som kan skape og kjøre en PnP-server. Fra den kan man laste opp IOS-imager og konfigurasjonsfiler man ønsker å rulle ut på maskinene. I neste steg kobler en til fabrikkinnstilt nettverksutstyr inn i eksisterende nettverk. Da vil den oppdage PnP-servere og utstyret vil automatisk motta en prekonfigurert konfigurasjonsfil. Figur 2.9 illustrerer i grove trekk hvordan PnP er satt opp.



Figur 2.9: PnP: Hvordan PnP er satt opp

2.7.1 Avhengigheter og synergier

For at PnP skal fungere er det nødvendig å koble den til en PnP-server. Den har oversikt over tilkoblede maskiner og kan sende nye imager og konfigurasjoner via applikasjonen [31]. Det er også nødvendig at nettverksutstyret har programvarekomponenten. Ettersom Cisco 2901 ISR er en eldre modell mangler den denne funksjonen. Den blir ikke vedlikeholdt av Cisco og vil derfor ikke ha denne funksjonaliteten i fremtiden.

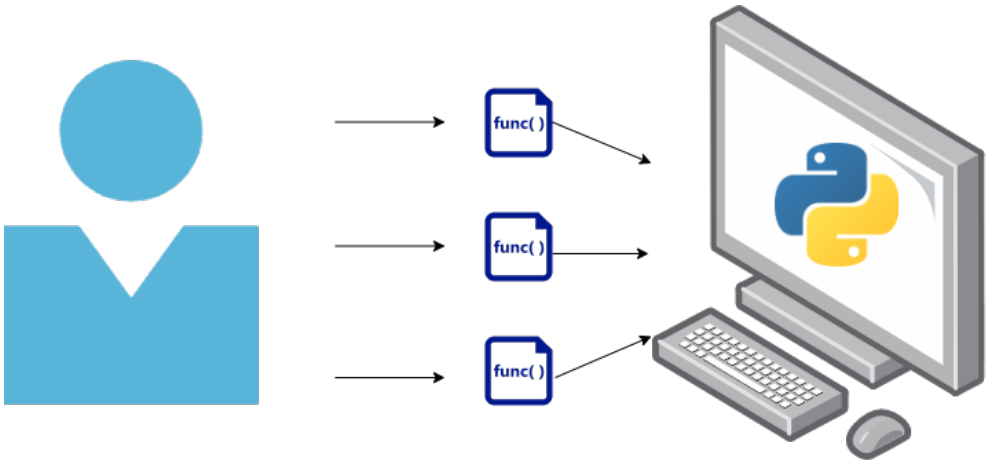
2.7.2 Vurdering av teknologi

I teorien kan PnP være en svært relevant teknologi for å løse problemstillingen i oppgaven, ettersom all konfigurasjon blir fjernet fra nettverksutstyret etter bruk.

Dessverre er Cisco 2901 ISR ikke kompatibel med denne teknologien. Cisco 2901 ISR er den mest utbredte ruteren i Cisco-laben, og dermed kan ikke denne teknologien benyttes for å løse oppgaven i sin helhet.

2.8 Python-skript

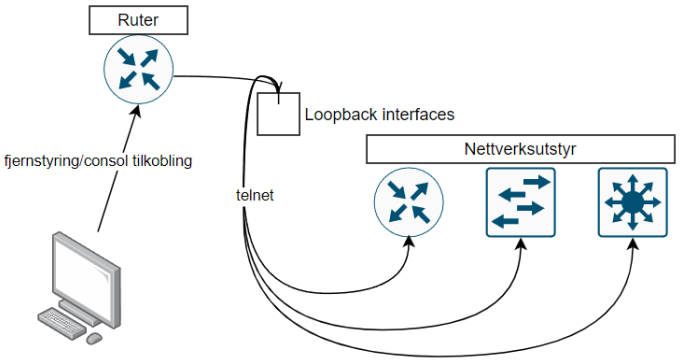
Python er et programmeringsspråk som ble utviklet av Guido van Rossum i 1989 [32]. Programmeringsspråket ble utviklet for å skape det distribuerte operativsystemet Amoeba 809 for å lage systemendringer. Selve navnet Python ble valgt av grunnleggeren da han var fan av den britiske 'komiker-truppen' Monty Python. Programmeringsspråket blir ansett som en nyere, men mer striglet versjon av programmeringspråket Perl, og blir i hovedsak benyttet til å automatisere og skrive kodesnutter som utfører en funksjon ved bruk av objektorientert programmering [33].



Figur 2.10: Python: Illustrasjonsbilde Python

2.8.1 Avhengigheter og synergier

En av fordelene med å benytte seg av et Python-skript er at kodespråket er lett å lese og forstå. Ikke minst er det lett å tilpasse kode senere ved behov, noe som er en stor fordel dersom en vil gjøre både større og mindre endringer på koden i ettertid. Python vil også være behjelpelig med å lage en struktur over sekvenser og automatiseringsoppgaver i oppgaven, noe som kommer godt med i et utviklingsprosjekt. Som funksjon i oppgaven kan Python bli benyttet til å sette opp enklere administrering av konfigurasjonsfiler for laboratorieøvelser i oppgaven.



Figur 2.11: Reverse Telnet: Ruteren kobler seg opp mot nettverksutstyr ved å først bruke Telnet gjennom Loopback-adressen

2.8.2 Vurdering av teknologi

Python er et oversiktlig og anvendelig programmeringsspråk, og vil være behjelpelig med å løse en rekke automatiseringsoppgaver. Et eksempel på en automatiseringsprosess er å *flushe* og oppdatere konfigurasjonen til svitsjene. Som resultat kan de tilpassede konfigurasjonsfilene bli overført til nettverksutstyret fra en ekstern datamaskin. Av denne grunn er det sannsynlig at teknologien vil ha en sentral rolle i automatiseringsarbeidet i den endelige løsningen.

2.9 Reverse Telnet

Reverse Telnet er en funksjon av Telnet som tilbyr en alternativ måte å fjernstyre nettverksutstyret på [34]. Med Reverse Telnet skal en rollover-kabel kobles til mellom AUX-porten på Cisco 2901 ISR til konsollporten på nettverksutstyret som skal konfigureres. I kilde-ruteren setter man deretter Telnet på AUX-port-grensesnittet. Deretter lager man et loopback-grensesnitt og setter en IP-adresse på den. Som vist i figur 2.11 bruker man til slutt Reverse Telnet til å fjernstyre nettverksutstyr via Telnet.

2.9.1 Avhengigheter og synergier

For å bruke Reverse Telnet er man avhengig av at nettverksutstyr man ønsker å fjernstyre fra faktisk har en AUX-port. Dette har kun rutere i Cisco-laben. Det er også nødvendig med en rollover-kabel.

2.9.2 Vurdering av teknologi

En stor utfordring i dette prosjektet er å lage en effektiv måte å konfigurere alt nettverksutstyret i Ciscoen uten å bruke eller endre på startup-config. Med Reverse Telnet kan man koble seg til en ruter og fra den fjernstyre annet nettverksutstyr uten at det andre utstyret har Telnet konfigurert. Når man er ferdig med å konfigurere alle maskinene i poden kan man fjerne konfigurasjonen til Reverse Telnet på det nettverksutstyr som er benyttet til fjernstyring.

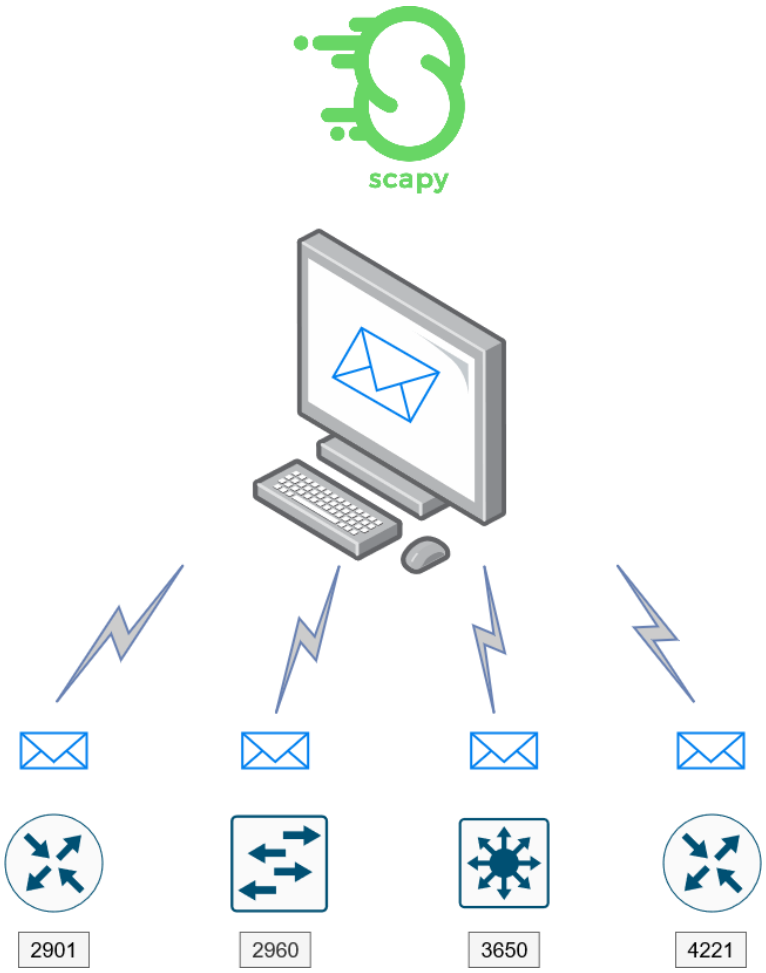
Ulempen med Reverse Telnet er at den kommuniserer gjennom AUX-porten. Denne er ikke dimensjonert for høy trafikk, og derfor vil opplasting av IOS-image være tidskrevende. For skalering til mengden nettverksutstyr i Cisco-laben kreves det i tillegg innkjøp av maskinvare-komponenter som kan medføre økte omkostninger.

2.10 Scapy

Scapy er et interaktivt pakkemanipuleringsverktøy for datanettverk, opprinnelig skrevet i Python av Phillippe Biondi [35]. Verktøyet er tilgjengelig som et pakkebibliotek for Windows og UNIX-operativsystemer som Linux, BSD, MAC OS X. Scapy har funksjoner som nettverksskanning, *tracerouting*, *probing*, *unit tests*, angrep og nettverksoppdagelse.

2.10.1 Avhengigheter og synergier

Scapy kan kjøres på Windows, Linux og andre UNIX-baserte operativsystem. Scapy har samme avhengigheter som Python, siden det er utviklet som en utvidelse av Python. Dette betyr at Python må være installert på datamaskinen for å kunne ta i bruk pakkebiblioteket. Scapy er også avhengig av å være på lokalnettverket for å samle inn nettverksinformasjon.



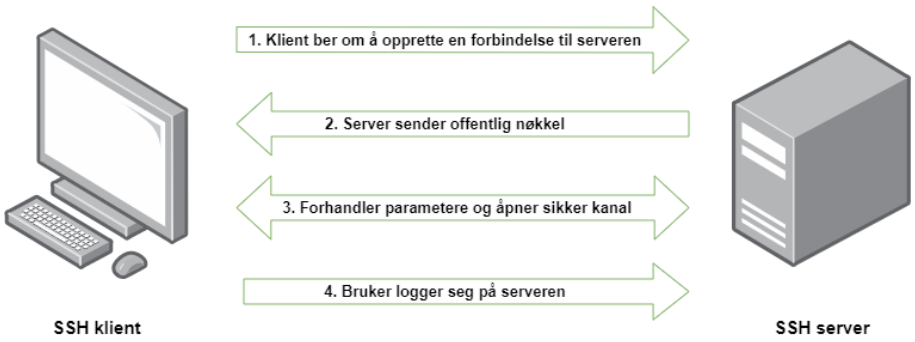
Figur 2.12: Scapy: Oppdagelse av nettverksenheter med Scapy

2.10.2 Vurdering av teknologi

I prototypen kan Scapy være nyttig som et verktøy for nettverksoppdagelse av andre enheter innenfor et spesifisert nettverksområde. Det vil da være mulig å hente ut IP-adresser og MAC-adresser for å identifisere nettverksutstyret.

2.11 SSH

Secure Shell (SSH) er en nettverksprotokoll som befinner seg på applikasjonslaget og brukes normalt for å få tilgang til en kommandolinje på en annen maskin. SSH ble designet for å ta over for Telnet, rlogin og rsh-protokollene. Uansett om SSH blir sett på som et dataprogram, regnes det også som et rammeverk for midlertidig tilkobling til andre maskiner. Figur 2.13 illustrerer stegene for fjernstyring med SSH.



Figur 2.13: SSH: Tilkobling til nettverksutstyret med SSH

2.11.1 Avhengigheter og synergier

For å kunne bruke SSH er det nødvendig å sette opp en bruker og generere en kryptografisk nøkkel til nettverksutstyr som legitimasjon. Denne konfigurasjonen er oppbevart i startup-config.

2.11.2 Vurdering av teknologi

SSH er et nyttig hjelpemiddel når man skal skape kommunikasjon mellom nettverksutstyr og en fjernstyringsmaskin. Ettersom SSH-konfigurasjon er oppbevart i *startup-config* vil den bli slettet kontinuerlig. Skal SSH bli benyttet i den endelige prototypen, er det avgjørende å finne en løsning på denne problemstillingen.

2.12 TCL

Tool Command Language (TCL) er et programmeringsspråk som blir brukt av nettverksutstyret i Cisco-laben. Siden språket kan kjøres direkte på nettverksutstyret vil et TCL-skript kunne lese informasjon fra maskinen og gjøre konfigurasjoner direkte, i motsetning til andre programmeringsspråk som Python. TCL-skripter kan eksekveres i IOS ved å bruke `tclsh`-kommandoen [16].

2.12.1 Avhengigheter og synergier

TCL er ikke avhengig av andre teknologier for å kjøre på nettverksutstyret, men for å være relevant for prototypen må den kombineres med andre teknologier for å automatisere sekvensen. Teknologier som Cisco AutoInstall, EEM eller konfigurasjonsstyringsverktøy har muligheten til å ta i bruk TCL for å eksekvere et skript direkte på nettverksutstyret.

2.12.2 Vurdering av teknologi

TCL kan ha nytteverdi for prosjektet siden den åpner muligheten for å kjøre skript på nettverksutstyret som skal konfigureres. Dette tillater den å hente informasjon

fra ruteren direkte for å kjøre spesialtilpasset konfigurasjon til nettverksutstyret individuelt. Det største problemet med TCL er at det er et relativt ukjent språk som kan gjøre vedlikeholdet eller modifiseringen av skriptet vanskelig dersom vedlikeholderen ikke har kjennskap til TCL. En annen ulempe er at det ikke er mulig å eksekvere TCL-skript fra konfigurasjonsmodusen, som gjør det vanskelig å kombinere den med teknologier som Cisco AutoInstall. Ved å ta i bruk kronjobber eller EEM [15] er det mulig å unngå dette problemet.

2.13 TFTP

Trivial File Transfer Protocol (TFTP) er en protokoll som er laget for overføring av filer. Protokollen er designet for å være så enkel som mulig, så den bør ikke benyttes til å overføre konfigurasjonsfiler som inneholder sensitiv informasjon. TFTP kan bli brukt til å overføre konfigurasjonsfiler til alt nettverksutstyr i Cisco-laben.

2.13.1 Avhengigheter og synergier

TFTP har ingen avhengigheter utenom kravet om å ha en datamaskin som er koblet til lokalnettverk i Cisco-laben. Det er imidlertid mange teknologier som er avhengig av TFTP. Cisco AutoInstall er den viktigste av disse teknologiene og vil ikke fungere uten en TFTP-server.

2.13.2 Vurdering av teknologi

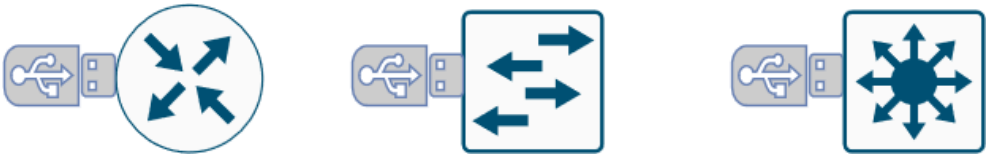
En TFTP-server vil på egenhånd ikke løse oppgaven i sin helhet, men den kan benyttes i kombinasjon med andre teknologier for å oppnå en helhetlig løsning. Cisco AutoInstall er avhengig av at det er konfigurert en TFTP-server på lokalnettverket, og av den grunn vil TFTP være relevant for oppgaven.

2.14 USB-pinne

Cisco 2901 ISR og Cisco 4221 ISR-ruterne og Cisco Catalyst 3650-svitsjen kan konfigureres ved hjelp av en Universal Serial Bus (USB)-pinne. Når nettverksadministratoren er fornøyd med gjeldende konfigurasjon kan denne lagres som en konfigurasjonsfil på USB-pinnen for senere bruk. Neste gang nettverksadministratoren eller studentene skal ta i bruk en ruter eller lag-3-svitsj, kan en sette inn USB-pinnen for å manuelt laste inn konfigurasjonen fra minnepinnen etter oppstart. Dette kan spare mye tid hvis testmiljøene skal ha lik konfigurasjon til f.eks en laboratorieøvelse.

2.14.1 Avhengigheter og synergier

En av fordelene med å overføre konfigurasjon til en ruter eller lag-3 svitsj er at man vil få fysisk oversikt over nettverksutstyret og progresjonen. En av ulempene kan være mer manuelt arbeid. I tillegg er det ikke særlig tidseffektivt å sette inn USB-pinnen og overføre konfigurasjonen manuelt til hver svitsj eller ruter illustrert i figur 2.14. Det er også lett å miste oversikt over hvilke rutere og svitsjer som allerede er blitt konfigurert, og hvilke som gjenstår. En mulig risiko tilknyttet bruk av USB-pinne er at den kan lastes inn med skadevare.



Figur 2.14: USB: USB-pinnen må settes inn i rutere og svitsjer manuelt

2.14.2 Vurdering av teknologi

Som man ser over kan USB-pinne være et brukbart hjelpemiddel i tilfelle det er snakk om å konfigurere et mindre antall enheter. Det medfører imidlertid mye manuelt arbeid, og det er lett å miste oversikt over hvilke svitsjer og rutere som allerede er konfigurert, og hvilke som mangler. Cisco Catalyst 2960 og Cisco Catalyst 2960 Plus-svitsjene mangler USB-port og vil ikke ha muligheten til å bruke denne løsningen [36]. I tillegg kan USB-pinnen lastes inn med skadevare som kan føre til skade og tap av utstyr.

Kapittel 3

Mulige løsninger

I dette kapittelet blir det reflektert over de ulike teknologiene fra kapittel 2 og hvordan de kan ha anvendelse i en fungerende løsning. Kapittelet er delt i tre hovedkategorier hvor det diskuteres hvordan de kan oppfylle kravene fastsatt i oppgaven; *etablere kommunikasjon*, *overføring*, *bekreftelse* og *kombinasjon av teknologier*.

3.1 Etablere kommunikasjon vil utforske metoder for å sette opp kommunikasjon med nettverksutstyret på en effektiv måte. Metoden må løse problemstillingen ved at *startup-config* vil periodisk bli slettet.

3.2 Overføring omhandler behovet for å sende filer og utføre oppdateringer på en rask og effektiv måte.

3.3 Bekreftelse er avhengig av metoder for å verifisere at kommunikasjon har blitt oppdatert og at overføringen har blitt gjennomført og satt i funksjon.

3.4 Kombinasjon av teknologier ser på hvordan de foregående løsningene kan kobles sammen til en helhetlig prototype.

Det er viktig å nevne at det ikke finnes en standardisert ferdigløsning for nettverkskonfigurering av nettverksutstyret, eller en ferdig utviklet løsning som kan benyttes til å løse omfanget av oppgaven. Av denne grunn er det avgjørende at det velges en løsning med teknologier som tar hensyn til alle rutere og svitsjer, uavhengig av modellvariant. Det har seg slik at de forskjellige modellvariantene tar i mot ulike kommandoer og det kreves tilpasninger for at prototypen skal kunne kommunisere med alt nettverksutstyret. Dette kapittelet vil illustrere disse løsningene og avslutte med å velge en helhetlig løsning som blir videreutviklet til en prototype.

3.1 Etablere kommunikasjon

I forberedelsesfasen skal nettverksutstyr bli koblet opp mot lokalnettverket. Teknologiene som er i stand til gjøre dette er: Terminal-server, PnP, Scapy og Cisco AutoInstall. En terminalserver vil kreve mye ressurser og har enkelte ulemper som diskutert i avsnitt 2.6. PnP mangler kompatibilitet med cisco2901 (se avsnitt 2.7).

Cisco AutoInstall har også andre avgjørende ulemper som at nettverksutstyret ikke kan ha *startup-config* konfigurert (avsnitt 2.1).
For å kunne oppdage og koble til nettverksutstyr kreves det en IP-adresse. Ved å benytte den eksisterende DHCP-serveren i Cisco-laben vil det automatisk bli tildelt IP-adresse til nettverksutstyret. Dette er noe som vil være nødvendig, uavhengig av løsning. Det var ønskelig å hente IP-adresse og nettverksinformasjon direkte fra DHCP-serveren, men var mer komplisert enn først antatt. Ved å ta i bruk Scapy kan vi identifisere og oppdage nettverksutstyr som er koblet til på lokalnettverket ved hjelp av ARP-protokollen, og få tilbake tildelt IP-adresse og MAC-adresse for identifikasjon.

3.2 Overføring

For å konfigurere nettverksutstyret kreves det at ønsket konfigurasjon blir overført til de respektive ruterne og svitsjene. Dette skal foregå så sømløst som mulig, med så lite manuelt arbeid som mulig. Mulige løsninger for overføring av konfigurasjonsdata er enten Cisco AutoInstall, enten alene eller i kombinasjon med TCL.En annen mulighet er å benytte Ansible eller TCL i kombinasjon med Ansible og Python.
Cisco AutoInstall har muligheten til å sende inn TCL-skript til nettverksutstyret. En mulig løsning er å lage et TCL-skript som oppdager hvilket nettverksutstyr det blir kjørt på og bruker den informasjonen til å laste ned de relevante konfigurasjonsfilene. Dette vil medføre et ganske komplekst TCL-skript som kan være problematisk med tanke på at TCL er et relativt nytt og ukjent programmeringsspråk. Dette kan komplisere eventuell vedlikehold av skriptet.
Det er også mulig å ta i bruk en funksjon hos Cisco AutoInstall hvor nettverksutstyret blir tildelt konfigurasjonsfiler basert på tildelt IP-adresse (avsnitt 2.1). I *network-config*-filen kobles en IP-adresse sammen med et navn ved å skrive inn `ip host <navn> <IP-addresse>` for hvert nettverksutstyr. Med denne løsningen må nettverksutstyr alltid ha den samme IP-adressen, noe som kan oppnås ved å konfigurere DHCP-serveren med statisk IP. Etter møte med oppdragsgiver ble det konkludert med at det var ønskelig å beholde dynamisk IP på DHCP-serveren, og konfigurasjon av statisk IP burde unngås. Dette gir mye statisk konfigurasjon på DHCP-serveren som medfører mer komplisert vedlikehold.
En annen løsning er å anvende Python i løsningen. Python gjør det mulig å kommunisere med nettverksutstyr med SSH. Fordelen med å benytte seg av Python er at det er anvendelig og lett å tilpasse i ettertid. Alternativt kan Ansible bli benyttet alene eller i en kombinasjon med Python, i og med at teknologien er anerkjent for å være et komplett nettverksadministreringsverktøy.

3.3 Bekreftelse

Like viktig som det er å opprette kommunikasjon til nettverksutstyret og overføre filene, er det å få en bekreftelse på at overføringen er blitt gjennomført og satt i funksjon. Dette er noe som de ulike teknologiene løser i ulik grad. For TFTP-filoverføring er det ingen god måte å bekrefte om filer har blitt sendt og satt i funksjon på nettverksutstyret. For å aktivere denne funksjonaliteten må nettverksutstyret kunne sende en melding tilbake. Dette kan muligens automatiseres med et TCL-skript.
Cisco AutoInstall tilbyr ingen verifikasjon om nettverksutstyret ble konfigurert og satt i funksjon. Men det er mulig å komme rundt dette problemet ved å lage et skript som sender en melding tilbake til brukeren. Dette kan anvendes med et TCL-skript, alternativt kan man få ruteren til å overføre en fil på TFTP-serveren i *ruter-confg* (se avsnitt 2.1).
Med Ansible får man tilbakemelding på hver kommando som blir sendt, og man kan da være sikker på at konfigurasjonsfilene er sendt og satt i funksjon.

3.4 Kombinasjon av teknologier

Dersom den endelige løsningen skal bestå av flere teknologier, må disse kombineres på en måte som gjør det mulig å automatisere arbeidsflyten sømløst. På grunn av dette er det ideelt å opprette et skript som tar i mot kommandoer fra brukeren, for å eksekvere ønsket program eller å ta i bruk funksjonaliteter som tillater kommunikasjon mellom teknologiene. Det ble oppdaget to mulige løsninger for dette. Første mulighet er å ta i bruk en dynamisk inventar som er skreddersydd prosjektet [37] [38]. Denne løsningen vil bare fungere med Ansible. Den andre muligheten er mer grunnleggende, og innebærer å lage et Bash-skript som kjører alle de relevante programmene og kommandoene i en spesifisert rekkefølge.

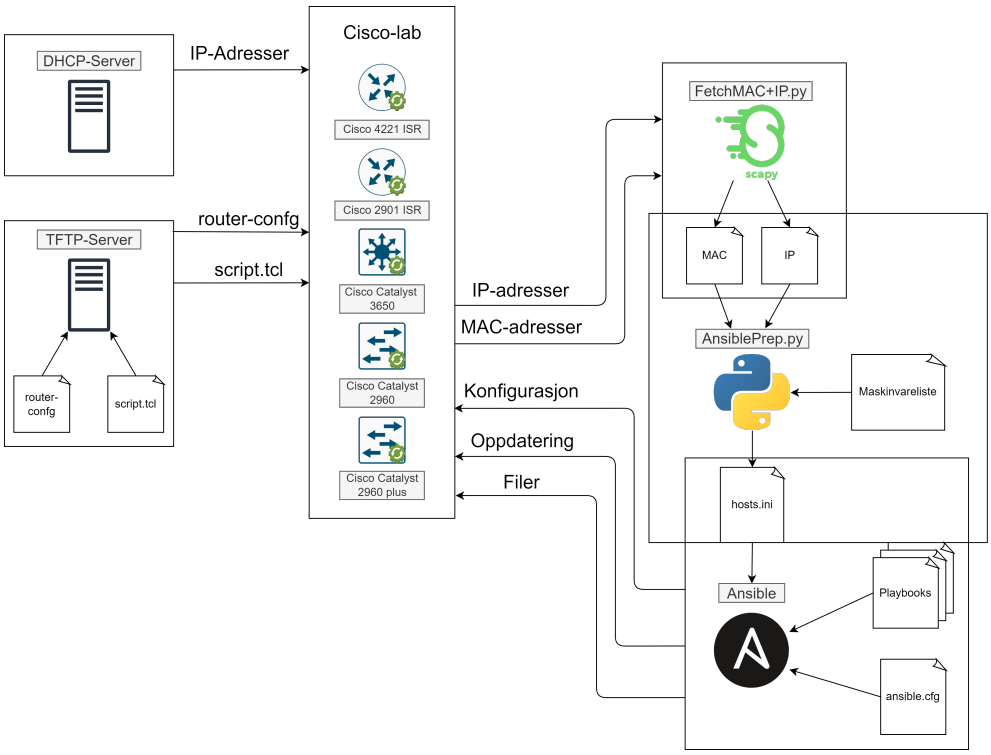
3.5 Valgt løsning

Basert på oppgavedefinisjonen i avsnitt 1.2 og samtaler med oppdragsgiver ble det konkludert med at foretrukket løsning bør ha mulighet til å overføre relevante konfigurasjonsfiler til de spesifikke modellvariantene og utføre oppdateringer av operativsystemet. Løsningen skal i tillegg gi status på konfigurasjonen av nettverksutstyr i sanntid. På bakgrunn av disse egenskapene ble det bestemt å ta i bruk Cisco AutoInstall for å sette opp SSH i nettverksutstyret, Python for å identifisere nettverksutstyrets IP-adresse og MAC-adresse med Scapy og skrive de til en fil med utstyrsinformasjon som Ansible kan tolke.

Kapittel 4

Prototypen

Formålet med prototypen er å lage en intuitiv og enkel løsning for automatisering av nettverksutstyr i Cisco-laben for oppdragsgiveren. Målet er å redusere tiden brukt til forberedelse av laboratorieøvelser og gi bedre oversikt over nettverksutstyr for laboratorieansvarlig. Løsningen vil anvende de tre ulike hovedteknologiene for å realisere en helhetlig prototype; derav Cisco AutoInstall, Python og Ansible. Cisco AutoInstall laster inn SSH-konfigurasjon for å konfigurere SSH. Python utfører en eksekverbar kode for å hente inn informasjon om nettverksutstyret og Ansible bruker den informasjonen til å sende oppdateringer og konfigurasjonsfiler til de respektive ruterne og svitsjene. Figur 4.1 viser hvordan prototypen er strukturert. Den viser at nettverksutstyr får tildelt IP-adresser fra en DHCP-server og at Cisco AutoInstall henter konfigurasjon, og hvordan Python og Ansible samarbeider for å sende konfigurasjonsfiler, og utføre oppdateringer og vedlikehold.



Figur 4.1: Diagram som viser hvordan prototypen er strukturert

4.1 Grunnkonfigurasjon

Cisco AutoInstall (se avsnitt 2.1) brukes til å konfigurere SSH på nettverksutstyret. For å bruke Cisco AutoInstall må det først settes opp en TFTP-server på det lokal-nettverket i Cisco-laben. Cisco AutoInstall er satt opp for å benytte følgende filer for prototypen; *router-confg* kodeliste 4.2 og *sshKonfig.tcl* kodeliste 4.3.

4.1.1 TFTP-serveren

For prototypen ble det benyttet *tftpd-hpa* som en TFTP-server [39] i og med den lett å administrere og sette opp. TFTP-serveren håndterer konfigurasjonen i 4.1, som er lagret i filen tftpd-hpa som ligger i */etc/default/* på Ubuntu-maskinen. Denne ble modifisert på linje 3, for å forandre mappen som TFTP-serveren vil ta i bruk.

Kodeliste 4.1: Skript: tftpd-hpa

```
1 # /etc/default/tftpd-hpa
2
3 TFTP_USERNAME="tftp"
4 TFTP_DIRECTORY="~/tftp"
5 TFTP_ADDRESS=":69"
6 TFTP_OPTIONS="--secure"
```

4.1.2 *router-confg* og *sshKonfig.tcl*-filene

Rollen til Cisco AutoInstall i prototypen er å konfigurere SSH på alt nettverksutstyr. For at dette skal være mulig, er det nødvendig med to filer på TFTP-serveren; derav *router-confg* og *sshKonfig.tcl*. Når AutoInstall-prosessen er satt i gang på nettverksutstyret vil den først laste ned *router-confg* fra TFTP-serveren og benytte den som *running-config*. Som resultat av konfigurasjonen i *router-confg* vil *sshKonfig.tcl* bli lastet ned og eksekvert ett minutt senere. En mer elegant løsning ville vært å sende inn en fungerende SSH-konfigurasjon med *router-confg*-filen. Det viser seg at løsningen ikke ville ha fungert på alle modellene med nettverksutstyr, siden noen av modellvariantene ikke tillater å generere en kryptografisk nøkkel i oppstartsfasen. På grunn av dette er det derfor nødvendig å bruke en kronjob for å generere denne utenfor oppstartsfasen. TCL-skriptet gjør det mulig å samle hele konfigurasjonen. Dette gir bedre oversiktlighet og lesbarhet av sekvensene.

Fungerende konfigurasjon

I *router-confg* (kodeliste 4.2) blir det først satt et vertsnavn på linje 1 for å lettere feilsøke i tilfelle kronjobben støter på en feil. På linjene 3,6 og 7 blir selve kronjobben konfigurert. Linje 6 setter opp når den skal kjøres, der *in 1 oneshot* indikerer at skriptet skal kjøre om *ett minutt* og *en gang*. På linje 4 kjøres kommandoen `cli tclsh tftp://255.255.255.255/sshKonfig.tcl`. `cli` betyr at det skal kjøres en kommando i EXEC-mode, som betyr at det ikke kan kjøres konfigurasjonsrelaterte kommandoer. *tclsh* betyr at det skal kjøres et TCL-skript og `tftp://255.255.255.255/sshKonfig.tcl` er URIen til `sshKonfig.tcl`. URIen kan deles opp i 3 ulike deler, *tftp://* sier at det skal hentes fra en TFTP-server, `255.255.255.255` skal normalt være IP-adressen til TFTP-serveren, men siden denne ikke nødvendigvis er den samme hele tiden blir en kringkastingsadresse benyttet istedenfor.

Kodeliste 4.2: Skript: *router-confg*

```
1 hostname preKonfig
2 kron policy-list SETUP
3 cli tclsh tftp://255.255.255.255/sshKonfig.tcl
4 exit
5 kron occurrence SETUP in 1 oneshot
6 policy-list SETUP
7 exit
```

I figur 4.3 blir selve konfigureringen av nettverksutstyr gjennomført. På linje 1 blir vertsnavnet satt, på linje 2-8 blir SSH konfigurert. Den kryptografiske nøkkelen blir generert på linje 3 og opsjonen *general-keys* blir brukt for at skriptet skal fungere på eldre versjoner av IOS. På linje 7 blir IOS-legitimasjonen som Ansible disponerer opprettet og rettighetsnivået er satt til 15 for å forsikre at Ansible har tilgang til nødvendige ressurser.

Kodeliste 4.3: Skript: *sshKonfig.tcl*

```
1 ios_config "hostname done"
2 ios_config "ip domain-name cisco.lab"
```

```
ios_config "crypto key generate rsa general-keys modulus 768"
ios_config "ip ssh version 2"
ios_config "line vty 0 4" "transport input ssh"
ios_config "line vty 0 4" "login local"
ios_config "username cisco priv 15 password cisco"
ios_config "ip scp server enable"
```

4.2 IP-adressebehandling

For å opprette en forbindelse til nettverksutstyr er det nødvendig å identifisere utstyret, både for å oppdage utstyret og for å kunne overføre tilpasset konfigurasjon til Cisco-nettverksutstyret. I dette avsnittet vil det bli beskrevet hvordan dette er gjennomført i praksis.

4.2.1 Oppdage nettverksutstyr med Scapy

I prototypen er Scapy nyttig som et verktøy for nettverksoppdagelse av andre enheter innenfor et spesifisert nettverksområde; da nettverksadresse og subnettmaske.

Scapy gjør i prinsippet to ting; sender pakker og får et resultat tilbake. Som bruker definerer du et sett med pakker og sender dem. Tilbake kommer et svar, og Scapy vil sammenligne forespørslene med svaret og returnerer en liste med pakker *(forespørsel, svar)* og pakker som ikke samsvarer. På denne måten kan Scapy skille mellom pakker fra nye enheter og gamle enheter da IP-adresse til hver enhet vil bli returnert med pakken.
Skriptet benyttet i prototypen kan sees under;

Kodeliste 4.4: Scapy: Definisjon av pakker

```
import scapy.all as scapy
import argparse
```

I figur 4.5 er det mulig å se at skriptet har som funksjon å ta et argument fra kommandolinjen. Den krever inndata i formen `'xx.xx.xx.xx/yy -t'`. I skriptet står Xene for tall i en vilkårlig IP-adresse, og Yene for subnettsmasken. Dersom kommandoen er skrevet riktig, vil skriptet kjøre.

Kodeliste 4.5: Scapy: Spesifisere nettverk og subnettmaske

```
def get_args():
    parser = argparse.ArgumentParser()
    parser.add_argument('-t', '--target', dest='target', hjelp='Target IP Address/
        Adresses')
    options = parser.parse_args()

    if not options.target:
        parser.error("[-] Spesifiser en IP adresse eller adresser, bruk --hjelp for
            mer informasjon.")
    return options
```

Figur 4.6 tar for seg selve kjernen av programmet. Her gjennomføres det skanning av IP-adresser og henting av Media Access Control (MAC)-adresser. Scapy gjennomfører en ARP-forespørsel til nettverksenheter innenfor det spesifiserte nettverksområdet. *Broadcast*-rammene blir returnert tilbake. Resultatet av skanningen lagres i en liste i sortert rekkefølge med ved hjelp av en for-løkke. Resultatet blir returnert til neste funksjon.

Kodeliste 4.6: Scapy: Oppdager IP-adresser og MAC-adresser

```
1     def scan(ip):
2     arp_req_frame = scapy.ARP(pdst = ip)
3
4     broadcast_ether_frame = scapy.Ether(dst = "ff:ff:ff:ff:ff:ff")
5
6     broadcast_ether_arp_req_frame = broadcast_ether_frame / arp_req_frame
7
8     answered_list = scapy.srp(broadcast_ether_arp_req_frame, timeout = 1, verbose =
            False)[0]
9     resultat = []
10    for i in range(0,len(answered_list)):
11        client_dict = {"ip" : answered_list[i][1].psrc, "mac" : answered_list[i
              ][1].hwsrc}
12        resultat.append(client_dict)
13
14
15    return resultat
```

I figur 4.7 blir resultatet skrevet ut som har blitt returnert fra funksjonen til to separate filer. Først åpnes filen som dataene skal lagres til, og dersom filen ble åpnet uten feil vil en for-løkke kjøres med resultatlisten, og IP-adressene og MAC-adressene vil bli skrevet til ønskede filer i spesifisert format definert i kjernefunksjonen 4.6 over.

Kodeliste 4.7: Scapy: Skriv IP-adresser og MAC-adresser til fil

```
1
2  def display_result(resultat):
3
4      with open('IP', 'w') as f:
5          for i in resultat:
6               f.write("{}\n".format(i["ip"]))
7
8      with open('MAC', 'w') as a:
9          for i in resultat:
10             a.write("{}\n".format(i["mac"]))
11
12     if (resultat):
13         print("IP-adresser har blitt lagret til IP og MAC-adresser har blitt lagret
               til MAC")
14
15
16
17 options = get_args()
18 scanned_output = scan(options.target)
19
20 display_result(scanned_output)
```

Som resultat henter skriptet både IP-adresser og MAC-adresser til nettverksenheter innenfor nettverksområdet, og skriver disse til de separate filene `'IP'` og `'MAC'`. Disse filene vil bli plukket opp i kodeliste 4.10 som sammenligner MAC-adressene med de som er i biblioteket, for identifisering av Cisco-utstyr som er i Cisco-laben.

4.2.2 Identifisere nettverksutstyr

Python-skriptet har i hovedsak to funksjoner; å sammenligne MAC-adressene hentet med Scapy med de MAC-adresseene som er lagret i `'host.ini'`-filen, og legge til/fjerne nettverksutstyr i Cisco-laben.

Kodeliste 4.8: Importering av pakker til skript

```
1 from ctypes import sizeof
2 from pickle import TRUE
3 from queue import Empty
4 import re
```

I figur kodeliste 4.8 er det mulig å se at det er det importert fire pakker til Python-skriptet. Den første pakken gjør det mulig å sammenligne størrelse på *characters* med *'sizeof'* operatoren. `'From ctypes import TRUE'` gjør det mulig å serialisere og deserialisere objektstruktur. `'From queue import Empty'` sjekker og gir beskjed dersom en 'kø-instans' har elementer i køen.

Kodeliste 4.9: Er stringen en faktisk MAC-adresse?

```
1 def ErStringMACAdr(input):
2     #Definerer mac adr sitt mønster
3     macAdrPattern = re.compile(r'(?:[0-9a-fA-F]:?){12}')
4     if re.match(macAdrPattern,input):
5         return True
6     else:
7         return False
```

Denne funksjonen i figur kodeliste 4.9 sjekker om innhentet MAC-adresse er identifiserbar. Hvis dette er tilfellet, returneres MAC-adressen til forespurt funksjon.

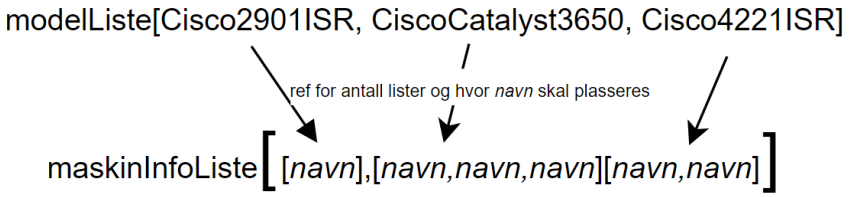
Kodeliste 4.10: Finnes MAC-adressen i *MaskinListe*?

```
1  def FinnesAdresse(MACadr):
2
3      #Lager et filobjekt som lar deg åpne, lese i filen MaskinListe
4      fil = open("MaskinListe","r")
5      #Looper gjennom hele filen
6      i = True
7      while i:
8          lesfil = fil.readline()
9
10         #Hvis vi finner en match til MAC-adressen, returner true
11         if lesfil == (MACadr.strip())+'\n':
12             return True
13         #Sjekker om vi har nådd slutten på filen.
14         if not lesfil:
15             i = False
```

```
16      #lukker fil
17      fil.close()
18      #Sier ifra at vi ikke fant matchende MAC-adr
19      return False
```

Ansible er avhengig av en del informasjon om nettverksutstyret den skal koble til. I *ansibleprep* genereres filen *host.ini* som gir Ansible denne informasjonen. Funksjonene som følger er hovedfunksjonen i *ansiblePrep*. Kildekoden i 4.11, 4.12, 4.13 og 4.14 referer til støttefunkjoner som kan finnes i vedlegg B.1.2 Figur 4.11 viser til første del av hovedfunksjonen *hentiptilmac*. Den begynner med å deklarere variabler som skal brukes i funksjonen. *maskinListe* blir åpnet og vil bli brukt for sammenligning av MAC-adresser. Fra linje 9 til 63 *loopes* det gjennom hele *maskinListe*. I linje 11-20 og 31-43 leses informasjonen om hver maskin. Det sjekkes om noen av MAC-adressene matcher de hentet av Scapy i *MAC*. Er dette tilfellet hentes IP-adresse fra *IP* hentet med Scapy som ligger på samme linjenummer som MAC-adressen i *MAC*. Mye av denne informasjonen blir i linje 44 lagt i variabelen *navn* og vil bli brukt av Ansible til å identifisere og muliggjøre fjernstyring av nettverksutstyr.

Oppdragsgiver ønsket at løsningen skulle være fleksibel nok til å kunne håndtere utskiftning av nettverksutstyr. Derfor er det tatt hensyn til dette i *ansiblePrep*. Koden må ta hensyn til at nye modeller kan bli lagt til eller fjernet. For å håndtere dette er det implementert en todimensjonal liste, og hver modell har en liste over *navn*-variabler som tilhører den modellen. figur 4.2 visualiserer denne listen. *navn* blir lagt inn i den modellen.



Figur 4.2: Strukturen på den todimensjonale modellen med eksempeldata

På linje 21-30 samler man opp modellnavnet på alt nettverksutstyret. Disse blir viktig videre i linje 47-59 for å legge til hvert *navn* i riktig `maskinInfoListe`, som selv ligger i `modellListe`.

Kodeliste 4.11: Leser fra `maskinInfoListe` og sammenligner med Scapy

```
1       fil = open("maskinListe","r")
2
3       #Liste for å registrere de forskjellige modellvariantene
4       modellListe = []
5       #Bibliotek for midlertidig oppbevaring av info som skal skriver under
            modeltyper.
6       maskinInfoListe = []
7       #Loop gjennom hele filen
```

```
    i = True
    while i:

        #Hvis vi kommer til nytt innslag av en maskin, legg i tempOppbevaring
        podKey = fil.readline()
        podKey = ryddOppLinjeskift(podKey)

        posisjonKey = fil.readline()
        posisjonKey = ryddOppLinjeskift(posisjonKey)

        modell = fil.readline()
        modell = ryddOppLinjeskift(modell)

        #Sjekker om denne modellen har blitt registrert før i funksjonen, hvis ikke
             legg den til i modellListe
        eksisterer = False
        for x in modellListe:
            if x == modell:
                eksisterer = True
        if (eksisterer == False):
            modellListe.append(modell)
            maskinInfoListe.append([]) #Lager plass for den nye modellen


        #leser in alle MAC-adressene
        #Stanser når vi når linjeskift eller slutt på filen
        ip =""
        e = True
        while e:
            lesfil = fil.readline()
            if ErStringMACAdr((lesfil)):
                if ErAdrInMACtxt(lesfil):
                    ip = HentIpFraMac(lesfil)
                    ip = ryddOppLinjeskift(ip)
            else:
                e = False

        navn = (podKey + posisjonKey + " ansible_host=" + ip)
        #Legger maskinen til i maskinInforListe riktig formatert med pod og
             posisjon kombinert.

        #Legger informasjonen om maskinen Ansible trenger til i maskinInfoListe
        #Først sjekker den hvilken modell maskinen er:
        teller = 0
        for i in modellListe:
            if i == modell:
                radNr = teller
            teller += 1
        #Så legger den til maskinen i riktig rad
        teller = 0
        for i in maskinInfoListe:
            if teller == radNr:
                    i.append(navn)
            teller += 1

        #Sjekker om vi har nådd slutten på filen.
        if not lesfil:
            i = False
    fil.close()
```

I figur 4.12 fortsetter *hentiptilmac* der figur 4.11 slapp. Denne seksjonen fokuserer på generere *host.ini* filen med informasjonen som ble hentet i kodeliste 4.11. Først åpnes *host.ini*-filen. Fra linje 4 til 22 går man gjennom hele `modellListe` og skriver til fil hver modell og *navn*-variablene som hører til den modellen.

Kodeliste 4.12: Skriver modeller og navn-variablene til *host.ini*

```
1  #Skriver alt til filen
2     fil = open("hosts.ini","w")
3
4     #Skriver inn hver modell
5     modellNr = 0
6     bruktIp = 0
7     for i in modellListe:
8         fil.write('[' + i + ']\n')
9         rad = 0
10
11        #Skriver inn hver maskin som tilhører hver modell
12        for y in maskinInfoListe:
13            if modellNr == rad:
14                for e in y:
15                    #Ta med modellen i filen hvis navn er lang nok til at den har
                          med ip
16                    if len(e) > 22:
17                        fil.write(e + '\n')
18                        #Tell opp Ip-er som faktisk fikk tilkobling
19                        bruktIp += 1
20            rad += 1
21        fil.write('\n')     #Legger til et linjeskift mellom hver modell
22        modellNr +=1
```

Etter å ha lest inn identifikatorene på enkelt-nettverksutstyr, legges det til ekstra informasjon som Ansible trenger i kodeliste 4.13. Denne må legges til for hver modell.

Kodeliste 4.13: Legger ved ekstra informasjon til *host.ini*

```
1     #Skriver variablene knytter til hver modell
2     for i in modellListe:
3         fil.write('[' + i + ':vars]\n')
4         fil.write('ansible_connection=ansible.netcommon.network_cli\n')
5         fil.write('ansible_network_os=cisco.ios.ios\n')
6         fil.write('ansible_user=cisco\n')
7         fil.write('ansible_password=cisco\n')
8         fil.write('ansible_become=yes\n')
9         fil.write('ansible_become_method=enable\n')
10        fil.write('ansible_become_password=cisco\n')
11        fil.write('\n')     #Legger til et linjeskift mellom hver modell
```

Til slutt sjekkes det i kodeliste 4.14 hvor mange IP-adresser som Scapy fant, og hvor mange av dem som er klare for bruk av Ansible. Et eksempel på hvordan *host.ini* ser ut etter å ha kjørt *ansiblePrep* kan man finne i kodeliste 5.1

Kodeliste 4.14: Gir tilbakemelding om suksessrate for *host.ini*

```
1     #Gir statistikk om hvor mange ip addresser som finnes, og hvor mange vi klarte
          å koble til en maskin
2     fil = open("IP","r")
3     lesfil = fil.readline()
4     totalIp = 1
5     #Teller hvor mange IP-er vi har
6     while lesfil:
7         lesfil = fil.readline()
8         if len(lesfil) > 6:
9             totalIp += 1
10    print("Totalt antall IP-er:", totalIp)
11    print("Antall tilkoblede IP-er:", bruktIp)
12    fil.close()
```

4.2.3 Legge til/fjerne nettverksutstyr i Cisco-laben

I *maskinListe* finnes en oversikt over alle podene og nettverksutstyret på dem. Denne har to viktige bruksområder. Først og fremst brukes den som referanse over MAC-adressene som hver enhet har i Cisco-laben. Dette bruker funksjonen 4.8 til å identifisere finne ut hvilken IP-adresse som tilhører hvilken maskin. Den brukes også til å holde oversikt over hvilke maskiner som finnes i Cisco-laben. Denne inneholder i synkende rekkefølge det følgende:

- Informasjon om hvilken pod nettverksutstyret ligger i.
- Informasjon om hvilken rad i poden nettverksutstyret ligger på. Dette telles fra toppen og nedover. Internett-svitsjene er ikke inkludert.
- Modellnavnet til nettverksutstyret.
- MAC-adressene tilnyttet dette nettverksutstyret.

Kodelisten 4.15 viser et eksempel på et innslag i *maskinListe*.

Kodeliste 4.15: Eksempel på et nettverksutstyr i *maskinListe*

```
1  pd9
2  ps2
3  Cisco2901ISR
4  10:05:ca:79:50:b8
5  10:05:ca:79:50:b9
```

For oppdragsgiver er det ønskelig at løsningen forblir funksjonell fremover i tid. Det er derfor hensiktsmessig å forsikre at prototypen er fleksibel og kan tilpasses i tilfelle utskiftning av maskinvare i Cisco-laben. *MacAdrBehandling.py* er laget for å sørge for at endringer i *maskinListe* blir gjennomført med redusert risiko for menneskelig feil. *macAdrBehandling* har flere støttefunksjoner som brukes funksjoner i kodelistene 4.16, 4.17, 4.18 og 4.19 som ikke blir forklart i besvarelsen. Støttefunksjonene kan finnes i vedlegg B.1.2.

Ved eksekvering av *MacAdrBehandling.py* blir man presentert med en meny som leder en gjennom slike endringer. I kodelisten 4.16 ser man strukturen til denne menyen. Med unntak av variabelen `fortsett` på linje 2, ligger hele funksjonen i en *while*-løkke som kun stanser når brukerer velger 'q' i menyen. På linje 5-10

printes menyen ut for å gi brukeren de ulike menyvalgene. På linje 11-30 utføres menyvalget. Det er verd å nevne at ettersom mye av det brukeren skriver inn blir sammenlignet med *maskinListe*, og derfor blir denne dataen behandlet før sammensligningsfunkjoner blir tilkalt for å redusere risikoen for feil underveis.

Kodeliste 4.16: Python: Meny for endring i *maskinListe*

```
1  #Så lenge ikke q(avslutt), la bruker bruke scriptet ved å loope menyen
2  fortsett = True
3  #Menyen
4  while fortsett:
5      print('\n')
6      print("1: Legg til maskin")
7      print("2: Slett maskin")
8      print("3: Sjekk om maskin finnes")
9      print("q: avslutt")
10     valg = input("Hva vil du gjøre?: ")
11
12     if valg == '1':
13         LeggTilMaskin()
14     elif valg == '2':
15         pod = input("Hviklen pod er den i?: ")
16         pod = ryddOppLinjeskift(pod)
17         pos = input("Hviklen posisjon skal slettes?: ")
18         pos = ryddOppLinjeskift(pos)
19         SlettMaskin(pod,pos)
20     elif valg == '3':
21         pod = 'pd' + input("Hviklen pod?: ")
22         pos = 'ps' + input("Hviklen posisjon?: ")
23         if FinnesMaskin(pod, pos):
24             print("Maskinen finnes")
25         else:
26             print("Maskinen finnes ikke")
27     elif valg == 'q':
28         fortsett = False
29     else:
30         print("----------------Ikke gyldig input")
```

Det er viktig at de nye maskin-innslagene er formatert riktig, og derfor er det lagt til en funksjon illustrert i kodeliste 4.17 som leder brukeren gjennom hvert punkt når en ny maskin skal legges til. På linje 4-39 tar imot informasjon om det nye nettverksutstyret og sjekker at alt er riktig formatert. Linje 40-50 legger den nye maksinen inn i *host.ini*.

Kodeliste 4.17: Python: Legge til en ny maskin i *maskinListe*

```
1  def LeggTilMaskin():
2
3      #Sjekk at kombinasjonen av pod og posisjon ikke er brukt allerede
4      i = True
5      while i:
6          pod = "pd" + (input("Hvilket Pod-nr?: "))
7          posisjon = "ps" + (input("Hvilket radnr i pod?: "))
8          #Hvis de prøver å legge til på en pod og posisjon som allerede finnes, gi
               feil og få dem til å gjøre dem på nytt
9          if FinnesMaskin(pod, posisjon):
10             print("Det er allerede en maskin på den posisjonen. Slett den
                   eksisterende maskinen hvis du ønsker å legge til en ny maskin i pod
```

```
                :", pod ," på posisjon:", posisjon)
11          else:
12              i = False
13      print("Eksisterende modeller er:")
14      PrintModeller()
15      modell = input("Case sensitiv. Modellnavn: ")
16
17      #Lar bruker legge til så mange adresser de vil
18      i = True
19      MACadr = []
20      while i:
21          adr = input("MAC adresse, 1 om gangen. "'"q"'" for ferdig: ")
22          #Hvis brukeren er ferdig, avslutt Mac-adr innlesning
23          if adr == "q":
24              #Hvis du har lagt til hvertfall en addresse kan du gå videre. Hvis ikke
                     må du legge inn minst en addresse
25              if (len(MACadr) > 0):
26                  i = False
27              else:
28                  print("Du må legge til minst en adresse")
29
30          #Hvis mac-adressen er feilformatert, gi feilmelding
31          elif not (len(adr) == 17 and ErStringMACAdr(adr)):
32              print("Ikke et gyldig MAC-adresseformat. Skal skrives i formatet
                    ??:??:??:??:??:??")
33          elif FinnesAdresse(adr):
34              print("Denne MAC-addressen er allerede registrert på en annen maskin.")
35          elif HarListeDuplikat(MACadr,adr):
36              print("Du har allerede prøvd å legge til denne addressen")
37          #Legg Mac-adr til i en liste
38          else:
39              MACadr.append(adr)
40      #Så legger du inn alle variablene: pod, posisjon, modell, macadr[] i den rekkef
            ølgen til filen maskinListe
41      fil = open("maskinListe","a")
42      fil.write('\n')
43      fil.write(pod +'\n')
44      fil.write(posisjon +'\n')
45      fil.write(modell +'\n')
46      #Loop for å gå igjennom lister
47      for x in range(len(MACadr)):
48          fil.write(MACadr[x]+'\n')
49      fil.close()
50      print("La til " + modell + " på pod " + pod + " i posisjon " + posisjon)
```

Det er også viktig å fjerne maskiner som har blitt tatt ut av bruk på en korrekt måte. Her er `SlettMaskin(pod,pos)` funksjonen i kodeliste 4.18 veldig nyttig for å forsikre for at sletting blir gjennomført skikkelig. Den kan også brukes for å rette opp hvis det er for mange linjehopp mellom to maskiner, noe som vanligvis skaper et problem når *maskinListe* skal leses. Sletting av en maskin fører til at *maskinListe* blir lagt på nytt uten det slettede nettverksutstyr.
På Linje 1-8 deklareres variabler og åpner *maskinListe* i lesemodus. Deretter leses det gjennom hele filen ved hjelp av en while-loop på linje 9-50. Finner programmet, ved hjelp av linje 40-48, en maskin som matcher poden og posisjonen brukeren ville slette, blir denne maskinen ekskludert fra den nye *maskinListe*-filen. *maskinListe* blir lukket i linje 51 og blir så laget på nytt på linje 53-66.

Kodeliste 4.18: Python: Fjerne en maskin fra *maskinListe*

```
1
2       def SlettMaskin(pod, pos):
3       fil = open("maskinListe","r")
4       #Liste med alt som skal beholdes
5       lagretInnhold = []
6
7       #Loop gjennom hele filen og legg inn alt som skal beholdes i lagretinnhold
8       i = True
9       while i:
10
11          #Liste søm brukes for å oppbevare informasjonen til en maskin før vi
                bestemmer om det skal beholdes elles slettes
12          tempOppbevaring = []
13          #Hvis vi kommer til nytt innslag av en maskin, legg i tempOppbevaring og
                sjekk om den skal beholdes
14              #Sjekk at det ikke er en tom linje før vi leser. Det kan tulle til
                    lesning av fil
15
16          podKey = fil.readline()
17          tempOppbevaring.append(podKey)
18          posisjonKey = fil.readline()
19          tempOppbevaring.append(posisjonKey)
20          modell = fil.readline()
21          tempOppbevaring.append(modell)
22
23          #leser in alle mac addressene og legger dem i tempOppbevaring.
24          #Stanser når vi når linjeskift eller slutt på filen
25          e = True
26          while e:
27              lesfil = fil.readline()
28              if ErStringMACAdr((lesfil[0:17])):
29                  tempOppbevaring.append(lesfil)
30              else:
31                  e = False
32
33          #legger in '\'n'
34          if lesfil:
35              tempOppbevaring.append(lesfil)
36
37
38          #Sjekker om maskinen er markert som slettes. er den det sier den ifra.
                ellers legges til i det som skal lagres
39          #Hvis input = nr
40          if ((podKey == (pod +'\n')) and (posisjonKey == (pos +'\n'))):
41              print("Sletter: ",modell, " På pod: ", podKey, " plass: ",posisjonKey )
42          #Hvis input = pd/ps i tilegg til nr
43          elif ((podKey == ('pd' + pod +'\n')) and (posisjonKey == ('ps' + pos +'\n')
                )):
44              print("Sletter: ",modell, " På pod: ", podKey, " plass: ",posisjonKey )
45          else:
46              lagretInnhold.extend(tempOppbevaring)
47
48              #Sjekker om vi har nådd slutten på filen.
49          if not lesfil:
50              i = False
51      fil.close()
52
53      #Overskrive den gamle filen med det som skal beholdes
```

```
54      fil = open("maskinListe","w")
55      linjeskift = False
56      for x in range(len(lagretInnhold)):
57          if not ((lagretInnhold[x] == ('\n' or '\n\t')) and linjeskift):
58              fil.write(lagretInnhold[x])
59
60          #Passer på at vi ikke skriver in to tomme linjer etter hverandre.
61              #Da dette kan føre til leseproblemer
62          if lagretInnhold[x] == ('\n' or '\n\t'):
63              linjeskift = True
64          else:
65              linjeskift = False
66      fil.close()
```

Til slutt er det nyttig å kunne bekrefte om en maskin finnes i *maskinListe* eller ikke, slik som blir gjort i kodeliste 4.19. Dette kommer til nytte både for å sjekke at maskinen du skal endre på faktisk finnes og for å bekrefte at utførte konfigurasjonsendringer er blitt gjennomført.
Linje 1-9 åpner *maskinListe* for lesning og deklarere variabler. På linje 10-22 leser programmet gjennom hele filen og ser etter matcher for pod og posisjon. Til slutt sjekker programmet i linje 23-27 om den fant en match for begge. Hvis dette var tilfellet, returnerer den `true`, ellers returneres `false`.

Kodeliste 4.19: Python: Sjekker om maskin finnes i *maskinListe*

```
1   def FinnesMaskin(pod, posisjon):
2       fil = open("maskinListe","r")
3
4       #Variabler som begge må bli true for å returnere true
5       podfinnes = False
6       posisjonfinnes = False
7
8       #Looper gjennom hele filen
9       i = True
10      while i:
11          lesfil = fil.readline()
12          #Hvis vi finner en match til pod eller posisjon, marker som true
13          if lesfil == (pod)+'\n':
14              podfinnes = True
15          if lesfil == (posisjon)+'\n':
16              posisjonfinnes = True
17          #Sjekker om vi har nådd slutten på filen.
18          if not lesfil:
19              i = False
20      #lukker fil
21      fil.close()
22
23      #Returnerer resultat. True hvis pod-posisjon kombo allerede finnes, false hvis
            ikke
24      if (podfinnes and posisjonfinnes):
25          return True
26      else:
27          return False
```

4.3 Kommunikasjon, filoverføring og oppdatering

Ansible benytter seg av inventar-filer og Playbook-filer med tasks for å konfigurere nettverksutstyret. For å tilfredstille oppgavekravet om å kunne oppgradere nettverksutstyret lages det forskjellige Playbooks etter type og modell.

4.3.1 Ansible-konfigurasjonsfil

Ansible benytter seg av en konfigurasjonsfil [40] slik at man kan tilpasse programvaren etter behov. Konfigurasjonsfilen blir automatisk lagt i */etc/Ansible/Ansible.cfg* på Linux-maskinen, ved installasjon av Ansible. En tilpasset konfigurasjonsfil kan opprettes etter behov. I prototypen er det laget en egen *Ansible.cfg* i mappen som man skal utføre arbeidet i, se i kodeliste 4.20. I *Ansible.cfg* filen vil man kunne sette forskjellige variabler avhengig av hva man trenger i forhold til ulike tasks man vil kjøre i *Playbookene*.

I denne oppgaven benyttes standardinnstillingene til Ansible med noen endringer som gjøres i *Ansible.cfg* i arbeidsmappen som gjør at Ansible skal eksekvere bedre for bruksområdene i oppgaven.

Kodeliste 4.20: Ansible: Ansible.cfg

```
# ansible.cfg
[defaults]
inventory = hosts.ini
deprecation_warnings=False
host_key_checking=false
action_warnings = false
forks = 50

[persistent_connection]
command_timeout = 120

[ssh_connection]
pipelining = True
ssh_args = -o ControlMaster=auto -o ControlPersist=15m
transfer_method = piped
```

`inventory = hosts.ini` sier at i mappen som arbeidet utføres i skal ha en fil som heter `hosts.ini`, som inneholder informasjon om Ansible vertene som man skal kjøre konfigurasjoner og endringer på.

`deprecation_warnings=False` forhindrer Ansible fra å kjøre uten å gi advarsler om at eventuelle spesifikke funksjoner må endres, denne er satt til `false` for å holde det mer ryddig slik at man får bedre kontroll. Der `action_warnings = false` vil slå av varsler om handlinger som utføres, dette er satt til *false* for at Ansible skal være mer leselig.

`host_key_checking=False` har som funksjon å unngå å sjekke fingeravtrykk når man benytter SSH for kontakt med Cisco-utstyret. Dette kan være til hjelp for å spare tid i oppkoblingsfasen mellom kontrollnoden og nettverksutstyret.

Mengden med nettverksutstyr i Cisco-laben tilsier at man vil sende tasks til flere

enn standard innstillingen i Ansible som er *forks = 5*, den skiftes til *forks = 50* slik at kommandoer og utførelse av arbeid på nettverksutstyret kan gjøres med femti om gangen istedet for fem.

For at Ansible-kommandoer som ikke sendes skal tidsgrensen settes til *command_timeout = 120* slik at Ansible venter 120 sekunder etter at kommandoen forsøkt sendt i stedet for standardinstillingen på 30 sekunder.

For å bedre og gi raskere SSH-tilkobling er endringer av Ansibles SSH-grunninnstillinger satt i funksjon [41]. `pipelining = True` benyttes for å gi Ansible raskere SSH tilkobling ved at den ikke vil opprette nye SSH-tilkoblinger, men benytte samme tilkobling som var opprettet før. Linje `ssh_args = -o ControlMaster=auto -o ControlPersist=15m` benyttes for å bedre SSH-konnektiviteten ved at `ControlMaster=auto` gir mulighet for flere samtidige SSH-forbindelser til samme administrert node slik at forskjellige tasks benytter samme SSH-forbindelse. `ControlPersist=15m` benyttes for at SSH-forbindelsen skal være åpen i bakgrunnen i 15 minutter. `transfer_method = piped` benyttes slik at standardoverføringen benytter dd-protokoll som vil overføre bits bitvis i en mer effektivt enn andre filoverføringsprotokoller.

4.3.2 Ansible inventar-fil

I Ansible grupperer administrert node i infrastrukturen i et inventar [42], i *Ansible.cfg* er det pekt på at inventaret er i *hosts.ini*-filen. I inventarfilen *hosts.ini* (se kodeliste 4.21) kan man gi `pd4ps2 Ansible_host=10.10.0.2` som forteller at den administrerte noden `pd4ps2` har IP-adressen `10.10.0.2` For at SSH skal fungere for å kommunisere og samtidig tilgjengeliggjøre en måte å konfigurere de administrerte nodene, må man gi variablene i `[pd4ps2:vars]` brukernavn, passord, mulighet for å eskalere privilegiene og kommunikasjon med Cisco-nettverksutstyr med `ansible_network_os=cisco.ios.ios`.

Kodeliste 4.21: Ansible: Vertsfilen `hosts.ini`

```
1
2  [Cisco2901ISR]
3  pd4ps2 Ansible_host=10.10.0.2
4
5  [pd4ps2:vars]
6  ansible_connection=ansible.netcommon.network_cli
7  ansible_network_os=cisco.ios.ios
8  ansible_user=cisco
9  ansible_password=cisco
10 ansible_become=yes
11 ansible_become_method=enable
12 ansible_become_password=cisco
```

4.3.3 Ansible-kolleksjoner

Ansible *collections* eller kolleksjoner er et distrubusjonsformat for innhold i Ansible som skal være til hjelp med automatisering og forenkle automasjonsprosessen av

oppgavene. Ansible-kolleksjoner er open-source som betyr at det åpent for alle til å publisere på Ansible Galaxy.

Ansible Network Collection for Common Code-kolleksjonen

Ansible Network Collection for Common Code (`netcommon`) [43] er en standard Ansible-kolleksjon utviklet for automatisering og administrering av nettverksutstyr. Denne kolleksjonen inneholder nyttige moduler for å kopiere filer fra kontrollnoden til nettverksutstyret, og sende relevante kommandoer til nettverksutstyr. Denne kolleksjonen er mindre spesifisert enn Cisco IOS-kolleksjonen, men kan lettere implementeres hvis man benytter seg av forskjellige leverandører som bruker lik syntaks, som for eksempel å endre på variabler i `'hosts.ini'`-inventarfilen for spesifikke handlinger.

Cisco IOS-kolleksjonen

Cisco IOS-kolleksjonen er laget for å kommunisere med Cisco-utstyr. For oppgradering og overføring av filer er fakta-, kommando- og konfigurasjonsmodulene relevante.

Faktamodulen `cisco.ios.ios_facts` [44] samler inn diverse fakta fra Cisco-enhetene og returnerer informasjon tilbake til kontrollnoden. Dette kan være nyttig informasjon som vertsnavn, IOS-versjon, modellvariant, IPv4 og IPv6-konfigurasjon og adresser som er i bruk, serienummer på enheten og *IOS-image*-filen som benyttes.

Kommandomodulen `cisco.ios.ios_command` [45] benyttes for å sende kommandoer til IOS i *enable*-modus (aktiveringsmodus). Modulen er nyttig for å sende CLI-kommandoer til en spesifikk Cisco IOS-node.
Konfigurasjonsmodulen `cisco.ios.ios_config` [46] kan benyttes for å skrive kommandoer til en Cisco-IOS-node i konfigurasjonsmodus.

4.3.4 Tasks

Ansible benytter seg av tasks, som i oppgaver som skal kjøres på nettverksutstyr og disse inngår i *Playbook*en. I prototypen benyttes det programvaremoduler som kommer med grunninstallasjonen av Ansible. Netcommon, og Cisco IOS-kolleksjoner må lastes ned og installeres i ettertid. En task kan være å benytte `Ansible.netcommon.cli_command`, i kodeliste 4.22, på nettverksutstyret for å lage en mappe i flashminnet.

Kodeliste 4.22: Ansible: *Ansible.netcommon.cli*_command task

```
1
2     tasks:
3    - name: Lag mappe
4       Ansible.netcommon.cli_command:
5         command: 'mkdir flash0:/mappe/'
6         prompt:
```

```
7           - '[mappe]?'
8         answer:
9           - 'mappe'
```

En annen netcommon-modul som er relevant for prototypen er navngitt *put*, som gjør det mulig å kopiere filer fra kontrollnoden til nettverksutstyret. Ved bruk av `Ansible.netcommon.net_put`-modulen vil man ved bruk av SCP [43] kunne overføre filer fra kontrollnoden på nettverksutstyr, men på grunn av endring i *ansible.cfg* vil den benytte `'dd'` på nettverksutstyret som kan benytte det. Eksempel på task som benytter seg av `Ansible.netcommon.net_put`-modulen er i kodeliste 4.23.

Kodeliste 4.23: Ansible: *Ansible.netcommon.net*_put task

```
1
2     tasks:
3     - name: Kopiere fil til mappe
4       Ansible.netcommon.net_put:
5         src: '~/mapper/fil.txt'
6         dest: 'flash:/mappe/fil.txt'
7       vars:
8         Ansible_command_timeout: 600
```

For å konfigurere nettverksutstyr med Ansible kan `cisco.ios.ios_config`-modulen benyttes. I Cisco-laben blir IP-adresse satt av en DHCP-server. Når nettverksutstyr starter på nytt er man avhengig av at den er lik for å kunne gå over til neste *task* i en *Playbook*. Ved å ta i bruk `cisco.ios.ios_config`-modulen kan man konfigurere IP-adressen statisk og benytte seg av `Ansible_host`-variablen som er definert i inventarfilen. Et eksempel på dette finnes i kodelisten 4.24.

Kodeliste 4.24: Ansible: *cisco.ios.ios*_config task

```
1
2     tasks:
3     - name: Sett ip
4       cisco.ios.ios_config:
5         lines:
6           - description IP på vlan 1 til lik som gitt av dhcp
7           - ip address {{ Ansible_ssh_host }} 255.255.255.0
8         parents: interface vlan 1
```

4.3.5 Playbooks

Playbooks er skrevet i programmeringsspråket YAML. Dette kan sammenlignes med et bibliotek som inneholder retningslinjene satt for å eksekvere en samling med skripter. I *Playbooken* blir hvert skript refert til som en *task*, og er strukturert slik at den vil sekvensielt kjøre gjennom *Playbooken*. I kodeliste 4.25 vil den første tasken være å benytte seg modulen IOS-facts for å skrive ut forespurt informasjon. Når Playbooken er spilt vil den gi utskrift som sett i kodeliste 4.26.

Kodeliste 4.25: Ansible: *Playbook*

```
1 ---
```

```
2  - name: Cisco Vis versjons info
3    hosts: CiscoCatalyst3650,Cisco2901ISR,CiscoCatalyst2960p,CiscoCatalyst2960,
         Cisco4221ISR
4    gather_facts: false
5
6    tasks:
7      - name: Vis modell, versjon av IOS og boot
8        ios_facts:
9      - debug:
10         msg:
11         - "{{ inventory_hostname }} {{ ansible_net_model }} har versjon {{
               ansible_net_version }} ios  og benytter {{ ansible_net_image }} til
               boot"
```

Kodeliste 4.26: Ansible: Utskrift etter å ha spilt *Playbook* på en Cisco 4221 ruter

```
1  PLAY [Vis ruter og info]
       ****************************************************************
2  TASK [Vis modell, versjon av IOS og boot]
       ****************************************************************
3  ok: [pd4ps1]
4
5  TASK [debug] ********************************************************
6  ok: [pd4ps2] => {
7      "msg": [
8          "pd4ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
               universalk9_ias.16.09.02.SPA.bin til boot"    ]
9  }
10
11 PLAY RECAP ********************************************************************
12 pd4ps1  : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0
          ignored=0
```

Playbooken i kodeliste 4.25 er en veldig enkel Playbook som kun benytter to tasks og har en host og vil være nyttig for å sjekke IOS-versjon på nettverksutstyret.

Oppgradere nettverksutstyr

*Playbook*en for å oppgradere nettverksutstyret til ny IOS-versjon er relativt like, men Cisco Catalyst 3650 lag-3-svitsjene er konfigurert annerledes og benytter pakker og ikke en enkelt BIN-fil-fil. Koden i kodeliste 4.27 brukes for å oppgradere en Cisco 2901-ruter. Ved å sette variabler i toppen av koden vil det være lettere å kunne bytte ut BIN-filen med en ny som inneholder et nytt IOS-operativsystem. Playbooken vil benytte seg av å samle inn Ansible-kolleksjonene for å sjekke fakta og gi beskjed om en ny versjon er tilstede, som resultat vil den ikke gå videre med oppgradering i tilfelle en ny IOS-versjon er blitt lansert. Ved linje 27-34 settes statisk IP for at den ikke skal få ny IP av DHCP etter *reload* etter byttet av *imaget*. Ved linje 36-43 blir en nytt *image* lastet opp på nettverksutstyret `net_put`-modulen. I linje 49-55 skiftes boot-variablen til det nye *imaget*, og ved linje 53-70 lagres konfigurasjonen. nettverksutstyret *reloades* og Ansible venter på at den får opprette en forbindelse på nytt via den IP-adressen som ble statisk konfigurert. Avslutningsvis, blir det bekreftet om programvaren ble oppdatert, og hvis dette er tilfelle slettes den gamle programvaren.

Kodeliste 4.27: Ansible: *Playbook* for å oppdatere Cisco 2901-ruter

```
---
# Ansible Playbook to upgrade Cisco IOS
- name: Oppgrader CISCO 2901
  hosts: Cisco2901ISR
  gather_facts: false

  vars:
    flash_name: flash0
    old_image_name: c2900-universalk9-mz.SPA.156-3.M9.bin
    new_image_name: c2900-universalk9-mz.SPA.157-3.M8.bin
    path_to_images: '~/IOS.spring2022/2900/'
    upgrade_ios_version: 15.7(3)M8
    old_version: "{{  ansible_net_version  }}"

  tasks:
    - name: Sjekk nåverende IOS
      ios_facts:
    - debug:
        msg:
          - "Nåverende versjon er {{ old_version }}"
          - "Oppgraderigns versjon er {{ upgrade_ios_version }}"
    - debug:
        msg:
          - "Ruter vil bli oppgradert!"
      when: ansible_net_version != upgrade_ios_version

    ## Sett  ip statisk for å forberede reload
    - name: Sett ip
      cisco.ios.ios_config:
        lines:
          - description test interface
          - ip address {{ ansible_ssh_host }} 255.255.255.0
        parents: interface g0/0
      when: ansible_net_version != upgrade_ios_version

    ## Kopiere software til device
    - name: Kopiere over bin filen
      net_put:
        src: '{{  path_to_images  }}/{{  new_image_name  }}'
        dest: '{{ flash_name }}:{{ new_image_name }}'
      vars:
        ansible_command_timeout: 10000
      when: ansible_net_version != upgrade_ios_version

    ##  Skift Boot Variable til nytt image
    - name: Skift boot variablen til nytt image
      ios_config:
        commands:
          - "no boot system"
          - "boot system {{  flash_name  }}:{{  new_image_name  }}"
      when: ansible_net_version != upgrade_ios_version

    ## Reload
    - name: Restart ruter
      cli_command:
        command: reload
        check_all: True
        prompt:
```

```
59          - Save?
60          - confirm
61        answer:
62          - y
63          - y
64      when: ansible_net_version != upgrade_ios_version
65
66    ## Vent på at utstyr kommer tilbake online
67    - name: Vent på at ruter kommer tilbake på nett
68      wait_for_connection:
69        delay: 10
70      when: ansible_net_version != upgrade_ios_version
71
72    ## Sjekk versjon
73    - name: Sjekk image versjon
74      ios_facts:
75    - debug:
76        msg:
77          - "Nåverende versjon er {{ ansible_net_version }}"
78    - assert:
79        that:
80          - upgrade_ios_version == ansible_net_version
81    - debug:
82        msg:
83          - "Programvare oppdateringer er komplett!"
84
85    ## Slett gammel versjon
86    - name: Slett gammel bin fil
87      cli_command:
88        command: 'delete /force {{ flash_name }}:{{ old_image_name }}'
```

Cisco Catalyst 3650 lag-3-svitsjene blir oppgradert på en annen måte. Forskjellen er at den tar i mot flere pakker som pakkes ut av en BIN-fil, i stedet for å ta i mot en kombinert fil. Disse filene blir tilkalt etterhvert som oppgraderingsprosessen er i gang. Overføring av den nye BIN-fil er lik, men benytter seg av kun en task for å pakke ut, installere og skifte bootvariabelen sett i kodeliste 4.28 linje 10 til 13. Siden den benytter pakker for BIN-filen blir den pakket ut slik. Versjonsnavn benyttes for å rydde opp og slette det gamle IOS-imaget, istedenfor å peke til filen som en variabel, for å se full kode for å oppdatere Cisco Catalyst 3650 se vedlegg B.2.1.

Kodeliste 4.28: Ansible: Installere IOS på Cisco Catalyst 3650

```
1
2   vars:
3     flash_name: flash
4     new_image_name: cat3k_caa-universalk9.16.12.07.SPA.bin
5     path_to_images: ~/IOS.spring2022/3650
6     upgrade_ios_version: '16.12.07'
7     old_ios_version: '16.12.05b'
8
9     tasks:
10    - name: Installer Bin filen
11      ansible.netcommon.cli_command:
12        command: "request platform software package install switch all file {{
              flash_name }}:{{ new_image_name }} new auto-copy"
13      when: ansible_net_version != upgrade_ios_version
```

For å automatisere oppgraderingen av nettverksutstyret, er alle *Playbookene* for å oppgradere nettverksutstyret satt sammen i en *Playbook* (se kodeliste 4.29 som vil iverksettes først for Cisco 4221 ISR, og så Cisco 2901 ISR, Cisco Catalyst 2960, Cisco Catalyst 2960 Plus og til slutt Cisco Catalyst 3650.

Kodeliste 4.29: Oppgradere IOS-image

```

  ---
- name: Upgrade 4221
  import_playbook: router-playbooks/upgrade_4221.yml
- name: Upgrade 2901
  import_playbook: router-playbooks/upgrade_2901.yml
- name: Upgrade 2960
  import_playbook: switch-playbooks/upgrade_c2960.yml
- name: Upgrade 2960 Plus
  import_playbook: switch-playbooks/upgrade_c2960plus.yml
- name: Upgrade 3650
  import_playbook: multisw-playbooks/upgrade_c3650.yml
```

Overføre filer

Et av målene satt i oppgaven var å overføre lab-filer med konfigurasjon som studenter vil kunne bruke i *running-config*. Ved å bruke `Ansible.netcommon.net_put`-modulen vil man kunne overføre filene med dd på Cisco Catalyst 3650 og SCP til resten av nettverksutstyret. Kodelisten 4.30 viser hvordan man overfører filen *R1.cfg* til en ny mappe som er navngitt *Lab1*. For at det ikke skal oppstå feil ved overføringen vil den slette en mappe med likt navn hvis den eksisterer med `Ansible.netcommon.net_cli`-modulen, ellers vil ikke *Playbooken* kjøres. Den vil så bruke `Ansible.netcommon.net_cli`-modulen til å lage en mappe og så bruke `Ansible.netcommon.net_put` for å kopiere filer over til den nye mappen som er opprettet. For å eksekvere kodeliste 4.30 benyttes `ansible-playbook send-lab-main.yml`.

Kodeliste 4.30: *Playbook* for å overføre filer til en mappe

```
---
- name: Send Lab filer
  hosts: Cisco2901ISR
  gather_facts: false
  vars:
    mappe: Lab1
    fil: ~/Labfiler/Lab1/R1.cfg ## Skriv sti til filen
    flash: flash0

  tasks:
    - name: opprydning
      Ansible.netcommon.cli_command:
        command: 'delete /force /recursive {{ flash }}:/{{ mappe }}'

    - name: Lag mappe
      Ansible.netcommon.cli_command:
        command: 'mkdir {{ flash }}:/{{ mappe }}/'
        prompt:
```

```
19            - '[{{ mappe }}]?'
20          answer:
21            - '{{ mappe }}'
22
23      - name: Kopiere fil til mappe
24        Ansible.netcommon.net_put:
25          src: '{{ mappe }}/{{ fil }}'
26          dest: '{{ flash }}:/{{ mappe }}/{{ fil }}'
27        vars:
28          Ansible_command_timeout: 600
```

For å kunne utforme et fullstendig laboppsett til alle modellvarianter, ble det benyttet en *Playbook* som knyttet alle *Playbookene* sammen (se kodeliste 4.31). For å eksekvere playbooken benyttes `ansible-playbook send-lab-main.yml`.

Kodeliste 4.31: Ansible: *Playbook* med alle *playbookene* for å overføre filer knyttet sammen

```
1 ---
2 - name: Send Lab til 2901
3   import_playbook: router-playbooks/send_lab_2901.yml
4 - name: Send Lab til 4221
5   import_playbook: router-playbooks/send_lab_4221.yml
6 - name: Send Lab til svitsjer
7   import_playbook: switch-playbooks/send_lab_switch.yml
8 - name: Send Lab til L3-svitsjer
9   import_playbook: multisw-playbooks/send_lab_c3650.yml
```

Opprydning

Etter endt konfigurering skal nettverksutstyret være i samme tilstand som ved oppstart, det vil si at all konfigurasjon skal være nullstilt. Dette kan løses ved å benytte en *Playbook* som målretter seg på å slette running-config og startup-config, og restarter nettverksutstyret. Koden i kodeliste 4.32 tar først å sletter konfigurasjon i med tasken på linje 7-12, som sletter og svarer på bekreftelsen. Deretter i linje 17 sendes `reload`-kommandoen, siden det kan være flere *prompts* som dukker opp når `reload`-kommandoen sendes. Benyttes `check_all: True` som vil gjøre at bekreftelse vil skrevet ut til rett ledetekst.

Kodeliste 4.32: Ansible: *Playbook* for å slette konfigurasjon og restarte nettverksutstyr

```
1  ---
2  - name: Rydde opp
3    hosts: CiscoCatalyst3650,Cisco2901ISR,CiscoCatalyst2960p,CiscoCatalyst2960,
         Cisco4221ISR
4    gather_facts: false
5
6    tasks:
7    - name: "tilbakestill nvram:"
8      cisco.ios.ios_command:
9        commands:
10       - command: 'erase nvram: '
11         prompt: '[confirm]'
12         answer: 'y'
```

```
13
14
15    - name: restart nettverks utstyr
16      ansible.netcommon.cli_command:
17        command: 'reload '
18        check_all: True
19        prompt:
20          - save?
21          - confirm
22        answer:
23          - n
24          - y
```

4.4 Kombinering av kode

For å kjøre Python-skriptene og Ansible med én kommando, ble det laget et felles skript som gjør dette. Skriptet, som er vist i kodeliste 4.33, tar en variabel, `$1`, som skal være stien til *Playbooken* som skal kjøres. Linjene 4-5 eksekverer Python-skriptene som vil klargjøre vertsfilen *hosts.ini* for å kunne kjøre en Playbook. På linje 3 sjekkes det for at den spesifiserte Playbooken eksisterer. Hvis en fil ikke eksisterer vil Python-skriptene bli forhindret fra å kjøre, eller Playbookene kjøres og det blir sendt ut en melding at Playbooken ikke ble funnet.

Kodeliste 4.33: Skript: kjor.sh

```
1  #!/bin/bash
2
3  if test -f $1 && ! [ -z "$1" ]; then
4          sudo python hentMacIp.py -t 10.10.0.0/24
5          python ansiblePrep.py
6          ansible-playbook $1
7  else
8          echo "Playbook ble ikke funnet, vennligst spesifiser hvilken playbook som
               skal kjøres"
9  fi
```

4.5 Brukermanual

En fullstendig bruksmanual for prototypen kan finnes i vedlegg E. Denne gir en steg for steg gjennomgang for bruk og installasjon av protypen. I tillegg gir den pekepinner for feilsøking om nødvendig.
Bruksmanualen tar utgangspunkt i en fabrikkinistillt Ubuntu maskin, men det er ikke nødvendig for å bruke prototypen.

Kapittel 5

Testing

Under utviklingen av prototypen ble det utført kontinuerlig testing for å forsikre at prototypen fungerer som tiltenkt. Dette ble gjort både for kvalitetssikring og for å forsikre om at hvert ledd i prototypen fungerer som tiltenkt.

5.1 Testmiljø

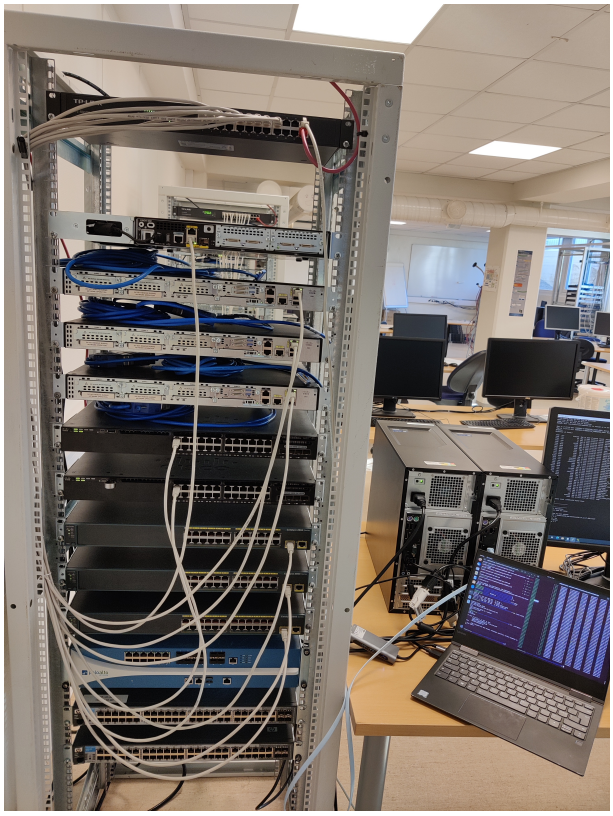
Testingen ble hovedsakelig utført på hele Cisco-laben, men for å teste *Playbookene* som oppdaterte operativsystemene ble testingen avgrenset først til pod-4 (vedlegg A.1) samt Cisco Catalyst 2960 Plus i pod-9 (vedlegg A.1), dette kalles testoppsett 1. Dette ble gjort for å forhindre at alt nettverksutstyr blir påvirket dersom noe går galt under testing. Figur 5.1 vises et testoppsett på pod-4.

Etter å gjennomgått alt nettverksutstyr viste det seg at pod-8 (vedlegg A.1) var uten internett-tilgang, og pd1ps8 og pd1ps9 (vedlegg A.1) Cisco Catalyst 2960 Plus-svitsjenes flashminne var korrupt.

Testing av skriptene ble utført på nettverksutstyret som fungerte, kalt testoppsett 2, med unntak av oppdatering av Cisco 4221 ISR pga mangel av ny IOS-image.

5.2 Testmetodikk

Alle elementene av prototypen ble først testet hver for seg under utviklingsfasen. Når de individuelle testene for hver teknologi var fullført, begynte integrasjonstestene for å forsikre at teknologiene fungerer sammen. Testingen ble utført på en datamaskin med Ubuntu versjon 22.04 LTS. Alle avhengigheter og programvare ble deretter lastet ned. Videre ble alt nettverksutstyret i Cisco-laben restartet for å også teste Cisco AutoInstall. Integrasjonstesten ble utført på hele Cisco-laben med unntak av Playbooken som oppdaterer operativsystemene på nettverksutstyret. Dette ble først utført på pod-4 med Cisco Catalyst 2960 Plus på pod-9.



Figur 5.1: Testmiljø: Test av prototype

5.3 Resultater

Resultatene gitt i denne delen baseres etter sekvensene de skal bli utført i prototypen, først med TFTP og Cisco AutoInstall, deretter python koden som finner og lager verts-filen *hosts.ini* og til slutt Ansible-Playbookene som eksekveres. For fullstendige resultater se vedlegg C.

5.3.1 TFTP og Cisco AutoInstall

I løpet av den første testen var det noe nettverksutstyr som ikke ble skikkelig konfigurert, men årsakene til dette var at nettverksutstyret hadde eksisterende konfigurasjoner, eller ble innstilt til å ignorere startup-config. Pod-8 fungerte heller ikke på grunn av manglende internett-tilgang i poden, samt to av Cisco Catalyst 2960 Plus, pd1ps8 og pd1ps9, på pod-1 hadde korruptert flash-minne.

Python

Etter å ha eksekvert *kjor.sh* med en Ansible *Playbook* får man skannet lokalnettverket i Cisco-laben og opprettet inventar-filen *hosts.ini* med vertene i kodelis-

te 5.1. For full utskrift se vedlegg C.1.

Kodeliste 5.1: Inventar-filen *hosts.ini* for testoppsett 1

```
1
2  [Cisco2901ISR]
3  pd1ps1 ansible_host=10.10.0.114
4  pd1ps2 ansible_host=10.10.0.111
5  pd1ps3 ansible_host=10.10.0.105
6  pd1ps4 ansible_host=10.10.0.109
7  pd4ps2 ansible_host=10.10.0.120
8  pd4ps3 ansible_host=10.10.0.121
9  pd4ps4 ansible_host=10.10.0.124
10
11 [CiscoCatalyst3650]
12 pd1ps5 ansible_host=10.10.0.161
13 pd1ps6 ansible_host=10.10.0.162
14 pd4ps5 ansible_host=10.10.0.156
15 pd4ps6 ansible_host=10.10.0.157
16
17 [CiscoCatalyst2960p]
18 pd9ps7 ansible_host=10.10.0.173
19 pd9ps8 ansible_host=10.10.0.172
20 pd9ps9 ansible_host=10.10.0.174
21
22 [Cisco4221ISR]
23 pd4ps1 ansible_host=10.10.0.73
24
25 [CiscoCatalyst2960]
26 pd4ps7 ansible_host=10.10.0.134
27 pd4ps8 ansible_host=10.10.0.135
28 pd4ps9 ansible_host=10.10.0.136
```

5.3.2 Vis versjon

For testingen av *Playbooken* kalt *vis versjon* ble alt nettverksutstyr koblet til lokalnettverket. Det er viktig å poengtere at dette ble kjørt etter TFTP-serveren og Cisco AutoInstall hadde konfigurert nettverksutstyret. For full utskrift se vedlegg C.2.

5.3.3 Oppgradere

Oppgradering ble testet først på testoppsett 1 med pod-4 og Cisco Catalyst 2960 Plus på pod-9. Til slutt ble det testet på hele Cisco-laben, og fungerte som tiltenkt. Cisco 4221 ISR-ruteren ble det ikke testet oppgradering på grunn av manglende IOS-image. For full utskrift se vedlegg C.3

Kodeliste 5.2: Redigert utskrift etter oppgradering av nettverksutstyr med testoppsett 1

```
1  PLAY RECAP *********************************************************************
2  pd4ps2                     : ok=12   changed=3    failed=0    skipped=1
3  pd4ps3                     : ok=12   changed=1    failed=0    skipped=1
4  pd4ps4                     : ok=12   changed=3    failed=0    skipped=1
5  pd4ps5                     : ok=13   changed=2    failed=0    skipped=0
6  pd4ps6                     : ok=13   changed=2    failed=0    skipped=0
7  pd4ps7                     : ok=14   changed=3    failed=0    skipped=0
```

```
pd4ps8                     : ok=14   changed=3    failed=0    skipped=0
pd4ps9                     : ok=14   changed=3    failed=0    skipped=0
pd9ps7                     : ok=14   changed=3    failed=0    skipped=0
pd9ps8                     : ok=14   changed=3    failed=0    skipped=0
pd9ps9                     : ok=14   changed=3    failed=0    skipped=0
```

5.3.4 Sende konfigurasjonsfiler

Å sende konfigurasjonsfiler ble testet på testoppsett nr. 1. På samme måte som på *vis versjon* i avsnitt 5.3.2 ble nettverksutstyret testet etter TFTP og Cisco Auto-Install var eksekvert. På kodeliste 5.3 viser at nettverksutstyr som skulle testes, sender tilbake en bekreftelse på at *Playbooken* ble utført uten feilmeldinger. Til slutt ble det sendt konfigurasjonsfiler til testoppsett nr. 2, der alt nettverksutstyr er inkludert i testingen. For full utskrift se vedlegg C.4.

Kodeliste 5.3: Redigert utskrift filer er blitt sendt til nettverksutstyret med testoppsett 1

```
PLAY RECAP *********************************************************************
pd4ps1                     : ok=3   changed=1    failed=0    skipped=0
pd4ps2                     : ok=3   changed=1    failed=0    skipped=0
pd4ps3                     : ok=3   changed=1    failed=0    skipped=0
pd4ps4                     : ok=3   changed=1    failed=0    skipped=0
pd4ps5                     : ok=3   changed=1    failed=0    skipped=0
pd4ps6                     : ok=3   changed=1    failed=0    skipped=0
pd4ps7                     : ok=3   changed=1    failed=0    skipped=0
pd4ps8                     : ok=3   changed=1    failed=0    skipped=0
pd4ps9                     : ok=3   changed=1    failed=0    skipped=0
pd9ps7                     : ok=3   changed=1    failed=0    skipped=0
pd9ps8                     : ok=3   changed=1    failed=0    skipped=0
pd9ps9                     : ok=3   changed=1    failed=0    skipped=0
```

5.3.5 Slett konfigurasjon på nettverksutstyr

Playbooken som sletter eksisterende konfigurasjon på nettverksutstyret ble testet på alt nettverksutstyr i Cisco-laben med testoppsett 2 og fungerte som planlagt, med unntak av Cisco 4221 ISR som ikke sendte konfirmasjon av reboot til Ansible, men ble restartet, se vedlegg C.5.

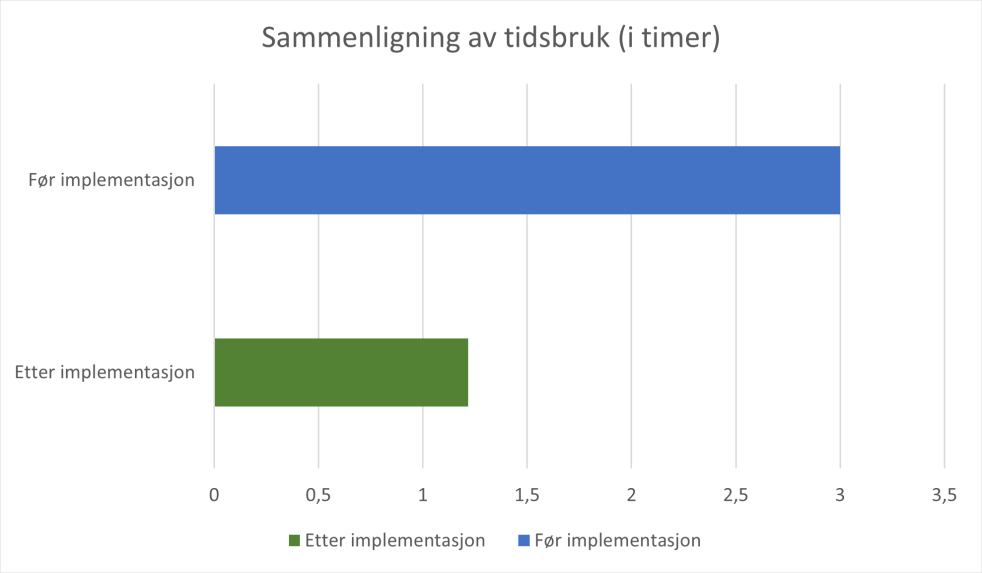
Kapittel 6

Konklusjon

Prototypen oppfyller de kravene og forventningene som ble satt i forkant av prosjektet, ved at den kan oppgradere nettverksutstyret og sende konfigurasjonsfiler fra et stadie uten konfigurasjon. Som resultat vil oppdragsgiveren ha mulighet til å teste prototypen i et reelt scenario, og dersom ønskelig ta i bruk den endelige løsningen i forespurt scenario for å automatisere og effektivisere tidsbruken i Cisco-laben.

6.1 Resultat

Hovedmålet med å automatisere nettverksutstyret i Cisco-laben var å redusere tidsbruken knyttet til forberedelse av nettverksutstyr i forkant av en laboratorieøvelse. Som resultat av prosjektet ble det utviklet en prototype som kan anvendes i Cisco-laben. Ved å benytte denne prototypen kan foreleser redusere forberedelsestiden i overkant av 59 prosent (se figur 6.1). Da har vi lagt til grunn anslaget, som ble gitt av arbeidsgiver, om at laboratorieansvarlig pluss en foreleser bruker opprinnelig 3 timer på å forberede Cisco-laben. Med prototypen kan *en person* sende en konfigurasjonsfil til nettverksutstyr, og overføringen tar maksimalt 58 minutter. I tillegg er det beregnet inn et kvarter for nettverksutstyr å oppdatere seg etter overføringen er gjennomført. Total forberedelsetid er da 1 time og 13 minutter. I mellomtiden kan foreleseren koble fra utstyr som nettverkskabler og gjøre klar Cisco-laben, som betyr mindre dødtid. I tillegg er det verdt å nevne at det opprinnelig anslaget var basert på at *to personer* oppgraderte nettverksutstyr samtidig, men med prototypen ble det gjort av en person da vi anser det som tilstrekkelig og mer anvendelig. Dette betyr økt effektivisering av arbeidsoppgavene for forelesere.



Figur 6.1: Sammenligning av tidsbruk

6.2 Videre arbeid

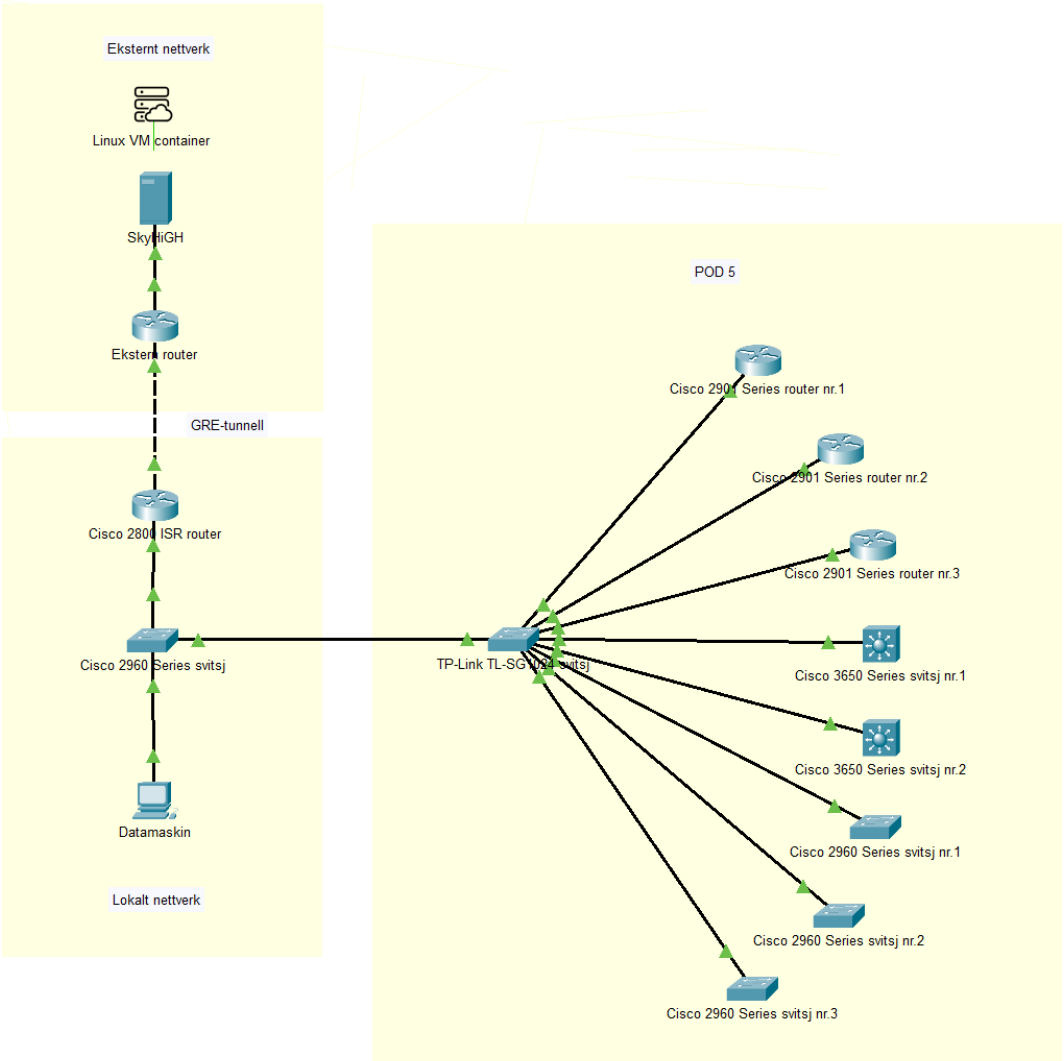
Prototypen fungerer som planlagt, men det er fremdeles forbedringer som kan gjennomføres for å videreutvikle produktet. Eksempler på dette kan være å implementere et grafisk brukergrensesnitt, konfigurere ekstern administrering og forbedre overføringshastigheten fra datamaskin til lag-3-svitsjene. Forbedringspunktene blir tatt opp i avsnittene nedenfor, og en strategi og en mulig løsning vil bli nevnt.

6.2.1 Grafisk brukergrensesnitt

Et grafisk brukergrensesnitt hadde vært til stor nytte i et potensielt ferdigstilt produkt. Med et grafisk brukergrensesnitt kan man fjerne behovet for å måtte manuelt eksekvere skript i en *Playbook*. Dette kan gjøre produktet mer intuitivt og sørge for en bedre sammensveising av de ulike teknologiene. En mulig løsning er å dra og slippe konfigurasjonsfiler til nettverksutstyret, for å gi bedre oversikt over filoverføringsprosessen. Hvis man ønsker å vite hvilket nettverksutstyr som for øyeblikket er fjernstyrt kan man også vise dette med bilder. Et grafisk brukergrensesnitt hadde bidratt både til å forenkle overføringsprosessen, holde oversikt over feil og avvik og senke vanskelighetsgraden for bruk av produktet. Et grafisk grensesnitt vil for eksempel være enklere å benytte for nyere studenter som ikke har fått innføring i mappestrukturer og kommandolinjen.

6.2.2 Ekstern administrering

Ved oppstart av oppgaven var det ønsket å opprette en GRE-tunnel for eventuell ekstern administrering av Cisco-laben. I dette tilfellet var det tenkt å opprette et testmiljø i en VM på SkyHiGH-serveren som er plassert på NTNU i Gjøvik. Denne løsningen er illustrert i figur 6.2. Dette hadde vært fint for ekstern administrering av Cisco-laben, men ble ikke gjennomført i praksis ettersom mye av testingen foregikk internt i Cisco-laben. Ved implementering av den nødvendige infrastrukturen kan man forbedre muligheten for fjernstyring utenfor Cisco-laben. Det vil fremdeles være nødvendig å fysisk koble til en Ethernet-kabel mellom nettverksutstyret og internett-svitsjen for å skape kommunikasjon.



Figur 6.2: Testoppsett med GRE-tunnel for ekstern administrering

6.2.3 Slette filer

Prototypen har blir programmert slik at den kan sende filer og IOS-imager til nettverksutstyr, men har ingen funksjonalitet for å slette filer. Dette kan være uheldig i lengden ettersom man kan risikere å fylle opp minnet på nettverksutstyret med ikke relevante konfigurasjonsfiler. Med den nåværende løsnigen må man fremdeles manuelt fjernstyre seg inn på utstyret og slette filer. Det kan være nyttig å utvikle en løsning som forenkler prossessen av å slette filer. Det kunne for eksempel blitt implementert som en del av grafisk brukergrensesnitt i avsnitt 6.2.1.

6.2.4 Forbedre hastigheten i overføring til lag-3-svitsjene

I testingsprosessen av prototypen ble det oppdaget at overføring av filer og IOS-image til lag-3-svitsjene Cisco Catalyst 3650 tok lengre tid på grunn av størrelsen på IOS-image. Ved å undersøke filoverføringsprotokoller litt nærmere kan det kanskje være mulig å finne en mulig løsning for denne modellvarianten. Blir dette forbedret kan det kutte ned den totale tiden det tar å konfigurere og oppdatere Cisco Catalyst 3650-svitsjen.

6.3 Refleksjon

Vi har hatt en god arbeidsmoral gjennom prosjektet med fastsatt arbeidstid mellom 9:15-16, mandag til fredag. Med noen unntak har vi klart å opprettholde tidsfristene som er satt. Vedlagt ligger et diagram over den totale arbeidstiden brukt gjennom prosjektet for alle gruppedeltakere. Gjennom prosessen med å utforske teknologiene, finne passende teknologi og designe prototypen har vi vært motivert og fokusert på å nå målene som er satt. Selve oppgaven var interessant og utfordrende, og vi har lært mye gjennom prosjektet.

Tildelingen av oppgavene har vært ganske klar, og fungert som forventet. Vi har oppnådd en god arbeidsfordeling som har blitt fulgt gjennom prosjektet. Vi har erfart at et ukentlig og daglig Scrum møte er til stor nytte for å sikre en forståelse av arbeidsprosessen som en helhet. I etterkant innser vi at det kunne vært fordelaktig å ha hatt mer regelmessige møter slik at vi alltid hadde konkrete oppgaver å gjennomføre.

6.4 Avslutning

Vår oppdragsgiver ville ha en helhetlig løsning som automatiserer nettverksutstyr i Cisco-laben. Vi er overbevist om at vi har levert en løsning som står til deres krav og forventninger. Som diskutert i avsnitt 1.6, konkluderer gruppen at kravene og målene satt for prosjektet er nådd. Denne oppgaven har gitt oss muligheten til å erfare dybdelæring når det gjelder de ulike teknologiene og metodikken som vi allerede er kjent med. I tillegg har vi fått lære om nye teknologier og sett hvordan disse fungerer i praksis. Oppgaven har gitt oss en god læringsopplevelse og har fått

oss til å utfordre ferdigheten våre. Gruppen er veldig fornøyd med resultatet av både prototypen og rapporten, uansett om det er enkelte aspekter som kunne ha vært forbedret (se avsnitt 6.2). Vi håper at både prototypen og rapporten vil være av verdi for oppdragsgiveren, og muligens gi inspirasjon til andre med lignende problemstillinger.

Bibliografi

[1] «Cisco 2901 Integrated Services Router.» (), adresse: https://www.cisco.com/c/en/us/support/routers/2901-integrated-services-router-isr/model.html. hentet 2022.04.27.

[2] «Cisco 4221 Integrated Services Router.» (), adresse: https://www.cisco.com/c/en/us/support/routers/4221-integrated-services-router-isr/model.html. hentet 2022.04.27.

[3] «Cisco Catalyst 2960 Series Switches.» (), adresse: https://www.cisco.com/c/en/us/support/switches/catalyst-2960-series-switches/series.html. hentet 2022.04.27.

[4] «Cisco Catalyst 3650 Series Switches.» (), adresse: https://www.cisco.com/c/en/us/support/switches/catalyst-3650-series-switches/series.html. hentet 2022.04.27.

[5] H. Øverby. «brukergrensesnitt.» (), adresse: https://snl.no/brukergrensesnitt. (hentet 2022.05.17).

[6] Cisco. «Cisco IOS Command Hierarchy.» (), adresse: https://www.cisco.com/E-Learning/bulk/public/tac/cim/cib/using_cisco_ios_software/02_cisco_ios_hierarchy.htm#:~:text=There%20are%20five%20command%20modes,IOS%20Software%20are%20hierarchically%20structured. (hentet 2022.05.19).

[7] H. Øverby. «lokalnett.» (), adresse: https://snl.no/lokalnett. (hentet 2022.05.09).

[8] H. Bothner-By. «protokoll (IT).» (), adresse: https://snl.no/protokoll_-_IT. hentet 2022.04.26.

[9] «Null Modem Cable.» (), adresse: http://www.nullmodem.com/NullModem.htm. hentet 2022.04.26.

[10] M. Aleksic. «What is SSH?» (), adresse: https://phoenixnap.com/kb/what-is-ssh. (hentet 2022.03.2).

[11] L. Fitzgibbons. «Defintion Telnet.» (), adresse: https://www.techtarget.com/searchnetworking/definition/Telnet. (hentet 2022.03.5).

[12] NTNU. «Digital infrastruktur og cybersikkerhet.» (), adresse: https://www.ntnu.no/studier/bdigsec. (hentet 2022.05.15).

[13] Cisco. «Configuration Fundamentals Configuration Guide, Cisco IOS Release 15M&T.» (), adresse: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/fundamentals/configuration/15mt/fundamentals-15-mt-book/cf-autoinstall.html. (hentet 2022.02.15).

[14] Cisco. «Modifying the Switch Boot Configuration.» (), adresse: https://www.cisco.com/en/US/docs/switches/lan/catalyst4000/7.5/configuration/guide/boot_support_TSD_Island_of_Content_Chapter.html#wp1019926. (hentet 2022.02.23).

[15] Cisco. «Embedded Event Manager Configuration Guide, Cisco IOS Release 12.4T.» (), adresse: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/eem/configuration/12-4t/eem-12-4t-book/eem-policy-cli.html. (hentet 2022.02.23).

[16] Cisco. «Cisco IOS Scripting with TCL Configuration Guide, Cisco IOS Release 12.4T.» (), adresse: https://www.cisco.com/c/en/us/td/docs/ios-xml/ios/ios_tcl/configuration/12-4t/ios-tcl-12-4t-book/nm-script-tcl.html. (hentet 2022.02.23).

[17] M. Alibi, *Ansible Quick Start Guide: Control and Monitor Infrastructures of Any Size, Physical or Virtual*. Packt Publishing, 2018, ISBN: 978-1-78953-293-7.

[18] R. Hat. «https://docs.ansible.com/ansible/latest/index.html.» (), adresse: https://docs.ansible.com/ansible/latest/index.html. (hentet 2022.03.12).

[19] M. Marschall, *Chef cookbook : master over 80 incredibly effective recipes to manage the day-to-day complications in your infrastructure*. Packt Publishing Ltd., 2017, ISBN: : 1-78646-566-3.

[20] U. K. L. Guin, *The Dispossed*. Harper & Row, 1974, ISBN: 0-06-012563-2.

[21] R. McKendrick, *Learn Ansible*. Packt Publishing Ltd., 2018, ISBN: 978-1-78899-875-8.

[22] (), adresse: https://galaxy.Ansible.com/cisco/ios. (hentet 2022.04.27).

[23] R. McKendrick, *Practical Network Automation, Second Edition*. Packt Publishing Ltd., 2018, ISBN: 978-1-78995-565-1.

[24] Puppet. «Puppet overview.» (), adresse: https://puppet.com/docs/puppet/7/puppet_overview.html. (hentet 2022.05.20.

[25] Puppet. «Man Page: puppet agent.» (), adresse: https://puppet.com/docs/puppet/7/man/agent.html. (hentet 2022.05.20).

[26] Puppet. «Welcome to Bolt.» (2022), adresse: https://puppet.com/docs/bolt/latest/bolt.html.

[27] «Supported operating systems.» (), adresse: https://puppet.com/docs/pe/2019.8/supported_operating_systems.html. (hentet 2022.05.11).

[28] Cisco. «Open IOS XE FAQ.» (), adresse: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/ios-xe/nb-06-cisco-ios-xe-faq-en.html. (hentet 2022.05.16).

[29] cisco. «Configuring a Terminal/Comm Server.» (), adresse: https://www.cisco.com/c/en/us/support/docs/dial-access/asynchronous-connections/5466-comm-server.html. (hentet 2022.04.10).

[30] Cisco. «Network Automation with Plug and Play (PnP) – Part 1.» (), adresse: https://community.cisco.com/t5/networking-blogs/network-automation-with-plug-and-play-pnp-part-1/ba-p/3658231. (hentet 2022.04.5).

[31] Cisco. «Configuring Plug and Play.» (), adresse: https://content.cisco.com/chapter.sjs?uri=%2Fsearchable%2Fchapter%2Fwww.cisco.com%2Fcontent%2Fen%2Fus%2Ftd%2Fdocs%2Fnet_mgmt%2Fprime%2Finfrastructure%2F3-0%2Fuser%2Fguide%2Fpi_ug%2Fplugandplay.html.xml. (hentet 2022.03.5).

[32] Python. «Foreword for Programming Python"(1st ed.)» (), adresse: https://www.python.org/doc/essays/foreword/. (hentet 2022.04.10).

[33] Python. «What is Python?» (), adresse: https://docs.python.org/3/faq/general.html#what-is-python. (hentet 2022.04.10).

[34] C. Community. «Reverse-Telnet?» (), adresse: https://community.cisco.com/t5/switching/reverse-telnet/td-p/2159217. (hentet 2022.02.11).

[35] P. Biondi og the Scapy community. «Credits - Scapy 2.4.5 Documentation.» (), adresse: https://scapy.readthedocs.io/en/latest/backmatter.html. (hentet 2022.05.09).

[36] Cisco. «Configuration Fundamentals Configuration Guide, Cisco IOS Release 15.0S.» (), adresse: https://www.cisco.com/c/en/us/td/docs/ios/fundamentals/configuration/guide/15_0s/cf_15_0S_book/usb_flash_keys.html. (hentet 2022.03.18).

[37] Puppet. «Working with dynamic inventory.» (mai 2022), adresse: https://docs.ansible.com/ansible/latest/user_guide/intro_dynamic_inventory.html#working-with-dynamic-inventory. (hentet 2022.05.16.

[38] Puppet. «Developing dynamic inventory.» (mai 2022), adresse: https://docs.ansible.com/ansible/latest/dev_guide/developing_inventory.html#developing-dynamic-inventory. (hentet 2022.05.16).

[39] H. P. Anvin. «index : tftp/tftp-hpa.git.» (), adresse: https://git.kernel.org/pub/scm/network/tftp/tftp-hpa.git/about/. (hentet 2022.05.16).

[40] «Ansible Configuration Settings.» ().

[41] G. Madapparambath. «8 ways to speed up your Ansible playbooks.» (), adresse: https://www.redhat.com/sysadmin/faster-ansible-playbook-execution. (hentet 2022.05.17).

[42] Ansible. «How to build your inventory.» (), adresse: https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html. (hentet 2022.05.10).

[43] Ansible. «Ansible Network Collection for Common Code (netcommon).» (), adresse: https://galaxy.ansible.com/Ansible/netcommon. (hentet 2022.05.04).

[44] S. J. Peter Sprygada. «cisco.ios.ios_facts module – Module to collect facts from remote devices.» (), adresse: https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_facts_module.html#ansible-collections-cisco-ios-ios-facts-module. (hentet 2022.05.04).

[45] P. Sprygada. «cisco.ios.ios_command module – Module to run commands on remote devices.» (), adresse: https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_command_module.html#ansible-collections-cisco-ios-ios-command-module. (hentet 2022.05.04).

[46] P. Sprygada. «cisco.ios.ios_config module – Module to manage configuration sections.» (), adresse: https://docs.ansible.com/ansible/latest/collections/cisco/ios/ios_config_module.html#ansible-collections-cisco-ios-ios-config-module. (hentet 2022.05.04).

Vedlegg A

Oversikt over nettverksutstyret i Cisco-laben

A.1 Layout i laboratoriet

Tabell A.1: Pod 1

Nr	Merke	Modell	Funksjon	Identifikator
1	TP-Link	TL-SG1024	Ekstern svitsj	N/A
1	Cisco	2901 Series	Router	pd1ps1
2	Cisco	2901 Series	Router	pd1ps2
3	Cisco	2901 Series	Router	pd1ps3
4	Cisco	2901 Series	Router	pd1ps4
1	Cisco	3650 Series	Svitsj	pd1ps5
2	Cisco	3650 Series	Svitsj	pd1ps6
1	Cisco	2960 Series	Svitsj	pd1ps7
2	Cisco	2960 Series	Svitsj	pd1ps8
3	Cisco	2960 Series	Svitsj	pd1ps9

Tabell A.2: Pod 2

Nr	Merke	Modell	Funksjon	Identifikator
2	TP-Link	TL-SG1024	Ekstern svitsj	N/A
1	Cisco	4221 Series	Router	pd2ps1
5	Cisco	2901 Series	Router	pd2ps2
6	Cisco	2901 Series	Router	pd2ps3
7	Cisco	2901 Series	Router	pd2ps4
4	Cisco	3650 Series	Svitsj	pd2ps5
5	Cisco	3650 Series	Svitsj	pd2ps6
4	Cisco	2960 Series	Svitsj	pd2ps7
5	Cisco	2960 Series	Svitsj	pd2ps8
6	Cisco	2960 Series	Svitsj	pd2ps9

Tabell A.3: Pod 3

Nr	Merke	Modell	Funksjon	Identifikator
3	TP-Link	TL-SG1024	Ekstern svitsj	N/A
2	Cisco	4221 Series	Router	pd3ps1
8	Cisco	2901 Series	Router	pd3ps2
9	Cisco	2901 Series	Router	pd3ps3
10	Cisco	2901 Series	Router	pd3ps4
6	Cisco	3650 Series	Svitsj	pd3ps5
7	Cisco	3650 Series	Svitsj	pd3ps6
7	Cisco	2960 Series	Svitsj	pd3ps7
8	Cisco	2960 Series	Svitsj	pd3ps8
9	Cisco	2960 Series	Svitsj	pd3ps9

Tabell A.4: Pod 4

Nr	Merke	Modell	Funksjon	Identifikator
4	TP-Link	TL-SG1024	Ekstern svitsj	N/A
3	Cisco	4221 Series	Router	pd4ps1
11	Cisco	2901 Series	Router	pd4ps2
12	Cisco	2901 Series	Router	pd4ps4
13	Cisco	2901 Series	Router	pd4ps5
8	Cisco	3650 Series	Svitsj	pd4ps6
9	Cisco	3650 Series	Svitsj	pd4ps7
10	Cisco	2960 Series	Svitsj	pd4ps8
11	Cisco	2960 Series	Svitsj	pd4ps9
12	Cisco	2960 Series	Svitsj	pd4ps10

Tabell A.5: Pod 5

Nr	Merke	Modell	Funksjon	Identifikator
5	TP-Link	TL-SG1024	Ekstern svitsj	N/A
4	Cisco	4221 Series	Router	pd5ps1
14	Cisco	2901 Series	Router	pd5ps2
15	Cisco	2901 Series	Router	pd5ps3
16	Cisco	2901 Series	Router	pd5ps4
10	Cisco	3650 Series	Svitsj	pd5ps5
11	Cisco	3650 Series	Svitsj	pd5ps6
13	Cisco	2960 Series	Svitsj	pd5ps7
14	Cisco	2960 Series	Svitsj	pd5ps8
15	Cisco	2960 Series	Svitsj	pd5ps9

Tabell A.6: Pod 6

Nr	Merke	Modell	Funksjon	Identifikator
6	TP-Link	TL-SG1024	Ekstern svitsj	N/A
5	Cisco	4221 Series	Router	pd6ps1
17	Cisco	2901 Series	Router	pd6ps2
18	Cisco	2901 Series	Router	pd6ps3
19	Cisco	2901 Series	Router	pd6ps4
12	Cisco	3650 Series	Svitsj	pd6ps5
13	Cisco	3650 Series	Svitsj	pd6ps6
16	Cisco	2960 Series	Svitsj	pd6ps7
17	Cisco	2960 Series	Svitsj	pd6ps8
18	Cisco	2960 Series	Svitsj	pd6ps9

Tabell A.7: Pod 7

Nr	Merke	Modell	Funksjon	Identifikator
7	TP-Link	TL-SG1024	Ekstern svitsj	N/A
6	Cisco	4221 Series	Router	pd7ps1
20	Cisco	2901 Series	Router	pd7ps2
21	Cisco	2901 Series	Router	pd7ps3
22	Cisco	2901 Series	Router	pd7ps4
14	Cisco	3650 Series	Svitsj	pd7ps5
15	Cisco	3650 Series	Svitsj	pd7ps6
19	Cisco	2960 Series	Svitsj	pd7ps7
20	Cisco	2960 Series	Svitsj	pd7ps8
21	Cisco	2960 Series	Svitsj	pd7ps9

Tabell A.8: Pod 8

Nr	Merke	Modell	Funksjon	Identifikator
8	TP-Link	TL-SG1024	Ekstern svitsj	N/A
23	Cisco	2901 Series	Router	pd8ps1
24	Cisco	2901 Series	Router	pd8ps2
25	Cisco	2901 Series	Router	pd8ps3
26	Cisco	2901 Series	Router	pd8ps4
16	Cisco	3650 Series	Svitsj	pd8ps5
17	Cisco	3650 Series	Svitsj	pd8ps6
22	Cisco	2960 Series	Svitsj	pd8ps7
23	Cisco	2960 Series	Svitsj	pd8ps8
24	Cisco	2960 Series	Svitsj	pd8ps9

Tabell A.9: Pod 9

Nr	Merke	Modell	Funksjon	Identifikator
9	TP-Link	TL-SG1024	Ekstern svitsj	N/A
27	Cisco	2901 Series	Router	pd9ps1
28	Cisco	2901 Series	Router	pd9ps2
29	Cisco	2901 Series	Router	pd9ps3
30	Cisco	2901 Series	Router	pd9ps4
18	Cisco	3650 Series	Svitsj	pd9ps5
19	Cisco	3650 Series	Svitsj	pd9ps6
25	Cisco	2960 Series	Svitsj	pd9ps7
26	Cisco	2960 Series	Svitsj	pd9ps8
27	Cisco	2960 Series	Svitsj	pd9ps9

Vedlegg B

Kode

B.1 Python-kode

B.1.1 ansiblePrep

Kodeliste B.1: ansiblePrep.py

```
1  from ctypes import sizeof
2  from pickle import TRUE
3  from queue import Empty
4  import re
5  #Sjekker om input er skrevet som en gylidg MAC-adresse.
6  def ErStringMACAdr(input):
7      #Definerer mac adr sitt mønster
8      macAdrPattern = re.compile(r'(?:[0-9a-fA-F]:?){12}')
9      if re.match(macAdrPattern,input):
10         return True
11     else:
12         return False
13
14 def FinnesAdresse(MACadr):
15
16     #Lager et filobjekt som åpner og leser i filen maskinListe
17     fil = open("maskinListe","r")
18     #Looper gjennom hele filen
19     i = True
20     while i:
21         lesfil = fil.readline()
22
23         #Hvis vi finner en match til MAC-adressen, returner true
24         if lesfil == (MACadr.strip())+'\n':
25             return True
26         #Sjekker om vi har nådd slutten på filen.
27         if not lesfil:
28             i = False
29     #lukker fil
30     fil.close()
31     #Sier ifra at vi ikke fant matchende MAC-adr
32     return False
33
34
35 #Forbereder filen AnsibleIpList.
```

```
#Lager liste over alle maskinene, modellvarianten og IP-adressen som assosiert med nettverksutstyret.
def FilprepForAnsible():
    fil = open("maskinListe","r")

    #Liste for å registrere de forskjellige modellvariantene
    modellListe = []
    #Bibliotek for midlertidig oppbevaring av info som skal skriver under modeltyper.
    maskinInfoListe = []
    #Loop gjennom hele filen
    i = True
    while i:

        #Hvis vi kommer til nytt innslag av en maskin, legg i tempOppbevaring
        podKey = fil.readline()
        podKey = ryddOppLinjeskift(podKey)

        posisjonKey = fil.readline()
        posisjonKey = ryddOppLinjeskift(posisjonKey)

        modell = fil.readline()
        modell = ryddOppLinjeskift(modell)

        #Sjekker om denne modellen har blitt registrert før i funksjonen, hvis ikke legg den til i modellListe
        eksisterer = False
        for x in modellListe:
            if x == modell:
                eksisterer = True
        if (eksisterer == False):
            modellListe.append(modell)
            maskinInfoListe.append([]) #Lager plass for den nye modellen


        #leser in alle MAC-adressene
        #Stanser når vi når linjeskift eller slutt på filen
        ip =""
        e = True
        while e:
            lesfil = fil.readline()
            if ErStringMACAdr((lesfil)):
                if ErAdrInMACtxt(lesfil):
                    ip = HentIpFraMac(lesfil)
                    ip = ryddOppLinjeskift(ip)
            else:
                e = False

        navn = (podKey + posisjonKey + " ansible_host=" + ip)
        #Legger maskinen til i maskinInforListe riktig formatert med pod og posisjon kombinert.

        #Legger informasjonen om maskinen Ansible trenger til i maskinInfoListe
        #Først sjekker den hvilken modell maskinen er:
        teller = 0
        for i in modellListe:
            if i == modell:
                radNr = teller
            teller += 1
        #Så legger den til maskinen i riktig rad
```

```
92          teller = 0
93          for i in maskinInfoListe:
94              if teller == radNr:
95                      i.append(navn)
96              teller += 1
97
98          #Sjekker om vi har nådd slutten på filen.
99          if not lesfil:
100             i = False
101     fil.close()
102
103
104     #Skriver alt til filen
105     fil = open("hosts.ini","w")
106
107     #Skriver inn hver modell
108     modellNr = 0
109     bruktIp = 0
110     for i in modellListe:
111         fil.write('[' + i + ']\n')
112         rad = 0
113
114         #Skriver inn hver maskin som tilhører hver modell
115         for y in maskinInfoListe:
116             if modellNr == rad:
117                 for e in y:
118                     #Ta med modellen i filen hvis navn er lang nok til at den har
                            med ip
119                     if len(e) > 22:
120                         fil.write(e + '\n')
121                         #Tell opp Ip-er som faktisk fikk tilkobling
122                         bruktIp += 1
123             rad += 1
124         fil.write('\n')     #Legger til et linjeskift mellom hver modell
125         modellNr +=1
126
127     #Skriver variablene knytter til hver modell
128     for i in modellListe:
129         fil.write('[' + i + ':vars]\n')
130         fil.write('ansible_connection=ansible.netcommon.network_cli\n')
131         fil.write('ansible_network_os=cisco.ios.ios\n')
132         fil.write('ansible_user=cisco\n')
133         fil.write('ansible_password=cisco\n')
134         fil.write('ansible_become=yes\n')
135         fil.write('ansible_become_method=enable\n')
136         fil.write('ansible_become_password=cisco\n')
137         fil.write('\n')     #Legger til et linjeskift mellom hver modell
138
139
140     #Gir statistikk om hvor mange ip addresser som finnes, og hvor mange vi klarte
            å koble til en maskin
141     fil = open("IP","r")
142     lesfil = fil.readline()
143     totalIp = 1
144     #Teller hvor mange IP-er vi har
145     while lesfil:
146         lesfil = fil.readline()
147         if len(lesfil) > 6:
148             totalIp += 1
149     print("Totalt antall IP-er:", totalIp)
```

```
150     print("Antall tilkoblede IP-er:", bruktIp)
151     fil.close()
152 
153 #Sjekker at MAC-addressen fra input korresponderer med Mac-adr funnet med Scapy og
        lagt i MAC
154 #Returnerer den korresponderernde IP-adressen lokalisert i IP
155 def HentIpFraMac(input):
156     input = ryddOppLinjeskift(input)
157 
158     fil = open("MAC","r")
159     lesfil = "Temp"
160     #Holder øye med hvilken linje vi er på
161     teller = 1
162     radNr = 0
163     match = False
164 
165     #Leser gjennom fila til du finner en match
166     while lesfil:
167             lesfil = fil.readline()
168             lesfil = ryddOppLinjeskift(lesfil)
169             if (lesfil == input):
170                 radNr = teller
171                 match = True
172             teller += 1
173     fil.close()
174 
175     #Åpner listen over IP-adresser og returnerer IPen som sammensvarer med MAC-
            adressen fra samme linje i MAC
176     if match:
177         fil = open("IP","r")
178         for x in range(radNr):
179             lesfil = fil.readline()
180         fil.close()
181         lesfil = ryddOppLinjeskift(lesfil)
182         return lesfil
183     else:
184         return ""
185 
186 #Sjekker om MAC-adressen fra input finnes i Scapy sin MAC
187 def ErAdrInMACtxt(input):
188     fil = open("MAC","r")
189     lesfil = fil.readline()
190     lesfil = ryddOppLinjeskift(lesfil)
191     input = ryddOppLinjeskift(input)
192 
193     while lesfil:
194         if  input == (lesfil):
195             fil.close()
196             return True
197         lesfil = fil.readline()
198         lesfil = ryddOppLinjeskift(lesfil)
199 
200     fil.close()
201     return False
202 
203 #Fjerner \n og \t på slutten av variabel
204 def ryddOppLinjeskift(input):
205     input = input.replace('\n',"")
206     input = input.replace('\t',"")
207     return input
```

```
208
209 FilprepForAnsible()
```

B.1.2 macAdrBehandling

Kodeliste B.2: MacAdrBehandling.py

```
1  from importlib.util import module_for_loader
2  from pickle import FALSE
3  from pkgutil import iter_modules
4  from unicodedata import name
5  import re
6
7
8  #Sjekker om en liste allerede har en duplikat av input variabelen
9  #Returnerer true hvis den har, false hvis ikke
10 def HarListeDuplikat(liste, input):
11     for x in range(len(liste)):
12         if liste[x] == input:
13             return True
14         else:
15             return False
16
17 #Sjekker om input er skrevet som en gylidg mac-adresse.
18 def ErStringMACAdr(input):
19     #Definerer mac adr sitt mønster
20     macAdrPattern = re.compile(r'(?:[0-9a-fA-F]:?){12}')
21     if re.match(macAdrPattern,input):
22         return True
23     else:
24         return False
25
26 #Sjekker om en spesifikk MAC-adresse i biblioteket er lik MAC-adresse i filen MAC.
       txt
27 def strCmp(MACadr, MACidentifier):
28     if(MACadr == MACidentifier):
29         return True
30     else:
31         return False
32
33 #Sjekker om alle MAC-adressene i biblioteket er lik noen av MAC-adressene i filen
       MAC.txt
34 def strCmpAlle(MACadr, MACidentifier):
35     for item in MACidentifier():
36         if(MACidentifier[item] == MACadr[item]):
37             print(MACidentifier.index() + " " + MACidentifier + "\n")
38             return MACidentifier
39         else:
40             return False
41
42 #Knytter MAC-adresse til IP med index og henter IP fra IP.txt
43 def hentIP(MACidentifier, IPidentifier):
44     if(MACidentifier == True):
45         indexNr = MACidentifier.index()
46         return IPidentifier[indexNr]
47     else:
48         return False
49
```

```
#Legger til IP-adresse i maskinliste
def leggtilIP(MACidentifier, IPidentifier, MACadr):
    MACADR = MACbibliotek(MACidentifier)
    if(strCmpAlle(MACADR, MACadr)):
        fil = open("maskinliste", "a")
        for item in list():
            hentIP(MACidentifier, IPidentifier)
            if(len(item) == 12):
                if(item == ""):
                    fil.write(IPidentifier[item] +'\n')
                    print(IPidentifier[item] + '\n')
    else: print("MAC-adressene matcher ikke")

#Går gjennom og leser listen med MAC-adresser hentet med Scapy
def MACbibliotek(MACidentifier):
    #Lager et filobjekt som åpner opp listen over MAC-adresser hentet med Scapy
    with open("MAC.txt", "r") as lesfil:
        for line in lesfil:
            MACidentifier(line.rstrip())
            if lesfil:
                return MACidentifier()
            if not lesfil:
                i = False
        lesfil.close()
        return False

#Går gjennom og leser listen med IP-adresser hentet med Scapy
def IPbibliotek(IPidentifier):
    #Lager et filobjekt som åpner opp listen over MAC-adresser hentet med Scapy
    fil = open("IP.txt", "r")
    #Looper gjennom filen
    i = True
    while i:
        #Leser linje for linje
        lesfil = fil.readline()
         #Dersom filen inneholder IPidentifier + linjeskift, returner True
        if lesfil == (IPidentifier) + '\n':
            return True
        #Slutten av filen?
        if not lesfil:
            i = False
        #Lukk fil
        fil.close()
        #Returnerer False dersom MAC-adresser ikke matcher
        return False

#Sjekker om Mac-adressen allerede eksisterer i maskinListe.txt
#Returnerer true hvis den finnes, false hvis ikke
def FinnesAdresse(MACadr):

    #Lager et filobjekt som lar deg åpne, lese i filen maskinListe
    fil = open("maskinListe","r")
    #Looper gjennom hele filen
    i = True
    while i:
        lesfil = fil.readline()
        #Hvis det er tomt linjebytte. (Markerer starten av et nytt Maskin-objekt)
        #TODO: rydd opp denne. Kan brukes i andre funksoner men trengs ikke her.
            Bruktes bare for å sjekke noen ting
        #if lesfil in ['\n', '\r\n']:
```

```
109         #    print("Space")
110         #Hvis vi finner en match til MAC-adressen, returner true
111         if lesfil == (MACadr.strip())+'\n':
112             return True
113         #Sjekker om vi har nådd slutten på filen.
114         if not lesfil:
115             i = False
116     #lukker fil
117     fil.close()
118     #Sier ifra at vi ikke fant matchende MAC-adr
119     return False
120 
121 #Sjekker om kombinasjonen av pod of posisjon allerede finnes i maskinListe.txt.
122 #Returnerer true hvis den finnes, false hvis ikke
123 def FinnesMaskin(pod, posisjon):
124     fil = open("maskinListe","r")
125 
126     #Variabler som begge må bli true for å returnere true
127     podfinnes = False
128     posisjonfinnes = False
129 
130     #Looper gjennom hele filen
131     i = True
132     while i:
133         lesfil = fil.readline()
134         #Hvis vi finner en match til pod eller posisjon, marker som true
135         if lesfil == (pod)+'\n':
136             podfinnes = True
137         if lesfil == (posisjon)+'\n':
138             posisjonfinnes = True
139         #Sjekker om vi har nådd slutten på filen.
140         if not lesfil:
141             i = False
142     #lukker fil
143     fil.close()
144 
145     #Returnerer resultat. True hvis pod-posisjon kombo allerede finnes, false hvis
          ikke
146     if (podfinnes and posisjonfinnes):
147         return True
148     else:
149         return False
150 
151 #Legger til ny ny maskin i maskinListe.txt
152 def LeggTilMaskin():
153 
154     #Sjekk at kombinasjonen av pod og posisjon ikke er brukt allerede
155     i = True
156     while i:
157         pod = "pd" + (input("Hvilket Pod-nr?: "))
158         posisjon = "ps" + (input("Hvilket radnr i pod?: "))
159         #Hvis de prøver å legge til på en pod og posisjon som allerede finnes, gi
              feil og få dem til å gjøre dem på nytt
160         if FinnesMaskin(pod, posisjon):
161             print("Det er allerede en maskin på den posisjonen. Slett den
                  eksisterende maskinen hvis du ønsker å legge til en ny maskin i pod
                  :", pod ," på posisjon:", posisjon)
162         else:
163             i = False
164     print("Eksisterende modeller er:")
```

```
165     PrintModeller()
166     modell = input("Case sensitiv. Modellnavn: ")
167 
168     #Lar bruker legge til så mange adresser de vil
169     i = True
170     MACadr = []
171     while i:
172         adr = input("MAC adresse, 1 om gangen. "'"q"'" for ferdig: ")
173         #Hvis brukeren er ferdig, avslutt Mac-adr innlesning
174         if adr == "q":
175             #Hvis du har lagt til hvertfall en addresse kan du gå videre. Hvis ikke
                    må du legge inn minst en addresse
176             if (len(MACadr) > 0):
177                 i = False
178             else:
179                 print("Du må legge til minst en adresse")
180 
181         #Hvis mac-adressen er feilformatert, gi feilmelding
182         elif not (len(adr) == 17 and ErStringMACAdr(adr)):
183             print("Ikke et gyldig MAC-adresseformat. Skal skrives i formatet
                  ??:??:??:??:??:??")
184         elif FinnesAdresse(adr):
185             print("Denne MAC-addressen er allerede registrert på en annen maskin.")
186         elif HarListeDuplikat(MACadr,adr):
187             print("Du har allerede prøvd å legge til denne addressen")
188         #Legg Mac-adr til i en liste
189         else:
190             MACadr.append(adr)
191     #Så legger du inn alle variablene: pod, posisjon, modell, macadr[] i den rekkef
          ølgen til filen maskinListe
192     fil = open("maskinListe","a")
193     fil.write('\n')
194     fil.write(pod +'\n')
195     fil.write(posisjon +'\n')
196     fil.write(modell +'\n')
197     #Loop for å gå igjennom lister
198     for x in range(len(MACadr)):
199         fil.write(MACadr[x]+'\n')
200     fil.close()
201     print("La til " + modell + " på pod " + pod + " i posisjon " + posisjon)
202 
203 #Sletter en maskin i maskinListe
204 def SlettMaskin(pod, pos):
205     fil = open("maskinListe","r")
206     #Liste med alt som skal beholdes
207     lagretInnhold = []
208 
209     #Loop gjennom hele filen og legg inn alt som skal beholdes i lagretinnhold
210     i = True
211     while i:
212 
213         #Liste søm brukes for å oppbevare informasjonen til en maskin før vi
              bestemmer om det skal beholdes elles slettes
214         tempOppbevaring = []
215         #Hvis vi kommer til nytt innslag av en maskin, legg i tempOppbevaring og
              sjekk om den skal beholdes
216             #Sjekk at det ikke er en tom linje før vi leser. Det kan tulle til
                  lesning av fil
217 
218         podKey = fil.readline()
```

```
219         tempOppbevaring.append(podKey)
220 
221         posisjonKey = fil.readline()
222         tempOppbevaring.append(posisjonKey)
223 
224         modell = fil.readline()
225         tempOppbevaring.append(modell)
226 
227         #leser in alle mac addressene og legger dem i tempOppbevaring.
228         #Stanser når vi når linjeskift eller slutt på filen
229         e = True
230         while e:
231             lesfil = fil.readline()
232             if ErStringMACAdr((lesfil[0:17])):
233                 tempOppbevaring.append(lesfil)
234             else:
235                 e = False
236 
237         #legger in '\'n'
238 
239         if lesfil:
240             tempOppbevaring.append(lesfil)
241 
242 
243         #Sjekker om maskinen er markert som slettes. er den det sier den ifra.
                ellers legges til i det som skal lagres
244         #Hvis input = nr
245         if ((podKey == (pod +'\n')) and (posisjonKey == (pos +'\n'))):
246             print("Sletter: ",modell, " På pod: ", podKey, " plass: ",posisjonKey )
247         #Hvis input = pd/ps i tilegg til nr
248         elif ((podKey == ('pd' + pod +'\n')) and (posisjonKey == ('ps' + pos +'\n')
                )):
249             print("Sletter: ",modell, " På pod: ", podKey, " plass: ",posisjonKey )
250         else:
251             lagretInnhold.extend(tempOppbevaring)
252 
253             #Sjekker om vi har nådd slutten på filen.
254         if not lesfil:
255             i = False
256     fil.close()
257 
258     #Overskrive den gamle filen med det som skal beholdes
259     fil = open("maskinListe","w")
260     linjeskift = False
261     for x in range(len(lagretInnhold)):
262         if not ((lagretInnhold[x] == ('\n' or '\n\t')) and linjeskift):
263             fil.write(lagretInnhold[x])
264 
265         #Passer på at vi ikke skriver in to tomme linjer etter hverandre.
266             #Da dette kan føre til leseproblemer
267         if lagretInnhold[x] == ('\n' or '\n\t'):
268             linjeskift = True
269         else:
270             linjeskift = False
271     fil.close()
272 
273 #Fjerner \n og \t på slutten av variabel
274 def ryddOppLinjeskift(input):
275     input = input.replace('\n',"")
276     input = input.replace('\t',"")
```

```
277     return input
278 
279 #Printer ut alle modelltyper som finnes is maskinliste
280 def PrintModeller():
281     fil = open("maskinListe","r")
282       #liste for å finne eksisterende modeller
283     modellListe = []
284 
285     #Loop gjennom hele filen
286     i = True
287     while i:
288 
289         #Posisjon og pod, ikke relevant, må leses forbi
290         podKey = fil.readline()
291         posisjonKey = fil.readline()
292 
293         #Infoen vi faktis bruker her. Modellnavn
294         modell = fil.readline()
295         modell = ryddOppLinjeskift(modell)
296 
297         #Legger inn modellnavne som finnes i maskinListe
298         eksisterer = False
299         for x in modellListe:
300             if x == modell:
301                 eksisterer = True
302         if (eksisterer == False):
303             modellListe.append(modell)
304 
305 
306         #looper oss forbi mac addresser
307         #Stanser når vi når linjeskift eller slutt på filen
308         e = True
309         while e:
310             lesfil = fil.readline()
311             if not ErStringMACAdr((lesfil)):
312                e = False
313 
314         #Sjekker om vi har nådd slutten på filen.
315         if not lesfil:
316             i = False
317     fil.close()
318     print(modellListe)
319 
320 
321 #Main. Meny for endringer i maskinListe
322 
323 #Så lenge ikke q(avslutt), la bruker bruke scriptet ved å loope menyen
324 fortsett = True
325 #Den sagnomsuste menyen
326 while fortsett:
327     print('\n')
328     print("1: Legg til maskin")
329     print("2: Slett maskin")
330     print("3: Sjekk om maskin finnes")
331     print("q: avslutt")
332     valg = input("Hva vil du gjøre?: ")
333 
334     if valg == '1':
335         LeggTilMaskin()
336     elif valg == '2':
```

```
337         pod = input("Hviklen pod er den i?: ")
338         pod = ryddOppLinjeskift(pod)
339         pos = input("Hviklen posisjon skal slettes?: ")
340         pos = ryddOppLinjeskift(pos)
341         SlettMaskin(pod,pos)
342     elif valg == '3':
343         pod = 'pd' + input("Hviklen pod?: ")
344         pos = 'ps' + input("Hviklen posisjon?: ")
345         print('\n')
346         if FinnesMaskin(pod, pos):
347             print("Maskinen finnes")
348         else:
349             print("Maskinen finnes ikke")
350     elif valg == 'q':
351         fortsett = False
352     else:
353         print("----------------Ikke gyldig input")
```

B.2 Ansible-kode

B.2.1 Playbook for oppgradere Cisco Catalyst 3650

Kodeliste B.3: oppgrader-c3650.yml

```
1  ---
2  # Ansible Playbook for å oppgradere Cisco IOS på C3650
3
4  - name: Oppgrader CISCO IOS Catalyst 3650
5    hosts: CiscoCatalyst3650
6    gather_facts: false
7    vars:
8      flash_name: flash
9      new_image_name: cat3k_caa-universalk9.16.12.07.SPA.bin
10     path_to_images: ~/IOS.spring2022/3650
11     upgrade_ios_version: '16.12.07'
12     old_ios_version: '16.12.05b'
13
14   tasks:
15     - name: Sjekk versjon
16       ios_facts:
17     - debug:
18         msg:
19           - "Versjon er {{ ansible_net_version }}"
20           - "Ny Versjon er {{ upgrade_ios_version }}"
21     - debug:
22         msg:
23           - "Versjon vil bli oppgradert"
24       when: ansible_net_version != upgrade_ios_version
25
26     ## Sett ip statisk for å forberede reload
27     - name: Sett ip statisk
28       cisco.ios.ios_config:
29         lines:
30           - description test interface
31           - ip address {{ ansible_ssh_host }} 255.255.255.0
32         parents: interface vlan 1
33       when: ansible_net_version != upgrade_ios_version
```

```
34
35
36      ## Kopier nytt ios image
37      - name: Kopiere over bin filen til maskin
38        ansible.netcommon.net_put:
39          src: '{{  path_to_images  }}/{{  new_image_name  }}'
40          dest: '{{ flash_name }}:{{ new_image_name }}'
41        vars:
42          ansible_command_timeout: 10000
43        when: ansible_net_version != upgrade_ios_version
44
45      ## Installer nytt IOS
46      - name: Installer Bin filen
47        ansible.netcommon.cli_command:
48          command: "request platform software package install switch all file {{
                flash_name }}:{{ new_image_name }} new auto-copy"
49        when: ansible_net_version != upgrade_ios_version
50
51      ## Reload the device
52      - name: Restart maskinen
53        ansible.netcommon.cli_command:
54          command: reload
55          check_all: true
56          prompt:
57            - Save?
58            - confirm
59          answer:
60            - y
61            - y
62
63      ## Wait for Reachability to the device
64      - name: Vent på at switch kommer tilbake på nett
65        wait_for_connection:
66          delay: 10
67
68      ## Check current image
69      - name: Sjekk IOS versjon
70        cisco.ios.ios_facts:
71      - debug:
72          msg:
73            - "Nåverende versjon er {{ ansible_net_version }}"
74
75      - name: ASSERT THAT THE IOS VERSION IS CORRECT
76        assert:
77          that:
78            - upgrade_ios_version == ansible_net_version
79      - debug:
80          msg:
81            - "Programvare oppdateringen er komplett!"
82
83
84      ## delete old version
85      - name: Slett gammel versjon
86        cli_command:
87          command: 'delete /force {{ flash_name }}:*{{ old_ios_version }}*'
```

Vedlegg C

Testresultater

C.1 Utskrift i hosts.ini etter ansiblePrep.py eksekveres

Kodeliste C.1: hosts.ini

```
[Cisco2901ISR]
pd1ps1 ansible_host=10.10.0.114
pd1ps2 ansible_host=10.10.0.109
pd1ps3 ansible_host=10.10.0.106
pd1ps4 ansible_host=10.10.0.108
pd2ps2 ansible_host=10.10.0.103
pd2ps3 ansible_host=10.10.0.153
pd2ps4 ansible_host=10.10.0.102
pd3ps2 ansible_host=10.10.0.98
pd3ps3 ansible_host=10.10.0.93
pd3ps4 ansible_host=10.10.0.147
pd4ps2 ansible_host=10.10.0.96
pd4ps3 ansible_host=10.10.0.92
pd4ps4 ansible_host=10.10.0.94
pd5ps2 ansible_host=10.10.0.137
pd5ps3 ansible_host=10.10.0.140
pd5ps4 ansible_host=10.10.0.141
pd6ps2 ansible_host=10.10.0.131
pd6ps3 ansible_host=10.10.0.132
pd6ps4 ansible_host=10.10.0.136
pd7ps2 ansible_host=10.10.0.121
pd7ps3 ansible_host=10.10.0.152
pd7ps4 ansible_host=10.10.0.122
pd9ps1 ansible_host=10.10.0.115
pd9ps2 ansible_host=10.10.0.110
pd9ps3 ansible_host=10.10.0.113
pd9ps4 ansible_host=10.10.0.111

[CiscoCatalyst3650]
pd1ps5 ansible_host=10.10.0.148
pd1ps6 ansible_host=10.10.0.150
pd2ps5 ansible_host=10.10.0.161
pd2ps6 ansible_host=10.10.0.160
pd3ps5 ansible_host=10.10.0.155
pd3ps6 ansible_host=10.10.0.158
pd4ps5 ansible_host=10.10.0.144
pd4ps6 ansible_host=10.10.0.157
```

```
pd5ps5 ansible_host=10.10.0.224
pd5ps6 ansible_host=10.10.0.225
pd6ps5 ansible_host=10.10.0.75
pd6ps6 ansible_host=10.10.0.73
pd7ps5 ansible_host=10.10.0.164
pd7ps6 ansible_host=10.10.0.163
pd9ps5 ansible_host=10.10.0.169
pd9ps6 ansible_host=10.10.0.168

[CiscoCatalyst2960p]
pd1ps7 ansible_host=10.10.0.120
pd9ps7 ansible_host=10.10.0.123
pd9ps8 ansible_host=10.10.0.124
pd9ps9 ansible_host=10.10.0.125

[Cisco4221ISR]
pd2ps1 ansible_host=10.10.0.175
pd3ps1 ansible_host=10.10.0.176
pd4ps1 ansible_host=10.10.0.174
pd5ps1 ansible_host=10.10.0.179
pd6ps1 ansible_host=10.10.0.178
pd7ps1 ansible_host=10.10.0.177

[CiscoCatalyst2960]
pd2ps7 ansible_host=10.10.0.104
pd2ps8 ansible_host=10.10.0.105
pd2ps9 ansible_host=10.10.0.101
pd3ps7 ansible_host=10.10.0.116
pd3ps8 ansible_host=10.10.0.95
pd3ps9 ansible_host=10.10.0.97
pd4ps7 ansible_host=10.10.0.89
pd4ps8 ansible_host=10.10.0.91
pd4ps9 ansible_host=10.10.0.90
pd5ps7 ansible_host=10.10.0.138
pd5ps8 ansible_host=10.10.0.142
pd5ps9 ansible_host=10.10.0.139
pd6ps7 ansible_host=10.10.0.134
pd6ps8 ansible_host=10.10.0.135
pd6ps9 ansible_host=10.10.0.133
pd7ps7 ansible_host=10.10.0.129
pd7ps8 ansible_host=10.10.0.128
pd7ps9 ansible_host=10.10.0.127

[Cisco2901ISR:vars]
ansible_connection=ansible.netcommon.network_cli
ansible_network_os=cisco.ios.ios
ansible_user=cisco
ansible_password=cisco
ansible_become=yes
ansible_become_method=enable
ansible_become_password=cisco

[CiscoCatalyst3650:vars]
ansible_connection=ansible.netcommon.network_cli
ansible_network_os=cisco.ios.ios
ansible_user=cisco
ansible_password=cisco
ansible_become=yes
ansible_become_method=enable
ansible_become_password=cisco
```

```
98
99  [CiscoCatalyst2960p:vars]
100 ansible_connection=ansible.netcommon.network_cli
101 ansible_network_os=cisco.ios.ios
102 ansible_user=cisco
103 ansible_password=cisco
104 ansible_become=yes
105 ansible_become_method=enable
106 ansible_become_password=cisco
107
108 [Cisco4221ISR:vars]
109 ansible_connection=ansible.netcommon.network_cli
110 ansible_network_os=cisco.ios.ios
111 ansible_user=cisco
112 ansible_password=cisco
113 ansible_become=yes
114 ansible_become_method=enable
115 ansible_become_password=cisco
116
117 [CiscoCatalyst2960:vars]
118 ansible_connection=ansible.netcommon.network_cli
119 ansible_network_os=cisco.ios.ios
120 ansible_user=cisco
121 ansible_password=cisco
122 ansible_become=yes
123 ansible_become_method=enable
124 ansible_become_password=cisco
```

C.2 Vis versjon

Kodeliste C.2: Utskrift etter playbook for å sende vise versjon av nettverksutstyret

```
1  ansible-playbook playbooks/show_version.yml
2
3  PLAY [Cisco show version example] ********************************************
4
5  TASK [run show version on the routers]
       ********************************************************************
6  ok: [pd6ps6]
7  ok: [pd3ps5]
8  ok: [pd3ps6]
9  ok: [pd2ps6]
10 ok: [pd7ps5]
11 ok: [pd9ps6]
12 ok: [pd3ps3]
13 ok: [pd1ps5]
14 ok: [pd4ps4]
15 ok: [pd3ps7]
16 ok: [pd5ps6]
17 ok: [pd4ps5]
18 ok: [pd1ps3]
19 ok: [pd9ps5]
20 ok: [pd2ps9]
21 ok: [pd3ps8]
22 ok: [pd2ps8]
23 ok: [pd4ps6]
24 ok: [pd2ps7]
```

```
ok: [pd1ps2]
ok: [pd5ps3]
ok: [pd1ps7]
ok: [pd9ps4]
ok: [pd2ps5]
ok: [pd4ps2]
ok: [pd7ps6]
ok: [pd9ps8]
ok: [pd7ps4]
ok: [pd1ps1]
ok: [pd1ps6]
ok: [pd9ps7]
ok: [pd5ps9]
ok: [pd5ps2]
ok: [pd1ps4]
ok: [pd5ps4]
ok: [pd9ps2]
ok: [pd3ps9]
ok: [pd2ps3]
ok: [pd6ps4]
ok: [pd3ps2]
ok: [pd4ps3]
ok: [pd7ps3]
ok: [pd4ps8]
ok: [pd5ps5]
ok: [pd9ps9]
ok: [pd9ps3]
ok: [pd6ps1]
ok: [pd6ps3]
ok: [pd6ps7]
ok: [pd7ps1]
ok: [pd7ps2]
ok: [pd6ps2]
ok: [pd6ps9]
ok: [pd3ps1]
ok: [pd5ps1]
ok: [pd2ps2]
ok: [pd9ps1]
ok: [pd5ps8]
ok: [pd2ps1]
ok: [pd7ps7]
ok: [pd5ps7]
ok: [pd7ps9]
ok: [pd4ps9]
ok: [pd6ps8]
ok: [pd7ps8]
ok: [pd4ps1]
ok: [pd4ps7]
ok: [pd3ps4]
ok: [pd2ps4]

TASK [debug] *******************************************************

ok: [pd4ps5] => {
    "msg": [
        "pd4ps5 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.
            conf-filen"
    ]
}

```

```
ok: [pd3ps6] => {
    "msg": [
        "pd3ps6 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd2ps4] => {
    "msg": [
        "pd2ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}
ok: [pd2ps2] => {
    "msg": [
        "pd2ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}
ok: [pd6ps6] => {
    "msg": [
        "pd6ps6 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd7ps5] => {
    "msg": [
        "pd7ps5 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd2ps6] => {
    "msg": [
        "pd2ps6 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd1ps5] => {
    "msg": [
        "pd1ps5 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd2ps5] => {
    "msg": [
        "pd2ps5 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}

ok: [pd1ps1] => {
    "msg": [
        "pd1ps1 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}
ok: [pd4ps6] => {
    "msg": [
        "pd4ps6 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
```

```
}
ok: [pd5ps6] => {
    "msg": [
        "pd5ps6 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd1ps6] => {
    "msg": [
        "pd1ps6 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd9ps5] => {
    "msg": [
        "pd9ps5 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd5ps5] => {
    "msg": [
        "pd5ps5 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}

ok: [pd1ps4] => {
    "msg": [
        "pd1ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}

ok: [pd3ps5] => {
    "msg": [
        "pd3ps5 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd1ps3] => {
    "msg": [
        "pd1ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}
ok: [pd7ps6] => {
    "msg": [
        "pd7ps6 WS-C3650-24PS har versjon 16.12.07 ios  og benytter flash:packages.conf-filen"
    ]
}
ok: [pd3ps4] => {
    "msg": [
        "pd3ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-universalk9-mz.SPA.157-3.M8.bin-filen"
    ]
}
ok: [pd4ps4] => {
    "msg": [
```

```
184         "pd4ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
185     ]
186 }
187 ok: [pd9ps6] => {
188     "msg": [
189         "pd9ps6 WS-C3650-24TS har versjon 16.12.07 ios  og benytter flash:packages.
                conf-filen"
190     ]
191 }
192 ok: [pd1ps2] => {
193     "msg": [
194         "pd1ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
195     ]
196 }
197 ok: [pd9ps1] => {
198     "msg": [
199         "pd9ps1 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
200     ]
201 }
202 ok: [pd4ps2] => {
203     "msg": [
204         "pd4ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
205     ]
206 }
207 ok: [pd5ps3] => {
208     "msg": [
209         "pd5ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
210     ]
211 }
212 ok: [pd4ps3] => {
213     "msg": [
214         "pd4ps3 CISCO2901/K9 har versjon 15.4(3)M2 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.154-3.M2.bin-filen"
215     ]
216 }
217 ok: [pd3ps3] => {
218     "msg": [
219         "pd3ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
220     ]
221 }
222 ok: [pd3ps2] => {
223     "msg": [
224         "pd3ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
225     ]
226 }
227 ok: [pd2ps3] => {
228     "msg": [
229         "pd2ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
230     ]
231 }
232 ok: [pd7ps2] => {
233     "msg": [
```

```
234         "pd7ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
235     ]
236 }
237 ok: [pd7ps3] => {
238     "msg": [
239         "pd7ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
240     ]
241 }
242 ok: [pd9ps2] => {
243     "msg": [
244         "pd9ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
245     ]
246 }
247 ok: [pd6ps4] => {
248     "msg": [
249         "pd6ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
250     ]
251 }
252 ok: [pd9ps9] => {
253     "msg": [
254         "pd9ps9 WS-C2960+24TC-L har versjon 15.2(7)E5 ios  og benytter flash:c2960-
                lanbasek9-mz.152-7.E5.bin-filen"
255     ]
256 }
257 ok: [pd5ps2] => {
258     "msg": [
259         "pd5ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
260     ]
261 }
262 ok: [pd6ps2] => {
263     "msg": [
264         "pd6ps2 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
265     ]
266 }
267 ok: [pd2ps9] => {
268     "msg": [
269         "pd2ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
270     ]
271 }
272 ok: [pd9ps3] => {
273     "msg": [
274         "pd9ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
275     ]
276 }
277 ok: [pd9ps4] => {
278     "msg": [
279         "pd9ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
280     ]
281 }
282 ok: [pd7ps4] => {
283     "msg": [
```

```
284         "pd7ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
285     ]
286 }
287 ok: [pd6ps3] => {
288     "msg": [
289         "pd6ps3 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
290     ]
291 }
292 ok: [pd9ps8] => {
293     "msg": [
294         "pd9ps8 WS-C2960+24TC-L har versjon 15.2(7)E5 ios  og benytter flash:c2960-
                lanbasek9-mz.152-7.E5.bin-filen"
295     ]
296 }
297 ok: [pd5ps4] => {
298     "msg": [
299         "pd5ps4 CISCO2901/K9 har versjon 15.7(3)M8 ios  og benytter flash0:c2900-
                universalk9-mz.SPA.157-3.M8.bin-filen"
300     ]
301 }
302 ok: [pd3ps8] => {
303     "msg": [
304         "pd3ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
305     ]
306 }
307 ok: [pd3ps7] => {
308     "msg": [
309         "pd3ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
310     ]
311 }
312 ok: [pd2ps7] => {
313     "msg": [
314         "pd2ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
315     ]
316 }
317 ok: [pd1ps7] => {
318     "msg": [
319         "pd1ps7 WS-C2960+24TC-L har versjon 15.2(7)E5 ios  og benytter flash:c2960-
                lanbasek9-mz.152-7.E5.bin-filen"
320     ]
321 }
322 ok: [pd9ps7] => {
323     "msg": [
324         "pd9ps7 WS-C2960+24TC-L har versjon 15.2(7)E5 ios  og benytter flash:c2960-
                lanbasek9-mz.152-7.E5.bin-filen"
325     ]
326 }
327
328 ok: [pd2ps8] => {
329     "msg": [
330         "pd2ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
331     ]
332 }
333 ok: [pd7ps8] => {
```

```
    "msg": [
        "pd7ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd4ps1] => {
    "msg": [
        "pd4ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
            universalk9_ias.16.09.02.SPA.bin-filen"
    ]
}
ok: [pd7ps9] => {
    "msg": [
        "pd7ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd4ps8] => {
    "msg": [
        "pd4ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd5ps1] => {
    "msg": [
        "pd5ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
            universalk9_ias.16.09.02.SPA.bin-filen"
    ]
}
ok: [pd3ps9] => {
    "msg": [
        "pd3ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd7ps7] => {
    "msg": [
        "pd7ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd7ps1] => {
    "msg": [
        "pd7ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
            universalk9_ias.16.09.02.SPA.bin-filen"
    ]
}
ok: [pd5ps9] => {
    "msg": [
        "pd5ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
            c2960-lanbasek9-mz.122-55.SE12.bin-filen"
    ]
}
ok: [pd3ps1] => {
    "msg": [
        "pd3ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
            universalk9_ias.16.09.02.SPA.bin-filen"
    ]
}
ok: [pd6ps1] => {
```

```
384     "msg": [
385         "pd6ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
                universalk9_ias.16.09.02.SPA.bin-filen"
386     ]
387 }
388 
389 ok: [pd4ps9] => {
390     "msg": [
391         "pd4ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
392     ]
393 }
394 
395 ok: [pd6ps7] => {
396     "msg": [
397         "pd6ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
398     ]
399 }
400 
401 ok: [pd2ps1] => {
402     "msg": [
403         "pd2ps1 ISR4221/K9 har versjon 16.09.02 ios  og benytter bootflash:isr4200-
                universalk9_ias.16.09.02.SPA.bin-filen"
404     ]
405 }
406 
407 ok: [pd5ps8] => {
408     "msg": [
409         "pd5ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
410     ]
411 }
412 
413 ok: [pd6ps8] => {
414     "msg": [
415         "pd6ps8 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
416     ]
417 }
418 
419 
420 ok: [pd5ps7] => {
421     "msg": [
422         "pd5ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
423     ]
424 }
425 ok: [pd6ps9] => {
426     "msg": [
427         "pd6ps9 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
428     ]
429 }
430 
431 ok: [pd4ps7] => {
432     "msg": [
433         "pd4ps7 WS-C2960-24TT-L har versjon 12.2(55)SE12 ios  og benytter flash:
                c2960-lanbasek9-mz.122-55.SE12.bin-filen"
434     ]
```

```
}

PLAY RECAP *********************************************************************
pd1ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd1ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

```
pd4ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

```
pd7ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

C.3 Oppgradering

Kodeliste C.3: Utskrift etter å oppgradere nettverksutstyret

```
olavast@DESKTOP-AVKC9IT:~/bachelor-v2022-automatisering-nettversklab/ansible-cisco$
     ansible-playbook upgrade-main.yml

PLAY [Upgrade CISCO IOS] ***************************************************

TASK [Sjekk nåverende IOS] *************************************
ok: [pd1ps1]
ok: [pd4ps2]
ok: [pd1ps4]
ok: [pd1ps3]
ok: [pd1ps2]
ok: [pd4ps4]
ok: [pd4ps3]

TASK [debug] ***************************************************************
ok: [pd4ps2] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd1ps2] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd1ps1] => {
```

```
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd1ps4] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd1ps3] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd4ps4] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}
ok: [pd4ps3] => {
    "msg": [
        "Current version is 15.7(3)M8",
        "Upgrade image is 15.6(3)M9",
        "Using flash0:/c2900-universalk9-mz.SPA.157-3.M8.bin"
    ]
}

TASK [debug] *******************************************************************
ok: [pd1ps2] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd1ps1] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd1ps4] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd1ps3] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd4ps2] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
```

```
}
ok: [pd4ps3] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd4ps4] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}

TASK [Sett ip] ***************************************************************
changed: [pd1ps3]
changed: [pd1ps1]
changed: [pd4ps2]
changed: [pd1ps2]
changed: [pd4ps3]
changed: [pd1ps4]
changed: [pd4ps4]

TASK [Kopiere over bin filen] ************************************************
changed: [pd4ps2]
changed: [pd1ps4]
changed: [pd1ps3]
changed: [pd4ps3]
changed: [pd1ps1]
changed: [pd1ps2]
changed: [pd4ps4]

TASK [Skift boot variablen til nytt image] ***********************************
changed: [pd4ps2]
changed: [pd1ps4]
changed: [pd1ps3]
changed: [pd4ps3]
changed: [pd1ps1]
changed: [pd1ps2]
changed: [pd4ps4]

TASK [Reload] ****************************************************************
ok: [pd1ps1]
ok: [pd4ps2]
ok: [pd1ps4]
ok: [pd1ps3]
ok: [pd1ps2]
ok: [pd4ps4]
ok: [pd4ps3]

TASK [Vent på at pd1ps1 kommer tilbake på nett]
    ***********************************
ok: [pd1ps1]
ok: [pd4ps2]
ok: [pd1ps4]
ok: [pd1ps3]
ok: [pd1ps2]
ok: [pd4ps4]
ok: [pd4ps3]

TASK [Sjekk image versjon]     ***********************************************
ok: [pd1ps1]
```

```
ok: [pd4ps2]
ok: [pd1ps4]
ok: [pd1ps3]
ok: [pd1ps2]
ok: [pd4ps4]
ok: [pd4ps3]

TASK [debug]     ************************************************************
skipping: [pd1ps1]
skipping: [pd1ps2]
skipping: [pd1ps3]
skipping: [pd1ps4]
skipping: [pd4ps2]
skipping: [pd4ps4]

TASK [Sjekke korrekt versjon]   ******************************************
ok: [pd1ps2] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd1ps3] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd1ps1] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd1ps4] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd4ps2] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd4ps4] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [debug] ************************************************************
ok: [pd1ps1] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
    ]
}
ok: [pd1ps3] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
    ]
}
ok: [pd4ps2] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
    ]
}
ok: [pd1ps4] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
```

```
    ]
}
ok: [pd1ps2] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
    ]
}
ok: [pd4ps4] => {
    "msg": [
        "Programvare oppdateringer er komplett!"
    ]
}

TASK [Slett gammel bin fil] ************************************************
ok: [pd1ps1]
ok: [pd4ps2]
ok: [pd1ps4]
ok: [pd1ps3]
ok: [pd1ps2]
ok: [pd4ps4]
ok: [pd4ps3]

PLAY [Upgrade CISCO IOS Catalyst 2960] *************************************

TASK [Sjekk Versjon] *******************************************************
ok: [pd4ps8]
ok: [pd4ps7]
ok: [pd4ps9]

TASK [debug] ***************************************************************
ok: [pd4ps8] => {
    "msg": [
        "Nåverende versjon er 12.2(55)SE12",
        "Ny versjon er 15.0(2)SE11"
    ]
}
ok: [pd4ps7] => {
    "msg": [
        "Nåverende versjon er 12.2(55)SE12",
        "Ny versjon er 15.0(2)SE11"
    ]
}
ok: [pd4ps9] => {
    "msg": [
        "Nåverende versjon er 12.2(55)SE12",
        "Ny versjon er 15.0(2)SE11"
    ]
}

TASK [debug] ***************************************************************
ok: [pd4ps7] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd4ps8] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
```

```
ok: [pd4ps9] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}

TASK [Sett ip] ************************************************************
changed: [pd4ps8]
changed: [pd4ps9]
changed: [pd4ps7]

TASK [Kopiere over bin filen] *********************************************
changed: [pd4ps7]
changed: [pd4ps9]
changed: [pd4ps8]

TASK [Skift Boot Variable til ny versjon] *********************************
changed: [pd4ps9]
changed: [pd4ps8]
changed: [pd4ps7]

TASK [Lagre config] *******************************************************
ok: [pd4ps7]
ok: [pd4ps8]
ok: [pd4ps9]

TASK [Reload] *************************************************************
ok: [pd4ps9]
ok: [pd4ps8]
ok: [pd4ps7]

TASK [Vent på at den kommer tilbake online] *******************************
ok: [pd4ps7]
ok: [pd4ps8]
ok: [pd4ps9]

TASK [Sjekk versjon] ******************************************************
ok: [pd4ps9]
ok: [pd4ps8]
ok: [pd4ps7]

TASK [debug] **************************************************************
ok: [pd4ps9] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}
ok: [pd4ps8] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}
ok: [pd4ps7] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}

TASK [Er den nye versjonen riktig] ****************************************
ok: [pd4ps8] => {
```

```
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd4ps9] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd4ps7] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [debug] ********************************************************************
ok: [pd4ps7] => {
    "msg": [
        "Programvare oppdatering er komplett!"
    ]
}
ok: [pd4ps8] => {
    "msg": [
        "Programvare oppdatering er komplett!"
    ]
}
ok: [pd4ps9] => {
    "msg": [
        "Programvare oppdatering er komplett!"
    ]
}

TASK [Slett gammel versjon] *****************************************************
ok: [pd4ps7]
ok: [pd4ps9]
ok: [pd4ps8]

PLAY [Upgrade CISCO IOS Catalyst 2960plus] **************************************

TASK [Sjekk Versjon] ************************************************************
ok: [pd1ps7]
ok: [pd9ps7]
ok: [pd9ps9]
ok: [pd9ps8]

TASK [debug] ********************************************************************
ok: [pd9ps7] => {
    "msg": [
        "Nåverende versjon er 15.2(7)E5",
        "Ny versjon er 15.0(2)SE11"
    ]
}
ok: [pd9ps9] => {
    "msg": [
        "Nåverende versjon er 15.2(7)E5",
        "Ny versjon er 15.0(2)SE11"
    ]
}
ok: [pd9ps8] => {
    "msg": [
        "Nåverende versjon er 15.2(7)E5",
        "Ny versjon er 15.0(2)SE11"
    ]
```

```
}
ok: [pd1ps7] => {
    "msg": [
        "Nåverende versjon er 15.2(7)E5",
        "Ny versjon er 15.0(2)SE11"
    ]
}

TASK [debug] ********************************************************************
ok: [pd9ps7] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd1ps7] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd9ps9] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd9ps8] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}

TASK [Sett ip] ******************************************************************
changed: [pd9ps9]
changed: [pd9ps7]
changed: [pd1ps7]
changed: [pd9ps8]

TASK [Kopiere over bin filen] ***************************************************
changed: [pd9ps9]
changed: [pd9ps8]
changed: [pd9ps7]
changed: [pd1ps7]

TASK [Skift Boot Variable til ny versjon] ***************************************
changed: [pd1ps7]
changed: [pd9ps8]
changed: [pd9ps7]
changed: [pd9ps9]

TASK [Lagre config] *************************************************************
ok: [pd1ps7]
ok: [pd9ps7]
ok: [pd9ps8]
ok: [pd9ps9]

TASK [Reload] *******************************************************************
ok: [pd9ps7]
ok: [pd1ps7]
ok: [pd9ps9]
ok: [pd9ps8]

```

```
TASK [Vent på at den kommer tilbake online] ********************************
ok: [pd1ps7]
ok: [pd9ps9]
ok: [pd9ps7]
ok: [pd9ps8]

TASK [Sjekk versjon] *******************************************************
ok: [pd1ps7]
ok: [pd9ps9]
ok: [pd9ps8]
ok: [pd9ps7]

TASK [debug] ***************************************************************
ok: [pd1ps7] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}
ok: [pd9ps7] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}
ok: [pd9ps9] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}
ok: [pd9ps8] => {
    "msg": [
        "Current version is 15.0(2)SE11"
    ]
}

TASK [Er den nye versjonen riktig] *****************************************
ok: [pd1ps7] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd9ps7] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd9ps9] => {
    "changed": false,
    "msg": "All assertions passed"
}
ok: [pd9ps8] => {
    "changed": false,
    "msg": "All assertions passed"
}

TASK [debug] ***************************************************************
ok: [pd1ps7] => {
    "msg": [
        "Software Upgrade has been completed"
    ]
}
ok: [pd9ps9] => {
    "msg": [
```

```
        "Software Upgrade has been completed"
    ]
}
ok: [pd9ps8] => {
    "msg": [
        "Software Upgrade has been completed"
    ]
}
ok: [pd9ps7] => {
    "msg": [
        "Software Upgrade has been completed"
    ]
}

TASK [Slett gammel versjon] ************************************************
ok: [pd9ps9]
ok: [pd9ps7]
ok: [pd9ps8]
ok: [pd1ps7]

PLAY [Upgrade CISCO IOS Catalyst 3650] *************************************

TASK [Sjekk versjon] *******************************************************
ok: [pd1ps5]
ok: [pd1ps6]
ok: [pd4ps6]
ok: [pd4ps5]

TASK [debug] ***************************************************************
ok: [pd4ps5] => {
    "msg": [
        "Versjon er 16.12.07",
        "Ny Versjon er 16.12.05b"
    ]
}
ok: [pd4ps6] => {
    "msg": [
        "Versjon er 16.12.07",
        "Ny Versjon er 16.12.05b"
    ]
}
ok: [pd1ps5] => {
    "msg": [
        "Versjon er 16.12.07",
        "Ny Versjon er 16.12.05b"
    ]
}
ok: [pd1ps6] => {
    "msg": [
        "Versjon er 16.12.07",
        "Ny Versjon er 16.12.05b"
    ]
}

TASK [debug] ***************************************************************
ok: [pd1ps5] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
```

```
ok: [pd4ps6] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd4ps5] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}
ok: [pd1ps6] => {
    "msg": [
        "Image is not compliant and will be upgraded"
    ]
}

TASK [Sett ip] ***********************************************************
changed: [pd1ps5]
changed: [pd4ps5]
changed: [pd1ps6]
changed: [pd4ps6]

TASK [Kopiere over bin filen] ********************************************
changed: [pd1ps6]
changed: [pd1ps5]
changed: [pd4ps6]
changed: [pd4ps5]

TASK [Installer Bin filen] ***********************************************
ok: [pd1ps6]
ok: [pd4ps5]
ok: [pd4ps6]
ok: [pd1ps5]

TASK [Reload] ************************************************************
ok: [pd1ps5]
ok: [pd1ps6]
ok: [pd4ps6]
ok: [pd4ps5]

TASK [Vent på at pd1ps5 kommer tilbake på nett] **************************
ok: [pd1ps5]
ok: [pd1ps6]
ok: [pd1ps6]
ok: [pd4ps6]

TASK [Sjekk IOS versjon] *************************************************
ok: [pd4ps6]
ok: [pd4ps5]
ok: [pd1ps5]
ok: [pd1ps6]

TASK [debug] *************************************************************
ok: [pd4ps6] => {
    "msg": [
        "Nåverende versjon er16.12.05b"
    ]
}
ok: [pd1ps5] => {
    "msg": [
```

```
629         "Nåverende versjon er16.12.05b"
630     ]
631 }
632 ok: [pd4ps5] => {
633     "msg": [
634         "Nåverende versjon er16.12.05b"
635     ]
636 }
637 ok: [pd1ps6] => {
638     "msg": [
639         "Nåverende versjon er16.12.05b"
640     ]
641 }
642
643 TASK [ASSERT THAT THE IOS VERSION IS CORRECT] ****************************
644 ok: [pd4ps6] => {
645     "changed": false,
646     "msg": "All assertions passed"
647 }
648 ok: [pd1ps5] => {
649     "changed": false,
650     "msg": "All assertions passed"
651 }
652 ok: [pd1ps6] => {
653     "changed": false,
654     "msg": "All assertions passed"
655 }
656 ok: [pd4ps5] => {
657     "changed": false,
658     "msg": "All assertions passed"
659 }
660
661 TASK [debug] *************************************************************
662 ok: [pd1ps5] => {
663     "msg": [
664         "Programvare oppdateringen er komplett!"
665     ]
666 }
667 ok: [pd4ps6] => {
668     "msg": [
669         "Programvare oppdateringen er komplett!"
670     ]
671 }
672 ok: [pd1ps6] => {
673     "msg": [
674         "Programvare oppdateringen er komplett!"
675     ]
676 }
677 ok: [pd4ps5] => {
678     "msg": [
679         "Programvare oppdateringen er komplett!"
680     ]
681 }
682
683 TASK [Slett gammel versjon] **********************************************
684 ok: [pd4ps5]
685 ok: [pd1ps5]
686 ok: [pd4ps6]
687 ok: [pd1ps6]
688
```

```
689 PLAY RECAP *********************************************************************
690 pd1ps1                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
691 pd1ps2                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
692 pd1ps3                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
693 pd1ps4                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
694 pd1ps5                      : ok=13   changed=2    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
695 pd1ps6                      : ok=13   changed=2    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
696 pd1ps7                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
697 pd4ps2                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
698 pd4ps3                      : ok=12   changed=1    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
699 pd4ps4                      : ok=12   changed=3    unreachable=0    failed=0
        skipped=1    rescued=0    ignored=0
700 pd4ps5                      : ok=13   changed=2    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
701 pd4ps6                      : ok=13   changed=2    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
702 pd4ps7                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
703 pd4ps8                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
704 pd4ps9                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
705 pd9ps7                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
706 pd9ps8                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
707 pd9ps9                      : ok=14   changed=3    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
```

C.4 Sende konfigfil

Kodeliste C.4: Utskrift etter playbook for å sende konfigfil nettverksutstyret

```
1  ansible-playbook playbooks/send-lab-main.yml
2
3  PLAY [Send Lab filer] ***************
4
5  TASK [opprydning] *******************
6  ok: [pd6ps3]
7  ok: [pd3ps4]
8  ok: [pd4ps4]
9  ok: [pd9ps2]
10 ok: [pd6ps4]
11 ok: [pd9ps1]
12 ok: [pd7ps4]
13 ok: [pd2ps3]
14 ok: [pd1ps3]
15 ok: [pd3ps3]
```

```
ok: [pd1ps1]
ok: [pd2ps2]
ok: [pd5ps3]
ok: [pd4ps2]
ok: [pd5ps4]
ok: [pd9ps4]
ok: [pd7ps3]
ok: [pd9ps3]
ok: [pd1ps2]
ok: [pd1ps4]
ok: [pd5ps2]
ok: [pd3ps2]
ok: [pd2ps4]
ok: [pd7ps2]
ok: [pd6ps2]
ok: [pd4ps3]

TASK [Lag mappe] ********************
ok: [pd1ps1]
ok: [pd2ps4]
ok: [pd4ps4]
ok: [pd2ps3]
ok: [pd3ps4]
ok: [pd2ps2]
ok: [pd6ps3]
ok: [pd1ps3]
ok: [pd5ps3]
ok: [pd9ps4]
ok: [pd5ps4]
ok: [pd7ps4]
ok: [pd7ps3]
ok: [pd3ps2]
ok: [pd6ps2]
ok: [pd4ps2]
ok: [pd9ps2]
ok: [pd9ps1]
ok: [pd1ps2]
ok: [pd1ps4]
ok: [pd3ps3]
ok: [pd6ps4]
ok: [pd7ps2]
ok: [pd9ps3]
ok: [pd5ps2]
ok: [pd4ps3]

TASK [Kopiere fil til mappe] ********
changed: [pd1ps3]
changed: [pd3ps2]
changed: [pd1ps4]
changed: [pd2ps4]
changed: [pd5ps2]
changed: [pd1ps1]
changed: [pd1ps2]
changed: [pd4ps2]
changed: [pd2ps2]
changed: [pd4ps4]
changed: [pd3ps3]
changed: [pd5ps4]
changed: [pd2ps3]
changed: [pd3ps4]
```

```
changed: [pd4ps3]
changed: [pd6ps4]
changed: [pd9ps4]
changed: [pd6ps2]
changed: [pd9ps2]
changed: [pd5ps3]
changed: [pd7ps2]
changed: [pd9ps1]
changed: [pd7ps3]
changed: [pd9ps3]
changed: [pd6ps3]
changed: [pd7ps4]

PLAY [Send Lab filer] ***************

TASK [opprydning] *******************
ok: [pd4ps1]
ok: [pd6ps1]
ok: [pd5ps1]
ok: [pd7ps1]
ok: [pd2ps1]
ok: [pd3ps1]

TASK [Lag mappe] ********************
ok: [pd4ps1]
ok: [pd2ps1]
ok: [pd6ps1]
ok: [pd3ps1]
ok: [pd5ps1]
ok: [pd7ps1]

TASK [Kopiere fil til mappe] ********
changed: [pd3ps1]
changed: [pd4ps1]
changed: [pd2ps1]
changed: [pd6ps1]
changed: [pd7ps1]
changed: [pd5ps1]

PLAY [Send Lab filer] ***************

TASK [opprydning] *******************
ok: [pd5ps7]
ok: [pd2ps7]
ok: [pd5ps9]
ok: [pd2ps8]
ok: [pd6ps9]
ok: [pd2ps9]
ok: [pd5ps8]
ok: [pd7ps9]
ok: [pd3ps8]
ok: [pd6ps8]
ok: [pd4ps7]
ok: [pd3ps9]
ok: [pd7ps7]
ok: [pd4ps9]
ok: [pd9ps7]
ok: [pd3ps7]
ok: [pd7ps8]
ok: [pd4ps8]
```

```
ok: [pd6ps7]
ok: [pd1ps7]
ok: [pd9ps9]
ok: [pd9ps8]

TASK [Lag mappe] ********************
ok: [pd9ps8]
ok: [pd3ps7]
ok: [pd7ps7]
ok: [pd4ps7]
ok: [pd6ps8]
ok: [pd7ps8]
ok: [pd5ps7]
ok: [pd6ps7]
ok: [pd2ps7]
ok: [pd2ps9]
ok: [pd6ps9]
ok: [pd5ps9]
ok: [pd2ps8]
ok: [pd1ps7]
ok: [pd9ps7]
ok: [pd4ps9]
ok: [pd9ps9]
ok: [pd4ps8]
ok: [pd3ps9]
ok: [pd7ps9]
ok: [pd5ps8]
ok: [pd3ps8]

TASK [Kopiere fil til mappe] ********
changed: [pd4ps9]
changed: [pd2ps7]
changed: [pd5ps9]
changed: [pd4ps8]
changed: [pd2ps9]
changed: [pd2ps8]
changed: [pd3ps7]
changed: [pd3ps9]
changed: [pd5ps7]
changed: [pd6ps7]
changed: [pd7ps9]
changed: [pd3ps8]
changed: [pd6ps8]
changed: [pd4ps7]
changed: [pd5ps8]
changed: [pd6ps9]
changed: [pd7ps7]
changed: [pd7ps8]
changed: [pd9ps7]
changed: [pd9ps9]
changed: [pd1ps7]
changed: [pd9ps8]

PLAY [Send Lab filer] ***************

TASK [opprydning] *******************
ok: [pd5ps6]
ok: [pd7ps5]
ok: [pd5ps5]
ok: [pd2ps6]
```

```
ok: [pd6ps6]
ok: [pd3ps5]
ok: [pd9ps6]
ok: [pd4ps5]
ok: [pd1ps5]
ok: [pd4ps6]
ok: [pd2ps5]
ok: [pd7ps6]
ok: [pd1ps6]
ok: [pd9ps5]
ok: [pd3ps6]

TASK [Lag mappe] ********************
ok: [pd3ps6]
ok: [pd5ps6]
ok: [pd1ps5]
ok: [pd4ps5]
ok: [pd1ps6]
ok: [pd6ps6]
ok: [pd5ps5]
ok: [pd4ps6]
ok: [pd3ps5]
ok: [pd7ps6]
ok: [pd2ps5]
ok: [pd2ps6]
ok: [pd7ps5]
ok: [pd9ps6]
ok: [pd9ps5]

TASK [Kopiere fil til mappe] ********
changed: [pd6ps6]
changed: [pd5ps6]
changed: [pd2ps5]
changed: [pd1ps6]
changed: [pd9ps6]
changed: [pd4ps5]
changed: [pd9ps5]
changed: [pd3ps5]
changed: [pd2ps6]
changed: [pd1ps5]
changed: [pd3ps6]
changed: [pd7ps6]
changed: [pd4ps6]
changed: [pd7ps5]
changed: [pd5ps5]

PLAY RECAP **************************
pd1ps1                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps2                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps3                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps4                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps5                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps6                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
```

```
pd1ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

```
pd5ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd6ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps7                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps8                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd7ps9                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps1                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps2                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps3                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps4                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps5                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd9ps6                     : ok=3    changed=1    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

```
pd9ps7                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd9ps8                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd9ps9                     : ok=3    changed=1    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
```

C.5 Opprydning

Kodeliste C.5: Utskrift etter playbook for å sende konfigfil nettverksutstyret

```
ansible-playbook playbooks/opprydning.yml

PLAY [Rydde opp] ***********************************************
TASK [Slett nvram] *********************************************
ok: [pd6ps9]
ok: [pd4ps6]
ok: [pd5ps9]
ok: [pd7ps8]
ok: [pd6ps3]
ok: [pd3ps2]
ok: [pd2ps6]
ok: [pd9ps5]
ok: [pd5ps4]
ok: [pd7ps7]
ok: [pd9ps2]
ok: [pd1ps4]
ok: [pd1ps6]
ok: [pd5ps5]
ok: [pd7ps3]
ok: [pd3ps5]
ok: [pd7ps9]
ok: [pd4ps5]
ok: [pd7ps6]
ok: [pd2ps8]
ok: [pd2ps7]
ok: [pd2ps9]
ok: [pd3ps8]
ok: [pd3ps7]
ok: [pd9ps9]
ok: [pd3ps9]
ok: [pd4ps8]
ok: [pd4ps9]
ok: [pd5ps7]
ok: [pd4ps7]
ok: [pd9ps8]
ok: [pd9ps7]
ok: [pd5ps8]
ok: [pd5ps3]
ok: [pd6ps8]
ok: [pd6ps7]
ok: [pd1ps7]
ok: [pd9ps6]
ok: [pd6ps2]
ok: [pd2ps3]
ok: [pd7ps5]
ok: [pd5ps6]
```

```
ok: [pd5ps2]
ok: [pd1ps1]
ok: [pd7ps2]
ok: [pd1ps5]
ok: [pd4ps4]
ok: [pd7ps4]
ok: [pd3ps4]
ok: [pd4ps2]
ok: [pd1ps3]
ok: [pd9ps4]
ok: [pd6ps4]
ok: [pd3ps6]
ok: [pd9ps3]
ok: [pd6ps6]
ok: [pd4ps3]
ok: [pd2ps5]
ok: [pd9ps1]
ok: [pd1ps2]
ok: [pd3ps3]
ok: [pd2ps2]
ok: [pd2ps4]
ok: [pd3ps1]:
ok: [pd6ps1]:
ok: [pd5ps1]:
ok: [pd4ps1]:
ok: [pd7ps1]:
ok: [pd2ps1]:

TASK [Slett og restart] ************************************
ok: [pd2ps8]
ok: [pd2ps7]
ok: [pd2ps9]
ok: [pd3ps8]
ok: [pd3ps7]
ok: [pd9ps9]
ok: [pd3ps9]
ok: [pd4ps8]
ok: [pd4ps9]
ok: [pd5ps7]
ok: [pd4ps7]
ok: [pd9ps8]
ok: [pd9ps7]
ok: [pd5ps8]
ok: [pd5ps3]
ok: [pd6ps8]
ok: [pd6ps7]
ok: [pd1ps7]
ok: [pd9ps6]
ok: [pd6ps9]
ok: [pd4ps6]
ok: [pd5ps9]
ok: [pd7ps8]
ok: [pd6ps3]
ok: [pd3ps2]
ok: [pd2ps6]
ok: [pd9ps5]
ok: [pd5ps4]
ok: [pd7ps7]
ok: [pd9ps2]
ok: [pd1ps4]
```

```
ok: [pd1ps6]
ok: [pd5ps5]
ok: [pd7ps3]
ok: [pd3ps5]
ok: [pd7ps9]
ok: [pd4ps5]
ok: [pd7ps6]
ok: [pd6ps2]
ok: [pd2ps3]
ok: [pd7ps5]
ok: [pd5ps6]
ok: [pd5ps2]
ok: [pd1ps1]
ok: [pd7ps2]
ok: [pd1ps5]
ok: [pd4ps4]
ok: [pd7ps4]
ok: [pd3ps4]
ok: [pd4ps2]
ok: [pd1ps3]
ok: [pd9ps4]
ok: [pd6ps4]
ok: [pd3ps6]
ok: [pd9ps3]
ok: [pd6ps6]
ok: [pd4ps3]
ok: [pd2ps5]
ok: [pd9ps1]
ok: [pd1ps2]
ok: [pd3ps3]
ok: [pd2ps2]
ok: [pd2ps4]
ok: [pd3ps1]:
ok: [pd6ps1]:
ok: [pd5ps1]:
ok: [pd4ps1]:
ok: [pd7ps1]:
ok: [pd2ps1]:

PLAY RECAP *********************************************
pd1ps1                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps2                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps3                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps4                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps5                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps6                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd1ps7                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd2ps1                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd2ps2                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
pd2ps3                     : ok=2    changed=0    unreachable=0    failed=0
    skipped=0    rescued=0    ignored=0
```

```
pd2ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd2ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps1                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd3ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps1                     : ok=2    changed=0    unreachable=0    failed=1
     skipped=0    rescued=0    ignored=0
pd4ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps7                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps8                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd4ps9                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps1                     : ok=2    changed=0    unreachable=0    failed=1
     skipped=0    rescued=0    ignored=0
pd5ps2                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps3                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps4                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps5                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
pd5ps6                     : ok=2    changed=0    unreachable=0    failed=0
     skipped=0    rescued=0    ignored=0
```

```
187 pd5ps7                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
188 pd5ps8                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
189 pd5ps9                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
190 pd6ps1                      : ok=2    changed=0    unreachable=0    failed=1
        skipped=0    rescued=0    ignored=0
191 pd6ps2                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
192 pd6ps3                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
193 pd6ps4                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
194 pd6ps6                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
195 pd6ps7                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
196 pd6ps8                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
197 pd6ps9                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
198 pd7ps1                      : ok=2    changed=0    unreachable=0    failed=1
        skipped=0    rescued=0    ignored=0
199 pd7ps2                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
200 pd7ps3                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
201 pd7ps4                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
202 pd7ps5                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
203 pd7ps6                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
204 pd7ps7                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
205 pd7ps8                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
206 pd7ps9                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
207 pd9ps1                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
208 pd9ps2                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
209 pd9ps3                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
210 pd9ps4                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
211 pd9ps5                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
212 pd9ps6                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
213 pd9ps7                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
214 pd9ps8                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
215 pd9ps9                      : ok=2    changed=0    unreachable=0    failed=0
        skipped=0    rescued=0    ignored=0
```

Vedlegg D

Git-filstruktur

Kodeliste D.1: GIT: Git-filstruktur

```
https://gitlab.stud.idi.ntnu.no/olavast/bachelor-v2022-automatisering-nettversklab
|-- MacAdrBehandling.py
|-- README.md
|-- ansible.cfg
|-- ansiblePrep.py
|-- hentMacIp.py
|-- kjor.sh
|-- maskinListe
|-- maskinListeBackup
|-- playbooks
|   |-- c3650-playbooks
|   |   |-- oppgrader-c3650.yml
|   |   |-- send-konfigfil-c3650.yml
|   |-- oppgrader-alle.yml
|   |-- opprydning.yml
|   |-- ruter-playbooks
|   |   |-- oppgrader-2901.yml
|   |   |-- oppgrader-4221.yml
|   |   |-- send-konfigfil-2901.yml
|   |   |-- send-konfigfil-4221.yml
|   |-- send-lab-main.yml
|   |-- svitsj-playbooks
|   |   |-- oppgrader-c2960.yml
|   |   |-- oppgrader-c2960plus.yml
|   |   |-- send-konfigfil-svitsjer.yml
|   |-- vis-versjon.yml
|-- tftp
    |-- router-confg
    |-- sshKonfig.tcl
```

Vedlegg E

Manual

Link til git

https://gitlab.stud.idi.ntnu.no/olavast/bachelor-v2022-automatisering-nettversklab

Manual

Manualen inneholder informasjon om pakkene som skal installeres og forberedelsene som gjøres i forkant.

Nødvendigheter for installasjon: Datamaskin med en Linux-distribusjon installert.

Førstegangsinstallasjon

Før du kan komme i gang og og administrere nettverksutstyr i Cisco-laboratoriet* må Ansible, TFTP og andre avhengigheter installeres på en maskin med et Linux OS. Merk: Ubuntu-distribusjonen er anbefalt da prototype-utviklingen er basert på denne. Testmiljøet kan installeres på eksempelsvis (en datamaskin med/uten liveboot fra USB, en VM eller Raspberry Pi). En forutsetning for at løsningen skal fungere er at testmiljøet må være på samme lokalnett som nettverksutstyret.

Scapy

Scapy er et Python-bibliotek som er nyttig som et verktøy for nettverksadministrering. Her benyttes verktøyet for å oppdage nettverksenheter på nettverket, og som resultat vil verktøyet skrive ut alle aktive IP-adresser og tilknyttede MAC-adresser.

For å installere Scapy-pakkebiblioteket skriv i kommandolinjen

```
sudo apt install python3-scapy
```

Ansible

For mer informasjon om installasjonsprosessen se her.

1. Installasjon av Ansible

```
sudo add-apt-repository --yes --update ppa:ansible/ansible
```

```
sudo apt install ansible
```

2. Denne må lastes ned for at Ansible skal kunne kjøre Python-skripter

```
sudo apt install python-is-python3
```

3. *Ansible galaxy collection* for å lettere konfigurere Cisco-utstyret

```
ansible-galaxy collection install cisco.ios
```

TFTP-server

1. Last ned TFTP-serveren

> sudo apt install tftpd-hpa

2. Konfigurer TFTP-serveren. Åpne filen */etc/default/tftpd-hpa* som burde se sånn ut:

```
# /etc/default/tftpd-hpa

TFTP_USERNAME="tftp"
TFTP_DIRECTORY="/srv/tftp"
TFTP_ADDRESS=":69"
TFTP_OPTIONS="--secure"
```

Her går det an å bytte ut innholdet i *TFTP_DIRECTORY* til TFTP-mappen i Git. Gjøres dette må eierskapet av mappen byttes til TFTP gruppen og brukeren og så restarte TFTP-serveren. Dette gjøres med disse kommandoene:

Kjøres i mappen TFTP-serveren bruker

> sudo chown -R tftp:tftp .

> sudo systemctl restart tftpd-hpa

Du kan også benytte mappestrukturen */srv/tftp*, men da må må filene i TFTP-mappen i Git bli overført til mappen som har valgt for TFTP-serveren.

Fremgangsmåte

Før du begynner sørg for at TFTP-serveren kjører.

1. Skru av strømmen på alle podene som skal oppdateres eller bli konfigurert
2. Koble til en ethernetkabel fra g0/0 for 2901, g0/0/0 for 4221, på 3650 g0/0/1, på 2960 og 2960 plus g0/1 til svitsjen på toppen av poddene. (Det er viktig å bruke rett ethernet port på nettverksutstyret for at oppdateringsscriptet skal fungere)
3. Skru på strømmen igjen og vent til alt utstyret har bootet opp og satt opp SSH. Dette kan ta opp til 10 minutter

Nå er nettverksutstyret klart for å bli konfigurert av Ansible. Det er laget flere *playbooks* for diverse ulike formål som bli forklart under. Den letteste måten å gjøre dette er ved å bruke bash-filen *script.sh*, som blir brukt på følgene måte:

> ./kjor.sh playbooks/<navn på playbook>

Her er en liste over de nåverende playbooks:

vis-versjon

vis versjon er en veldig enkel *playbook* som bare vil sende tilbake hvilken IOS versjon nettverksutstyret kjører på, hvilken type det er og hvilken oppstarts-image den benytter seg av.

opprydning

Denne *playbooken* vil slette oppstartskonfigurasjonen og restarte ruteren på nettverksutstyret.

oppgrader-*

Brukes til å oppdatere nettverksutstyret, for ruterne og L2-svitsjer må disse fylles inn.

`old_image_name: *`
Fyll inn navnet på den gamle .BIN-filen
`new_image_name: *`
Fyll inn navnet på den nye .BIN-filen
`path_to_images: *`
Hvor på kontrollnoden nye imaget ligger
`upgrade_ios_version: *`
Versjonsnavn på det nye imaget

På Cisco 3650 L3-svitsjene må disse variablene fylles inn

`new_image_name: *`
Fyll inn navnet på den nye .BIN-filen
`path_to_images: *`
Fyll inn navnet på den nye .BIN-filen
`upgrade_ios_version: *`
Versjonsnavn på det nye imaget
`old_ios_version: *`
Versjonsnavn på det gamle imaget

send_konfigfil-*

Brukes til å sende konfigurasjonsfiler til nettverksutstyret, variabler som må fylles inn før.

`mappe: *`
Fyll mappenavnet på destinasjonen på nettverksutstyret
`vei: *`
Hvor mappen er på kontrollnoden
`fil: *`
Filnavn på filen som skal overføres
`flash: *`
Navn på flash på de administrerte nodene

Oversikt over maskiner i Cisco-laben

På Cisco-laben har vi ni stativer der nettverksutstyret er montert. Ett stativ regnes som en pod. I filen maskinListe finnes en oversikt over alle disse maskinene. Denne listen er formatert som følger:

podnr-> skrevet som nummeret på poden + pd foran. Feks "pd8" posisjonsnr-> skrevet som hvilken rad i poden maskinene ligger, telt ovenfra og nedover, + ps. Ref. "ps2" modell-> Modellen på maskinen. Skrevet uten mellomrom for at koden skal fungere. Ref. "CiscoCatalyst2960p" MAC-adresser-> Liste med MAC-adresser som tilhører denne maskinen. Ref. "dc:eb:94:ad:4c:1a"

En tom linje indikerer at slutten på MAC-adresser er nådd, og det er mulig å begynne på en ny maskin.

maskinListeBackup er en backupliste i tilfelle maskinListe skulle bli korrupt. Dette er nødvendig ettersom MAC-adressene er lest av manuelt fra utstyret, og det vil ta unødvendig mye tid å gjøre dette dersom det kan unngås.

Legg til nytt utsyr i oversikten over maskiner i laben

For å registrere at du har koblet til en ny maskin i en pod, gjør det følgende: kjør scriptet macAdrBehandling.py velg alternativ 1: "Legg til maskin" *Det er ikke nødvendig å legge til ps/pd i podnr og posisjonnr* Skriv inn podnr. Skriv inn posisjonsnr: Kopier eksisterende modellnavn og lim inn eller skriv ditt eget hvis det er en helt ny modell. Husk, ingen mellomrom. Skriv inn MAC-adresser. En om gangen. Skriv 'q' når ferdig

Da skal maskinen ha bli lagt til. Det er anbefalt å kopiere den nye maskinen fra maskinListe til maskinListeBackup manuelt hvis du ønsker at backupen skal ha denne dataen.

Fjerne en maskin i oversikten over maskiner i laben

For å registrere at du har fjernet en maskin fra en pod, gjør det følgende: kjør scriptet macAdrBehandling.py velg alternativ 2: "Slett maskin" *Det er ikke nødvendig å legge til ps/pd i podnr og posisjonnr* Skriv inn podnr. Skriv inn posisjonsnr:

NB: Hvis det er to tomme linjer etter hverandre vil ikke nødvendigvis maskinen bli slettet. Da vil scriptet kun rette opp i mengden linjehopp. Er du usikker på om maskinen fremdeles er i lista kan du kjøre alternativ 3: "Sjekk om maskin finnes" eller bare kjøre "Slett maskin" en gang til.

Da skal maskinen ha bli fjernet. Det er anbefalt å slette den gamle maskinen fra maskinListeBackup manuelt hvis du ønsker at backupen skal registrere at denne maskien har blitt fjernet. Backupen er der i tilfelle noe går galt under endring av maskinListe, ettersom det tar lang tid å få tak i alle adressene i laben og organisere dem igjen.

Sjekke at en maskin er registrert i maskinListe

For å sjekke at en maskin finnes i maskinListe, gjør det følgende: kjør scriptet macAdrBehandling.py velg alternativ 3: "Finnes maskin" *Ikke legg til ps/pd i podnr og posisjonnr* Skriv inn podnr. Skriv inn posisjonsnr:

Forklaring av enkelte skripter

Ved endring av prototypen eller for å få bedre forståelse av funksjonene, finnes det relevant informasjon under.

madAdrBehandling.txt

Fil som brukes ved endring av oversikt over maskiner. Vær det å legge til, slette, eller se om maskinen finnes. For utdypning av hvordan å bruke dette scriptet, se i "Oversikt over maskiner i Cisco-laben" i manualen.

ansiblePrep

Denne filen lager en fil som Ansible trenger når den skal skape en SSH-tilkobling til maskinene i laben.

Feilsøking

Hvis *ansible-galaxy collection install cisco.ios* feiler kan disse kommandoene være til hjelp

```
sudo apt install python3-pip
```

```
sudo -H pip install -Iv 'resolvelib<0.6.0' sudo apt install python3-scp
```

Hvis installasjonen av pip eller Scapy feiler, kan denne kommandoen være til hjelp

```
sudo add-apt-repository universe
```

Hvis SCP-filer får feilmeldingen *Insert error message*, kan det være fordi Ansible prøver å bruke *ansible-pylibssh* som ikke fungerer med Cisco-utstyret, dette kan løses ved å avinstallere *ansible-pylibssh*.

```
pip uninstall ansible-pylibssh
```

- Relevant for konfigurasjon av nettverksutstyret på Cisco-laboratoriet ved NTNU i Gjøvik

Vedlegg F

Prosjektavtale

NTNU
Norges teknisk-naturvitenskapelige universitet

Fastsatt av prorektor for utdanning 10.12.2020

STANDARDAVTALE

om utføring av studentoppgave i samarbeid med ekstern virksomhet

Avtalen er ufravikelig for studentoppgaver (heretter oppgave) ved NTNU som utføres i samarbeid med ekstern virksomhet.

Forklaring av begrep

Opphavsrett
Er den rett som den som skaper et åndsverk har til å fremstille eksemplar av åndsverket og gjøre det tilgjengelig for allmennheten. Et åndsverk kan være et litterært, vitenskapelig eller kunstnerisk verk. En studentoppgave vil være et åndsverk.

Eiendomsrett til resultater
Betyr at den som eier resultatene bestemmer over disse. Utgangspunktet er at studenten eier resultatene fra sitt studentarbeid. Studenten kan også overføre eiendomsretten til den eksterne virksomheten.

Bruksrett til resultater
Den som eier resultatene kan gi andre en rett til å bruke resultatene, f.eks. at studenten gir NTNU og den eksterne virksomheten rett til å bruke resultatene fra studentoppgaven i deres virksomhet.

Prosjektbakgrunn
Det partene i avtalen har med seg inn i prosjektet, dvs. som vedkommende eier eller har rettigheter til fra før og som brukes i det videre arbeidet med studentoppgaven. Dette kan også være materiale som tredjepersoner (som ikke er part i avtalen) har rettigheter til.

Utsatt offentliggjøring
Betyr at oppgaven ikke blir tilgjengelig for allmennheten før etter en viss tid, f.eks. før etter tre år. Da vil det kun være veileder ved NTNU, sensorene og den eksterne virksomheten som har tilgang til studentarbeidet de tre første årene etter at studentarbeidet er innlevert.

1. Avtaleparter

Norges teknisk-naturvitenskapelige universitet (NTNU) Institutt: Informasjonssikkerhet og kommunikasjonsteknologi (IIK)
Veileder ved NTNU: Ernst Gunnar Gran e-post og tlf: ernst.g.gran@ntnu.no +4799644916
Ekstern virksomhet: NTNU gjøvik Ekstern virksomhet sin kontaktperson, e-post og tlf.: Egil Obrestad eigil.obrestad@ntnu.no 61135143
Student: Fødselsdato:
Ev. flere studenter[1]
Vilja Langseth Lauritsen Olav Andreas Strandjord Petter Jacob Brautaset Edward Cornelius Haukø Svihus

Partene har ansvar for å klarere eventuelle immaterielle rettigheter som studenten, NTNU, den eksterne eller tredjeperson (som ikke er part i avtalen) har til prosjektbakgrunn før bruk i forbindelse med utførelse av oppgaven. Eierskap til prosjektbakgrunn skal fremgå av eget vedlegg til avtalen der dette kan ha betydning for utførelse av oppgaven.

2. Utførelse av oppgave

Studenten skal utføre: (sett kryss)

Masteroppgave	
Bacheloroppgave	x
Prosjektoppgave	
Annen oppgave	

Startdato: 10.01.2022
Sluttdato: 10.06.2022

[1] Dersom flere studenter skriver oppgave i fellesskap, kan alle føres opp her. Rettigheter ligger da i fellesskap mellom studentene. Dersom ekstern virksomhet i stedet ønsker at det skal inngås egen avtale med hver enkelt student, gjøres dette.

Oppgavens arbeidstittel er: Automatisering av konfig og testoppsett i nettverkslaben

Ansvarlig veileder ved NTNU har det overordnede faglige ansvaret for utforming og godkjenning av prosjektbeskrivelse og studentens læring.

3. Ekstern virksomhet sine plikter

Ekstern virksomhet skal stille med en kontaktperson som har nødvendig faglig kompetanse til å gi studenten tilstrekkelig veiledning i samarbeid med veileder ved NTNU. Ekstern kontaktperson fremgår i punkt 1.

Formålet med oppgaven er studentarbeid. Oppgaven utføres som ledd i studiet. Studenten skal ikke motta lønn eller lignende godtgjørelse fra den eksterne for studentarbeidet. Utgifter knyttet til gjennomføring av oppgaven skal dekkes av den eksterne. Aktuelle utgifter kan for eksempel være reiser, materialer for bygging av prototyp, innkjøp av prøver, tester på lab, kjemikalier. Studenten skal klarere dekning av utgifter med ekstern virksomhet på forhånd.

Dekning av utgifter til annet enn det som er oppført her avgjøres av den eksterne underveis i arbeidet.

4. Studentens rettigheter

Studenten har opphavsrett til oppgaven[2]. Alle resultater av oppgaven, skapt av studenten alene gjennom arbeidet med oppgaven, eies av studenten med de begrensninger som følger av punkt 5, 6 og 7 nedenfor. Eiendomsretten til resultatene overføres til ekstern virksomhet hvis punkt 5 b er avkrysset eller for tilfelle som i punkt 6 (overføring ved patenterbare oppfinnelser).

I henhold til lov om opphavsrett til åndsverk beholder alltid studenten de ideelle rettigheter til eget åndsverk, dvs. retten til navngivelse og vern mot krenkende bruk.

Studenten har rett til å inngå egen avtale med NTNU om publisering av sin oppgave i NTNUs institusjonelle arkiv på Internett (NTNU Open). Studenten har også rett til å publisere

[2] Jf. Lov om opphavsrett til åndsverk mv. av 15.06.2018 § 1

oppgaven eller deler av den i andre sammenhenger dersom det ikke i denne avtalen er avtalt begrensninger i adgangen til å publisere, jf. punkt 8.

5. Den eksterne virksomheten sine rettigheter

Der oppgaven bygger på, eller videreutvikler materiale og/eller metoder (prosjektbakgrunn) som eies av den eksterne, eies prosjektbakgrunnen fortsatt av den eksterne. Hvis studenten skal utnytte resultater som inkluderer den eksterne sin prosjektbakgrunn, forutsetter dette at det er inngått egen avtale om dette mellom studenten og den eksterne virksomheten.

Alternativ a) (sett kryss) Hovedregel

x	Ekstern virksomhet skal ha bruksrett til resultatene av oppgaven

Dette innebærer at ekstern virksomhet skal ha rett til å benytte resultatene av oppgaven i egen virksomhet. Retten er ikke-eksklusiv.

Alternativ b) (sett kryss) Unntak

	Ekstern virksomhet skal ha eiendomsretten til resultatene av oppgaven og studentens bidrag i ekstern virksomhet sitt prosjekt

Begrunnelse for at ekstern virksomhet har behov for å få overført eiendomsrett til resultatene:

6. Godtgjøring ved patenterbare oppfinnelser

Dersom studenten i forbindelse med utførelsen av oppgaven har nådd frem til en patenterbar oppfinnelse, enten alene eller sammen med andre, kan den eksterne kreve retten til oppfinnelsen overført til seg. Dette forutsetter at utnyttelsen av oppfinnelsen faller inn under den eksterne sitt virksomhetsområde. I så fall har studenten krav på rimelig godtgjøring. Godtgjøringen skal fastsettes i samsvar med arbeidstakeroppfinnelsesloven § 7. Fristbestemmelsene i § 7 gis tilsvarende anvendelse.

7. NTNU sine rettigheter

De innleverte filer av oppgaven med vedlegg, som er nødvendig for sensur og arkivering ved NTNU, tilhører NTNU. NTNU får en vederlagsfri bruksrett til resultatene av oppgaven, inkludert vedlegg til denne, og kan benytte dette til undervisnings- og forskningsformål med de eventuelle begrensninger som fremgår i punkt 8.

8. Utsatt offentliggjøring

Hovedregelen er at studentoppgaver skal være offentlige.

Sett kryss

x	Oppgaven skal være offentlig

I særlige tilfeller kan partene bli enige om at hele eller deler av oppgaven skal være undergitt utsatt offentliggjøring i maksimalt tre år. Hvis oppgaven unntas fra offentliggjøring, vil den kun være tilgjengelig for student, ekstern virksomhet og veileder i denne perioden. Sensurkomiteen vil ha tilgang til oppgaven i forbindelse med sensur. Student, veileder og sensorer har taushetsplikt om innhold som er unntatt offentliggjøring.

Oppgaven skal være underlagt utsatt offentliggjøring i (sett kryss hvis dette er aktuelt):

Sett kryss		Sett dato
	ett år	
	to år	
	tre år	

Behovet for utsatt offentliggjøring er begrunnet ut fra følgende:

Dersom partene, etter at oppgaven er ferdig, blir enig om at det ikke er behov for utsatt offentliggjøring, kan dette endres. I så fall skal dette avtales skriftlig.

Vedlegg til oppgaven kan unntas ut over tre år etter forespørsel fra ekstern virksomhet. NTNU (ved instituttet) og student skal godta dette hvis den eksterne har saklig grunn for å be om at et eller flere vedlegg unntas. Ekstern virksomhet må sende forespørsel før oppgaven leveres.

De delene av oppgaven som ikke er undergitt utsatt offentliggjøring, kan publiseres i NTNUs institusjonelle arkiv, jf. punkt 4, siste avsnitt. Selv om oppgaven er undergitt utsatt offentliggjøring, skal ekstern virksomhet legge til rette for at studenten kan benytte hele eller deler av oppgaven i forbindelse med jobbsøknader samt videreføring i et master- eller doktorgradsarbeid.

9. Generelt

Denne avtalen skal ha gyldighet foran andre avtaler som er eller blir opprettet mellom to av partene som er nevnt ovenfor. Dersom student og ekstern virksomhet skal inngå avtale om konfidensialitet om det som studenten får kjennskap til i eller gjennom den eksterne virksomheten, kan NTNUs standardmal for konfidensialitetsavtale benyttes.

Den eksterne sin egen konfidensialitetsavtale, eventuell konfidensialitetsavtale den eksterne har inngått i samarbeidprosjekter, kan også brukes forutsatt at den ikke inneholder punkter i motstrid med denne avtalen (om rettigheter, offentliggjøring mm). Dersom det likevel viser seg at det er motstrid, skal NTNUs standardavtale om utføring av studentoppgave gå foran. Eventuell avtale om konfidensialitet skal vedlegges denne avtalen.

Eventuell uenighet som følge av denne avtalen skal søkes løst ved forhandlinger. Hvis dette ikke fører frem, er partene enige om at tvisten avgjøres ved voldgift i henhold til norsk lov. Tvisten avgjøres av sorenskriveren ved Sør-Trøndelag tingrett eller den han/hun oppnevner.

Denne avtale er signert i fire eksemplarer hvor partene skal ha hvert sitt eksemplar. Avtalen er gyldig når den er underskrevet av NTNU v/instituttleder.

Signaturer:

Instituttleder: Dato:
Veileder ved NTNU: Dato:
Ekstern virksomhet: Dato: 31.01.22 Eigil Obrestad
Student: se under Dato: 14.01.22
Ev. flere studenter
Edward C. H. Sridhus Vilja Langseth Lauritsen Olew A. Strandskog Petter Jacob Braathen

Vedlegg G

Timeliste

	Olav		Vilja		Petter		Edward		Felles kommentar
dato	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	
Uke 2									
10.01.2022	1		1		1		1		jobbet på arbeidskontrakt
11.01.2022	4		4	Avtalte møter med oppgavegiver og veileder	4		4		Lynkurs i bachelorskriving og spørretime
12.01.2022	2	Jobbet med å lese stoff rundt automatise cisco	6,5		6,5		6,5		lese tidligere bachelor oppgave, se på arbeids prosesser, begynne på forprosjektet/gantt
13.01.2022	4		0	Syk	4		4		Skrevet på prosjektplanen
14.01.2022	6		6		6		6		Hadde møte med arbeidsgiver og veileder for å klarere oppgaven. Vi satt oss inn i scrum og lagde scrumboard. Vi fullførte risikoanalysen i forprosjektet.
Uke 3									
17.01.2022	5		3		7		7		jobbet med forprosjekt
18.01.2022	0		3		4,5		4,5		jobbet med forprosjekt
19.01.2022	2		3		5		5		jobbe med forprosjekt
20.01.2022	6		6		5		5		gjøre ferdig forprosjekt
21.01.2022	5		5		5		5		Innhente stoff
Uke 4									
24.01.2022	6		5		6		6		Innhente Stoff
25.01.2022	6		6		6		6		Innhente Stoff
26.01.2022	5		6		7		7		Innhente Stoff
27.01.2022	7		7		7		7		Innhente Stoff
28.01.2022		Bortreist	7		7		7		Innhente Stoff
Uke 5									
31.01.2022	0	Bortreist	2	Jobbet med CONFIG_FILE	6	Skrive fremgangsmåte for Boot fra USB	6	utarbeide strukturen på rapporten (2) tftp på packet tracer(1)	sette opp listen med ting som må gjøres (3)
01.02.2022	4	Innhentet informasjon om Ansible	4		5	Undersøke Packet Tracer	5	rapport struktur (3) tftp på packet tracer(2)	
02.02.2022	7	Sjekket hva som skjedde når oppstart av nettverksutstyr	7	Konstanterte at CONFIG_FILE lønsinga ikke funka	7	Lage et forenklet simulert nettverksoppsett i Packet Tracer	7	sette opp tftp og dhcp server i labben (1) sette opp cisco 2901 til å automatisk hente konfig fra tftp server (5.5)	møte med egil (0.5)
03.02.2022	7	Ansible oppsett for å SSH inn i nettverksutstyr	7	Utforske Reveerse telnet	2	Jobbet digitalt. Lese gjennom forprosjekt + Forenklet simulert nettverksoppsett	7	tftp på 4221, 2650 og 2901 (4) teste ut på restarte en hel pod(2.5)	møte med ernst(0.5)
04.02.2022	7	Ansible med Telnet for å koble opp SSH	5	forts telnet	0	Bortreist	5	prøve å boote fra tftp (feila) (2) få svitsj til å kommunisere med tftp server(feila)(3)	
Uke 6									
07.02.2022	7	Gjennomførte tester med å SSH inn med Ansible, hadde en feil i config filen som gjorde til at jeg slet, men fikk det til til slutt	3		6	Utvikling av python-prototype	6	skrive om tftp i rapport (3)	sprint planning og review (2) lese igjennom kommentarer fra ernst (1)
08.02.2022	7	Ansible (6), Overleaf (1)	3	overleaf lesning (3)	7	Python	6	legge til ordliste og skrive om tftp kappitel (6)	
09.02.2022	7	Ansible (6)	7	Config_file (1), Overleaf (5)	6	Python	7	jobbing med å boote netverksutstyr fra tftp server(6)	møte om grupperegler/forbere delse til møte med veileder (1)
10.02.2022	7	Ansible	7	Jobbet med CONFIG_FILE som liste ideen(6)	7	Python	7	boot fra tftp(7)	møte med veileder (1)

	Olav		Vilja		Petter		Edward		Felles kommentar
dato	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	
11.02.2022	7	Ansible	7	Jobbet med CONFIG_FILE som liste ideen(6)	7	Python	7	jobbe med booting fra tftp(7)	
Uke 7									
14.02.2022	0	Syk	3	1 standup, 2 pnp	8		7	få 2lag svitsj og 3lag svitsj til å hente konfig fra tftp-server i oppstartsprosessen(7)	
15.02.2022	3	Ansible / Syk	3	3 pnp	7		7	Cisco Autoinstall	
16.02.2022	0	Seminar	0		0	Seminar	0	IØ	
17.02.2022	0	Seminar	0		0	Seminar	0	IØ	
18.02.2022	7	(2 fredag) Leste meg opp på problemer med Ansible, (5 lørdag) Løste problemer på lørdag	7	7 Pnp	5		7	Cisco Autoinstall	
Uke 8									
21.02.2022	7	Ansible	3	Scrum plan (1) Pnp notater (2)	3		7		
22.02.2022	7	Ansible	3	Tcl og EEM teori lesing	3		7		
23.02.2022	7	Ansible	7,5	TCL og EMM testing og notater	5		7		
24.02.2022	7	Ansible	7		0	STi-møte	7		
25.02.2022	7	Rapportskriving	7		0	Seminarhelg	0	korona	
Uke 9									
28.02.2022	7		3		3		0	korona	
01.03.2022	4	Terraform	3		3		0	korona	
02.03.2022	2	Seminar, (2) Terraform	1,5		0	NSO	0	IØ	
03.03.2022	2	Seminar, (2) Ansible	2		0	NSO	0	IØ	
04.03.2022	7	Terraform	7	terraform	7		7	så på puppet	
Uke 10									
07.03.2022	7	(1) Scrum Møte, Ansible	3	Scrum møtet (1) Se på fjernkontrollen av den ene poden	7	Scrum møte	7	jobbe på script som setter hostname til en unik id	
08.03.2022	7	Network Discovery	3	Undersøke bruken av fjernstyring	7	Network discovery	7	puppet(7)	
09.03.2022	7	Ansible med oppdatering av router	7	Leste om forskjellene mellom konfigurasjonsadministreringsverktøyene (7)	7	Network discovery	6	jobbe med puppet(3) skrive i rapport(3)	
10.03.2022	7	Ansible med oppdatering av router	7	Møte veileder (1) Konf adm verktøy jobb (6)	7	Network discovery	7	lese om orkestrerings verktøy(5)	(1)møte med veileder
11.03.2022	7	Ansible med oppdatering av router	7	Overleaf (7)	7	Network discovery	3	jobbe med rapport(3)	
Uke 11									
14.03.2022	7	Undersøke TCL script	5	overleaf (4)	7	Overleaf	7		diskutere hvilke metoder passer best og planlegge prototype utviklingen (3)
15.03.2022	7	(3) planlegging av metode (4) TCL script	3	Planlegging av hvilken metode å bruke(3)	7	Planlegging av metode	3		
16.03.2022	0	Seminar	0	seminar	0	Seminar	0	IØ	
17.03.2022	2	Skriving	2	Overleaf (2)	2	Rapportskriving	0	IØ	
18.03.2022	7	5 Undersøking av metoder Diskuter metoder	7	Overleaf (5) Diskuter metode (2)	7,5	Rapportskriving	7		
Uke 12									
21.03.2022	7	Scrum Møte(1), Ansible laste opp filer	7	Scrum møtet (1) Jobbet med hvordan å finne mac-addresser	7	Bestemme metode	7		
22.03.2022	7	Ansible Laste opp filer	3	Samlet MAC adresser fra pod 4,5 og 6 (3)	7	Python-script	3	skrive om autoinstall i rapport	
23.03.2022	7	Ansible Oppdatere utstyr	7	Jobbet med MAC-adr liste programmering(7)	7	Python-script	4		møte med arbeidsgiver(1)
24.03.2022	7	Ansible (3) Oppdatere utstyr, (4) Rapport	6	Jobbet med MAC-adr liste programmering(6)	0	Borte på STi-møte	7		
25.03.2022	7		6	Jobbet med MAC-adr liste	0	Borte på STi-møte	7	lage autoinstall konfigurasjon	
Uke 13									

	Olav		Vilja		Petter		Edward		Felles kommentar
dato	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	
28.03.2022	5	Ansible	7		4	Python-script	7	jobbe med oppstartskonfigurasjon en	
29.03.2022	7	Scrum Møtet, (4) Ansible Oppdatere	4	Fikset formatering i Mac-Adr lista (3) Scrum møtet (1)	7	Python-script	7	jobbe med oppstartskonfigurasjonen	
30.03.2022	7	Ansible opddatere	7		7	Python-script	6		
31.03.2022	7	Rapport	4		7	Rapport	7	jobbe med gre tunnelfra skyhigh til cisco-lab(3) jobbe med oppstartskonfigurasjon en	møte med veileder (0.5)
01.04.2022	7	Rapport	7	Prog slette amskin fra mac-adresse lista(7)	7	Rapport	7	omstrukturere rapport, skrive om autoinstall i rapport	
Uke 14									
04.04.2022	7	Scrum Møtet, (4) Rapport	7	Overlead skriving på introdukson	2	Jobbet hjemmefra	7	skrive på rapport i kravspesifikasjoner og fikse headeren	
05.04.2022	7	Rapport	4		7		2,5	skrive om kravspesifikasjonene og fikse noen errors og warnings	sprint planning 0.5
06.04.2022	7	Rapport	7		7		0		
07.04.2022	7	Rapport	4		0	Borte på STi-møte	0		
08.04.2022	7	Rapport	7		0	Borte på STi-møte	0		
Uke 15									
11.04.2022		Ferie	0	Ferie	0	Ferie	0	Ferie	
12.04.2022		Ferie	0	Ferie	0	Ferie	0	Ferie	
13.04.2022		Ferie	0	Ferie	0	Ferie	0	Ferie	
14.04.2022		Ferie	0	Ferie	0	Ferie	0	Ferie	
15.04.2022		Ferie	0	Ferie	0	Ferie	0	Ferie	
Uke 16									
18.04.2022		Ferie	0	Ferie	0	Ferie	0		
19.04.2022	7	Ansible	0	Syk	4	Rapportskriving	0		
20.04.2022	7	Rapport	0	Syk	7	Rapportskriving	0	IØ	
21.04.2022	7	Ansible	3	Møte med veilder (1) se på tilbakemedlinger (2)	7	Rette feil i rapporten	0	IØ	
22.04.2022	7	Ansibe	7	Jobbet emd samabreid mellom Ansible og Python (7)	7	Kombinere Python-script og Scapy	0	IØ	
Uke 17									
25.04.2022	7	Ansible	7	Python script(7)	7	Python-script	7		
26.04.2022	7	Ansible	3	Python script(3)	7	Python-script	7	skriv in lag-2-svitsj konsekvent, fikset bugg i bibliogafien, fiksa cref og byttet språk til norsk (3)	se på kommentarer fra EGG(4)
27.04.2022	7	Ansible	7	Python script(7)	7	Python-script	7		
28.04.2022	7	Ansible	3	Python script (2) Møte med veileder(1)	0	Borte på STi-møte	7		
29.04.2022	18	Fredag: Ansible, Lørdag:ansible Søndag Ansible	7	Ferdigstilte python script (/)	0	Borte på STi-møte	7		
Uke 18									
02.05.2022	7	Rapport	6	overleaf(6)	7	Rapportskriving	7		
03.05.2022	8	Ansible	3	overleaf(3)	8	Rapportskriving	8	jobbe med ansible	
04.05.2022	8	Ansible	7	Modifiserte Python script (4) Overleaf (3)	8		7	jobbe med ansible	
05.05.2022	8	Ansible	6	Planla struktur på innlevering(6)	8	Rapportskriving	9	jobbe med skript og ansible,	møte med veileder (0.5)
06.05.2022	10	Rapport	8	Hjalp fikse kode (4) Overleaf (4)	10	Rapportskriving	10	sett opp skript for enkel bruk av prototype, planla struktur på rapport	
Uke 19									
09.05.2022	10	Skrive rapport	7,5	Skrev i overleaf kapittel 4 (4) Oppdaterte python script (3,5)	10	Rapportskriving	8	skrive guide, struktur på overleaf	
10.05.2022	10	skriv rapport	5	Oppdaterte ptyhon script (5)	10	Rapportskriving	10	skrive om autoinstal i teknologier, skrive om autoinstall i prototype	

	Olav		Vilja		Petter		Edward		Felles kommentar
dato	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	Tid	Kommentar	
11.05.2022	13	Skriv rapport, oppdatere ansible playbooks	8	Oppdaterte ptyhon script og skrev manual(8)	10,5	Rapportskriving	10	rapport, og ansible	
12.05.2022	12	møte, skrive rapport og prototype	7,5	veildermøte og scrum(2) Skrev manual (2) Laget diagrammer og fiksa kode (3,5)	11	Scrum meeting og rapportskriving	11	scrum, rapport	
13.05.2022	8	skrive rapport og prototype	10,5	Overleaf(7), Leste gjennom felles (3.5)	10	Rapportskriving og gjennor	11	rapport	
14.05.2022	8	rapport	8,5	Overleaf(8,5)	11	Rapportskriving	11	rapport, ansible testing	møte med veileder
15.05.2022	8	rapport	5	Overleaf(5)	11	Rapportskriving	11	rapport	
Uke 20									
16.05.2022	14	rapport, testing	10,5	Rapportskriving(10,5)	13,5	Rettskriving av prosjektplan, manualen, rapportskriving etc.	13	rapportskriving	
17.05.2022	14	testing	12	Rapportskriving (12)	13	Rapportskriving	13	rapportskriving	
18.05.2022	12	Rapportskriving	10,5	Rapportskriving(10,5)	13	Rapportskriving	13	rapportskriving	
19.05.2022	13	Rapportskriving	13	Rapportskriving og levering(13)	13	Rapportskriving	13	rapportskriving	
20.05.2022	1	Rapportskriving	1	Dobbelsjekker at alt er asjur (1)	1	Innlevering	1	Innlevering	møte med veileder
Totalt	585		474,5		519,5		519,5		

Vedlegg H

Prosjektplan

NTNU
Kunnskap for en bedre verden

Institutt for informasjonssikkerhet og kommunikasjonsteknologi

DCSG2900 - Prosjektplan

Automatisering av konfigurasjon og testoppsett i nettverkslaben

Skrevet av:

Petter Jacob Brautaset

Vilja Lauritsen Langseth

Olav Andreas Strandjord

Edward Cornelius Haukø Svihus

Januar, 2022

Table of Contents

1 Mål og rammer

1.1 Bakgrunn

Ved Cisco-laboratoriet på NTNU i Gjøvik blir det undervist i nettverksadministrering i tre forskjellige fag. For øyeblikket brukes det unødvendig mye tid på å oppdatere nettverksutstyret i Cisco-laboratoriet. Samtidig bruker foreleser og elever mye tid på å manuelt sette opp testmiljøene for gjennomføring av hver laboratorieøvelse, noe som legger beslag på tid og ressurser. Det er derfor ønskelig å se hvordan disse arbeidsoppgavene kan effektiviseres ved hjelp av automatisering.

1.2 Mål

1.2.1 Resultatmål

- Å utforske mulige løsninger for hvordan vi kan effektivisere oppdateringer av svitsjene og ruterene i Cisco-laboratoriet. Denne løsningen skal også gi brukeren mulighet til å laste opp og endre konfigurasjonsfiler på maskinene for å redusere tiden det tar å forberede laboratoriet for undervisning og lignende.
- Å lage en prototype av den beste løsningen fra informasjons-innsamlingsfasen.
- Å lage en helhetlig brukermanual for prototypen.

1.2.2 Effektmål

- Fjerne avhengigheten av å måtte konfigurere hver svitsj og ruter manuelt.
- Få oversikt over status på nettverksutstyr
- Oppdage feilkonfigurert nettverksutstyr eller som ikke fungerer som det skal

1.3 Rammer

Som resultat av løsningen vår vil nettverksutstyret bli automatisk konfigurert og tilbakestilt for å utføre ønskede oppgaver. Dette betyr at vi ikke skal bruke lokale konfigurasjoner på nettverksutstyret.

Selve løsningen som blir utviklet skal fungere på versjon (15.9(3)M) av IOS.

For at oppdragsgiver lettere skal kunne oppdatere og videreutvikle løsningen vår må den være skrevet med et gjenkjennelig kodespråk. Java, C++ eller Python ble anbefalt av oppdragsgiver.

Løsningen skal fungere for alle stasjonene i Cisco-laben på NTNU i Gjøvik.

Guiden er rettet mot erfarne nettverksadministratorer. Derfor kan deler av ord og uttrykk være ukjent for enkelte, men vi vil utforme en ordbok for de uttrykkene som er brukt hyppigst. Da oppgaven er ment til å bli lest av erfarne nettverksadministratorer ser vi ikke på dette som et problem, men en nødvendighet for at oppgaven skal tilfredstille oppdragsgiverens ønske om kvalitet og innhold.

2 Omfang

2.1 Avgrensning

Det er kun nødvendig at løsningen fungerer i Cisco-laboratoriet på NTNU i Gjøvik. Det er derfor hensiktsmessig at vi utvikler en løsning som kompatibelt med hovedsakelig Cisco-nettverksutstyret som er i bruk i nettverkslaben.

2.2 Fagområde

Oppgaven omhandler forenkling av oppdatering og opplasting av filer for en Cisco-nettverkslaboratorie. Tematikk som nettverksadministrering, automatisering og programvareutvikling vil bli drøftet i oppgaven.

2.3 Oppgavebeskrivelse

Oppgavens beskrivelse kom med de følgene punker. Appendix B

- Se på hvilke løsninger som finnes for automatisering av oppsett/konfigurasjon av rutere, svitsjer etc. i et lab-miljø som Cisco-laben ved IIK, NTNU, Gjøvik.
- Vurdere fordeler og ulemper knyttet til identifiserte løsninger.
- Velge den løsninger som synes å være best egnet, og implementere en prototype.
- Lage en selvstendig, kortfattet guide som beskriver hvordan prototypen fungerer og hvordan den er tenkt brukt.

[1]

2.3.1 Prototype

En prototype av et program skal utvikles og skal bistå i å automatisere konfigurasjonsfiler på nettverksutstyr i Cisco-laboratoriet på NTNU. Prototypen skal bestå av et grensesnitt eller programfiler som skal utføre ønskede oppgaver avhengig av administratorens ønske for konfigurasjonen. Formålet er å effektivisere driften, og kunne sende ut konfigurasjon til samtlige rutere og svitsjer, uten å måtte manuelt koble seg til hver enhet med en dedikert konsollkabel. Konfigurasjonsfilene utdeles over det kablede nettverket, og vil kvittere en melding tilbake om konfigurasjonsendringen var vellykket.

Prototypen består av: En skisse over de funksjonene prototypen tilbyr En kort beskrivelse av de ulike konfigurasjonsalternativene En kombinert skissering av grensesnittet og konfigurasjonsinnstillingene.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

3.1.1 Petter Jacob Brautaset

Sektretær Petter skriver ned notater under alle møter, og er også ansvarlig for å holde alle filer organisert.

3.1.2 Vilja Lauritsen Langseth

Kommunikasjonsansvarlig Vilja er ansvarlig for kommunikasjon med veileder og oppdragsgiver, de er også ansvarlig for innleveringer.

3.1.3 Olav Andreas Strandfjord

Organisator Olav er ansvarlig for å reservere rom for gruppemøter og arbeidsøkter.

3.1.4 Edward Cornelius Haukø Svihus

Lagleder, Scrum Master

Ansvarlig for å holde prosjektet på skinnene og at det følger Gantt-skjemaet. Scrum master skal holde møtene sprint møtene. section 6.1

3.2 Rutiner og grupperegler

3.2.1 Regelmessige møter

- Møte hver torsdag med veileder (saker blir skrevet i møtereferatet i uken før møtet)
- Daglig scrum kl 10:15
- Scrum-møte som inkluderer sprint planning, sprint review og sprint retrospective hver mandag kl 09:15

3.2.2 Regelmessige backups

Teamlederen tar daglige backup av dokumenter på Google Docs og Overleaf.

3.2.3 Grupperegler

- Gjennomsnittlig 30 timers arbeidsuke
- Normale arbeidstimer 9:15-16:00
- Arbeid 45 min med 15 minutters pause
- Hvis det danner seg en konflikt vil saken bli diskutert frem til alle er enige, hvis en enighet ikke oppnås vil saken bli tatt opp med veileder

4 Planlegging, oppfølging og rapportering

4.1 Prosessrammeverk

Vi har valgt Scrum som agile tankegang i prosjektet. Rammeverket er basert på utvikling og leveranse i korte iterasjoner med fast lengde. Dette skal bistå oss i utviklingen av automasjonsopppsettet med tydelige krav og spesifikasjoner, og kontinuerlig tilbakemelding mellom hver iterasjon .

4.1.1 Hva er Scrum?

Scrum er er et rammeverk for å hjelpe med å a generere verdier gjennom adaptive løsninger. For å oppnå målene vil Scrum være benyttet for å kunne løse oppgavene iterativt i gruppen i form av kontinuerlig tilbakemelding. Filosofien bak Scrum er at prosjektleder deler et stort prosjekt opp i flere mindre deler kalt for sprints, som er mellom en og fire uker. Ved å sette en begrensing på arbeidsmengden til en måned eller mindre skaper man en oversiktlighet som vil hjelpe med å holde personer ansvarlige for deres arbeidsoppgaver slik at gruppedeltakerne alltid vet hva som skal gjøres og hvorfor. Ved benyttelse av Scrum vil teamet kunne jobbe på en slik måte at man kan redusere risikoen for å sette sprinten i fare. En backlog vil være lett tilgjengelig, som vil være behjelpelig for gruppedeltakernes arbeid. I tillegg vil en agile tankegang bli implementert for at gruppemedlemmene kan jobbe mer selvstendig. Denne formen vil gjøre det mulig å kunne jobbe sømløst i gruppen på en kontrollert måte, og etterhvert som målene oppnås vil en få oversikt over progresjonen. Teamet vil jobbe slik at de har en daglig scrum, slik at man alltid kunne vite status for prosjektet samt delegere arbeid mellom gruppemedlemmene. Møtet avholdes i 15 minutter. Kommunikasjon er det viktigste man kan ha i Scrum slik at men hva som må gjøres og hvorfor. Etter hver "sprint" vil det gjennomføres en statusrapport og evaluere arbeidet som er gjort. Da man har et møte med arbeidsgiver og bestemmer hva som må gjøres framover og sette en plan for arbeidet som skal gjøres videre.
[2]

4.1.2 Hvorfor Scum?

Scrum ble valgt siden å benytte seg av små iterative leveringer på den måten at gruppen alltid vil kunne arbeide videre med klare mål. Ved benyttelse av Scrum vil gruppen kunne arbeide mer effektivt ved at man alltid har arbeidsmålene fastsatt mellom hver iterasjon.

4.1.3 Gjennomføring

Gruppen har satt av en uke som optimalt sprintintervall, og ved benyttelse av mindre sprinter vil tidsbruken forbli akseptabel. Lengre sprinter på et prosjekt av denne lengden blir for langt, da det vil bli færre muligheter for produkteier å komme med innspill og styre prosjektteamet inn på ønsket kurs. Kortere sprinter ville ha ført til mye mer overhengende tidsbruk som går til sprint planning og reviews. De daglige sprintene vil kunne hjelpe gruppen å jobbe optimalt opp mot sine egne mål og kunne gi oppdragsgiver tilfredstillende resultater.

Før hver sprint vil det gjennomføres en sprint planning meeting der arbeidsoppgavene for kommende sprint blir identifisert og planlagt for den kommende sprinten. Etter hver sprint meeting vil det gjennomført et møte med arbeidsgiver der gruppen presenterer det som har blitt gjort siden sist, dersom dette er relevant for arbeidsgiver. Dette er en fin mulighet for arbeidsgiveren å gi tilbakemelding, og for å forsikre prosjektteamet at prosjektet er på rett vei. Det vil bli gjennomført daglige stand-up møter, daglig scrum, der involverte diskuterer arbeidsoppgavene og eventuelle problemer som må løses for å opprettholde progresjon i prosjektet.

4.2 Status-møter og beslutningspunkter

Prosjektet benytter seg av Scrum-metoden og et sprint møte blir arrangert mandag annenhver uke for å få en samlet oversikt over framgang og eventuelle forbedringspunkter i prosjektet.

Hvis informasjonssamlingsprosessen går forbi fristen kuttes deler av prototypen ut.

Beslutningsmøte for å beslutte om deler av prosjektet skal kuttes ut er den 20.04.2022

5 Organisering av kvalitetssikring

5.1 Dokumentasjon og standarder

Referansestandard for kilder: Vancouver

Dokumentasjon blir gjennomført og lagret i Google Docs.

Scrum-board blir utformet på nettsiden Trello.

Kommunikasjon innad i teamet blir gjennomført på kommunikasjonsplattformen Discord.

Rapportskriving og prosjektplan blir skrevet i tekstbehandlingsprogrammet Overleaf.

Kvalitetsikring opprettholdt ved at all dokumentasjon som skal leveres blir sjekket før innlevering i form av en iterasjon. Det skal bli gjennomgått en dobbeltsjekk av alle dokumenter, samtidig for å fastsette at korrekt grammatikk er benyttet.

For å sikre god oversikt over hva som blir sagt på møter vil sekretæren skrive møtereferater etter hvert møte, dette vil sikre god oversikt og gir samtidig god orientering om hva som bør være i fokus.

5.2 Plan for inspeksjoner og testing

Vi følger Scrum-metodens interasjoner. En gjennomgangsperiode vil bli utført etter hver Sprint (iterasjon) for å få en oversikt over progresjon. Prototypen skal bli testet etter hver iterasjon for å opprettholde en forståelse av dens funksjon og praksis. For hver iterasjon av gjennomførelse av rapporten vil gruppen ha som mål å få så god tilbakemelding som mulig for å ende opp med det beste sluttproduktet.

5.3 Risikoanalyse på prosjektnivå

Risikoanalysen skal redegjøre for potensielle risikoscenarier i relasjon til prosjektet samtidig som den skal vise hvilke tiltak som er i plass for å redusere konsekvensen eller sannsynligheten for en gitt hendelse.

5.3.1 Verdier

Verdiene er delt inn i fire nivåer; `lav`, `medium`, `høy` og `veldig høy`. `Lav` vil indikere at en eventuell tap av verdien ikke vil påvirke prosjektet i nevneverdig grad. Tap av en `veldig høy` verdi vil drastisk redusere forventet karakter for prosjektet eller føre til at prosjektet ikke blir godkjent. Se tabell 1 for verdier.

Table 1: Verdier

Navn på verdi	**Verdi**
Gruppemedlemmer	Veldig høy
Rapport i Overleaf	Veldig høy
Dokumenter i Google Drive	Høy
Laboratorieutstyr	Høy
Produkt	Høy
Personlige notater	Medium
Tid	Medium

5.3.2 Risikoscenarier

Et risikoscenarie er en potensiell hendelse som kan negativt påvirke sluttproduktet av prosjektet. Tabell 2 under har hvert risikoscenarie en verdi som beskriver sannsynligheten og konsekvensen til scenariet, det er inkludert hvilke verdier som er påvirket og hvilke tiltak som er satt i plass for dette scenariet. Sannsynligheten er delt opp i 4 kategorier, lav, moderat, høy og veldig høy.

Lav Forventet at det forekommer mindre en en gang i året

Moderat Forverntet at det forekommer mellom en og tolv ganger i året

Høy Forventet at det forekommer mellom 12 og 52 ganger i året

Veldig høy Forventet at det forekommer mer en 52 ganger i året

Konsekvensene er delt opp i lignende kategorier, der kategoriene er definert på hvordan det vil påvirke slutt resultatet

Lav Ingen/neglisjerbar påvirkning av sluttresultat

Moderat Noe påvirkning av sluttresultat

Høy Høy påvirkning av sluttresultat

Veldig høy Kan føre til ikke godkjent oppgave

Table 2: Risikoscenarier

Risiko	Sannsynlighet Etter Tiltak	Konsekvens Etter Tiltak	Tiltak	Påvirker verdi
Sykdom og fravær	Høy	Moderat	Koronatiltak, Fleksibilitet	Gruppemedlem, Tid
Gruppemedlem forlater gruppe	Lav	Høy	Gruppeavtale	Tid
Gruppemedlem bidrar ikke i gruppearbeidet	Lav	Høy	Gruppeavtale	Tid
Cisco-laboratoriet blir utilgjengelig i fremover i tid	Lav	Moderat	Jobb eksternt	Laboratorieutstyr
Tjenesteavbrudd på nødvendige applikasjoner	Lav	Moderat	Backup	Rapport i Overleaf, Dokumenter i Google Drive, Tid
Mister tilgang til filer på Google Drive eller Overleaf	Lav	Lav	Backup	Rapport i Overleaf, Dokumenter i Google Drive
Campus stenger ned pga. Korona	Moderat	Moderat	Hjemmekontor	Tid
Vi faller bak på framgangsplanen for prosjektet	Moderat	Høy	Forminske oppgaveomfang, Fleksibilitet	Rapport i Overleaf
Prototype blir ikke ferdig i tide	Lav	Veldig Høy	Forminsk oppgaveomfang	Produkt
Rapport blir ikke ferdig i tide	Lav	Høy	Forminsk oppgaveomfang, Iterasjoner	Rapport i Overleaf
Skade og tap av personlig utstyr	Lav	Lav	Backup, Låne utstyr fra skolen	Notater
Dårlig kommunikasjon med veileder	Lav	Moderat	Jobbe uavhengig	Produkt, Tid
Større endringer i kravspesifikasjon fra kunden underveis i prosjektperioden	Lav	Moderat	Diskuter omfang	Produkt, Tid
Oppdragsgiver skrinlegger prosjektet	Lav	Moderat	Jobbe uavhengig	Produkt
Veileder forlater prosjektet	Lav	Moderat	Jobbe uavhengig	Produkt

5.3.3 Beskrivelse av risiko

Sykdom og fravær
Sykdom og fravær vil føre til at gruppemedlemmene ikke får gjort nødvendig arbeid. Dette kan føre til at det vil være vanskeligere å nå tidsfristene som er satt underveis i prosjektet. Dette kan føre til skjev arbeidsfordeling i gruppen, og i værste fall store forsinkelser i arbeidet.

Gruppemedlem forlater gruppe
Dersom en gruppemedlem forlater gruppen vil arbeidsoppgavene bli fordelt på færre. Dette vil føre til større arbeidspress. Dette scenariet vil sannsynligvis oppstå uforutsett. Dette kan gjøre

det vanskelig for gruppen å nå målene som er satt i planen.

Gruppemedlem bidrar ikke i arbeidet
Dersom en gruppemedlem ikke bidrar i gruppearbeidet vil det føre til skjevfordeling av oppgaver og at andre gruppemedlemmer må gjøre arbeidet istedenfor.

Cisco-laboratoriet blir utilgjengelig
Cisco-laboratoriet kan bli utilgjengelig fremover i tid dersom en uforutsett skade har skjedd på utstyr i form av naturlige årsaker. Dette vil være vanskelig å forutse. I de fleste tilfeller vil laben være tilgjengelig ila. en arbeidsdag. Men store skader på teknisk utstyr og inventar kan føre til lang gjenopprettingstid.

Tjenesteavbrudd på nødvendige applikasjoner
Avbrudd hos leverandører som leverer programvareløsninger nødvendig for gjennomføring av prosjektet. Data nødvendig for gjennomføring vil være utilgjengelig som konsekvens.

Campus stenger ned pga. Corona
Dersom Campus stenger ned vil vi ikke lenger a mulighet til å sitte fysisk sammen og jobbe med prosjektet. Dette kan føre til mindre produktivitet og dårligere kommunikasjon.

Vi faller bak på fremgangsplanen for prosjektet
Dersom vi faller bak på fremgangsplanen vil vi ha vanskeligheter med å opprettholde tidsfristene satt, samt. at rapporten ikke blir ferdig til akseptabel tid.

Prototype blir ikke ferdig i tide
Dersom prototypen ikke er klar i tide vil vi ha vanskeligheter med å gjennomføre prototypen. Uten grunnlaget prototypen vil gi vil det være vanskelig å skrive en helhetlig rapport som skriver til oppdragsgiverens forventninger.

Skade og tap av personlig utstyr
Skade og tap av personlig utstyr kan føre til uopprettelig tap av data og maskinvare. Tilgang på data og maskinvare er nødvendig for å gjennomføre prosjektet.

5.3.4 Tiltak

Det er implementert flere tiltak for å redusere sannsynligheten eller konsekvensen for risikoscenariene.

Table 3: Tabell over tiltak

Tiltak	**Beskrivelse**
Koronatiltak	Følge anbefalte koronatiltak. Melde ifra om sykdom så langt det lar seg gjøre.
Kontakt veileder	Kontraktsbrudd. Ta kontakt med veileder.
Gruppeavtale	Minne på gruppemedlem om gruppeavtalen. Ettersom avtalen klargjør roller for hvert gruppemedlem gjør det det lettere å delegere ut ansvaret for medlemmets oppgaver til resterende medlemmer.
Jobb eksternt	Jobbe eksternt med dedikert ruter og svitsj fra Cisco-laboratoriet.
Backup	Ta sikkerhetskopi av data lagret i skyen hos de ulike leverandørene. Ha alternative verktøy klart som kan brukes istedenfor.
Hjemmekontor	Bruke Discord til å holde møter og arbeidsøkter
Forminsk oppgaveomfang	Definerer deler av prosjektet vi kutter ut for å spare tid.
Rapport	Sørg for at den fullførte rapporten har god dokumentasjon og forklaring på hvordan man kan bygge videre på prototypen. Dermed kan oppgavegiver bruke rapporten som utgangspunkt for å bygge produktet selv om de ønsker det.
Låne utstyr fra skolen	IT-tjenesten kan bistå med maskinvare som PC
Iterasjoner	Tre iterasjoner av rapporten for å forsikre at gruppen holder tritt med progresjonen.
Regelmessige møter	Ukentlig møte med veileder.
Jobbe uavhengig	Benytte seg av alternative ressurser og jobbe uavhengig hvis eventuelle veiledningsproblemer oppstår.
Fleksibilitet	Fleksibilitet i forhold til møtested og romvalg
Jobbe individuelt	Sette av tid selv slik at man kan hente inn eventuelle forsinkelser
Diskuter omfang	Diskuter omfang med oppdragsgiver.

6 Plan for gjennomføring

6.1 Gantt-skjema

Gantt-skjema blir utformet for å kunne sette realistiske tidsfrister på en slik måte at gruppen har oversikt over gjennomføringen og fremtidige tidsfrister. Gantt skjemaet til dette prosjektet er lagt inn som et vedlegg. Appendix A

6.2 Milepæler

6.2.1 Ferdig forprosjekt

For å jobbe mest mulig effektivt må vi bestemme oss for hvordan vi skal arbeide, bestemme og sette realistiske tidsfrister og tildele oppgaver. Dette blir bestemt i forprosjektsstadiet som vil stå som et grunnlag for hele prosjektet.

Dato: 31.01.2022

6.2.2 Iterasjon nr. 1 av prosjektrapporten

Dette skal være det første utkastet av prosjektrapporten, vi kommer ikke nødvendigvis til å være ferdig med selve prosjektet til da, men selve strukturen til rapporten skal være ferdig og det skal fylles ut så mye som er mulig. Hver iterasjon skal få tilbakemeldinger som blir brukt til å forbedre neste iterasjon.

Dato: 08.04.2022

6.2.3 Iterasjon nr. 2 av prosjektrapporten

Andre iterasjon av prosjektrapporten.

Dato: 02.05.2022

6.2.4 Siste og tredje iterasjon av prosjektrapporten

Dette er siste utkast av rapporten som blir levert inn.

Dato: 20.05.2022

6.2.5 Ferdig prototype

Som del av prosjektet skal det lages en prototype av et automatisert oppsett på labben

Dato: 02.05.2022

6.2.6 Presentasjon

For presentasjonen skal det lages en plakat og en presentasjon av bacheloroppgaven.

Dato: Datoen på presentasjonene var ikke satt når dette dokumentet var skrevet.

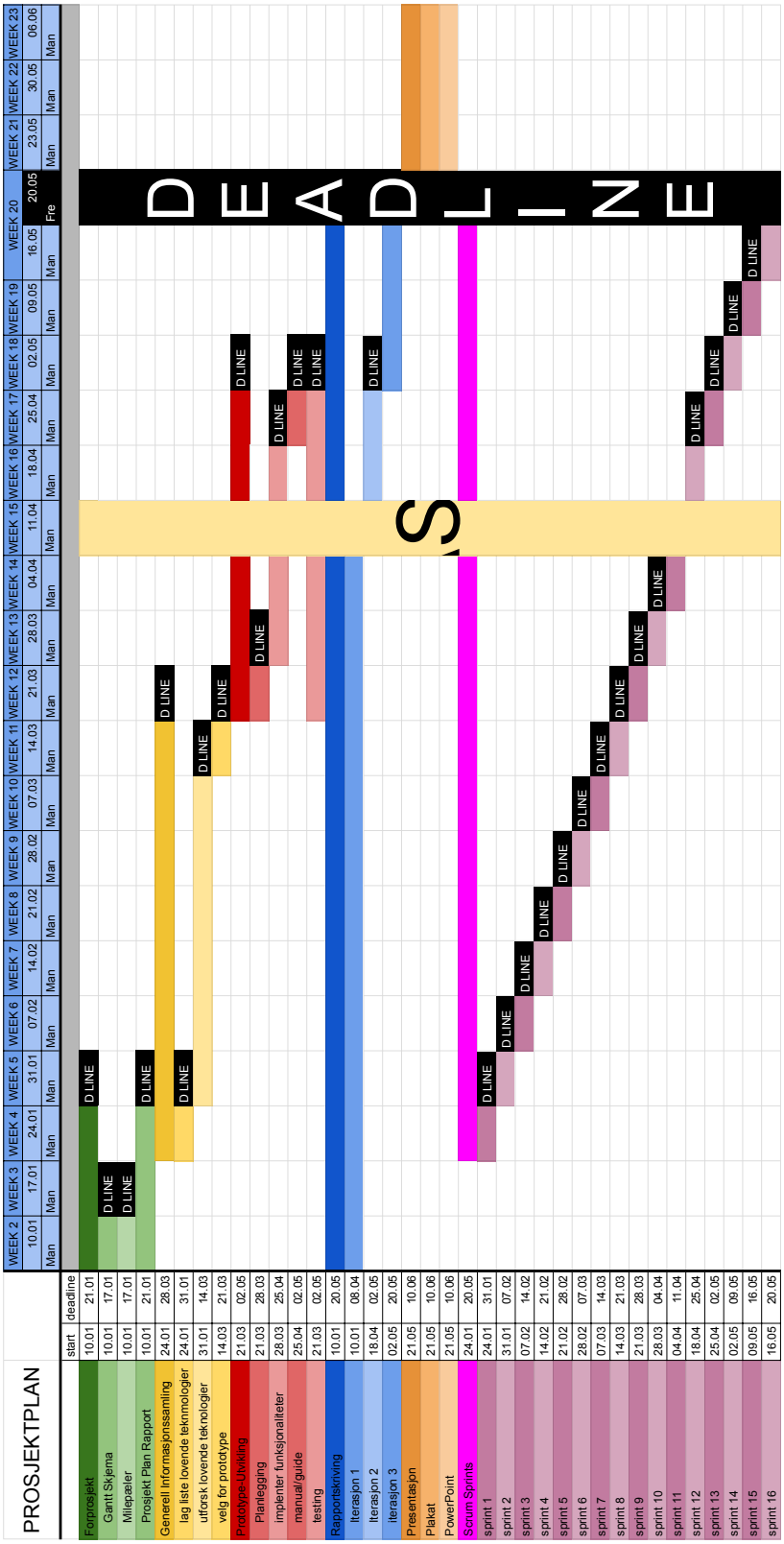
Bibliografi

[1] Eigil Obrestad and Ernst Gunnar Gran. *Automatisering av konfig og testoppsett i nettverkslaben.* 2022.

[2] Jeff Sutherland and Ken Schwaber. *The 2020 Scrum Guide.* URL: https://scrumguides.org/scrum-guide.html (visited on 17th Jan. 2022).

Appendix

A Gantt-skjema



B Oppgavebeskrivelse

Bacheloroppgave ved NTNU

Oppdragsgiver: IIK, NTNU - ved Ernst Gunnar Gran

Automatisering av konfig og testoppsett i nettverkslaben

I Cisco-laben ved NTNU i Gjøvik, har vi stadig behov for å kunne sette opp ulike testmiljøer, rulle ut forskjellige konfigurasjoner, oppdateringer m.m. – og slik legge til rette for forskjellige læringsaktiviteter, generell lab-gjennomføring og vurderings-/eksamensaktivitet. Per dags dato utføres mye av dette arbeidet (semi)manuelt, noe som legger beslag på unødvendig mye ressurser. Det er derfor ønskelig å se på hvordan oppsett av testmiljøer, konfigurasjon av nettverksutstyr m.m. kan effektiviseres gjennom automatisering i Cisco-laben.

Bachelorgruppen som velger å se nærmere på utfordringen skissert over vil ha anledning til å diskutere/presisere oppgaven nærmere i samråd med oppdragsgiver, men det vil i utgangspunktet være naturlig at oppgaven inkluderer følgende:

- Se på hvilke løsninger som finnes for automatisering av oppsett/konfigurasjon av rutere, svitsjer etc. i et lab-miljø som Cisco-laben ved IIK, NTNU, Gjøvik.
- Vurdere fordeler og ulemper knyttet til identifiserte løsninger
- Velge den løsningen som synes å være best egnet, og implementere en prototype
- Lage en selvstendig, kortfattet guide som beskriver hvordan prototypen fungerer og hvordan den er tenkt brukt

Vedlegg I

Oppgavetekst

Bacheloroppgave ved NTNU

Oppdragsgiver: IIK, NTNU - ved Ernst Gunnar Gran

Automatisering av konfig og testoppsett i nettverkslaben

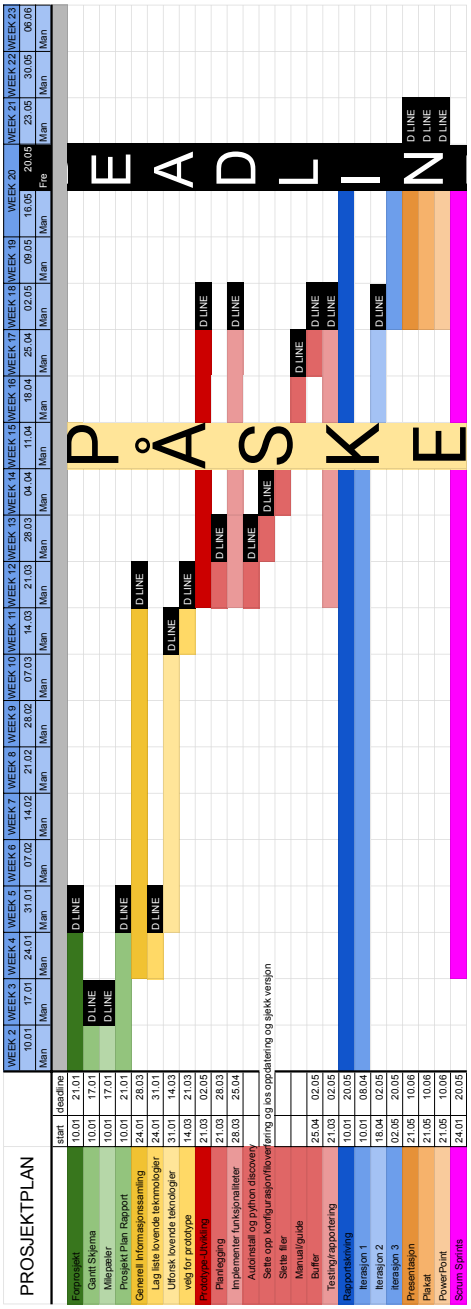
I Cisco-laben ved NTNU i Gjøvik, har vi stadig behov for å kunne sette opp ulike testmiljøer, rulle ut forskjellige konfigurasjoner, oppdateringer m.m. – og slik legge til rette for forskjellige læringsaktiviteter, generell lab-gjennomføring og vurderings-/eksamensaktivitet. Per dags dato utføres mye av dette arbeidet (semi)manuelt, noe som legger beslag på unødvendig mye ressurser. Det er derfor ønskelig å se på hvordan oppsett av testmiljøer, konfigurasjon av nettverksutstyr m.m. kan effektiviseres gjennom automatisering i Cisco-laben.

Bachelorgruppen som velger å se nærmere på utfordringen skissert over vil ha anledning til å diskutere/presisere oppgaven nærmere i samråd med oppdragsgiver, men det vil i utgangspunktet være naturlig at oppgaven inkluderer følgende:

- Se på hvilke løsninger som finnes for automatisering av oppsett/konfigurasjon av rutere, svitsjer etc. i et lab-miljø som Cisco-laben ved IIK, NTNU, Gjøvik.
- Vurdere fordeler og ulemper knyttet til identifiserte løsninger
- Velge den løsningen som synes å være best egnet, og implementere en prototype
- Lage en selvstendig, kortfattet guide som beskriver hvordan prototypen fungerer og hvordan den er tenkt brukt

Vedlegg J

GANTT-skjema



Vedlegg K

Møtereferat

K.1 Møtereferater med oppdragsgiver

Møte med oppdragsgiver 14.januar

Oppdragsgiver: Eigil Obrestad

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold

Tidspunkter:
Ernst vil gjerne ha ukentlige møter.
Eigil vil ha fortløpende møter.

Ernst: Torsdager eller fredager. Torsdag klokken 9.

- Viktig å huske på veilederrollen til Ernst. Vi kontakter han for møte.

- Sende kort e-post om hva vi har snakket om og blitt enig om på slutten av hvert møte med veileder.

Eigil og oppgaven
Håndtere filene som ligger på switcher og ruterne.
Usikkerhet i forhold til om det ligger konfigurasjon på switcher og rutere allerede.
Ønske: Sjekke om det ligger konfig på ruter og switch. Sjekke firmware på iOS. Slette konfig som ikke skulle vært der.
Tips: Komme i gang med å se på oppstartsverktøy. Finne ut av status uten å trykke på knapper på ruter og switch.
Distribuere og administrere filer på switchene med måltilpassede filer tilpasset arbeidsformålet f.eks Cisco-lab undervisningstime.
Formål: Slippe unødvendig mye tid brukt på å skrive manuelle kommandoer

Annen nyttig info:

- Vi kan koble ruter til internettet til NTNU via ethernet.
- Ferdig produkt er ønsket, hvis for omfattende vil en prototype være nok.

4) **Cisco-lab og tilgjengelighet: Rack først i Cisco-laben kan brukes eksternt og kan fjernstyres.**

Fra 10-14 er det satt opp labøvelse for studenter.

Hvilke verktøy anbefales å bruke for å automatisere oppsett?

- Ingen spesifikke. Må fungere med Cisco IOS.

Hvilke programmeringsspråk anbefales? Ansible, python etc.

- Må fungere med Cisco IOS. Noe Eigil kan fra før. Python, C, Java.

Bør det utvikles et grensesnitt for automatiseringene?

- Har ikke så mye å si om det er en knapp eller en fil som skal åpnes. Bør prioritere effektivitet. Hva skal man bruke grensesnittet til? Funksjon i forhold til vedlikehold?
- Skal det være brukerinteraksjon og skal godkjenning kreves av administrator?

Spør om hvilket språk vi skal bruke?

- Norsk eller Engelsk er fint. Velg det dere er komfortabel med. Ikke skriv vi eller jeg haha. Språkbruk er viktig.

Når starter bachelor presentasjonene?

- Starten av juni. 1 eller 2 uke av juni. Styres av administrasjonen.
- Problemavgrensing av prosjektet (skal vi automatisere en pod eller hele Cisco-laben)
- Hele Cisco-laben. Koble seg på nettet til Cisco-laben for å automatisere. Fjerne gamle filer. Hvilke rutere og switcher som ikke responderer. Få oversikt over alle rutere/switcher. Lage en konfigurasjonsdatabase over rutere og switcher dersom en skulle havarere og må byttes ut.

Møte med oppdragsgiver 02.februar

Oppdragsgiver: Eigil Obrestad

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold:

Er det mulig å bake inn konfigurasjon i IOS?

Event Manager er lagret i NVRAM? Studenter sletter NVRAM

- Ett eller annet som kan utnytte ruterne mens de er blanke. Må gjøre det mulig å lagre og slette.

Lete etter IOS-filer med en boot variabel med FTP? Hvordan gjennomfører man dette?

- Config-File setter opp SSH automatisk når ruter og switch blir bootet. Startup-config inneholder setup for SSH. Men det er ikke mulig da de peker for hverandre.
- Sette opp et script som automatisk sletter SSH-konfig etter konfigurasjon.

Bruke Telnet?

- Når det kommer til sikkerhet. FTP er en usikker protokoll. Det er ikke superviktig i dette oppdraget da oppdatering og konfigurering skjer internt.

Bruke reverse Telnet

Bruke en tankeløsning med USB-pinne. Er en mulighet. Kan være en vel så bra løsning dersom en løsning krever å sette inn 50 nettverkskabler hver gang sammenlignet med å sette inn 50 USB-pinner.

Rubber Ducky
Kontrollere USB-konfig. Må vente en udefinert tidsperiode.

Ansible
God automatisering, og mulighet for output og tilbakemeldinger. Kan bruke YAML, enkelt å konfigurere. Problemet er at du trenger å konfigurere en SSH-forbindelse fra ruteren allerede.

Puppet
Fungerer på en litt annerledes måte. Krever at et puppet-agent allerede installert.

Eigil mener: Puppet og Ansible er godt kjent i forhold til nettverkskonfigurasjon.

Event Manager er en tilleggsfunksjon til en løsning.

Python

En annen mulighet. Finnes allerede python-scripts som kan brukes til å automatisere rutere.

Eigil mener:
IP-adresser kan man ordne med DHCP. SSH-nøkler kreves for å konfigurere, bootstrap for å sette opp konfigurasjonsfiler. Det kan være en flere-stegs-løsning.

Er en mulighet å slette IOS-imager med et uhell. Løsning: Ha en USB-pinne klar for å laste inn IOS-iamge på nytt igjen.

Generelt: Skriver bacheloroppgave, gjøre ting en skjønner. Veldig viktig å forstå hva man gjør. Så lenge du kan kommandoen, vet du hva du gjør.

Møteref 24.02.22

Oppdragsgiver: Eigil Obrestad

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold:

- Edward har jobbet med TFTP-server
    - Hver ruter og switch benytter seg av IP-adresser og bundet til en konfigurasjonsfil.
- Vilja har jobbet med pnp som ikke fungerte på switchene siden de var for gamle.
- Olav har jobbet med Ansible.
- Petter har jobbet med konfigurasjon med Python.
- Daglige Scrum-møter som gjør at vi samarbeider
- To uker til med informasjonshenting
    - Så en uke med valg av prototype og opplæring
    - Rapporten: kort og konsis, ikke vær bekymret for side nr.
    - Husk hva som er selvfølgelig og det som ikke er det.
- Legg til Secret i glossary.
    - Opp til oss hvordan vi bruker terminologien.
    - Definere terminologi passet til oppgaven. OBS! husk å definere det best mulig
    - Bruk norske og engelske ord slik vi mener er naturlig.

- Hvem styrer DHCP server i laben?
- Flash? Ikke lurt siden det kan bli fjernet.

Møte med oppdragsgiver

Dato: 19.05.22
Oppdragsgiver: Eigil Obrestad

Deltakere:
Petter Brautaset
Vilja Langseth
Olav Strandjord
Edward Svihus

Møteinnhold:

- Informerte oppdragsgiver om detaljer knyttet til prototypen.

Erfaring underveis:

- Pod 8 har ikke nettverkstilgang
- Enkelte kommandoer fungerte ikke

Kan oppsummere følgende:

- Prototype fungerer som tiltenkt

Oppsummering av metode:

Før implementasjon: Manuelt, masse dødtid. Brukt notepad til å føre utstyr som har blitt oppdatert.
Etter implementasjon: Tar maksimum 58 minutter å sende konfigurasjonsfiler til nettverksutstyret. Tar 1 time og 18 minutter for hele prosessen.

K.2 Møtereferater med veileder

Veiledningsmøte 12.januar

Veileder: Ernst Gunnar Gran

Gruppedeltakere:
Petter Brautaset
Vilja Langseth
Olav Strandjord
Edward Svihus

Møtets innhold:

- Sett på bacheloroppgavene Sikkerhet i VLAN og Viten i Senter.
- Fått et overblikk over Agile tankegang og Scrum.
- Utformet en timeplan.
- Fikset innholdsliste til forprosjekt i Overleaf.

Veiledningsmøte 20.januar

Deltakere: Petter Brautaset, Olav Strandjord, Vilja Langseth, Edward Svihus

Møtets innhold:

Spørsmål til veileder

Bør vi kort forklare hva en ruter og svitsj *er* i bacheloroppgaven?

- Skrive en halv side, skrive det som er relevant for det vi gjør i oppgaven. Eksempelvis: Forklare pakkelag osv. er uinteressant for oppgaven.

Er avsnittet innledningsvis i bakgrunn plagiat.

- Ja, bør gjøre om setningene til egne ord

Hvor lang tid bruker dere på å konfigurere en pod og hvor lang tid vil dere redusere det til?

- Må spørre oppdragsgiver Eigil. Han sitter muligens på et konkret anslag.

Siterings-stil Vancouver eller Harvard?

- Vancouver er foretrukket stil

Planen fremover:

- Sette effektmål
- Fjerne avhengighetsmål

Møte med veileder 27.januar

Veileder: Ernst Gunnar Gran

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold:

- Korrekturlesing, gjerne noen andre enn den som har skrevet gjeldende avsnitt.
- Sjekke om valgte teknologier samsvarer med det Eigil ønsker at vi bruker.
- Switcher og rutere blir nullstilt hele tiden. Må studere hva som skjer når rutere og switcher skrur seg på. Undersøke prosedyren nærmere.
- VELDIG VIKTIG å snevre inn oppgaven og BESKREVET oppgaven nøye ift. Teknologivalg og hvorfor vi har valgt den løsningen vi har valgt, og hvorfor andre løsninger ikke passer.
- Forklare utfordringer med løsningene valgt.

Møtereferat 3.februar

Veileder: Ernst Gunnar Gran

Gruppedeltakere:
Petter Brautaset
Vilja Langseth
Olav Strandjord
Edward Svihus

Møtets innhold:

Innhold:
Gått kort gjennom arbeidsoppgavene til gruppedeltakerne

Vilja: Laste inn konfigurasjon fra CONFIG-FILER
Edward: Test av TFTP-konfig i nettverkslaben
Olav: Konfigurering av nettverksutstyr med Ansiblr
Petter: Simulering av nettverksoppsett i Packet Tracer

Snakket om retting av forprosjekt.

Møte med veileder 10.februar

Veileder: Ernst Gunnar Gran

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold: Spørsmål og svar

Reverse Telnet med en komponent

- Kjøpe en komponent til 800kr for å kunne kjøre reverse telnet

Formatere hele disken

- Om det finnes en kommando som kan gjøre dette. Register som kan bli nullstilt med en factory reset.

Non-recurring vs recurring

- Environment variabel blir resatt, men den andre vil gjøre det hver gang den booter.

Hvordan forhindrer man overkjøring?

- Filen ligger ikke lokalt, men hentes over TFTP-serveren. Alltid skal ruterne lese “Bob” TFTP-server. VIKTIG! Konfigurere Read-only på TFTP-serveren.

Uninett-forespørsler til ruterne.

- DHCP-server som prøver å få kontakt med ruterne. Handler om autentisering av rutere og svitsjer på universitetsnettet.

Filer som ligger på ruter i forbindelse med faget “Campus nettverk”

- Kan slettes.

Sende kontrollsekvenser i menyen i Putty.

- Sende break. Mulig.

Forkortelser

- Husk å skrive akronymer helt ut, med forkortelser i parentes. Dersom forkortelsen er mest kjent er det naturlig å bruke forkortelsen.

- Dersom forkortelsen til akronymet skrives i en parentes og har ett sammenhengende ord etterpå bør setningen reformuleres.

Snakk om mulig konfigurasjon:

- Bruke bootvariabelen til å først boote ruteren, og så bruke Ansible og Puppet til å konfigurere.

Veiledningsmøte 3.mars

Veileder: Ernst Gunnar Gran

Gruppedeltakere:
Petter Brautaset
Vilja Langseth
Olav Strandjord
Edward Svihus

Spørsmål

- Opphavsrett til Cisco-grafikk som inneholder Cisco-logoen.
    - Bilde var lisensiert.

Konklusjon: Veileder anbefaler å lage nye figurer med vektorgrafikk da dette ser mest profesjonelt ut.

Veiledningsmøte 10.mars

- Skal vi bruke DHCP-pool eller RGP til å finne IP-adresser og knytte de til MAC-adressene?

- Hva hindrer oss fra å legge en konfigurasjonsfil lokalt og laste den over til ruter/switch?
    - Ansible gir oss bekreftelsesmeldinger på at ting er gjort.

- Hvorfor ikke benytte oss av en TFTP-server for å overføre konfigurasjonen til ruter/switch?
    - Viktig å skrive om
    - Sende en tekstfil med status på ruter/switch fra TFTP-server? Hvorfor ikke gå for denne løsningen? Utdyp i oppgave.

- CDP-meldinger sendes ut fra switch og ruter. Kan vi benytte oss av dette?

Tips fra Ernst
Ta for seg tre teknologier som er mest lovende, og skrive litt mer utdypende i forhold til dette.

Møte med veileder 24.mars

Veileder: Ernst Gunnar Gran

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold:

Kort forklaring av ønsket prototype:

1. Benytte oss av en Ubuntu-VM på SkyHiGH og en GRE-tunnell inn til nettverkslaben.
2. DHCP-server gir ut IP til ruter, svitsjer og multilagssvitsjer.
3. TFTP-server startes for å laste over konfigurasjon og sette opp SSH.
4. Skanne nettverket. Bruker et Python-script til å hente MAC-adresser og IP-adresser. Vi henter en predefinert liste med tildelte MAC-adresser til enhetene for å identifisere enheten, og om enheten er en ruter, multilagssvitsj eller en svitsj.
5. Koble til SSH med Ansible.
6. Laste over konfigurasjon til ønskede enheter med Ansible..
7. Slette grunnoppsett.

Kort diskusjon om ønsket prototype er realiserbar:

Konkludert med at ønsket prototype vil fungere som tiltenkt. Tidligere vært litt i tvil om Ansible vil kunne fungere 100% som tiltenkt. Prototypen vil bestå av en konfigurasjon av både Python og Ansible. Vi legger til grunn at Ansible vil fungere som tiltenkt, og lager en prototype basert på begge teknologiene.

Veiledningsmøte 31.mars

Veileder: Ernst Gunnar Gran

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Møtets innhold:

- Problemer med at svitsj i lab gir ut feil tegn ved bruk av - show run kommandoen.
- Problemer med konfigurasjon med crypto-key i oppstartsprosessen. Edward har funnet en løsning med en cron-jobb som startes ved siden av prosessen for å lage en key ved siden av.
- Leveringsklart til fredagen.

Møtereferat fra veiledningsmøte 19.05.22

Deltakere:
Petter Brautaset
Olav Strandjord
Vilja Langseth
Edward Svihus

Referat:

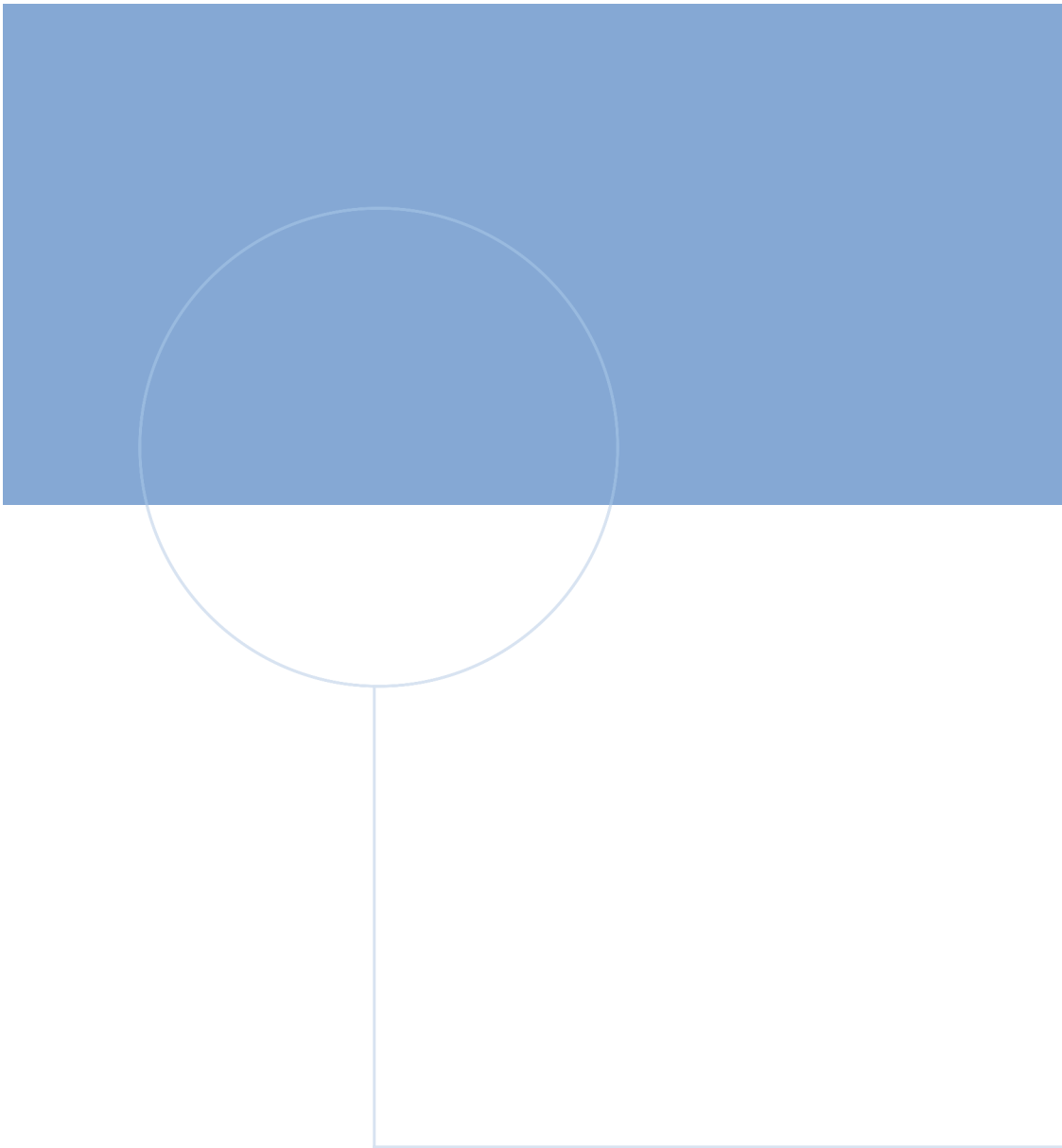
- Ved S2 legg til ordet figur først
- Lag testing som et eget kapittel
    - Noter dette i 1.10
- Vedlegg - sjekk litt om vedlegg om Ansible
    - Legg til timeliste

Refleksjon:

- Det som kan gjøres annerledes i gruppen
    - Noter at alle diagram ble laget med diagrams.net
    - Flytt faste figurer til å være midt i forklaringslinjen.

Resultat:

- Flytt optimalisering av config-fil fra testing til prototype.
- Rett på noe i resultater til riktig testnummer.



NTNU

Kunnskap for ei betre verd