Jaakko Lehtinen

OpenStack Swift -objektitallennus etäresurssi-palveluna

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

28.3.2014

Metropolia

Tekijä Otsikko Sivumäärä Aika	Jaakko Lehtinen OpenStack Swift -objektitallennus etäresurssipalveluna 84 sivua + 9 liitettä 28.3.2014
Tutkinto	insinööri (AMK)
Koulutusohjelma	tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	tietotekniikka
Ohjaaja	yliopettaja Harri Ahola

Insinöörityön tavoitteena oli luoda tiivis katsaus tiedon tallennukseen liittyviin asioihin ja siihen, miten tiedon tallennus voidaan toteuttaa nykyaikaisesti etäresurssipalveluna. Työssä käsitellään yleisimmät tiedon tallennukseen liittyvät komponentit, protokollat sekä niihin liittyvä teoria. Työssä esitellään tarkemmin objektipohjaiseen tiedon tallennukseen liittyvä teoria ja avoimen lähdekoodin ratkaisuna objektitallennuksen toteutus.

Työn tuloksena saatiin raportti, jonka avulla tallennusratkaisuja ennalta tuntemattoman henkilön on helppo saada tarvittava perustietous aiheesta. Lisäksi käytännön osuudessa esitellään yksi vaihtoehto etäresurssipalvelun tallennusratkaisuksi ja se, miten se liittyy esiteltyyn teoriaan. Käytännön osuuden rakentaminen osoitti, että avoimen lähdekoodin ohjelmistolla voidaan helposti toteuttaa teknisesti toimiva tallennusratkaisu. Käytännön osuudessa huomattiin myös se, että avoimen lähdekoodin toteutus yritysympäristöön ei ole avaimet käteen ratkaisu, vaan sen toteuttaminen käytännöllisesti ja tietoturvallisesti vaatii organisaatiolta OpenStack-pinon osaamista ja ymmärrystä, koska laadukasta tukea ei välttämättä ole helposti saatavilla.

Toimivan avoimen lähdekoodin etäresurssitallennuspalvelun suunnittelu ja toteutus osoittautui helpoksi tehtäväksi. Työn aikana kuitenkin havaittiin, että ratkaisussa on suuria puutteita esimerkiksi salasanojen käsittelyssä ja järjestelmän sisäisessä tietoliikenteessä. Kokonaisuutena yrityskäyttöön tarkoitetun toimivan avoimen lähdekoodin tallennusratkaisun ymmärtäminen ja toteuttaminen vaativat paljon työtä sekä syvällistä asiaan perehtyneisyyttä.

Avainsanat	etäresurssipalvelu, pilvipalvelu, tallennus, objektitallennus, swift, OpenStack, tallennusratkaisut

Author Title Number of Pages Date	Jaakko Lehtinen Information storage on object storage cloud 84 pages + 9 appendices 28 March 2014
Degree	Bachelor of Engineering
Degree Programme	Information Technology
Specialisation option	Information Technology
Instructor	Harri Ahola, Principal Lecturer

The purpose of this bachelor's thesis is to deliver a brief but comprehensive overview of data storage technologies, storage components, protocols and theory behind those. The thesis also discusses how data storage can be implemented as an open source based object storage solution.

The thesis demonstrates what is needed when implementing an open source code solution of object storage. The thesis shows that open-source software can be used to implement a technically sound storage solution. However it also demonstrates that the selected open-source implementation of object storage is not a turnkey solution for enterprise environment, as it requires the organization has a lot of OpenStack expertise available.

The implementation of object storage turned out to be easy. However, it was found out that the solution has major shortcomings, such as how it stores passwords and how the internal communication in the system is implemented. On the whole, understanding and implementing a functioning open source object storage solution to business environment requires a lot of work and in-depth understanding on the subject.

Keywords	cloud, storage, object storage, swift, OpenStack

Sisällys

Lyhenteet

1 Johdanto 1

2 Etäresurssipalvelun teoria ja käsitteet 3

2.1 Etäresurssipalvelun määritelmä 3
2.2 Etäresurssipalveluluiden palvelumallit 5
2.3 Etäresurssipalvelun sijoitusmallit 8
2.4 Etäresurssipalvelun tietotekninen infrastruktuuri 12

3 Tallennusjärjestelmien keskeisimmät komponentit 14

3.1 Massamuistilaitteet (mass storage device) 15
3.1.1 Mekaaninen kiintolevyasema (HDD) 15
3.1.2 Puolijohdelevyasema (EFD, Enterprise Flash Drive) 17
3.1.3 Magneettinauha-asemat 18
3.2 RAID-teknologia 19
3.3 Välimuisti (Cache) 23
3.4 Yhteysprotokollat 24

4 Tiedostojärjestelmät ja tiedonsiirto tallennusverkoissa 25

4.1 Lohkopohjainen tallennus 26
4.1.1 Suorakytkentäinen tallennusverkkoarkkitehtuuri (DAS) 27
4.1.2 Kuituväylätallennusverkko (FC SAN) 28
4.1.3 IP-verkkotallennus 32
4.1.4 iSCSI 32
4.1.5 IP-välitetty kuituväylä (FCIP) 35
4.2 Tiedostopohjainen tallennus 36
4.2.1 Verkkoliitännäinen tallennusjärjestelmä (NAS) 36
4.2.2 NFS 37
4.2.3 CIFS 38
4.3 Objektipohjainen tallennus (ObS) 39
4.4 Ohjelmointirajapinta (API) 42
4.5 Esittävä tilallinen (ReST) 42
4.6 Yksinkertainen objektien käsittely protokolla (SOAP) 43

5 OpenStack Swift-objektitallennuksen käsitteistö ja teoreettinen malli 44

5.1 Välityssolmut 45
5.2 Rengas 46
5.3 Alueet ja vyöhykkeet 46
5.4 Objektien kopiointi 47
5.5 Objektien auditointi 48
5.6 Tilit ja säiliöt 48
5.7 Tallennussolmut 48
5.8 Tunnistuspalvelu 50

6 OpenStack Swift -objektitallennuksen toteutus 52

6.1 Toteutuksen lähtökohdat 52
6.2 Infrastruktuuri 53
6.3 OpenFiler-näennäislevyjärjestelmä 56
6.4 Keystone-todennuspalvelimen asennus 57
6.4.1 Keystone-tietokannan asennus 58
6.4.2 Keystone-sovelluksen konfigurointi 59
6.4.3 Keystone-käyttäjä- ja palvelutunnusten määrittelyt 61
6.5 Swift-välitys- ja tallennussolmupalvelimien asennus 64
6.5.1 Palvelimien perusasetukset 64
6.5.2 Tallennussolmujen konfiguraatio 66
6.5.3 Välityssolmun konfiguraatio 67
6.6 Ympäristön testaus 72

7 Päätelmät 78

Lähteet 82

Liitteet

Liite 1. Keystone tenant- ja käyttäjäluonti
Liite 2. Keystone- ja Swift-palvelujen rekisteröinti
Liite 3. account-server.conf
Liite 4. container-server.conf
Liite 5. object-server.conf
Liite 6. proxy-server.conf
Liite 7. Renkaiden luonti
Liite 8. swift-openrc.sh
Liite 9. Testau-kuvakaappaukset

Lyhenteet

ANSI *American National Standards Institute*, yhdysvaltalainen organisaatio, joka vahvistaa eri organisaatioiden, yritysten ja muiden tahojen kehittämiä standardeja.

API *Application Programming Interface*, ohjelmointirajapinta, jonka avulla eri ohjelmat voivat tehdä pyyntöjä ja vaihtaa tietoja.

ATM *Asynchronous Transfer Mode*, yhteydellinen asynkroniseen tiedosiirtoon tarkoitettu protokolla.

BIOS *Basic Input/Output System*, ohjelma, joka vastaa käyttöjärjestelmän latauksesta sekä vastaa matalan tason kommunikaatiosta laitteiston kanssa.

CaaS *Communication as a Service*, viestintäratkaisu etäresurssipalveluna.

CHS *Cylinder, Head, Sector* kiintolevyllä sijaitsevan tietolohkon osoitukseen käytetty metodi.

CMU *Carnegie Mellon University*

DAS *Direct Attached Storage*, suoraliitännäinen tallennusarkkitehtuuri. Käytetään yleensä, tallennusarkkitehtuurista, joissa tallennusratkaisu on liitetty suorakaapelilla isäntäkoneeseen.

Etäresurssipalvelu
Sanastokeskuksen suosittama termi puhekielessä yleisemmin käytetylle termille pilvipalvelu.

EUI *Extended Unique Identifier*, määrittää ainutkertaisen nimen iSCSI kohteille.

FC *Fibre Channel*, verkkoteknologia, joka perustuu valokuitujen hyödyntämiseen kuljetusmediana.

FCIP *Fibre Channel over IP*, tunnelointiprotokolla, jolla FC-kehyksiä voidaan välittää IP-verkoissa.

FCP *Fibre Channel Protocol*, FC-verkoissa käytettävä kuljetusprotokolla.

HBA *Host Bus Adapter*, verkkokortti, jolla isäntäkone liitetään tietoverkkoon. Termiä käytetään yleisimmin kuituverkkojen verkkokorteista.

HIPPI *High Performance Parallel Interface*, nopeaan lyhyillä etäisyyksillä tapahtuvaan tiedonsiirtoon tarkoitettu protokolla.

HTML *Hypertext Markup Language*, avoimen standardin kuvauskieli, jolla kuvataan hypertekstiä.

HTTP *Hypertext Transfer Protolcol*, protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon.

I/O *Input/Output,* tietoteknisten resurssien tai komponenttien välinen kommunikaatio.

IaaS *Infrastructure as a Service (IaaS)*, etäresurssipalvelu joka tarkoittaa palvelimien ja palvelinsalien tarjoamista. Kokonaisuuteen sisältyy yleensä verkkoyhteydet, tallennustila, palvelimet ja niiden ylläpito.

IDE *Integrated Drive Electronics*, rinnakkaisliikenteinen liitäntäväylä massamuistin ja emolevyn tai ohjainkortin välillä.

IEEE *Institute of Electrical and Electronics Engineers*, kansainvälinen tekniikan alan järjestö.

INCITS *InterNational Committee for Information Technology Standards*, ANSI ja ISO/IEC komiteoiden yhteyselin.

IP *Internet Protocol*, protokolla, jota käytetään tietoliikennepakettien toimittamisessa perille pakettikytkentäisessä tietoliikenneverkossa.

IQN *iSCSI Qualified Name*, iSCSI nimeämistapa, jota voivat käyttää kaikki organisaatiot jolla rekisteröity verkkoalue nimi.

iSCSI	*internet Small Computer System Interface*, IP-verkoissa käytettävä tallennusverkko standardi, jolla voidaan liittää tallennuslaitteita.
iSNS	*Internet Storage Name Service*, protokolla mahdollistaa FC- tai iSCSI-laitteiden automaattisen haun, konfiguraation ja hallinnan TCP/IP-verkoissa.
ITC	*Information and Communications Technology*, tieto- ja viestintäteknologia.
ITU	*International Telecommunication Union*, YK:n alainen organisaatio, joka hallinnoi radiotaajuuksia sekä kehittää ja standardoi verkkoteknologioita.
LAN	*Local Area Network*, lähiverkko, eli rajoitetulla maantieteellisellä alueella toimiva tietoliikenneverkko.
LBA	*Logical Block Address,* kiintolevyllä sijaitsevan tietolohkon osoitukseen käytetty metodi.
LDAP	*Lightweight Directory Access Protocol*, hakemistopalvelujen käyttöön tarkoitettu verkkoprotokolla.
LUN	*Logical Unit Number*, tunnistenumero, jolla identifioidaan looginen yksikkö, jota voidaan osoittaa SCSI-protokollalla tai kuljetusprotokollilla, jotka kuljettavat SCSI-protokollaa sisällään.
MAC	*Media Access Control*, verkkosovittimen ethernet-verkoissa yksilöivä tunnusnumero.
MIT	*Massachusetts Institute of Technology.*
NaaS	*Networking as a Service*, tietoverkon toteutus etäresurssipalveluna.
NAND	*Negated AND*, looginen ei-ja operaatio.
NAS	*Network Attached Storage*, yleensä IP verkkoliitännäinen tallennusjärjestelmä.

NFS	*Network File System*, Sun Microsystems kehittämä hajautettu tiedostojärjestelmä.
NIST	*National Institute of Standards and Technology*, Yhdysvaltain kauppaministeriön alainen virasto joka edistää mittaustekniikoita, standardeja ja tekniikkaa.
NSA	*National Security Agency*, Yhdysvaltain kansallinen turvallisuusvirasto.
ObS	*Object based Storage*, objektitallennus.
OSD	*Object storage Device*, objektipohjaiseen tallennukseen perustuva tallennuslaite.
OSI	*Open Systems Interconnection Reference Model*, seitsemänkerroksinen tiedonsiirtoprotokollien yhteistoimintaa kuvaava teoreettinen malli.
OVF	*Open Virtualization Format*, avoimen standardin tapa pakata ja jaella näennäistysalustoilla ajettavia sovelluksia.
PaaS	*Platform as a Service*, etäresurssipalvelu jossa tarjotaan tilaajalle kokonaisuus, joka sisältää palvelinalustan lisäksi ohjelmiston ajamisen tai kehityksen tarvitsemat tietokanat, sovellukset, palvelut, kehitysympäristöt, tapahtumanvälityspalvelut.
PATA	*Parallel Advanced Technology Attachment*, rinnakkaisliikenteinen liitäntäväylä massamuistin ja emolevyn tai ohjainkortin välillä.
PDU	*Protocol Data Unit*, iSCSI yhteyksissä käynnistäjän ja kohteen välillä siirrettä informaatiopaketti.
PKI	*Public Key Infrastructure*, julkisen avaimen hallintajärjestelmä.
PRISM	NSA, vakoiluun suunniteltu tietokoneohjelma, jolla on pääsy Yhdysvalloissa toimivien etäresurssipalveluntarjoajien ympäristöihin.

RAID *Redundant Array of Independent Disks*, tietokoneiden kiintolevyjen vikasietoisuutta ja/tai nopeutta kasvattava tekniikka.

ReST *Representional State Transfer*, http-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseksi.

RPC *Remote Procedure Call*, prosessi, jolla ohjelmisto voi suorittaan rutiineja tai proseduureja ulkopuolisessa osoiteavaruudessa, yleensä toisessa tietokoneessa, johon on verkkoyhteys.

SaaS *Software as a Service*, ohjelmisto tarjotaan etäresurssipalveluna perinteisen lisenssipohjaisen tavan sijasta.

SAN *Storage Area Network*, tallennusverkko. Yleisesti nimitystä käytetään lohkopohjaiseen tiedonsiirtoon perustuvista verkoista.

SAS *Serial Attached SCSI*, sarjaliikenteinen SCSI väylä tiedon välittämiseksi tietokoneen tai oheislaitteen välillä.

SCSI *Small Computer System Interface*, rinnakkaisliikenne väylä tiedon välittämiseksi tietokoneen tai oheislaitteen välillä. SCSI on myös nimi kokoelmalle tietokoneen tai oheislaitteen välisen tietoliikenteen standardeja.

SCSITA *Small Computer System Interface Trade Association*, SCSI standardoinnista vastaava organisaatio.

SLED *Single Large Expensive Disk*, aikaisemmin tietotekniikassa keskus- ja minitietokoneiden kiintolevyistä käytetty käsite.

SMB *Server Message Block*, verkkoprotokolla, jota tyypillisesti käytetään tiedostojen jakamiseen Microsoft Windows käyttöjärjestelmissä.

SNIA *Storage Networking Industry Association*, yhdistys, joka toimii tallennusverkkoratkaisujen valmistajien ja käyttäjien yhteistyöelimenä.

TCP/IP *Transmission Control Protocol / Internet Protocol*, IP- ja TCP-protokollien yhdistelmä, jota käytetään verkkoliikenteen toteuttamiseen.

TOE	*TCP Offload Engine*, teknologia, jossa verkkokortti suorittaa TCP/IP pinon prosessoinnin keskusyksikön puolesta.
TPI	*Tracks per Inch,* kiintolevyn uratiheyden ilmaiseva luku.
UDP	*User Datagram Protolcol*, yhteydetön protokolla, jota käytetään tietoliikennepakettien toimittamisessa perille pakettikytkentäisessä tietoliikenneverkossa.
URI	*Uniform Resource Identifiers*, tiedon paikan tai yksikäsitteisen nimen kertova merkkijono.
URL	*Uniform Resource Locator*, tietyn tiedon paikan osoittava merkkijono. Käytetään mm. http-protokollassa määrittämän verkkopalvelimen osoite.
WAN	*Wide Area Network*, laajaverkko, eli tietoliikenneverkko joka peittää laajoja maantieteellisiä alueita.
WWN	*World Wide Name*, laitteen FC-, ATA- ja SAS-teknologioissa laitteen yksilöivä tunnusnumero.
WWNN	*World Wide Node Name*, kuituväyläverkoissa päätelaitteelle annettava ainutkertainen tunnistenumero.
WWPN	*World Wide Port Name*, kuituväyläverkoissa portille annettava ainutkertainen tunnistenumero.
XATTRS	*Extended Attribute Set*, tiedostojärjestelmän ominaisuus, jolla käyttäjä voi liittää tietoon metatietoa, jota tiedostojärjestelmä ei pysty tulkitsemaan.
XML	*Extensible Markup Language*, standardi merkintäkieli, jolla tietoon sisällytetään sen merkitys.
XOR	*Exclusive OR*, looginen poissulkeva tai operaatio.
YaST	*Yet Another Setup Tool*, SuSE-Linux-jakelujen asennus- ja konfigurointiohjelma.

1 Johdanto

Insinöörityön tarkoituksena on tehdä selvitys objektipohjaisen tallennusratkaisun toteutuksesta tietotekniikan etäresurssipalveluna (cloud computing). Aiheenvalinnan taustalla on oma työkuvani Capgemini Finland Oy:ssä tallennus- ja automaatioratkaisujen parissa sekä henkilökohtainen kiinnostus erilaisia tallennusratkaisuja kohtaan.

Aiheen tekee ajankohtaiseksi yritysmaailmassa meneillään oleva tuottavuus- ja kannattavuusvaatimuksien muutos, jossa laiteinvestoinneille asetetaan yhä tiukemmat tuottavuus- ja kannattavuusvaatimukset. Tämä on saanut yritykset kiinnostumaan siitä, että ne eivät enää maksa IT-infrastruktuurinsa omistamisesta vaan ostavat sen palveluna. Palveluna toteutettujen IT-infrastruktuurien tarve onkin saanut palveluntarjoajat kehittämään palveluina tarjottavia etäresurssitarjoamiaan. Tämä muutos palveluntarjoajien palvelutarjoamissa, on aiheuttanut aivan uusien tallennusratkaisujen ja -konseptien esille nousun viime vuosina. Uusien tallennusratkaisujen ja -konseptien esille nousu on myös tehnyt selvän muutoksen kuluttaja-asiakkaiden tapoihin tallentaa tietoa. Useat kuluttajat tallentavat jo nyt kaiken tietonsa etäresurssipalveluna tarjottuihin tallennusratkaisuihin, ja Gartner ennustaakin, että noin kolmannes kaikesta yksityisten kuluttajien tallentamasta tiedosta sijaitsee etäresurssipalveluissa vuonna 2016 [1]. Myös tapamme käsitellä tietoa ja vaatimukset sen saatavuudelle ovat muuttuneet. Enää ei riitä, että tieto sijaitsee jossain laitteessa ja se on mahdollisesti siirrettävissä eri laitteiden välillä. Nykypäivänä tiedon on oltava saatavilla millä tahansa päätelaitteella siten, että erillistä siirtotoimenpidettä laitteiden välillä ei tarvita. Ja tiedon on oltava saatavilla sekä jaettavissa muille käyttäjille 24/7. Sama muutos on jo havaittavissa yrityskäyttäjien tavoissa käsitellä ja tallentaa tietoa. Enää ei riitä, että tieto sijaitsee yrityksen tiedostonjakopalvelimella, josta sen voi hakea omalle työasemalle käsiteltäväksi, kun kone on yrityksen verkossa. Yrityskäyttäjätkin haluavat käsitellä ja tallentaa tietoa työasemalla, älypuhelimella tai muulla päätelaitteellaan, joka on sillä hetkellä paras ratkaisu tiedonkäsittelyyn.

Etäresurssipalveluihin tallennetun tiedon määrä onkin räjähtämässä käsiin, International Data Corporation (IDC) julkaisi vuoden 2012 lopussa niihin tallennetun tiedon määrästä olevan 2765 exatavua. Muutos vuoden 2010 tilanteeseen oli 2048 exatavua. Tarvitaan siis uusia tallennusratkaisuja, jotka kykenevät joustavasti skaalautumaan yhä isommiksi ja isommiksi. Objektipohjaiset etäresurssitallennuspalvelut ovat yksi suurimmista ja nopeimmin kehittyvistä tallennusratkaisuista, joilla kyetään vastaamaan tähän haasteeseen. [2, s. 20.]

Insinöörityö tutustuttaa lukijansa etäresurssipalvelun käsitteeseen, tallennusjärjestelmien yleisimpiin komponentteihin sekä etäresurssipalveluna toteutetun objektitallennuksen teoriaan sekä käytäntöön. Työ antaa lukijalle peruskäsityksen tallennusjärjestelmiin liittyvästä teoriasta ja komponenteista sekä siitä, miten nämä asiat liittyvät avoimen lähdekoodin etäresurssipalveluna toteutettuun tallennusjärjestelmään.

Teoriaosuudessa esitellään etäresurssipalvelun teoria sekä tallennusjärjestelmien yleisimmät komponentit ja käsitteet, jotka liittyvät tiedon tallentamiseen. Insinöörityössä keskitytään kuitenkin tiedon objektipohjaiseen tallennusratkaisuun, joten muiden etäresurssipalvelujen tallennusratkaisut ja varmuuskopiointiratkaisujen toteutus on suljettu insinöörityön alueen ulkopuolelle. Asioita pyritään käsittelemään käytännönläheisesti keskittyen eri vaihtoehtojen eroihin tallennuksen toteutuksessa. Työssä ei käydä syvällisesti lävitse, miten eri ratkaisujen teknologiat toimivat. Ratkaisut käsitellään siten, että lukija voi muodostaa kuvan kyseisen komponentin tai konseptin toiminnasta tallennuksessa sekä sen eri vaihtoehtojen eduista ja haitoista.

Käytännön osuudessa toteutetaan hajautettu objektipohjainen tallennusratkaisu avoimen lähdekoodin OpenStack Swift:llä näennäistetyillä VMware-palvelimilla.

Haluan tässä yhteydessä myös kiittää vaimoani, joka ymmärsi ja jaksoi sen, että elin erakoituneena työhuoneessani ilta illan jälkeen. Lisäksi kiitoksen sanat ohjaajalleni, Harrille. Hän oli ehtymätön lähde ajatuksia herättäville kysymyksille, sekä jaksoi haastaa kaikki mitä teoreettisesti väitin, palautteesi oli korvaamatonta tälle insinöörityölle.

2 Etäresurssipalvelun teoria ja käsitteet

2.1 Etäresurssipalvelun määritelmä

Vaikka käsitteenä etäresurssipalvelu onkin uusi, ajatus johon se pohjautuu, ei sitä ole. Jo vuonna 1961 tekoälykäsitteen isä John McCarthy esitti MIT:n satavuotisjuhlan puheessaan, että tulevaisuudessa tietotekniikkaa tarjottaisiin julkisena palveluna, kuten puhelinverkko tai sähkö. [3.]

Yleistäen etäresurssipalveluilla tarkoitetaan näennäisiä, käyttäjän kannalta rajoittamattomasti saatavilla olevia tietotekniikkaresursseja, kuten laskenta- ja tallennuskapasiteettia, tietoliikenneyhteyksiä, sovelluksia, palveluita tai näiden tietotekniikkaresurssien yhdistelmiä. Nämä resurssit ovat eri organisaatioiden tai yksittäisten käyttäjien käytettävissä ja hallittavissa standardisoitujen Internet- ja verkkoyhteyskäytäntöjen avulla. Käyttäjä hallinnoi kulloinkin käytössään olevien resurssien määrää täysin vapaasti, esimerkiksi palvelun hallintaportaalin kautta. Yksi yleisimmin siteeratuista määritelmistä etäresurssipalveluille on Yhdysvaltain julkishallinnon standardeja määrittävän elinkeinoministeriön alaisen National Institute of Standards and Technology (NIST) määritelmä pilvipalveluille.

> Etäresurssipalvelu on toimintamalli, joka mahdollistaa pääsyn kaikkialta saatavilla oleviin, tarpeenmukaisesti käyttöönotettaviin, vapaasti konfiguroitaviin jaettuihin tietotekniikkaresursseihin (kuten verkot, palvelimet, tallennusjärjestelmä, sovellus tai palvelu), jotka voidaan käyttöön tai poistaa käytöstä nopeasti [4, s. 6].

Etäresurssipalvelu ottaa siis eri teknologiat, palvelut ja sovellukset ja muodostaa niistä hyödykkeitä itsepalveluun. Englanninkielisessä nimensä Cloud Computing sana ”cloud” viittaa hyvin etäresurssipalveluun liittyviin kahteen ydinkonseptiin, määrittelemättömyyteen ja näennäisyyteen.

Määrittelemättömyys (Abstraction), tarkoitta että, etäresurssipalvelu erottaa toteutuksen tekniset yksityiskohdat sen käyttäjiltä. Sovelluksia ajetaan määrittämättömillä fyysisillä laitteilla, tieto tallennetaan määrittämättömään sijaintiin, järjestelmien hallinta on ulkoistettu ja käyttäjiltä on pääsy kaikkialta järjestelmään.

Näennäisyys (Virtualization), tarkoittaa, että etäresurssipalvelu näennäistää käytössään olevat resurssit, ja muodostaa niistä näennäisten resurssien jaettuja resurssipooleja. Käyttäjä voi hankkia järjestelmiä ja tallennustilaa joustavasti vain niitä tarvitessaan jaetusta infrastruktuurista, jota käyttävät samanaikaisesti useat käyttäjät. Laskutus tapahtuu vain käytön perusteella.

Yleisen määrittelyn lisäksi NIST määrittää tietotekniikan etäresurssipalvelulle seuraavat ominaispiirteet:

- itsepalvelullisuus (On-demand self-service)
- pääsy resursseihin eri päätelaitteilla (Broad Network Access)
- resurssien yhteiskäyttö (Resource pooling)
- nopea joustavuus (Rapid elasticity)
- käytön tarkka mittaaminen (Measured service)

Kuten John McCarthy totesikin yli 50 vuotta sitten, ehkä paras analogia etäresurssipalveluille löytyy vertaamalla sitä sähköverkkoon. Sähkönjakelussa käyttäjän ei tarvitse tietä, miten sähköä tuotetaan, siirretään (resurssien yhteiskäyttö) tai paljonko kapasiteettia verkosta löytyy (nopea joustavuus). Kunhan vain kytkee laitteensa sähköverkkoon (pääsy kaikilla päätelaitteilla ja itsepalvelullisuus) ja maksaa siitä sähköstä, mitä on käyttänyt (käytön tarkka mittaaminen).

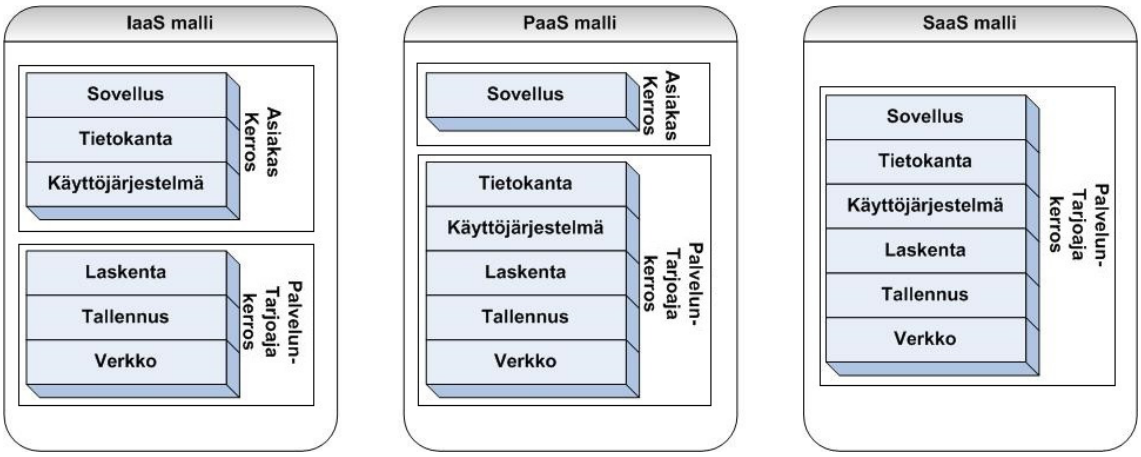
2.2 Etäresurssipalveluluiden palvelumallit

NIST:n mukaan, etäresurssipalvelu tarjoamat luokitellaan pääsääntöisesti kolmeen eri palvelumalliin [4, s. 6–7], nämä ovat:

- verkkosovelluspalvelu (SaaS – Software as a Service)
- alustapalvelu (PaaS – Platform as a Service)
- infrastrukstuuripalvelu (IaaS – Infrastructure as a Service)

On kuitenkin huomattava, että näiden lisäksi on määritelty useita eri palvelumalleja, joiden voidaan katsoa olevan näiden pääluokkien alaluokkia. Insinöörityön voidaan katsoa käsittelevän erään IaaS-palvelumallin alaluokan tallennuspalvelun (STaaS – Storage as a Service) toteutusta ja ratkaisuja. Kuten edellä mainituista etäresurssipalvelumallien nimistä voidaan havaita, eri etäresurssipalvelujen nimistandardiksi on muodostunut nimeäminen, jossa ensimmäinen kirjain kuvaa palvelun tyypin ja siihen lisätään as-a-Service. Edellä mainittujen NIST-standardissa mainittujen etäresurssipalvelun pääluokkien lisäksi on olemassa muita standardoituja etäresurssipalvelumalleja, kuten ITU-standardoimat tietoverkkopalvelu (NaaS – Networking as a Service) ja viestintäpalvelu (CaaS – Communication as a Service) [5, s. 7].

Se mitä NIST-standardoimat etäresurssipalvelumallit ovat, on esitelty seuraavissa alaluvuissa tarkemmin. Alla oleva kuva 1 esittää, miten eri etäresurssipalvelumalleissa ylläpitovastuu ja -mahdollisuudet jakautuvat etäresurssipalvelun tilaajan sekä etäresurssipalvelun toteuttajan välillä.



Kuva 1. Etäresurssipalvelujen palvelumallit.

Verkkosovelluspalvelu (SaaS)

SaaS-etäresurssipalvelussa palveluntarjoaja tarjoaa sovelluksen, kuten esimerkiksi CRM-, sähköposti-, pikaviestintä-, tekstinkäsittely-, kirjanpitosovelluksen sekä sovelluksen mahdollisesti tarvitsemat tietokannat palveluna käyttäjälle. Sovelluksen omistamisen, asentamisen, ylläpidon ja päivittämisen sijasta käyttäjä ostaa tai saa käyttöönsä palveluntarjoajan ylläpitämän sovelluksen. Sovellusta ajetaan palveluntarjoajan etäresurssipalvelussa, ja palveluntarjoaja on vastuussa sovelluksen sekä infrastruktuurin ylläpidosta.

Sovellus on useimmiten saatavilla useilla erilaisilla päätelaitteilla, kuten esimerkiksi älypuhelimella, selaimella tai sovellusrajapinnalla. Käyttäjä ei siis hallinnoi sovellusta tai pysty vaikuttamaan, millaisella alustalla sovellus toteutetaan tai miten sovellus on konfiguroitu. Tosin useat palveluntarjoajat tarjoavat jonkin asteista kustomointia varsinkin yrityssovelluksille. Käytännössä tämä useimmiten tarkoittaa sitä, että sovelluksen esitystapa kustomoidaan esitysrajapinnassa esimerkiksi tilaajayrityksen logoilla, väreillä tai muulla vastaavalla.

Tunnettuja SaaS etäresurssipalveluja ovat:

- Gmail-sähköpostipalvelu
- GoogleDocs-tekstinkäsittelypalvelu
- SalesForce.com-CRM-palvelu
- SQL Azure-tietokantapalvelu
- Facebook-sosiaalinen verkosto

Alustapalvelu (PaaS)

PaaS-etäresurssipalvelussa palveluntarjoaja tarjoaa tilaajalle mahdollisuuden tuottaa ja asentaa etäresurssipalveluunsa päältä tarjottavia sovelluksia. Palveluntarjoaja antaa tilaajan käyttöön tietyt ohjelmointikielet, kirjastot, API:t, palvelut ja työkalut, joiden ylläpidosta vastaa palveluntarjoaja. Lisäksi palveluntarjoaja vastaa alla olevasta etäresurssipalveluinfrastruktuurista, jolla palvelu tuotetaan. Tilaajan vastuulle jää mahdollinen sovelluskehitys, sovelluksen asentaminen ja konfigurointi.

Tunnettuja PaaS etäresurssipalveluja ovat:

- GoogleAppEngine
- Windows Azure Platform
- Force.com
- Facebook

Infrastruktuuripalvelu (IaaS)

IaaS-etäresurssipalvelussa palveluntarjoaja tarjoaa tilaajalle mahdollisuuden hankkia laskenta- ja tallennuskapasiteettia, tietoliikenneyhteyksiä ja muita perustavanlaatuisia tietokoneresursseja, joihin kuluttaja voi asentaa mielivaltaisesti ohjelmistoja, kuten käyttöjärjestelmiä ja sovelluksia. Kuluttaja ei ylläpidä ja hallinnoi etäresurssipalvelun fyysisiä laitteita tai ohjelmistoja, joilla edellä mainitut resurssit toteutetaan. Palveluntarjoajan toimesta tarjottu kapasiteetti on yleensä pitkälle näennäistetty ja tilaaja hankkii ja hallinnoi tarvitsemiansa resursseja tästä loogisesta näennäiskerroksesta API:en ja/tai hallintaliittymän kautta. Korkea näennäisyysaste tarjoaa hyvän skaalautuvuuden ja sen toteutuksessa käytetty suuri automaation määrä mahdollistaa sen, että vuorovaikutusta palveluntarjoajan ja tilaajan henkilöstön välillä ei välttämättä synny missään muodossa palvelunkäytön elinkaaren aikana.

Tunnettuja IaaS etäresurssipalvelujen tuottajia ovat

- Amazon Elastic Compute Cloud (EC2)
- Rackspace
- HP
- Eucalyptus
- Terremark

2.3 Etäresurssipalvelun sijoitusmallit

Kuten edellisistä luvuista voidaan havaita, etäresurssipalveluja on tarjolla useilla erilaisilla palvelumalleilla ja useilta erilaisilta julkisilta toimijoilta. Mutta etäresurssipalvelu ei välttämättä tarkoita julkiselta palveluntuottajalta hankittua palvelua. NIST on määritellyt etäresurssipalveluille neljä erilaista sijoittelumallia. Nämä sijoittelumallit määrittävät, missä etäresurssipalvelu on toteutettu ja kenelle sitä toimitetaan. Etäresurssipalvelun sijoittelumallit esitellään seuraavissa luvuissa.

Julkinen (public) etäresurssipalvelu

Julkisen etäresurssipalvelun sijoittelumallissa etäresurssipalvelun toteutus on tarjottu julkiseen käyttöön kokonaan tai osittain. Palvelua voi tuottaa ja hallinnoida yritys, akateeminen yhteisö, julkinen organisaatio tai mikä tahansa näiden yhdistelmistä. Etäresurssipalvelun toteuttava laitteisto sijaitsee palveluntarjoajan hallinnoimissa tiloissa.

Yleensä käyttäjät hallinnoivat ja hyödyntävät julkisen etäresurssipalvelun palveluja Internetin ylitse. Palveluntarjoajat, kuten esimerkiksi Amazon, tarjoavat myös mahdollisuutta hyödyntää näitä palveluja kiinteiden yhteyksien ylitse [6.]

Julkisen etäresurssi palvelun suurimpana etuna voidaan pitää, koon tuomia taloudellisia näkökohtia. Pääomakulut, ylläpitokulut sekä muut kustannukset jakautuvat kaikkien käyttäjien kesken. Tämä mahdollistaa erittäin suuren skaalautuvuuden tarjoamisen käyttäjille, kuitenkin kohtuullisilla kustannuksilla. Tätä voidaankin juuri pitää suurimpana syynä sille, että yritysmaailma on kiinnostunut etäresurssipalvelujen käytöstä viime vuosina. Yrityksen käyttöön saadaan kulloisenkin tarpeen mukaan skaalautuva, tarvittaessa hyvinkin iso ICT-infrastruktuuri ilman suuria laite- ja sovellusinvestointeja. Mutta valitettavasti tämäkään pilvi ei ole pelkästään kullalla silattu. Julkinen etäresurssipalvelu tuo mukanaan myös joukon riskejä. Yleisimmiksi riskeiksi on noussut kysymykset tietoturvasta, verkkoyhteyksien suorituskyvyistä ja eri etäresurssipalvelujen yhteensopivuusongelmat. Kattavan standardoinnin puutuessa, ei ole mitenkään varmaa, että esim PaaS- tai IaaS-palveluiden käyttäjän on mahdollista siirtyä palveluntarjoajalta toiselle. Vuonna 2013 esille noussut NSA:n PRISM-hankkeen paljastuminen, nosti taas hyvin esille sen kysymyksen, ovatko tiedot riittävän turvassa jonkun muun ylläpitämässä infrastruktuurissa? Tähän kysymykseen vastaaminen, ei kuulu tämän insinöörityön laajuuteen, mutta kysymys on hyvä esimerkki asioista, joita julkiseen etäresurssipalveluun siirtyvän on syytä pohtia.

Yksityinen (private) etäresurssipalvelu

Yksityisen etäresurssipalvelun sijoittelumalli on julkisen etäresurssipalvelun vastakohta. Sen käyttäjänä on rajattu käyttäjäjoukko, kuten esimerkiksi yrityksen työntekijät tai akateemisen yhteisön yksiköt, mutta palvelu on aina varattu vain tämän käyttäjäjoukon käyttöön. Yleensä palvelua tuottaa ja hallinnoi sama yritys, akateeminen yhteisö tai julkinen organisaatio, joka myös käyttää palvelua. Standardointi tuntee yksityisestä etäresurssipalvelusta kaksi vaihtoehtoa:

Sisäisen yksityisen etäresurssipalvelun, esimerkkinä voitaisiin mainita yrityksen informaatiotekniikan yksikkö, joka tuottaa palvelua yrityksen muille yksiköille. Etäresurssipalvelun toteuttava laitteisto sijaitsee yrityksen hallinnoimissa tiloissa.

Ulkoissessa yksityisessä etäresurssipalvelussa, yritykseen yksityiseen käyttöön on hankittu osa jonkin julkisen etäresurssipalvelun tuottajan etäresurssipalvelusta. Palveluntarjoaja omistaa, hallinnoi ja ylläpitää etäresurssipalvelua tuottavaa infrastruktuuria. Mutta pääsy tuohon palveluun ja sen toteuttaviin laitteistoihin on rajattu vain yrityksen käyttöön.

Yhteisöllinen (community) etäresurssipalvelu

Yhteisöllinen etäresurssipalvelu on sijoittelumalli, jota voidaan pitää julkisen ja yksityisen etäresurssipalvelun välimuotona. Palvelua tuotetaan määritellylle käyttäjien yhteisölle, jolla on yhteiset vaatimukset ja määrittely palvelulle. Palvelua voi tuottaa yksi tai useampi tämän yhteisön jäsenistä, tai se voi olla ulkopuoliselta palveluntuottajalta hankittu yksityinen etäresurssipalvelu. Toisin kuin täysin julkinen etäresurssipalvelu tämä on käyttäjilleen kalliimpi vaihtoehto, mutta tarjoaa käyttäjille suuremman vaikutusmahdollisuuden kuinka ja millä palvelua tuotetaan. Esimerkkinä käyttäjäryhmästä, jolle yhteisöllisen etäresurssipalvelu voisi olla soveltuva sijoittelumalli, on julkishallinnon yksiköt. Niillä on usein samat tietoturva, menettelytapa ja lainsäädännölliset vaatimukset tuotetulle palvelulle, joten näiden yksiköiden etäresurssipalvelun toteuttaminen yhteisöllisenä palveluna vähentäisi yhteen yksikköön kohdistuvia kustannuksia.

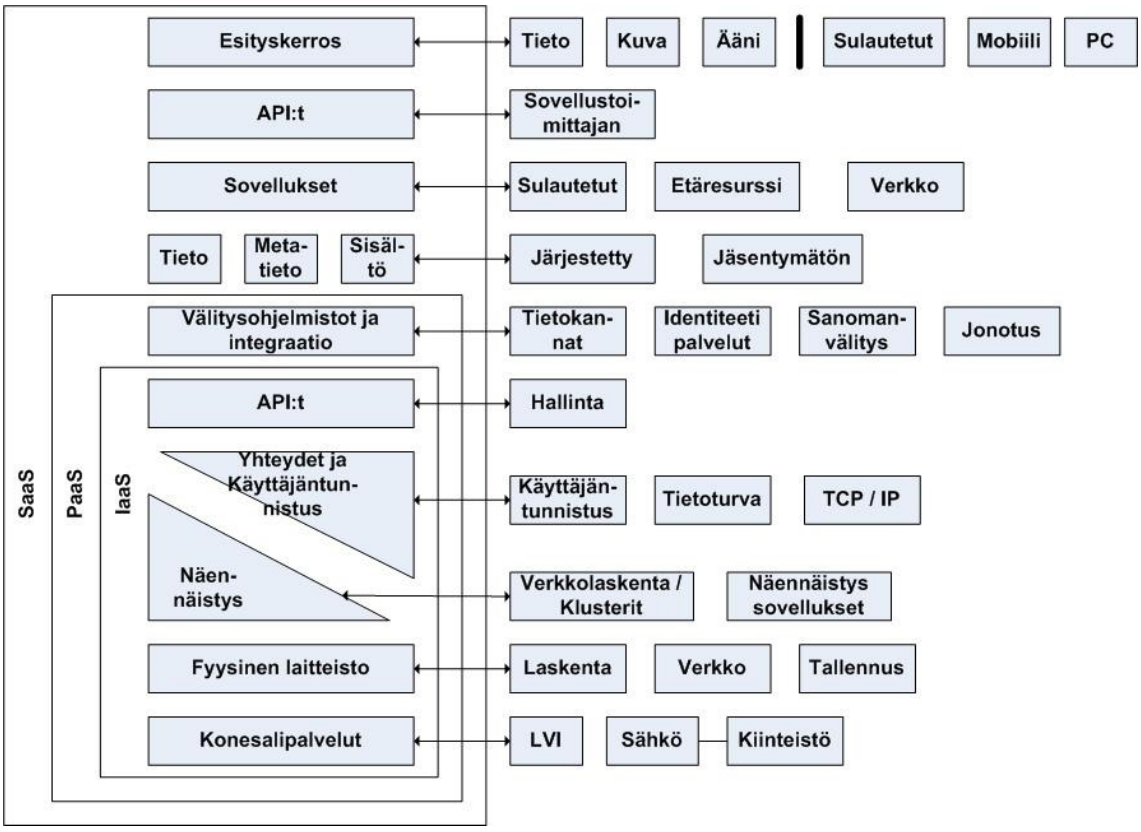
Sekamuotoinen (hybrid) etäresurssipalvelu

Sekamuotoinen etäresurssipalvelu on sijoittelumalli, jossa palvelua toteutetaan edeltävien mallien yhdistelmänä. Eri sijoittelumallien mukaan toteutetut etäresurssipalvelut, jotka muodostavat tämän sekamuotoisen etäresurssipalvelun, ovat itsenäisiä kokonaisuuksia. Nämä etäresurssipalvelut on sidottu yhteen standardoinnilla ja toteuttavien resurssien yhdenmukaistamisella. Tämä mahdollistaa sen, että tieto ja eri palvelun ilmentymät voivat siirtyä eri mallien välillä. Esimerkkinä tästä voidaan antaa yksityinen etäresurssipalvelu, joka on toteutettu siten, että kapasiteettia voidaan tarvittaessa laajentaa julkisesta pilvestä.

Edellä esitellyn NIST-standardin mukaisen etäresurssipalvelun määrittelyn lisäksi on olemassa muita määrittelyjä etäresurssipalvelulle. Toinen yleisesti tunnettu on Open Groupin Jericho Forum-kuutiomalli [7.]. Koska tämän insinöörin tarkoituksena ei ole etäresurssipalvelun määrittely, sitä tai muita määrittelyjä ei esitellä syvemmin.

2.4 Etäresurssipalvelun tietotekninen infrastruktuuri

Etäresurssipalvelun toteuttava infrastruktuuri on kokoelma laitteistoja ja sovelluksia, jotka mahdollistavat aikaisemmin esitellyn viiden ominaispiirteen toteuttamisen. Etäresurssipalvelun infrastruktuuri jaetaan yleensä neljään eri kerrokseen. Se miten erilaiset komponentit ja palvelut liittyvät etäresurssipalvelun eri kerroksiin, on esitelty etäresurssipalvelun viitemallina toimivassa kuvassa 2.



Kuva 2. Etäresurssipalvelun viitemalli.

Fyysinen laitteisto

Fyysinen laitteisto koostuu fyysisistä palvelimista, tallennusjärjestelmistä ja verkkolaitteista. Fyysiset palvelimet on yhdistetty toisiinsa, tallennusjärjestelmiin ja käyttäjiin erilaisten verkkojen avulla.

Etäresurssipalvelua tarjoava palveluntarjoaja voi käyttää fyysisiä tietoteknisiä resursseja yhdestä tai useammasta palvelinkeskuksesta. Palvelinkeskukset on yleensä hajautettu joko paikallisesti ja mahdollisesti myös maantieteellisesti. Varsinkin maantieteellisen hajauttamisen toteuttaminen tallennusratkaisuissa on vaatinut uusien tallennustekniikkojen käyttöönottoa.

Tässä insinöörityössä tullaan käsittelemään juuri tämän kerroksen tekniikkoja sekä ratkaisuja, joilla käyttäjän informaation sisältävä tieto tallennetaan.

Näennäistetty infrastruktuuri

Etäresurssipalvelun palveluntarjoajat hyödyntävät merkittävästi erilaisia näennäistystekniikkoja ja -ratkaisuja. Näillä rakennetaan fyysisen laitteiston päälle näennäisyyskerros, joka mahdollistaa resurssivarantojen muodostamisen, palvelun elastisuuden ja yhteiskäyttöisyyden. Esimerkki tällaisesta resurssivarannosta voisi olla fyysisen palvelinryppään yhteenlaskettu laskentakapasiteetti. Etäresurssipalvelun laskentakapasiteettia käyttävä asiakas antaa esimerkiksi ohjelmistonsa laskennan tämän resurssivarannon suoritettavaksi tietämättä tai voimatta vaikuttaa, millä prosessoriytimellä palvelinryppäässä suoritus itse asiassa tapahtuu. Se voi olla palveluntarjoajan tallennusjärjestelmien tallennuskapasiteeteista muodostettu resurssivaranto, josta käyttäjälle tarjotaan näennäistetty looginen kiintolevy. Käyttäjä ei tiedä tai voi vaikuttaa, mille fyysiselle kiintolevylle tiedot todellisuudessa tallennetaan.

Sovellukset ja ohjelmistot

Sovellukset- ja ohjelmistot-kerros sisältää etäresurssipalvelun toteuttavat sovellukset, ohjelmistot ja tietokannata.

Etäresurssipalvelun hallinta- ja palvelutuotantotyökalut

Hallinta- ja palvelutuotantotyökalu kerroksessa sijaitseva hallintakerroksen, sovellukset joilla palvelua hallitaan sekä palvelutuotannon toteuttamisen vaativat työkalut. Yleisesti ne jaetaan kolmeen ryhmään,

- fyysisen ja näennäisinfrastruktuurin hallintasovellukset
- palvelun eri komponenttien yhdistämisen vaatimat sovellukset
- pääsynhallintasovellukset

Luokittelu perustuu eri kerrosten sovellusten suorittamiin toimintoihin. Näiden sovellusten yhdistelmät mahdollistavat sen, että palvelun toimittaminen, hallinta ja laskutus on automaattista.

3 Tallennusjärjestelmien keskeisimmät komponentit

Koska käsite tallennusjärjestelmä on erittäin laaja kokonaisuus, joka koostuu useista erilaisista komponenteista, tekniikoista, protokollista ja niiden yhdistelmistä, tässä luvussa käsitellään ja esitellään niistä yleisimmin käytössä olevat. Luvun tarkoituksena on antaa lukijalle käsitys siitä, miten eri osaset toimivat ja minkä toiminnallisuuden ne toteuttavat tallennusjärjestelmässä.

3.1 Massamuistilaitteet (mass storage device)

Tallennusvälineet muodostavat tallennusjärjestelmän ytimen, jolle tiedot tallennetaan. Tallennusvälineet ovat erilaisia magneettisia, optisia ja puolijohdemuistilaitteita. Kiintolevyt, nauha-asemat ja levykkeet ovat magneettimuistilaitteita, kun taas CD ja DVD ovat optisia muistilaitteita. USB-asemat ja flash-asemat ovat taas esimerkki puolijohdemuistilaitteista.

Aikaisemmin nauha-asemat olivat yleisin tallennusvaihtoehto, mutta niissä on useita rajoituksia suorituskyvyssä. Tämä ja edullisten suurikapasiteettisten kiintolevyjen saatavuus on johtanut siihen, että niiden käyttö tallennusvälineenä on nykyään merkittävästi vähentynyt. Myöskään optisia muistilaitteita ei yritysympäristöissä käytetä. Tämä johtuu näiden tallennusvälineiden pienestä kapasiteetista sekä rajallisesta nopeudesta verrattuna magneettisiin tallennusvälineisiin. Seuraavaksi esitellään nykypäivänä yleisimmin yrityskäytössä olevat tallennusvälineet.

3.1.1 Mekaaninen kiintolevyasema (HDD)

Mekaaninen kiintolevyasema on edelleen yleisin tallennusväline tiedon tallennukseen nykyisin käytössä olissa tallennusjärjestelmissä. Kiintolevy koostuu seuraavista komponenteista: yhteiseen akseliin (spindle) kiinnitetyt magneettipinnoitteiset levylautaset (platter), ohjainvarteen (actuator arm) kiinnitetyistä luku-/kirjoituspäistä (Read/Write head), tämän ohjaimesta (Actuator) ja ohjainkortista (controller board), joka sisältää muun muassa välimuistin. [8, s 31–33.]

Kirjoitus- ja lukuoperaatiot kiintolevylle tapahtuvat siten, että isäntä (Host) lähettää kirjoitus/lukuoperaation kiintolevyasemalle, jolloin kiintolevyaseman ohjainkortti tallettaa operaation välimuistiinsa (cache). Tämän jälkeen operaatio välitetään ohjainvarteen kiinnitetylle luku-/kirjoituspäälle, joka lukee tai kirjoittaa binääriarvon levylautasen magneettipäällysteeseen. Levylautanen on päällystetty molemmilta puolilta magneettisella pinnoitteella, joten kummallekin puolelle voidaan haluttaessa suorittaa eri operaatio samanaikaisesti. [8, s. 31–33.]

Akseli, johon levylautaset on kiinnitetty, pyörii useita tuhansia kierroksia minuutissa (rpm). Yleisesti kiintolevyaseman suorituskykyä kuvataan juuri akselin pyörimisnopeudella. Yleisimmät pyörimisnopeudet ovat 5 400 rpm, 7 200 rpm, 10 000 rpm ja 15 000 rpm. Vaikka kiintolevyn suorituskyky teoriassa nouseekin pyörimisnopeutta nostamalla, se ei ole kannattavaa yli 15 000 rpm. Suunnittelijat ovat havainneet tutkimuksissa, että yli 15 000 rpm pyörivien levyjen hyötysuhde ei nouse riittävästi virrankulutukseen nähden, jotta nopeutta kannattaisi nostaa yli tämän arvon. [9.]

Ohjainkortti vastaa kiintolevyn toimintojen, kuten virransyötön hallinnasta, siitä mistä kohtaa levyjä tietoa kulloinkin luetaan/kirjoitetaan sekä kommunikaatiosta isännän ja kiintolevyn välillä.

Kukin levylautanen on jaettu samankeskeisiin uriin (track), joilla sijaitseviin sektoreihin (sector) tiedot kirjoitetaan. Urat on numeroitu alkaen levylautasen ulkoreunasta siten, että uloimman uran numero on nolla. Näiden urien lukumäärä on toinen levyn suorituskykyyn merkittävästi vaikuttava tekijä. Urien lukumäärä ilmoitetaan TPI (Tracks per Inch)-arvona, joka kertoo, kuinka tiheässä näitä uria on levyllä. Sektori on tyypillisesti pienin osoitettava yksikkö levyllä ja yleensä yksi sektori sisältää 512 tavua tietoa, tosin joissakin levyissä tätä kokoa on kasvatettu. Koko levypakan lävitse samassa kohdassa olevat urat muodostavat sylinterin (cylinder), jota käytetään luku-/kirjoistuspään sijainnin määrittämiseen.

Kirjoitettavan tai luettavan kohdan määrittäminen levyllä tapahtuu CHS (Cylinder, Head, Sector) -osoitteen avulla. Tämä osoitusmetodi aiheuttaa kuitenkin sen ongelman, että levyä käyttävän isäntäkoneen käyttöjärjestelmän täytyy olla tietoinen kiintolevyn geometriasta eli siitä kuinka monta sylinteriä, levypintaa ja sektoria levyllä on. Tämän ongelman kiertämiseksi on nykyään siirrytty LBA (Logical Block Address) -osoitusmetodiin, jossa sektorit on nimetty juoksevalla numeroinnilla. Isäntäkoneen käyttöjärjestelmän täytyy tietää vain levyn koko ja käytetty sektori koko osoiteavaruuden määrittämiseksi. Kiintolevyn ohjainkortin vastuulle jää muuntaa isäntäkoneen antaman LBA-osoitteen levyn käyttämäksi CHS-osoitteeksi.

Kuten aikaisemmin todettiin, kiintolevyn suorituskykyyn vaikuttavat levypakkaa pyörittävän akselin pyörintänopeus ja urien lukumäärä. Muita suorituskykyyn vaikuttavia asioita ovat kiintolevyn mekaanisesta toteutuksesta johtuvat hakuaika (seek time) ja pyörintäviive (rotational latency). Hakuaika on aika, joka kuluu siihen, että ohjainvarsi siir-

tää luku-/kirjoituspään oikean uran kohdalle. Pyörintäviive on taas aika, joka kuluu siihen, että oikea sektori tulee luku-/kirjoituspään kohdalle. Yleensä ohjainkortin mikrokoodiohjelma pyrkii optimoimaan luku-/kirjoituspyyntöjen suoritusjärjestyksen siten, että edellä mainitut ajat ovat mahdollisimman lyhyet. Lisäksi kiintolevyn suorituskykyyn vaikuttaa merkittävästi I/O (Input/Output)-ohjaimen (controller) käyttöaste (utilization). Nyrkkisääntönä voidaan pitää, että niin kauan kuin käyttöaste on alle 70 %, kiintolevyn vasteaika (response time) on hyväksyttävällä tasolla. Tähän vaikuttaa eniten mikrokoodin (firmware) ja mekaanisten komponenttien laatu, eli kuinka nopeasti kiintolevyn laitteisto pystyy käsittelemään saadut I/O-pyynnöt. Kuten todettua, edellä mainitut suorituskykyä heikentävät viiveet ovat siis perinteisen kiintolevyn mekaniikasta johtuvia eikä niitä voida välttää. Tästä johtuen on viime vuosina suurta suorituskykyä vaativissa kiintolevyissä siirrytty käyttämään puolijohdelevyasemia, jotka esitellään seuraavassa luvussa.

3.1.2 Puolijohdelevyasema (EFD, Enterprise Flash Drive)

Perinteisesti jos tallennusjärjestelmän suorituskykyä on haluttu parantaa, on ainoa ratkaisu ollut lisätä lisää rinnakkaisia kiintolevyjä, joille operaatioiden suoritusta on jaettu. Tämän vaihtoehdon korvaajaksi ovat nousseet puolijohdemuistielementeillä toteutetut puolijohdelevyasemat. Toisin kuin perinteinen kiintolevy, ne eivät sisällä mekaanisia komponentteja, joten hakuaikaa tai pyörintä viivettä näissä ei ole. Tästä johtuen puolijohdelevyasemat soveltuvat erityisen hyvin pienissä lohkoissa tapahtuvaan satunnaisiin kirjoitus- ja lukuoperaatioihin. Suorituskyvyltään puolijohdelevyasemat tarjoavat 30 kertaisen suorituskapasiteetin, yhden kymmenesosan vasteajan sekä 38 % pienemmän virrankulutuksen verrattuna vastaaviin perinteisiin kiintolevyihin.

Puolijohdelevyasema koostuu seuraavista komponenteista: ohjainkortti, I/O-rajapinta, massamuisti, joka koostuu useista muistisiruista (memory chip) sekä välimuistista (cache). Fyysisiltä ulkomitoiltaan nämä vastaavat perinteisiä kiintolevyasemia, joten niitä voidaan käyttää täysin vapaasti samoissa kiintolevypaikoissa kuin perinteisiäkin.

Kirjoitus- ja lukuoperaatiot puolijohdelevyasemilla tapahtuvat siten, että isäntä lähettää kirjoitus/lukuoperaation I/O-rajapinnan kautta puolijohdelevyasemaan, jolloin ohjainkortti tallettaa operaation välimuistiinsa. Tämän jälkeen operaatio välitetään rinnakkaisten I/O-väylien kautta massamuistiin, joka on useimmiten NAND-muistisiruista koostuva ryhmästö (cluster). Rinnakkaisten I/O-väylien lukumäärä, joilla luku-/kirjoitusoperaatiota

välitetään muistisiruille, ovat yksi puolijohdelevyasemien suorituskyvyn määrittävistä tekijöistä. Tyypillisesti puolijohde levyasemissa näitä väyliä on 8–24 kappaletta.

Mikrosirut, jotka muodostavat puolijohdelevyaseman massamuistin, on ryhmitelty loogisiin sivuihin (pages) ja lohkoihin (blocks). Sivu on pienin muistiyksikkö, johon tiedon lukuoperaatioita voidaan kohdistaa. Sivujen kokoa ei ole standardisoitu, mutta yleisiä sivukokoja ovat 4 kilotavua ja sen kerrannaiset. Sivuista on muodostettu lohkoja jotka ovat taas pienin yksikkö, mihin kirjoitus- ja tyhjennysoperaatio voidaan kohdistaa. Tätä lohkoa ei pidä sekoittaa perinteisen kiintolevyn 512 tavun lohkoon, koska puolijohdelevyaseman massamuistin lohko koostuu joko 32 sivusta tai sen monikerran määrästä sivuja.

Kirjoitettavan tai luettavan tiedon osoitus tapahtuu LBA-osoituksella. Koska puolijohdelevyt joutuvat emuloimaan mekaanisten kiintolevyjen osoitusta, yksittäinen sivu on jaettu useamman perättäisen lohkon alueelle, jolloin eri tietolohkojen alueilla sijaitsevilla sivun osilla on peräkkäinen osoite. Esimerkiksi 4 kilotavun sivu on jaettu kahdeksan 512 tavun alueena perättäisten lohkojen alueelle. [10.]

3.1.3 Magneettinauha-asemat

Magneettinauha-asemat (tape storage) ovat edelleen käytössä oleva tallennusratkaisu tapauksissa, joissa halutaan tallentaa muuttumatonta tai erittäin harvoin muuttuvaa tietoa edullisesti pitkiä aikoja. Mutta koska edulliset suuri kapasiteettiset mekaaniset kiintolevyt ovat nykypäivänä korvanneet magneettinauha ja -asemat muuttumattoman tiedon lyhytaikaisessa (muutamia kuukausia) tallennuksessa, esitellään ne vain lyhyesti. Tyypillisesti magneettinauharatkaisut koostuvat seuraavista komponenteista: magneettinauhakasetti, kirjoitus-/lukuyksikkö, automatiikkaratkaisu, jolla kasetit siirretään varastotilasta kirjoitus-/lukuyksikköön, sekä ohjauselektroniikka, joka huolehtii tiedon hausta ja kirjoituksesta oikeaan kasettiin ja oikean kasetin syöttämisestä kirjoitus-/lukuyksikköön.

Vaikka kyseessä onkin harvinaisemmaksi muuttuva teknologia, magneettinauhatallennus on edelleen kilpailukykyisin vaihtoehto suurten muuttumattomien tietomäärien pitkäkestoiseen tallennukseen. [11.]

3.2 RAID-teknologia

Kuten aikaisemmin mekaanisia ja puolijohdekiintolevyjä esittelevissä kappaleissa kuvailtiin, ne ovat erilaisista mekaanisista ja elektronisista komponenteista koostuvia laitteita. Kuten millä tahansa tällaisella laitteella, niillä on oma elinkaarensa, jonka päätteeksi laite hajoaa. Elinkaaren päätyttyä niillä oleva informaatio ei ole normaalitoimin pelastettavissa. Toki on olemassa yrityksiä kuten Kroll Ontrack, jotka pystyvät pelastamaan informaation hajonneelta kiintolevyltä. Mutta tällainen toiminta on aikaa vievää, ja informaation puuttuminen aiheuttaa haittaa sitä tarvitsevalle yksilölle tai organisaatiolle. Tämän haitan minimoimiseksi on yksittäisistä kiintolevyistä yleensä muodostettu joko laitteistopohjaisia tai sovelluksella toteutettuja RAID (Redundant Array of Independent Disks) -levyryhmiä. RAID-teknologia mahdollistaa sen, että joukosta voi hajota yksi tai useampi levy, mutta tieto on saatavilla tai palautettavissa ryhmän muista levyistä.

RAID-teknologia perustuu Norman Ken Ouchin IBM:lle patentoimaan teknologiaan, jossa määritellään järjestelmä, jolla voidaan palauttaa hajonneen muistilaitteen tiedon. Tässä patentissa käytännössä määriteltiin RAID-teknologiat 1, 4 ja 5. Käsitteen RAID määrittelivät Berkeleyn yliopiston tutkijat Patterson, Gibson ja Katz vuonna 1987 julkaisemassa tutkielmassaan. Tuossa tutkielmassa määritettiin käsite vikasietoiselle edullisten levyjen joukolle. Tosin tämän tutkielman lähtökohtana oli ennemminkin, miten löytää ratkaisu sille, että tuohon aikaan palvelinkäytössä olleet SLED (Single Large Expensive Disk) -kiintolevyt olivat erittäin kalliita eikä niiden suorituskyky ollut hyvä. Tutkielma esitteli teorian halvoista kotikoneiden käyttöön tarkoitetuista kiintolevyistä muodostetulle levyryhmälle. Tämä levyryhmä oli suorituskykyisempi ja halvempi kuin vastaavan kokoinen SLED-kiintolevy. Lisäksi levyryhmä oli vikasietoisempi kuin yksittäinen SLED-levy. RAID-levyryhmistä onkin tämän jälkeen muodostunut standardi, jolla tallennusjärjestelmät voivat hajauttaa kirjoitus- ja lukuoperaatiot useille fyysisille kiintolevyille, jolloin vikasietoisuus ja suorituskyky paranee. [12, s. 4–6.]

RAID-levyryhmän toteutukselle on olemassa kaksi eri toteutustapaa, sovelluspohjainen (Software RAID) ja laitteistopohjainen (Hardware RAID). Sovelluspohjaisessa toteutuksessa RAID toiminnallisuus toteutetaan yleensä käyttöjärjestelmätasolla, mutta on olemassa myös vaihtoehtoja, jossa toteutus tapahtuu BIOS (Basic Input/Output System) -tasolla. Sovelluspohjaisen RAID-toteutuksessa on seuraavanlaisia haittapuolia:

- Vaikutus keskusyksikön suorituskykyyn, RAID-toteutus vaatii prosessoriaikaa.

- Rajoitukset käytettävissä olevissa RAID-tasoissa. Sovelluspohjaisissa RAID-toteutuksissa on yleensä rajoitteita sille, mitkä RAID-tasot ovat käytettävissä.

- Yhteensopivuusongelmat. Koska sovelluspohjainen RAID on yleensä toteutettu käyttöjärjestelmätasolla, käyttöjärjestelmäpäivitykset tai RAID-sovelluksen päivitykset saattavat aiheuttaa haasteita.

Laitteistopohjaisissa RAID-toteutuksissa, käytetään yleensä erityistä RAID-ohjainkorttia, jolla toiminnallisuus toteutetaan. Ohjainkortti liitetään yleensä johonkin emolevyn laajennusväylään ja kiintolevyt liitetään tähän ohjainkorttiin.

RAID-toiminnallisuuden toteutuksiin liittyy kiinteästi kolme eri teknologiaa, joilla eri RAID-tasojen toiminnallisuudet on toteutettu.

- Lomituksessa (striping) tietoa jaetaan useammille fyysisille kiintolevyille, joita käytetään yhtäaikaisesti rinnakkain.

- Peilauksessa (mirroring) tieto kirjoitetaan yhtäaikaisesti yhdelle tai useammalle fyysiselle kiintolevylle.

- Pariteetti (parity) on metodi, jolla useammalle fyysiselle kiintolevylle lomitettua tietoa voidaan suojata levyrikolta ilman peilauksen vaatimaa kaksinkertaista levykapasiteettiä. Levyryhmässä on aina vähintään yksi fyysinen kiintolevy, jolle muiden levyryhmän fyysisten levyjen sisältämästä tiedoista lasketaan XOR (exclusive OR) -operaatiolla pariteetti. Tämä mahdollistaa menetetyn kiintolevyn tietojen palautuksen.

Nykyisellään RAID-standardi tuntee 6 erilaista käytössä olevaa RAID-päätasoa [13.]. Näistä tasoista on olemassa muutamia standardisoituja variaatioita, joissa päätasoja yhdistellään halutun suojauksen saavuttamiseksi. Standardoidut ovat:

RAID-0, lomitettu levyryhmä ilman pariteettia, tieto lomitetaan useammalle levylle siten, että se jakautuu tasaisesti kaikille levyille. Luku-/kirjoitusnopeus on yhden ryhmään kuuluvan levyn luku-/kirjoitusnopeus kertaa ryhmän levyjen lukumäärä, joten tämä on suorituskykyisin RAID taso. Ryhmä ei ole vikasietoinen, mikäli yksikin levy ryhmästä hajoaa, koko ryhmän sisältämä tieto menetetään. Levyryhmästä voidaan hyödyntämään kaikki kapasiteetti. [13.]

RAID-1, peilattu levyryhmä on yksi yleisimmin käytetyistä RAID levyryhmistä. Kirjoituksessa kirjoitusoperaatio tehdään kaikille ryhmän levyille. Teoriassa ryhmän lukunopeus voi olla ryhmään kuuluvan levyn lukunopeus kertaa levyjen määrä, mutta kirjoitusnopeus on jonkin verran heikompi kuin yksittäisellä levyllä, koska sama kirjoitusoperaatio joudutaan tekemään kaikille levyryhmään kuuluville levyille. Vikasietoisuuden kannalta tämä on kestävin levyryhmä. Niin kauan kuin ryhmässä on yksikin toimiva levy, ryhmä on käytettävissä. Kapasiteetin kannalta tämä on kuitenkin heikoin ryhmä, levyryhmän koko on vain yhden ryhmään kuuluvan levyn koko, huolimatta sitä, kuinka monta levyä ryhmään kuuluu. [13.]

RAID-3, lomitettu levyryhmä ei kiertävällä pariteetilla, optimoitu pitkille yhden säikeen siirroille. RAID-3 levyryhmässä on aina nimetty pariteettikiintolevy, jolle muille levyryhmään kuuluville kiintolevyille lomitetusta tiedoista saatu pariteetti arvo tallennetaan. Suorituskyvyllisesti tämä levyryhmä ei ole kovin hyvä, koska tässä levyryhmässä kirjoitus- ja lukuoperaatiot suoritetaan aina koko lomitetulle tiedolle. Tästä johtuen, levyryhmä on optimoitu pitkille perättäin kirjoitetuille tietolohkoille, kuten esimerkiksi varmuuskopiointi tiedolle. Vikasietoisuuden kannalta tämä on erittäin vikasietoinen levyryhmä. Mikä tahansa levy voidaan menettää levyryhmästä, mutta sillä ollut tieto pystytään palauttamaan ryhmän muilta levyiltä. Kapasiteetin käytön kannalta tämä on myös yksi parhaista levyryhmistä, ryhmän kokonaiskapasiteetti on ryhmän yksittäisen levyn kapasiteetti kertaa levyjen lukumäärä miinus yhden levyn kapasiteetti. [13.]

RAID-4, lomitettu levyryhmä ei kiertävällä pariteetilla, optimoitu lyhyille usean säikeen siirroille, levyryhmä on toiminnaltaan ja ominaisuuksiltaan pääsääntöisesti sama kuin RAID-3 levyryhmä. Erona on, että toisin kuin RAID-3:ssa, jossa luettiin tai kirjoitettiin aina koko lomituksen sisältämä tieto, RAID-4:ssa luku- tai kirjoitusoperaatio kohdistetaan yksittäisen kiintolevyn sisältämään tietoon. Tästä syystä levyryhmä tarjoaa paremman lukusuorituskapasiteetin kuin RAID-3. [13.]

RAID-5, lomitettu levyryhmä kiertävällä pariteetilla on yksi yleisimmin käytetyistä RAID-levyryhmistä. Kirjoitusoperaatiossa tieto kirjoitetaan aina yhdelle ryhmän levyistä. Kirjoitusoperaation yhteydessä kirjoitetusta tietolohkosta ja muilla ryhmän fyysisillä levyillä olevista vastaavista lohkoista lasketaan XOR-operaatiolla pariteettiarvo, joka tallennetaan kulloinkin pariteettilevyn roolia suorittavan fyysisen levyn vastaavaan lohkoon. Pariteettilevyn rooli on kiertävä siten, että samassa kohtaa eri levylautasilla sijaitsevien lohkojen pariteettilevyn roolia hoitaa eri levy levyryhmässä. Tämä aiheuttaa sen, että kirjoitusoperaatioissa ryhmän suorituskyky ei ole yhtä hyvä kuin yksittäisen levyn, koska joka operaatiossa on kirjoitettava uusi tieto, luettava muiden samaan lomitukseen kuuluvien lohkojen tiedot, suoritettava pariteetti lasku ja kirjoitettava se lomituksen pariteettilohkoon. Lukuoperaatioiden suorituskyky on taas yksittäistä levyä parempi, koska lukuoperaatiot voidaan tehdä yhtäaikaisesti kaikille ryhmään kuuluville levyille. Vikasietoisuuden kannalta tämä on erittäin vikasietoinen levyryhmä. Mikä tahansa levy voidaan menettää levyryhmästä, mutta sillä ollut tieto pystytään palauttamaan ryhmän muilta levyiltä. Kapasiteetin käytön kannalta tämä on myös yksi parhaista levyryhmistä, ryhmän kokonaiskapasiteetti on ryhmän yksittäisen levyn kapasiteetti kertaa levyjen lukumäärä miinus yhden levyn kapasiteetti. [13.]

RAID-6, lomitettu levyryhmä kahdella kiertävällä pariteetilla. Toiminnallisuudeltaan lähes sama kuin RAID-5. Erotuksena on, että pariteettitieto tallennetaan aina kahdelle pariteettilevylle. Ryhmän vikasietoisuus on paras mahdollinen, koska levyryhmä kestää kahden levyn menetyksen. Kirjoitussuorituskapasiteetti on kuitenkin heikompi kuin edellä mainituilla levyryhmillä. Tämä johtuu siitä, että jokaista kirjoitusoperaatiota varteen tarvitaan kaksi pariteetin lukuoperaatiota, tiedon luku ryhmän muilta kuin kirjoitettavalta fyysisiltä levyiltä, kaksi pariteettilaskentaan ja lopuksi kirjoitettavan tiedon kirjoitus ja molempien pariteettitietojen kirjoitus pariteettilevyille. Levyryhmän kokonaiskapasiteetti on ryhmän yksittäisen levyn kapasiteetti kertaa levyjen lukumäärä miinus kahden levyn kapasiteetti. [13.]

3.3 Välimuisti (Cache)

Nykyaikaisissa tallennusjärjestelmissä on useimmiten erillinen välimuisti, joka sijaitsee isäntäkoneet liittävien edustaporttien ohjainten (Front-End controllers) ja taustaporttien ohjainten (Back-End controllers), joihin on liitetty fyysiset kiintolevyt, välissä. Välimuisti on puolijohdemuistipiireillä toteutettu välittäjäkerros, jonka tarkoituksena on erottaa tallennusjärjestelmää käyttävät isäntäkoneet kiintolevyjen viiveistä. Yleisesti välimuisti on kahdennettu vikasietoisuuden parantamiseksi.

Välimuisti koostuu yleisesti kahdesta erillisestä muistialueesta, tietomuistista (data store) ja tunnistemuistista (Tag RAM). Tietomuistille varattu alue välimuistista on jaettu sivuihin, joka on myös pienin mahdollinen kerrallaan osoitettava muistialue. Välimuistin sivut ovat kooltaan dynaamisia, sivun koko määritellään kulloinkin muistia käyttävän sovelluksen I/O:n koon mukaan. Tunnistemuisti sisältää tiedot siitä, missä osassa välimuistia kukin tieto sijaitsee tietomuistissa, mikä on tai tulee olemaan sen sijainti fyysisillä levyillä sekä tiedon siitä onko tieto kirjoitettu jo levyille.

Kirjoitusoperaatioissa isäntäkoneen lähettämä tieto yleensä tallennetaan ensin välimuistiin ja tallennusjärjestelmä tiedottaa isäntäkoneelle, että kirjoitusoperaatio on suoritettu onnistuneesti. Tämän jälkeen tieto kirjoitetaan tausta-ajona välimuistista kiintolevyille. Joissakin tapauksissa, kuten suurikokoisten kirjoitus-I/O:den ollessa kyseessä, tieto kirjoitetaan välimuistin ylitse suoraan kiintolevyille.

Lukuoperaatioissa levyjärjestelmä tarkistaa ensin tunnistemuistista, onko haluttu tieto mahdollisesti saatavilla suoraan tietomuistista. Mikäli näin on, palvellaan isäntäkoneen lukuoperaatio suoraan tietomuistista. Mikäli tietoa ei löydy tallennusjärjestelmän tietomuistista, haluttu tieto haetaan tietomuistiin, päivitetään tunnistemuisti ja haluttu tieto toimitetaan isäntäkoneelle. Tallennusjärjestelmien mikrokoodiohjelmat sisältävät yleensä hyvin tehokkaita algoritmeja, joilla pyritään edellisen lukuoperaation perusteella tunnistamaan se tieto, mihin isäntäkone tulee seuraavaksi kohdistamaan lukuoperaation. Tämä tieto haetaan ennakoivana toimenpiteenä tallennusjärjestelmän tietomuistiin, jotta isäntäkoneen seuraavaa lukuoperaatiota voidaan palvella suoraan välimuistista. [17, s. 72–77.]

3.4 Yhteysprotokollat

Yhteysprotokollat (interface connection protocols) toteuttavat kommunikaation isäntäsovittimen (host adpater) ja tallennusmedian (storage) välillä. Protokollat toteutetaan käyttämällä rajapintakomponenttia (Interface device) tai -ohjainta (Interface Controller) niin lähde- kuin kohdepäässä kommunikaatiopolkua. Nykyisellään yleisimmin käytettävät liittymäprotokollat käsitellään seuraavassa.

Sarjaliikenteinen ATA (SATA). Virallisesti protokolla on nimeltään ATA/ATAPI-7 Volume 3, mutta se tunnetaan paremmin nimellä SATA (Serial Advanced Technology Attachment). Se on sarjaliikenteinen IDE/ATA standardiperheen uusin protokolla. Yleisimmin SATA-liitäntäprotokolla on pääasiassa käytössä kotikäyttäjille tarkoitetuissa tallennusjärjestelmissä, korvaten vanhempia IDE- (Integrated Drive Electronics) ja PATA (Parallel Advanced Technology Attachment)-protokollia käyttävät laitteet. Viime aikoina on SATA-liitäntäprotokollaa käyttäviä laitteita tullut käyttöön myös alhaisemman suorituskyvyn ammattikäyttöön tarkoitetuissa tallennusjärjestelmissä. Uusin SATA-versio 3.2 (SATA Express) tukee 16 Gb/s liitäntänopeutta. [14.]

Rinnakkainen SCSI. Vuonna 1986 standardoidusta SCSI-standardiperheestä on muodostunut yleisin käytössä oleva liitäntästandardien (Connection Standards) kokoelma, jota käytetään tiedonsiirtoon korkean suorituskyvyn liitäntälaitteiden välillä. SCSI-protokollat ovat oheislaitteille tarkoitettuja rinnakkaistiedonsiirtoon perustuvia, älykkäitä, puskuroituja vertaisliittymäprotokollia (peer to peer), joilla piilotetaan liitettävien laitteiden fyysisten ominaisuuksien monimutkaisuudet isäntäkoneelta. Tämä on saavutettu siten, että jokainen SCSI-laite kiinnittyy SCSI-väylään samalla tavalla. SCSI-standardien, joita on tällä hetkellä kolme SCSI-1, SCSI-2 ja SCSI-3, kehityksestä vastaa T10-komitea, mutta SCSI:in liittyvien termistöjen kehityksestä vastaa taas SCSITA (Small Computer System Interface Trade Association). Kaikki SCSI-standardit ovat modulaarisia, määritellen useita ominaisuuksia, joista saman standardin sisällä voidaan tehdä erilaisia toteutuksia.

Nykystandardien mukaan samassa väylässä voi olla jopa 16 laitetta ja suurin tuettu tiedonsiirtonopeus on SCSI-3-standardin Ultra-640-protokolla, jolla tiedonsiirtonopeus on 640 MB/s. SCSI-3-standardin protokollat, ovat tällä hetkellä yleisimmin käytössä olevat protokollat, joilla toteutetaan tiedonsiirto isäntäkoneiden ja tallennusjärjestelmien välillä. [15 & 16.]

Sarjaliikenteinen SCSI (SAS) on SCSI standardikokoelman uusin standardi, joka määrittelee sarjaliikenteisen SCSI-protokollan. SAS on suunniteltu korvaamaan vanhempi rinnakkaisväyläinen SCSI, joten se on suorituskyvyltään ja skaalautuvuudeltaan parempi kuin edeltäjänsä. SAS-kehityksestä vastaavat samat organisaatiot kuin muidenkin SCSI-standardien kehityksestä. SAS-standardi määrittelee kaksipisteyhteyksinen sarjaliikenteisen protokollan, jonka uusimmalla versiolla 2.0 päästään jopa 6 Gb/s tiedonsiirtonopeuteen. Tosin stadardin kehittyksestä vastaavilla organisaatioilla on jo olemassa suunnitelmat jopa 24 Gb/s väylänopeuksista.

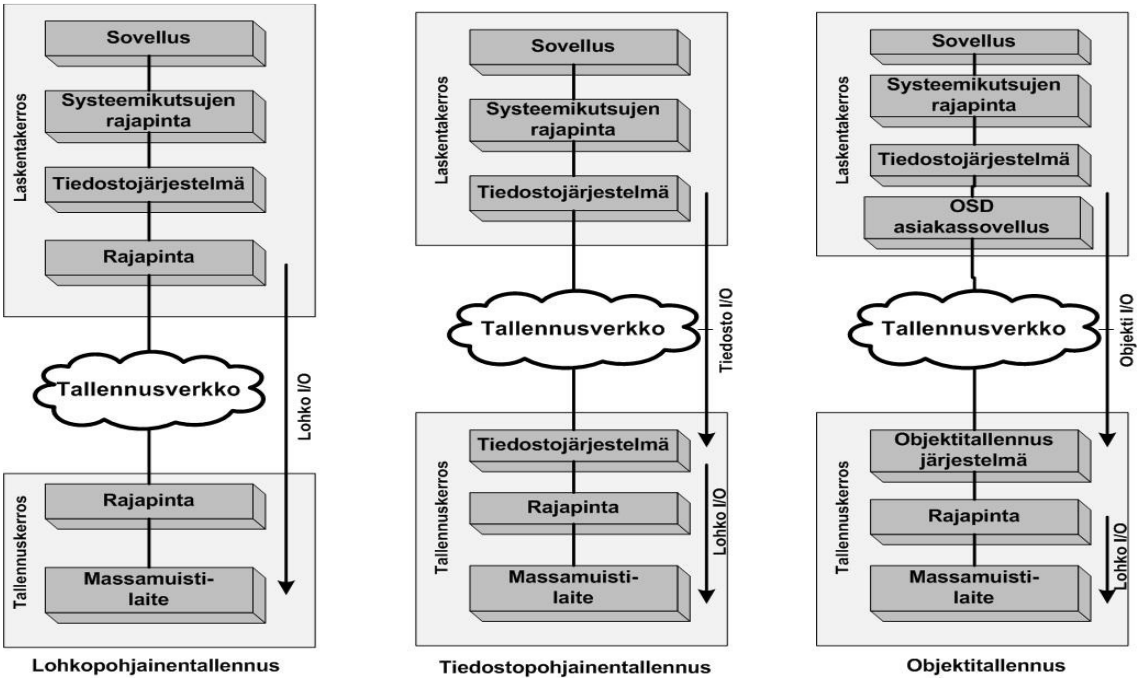
Varsinkin suorituskyvyltään keskitasoisissa tallennusjärjestelmissä on tiedonsiirto taustaporttien ja kiintolevyjen välillä, usein toteutettu SAS-protokollalla. Tämän protokollan etuina on muunmuassa yhteensopivuus SATA-levyjen kanssa. 3 Gb/s tai 6 Gb/s väylänopeuksisia SATA-levyjä on mahdollista liittää SAS-takalevyyn. [15 & 16.]

4 Tiedostojärjestelmät ja tiedonsiirto tallennusverkoissa

Tallennusverkot (Storage Area Networks) mahdollistavat tiedonsiirron tallennusjärjestelmien ja niitä käyttävien isäntäkoneiden välillä. Käsitteenä tallennusverkko on ehkä jossain määrin harhaanjohtava. Tämä siitä syystä, että tallennusverkkon looginen kokonaisuus ei pelkästään koostu tallennusjärjestelmistä, niitä käyttävistä koneista, yhteydet mahdollistavista laitteista ja kaapeloinneista, vaan myös käyttö- ja tiedostojärjestelmien voidaan katsoa kuuluvan osaksi tallennusverkon loogista kokonaisuutta. Tallennusverkon olemassaolo ei välttämättä vaadi minkään verkkokomponentin tai -väylän olemassaoloa, vaan kyseessä voi olla suoraan kytketty toteutus, jolloin käytettävät massamuistilaitteet ovat isäntäkoneen sisällä. Se, miten tietoa siirretään tallennusverkon ylitse, on yleisesti jaoteltu kolmeen eri tiedonsiirto tyyppiin, jotka ovat

- lohkopohjainen tallennus (Block Storage)
- tiedostopohjainen tallennus (File Transfer)
- objektipohjainen tallennus (Object Storage)

Edellä mainitut tyypit määrittävät sen, missä kohtaa tallennusverkkon loogista arkkitehtuuria tiedostojärjestelmä sijaitsee, sekä se millä protokollilla tietoa siirretään tallennusverkossa ja mitä tallennusverkkokerroksen tiedonsiirtoprotokollaa käytettään. Tiedostojärjestelmän ja muiden tiedonsiirtoon liittyvien komponenttien sijainti tallennusverkon loogisessa arkkitehtuurissa on esitelty kuvassa 3.



Kuva 3. Tiedonsiirto tallennusverkossa.

Seuraavissa kappaleissa käsitellään, miten tietoa siirretään tallennusverkoissa ja miten tallennusjärjestelmät on silloin liitetty isäntäkoneeseen.

4.1 Lohkopohjainen tallennus

Lohkopohjaista tallennusta käyttävissä toteutuksissa tiedonsiirto tallennusverkossa isäntäkoneen ja tallennusjärjestelmän välillä tapahtuu tietopaketteina, jotka sisältävät yhden tai useampia LBA-osoitettuja massamuistilaitteen tietolohkoa (Block). Tässä syystä tätä tiedonsiirtometodia kutsutaan lohkopohjaiseksi tallennukseksi. Tiedostojärjestelmä (File system), jolla massamuistilaitteiden tekninen rakenne ja tiedon fyysinen sijainti massamuistilaitteilla on piilotettu käyttäjältä, sijaitsee isäntäkoneella. Lisäksi tiedostojärjestelmään liittyvä metatieto sijaitsee lohkopohjaista tallennusta käyttävissä järjestelmissä isäntäkoneen yhteydessä. Kommunikaatio isäntäkoneen väylä- (host bus

adapter) tai verkkosovittimen (network interface card) ja tallennusmedian tai tallennusjärjestelmän väylä- tai verkkosovittimen välillä tapahtuu hyödyntäen aikaisemmin esiteltyjä liittymäprotokollia, joiden tietopaketit paketoidaan tallennusverkon kuljetusprotokollan (transfer protocol) määrittämään tietopakettiin. Yleisimmät lohkopohjaisen tallennuksen tallennusverkon toteutukset ovat suorakytkentäinen, kuituväylätallennusverkko sekä iSCSI, jotka esitellään seuraavissa kappaleissa.

4.1.1 Suorakytkentäinen tallennusverkkoarkkitehtuuri (DAS)

Suorakytkentäinen (Direct Attached Storage) tallennusverkkoarkkitehtuuri on ehkä suurimman määrän erilaisia toteutusvaihtoehtoja tarjoava arkkitehtuuri. Tähän ryhmään kuuluvat niin yksittäinen isäntäkoneen sisällä olevan massamuistin laitteen toteutus kuin myös vaihtoehto, jossa massamuistit ovat suuressa monia isäntäkoneita palvelevassa levyjärjestelmässä ja kaikenlaiset variaatiot siltä väliltä. Myös kuljetuskerroksen protokollina voidaan DAS-toteutuksessa käyttää kaikkia myöhemmin esiteltäviä NAS- (Network Attached Storage) ja SAN-tallennusverkkoarkkitehtuurien protokollia. Joten raja, mikä on DAS-, NAS- tai SAN-toteutus, on erittäin epäselvä. Yleisesti erottavana tekijänä pidetään sitä, että isäntäkone ja massamuistilaite tai levyjärjestelmä on liitetty toisiinsa suorakaapeloinnilla ja näiden välillä ei ole kuljetuskerroksen (transport layer) laitteita kuten esimerkiksi kytkimiä (switch). Suorakytkentäisen tallennusverkon toteutukset jaotellaan sisäisesti ja ulkoisesti kytkettyihin suorakytkentäisiin vaihtoehtoihin. [18, s. 41.]

Sisäinen suorakytkentä

Sisäisessä suoraan kytketyssä tallennusverkkovaihtoehdossa massamuistilaitteet ovat isäntäkoneen sisällä, ja ne on kytketty joko sarja- tai rinnakkaisväylällä isäntäkoneen ohjainkorttiin. Tiedostojärjestelmä sijaitsee tässä arkkitehtuurissa aina isäntäkoneessa ja tieto siirretään tallennusverkossa lohkotasolla. Yleisimmät väyläprotokollat ovat SATA ja SCSI.

Etuna tässä toteutuksessa ovat pienet viiveet tallennusverkossa koska kaapeli pituudet ovat erittäin lyhyitä eikä tallennusverkko ei sisällä mitään kuljetuskerroksen laitteita tai sen protokollia, joiden toteutus aiheuttaisi viivettä. Yleensä tallennuskapasiteetit, joita tällä arkkitehtuurilla on mahdollista saavuttaa, ovat korkeintaan teratavu luokkaa. Tämä johtuu siitä, että massamuistien määrä mitä isäntäkoneeseen voidaan sijoittaa, on ra-

joitettu. Massamuistilaitteista on usein muodostettu RAID-1 tai RAID-5-ryhmiä, jolloin niillä oleva tieto on saatu suojattua laiterikoilta. [18, s. 41.]

Ulkoinen suorakytkentä

Ulkoinen suorakytkentä-tallennusverkkoarkkitehtuurissa isäntäkoneen väyläliitin on kytketty massamuistilaitteen tai tallennusjärjestelmän väyläliittimeen suorakytkentäkaapelilla. Useimmiten tiedostojärjestelmä sijaitsee isäntäkoneessa, mutta myös vaihtoehto, jossa tiedostojärjestelmä sijaitsee tallennusjärjestelmässä, on mahdollinen. Tietoa siirretään tallennusverkossa joko lohko- tai tiedostotasolla. Lohkotasolla tietoa siirrettäessä, fyysisen kerroksen protokollina toimivat yleisimmin joko SCSI tai SAS. Tiedostotasolla tietoa siirrettäessä NFS (Network File System) tai CIFS (Common Internet File System) ovat yleisimmät sovelluskerroksen (application layer) protokollat. Kuljetuskerroksen protokollina toimivat taas joko FC-, SAS- tai TCP-protokolla, riippuen siitä, siirretäänkö tietoa lohko- vai tiedostotasolla. Tallennuskapasiteetit tässä vaihtoehdossa voivat olla jopa petatavu luokkaa ja on erittäin yleistä, että massamuistilaitteet ovat tallennusjärjestelmissä konfiguroitu RAID-ryhmiksi. Etuina tässä toteutuksessa ovat pienet viiveet tallennusverkossa, koska kaapelipituudet ovat yleensä lyhyitä, tallennusverkossa ei ole viiveitä aiheuttavia kuljetuskerroksen laitteita sekä, suorituskyky koska tallennusjärjestelmän koko suorituskyky on vain kohtuullisen pienen määrän isäntäkoneita käytössä. [18, s. 41.]

4.1.2 Kuituväylätallennusverkko (FC SAN)

Kuituväylätallennusverkko (Fibre Channel SAN) on suurinopeuksinen tallennusjärjestelmien ja isäntäkoneiden väliseen tiedonsiirtoon tarkoitettu verkko, jossa tietoa siirretään lohkotasolla. Yleisesti fyysisen kerroksen kaapelointi on kuituväylätallennusverkossa toteutettu joko yksi- tai monimuotovalokuiduilla, mutta myös kuparikaapeloinnilla toteutettu vaihto on mahdollinen. Kuituväylätallennusverkossa voi olla yksi tai useampia siirtokerroksen kytkimiä tai keskittimiä, jotka mahdollistavat verkon joustavan laajennettavuuden ja sen, että yksittäisen verkon laitteen tietopolkuja voidaan muuttaa helposti ilman kaapelointi muutoksia. Kuljetusprotokollana kuituväylätallennusverkossa toimii vuonna 1994 standardoitu FCP (Fibre Channel Protocol)-protokolla, joka kehityksestä vastaavat T11-tekninen lautakunta, INCITS (InterNational Committee for Information Technology Standards) ja ANSI (American National Standards Institute). Tiedon-

siirto kuituväylätallennusverkossa tapahtuu lohkotasolla, koska käytettävä protokolla on yleisimmin joku SCSI-protokolla perheeseen kuuluva protokolla.

Kuituväylätallennusverkon etuina ovat suorituskyky, laajennettavuus ja suuret laitteiden väliset etäisyydet. Suorituskapasiteetti kuitutallennusverkkoarkkitehtuurissa on jopa 16 Gb/s ja verkon osoiteavaruus mahdollistaa jopa 15 miljoonan laitteen liittämisen verkkoon. 1 550 nm:n aallonpituudella toimivalla yksimuotovalokuidulla, on mahdollista siirtää tietoa jopa 50 km:n päähän ilman välivahvistimia.

Kuituväylätallennusverkossa käytettävän FCP-protokollan protokollapinossa on viisi kerrosta:

- FC-4 on protokollien liitoskerros. Se määrittää sovelluskerroksen liittymät ja miten ylemmän kerroksen protokollat kuten esimerkiksi SCSI, IP (Internet Protocol), HIPPI (High Performance Parallel Interface), ATM (Asynchronous Transfer Mode) on liitetty alempiin kerroksiin.

- FC-3 eli palvelukerros ei ole käytössä tällä hetkellä, varattu tulevia laajennuksia varten.

- FC-2 eli verkkokerros vastaa verkon osoitteistuksesta sekä tiedon organisoinnista (kehykset, järjestys ja tiedonvaihto). Lisäksi vastaa verkon palveluista, palvelu luokista, vuo-ohjauksesta ja reitityksestä.

- FC-1 eli koodauskerros määrittää miten tieto koodataan ennen lähetystä ja miten se puretaan vastaanotossa. Koodaus tapahtuu muuntamalla 8 bittinen merkki 10 bittiseksi lähetysmerkiksi.

- FC-0 eli fyysinen kerros määrittää fyysiset liitännät, kuljetusmediat, bittien siirron.

Kuituväylätallennusverkon laitteiden osoitus tapahtuu dynaamisesti laitteen portin liittyessä verkkoon. Tällöin laitteen portille määritetään 24-bittinen osoite, joka koostuu kolmesta osasta, verkkoalueesta, alueesta ja portista. Osoitteen kahdeksan ensimmäistä bittiä määrittävät verkkoalueen, seuraavat kahdeksan alueen ja viimeiset portin. Jokainen kuituväyläverkkoon kuuluva kytkin on oma verkkoalueensa ja kytkimen por-

teista muodostetaan alueita yleensä sen perusteella, mihin fyysiseen korttiin portit kuuluvat. Lisäksi kullakin verkon laitteella on omat kiinteät 64-bittiset WWN (World Wide Name) -osoitteet, jotka vastaavat IP-verkon MAC (Media Access Control)-osoitetta. Kullakin laitteella on useita WWN-osoitteita, jotka ovat portti WWN (World Wide Port Name) eli WWPN ja noodi WWN eli WWNN (World Wide Node Name).

Kuituväylätallennusverkkoarkkitehtuurissa verkkoon kuuluvat laitteet voidaan liittää kolmella erityyppisellä topologialla:

- FC-P2P, kaksipisteliitäntä (point-to-point). Kaksi laitetta on suoraan liitetty toisiinsa, eli kyseessä on aikaisemmin esitelty DAS.

- FC-AL, vuorovälitteinen silmukka (arbitrated loop). Tässä arkkitehtuurissa laitteet muodostavat renkaan. Laitteet lähettävät vuorotellen tietoa verkossa. Skaalautuvuus ei ole kovinkaan hyvä, mikäli uusia laitteita lisätään silmukkaan, silmukka joutuu katkaisemaan tiedonsiirron uuden laitteen liitoksen ajaksi.

- FC-SW, kytketty kudos (switched fabric). Laitteet on kytketty toisiinsa kytkimillä ja jokaisen laitteen yhteyksille tarjotaan omistettu tietopolku (data path) verkon läpi. Topologia on erittäin hyvin skaalautuva ja tarjoaa myös parhaan suorituskapasiteetin laitetta kohden. Kytketty kudos on yleisin käytössä oleva vaihtoehto kuituväylätallennusverkkojen toteutuksissa.

Kuituväylätallennusverkon kytkimien muodostama kudos (fabric) tarjoaa aluejakotoiminnallisuuden (zoning), jossa eri laitteiden porteista muodostetaan looginen ryhmä, joiden sisällä portit voivat kommunikoida keskenään. Aluejaot kategorisoidaan yleensä kolmeen tyyppiin:

- Porttialuejakoihin (Port zoning). Aluejaot määritellään käyttämällä siihen kuuluvien porttien tunnistetta eli verkkoalueen ja portin tunnistetietoa. Haittapuolena tässä on se, että mikäli aluejakoon kuuluva laite siirretään toiseen kytkimeen, on aluejako konfiguroitava uudelleen. Toisaalta tämäntyyppisten aluejakojen etuna on se, että mikäli aluejakoon kuuluva isäntäkoneen tai tallennuslaitteen portti hajoaa, voidaan komponentti vaihtaa tekemättä muutoksia aluejakoon.

- WWN-aluejakoihin (WWN zoning). Aluejako määritellään käyttämällä siihen kuuluvien laitteiden WWN-osoitteita. Tämän aluejaon etuina on helppo siirrettävyys kytkinportista toiseen. Haittapuolena on se, että jos aluejakoon kuuluva isäntäkoneen- tai tallennuslaitteenportti hajoaa ja komponentti vaihdetaan, on aluejako konfiguroitava uudelleen.

- Sekoitettuihin aluejakoihin (mixed zoning). Tässä aluejakotyypissä on aluejako muodostettu käyttämällä porttien tunnisteita ja WWN-osoitteita. Se on harvemmin käytetty aluejakotyyppi, koska aluejaon konfiguraatiosta tulee helposti epäselvä.

Ei voida sanoa, että olisi mitään perusteita, minkä vuoksi aluejakoja luodessa pitäisi ehdottomasti käyttää jotain tiettyä edellä mainituista aluejakotyypeistä. Mutta yleisintä on, että ne luodaan ylläpidon selkeyden vuoksi joko portti- tai WWN-aluejakoina.

Kuituväylätallennusverkkoon liitetty isäntäkone on varustettu erillisellä kuituväyläverkkokortilla (Host Bus Adapter), jossa on verkkoon liitetyt portit ja joka toteuttaa FCP-protokollan vaatiman laskennan. Kuituväylätallennusverkkoon liitetty isäntäkone on mahdollista käynnistää kuituväyläverkon yli esitetyltä kiintolevyltä, jolloin isäntäkoneessa ei välttämättä tarvitse olla paikallisia kiintolevyjä ollenkaan. [19, s. 95 – 128]

4.1.3 IP-verkkotallennus

Vaikka kuituväylätallennusverkko onkin tehokas ja skaalautuva tapa siirtää tietoa lohkotasolla isäntäkoneiden ja tallennusjärjestelmien välillä, on sillä haittapuolensa. Kuituväylätallennusverkon toteutusta varten joudutaan hankkimaan omat kytkimet, kaapeloinnit ja isäntäkoneille verkkokortit. Nämä investoinnit aiheuttavat lisäkustannuksia. Koska useimmilla organisaatioilla on jo olemassa oleva IP-pohjainen tiedonsiirtoverkko, voidaan nämä lisäkustannukset välttää toteuttamalla lohkopohjaista tiedonsiirtoa toteuttava tallennusverkko tässä IP-tiedonsiirtoverkossa. Lisäksi hyödyntämällä IP-pohjaista verkkoa tallennusverkkoina on tiedonsiirto helposti toteutettavissa pitkienkin etäisyyksien ylitse. Yleisimmät protokollat, joilla tallennusverkko voidaan toteuttaa hyödyntäen IP-protokollaa kuljetusprotokollana, esitellään seuraavissa luvuissa.

4.1.4 iSCSI

iSCSI (internet SCSI) -protokollassa SCSI komennot ja tieto on kapseloitu IP-paketin sisälle ja tietopaketin kuljetus tapahtuu tietoverkossa TCP/IP:n (Transmission Control Protocol / Internet Protocol) avulla. iSCSI onkin erittäin yleinen ja edullinen tapa toteuttaa tiedonsiirto isäntäkoneiden ja iSCSI tukevien tallennusjärjestelmien välillä, koska useimmat nykyaikaiset käyttöjärjestelmät tukevat iSCSI-liitännäisiä tallennusjärjestelmiä suoraan ilman lisälaitteita. Mutta tämäkään toteutus ei ole täysin ongelmaton, sillä iSCSI-protokolla tietoyksikön eli PDU:n (Protocol Data Unit) luonti aiheuttaa ylimääräistä kuormaa isäntäkoneen prosessorille. Joten toteutuksissa joissa tietoa siirretään paljon, joudutaan isäntäkone varustamaan erillisellä iSCSI-verkkokortilla.

iSCSI-protokollassa tietoa siirretään käynnistäjäksi kutsutun isäntäkoneen ja kohteeksi kutsutun vastaanottajan, joka on yleensä tallennusjärjestelmä tai yhdyskäytävän välillä. Tiedonsiirtoverkkona voi olla joko LAN (Local Area Network), WAN (Wide Area Network) tai jopa internet. Mikäli tiedonsiirto tapahtuu isäntäkoneen ja iSCSI-liitännällä olevan tallennusjärjestelmän välillä, kyseessä on niin sanottu luonnollinen iSCSI-yhteys. Mikäli taas tiedonsiirto tapahtuu isäntäkoneen ja yhdyskäytävä laitteen, joka on liitetty kuituväyläverkkoon, puhutaan sillatusta (bridged) iSCSI-yhteydestä.

Jos isäntäkone on liitetty tietoverkkoon normaalilla verkkokortilla, käytetään protokollan toteuttamiseen ohjelmistopohjaista iSCSI-käynnistäjää isäntäkoneessa. Tämä on halvin ja helpoin tapa toteuttaa iSCSI-yhteys. Mutta haittapuolena on se, että verkkokortti

tarjoaa vain normaalit IP-toiminnallisuudet joten, TCP- ja iSCSI-pakettien kapselointi joudutaan toteuttaman isäntäkoneen prosessorilla joka lisää sen kuormaa. Saatavilla on kuitenkin TOE (TCP Offload Engine)-verkkokortteja jotka siirtävät TCP:n toteuttamisen vaatimat toiminnallisuudet isäntäkoneen prosessorilta verkkokortille, jolloin vain iSCSI:n toteutuksen vaatima laskenta jää isäntäkoneen prosessorin vastuulle. Parhaan suorituskyvyn takaamiseksi, on iSCSI-toteutus tehtävä laitteistopohjaisena toteutuksena, jolloin isäntäkone varustetaan iSCSI HBA:lla (Host Bus Adapter). iSCSI HBA on verkkokortti, joka kykenee toteuttamaan sekä iSCSI:n että TCP/IP:n vaatimat laskennat. Lisäetuna iSCSI HBA:n käytössä on se, että isäntäkone on mahdollista käynnistää tallennusverkon yli tarjotulta kiintolevyltä, jolloin isäntäkoneessa ei välttämättä tarvitse olla paikallisia kiintolevyjä ollenkaan.

OSI-mallin (Open Systems Interconnection Reference Model) mukaisessa iSCSI-protokollapinossa iSCSI on istuntokerroksen (Session Layer) protokolla, joka vastaa siitä, että luotettava tiedonsiirtoyhteys muodostetaan käynnistäjien (Initiator) ja kohteiden (Target) välille. Protokollan vastuulle kuuluu tällöin verkkoon kirjautumiset, todennus, kohteiden etsintä ja sessioiden hallinta. Istuntokerroksen paketteja joilla käynnistäjät ja kohteet kommunikoivat, kutsutaan protokolladatayksiköksi (PDU). Tähän kommunikointiin kuuluvat mm. iSCSI-yhteyksien avaus sekä lopetus, sessioiden luonti, SCSI-komentojen ja tiedon lähetys sekä vastaanotto ja SCSI-statusten vastaanotto. [20.]

Jotta käynnistäjä voi aloittaa tiedonsiirron kohteen kanssa, on sen ensin löydettävä kohde tietoverkosta, että sessio voidaan muodostaa. Tätä istuntokerroksen toiminnallisuutta kutsutaan hakupalveluksi, ja se voi tapahtua kahdella tavalla:

- Lähetä kohteet-haku (SendTargets discovery). Tässä vaihtoehdossa käynnistäjälle on konfiguroitu kohteen palveluportit, joihin muodostaa sessio. Käynnistäjä lähettää SendTargets-komennon, johon kohteen palveluportti vastaa saatavilla olevien kohteiden nimillä ja osoitteilla.

- Internet-tallennusnimipalvelu (Internet Storage Name Service). iSNS mahdollistaa käynnistäjän saatavilla olevien kohteiden automaattisen löytämisen. Sekä käynnistäjät että kohteet ovat konfiguroitu siten, että ne rekisteröivät itsensä automaattisesti iSNS-palveluun. Kun käynnistäjä haluaa ottaa yhteyden käytössä oleviin kohteisiinsa, se lähettää kyselyn palvelulle, joka palauttaa listan saatavilla olevista kohteista.

Jotta iSCSI-käynnistäjät ja -kohteet voivat kommunikoida keskenään tietoverkossa, on kummallakin oltava ainutkertainen nimi jolla resurssit voidaan tunnistaa. Tätä nimeä kutsutaan iSCSI-nimeksi. Nimet muodostuvat numeroista, isoista ja pienistä kirjaimista sekä hyväksytyistä erikoismerkeistä, jotka ovat piste, väliviiva, välilyönti ja kaksoispiste. Yleisimmät iSCSI-nimeämistyypit ovat seuraavat:

- Muodollinen iSCSI-nimi IQN (iSCSI Qualified Name). Muodollinen iSCSI-nimi koostuu verkkoaluenimestä muodostuvasta osasta ja käyttäjän määrittelemästä osasta, joka voi olla esim laitteen nimi. Muodollisia iSCSI-nimiä käyttävillä organisaatiolla täytyy olla käytössään rekisteröity verkkoaluenimi. Mutta sen ei tarvitse olla aktiivinen tai olla ratkaistavissa osoitteeksi. Sen täytyy olla vain varattu, jotta ei synny sekaannuksia muiden organisaatioiden kanssa. Muodollinen iSCSI-nimi voi olla muotoa *iqn.2013-10.fi.malli:tallennujärjestelmä1*

- Laajennettu ainutkertainen tunniste EUI (Extended Unique Identifer) on globaali ainutkertainen tunniste, joka perustuu IEEE (Institute of Electrical and Electronics Engineers) EUI-64-nimistandardiin. Nimi koostuu etuliitteestä, jota seuraa 16 merkin heksadesimaaliarvossa oleva nimi, esimerkiksi *eui.0300732A32598D26*.

4.1.5 IP-välitetty kuituväylä (FCIP)

Vaikka tässä insinöörityössä käytettävä laitteisto sijaitsikin ainoastaan yhdessä tietokonekeskuksessa, useimmilla etäresurssipalveluja tarjoavilla organisaatioilla on tarve siirtää tietoa luotettavasti tallennusverkossa useampien, jopa eri puolilla maailmaa sijaitsevien tietokonekeskusten välillä. Tällöin kuitenkin etäisyydet asettavat rajoitteita kuituväylätallennusverkon käytölle. Kuituväylätallennusverkkoa käyttämällä siirtoetäisyydet ovat yleensä suurimmillaan joidenkin kymmenien kilometrien luokkaa. Jotta tiedonsiirto kaukana toisistaan sijaitsevien kuituväyläverkkojen välillä voitaisiin toteuttaa, on käytettävä protokollaa, jolla tietolohkoja voidaan siirtää IP-verkkojen kautta. FCIP (Fibre Channel over IP) on yleisimmin käytetty tunnelointiprotokolla (tunneling protocol), jolla kuituväyläverkon pakettien siirto IP verkoissa voidaan toteuttaa. Vaikka tässä insinöörityössä ei tällaista globaalia hajautusta toteutettukaan, käydään FCIP-protokolla lävitse, koska yritysratkaisuissa se on useimmiten osa toteutusta.

FCIP on mekanismi, jolla voidaan tunneloida kaksi tai useampia toisistaan erillään olevia kuituväyläverkkoja IP-verkon ylitse siten, että eri verkoissa olevat laitteet voivat keskustella keskenään kuvassa 4 esitetyn FCP-protokollan avulla. FCIP-topologiassa kuituväyläverkot on yhdistetty yhdyskäytävälaitteella IP-verkkoon, kuten esimerkiksi Internetiin. Yhdyskäytävälaiteet toteuttavat FCIP-protokollapinon, jossa kuituväyläverkon FCP-kehys paketoidaan IP-verkon pakettiin sekä tunneloinnin yhdyskäytävälaitteiden välille. [21.]

Kuva 4. FCIP-protokollapino.

4.2 Tiedostopohjainen tallennus

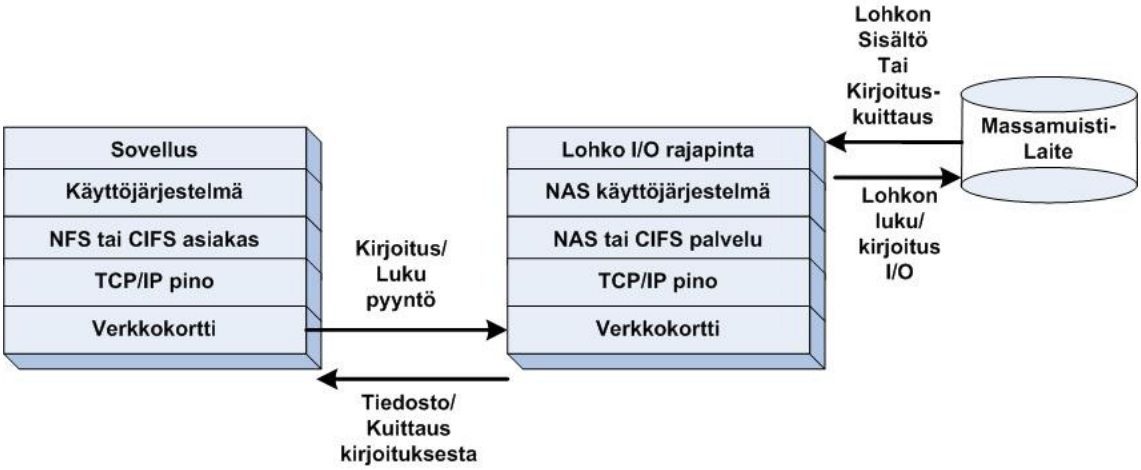
Tiedostopohjaisessa tallennuksessa, jota myös verkkotiedostojärjestelmäksi kutsutaan, tiedostojärjestelmä ja siihen liittyvä metatieto ei sijaitse isäntäkoneen yhteydessä, vaan tallennusverkossa olevassa tallennusjärjestelmässä. Tietoliikenne isäntäkoneen verkkosovittimen ja tallennusjärjestelmän verkkosovittimen välillä tapahtuu IP- sekä TCP-protokollia käyttäen. Tallennusjärjestelmä jakaa tiedostot käyttäen erilaisia tiedostonjakoprotokollia (file sharing protocol), jolloin isäntäkoneiden on mahdollista kohdistaa I/O-operaatioita tiedostoihin ja tiedostojärjestelmään.

4.2.1 Verkkoliitännäinen tallennusjärjestelmä (NAS)

NAS-toteutukset ovat korkean suorituskyvyn, tiedostotasolla tapahtuvaan tiedonsiirtoon tarkoitettuja ratkaisuja, jossa tiedonsiirto asiakaskoneen (client) ja NAS-laitteen välillä tapahtuu aina IP-verkossa. Koska tiedonsiirto asiakakoneen ja NAS-laitteen välillä tapahtuu tiedostotasolla, tiedostojärjestelmä sijaitsee NAS-laitteessa ja tiedonsiirto tapahtuu tiedostonjakoprotokollia käyttäen. NAS-ratkaisut on yleensä toteutettu joko NAS-tallennusjärjestelmällä tai NAS-yhdyskäytävällä. NAS:n yhtenä suurimpana etuna voidaan pitää suurta skaalautuvuutta, yksittäisistä NAS-laitteista voidaan helposti rakentaa suuria NAS-ryppäitä (cluster), jotka näkyvät yksittäisenä laitteena asiakaskoneille.

NAS-tallennusjärjestelmissä koostuu NAS-yksiköstä (NAS head), joka sisältää verkkoliitännät, käyttöjärjestelmän ja palvelut tiedostonjako protokollille, sekä sille omistetusta tallennusjärjestelmästä, jossa massamuistilaitteet sijaitsevat. NAS-yhdyskäytävä (NAS gateway) on käytännössä täysin vastaava toteutus, ainoastaan sillä erotuksella, että se hyödyntää jotain yleiskäyttöistä tallennusjärjestelmää, johon se liittyy kuituväyläverkolla.

NAS I/O-operaatiot tapahtuvat tarjoamalla asiakaskoneille pääsy tietoon tiedostotasolla, kuten kuvassa 5 on esitetty. Tiedosto I/O on korkeantason pyyntö, joka kuvaa tiedoston, jota halutaan käyttää, joko määrittelemällä sen nimen, sijainnin tai muun tiedostoa kuvaavan ominaisuuden. NAS-laitteen tiedostojärjestelmä pitää lukua millä massamuistilaitteella kukin tiedosto sijaitsee ja muuntaa asiakaskoneen pyynnön lohkotason I/O:ksi, joka välitetään joko SCSI- tai FCP-protokollilla tallennusjärjestelmälle.



Kuva 5. NAS I/O -operaation kulku

Kuten aikaisemmin todettiin, tiedonsiirto NAS-laitteiden ja asiakaslaitteiden välillä tapahtuu käyttämällä tiedostonjakoprotokollia, näistä yleisimmät protokollat ovat NFS, SMB/CIFS ja AFS. Seuraavissa luvuissa esitellään näistä yritysympäristöissä eniten käytössä olevat NFS- ja CIFS-protokollat.

4.2.2 NFS

NFS on OSI-mallin sovelluskerroksessa toimiva, UNIX-käyttöjärjestelmille tarkoitettu hajautettu asiakas-palvelintiedostonjakoprotokolla (distributed client-server file system protocol). Sun Microsystems vuonna kehitti vuonna 1989 ensimmäisen version NFS-protokollasta, mutta nykyisin on käytössä viimeisin kehitysversio NFSv4.1, joka kehitettiin IETF:n (Internet Engineering Task Forec) johdolla vuonna 2010. Alkujaan NFS perustui yhteydettömän UDP (User Datagram Protolcol)-kuljetuskerroksen protokollan käyttöön, mutta NFSv3-julkaisun yhteydessä vuonna 1995 Sun Microsystems lisäsi mukaan tuen TCP-protokollan käyttöä varten.

Protokolla käyttää laite riippumatonta mallia jolla tieto esitellään, sekä RPC (Remote Procedure Call)-kutsuja, jotka mahdollistavat eri prosessien kommunikointi laitteiden välillä. RPC-kutsuilla mahdollistetaan se, että etätiedostojärjestelmää voidaan käsitellä ja suorittaa sille seuraavat toiminnot,

- tiedostojen ja hakemistojen etsintä
- tiedostoon kirjoitus
- tiedoston luku
- tiedoston sulkeminen
- tiedoston attribuuttien muuttaminen
- tiedosto ja hakemisto linkkien muokkaus.

Kuten aikaisemmin todettiinkin, NFS oli alun perin tilaton protokolla, mikä tarkoittaa, että se ei pidä kirjaa avoimista tiedostoista ja osoittimista. Niinpä jokaisen kutsun täytyi tarjota kaikki mahdolliset argumentit, jotta se saattoi käsitellä tiedostoja palvelimilla. Lisäksi UDP-pakettien lähetystä tai järjestystä ei varmistettu siirtotien päätepisteiden välillä. Nämä puutteet aiheuttivat monesti ongelmia, varsinkin Internetissä yli tapahtuvassa tiedonsiirrossa. Uudemmissa NFS-versiossa nämä ongelmat ovat poistuneet TCP-yhteyksien käyttöön oton vuoksi ja NFS-toteutukseen lisätyn NFS-palvelimelle säilytettävän tilallisuus tiedon tuen vuoksi. [22 & 23.]

4.2.3 CIFS

CIFS (Common Internet Filee System) on tilallinen sovelluskerroksen asiakas-palvelinprotokolla. CIFS on Microsoft Windows käyttöjärjestelmien SMB (Server Message Block) -tiedostonjakoprotokollan murre (dialect). SMB-protokollan alkuperäisen kehitystyön teki Barry Feigenbaum IBM:lle. Microsoft hankki protokollan, liittäen sen osaksi LAN Manager-tuotettaan. Vuonna 1996 Microsoft käynnisti aloitteen, jossa SMB nimettiin uudelleen CIFS:ksi ja siihen lisättiin ominaisuuksia. SMB:stä on olemassa myös unix-käyttöjärjestelmille tarkoitettu versio, joka tunnetaan nimellä Samba. CIFS:n

kehitys kulkee tällä hetkellä vuonna 2012 julkaistussa versiossa SMB 3.0. CIFS on tilallisia yhteyksiä hyödyntävä protokolla, jonka liikenne kulkee TCP/IP:n päällä.

Kuten NFS:kin CIFS tarjoaa asiakaskoneilla mahdollisuuden käyttää tiedostoja ja hakemistoja etäpalvelimelta. CIFS käyttää tiedosto- ja rekisterilukkoja, joilla tiedon eheys saadaan varmistettua koska, toisilta käyttäjiltä on estetty kirjoitus näihin tiedostoihin ja rekistereihin. CIFS:ssä on myös sisäänrakennettu vikasietoisuus, joka palauttaa yhteyden ja uudelleen avaa tiedostot jotka olivat auki yhteyden katketessa. Lisäksi koska CIFS on tilallinen protokolla, CIFS-palvelin säilyttää tiedot avoinna olevista yhteyksistä ja virhetilanteessa asiakas saa tiedon palvelimelta. [24.]

4.3 Objektipohjainen tallennus (ObS)

Objektipohjainen tallennus (Object based Storage) on tallennusprotokolla, joka perustuu Carnegie Mellon yliopiston (CMU) Parallel Data-laboratorioiden vuonna 1996 alkaneeseen kehitysprojektiin. CMU on edelleen yksi objektitallennusta kehittävistä organisaatioista, mutta vuosituhannen vaihteen jälkeen objektitallennusta ovat erityisesti edistäneet suuret yritykset kuten, EMC (Centera), HP (OpenStack), Amazon (AWS S3) ja Google (GoogleFS), jotka ovat kehittäneen omia yksinoikeudellisia objektitallennusratkaisuja. Lisäksi meneillään on useita avoimen lähdekoodin kehitysprojekteja, kuten esimerkiksi tämän insinöörityön toteutuksessa käytetty OpenStack Swift. Objektitallennuksen standardoinnin kehityksestä vastaavat, SNIA (Storage Networking Industry Association) OSD (Object storage device)-työryhmä sekä INCITS T10-komitea. Ensimmäisen SCSI-protokolla perheeseen kuuluvan standardin OSDv1 julkaisi T10 vuonna 2004. Tämä OSD-standardi julkaistiin nimellä ANSI INCITS 400–2004. Viimeisin julkaistu standardointi on tällä hetkellä OSD-2-standardissa, joka on julkaistu nimellä ANSI INCITS 458–2011. [25.]

Toisin kuin lohko- ja tiedostopohjaisessa tallennuksessa, jossa isäntäkoneen täytyy olla tietoinen käsiteltävän tiedoston (tiedostopolku) tai tietolohkon (esim. LBA osoite) sijainnista tallennusjärjestelmässä, objektipohjaisessa tallennusarkkitehtuurissa tiedon sijainti on peitetty asiakkaalta. Asiakkaiden tietoon kohdistamat operaatiot kohdistuvat tiedostonomaisiin tallennusobjekteihin, jotka sisältävät tallennetun tiedon lisäksi objektin metatiedot (nimi, koko, päiväys, omistajuus, jne.), attribuutit (käyttö-oikeudet, säilytysaika, jne.) ja 64-bittisen käyttäjäobjekti (user object) tunnisteen. Esimerkiksi tiedos-

topohjaisessa tallennuksessa, isäntäkoneen täytyy tietää tarkka osoite tiedostoon, johon operaatioita halutaan kohdistaa. Objektitallennuksessa asiakkaan tarvitsee vain kuvailla sen tiedoston metatietoja (esimerkiksi nimi), johon operaatiot halutaan kulloinkin kohdistaa. Objektitallennusjärjestelmän objektienhallintakomponentti (OSD storage component) määrittää tämän annetun metatiedon pohjalta, mikä tallennusobjekti (object) on kyseessä ja välittää kyseisen objektin käyttäjäobjektintunnisteen ja partitio-objektitunnisteen pohjalta muodostetun 128-bittisen tallennusobjektitunnisteen asiakkaalle, jolloin tallennusobjektin sisältämä tieto voidaan hakea asiakkaan käsiteltäväksi. Tämän tunnisteen avulla asiakas voi jatkossa operoida objektia suoraan siitä tallennuslaitteesta (Storage Node), missä tallennusobjekti sijaitsee. Tästä logiikasta johtuen objektitallennuksen I/O-operaatiot ovat yhtä tehokkaita kuin lohkotallennuslaitteissa. Mutta koska asiakaan ei tarvitse tietää tallennusobjektin fyysistä sijaintia massamuistilaitteella tai tallennusjärjestelmässä, objektitallennuksella saavutetaan sama isäntä-asiakasriippumattomuus joka on tiedostopohjaisissa tallennusjärjestelmissä.

Koska objektitallennuksessa ei ole tarvetta hakemistorakenteelle tiedon löytämiseksi, on objektitallennuksen nimiavaruus litteä. Tämä tarkoittaa sitä, että asiakkaan näkökulmasta katsottuna kaikki tallennusobjektit ovat hierarkkisesti samalla tasolla. Tästä johtuen objektitallennus täyttää etäresurssipalvelulle asetetut määritelmät, se on näennäinen ja määrittelemätön. Asiakkaan kannalta näennäistä tallennustilaa on saatavilla äärettömästi, ja koska asiakas ei tiedä, millä massamuistilaitteella tai missä tallennusjärjestelmässä tieto on, objektitallennus täyttää määrittelemättömyyden määritteen. Objektitallennus soveltuukin erittäin hyvin juuri etäresurssipalvelujen käyttöön objektitallennusjärjestelmän helpon skaalautuvuuden vuoksi. Jos objektitallennusjärjestelmän kokoa täytyy muuttaa, tarvitsee vain lisätä tai poistaa tallennussolmuja objektitallennusjärjestelmään. Tallennussolmut ovat yleensä näennäispalvelimia, joilla on tietty määrä tallennuskapasiteettia käytössään. Koska jokainen objektitallennusjärjestelmä tallentaa vikasietoisuuden vuoksi samasta tiedosto useampia tallennusobjekteja eri tallennussolmuille, voidaan yksittäinen tallennussolmu poistaa käytöstä ilman hallinnollisia toimenpiteitä tai tiedonsiirtoa pois tallennussolmulta (storage node). Objektienhallintakomponentti havaitsee, jos tallennusobjekteista ei ole riittävästi kopioita (replica) eri tallennussolmuilla, ja luo automaattisesti tarvittavat kopiot eri tallennussolmuille. Koska objektitallennusjärjestelmään on sisäänrakennettuna vikasietoisuudesta huolehtiva logiikka, voidaan objektitallennuskerroksen alla oleva fyysinen tallennuskerros toteuttaa edullisilla lohkotallennusjärjestelmillä.

Koska objektitallennusta käyttävän asiakkaan ei tarvitse tietää tiedon sijaintia, vaan tallennusobjekti haettiin kuvailemalla tiedon metatietoa, on objektitallennukseen tallennettu tieto helposti monistettavissa ympäri maailmaa. Samasta objektitallennusjärjestelmään tallennetusta tiedosta voidaan haluttaessa luoda eri puolille maailmaa omat erilliset tallennusobjektit. Se, minkä objektin objektitunniste tietoa kysyttäessä asiakkaalle palautetaan, riippuu sitten, mistä päin maailmaa asiakas tietoa pyytää. Yleisesti objektitallennuksessa tiedon monistuksesta puhuttaessa käytetään käsitettä löyhä yhdenmukaisuus (loose uniformity). Löyhä yhdenmukaisuus tarkoittaa sitä, että on hetkellisesti mahdollista, että tiedon eri ilmentymät eroavat toisistaan, mutta tieto on aina kuitenkin jotain, mitä tallennusobjektiin on joskus tallennettu. Saattaa vain olla, että tietoa on juuri muutettu toisella puolella maailmaa, mutta muutos ei ole vielä kopioitunut paikalliseen objektiin. Tämä johtuu siitä, että asiakas operoi aina paikallista tallennusobjektia ja operaatio kuitataan valmiiksi asiakkaalle kun muutokset on tallennettu paikalliseen tallennusobjektiin. Mikäli tallennusobjektista on monistettu ilmentymä useampaan sijaintiin, muutosten replikointi eri ilmentymiin tehdään sen jälkeen kun muutos on kuitattu valmiiksi asiakkaalle. Samasta tiedosta eri sijainneissa olevien tallennusobjektien olemassaolon mahdollistaa OSD-arkkitehtuurin käsite alue (region). Kukin alue koostuu itsenäisistä OSD-instansseista ja ala-alueista. Toisistaan riippumattomia alueita voi samassa OSD-toteutuksessa olla rajaton määrä. Kunkin alueen hallinnasta vastaa aluehallitsijapalvelu (regional manager). Aluehallitsija on metatietopalvelu, joka sisältää tiedon alueeseen kuuluvista tallennusobjekteista. Lisäksi palvelun vastuulle kuuluvat oman alueensa tietoturvasta huolehtiminen, asiakkaitten tiedonhakupyyntöihin vastaaminen sekä eri alueiden välillä tapahtuvasta tallennusobjektien kopioinnista. Saadessaan tiedonhakupyynnön asiakkaalta, aluehallitsija palauttaa asiakkaalle tiedon omasta sijainnista jotta asiakas voi saamiensa tietojen perusteella päättää mikä on sitä lähin alue. [27.]

Vaikka objektitallennuksen nimiavaruus onkin asiakkaan näkökulmasta litteä, objektitallennuksen sisäinen hierarkia kuitenkin koostuu erityyppisistä ja tasoisista objekteista. Nämä ovat juuriobjekti (root object), partitio-objekti (partition object), käyttäjäobjekti ja kokoelmaobjekti (collection object). Käyttäjäobjekti on 64-bittisellä tunnisteella varustettu tallennusobjekti, johon tallennettu tieto sijoitetaan. Juuriobjekti toimii jokaisen loogisen OSD-yksikön ylimmän tason objektina, juuriobjektin voidaan ajatella käsitteenä vastaavan lohkotallennuslaitteiden osiointitauluja (partition table). Partitio-objektit, joiden voidaan ajatella vastaavan lohkotallennusjärjestelmän osioita, muodostavat kokoelman käyttäjäobjekteja siten, että yksittäinen käyttäjäobjekti voi sijaita vain yhdessä

partitio-objektissa. Partitio-objekteja muodostetaan kunkin juuriobjektin alle juuriobjektin kiintiömääritysten (quota) mukaisesti ja jokainen partitio-objekti saa luodessa oman 64-bittisen tunnisteen. Kokoelmaobjekti muodostaa loogisen hallinnollisen kokonaisuuden yhden partitio-objektin alla sijaitsevista saman käyttäjän käyttäjäobjekteista, tarjoten objektitallennukselle tehokkaamman tavan organisoida navigointi käyttäjäobjektien välillä.

Vaikka tallennusobjektit ovatkin objektitallennuksessa asiakkaan näkökulmasta hierarkkisesti samalla tasolla, on objektit pystyttävä esittämään käyttäjän kannalta käsiteltävissä olevana tietorakenteena eli tiedostoina ja kansioina, joten objektitallennusarkkitehtuurissa tarvitaan tiedostojärjestelmä, joka sijaitsee isäntäkoneella. Isäntäkone keskustelee OSD-laitteiden kanssa objektiliitännän kautta, joka toteuttaa tarvittavat toimenpiteet, joilla isäntäkoneella olevaan tiedostojärjestelmään kohdistuvat operaatiot voidaan muuntaa objektitallennuksen käsittämiksi operaatioiksi. Tämä kommunikaatio on mahdollistettu tarjoamalla objektiliitännälle ohjelmointirajapinnat (API), joilla tarjotaan käyttömahdollisuudet objektitallennuksen toiminnallisuuksille. Tiedonsiirto isäntäkoneiden ja OSD-laitteiden välillä tapahtuu käyttämällä ReST (Representional State Transfer)-arkkitehtuurimallia tai SOAP (Simple Object Access Protolcol)-protokollaa käyttäen. [27.]

4.4 Ohjelmointirajapinta (API)

Ohjelmointirajapinta (Application Programming Interface) määrittää, miten eri sovelluskomponentit toimivat yhteistyössä. Yleisesti sovellusten tekijät tekevät tämän tarjoamalla API:n kautta kokoelman funktioita tai rutiineja, joilla tarjotaan käyttömahdollisuus yleisimmille toiminnoille. Objektitallennuksessa API:t tarjoavat SOAP- ja ReST-protokollille määrittelyt niille etäkutsuille, joita tarjotaan API:n käyttäjille.

4.5 Esittävä tilallinen (ReST)

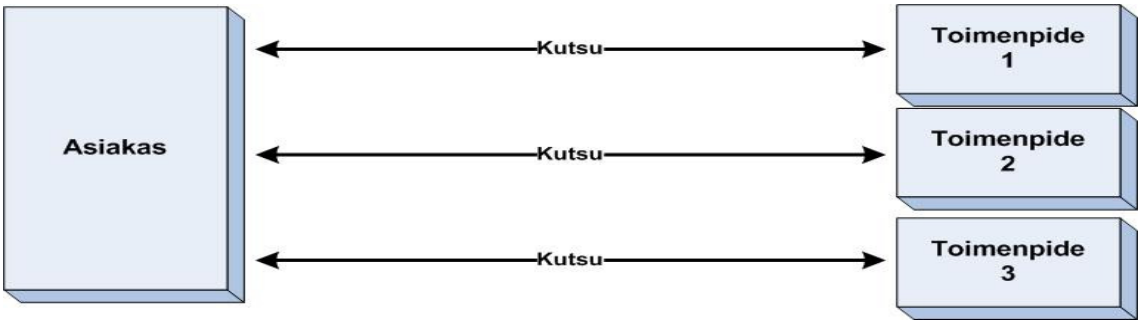
HTTP (Hypertext Transfer Protolcol) -protokollaan perustuva arkkitehtuurimalli, joka on kehitetty modernien verkkosovellusten käyttämiseksi. ReST tarjoaa mahdollisuuden suorittaa muutamia kuvassa 6 esiteltyjä perustoimintoja, kuten tiedostonhaku,

tiedostonmuokkaus, -luonti ja –poisto, käyttäen HTTP:n määrittelyjä. Kun isäntäkone kohdistaa pyynnön OSD-laitteelle ReST:iä käyttäen, kulloinkin käsiteltävän objektin osoittamiseen käytetään URI (Uniform Resource Identifiers)-merkkijonoa, jossa määritellään käsiteltävänä olevan objektin paikka ja yksikäsitteinen nimi. Pyynnössä määriteltyjen parametrien perusteella, OSD-laite palauttaa isäntäkoneelle objektin tietoa joko XML (Extensible Markup Language) tai HTML (Hypertext Markup Language) muodossa. [28 & 29.]

Kuva 6. REST:n perustoiminnot

4.6 Yksinkertainen objektien käsittely protokolla (SOAP)

SOAP-protokolla on kevytrakenteinen, XML-pohjainen hajautetun tiedon käsittelyyn soveltuvat verkkoprotokolla. SOAP mahdollistaa proseduurien etäkutsun eri käyttöjärjestelmissä ajettujen ja eri ohjelmointikieliin pohjautuvien sovellusten välillä kuten kuvassa 7 on esitetty. SOAP:in prosessit mahdollistavat http-otsikkojen ja XML-tiedostojen koodauksen siten, että on mahdollista siirtää tietoa eri järjestelmien välillä ja kutsua eri proseduurien toimintoja. Kuten ReST, SOAP käyttää URI-merkkijonoja kulloinkin käsiteltävän olevan objekti osoittamiseen. [30.]



Kuva 7. SOAP toiminta

5 OpenStack Swift-objektitallennuksen käsitteistö ja teoreettinen malli

Insinöörityön käytännöntoteutuksen pohjaksi valittiin OpenStack Swift, joka on Yhdysvaltalaisen etäresurssipalveluntarjoajan Rackspacen objektitallennusratkaisun lähdekoodiin pohjautuva avoimen lähdekoodin objektitallennusratkaisu. Se on osa OpenStack-säätiön avoimen lähdekoodin perustuvaa etäresurssipalvelukäyttöjärjestelmää, jolla on mahdollista toteuttaa ja hallita etäresurssipalveluina laskenta-, tallennus- ja verkkoratkaisuja.

OpenStack Swift valittiin, koska se on objektitallennusjärjestelmä, jonka toteutus on helposti tehtävissä paikallisena, vain muutamia kiintolevyjä sisältävien palvelimien toteutuksena. Mutta se on helposti skaalattavissa aina globaalisti hajautetuksi, tuhansia palvelimia ja petatavuja tallennustilaa sisältäväksi yritysratkaisuksi vain lisäämällä palvelimia toteutukseen. Swift poikkeaa muista avoimen lähdekoodin objektitallennustoteutuksista siinä, että sen juuret ovat laajassa tuotantokäyttöön tarkoitetussa järjestelmässä. Näinpä se kestää virhetilanteita merkittävän hyvin, se on helposti laajennettavissa ja huollettavissa ilman suuria toimenpiteitä ja se on optimoitu toimimaan tietovarastona jota käytetään verkkoprotokollilla. Sen tärkeimmät ominaisuudet ovat seuraavat:

- Kaikki tallennusobjektit osoitetaan URL:llä (Uniform Resource Locator), osoite on muotoa http://hostname.domain.fi/v1/account/container/object.

- Jokainen tallennusobjekti on replikoitu vähintään kolmeen eri partitio-objektiin, jotka on hajautettu järjestelmän sisällä, joten tieto voi sijaita tai sitä voidaan siirtää, mihin tahansa toteutuksen sisällä.

- Käyttäjät tai sovellukset ovat vuorovaikutuksessa tallennetun tiedon kanssa ReST http API:en yli.

- Se on skaalattavissa ilman huoltokatkoja tai vaikutusta ryväksen toimintaan.

- Vioittunut tieto korjataan automaattisesti.

- Se on toteutettavissa millä tahansa standardipalvelinlaitteistoon asennetulla Linux-käyttöjärjestelmällä.

Swift-toteutus on siis erittäin skaalautuva ja vikasietoinen käyttäjiään standardoiduilla protokollilla palveleva objektitallennusratkaisu. Se koostuu joukosta paikallisesti tai maantieteellisesti hajautettuun ryväkseen kuuluvia palvelimia sekä niitä palvelevista välittäjäkerroksen palvelimista. Toteutukseen kuuluvat palvelimet on jaettavissa kolmeen päätyyppiin, proxy- ja tallennussolmuihin sekä identiteettipalvelua tarjoaviin palvelimiin. Eri palvelimien toiminnallisuudet esitellään seuraavissa luvuissa. [31; 32; 33.]

Koska tätä insinöörityötä varten ei ollut käytettävissä fyysisiä palvelimia, poikettiin normaalista Swift toteutuksesta siten että, kaikki palvelimet ovat VMwaren päällä ajettavia näennäispalvelimia. Palvelimet saavat Swiftin käyttöön tarjotut levynsä iSCSI-levyinä erillisestä OpenFiler-levyjärjestelmästä.

5.1 Välityssolmut

Välityssolmuina (proxy node) toimivat palvelimet ovat yhdyskäytäväpalvelimia (gateway servers), jotka liittävät asiakasverkot (esimerkiksi Internet) objektitallennuspalveluun ja tarjoavat julkiset API:t asiakkaille. Kukin välityspalvelin toimii oman alueensa aluehallitsijana sekä metatietopalvelimena, jonka vastuulla on asiakkaan lähettämän tietopyynnön ohjaaminen oikean tallennusobjektin käsiteltäväksi. Tallennettujen tallennusobjektien sisällys ja sijainti tietoa säilytetään juuriobjekteissa, joita kutsutaan Swift-toteutuksessa renkaiksi. Rengas käsitteenä käydään lävitse seuraavassa luvussa.

Välityspalvelimen vastuulle kuuluu myös oman alueensa objektitallennuksen vikasietoisuuden ylläpito. Kun välityspalvelin havaitsee, että pyydettyä operaatiota ei voida välittää oikealle tallennusobjektille, se kommunikoi muiden juuriobjektien kanssa, mistä muusta partitio-objektista löytyy kopio samasta tallennusobjektista jotta pyydetty operaatio voidaan suorittaa loppuun. Samalla käynnistyy kunkin renkaan sisäinen korjaustoiminto, jossa varmistetaan, että partitio-objekteista on saatavilla riittävä määrä käytettävissä olevia kopioita.

Lisäksi yhdyskäytäväpalvelin vastaa tietoliikenteestä asiakkaiden ja tallennusobjektien välillä kuitenkin siten, että mitään tietoa ei tallenneta yhdyskäytäväpalvelimelle. Ainoastaan poikkeustilanteissa, joissa välityssolmu ei kykene suorittamaan pyydettyä operaatiota tallennusobjektille, operaatio saatetaan tallentaa paikallisesti ja se suoritetaan myöhemmin loppuun. [31; 32; 33.]

5.2 Rengas

OpenStack Swift-toteutuksen päätasona on renkaaksi (ring) kutsuttu juuriobjekti. Renkaita on olemassa kolmea eri tyyppiä, objekti- (object), tili- (account) ja säiliörengas (container). Kukin rengas muodostaa kokoelman tiedoista, josta löytyy tietyn toiminnallisuuden toteuttavat tallennusobjektit. Juuriobjekti hyödyntää vyöhykkeiksi ja alueiksi kutsuttuja kokoelmaobjekteja hajauttaakseen partitio-objektien sijainnit eri fyysisiin sijainteihin objektitallennuspalvelinryväksen sisällä. Kunkin juuriobjektin vastuulla on huolehtia siitä, että kustakin sille kuuluvista partitio-objekteista on vähintään kolme kopiota, jotka on hajautettu eri kokoelmaobjekteihin. Tätä renkaan ylläpitämään hajautustietoa käytetään muun muassa objektitallennuksen vikasietoisuuden ylläpitoon. Kun objektitallennus havaitsee että tallennusobjektiin joka sijaitsee renkaan tietyssä partitio-objektissa, ei voida suorittaa operaatioita, se pyytää rengasta poistamaan kyseisen partitio-objektin pois käytöstä ja huolehtimaan siitä, että tästä partitio-objektista sekä sen sisältämistä tallennusobjekteista on olemassa riittävä määrä kopioita hajautettuna eri kokoelmaobjekteihin. Jotta vikasietoisuus ja hajautus olisi mahdollisimman korkealla tasolla, kukin rengas sijoittaa uuden kopion partitio-objektista siihen kokoelmaobjektiin, mikä on sille vähiten käytetty. [31; 32; 33.]

5.3 Alueet ja vyöhykkeet

Alueet (zone) ja vyöhykkeet (region) ovat eri partitio-objekteja sisältäviä kokoelmaobjekteja, joilla varmistetaan objektitallennuksen vikasietoisuus. Yksittäinen kokoelmaobjekti voi muodostua yhdestä tai useasta kiintolevystä, yhdestä tai useammasta palvelimesta, palvelinräkistä tai jopa tietokonekeskuksen sisältämistä palvelimista ja kiintolevyistä. Jokaiseen kokoelmaobjektiin kuuluu vähintään yksi partitio-objekti jostain renkaasta, mutta käytännössä ne muodostuvat sadoista, jopa miljardeista eri renkaisiin kuuluvista partitio-objekteista. Ainoa ehto on, että kaksi saman tallennusobjektin sisäl-

tävää partitio-objektia, eivät voi kuulua samaan alueeseen. Ne voivat kuulua samaan vyöhykkeeseen, mutta silloinkin tuolla vyöhykkeellä on oltava vähintään kopioiden määräksi määritellyn luvun verran alueita aluetta. [31; 32; 33.]

5.4 Objektien kopiointi

Kuten aikaisemmin todettiinkin, objektitallennus käyttää objektien kopiointia vikasietoisuuden ja suorituskyvyn optimointiin hajauttamalla kunkin partitio-objektin eri kokoelmaobjekteihin. Objektienreplikointia käytetään myös kopiointitiedon siirtoon samojen tallennusobjektien välillä. Kun jotakin tallennusobjektia on muokattu, päivitetään sen metatietoja, kuten aikaleimaa. Objektitallennuksen sisäiset prosessit muodostavat eräajoina ajastetusti tiivistelistoja eri objekteista ja vertailevat näin onko jotakin objektia tarvetta päivittää. Mikäli päivitystarve havaitaan, prosessit suorittavat objektien päivityksen siten, että se objekti joka sisältää uusimman aikaleiman, pakottaa muutoksensa muihin saman objektin kopioihin.

Edeltävässä kappaleessa kuvatusta säännöstä on kuitenkin olemassa poikkeus. Tämä poikkeus liittyy poistettujen objektien käsittelyyn. Kun objektille suoritetaan poisto-operaatio, asetetaan kyseisen objektin päätteeksi ns. hautakivitunniste ja objektin sisältö nollataan. Objektit, joilla on hautakivitunniste, replikoidaan aina muihin saman objektin kopioihin, huolimatta aikaleimasta. Näin vältetään tilanne, jossa jo poistettu objekti voisi palata kummittelemaan mahdollisessa järjestelmävirheessä. Objektien kopiointi siis hyödyntää objektin aikaleimaa sekä hautakivitunnistetta tiedon yhdenmukaisuuden ja poiston hallintaan. Koska kopiointi suoritetaan ajoittain, siihen liittyy kuitenkin käsitteenä jo aikaisemmin käsitelty löyhä yhdenmukaisuus. On siis mahdollista, että käyttäjälle palautetaan jo poistettu tai vanhentunutta tietoa sisältävä objekti, koska kopiointiajoa ei ole vielä suoritettu. Mutta koska objektitallennusjärjestelmä pyrkii palauttaman sen tallennusobjektin kopion osoitteen, joka on viimeksi käsitelty, tämä on yleensä vain järjestelmävirheessä esiintyvä poikkeus. [31; 32; 33.]

5.5 Objektien auditointi

Objektien auditointi on objektitallennusjärjestelmän sisäinen prosessi, jolla huolehditaan siitä, että objektitallennusjärjestelmä ei sisällä objekteja, jotka olisivat turmeltuneet. Mikäli turmeltunut objekti löytyy, kyseinen kopio laitetaan karanteeniin ja sen muista kopiosta kopioidaan kelvollinen kopio partitio-objektiin. [31; 32; 33.]

5.6 Tilit ja säiliöt

Jotta voitaisiin hallita, mihin tallennusobjekteihin kullakin käyttäjällä on käyttö-oikeus ja tallennusobjekteille on olemassa hallittava hierarkia, Swift-objektitallennuksen juuriobjekteihin kuuluvat tili- ja säiliörenkaat. Käytännössä nämä renkaat sisältävät joukon partitio-objekteja, joihin on tallennettu SQLite-tietokantoja sisältäviä tallennusobjekteja. Kukin asiakkuuden tilitietokanta-tallennusobjekti sisältää tiedon siitä, mihin säiliötietokanta-tallennusobjekteihin kullakin käyttäjällä on oikeus. Säiliötietokanat taas sisältävät tiedon kuhunkin säiliöön kuuluvista tallennusobjekteista ja käyttäjän niihin kohdistuvista käyttö-oikeuksista, eli säiliötietokannan voidaan ajatella vastaavan Windows käyttöjärjestelmän kansiota tai Unixin hakemistoa. [31; 32; 33.]

5.7 Tallennussolmut

Tallennussolmuina toimivat Swift-infrastruktuurin ytimen muodostavat palvelimet, joille erityyppiset objektit tallennetaan. Tyypillisessä OpenStack Swift-toteutuksessa nämä tallennussolmut ovat hyvin peruslaitteistolla varustettuja fyysisiä palvelimia, joissa on yksi tai useampia kiintolevyjä. Koska kirjoitus- ja lukuoperaatioiden optimointi sekä vikasietoisuus toteutetaan sovellustasolla, voidaan kiintolevyinä käyttää edullisia ja suurikapasiteettisia mekaanista kiintolevyä, joka on liitetty SATA-väylällä. Edellä mainituista syistä, mitään RAID-levyryhmä-konfiguraatioita ei tarvita tiedon suojaamiseksi, ja Rackspacen suorittamissa testauksissa on havaittu, että RAID-levyryhmien käyttö jopa heikentää järjestelmän suorituskykyä.

Swift-arkkitehtuuri sisältää kunkin juuriobjektin tyyppiset tallennussolmut. Suosituksena on kuitenkin, että parhaimman hajautuksen saavuttamiseksi kullekin tallennussolmulle asennettaisiin kaikkien juuriobjektien palvelut. Tallennussolmujen tyypit ovat objekti-, säiliö- sekä tilitallennussolmu. Kunkin tallennussolmun palvelut esitellään seuraavissa kappaleissa. [31; 32; 33.]

Objektitallennussolmu (storage node) on objektien tallennuksesta vastaava tallennussolmu. Se ei tiedä, mitä tietoa kukin tallennettu objekti sisältää, vaan sen tehtävänä on ainoastaan tallentaa objektit palvelimien massamuistilaitteille. Kaikki objektit tallennetaan binääritiedostoina palvelimen tiedostojärjestelmään metatietojensa kanssa, jotka tallennetaan tiedoston XATTRS (Extended Attribute Set) -attribuutteihin.

Objektitallennussolmupalvelinta konfiguroitaessa, on palvelimen objektitallennuksen käyttöön tulevien massamuistilaitteiden tiedostojärjestelmää valittaessa muistettava käyttää tiedostojärjestelmää, joka tukee XATTRS:ia. Poiketen normaaleista tiedostoattribuuteista, XATTRS mahdollistaa liittää tallennettuun tiedostoon attribuutteja, joita tiedostojärjestelmä ei kykene tulkitsemaan. Useimmat nykyaikaiset tiedostojärjestelmät, kuten ext2-4, JFS, ReiserFS, XFS ja Btrfs, tukevat XATTRS:ja, jos ne on aktivoitu käyttöjärjestelmän kernelistä. OpenStack kuitenkin suosittaa XFS:n käyttöä, koska se on tiedostojärjestelmä, joka kiinnittää huomionsa metatiedon eheyteen, eikä niinkään tallennetun tiedoston eheyteen.

Objektitallennussolmu muodostaa kullekin tallennettavalle objektille tallennuspolun käyttäen parametreina objektin nimeä ja tallennusaikaa. Käytännössä tämä tarkoittaa sitä, että kukin tallennusoperaatio muodostaa uuden tallennusobjektin ja se kopioidaan muihin partitio-objekteihin. Tallennusobjektin vanhemman version tallennusobjekti tyhjennetään, jolloin kooksi tulee 0 tavua ja sen tiedostopääte muutetaan hautakivitiedostopäätteeksi ”.ts”. Objektitallennusjärjestelmän objektinkopiointi toiminnallisuus huolehtii tämän jälkeen vanhemman version poistosta objektitallennussolmuilta. [31; 32; 33.]

Tilitallennussolmut (account node) vastaavat niistä tallennusobjekteista, jotka sisältävät käyttäjätilien SQLite-tietokannat. Tietokanta sisältää tiedot käyttäjätileistä, niihin liitetyistä säiliöistä sekä niiden käyttö oikeuksista.

Säiliötallennussolmut (container node) vastaavat niistä tallennusobjekteista, jotka sisältävät SQLite-säiliötietokannat. Tietokanta sisältää tiedon siitä, missä partitio-objektissa kukin käyttäjän tallennusobjekti sijaitsee.

5.8 Tunnistuspalvelu

Tunnistuspalvelu (identity service) vastaa objektitallennuksen käyttäjätunnistuksesta ja heidän käyttö-oikeuksiensa määrittelystä. OpenStack Swift:iin on mahdollista luoda palvelun sisäisiä käyttäjätunnuksia ja määritellä niille käyttö-oikeuksia tallennettuihin objekteihin. Tämä ei kuitenkaan ole ylläpidettävyyden tai käytettävyyden kannalta hyvä ratkaisu. OpenStack suosittaakin, että kaikkien OpenStackin sovellusten käyttäjienhallintaan käytettäisiin jotain konfiguroitavissa olevaa WSGI-yhteensopivaa väliohjelmistoa, joka suorittaa käyttäjätunnistuksen.

OpenStack tarjoaa käyttäjätunnistuksen toteuttamiseksi OpenStack Keystone-nimistä sovellusta. Keystone voidaan integroida Swiftiin siten, että Swiftin käyttäjätunnistus tapahtuu Keystonen avulla ja Keystone tarjoaa käyttäjille palvelukirjaston, josta käyttöoikeudet omaava henkilö voi hakea eri OpenStack Swift API:en verkko-osoitteet. Näin Keystone-palvelu ja Swift välityspalvelimet muodostavat Swift OSD-arkkitehtuurissa aluehallitsijapalvelun, jolla voidaan määritellä mistä Swift välityspalvelimesta asiakkaasta kulloinkin palvellaan.

Jotta käyttäjienhallintaa voitaisiin suorittaa keskitetyllä ratkaisulla, on Keystone integroitavissa LDAP:in (Lightweight Directory Access Protocol) avulla esimerkiksi Mircosoftin aktiivihakemistoon (Active Directory) tai vastaaviin UNIXin hakemistopalveluratkaisuihin. Käyttäjänhallinta voidaan toteuttaa myös pelkän Keystone-palvelun avulla, jolloin käyttäjätiedot tallennetaan paikallisesti Keystone-palvelimelle MySQL-tietokantaan.

Keystonen toteutukseen liittyvät seuraavat käsitteet:

- käyttäjä (user)
- valtuutustieto (credentials)
- todennus (authentication)
- tunnisteavain (token)
- asiakas (tenant)
- palvelu (service)
- päätepiste (endpoint)

Keystonessa käyttäjä on digitaalinen määritelmä jollekin, joka käyttää OpenStackin etäresurssipalveluja. Tämä voi olla esimerkiksi henkilö, järjestelmä tai jokin palvelu. Keystone vaatii kaikkia käyttäjiä, jotka pyytävät tunnistusta, tarjoamaan jonkin valtuutustiedon, joka on Keystonessa yhdistetty käyttäjään ja on vain käyttäjän tiedossa. Tämä valtuutustieto voi olla esimerkiksi salasana, käyttäjätunniste ja API-avain tai tunnistekoodi. Näiden tietojen pohjalta Keystone suorittaa todennuksen jolla varmistetaan se, että käyttäjä on, se kuka hän väittää olevansa. Mikäli todennus onnistuu, käyttäjälle tarjotaan tunnistekoodi jota Swift käyttää käyttäjän käyttö-oikeuksien määrittelemiseksi. Käytännössä tämä tunnistekoodi on satunnaislukugeneraattorin luoma teksti. Kukin tunnistekoodi määrittää myös, mitä resursseja sillä voidaan käyttää ja sillä on voimassaoloaikansa. Tämä tarkoittaa sitä, että tunnistekoodia luotaessa määritetään sille myös päättymispäivä, jonka jälkeen se ei ole voimassa. Keystone voi myös peruuttaa tunnistekoodin voimassaolon aikaisemmin kuin mikä on määritelty päättymispäivä.

Asiakas on Keystonen hallinnollinen säiliö, jolla voidaan määritellä resursseja, käyttäjiä tai muita Keystonen tietueita joukkioiksi. Tällainen joukko voi olla esimerkiksi tiettyyn projektin parissa työskentelevät tai tiettyyn yritykseen kuuluvat käyttäjät. Asiakkuuksilla voidaan rajata, mihin OpenStack palveluihin tai päätepisteisiin joukkiolla on käyttöoikeus.

Palveluja ovat eri OpenStack sovellukset, kuten Swift. Kullekin palvelulle voidaan Keystonessa määritellä yksi tai useampi päätepiste mistä tuon palvelun API:t löytyvät, eli esimerkiksi jokainen Swift välityspalvelin on oma päätepisteensä. [31; 32; 33.]

6 OpenStack Swift -objektitallennuksen toteutus

6.1 Toteutuksen lähtökohdat

Vaikka OpenStackin kehittäjä Rackspace suosittaa, että Swift-infrastruktuuri koostuisi vain fyysisistä palvelimista, joissa on halpoja suurikapasiteettisia massamuisteja. On suositusten mukaisessa vaihtoehdossa helposti nähtävissä ongelmia, jos sitä verrataan siihen, miten yrityksen nykyään rakentavat palvelinympäristönsä ja millaisia käytettävyysaste vaatimuksia niille asetetaan. Ensinnäkin monet yrityksen ovat panostaneet vahvasti näennäispalvelin ympäristöihin, joten ne eivät välttämättä ole kovinkaan kiinnostuneita hankkimaan fyysisiä palvelimia jonkin ratkaisun toteutuksen vuoksi. On helposti arvioitavissa, että eri Swift-solmujen muistin- ja laskentatehonkäyttö ei ole vakio koko ajan, jolloin ajoittaista ylikapasiteettia saattaa ilmetä, jos tarkastellaan yksittäisten solmujen muistia ja laskentatehoa. Näin ollen, jotta hankittu muisti- ja laskentatehokapasiteetti olisi mahdollisimman tehokkaassa käytössä, olisi ehkä järkevä käyttää näennäispalvelimia, jolloin käyttämätöntä muisti- ja laskentakapasiteettia voidaan hyödyntää muualla silloin kun jokin Swift-solmu ei sitä tarvitse, eli käyttöaste näille resursseille pysyisi mahdollisimman korkeana. Lisäksi laajennettavuus helpottuisi huomattavasti, koska uuden Swift-solmun lisääminen infrastruktuuriin on erittäin helppoa. Luodaan vain uusi näennäispalvelin ja määritellään sille levyt levyjärjestelmästä. Tällöin vältettäisiin useimmissa yrityksissä uuden fyysisen palvelimen hankintaan liittyvä hankintaprosessi ja laajennus voitaisiin sopivalla automatisointityökalulla automatisoida. Mikäli vastaavaa nopeutta haluttaisiin saada fyysisillä palvelinlaitteistoilla, tämä tarkoittaisi sitä, että yrityksen olisi varastoitava näitä palvelimia omaan tietokonekeskukseensa. Tämä tarkoittaisi myös palvelinlaitteistoon sitoutuneita pääomia sekä mahdollisia juok-

sevia kustannuksia, kuten sähkö ja jäähdytys, koska palvelimia jouduttaisiin pitämään valmiustilassa, jotta ne voitaisiin tarvittaessa automaattisesti lisätä Swift-infrastruktuuriin.

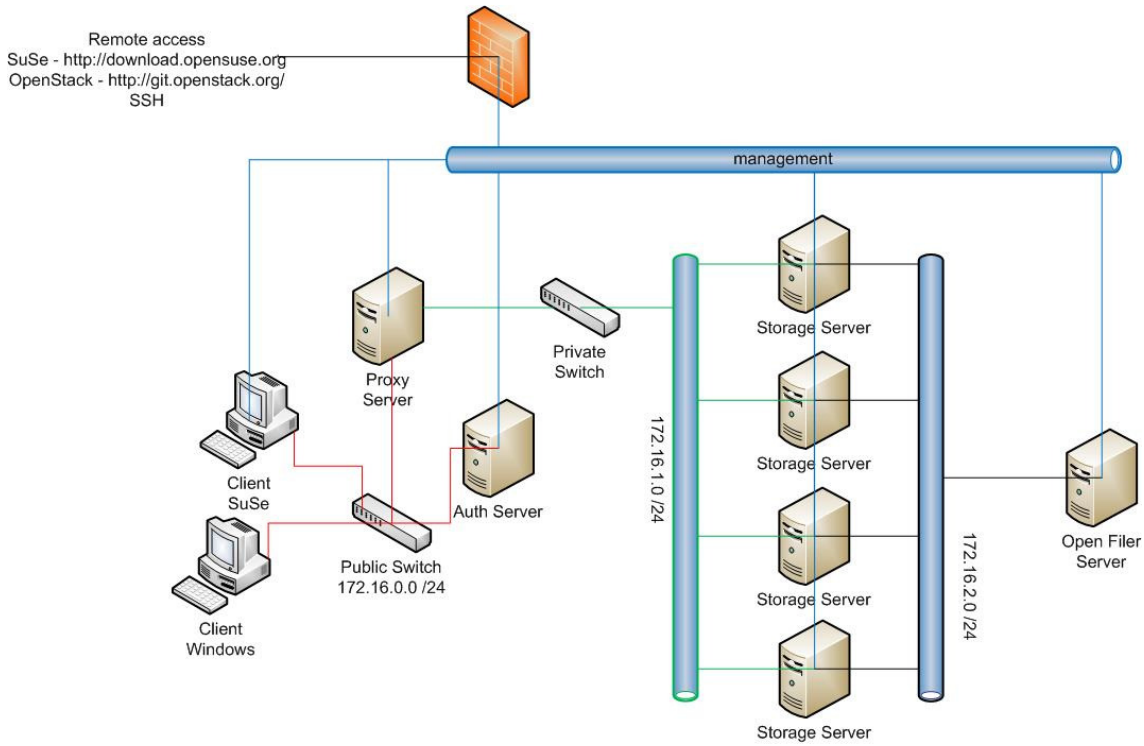
Edellä mainituista syistä johtuen haluttiin tutkia, onko mahdollista rakentaa toimiva OpenStack Swift-toteutus yhdistelmälle virtuaalipalvelimet ja erillinen levyjärjestelmä. Koska fyysistä laitteistoa ei ollut saatavilla, päädyttiin käytännön toteutuksessa vaihtoehtoon, jossa insinöörityön käytännön osuutta ryhdyttiin rakentamaan Metropolian laboratorion VMware ESX-klusterin näennäispalvelimille, jotka saavat levynsä näennäisestä OpenFiler-levyjärjestelmästä iSCSI-levyinä.

6.2 Infrastruktuuri

Koska OpenStack Swift on Linux-käyttöjärjestelmän päällä ajettava sovellus, valittiin Swift- ja Keystone-palvelinten käyttöjärjestelmäksi hyvin tuettu ja yritysympäristöissäkin yleinen SuSe Linux Enterprise Server-jakelu. Versio, jota toteutuksessa käytettiin, oli SLES 11 SP3. Infrastruktuuriin kuului seuraava laitteistokokoonpanon:

- 1 kpl OpenFiler näennäislevyjärjestelmä
    - 2 vCPU
    - 8 GB muistia
    - 10 GB:n kiintolevy käyttöjärjestelmälle
    - 5 kpl 20 GB:n kiintolevyjä Swift levyjen käyttöön
- 1 kpl OpenStack Keystone tunnistuspalvelin
    - 1 vCPU
    - 4 GB muistia
    - 10 GB:n kiintolevy käyttöjärjestelmälle

- 1 kpl OpenStack Swift välityspalvelin
    - 1 vCPU
    - 4 GB muistia
    - 10 GB:n kiintolevy käyttöjärjestelmälle
- 4 kpl OpenStack Swift tallennussolmuja
    - 2 vCPU
    - 8 GB muistia
    - 10 GB:n kiintolevy käyttöjärjestelmälle
- 2 kpl asiakaskoneitta Swift-objektitallennuksen testaukseen
    - 1 kpl Microsoft Windows Server 2008 R2 Enterprise
        - 1 vCPU
        - 4 GB muistia
        - 40 GB:n kiintolevy käyttöjärjestelmälle
    - 1 kpl SuSe Linux Enterprise Server
        - 1 vCPU
        - 4 GB muistia
        - 40 GB:n kiintolevy käyttöjärjestelmälle



Kuva 8. Swift -verkkoinfrastruktuuri.

Koneet kytkettiin VMwaren virtuaalikytkinten avulla tietoverkkoihin kuvan 8 mukaisesti siten, että asiakaskoneet, välitys- ja tunnistuspalvelin liitettiin samaan julkiseen verkkoon, jolla mallinettiin kuvitteellista Internetiä ja siihen liittyviä Swift-toteutuksen komponentteja. Todellisessa tuotantototeutuksessa asiakaskoneiden ja välityspalvelimien välissä olisi luonnollisesti kuormantasauspalvelu sekä palomuurit. Välityspalvelin liittyi niin ikään Swiftin yksityiseen sisäverkkoon johon liitettiin myös tallennussolmut. Tallennussolmut liittyivät myös erilliseen tallennusverkkoon, johon myös OpenFiler levyjärjestelmä oli liitetty. Lisäksi kaikki SLES-koneet ja OpenFiler olivat liitetty erilliseen hallintaverkkoon, jotta koneita voitiin hallita etänä koulun verkosta. Palomuurista oli sallittu pääsy ulkoverkkoihin siten, että ssh-yhteys sekä yhteys SuSe:n ja OpenStackin ohjelmistopakettien latauspalveluihin oli sallittu.

6.3 OpenFiler-näennäislevyjärjestelmä

OpenFiler-näennäislevyjärjestelmän asennus VMware alustalle oli erittäin yksioikoinen toimenpide, se toimitetaan valmiina OVF (Open Virtualization Format)-formaatissa olevana näennäiskonemallina, jonka perusteella itse OpenFiler-näennäiskone luodaan. Kun näennäiskone käynnistetään ensimmäisen kerran, se käynnistyy suoraan asennusvelhoon, jolla OpenFilerin käyttöjärjestelmäkonfiguraatio suoritetaan. Asennusvelhossa määritellään muun muassa, mille massamuistilaitteille käyttöjärjestelmä asennetaan ja mitkä ovat verkko-, kieli- ja lokaatioasetukset. Määrittelyjen syötön jälkeen asennusvelho suorittaa itse asennuksen ja näennäispalvelin, jolle asennus tapahtui, uudelleenkäynnistyy asennuksen päätteeksi. Uudelleenkäynnistyksen jälkeen OpenFiler on valmis käytettäväksi.

Koska OpenFiler:lla haluttiin jäljitellä oikeata tallennusjärjestelmää, josta osoitetaan tallennustilaa Swift-tallennussolmujen käyttöön, sen tallennusjärjestelmäosuuden konfiguraatio tehtiin vastaamaan aitoa tallennusjärjestelmää. Kuten aidossakin tallennusjärjestelmässä, tallennuskapasiteetiksi varatuista massamuistilaitteista muodostettiin RAID-5-levyryhmä, josta kullekin tallennussolmulle lohkottiin omat LUN:nsa (Logical Unit Number) jotka osoitettiin iSCSI:n ylitse tallennussolmulle. Poiketen aidoista tallennusjärjestelmästä, joissa kaikki RAID levyryhmät asetetaan laitteiston asennus- tai laajennusvaiheissa erillisellä ohjaimella olevina laitteistopohjaisina toteutuksina, OpenFilerssa jouduttiin käyttämään sovelluspohjaista RAID-ohjainta.

RAID-levyryhmän luonti tapahtuu sangen yksinkertaisesti OpenFilerin Volumes – Block Devices-valikosta. Kunkin asiakaskoneiden käyttöön varatun 20 GB:n massamuistilaitteen tyypiksi asetetaan RAID Array Member-osio. Tämän jälkeen valitaan alasvetovalikosta, minkä tyyppinen RAID-levyryhmä halutaan luoda käyttöön sekä mitkä osiot kuuluvat luotavaan levyryhmään. Tässä vaiheessa valinnoiksi tulivat RAID-5 sekä kaikki osiot. Tämän jälkeen OpenFiler suorittaa tarvittavat konfiguraatiot ja levyryhmän luonti on valmis.

Kun haluttu fyysinen RAID-5-levyryhmä oli luotu, luotiin LUN:eja varten looginen levyryhmä, josta LUN:it luodaan. Tämä tapahtuu Volumes – Volume Groups-valikon työkaluilla siten, että ensin annetaan levyryhmän nimi ja valitaan, mitkä fyysiset levyryhmät tähän loogiseen levyryhmään liitetään. Tämän operaation jälkeen voitiin tästä loogisesta levyryhmästä luoda halutut LUN:it Swift-tallennussolmujen käyttöön. Tämä tapahtui Volumes – Add Volumes-valikon työkaluilla. Tallennussolmujen käyttöön luotiin neljä kappaletta 25 GB LUN:eja, joiden tyypiksi valittiin iSCSI.

Kun tarvittavat levykonfiguraatiot oli tehty, käynnistettiin Services – Manage Services-valikosta iSCSI-käynnistäjä ja iSCSI-kohde-palvelut. Kun palvelut olivat käynnistyneet, luotiin kutakin tallennussolmua varten oma iSCSI-kohde ja sen käyttöön varattu LUN liitettiin iSCSI-kohteeseen siten, että LUN:in tyypiksi tuli lohkopohjainen tiedonsiirto. Koska erillistä iSCSI tunnistusta ei haluttu käyttää, tunnistus jätettiin toimimaan ilman salasanoja sekä verkkoliikenne sallittiin koko 172.16.2.0/24-verkosta.

6.4 Keystone-todennuspalvelimen asennus

Keystone-identiteetti palvelun asennus tapahtui seuraamalla OpenStack installation guide for opensuse and suse linux Enterprise server – Havanna edition -manuaalin sekä SuSe Linux Administration Guiden ohjeistusta ja suosituksia. [34 & 35.]

Käyttöjärjestelmän asennus authserver-palvelimelle tapahtui SuSe-asennusmedian asennusvelhon avulla. Käyttöjärjestelmäasennuksessa käytettiin pääasiassa asennusvelhon suosittamia oletusasetuksia. Asennuksen yhteydessä määritettiin pääkäyttäjien salasana, palvelimen nimi, verkko- ja palomuuriasetukset sekä sijainti- ja kieliasetukset. Lisäksi palvelin rekisteröitiin SuSe:lle, jotta pystyttiin hakemaan viimeisimmät päivitykset, jota oli tarjolla. Authserver-palvelimelle määritettiin seuraavat asetukset:

- Palvelimen nimeksi määritettiin authserv.
- Palomuurin tilaksi määritettiin ”pois päältä”.
- Verkkoliitännöille annettiin seuraavat määritykset:
    - eth0 – hallintaverkon ip 10.94.154.196 / 24, gateway 10.94.154.255
    - eht1 – julkisen verkon ip 172.16.0.3 / 24
    - DNS-palvelimen osoitteeksi määritettiin 10.94.1.4
- NTP (Network Time Protocol)-palvelun osoitteeksi määritettiin koulun NTP palvelin 10.94.1.3

Kun käyttöjärjestelmä oli asennettu palvelimelle, asennettiin siihen VMware tools jotta palvelin pystyy paremmin hyödyntämään kaikkia VMwaren ominaisuuksia. Lisäksi ylläpidon helpottamiseksi asennettiin Nano-tekstieditori. Palvelimen hosts.conf-tiedostoa muokattiin siten, että Keystone palvelin pystyy selvittämään muiden palvelimien nimet ip-osoitteiksi. Kun perusasennus oli saatu valmiiksi, palvelimesta otettiin levykuva, jotta ongelmatilanteissa voitaisiin palata helposti alkutilanteeseen, jossa palvelimelle on asennettu vain käyttöjärjestelmä. Tämä osoittautui myöhemmässä vaiheessa oikeaksi ratkaisuksi, koska Keystonen konfiguraation jouduttiin todennusongelmien takia suorittamaan uudestaan.

6.4.1 Keystone-tietokannan asennus

Ennen Keystone-sovelluksen asennusta authserv-palvelimelle, palvelimelle asennettiin MySQL-tietokanta, MySQL-asiakassovellus sekä tarvittavat Python-kirjastot OpenStack-ohjeistuksen mukaisesti. Näiden asennus tapahtuu komentokehotteessa komennolla *zypper install mysql-client mysql python-mysql*, joka asentaa kaikki tarvittavat paketit. Tämän jälkeen MySQL:n konfiguraatiotiedostoon /etc/my.cnf määritettiin ip-osoite, johon MySQL-demoni kiinnittyy. Tämän jälkeen voitiin käynnistää MySQL-palvelu sekä asettaa tietokanta käynnistymään automaattisesti, kun palvelin käynnistyy. Tämä tapahtui komennoilla *service mysql start* ja *chkconfig mysql on*.

Itse MySQL:n konfiguraatio aloitetaan komennolla *mysql_install_db*. Komento määrittää tarvittavat MySQL:n järjestelmätaulut ja valmistelee MySQL-asennuksen konfiguraatiota varten. Tämän jälkeen voidaan suorittaa itse MySQL:n konfiguraatio komennolla *mysql_secure_installation*. Komento käynnistää asennusohjelman, jonka aikana määritellään pääkäyttäjän salasana, suoritetaan tietokannan turvallisuutta parantava anonymous-käyttäjän tunnuksen ja oikeuksien poisto sekä määritellään, mistä verkkoliitännöistä sallitaan pääkäyttäjätason kirjautuminen tietokantaan. Oletusarvoisesti asennusohjelma ja käytössä ollut ohjeistus ohjeisti, sallimaan pääkäyttäjätason kirjautumisen vain localhost:lta. Ensimmäisellä MySQL:n asennuskerralla noudatettiinkin ohjeistusta, tämä aiheutti tilanteen, jossa Keystone-todennus ei toiminut halutulla tavalla. Tästä johtuen koko Keystone sovelluksen ja MySQL:n asennus jouduttiin tekemään uudelleen. Jotta Keystonen todennus toimisi, kuten halutaan, on tietokanan pääkäyttäjälle sallittava kirjautuminen etänä tietokantaan, myös muista verkkoliitännöistä kuin localhostista. Muilta osin voidaan käyttää asennusohjelmassa ja ohjeistuksessa suositeltuja oletusasetuksia.

6.4.2 Keystone-sovelluksen konfigurointi

Jotta OpenStack Keystonen asennuspaketit voidaan hakea verkosta, on käyttöjärjestelmälle määritettävä OpenStack tietolähteiden verkko-osoite, tämä tapahtuu komennolla *zypper addrepo -f obs:Cloud:OpenStack:Havana/SLE_11_SP3 Havana*. Komennon suorituksen jälkeen on syytä päivittää tietolähteiden hakemistot komennoilla *zypper refresh* ja *zypper update*, sekä uudelleenkäynnistää palvelin. Uudelleenkäynnistyksen jälkeen, on syytä testata uuden tietolähteen asetus ja toiminta asentamalla openstack-utils-paketti, joka sisältää OpenStack-sovellusten asennuksessa ja konfiguroinnissa käytettäviä työkaluja.

Itse OpenStack Keystone identiteettipalvelun asennus käynnistyy asentamalla tarvittavat Keystone- ja Python-paketit komennolla *zypper install openstack-keystone python-keystoneclient*. Kun paketit ovat asentuneet, on ensimmäiseksi määritettävä Keystonen konfiguraatiotiedostoihin tietokantayhteyden tiedot, käyttäjätunnus ja salasana, tämä tapahtuu komennolla

> *openstack-config --set /etc/keystone/keystone.conf sql connection mysql://keystone:KEYSTONE_DBPASS@authserv/keystone*

Openstack-config on yksi openstack-utils-paketin mukana asentuvista työkaluista, joilla on mahdollista tehdä komentoriviltä muutoksia konfiguraatiotiedostoihin. Komennon ensimmäinen parametri *--set* määrittää konfiguraatiotiedoston, jota halutaan muokata, sijainnin. Seuraava parametri sql *connection* on konfiguraatiotiedostossa oleva parametrikenttä, jota halutaan muokata, sekä parametrin uusi arvo. Parametrin arvo on tässä tapauksessa URL (Uniform Resource Location), joka muodostuu osoitteen kohteena olevana palvelun tyypistä *mysql*, käyttäjätunnuksesta *keystone*, käyttäjätunnuksen salasanasta *KEYSTONE_DBPASS* ja kohteesta, eli *@authserv/keystone*. Arvo KEYSTONE_DBPASS on salasana, jota Keystone-palvelu käyttää ottaessaan yhteyden tietokantaan, joten sitä generoidessa on hyvä pitää mielessä salasanojen luonnin parhaat käytännöt, eli salasanan on sisällettävä pieniä ja isoja kirjaimia, numeroita sekä erikoismerkkejä. Salasanan on myös oltava riittävän pitkä, esimerkiksi 16 merkkiä ja eikä se saa sisältää sanakirjoista löytyviä sanoja tai tunnettuja nimiä. Kun edellä mainittu tietokantapalvelun parametri on asetettu konfiguraatiotiedostoon, voidaan MySQL-tietokantaan luoda Keystonen tietokanta ja sen taulut. Tämä tapahtuu openstack-utils-työkalulla openstack-db-komennolla o*penstack-db --init --service keystone --password KEYSTONE_DBPASS*.

Tietokannan luonnin jälkeen määritellään Keystonen konfiguraatiotiedostoon Keystone-todennuspalvelun ja muiden OpenStack-palvelujen jaettu valtuutusavain. Valtuutusavaimen voi luoda ohjeistuksen mukaan openssl:lä tai jollain muulla vastaavalla satunnaislukugeneraattorilla. Avainta luotaessa on muistettava, että eri palvelut käyttävät sitä toistensa valtuutuksen tunnistamiseen, joten valtuutusavaimen on syytä olla riittävän pitkä ja vaikea sekä noudattaa hyvän salasanan parhaita käytäntöjä. Koska kyseessä oleva valtuutusavain tallennetaan eri palvelujen konfiguraatiotiedostoihin, eli sitä ei ole jatkossa missään vaiheessa tarvetta muistaa ulkoa tai kirjoittaa komentokehotteeseen, olisi tuotantoympäristöissä syytä käyttää huomattavasti pidempää valtuutusavainta, kuin ohjeen mainitsema 10:tä merkkiä. Valtuutusavain lisätään Keystonen konfiguraatiotiedostoon openstack-config-työkalun avulla komennolla *openstack-config --set /etc/keystone/keystone.conf DEFAULT admin_token ADMIN_TOKEN*.

Keystone käyttää PKI (Public Key Infrastructure) -avaimia, joten sitä varten on luotava tarvittavat allekirjoitusavaimet, sertifikaatit, lisäksi Keystone-palvelu on vaihdettava omistajaksi sen käyttämiin hakemistoihin. PKI-avainten asetus tapahtuu komennolla *keystone-manage pki_setup --keystone-user openstack-keystone --keystone-group openstack-keystone* ja Keystonen käyttämien kansioiden /etc/keystone- sekä

/var/log/keystone/-omistajuuden muutos komennolla *chown -R openstack-keystone:openstack-keystone /etc/keystone/* /var/log/keystone/**. Kun kaikki edellä mainitut konfiguraatiot on tehty, voidaan Keystone-palvelu käynnistää ja asettaa käynnistymään automaattisesti uudelleen käynnistyksen yhteydessä komennoilla *service openstack-keystone start* ja *chkconfig openstack-keystone on*.

6.4.3 Keystone-käyttäjä- ja palvelutunnusten määrittelyt

Koska insinöörityössä käytetty Keystone-asennus on tarkoitettu vain paikallisen Swift-ympäristön käyttöön ja sen integrointi hakemistopalveluihin on insinöörityön alueen ulkopuolella, käyttäjä- ja palvelutunnusten luonti tehtiin paikallisesti Keystone palvelimen tietokantaan. Koska Keystone-todennuspalvelulle ei ole vielä tässä vaiheessa määritelty hallintatunnusta, on palvelun konfiguraatio tässä vaiheessa tehtävä käyttäen aikaisemmin määriteltyä jaettua valtuutusavainta. Kaikissa Keystone-komennoissa on mahdollista antaa parametreina tarvittavat valtuutusavain ja palvelun päätepiste tiedot. Jotta tietoja ei tarvitsisi syöttää jokaisen komennon yhteydessä, on kuitenkin järkevintä tallettaa tiedot tilapäisesti komentokehotteen ympäristömuuttujiksi. Jos valtuutusavain ja palvelun päätepisteen tietoja ei anneta parametreina, Keystone hakee tietoja OS_SERVICE_TOKEN- ja OS_SERVICE_ENDPOINT-nimisistä ympäristömuuttujista.

Ensimmäiseksi on luotava Keystonen pääkäyttäjän asiakas, käyttäjä sekä rooli. Ajan säästämiseksi samalla luotiin myös Swiftin pääkäyttäjä asiakas, käyttäjä sekä rooli. Roolia luotaessa on huomattava, että rooli on Keystonessa vain tekstirivi. Se mitä oikeuksia kullakin roolilla on palveluun, määritellään erillisesti kunkin palvelun policy.json tiedostossa. Koska molemmissa, sekä Keystone että Swift palveluissa on rooli *admin*, joten samaa roolia voitiin hyödyntää molemmissa palveluissa. Esimerkiksi Keystonen pääkäyttäjä-asiakkuuden luonti tapahtuu komennolla *Keystone tenant-create --name=admin --description="Admin Tenant"*. Kaikki Keystone-asiakkaan, käyttäjän ja roolin luontikomennot palauttavat tulosteen, jossa esitetään luodun tietueen kuvaus, status ja tunnistenumero (id) sekä nimi. Keystone-pääkäyttäjäasiakkuuden luonnin tuloste on kuvassa 9, loput tässä vaiheessa annetuista komennoista ja niiden tulostamista tiedoista löytyvät liitteenä (liite 1.).

Property	Value
description	Admin Tenant
enabled	True
id	4333601479d94e339ed4b7ec2228b704
name	admin

Kuva 9. keystone tenant-create –name=admin –description=”Admin Tenant”-tuloste.

Merkittävin tieto tulosteessa on tunnistenumero (id). Tämä satunnaisesti generoitava numero luodaan jokaiselle asiakkaalle, käyttäjälle, roolille ja palvelun päätepisteelle. Kun myöhemmässä vaiheessa, liitetään yhteen eri asiakkuuksia, käyttäjiä, rooleja ja palvelujen päätepisteitä, tunnistenumeroa käytetään määritellyissä kunkin asiakkuuden, käyttäjän, roolin ja palvelun päätepisteen tunnisteena niille annettujen nimien sijasta. Esimerkiksi kun Keystonen identiteettipalvelun pääkäyttäjän tunnukseen liitetään asiakkuus sekä käyttäjälle rooli, komento oli muotoa

keystone user-role-add --user=d8332a64afdf4f2092e34e2cc79d8b2e \

--tenant=4333601479d94e339ed4b7ec2228b704 \

--role=91efc7229211415797fc5c2edcdd0cb1

Jotta Keystone tietää, mistä verkko-osoitteista eri palvelut löytyvät ja osaa vastata asiakkaiden, eri palveluiden päätepistettä koskeviin kyselyihin, on kutakin Keystonea käyttävälle palvelulle luotava rekisteröinnit Keystoneen. Rekisteröintiä varten on kullekin palvelulle luotava palvelukuvaus sekä palvelun API:en päätepisteiden rekisteröinti. API-päätepisteitä rekisteröidessä, määritetään URL:t palvelun julkiselle API:lle, sisäiselle API:lle ja hallinta-API:lle. Esimerkiksi Swift-palvelun rekisteröinti Keystoneen tapahtui komennolla *keystone service-create --name=swift --type=object-store --description="Swift Object Storage Service"*. Komento antaa tulosteen joka esiteltiin aikaisemmin asiakkaita, käyttäjiä ja rooleja luodessa. Tulosteessa olevaa *service-id*-tunnusnumeroa käyttäen luodaan API-päätepiste rekisteröintikomennolla,

Keystone endpoint-create \

--service-id=a48897e2c2e44cde80e3c08657172e85 \

--publicurl="http://proxyserv:8080/v1/AUTH_$(tenant_id)s" \

--internalurl="http://proxyserv:8080/v1" \

--adminurl=http://proxyserv:8080/v1/AUTH_\$(tenant_id)s

Tarkemmat tiedot Keystone- ja Swift-palvelujen rekisteröinnin komennoista ja niiden tulosteista löytyvät liitteenä (liite 2). Tässä vaiheessa, kun sekä Keystone että Swift palveluille oli määritelty asiakkaat, käyttäjät, käyttäjäroolit sekä palvelutunnukset, testattiin Keystone toimivuus komennolla

keystone --os-username=admin --os-password='EL1E$Ta!nS+ooM' \

--os-tenant-name=admin \

--os-auth-url=http://authserv:35357/v2.0 token-get

Komennon palautteena Keystone todennuspalvelu palautti tälle tunnukselle luomansa valtuutusavaimen, jolla tunnus jatkossa voi suorittaa toimenpiteitä Keystone palveluun.


```
+-----------+-----------------------------------------------------------------------------------------------------------+
|  Property |                                                Value                                                      |
+-----------+-----------------------------------------------------------------------------------------------------------+
|  expires  | 2014-02-04T17:28:35Z                                                                                      |
|     id    | MIIF6QYJKoZIhvcNAQcCoIIF2jCCBdYCAQExCTAHBgUrDgMCGjCCBD8GCSqGSIb3DQEHAaCCBDAEggQseyJhY2Nlc3MiOiB7InRva2 |
|           | VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi0wM1QxNzoyODozNS4yNjcwNzUiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTA0VDE3OjI4 |
|           | OjM1WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIkFkbWluIFRlbmFudCIsICJlbmFibG |
|           | VkIjogdHJ1ZSwgImlkIjogIjQzMzM2MDE0NzlkOTRlMzM5ZWQOYjdlYzIyMjhiNzAOIiwgIm5hbWUiOiAiYWRtaW4ifXOsICJzZXJ2 |
|           | aWN1Q2FOYWxvZyI6IFt7ImVuZHBvaW50cyI6IFt7ImFkbWluVVJMIjogImhOdHA6Ly9wcm94eXNlcnY6ODA4MC92MS9BVVRIX3MiLC |
|           | AicmVnaW9uIjogInJlZ2lvbk9uZSIsICJpbnRlcm5hbFVSTCI6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEiLCAiaWQiOiAiMjkO |
|           | MTgzZDhmNGRkNDk5ZDkwZWU1MGVjM2Y4ODVmMzMiLCAicHVibGljVVJMIjogImh0dHA6Ly9wcm94eXNlcnY6ODA4MC92MS9BVVRIXz |
|           | QzMzM2MDEONzlkOTRlMzM5ZWQOYjdlYzIyMjhiNzAOIn1dLCAiZW5kcG9pbnRzX2xpbmtzIjogW1OsICJOeXBlIjogIm9iamVjdC1z |
|           | dG9yZSIsICJuYW1lIjogInN3aWZOInOsIHsiZW5kcG9pbnRzIjogW3siYWRtaW5VUkwiOiAiaHROcDovL2F1dGhzZXJ2OjM1MzU3L3 |
|           | YyLjAiLCAicmVnaW9uIjogInJlZ2lvbk9uZSIsICJpbnRlcm5hbFVSTCI6ICJodHRwOi8vYXVOaHNlcnY6NTAwMC92Mi4wIiwgImlk |
|           | IjogIjI5MmUzZjc3YzkwOTRiZmI4Yjg5NTFjZjVmNzU4ZDBjIiwgInB1YmxpY1VSTCI6ICJodHRwOi8vYXVOaHNlcnY6NTAwMC92Mi |
|           | 4wIn1dLCAiZW5kcG9pbnRzX2xpbmtzIjogW1OsICJOeXBlIjogImlkZW5OaXR5IiwgIm5hbWUiOiAia2V5c3RvbmUifVOsICJ1c2Vy |
|           | IjogeyJ1c2VybmFtZSI6ICJhZG1pbiIsICJyb2xlc19saW5rcyI6IFtdLCAiaWQiOiAiZDgzMzJhNjRhZmRmNGYyMDkyZTMOZTJjYz |
|           | c5ZDhiMmUiLCAicm9sZXMiOiBbeyJuYW1lIjogImFkbWluIn1dLCAibmFtZSI6ICJhZG1pbiJ9LCAibWVOYWRhdGEiOiB7ImlzX2Fk |
|           | bWluIjogMCwgInJvbGVzIjogWyI5MWVmYzcyMjkyMTEOMTU3OTdmYzVjMmVkY2RkMGNiMSJdfX19MYIBgTCCAXOCAQEwXDBXMQswCQ |
|           | YDVQQGEwJVUzEOMAwGA1UECAwFVW5zZXQxDjAMBgNVBAcMBVVuc2VOMQ4wDAYDVQQKDAVVbnNldEYMBYGA1UEAwwPd3d3LmV4YW1w |
|           | bGUuY29tAgEBMAcGBSsOAwIaMAOGCSqGSIb3DQEBAQUABIIBAAeiFcejw6LUzh+gCQVu8ylYlsCW4pluLOw7pInEP4HLgp31-TcGJB |
|           | 6t+sA5B-ZFOAnDHbbbTraccWHtoGBEF4zs+d8a1W3O1kaucNhBCWUPI9wyFj9xVxNj6cEQRj9a3g-b2GCLygZqhC7cPvH2YMaDERSa |
|           | Y6YFVCgb7nEOOr6Fq48sJU7BHd35K9eAlWsGNP9i7oZF+IjdoJ7p125OdrUJnSQlo2OzmpfOYOvPgXGjsBLV9kt-XOIIv7bPy2qxup |
|           | NZdkQXfimtmOYOrxHa1BI8JRFVMYGOZRkJG84fW5mAABC6kjtq+XWwigUjSkEEedBOhfCcgWZFK8l5zvFeh1o=                |
| tenant_id | 4333601479d94e339ed4b7ec2228b704                                                                          |
|  user_id  | d8332a64afdf4f2092e34e2cc79d8b2e                                                                          |
+-----------+-----------------------------------------------------------------------------------------------------------+
```


Kuva 10. Keystone-valtuutusavain.

Kuten kuvan 10 kuvakaappauksesta näkyy, saadun valtuutusavaimeen tietoihin kuuluvat kentät ovat voimassaolo, valtuutusavain, asiakkaan numeerinen tunniste ja käyttäjän numeerinen tunniste.

6.5 Swift-välitys- ja tallennussolmupalvelimien asennus

6.5.1 Palvelimien perusasetukset

Välitys- ja tallennussolmupalvelimien asennus tapahtui seuraten samoja ohjeistuksia, suosituksia ja työtapoja kuin Keystone-palvelimenkin kanssa. Palvelimille määritettiin seuraavat asetukset [34 & 35.]:

- Välityssolmupalvelimen nimeksi määritettiin proxyserv
    - sen verkkoliitännöille annettiin seuraavat määritykset:
        - eth0 – hallintaverkon ip 10.94.154.197/24, gateway 10.94.154.255
        - eht1 – julkisen verkon ip 172.16.0.4/24
        - eht2 – sisäisen verkon ip 172.16.1.1/24
    - NTP (Network Time Protocol)-palvelun osoitteeksi määritettiin koulun NTP palvelin 10.94.1.3
- Tallennussolmupalvelimien nimiksi määritettiin StorageServ01 - StorageServ05.
    - Niiden verkkoliitännöille annettiin seuraavat määritykset:
        - eth0 – hallintaverkon ip 10.94.154.198-10.94.154.202/24, gateway 10.94.154.255
        - eht1 – sisäverkon ip 172.16.1.2–172.16.1.6/24
        - eht2 – storageverkon ip 172.16.2.2–172.16.2.6/24
- NTP-palvelun osoitteeksi määritettiin proxyserv NTP palvelu 172.16.1.1.

Kun käyttöjärjestelmän perusasennus oli suoritettu, välityssolmupalvelimelle asennettiin ntpd-demoni ja palvelin konfiguroitiin Swiftin sisäverkon NTP-palvelimeksi ja muut palvelimet määritettiin käyttämään sitä NTP-palvelimena. NTP-palvelun konfiguraatio tapahtui tiedoston */etc/ntp.conf* avulla. Lisäksi palvelimille asennettiin VMware tools, jotta palvelin pystyy paremmin hyödyntämään kaikkia VMwaren ominaisuuksia. Ylläpidon helpottamiseksi palvelimille asennettiin myös Nano-tekstieditori. Kun perusasennukset oli saatu valmiiksi, kustakin palvelimesta otettiin levykuva, jotta ongelmatilanteissa voitaisiin palata helposti alkutilanteeseen, jossa palvelimella on asennettuna vain käyttöjärjestelmä ja peruskonfiguraatio.

Kuten Keystonea asennettaessa ensimmäisenä Swiftin asennustoimenpiteenä palvelimille määriteltiin OpenStack-tietolähteiden verkko-osoite ja suoritettiin tietolähteen lisäyksen vaatimat toimenpiteet. Kun tämä oli suoritettu, palvelimille tuotiin ja asennettiin Swiftin ydintiedostot ja OpenSSH:n vaatimat paketit. Tämä tapahtui komennolla

zypper install openstack-swift openstack-swift-proxy \

openstack-swift-account openstack-swift-container \

openstack-swift-object memcached

Kun Swiftin ydintiedostot oli asennettu palvelimille, luotiin hakemisto Swift:iä varten /etc hakemistoon. Hakemiston luonti tapahtui komennolla *mkdir – p /ect/swift*. Swift-palvelun paikallinen käyttäjätunnus ja ryhmä määritettiin hakemiston omistajaksi komennolla *chown -R openstack-swift:openstack-swift /etc/swift/*. Tämän jälkeen luotiin ensimmäinen Swift-konfiguraatio tiedosto swift.conf hakemistoon /etc/swift ja määriteltiin *swift_hash_path_suffix* Python Paste Deploy-parametri tiedostoon, parametriin määriteltiin satunnaislukugeneraattorilla teksti. Swift käyttää tässä parametrissä määriteltyä tekstiä hajautinalgoritmin suolana, kun se määrittää tiedon sijoittelua Swift-ryppäässä. Tekstiä määrittäessä on muistettava, että sen on oltava sama kaikilla Swift-solmuilla ja eikä sitä voi myöhemmin muuttaa.

6.5.2 Tallennussolmujen konfiguraatio

Tallennus-solmujen konfiguraatio oli hyvin yksinkertainen toimenpide. Kullekin solmulle asennettiin ja konfiguroitiin iSCSI-käynnistäjä. Asennettiin Swiftin tarvitsemat asennuspaketit. Massamuistilaitteelle, joka on tarkoitettu tiedon tallennukseen, alustetaan XFS-tiedostojärjestelmä, lisäksi suoritettiin rsync:in konfigurointi ja konfiguroitiin account-server-, container-server- ja object-server-palvelut.

iSCSI-käynnistäjän asennus tapahtui käynnistämällä YaST (Yet Another Setup Tool)-asennustyökalu, valitsemalla sen Network Services valikosta iSCSI Initiator-työkalu, jolloin asennustyökalu asentaa tarvittavat iSCSI-paketit automaattisesti. iSCSI:n konfigurointi tapahtui niin ikään YaST:in avulla Network Services / iSCSI Initiator-valikosta. Asennustyökalussa määritettiin palvelun käynnistymisasetukset, määritettiin iSCSI-kohde, joka oli luotu palvelinta varten OpenFiler-levyjärjestelmään sekä määriteltiin miten, palvelin liittää levyn uudelleenkäynnistyksen yhteydessä. Tämän jälkeen tarkistettiin YaST:in System valikon Partitioner-työkalusta, että haluttu levy oli tullut järjestelmän käyttöön.

Swift tallennussolmun tarvitsemien asennuspakettien asennus tapahtui komennolla

zypper install openstack-swift-account openstack-swift-container \

openstack-swift-object xfsprogs xinetd

Komento asentaa Swift account-server-, container-server- ja object-server-palvelujen tarvitsemat asennuspaketit, xfsprogs-asennuspaketin, jota tarvitaan XFS-tiedostojärjestämän konfigurointiin ja hallintaan sekä xinetd (extended Internet daemon)-demonin asennuspaketit. Xinetd-demoni kuuntelee ja välittää verkosta Swift palveluille tulevat pyynnöt.

Kiintolevyjen osiointi ja alustus XFS-tiedostojärjestelmälle käynnistyy komennolla *mkfs.xfs /dev/sdb1*, joka käynnistää asennusvelhon, jossa tarvittavat määritykset annetaan. Komennon viimeinen osa (tällä kertaa sdb1) kertoo, mihin fyysiseen massamuistilaiteeseen komento kulloinkin kohdistuu. Koska tässä laitteistossa ei ollut kuin yksi massamuistilaite, voitiin asennusvelho ajaa vain kerran. Mikäli massamuistilaitteita olisi ollut useampia, asennusvelho olisi pitänyt suorittaa kaikille. Kun massamuistilaite oli osioitu, luotiin hakemisto, johon se kiinnitetään tiedostojärjestelmässä /srv/node-

hakemistoon. Hakemiston luonti tapahtui komennolla *mkdir -p /srv/node/sdb1* ja laitteen kiinnitys hakemistoon komennolla *mount /srv/node/sdb1*. Lisäksi /etc/fstab asetustiedostoon lisättiin rivi */dev/sdb1 /srv/node/sdb1 xfs noatime, nodiratime, nobarrier, logbufs=8 0 0,* jotta laite ja sen tiedostojärjestelmä liitetään käynnistyksen yhteydessä oikein hakemistopuuhun. Swift-käyttäjätunnus openstack-swift vaihdettiin omistajaksi /srv/node-hakemistoon.

Swift käyttää rsync:iä tiedon replikointiin eri tallennussolmujen välillä, replikointia varten tehtiin tarvittavat konfiguraatiot */etc/rsyncd.conf* tiedostoon. Lisäksi account-server-, container-server- ja object-server-palvelujen konfiguraatiotiedostoihin */etc/swift/account-server.conf, /etc/swift/container-server.conf, /etc/swift/object-server.conf* lisättiin tarvittavat tiedot, eli ip sekä portti, jota palvelu kuuntelee, palvelun käyttäjätunnus, palvelun hakemistot, lokipalvelun tiedot sekä määrittelyt sille, että palvelu tarkistaa käynnistyessään käytettävien massamuistilaitteiden tilan. Konfiguraatiotiedostot ovat tämän insinöörityön liitteenä (liitteet 3 – 5). Lopuksi varmistettiin, että openstack-swift-käyttäjätunnus sekä ryhmä ovat kaikkien konfiguraatiotiedostojen omistajana.

6.5.3 Välityssolmun konfiguraatio

Välityssolmua asennettaessa ja konfiguroitaessa sille asennettiin palvelu, jota se tarvitsee käsitellessään Swift-asiakkailta ja tallennussolmuilta tulevat pyynnöt, luotiin palvelun konfiguraatiotiedosto, generoitiin tarvittavat SSL (Secure Socket Layer) sertifikaatit sekä luotiin account, container ja object renkaiden Python määrittely tiedostot.

Swift välityspalvelun asennuspakettien haku ja asennus tapahtui komennolla

*zypper install openstack-swift-proxy memcached openstack-utils *

python-swiftclient python-keystoneclient

Komento asentaa välityspalvelun tarvitsemat asennuspaketit, memcached hajautetun muistinkäyttöpalvelun asennuspaketit, OpenStack asennus- ja konfigurointityökalun asennuspaketit. Lisäksi asennettiin Swiftin ja Keystonen integraatiossa tarvittava Python-pohjainen Keystone asiakassovellus ja järjestelmän testausta varten tarvittava Swift asiakassovellus.

Ennen varsinaisen Swift-välityspalvelun konfigurointia, luotiin tarvittavat SSL-sertifikaatit OpenSSL-ohjelmalla, asennusohjeen mukaisesti luotiin X.509 standardiin perustuva sertifikaatti /etc/swift-hakemistoon. Lisäksi memcached-palvelun konfiguroititiedostoa */etc/memcached.conf* muokattiin siten, että palvelu kuuntelee Swift-sisäverkon ip-osoitetta 172.16.1.1.

Itse Swift välityspalvelun konfigurointi aloitettiin luomalla sen konfigurointitiedosto *etc/swift/proxy-server.conf* ja määrittelemällä siihen palvelun parametrit. Konfigurointitiedostoon määriteltiin palvelun SSL-sertifikaattien sijainti, lokiasetukset, Keystone identiteettipalvelun määritykset, palvelun käyttäjätunnus sekä portti, jota palvelu kuuntelee. Konfigurointitiedosto on tämän insinöörityön liitteenä (liite 6).

Kun itse välityspalvelu on konfiguroitu, voidaan luoda tili-, säiliö- ja objektirenkaiden luontitiedosto. Tiedoston luonti tapahtuu *swift-ring-builder*-työkalun *create*-komennolla. Tiedostoa luotaessa määritellään renkaan luontitiedoston nimi (account-, container- tai object.builder), se kuinka monta osiointia renkaaseen luodaan (luku annetaan kahden potenssina), yksittäisestä osiosta olevien kopioiden määrä sekä minimiaika osioiden uudelleenjärjestelylle. Luotavien osioiden määrä annetaan kahden potensseina siten, että luku joka saadaan kaavasta

> *100 * (massamuistilaitteiden maksimi lukumäärä) * (luontihetkellä käytössä olevien massamuistilaitteiden lukumäärä)*

pyöristetään ylöspäin lähimpään kahden potenssina saatavaan kokonaislukuun. Koska käytössä oli viisi massamuistilaitetta ja päätin, että 10 massamuistilaitetta olisi massamuistilaitteiden maksimi määrä, laskentakaavaksi tuli 100 * 10 * 5, tulokseksi saatiin 500. Koska 512 (2^9) on ylöspäin pyöristettäessä lähin kahden potenssin kokonaisluku, tuli luotavien partioiden määräksi 512. Koska asennus haluttiin muilta osin tehdä oletusarvoilla, osioiden kopioiden määräksi tuli kolme ja minimiajaksi osioiden uudelleenjärjestelyjen välillä yksi tunti. Näin ollen renkaiden luontitiedostojen luontikomennoiksi tulivat

- swift-ring-builder account.builder create 9 3 1
- swift-ring-builder container.builder create 9 3 1
- swift-ring-builder object.builder create 9 3 1

Kun renkaan luontitiedostot on luotu, niihin määritellään tallennus-solmuissa olevat massamuistilaitteet, tämä tapahtuu *swift-ring-builder*-työkalun *add*-komennolla. Kutakin tiedostoon määriteltävää massamuistilaitetta varten määritellään alue ja vyöhyke, mihin ne kuuluvat, ip ja portti jota laite käyttää tietoliikenteeseen sekä ip ja portti, jota laite käyttää kopiointiin. Lisäksi kullekin laitteelle määritellään painoarvo. Painoarvo määrittää, kuinka suuri osa renkaan osioista tullaan sijoittamaan kyseiselle massamuistilaitteelle. Tämä mahdollistaa erikoisten massamuistilaitteiden käytön renkaassa. Alla on esimerkki StorageServ01:n massamuistilaitteen lisäyksestä tilirenkaan luontitiedostoon, täydellinen lista komennoista on liitteenä (liite 7).

swift-ring-builder account.builder and z1-172.16.1.2:6002R172.16.2.2:6005/sdb1 100

Kun kaikki massamuistilaitteet on lisätty renkaiden luontitiedostoihin, voi komennolla *swift-ring-builder <luontitiedoston_nimi>* tarkistaa miten massamuistilaitteet on määritelty tiedostoon. Kuvassa 11 komennon *swift-ring-builder account.builder* tuloste.

```
jakke@proxyserv:swift-ring-builder account.builder
#account.builder, build version 7
#512 partitions, 3.000000 replicas, 1 regions, 4 zones, 4 devices, 100.00 balance
#The minimum number of hours before a partition can be reassigned is 1
#Devices:    id  region  zone      ip address  port  replication ip  replication port
             0       1     1      172.16.1.2  6002      172.16.2.2              6005
             1       1     2      172.16.1.3  6002      172.16.2.3              6005
             2       1     3      172.16.1.4  6002      172.16.2.4              6005
             3       1     4      172.16.1.5  6002      172.16.2.5              6005
             4       1     5      172.16.1.6  6002      172.16.2.6              6005
```

Kuva 11. account.builder-tiedosto.

Tämän jälkeen luodaan varsinainen rengastiedosto *swift-ring-builder*-työkalun *rebuild*-komennolla. Rebuild-komento määrittää, kuinka monta osiota sijoitetaan millekin kiintolevylle, ja suorittaa samalla write_ring-komennon, jolla kirjoitetaan varsinainen rengastiedosto. Luotavat rengastiedostot ovat

- account.ring.gz
- container.ring.gz
- object.ring.gz

Rebuild-komennon jälkeen on komennolla *swift-ring-builder <luontitiedoston_nimi>* mahdollista tarkistaa, miten osiot sijoitettiin eri massamuistilaitteille, kuvassa 12 on esimerkkinä komennon *swift-ring-builder account.builder*-tuloste.

```
jakke@proxyserv:/etc/swift$ sudo swift-ring-builder account.builder
account.builder, build version 5
512 partitions, 3.000000 replicas, 1 regions, 5 zones, 5 devices, 0.26 balance
The minimum number of hours before a partition can be reassigned is 1
Devices:    id  region  zone      ip address  port  replication ip  replication port      name weight partitions balance meta
             0       1     1      172.16.1.2  6002      172.16.2.2              6005      sdb1 100.00        307   -0.07
             1       1     2      172.16.1.3  6002      172.16.2.3              6005      sdb1 100.00        307   -0.07
             2       1     3      172.16.1.4  6002      172.16.2.4              6005      sdb1 100.00        308    0.26
             3       1     4      172.16.1.5  6002      172.16.2.5              6005      sdb1 100.00        307   -0.07
             4       1     5      172.16.1.6  6002      172.16.2.6              6005      sdb1 100.00        307   -0.07
```

Kuva 12. account.builder-tiedosto balansoinnin jälkeen.

Tämän jälkeen jäljellä on enää rengastiedostojen kopiointi kaikille solmuille */etc/swift*-hakemistoon, tiedostojen omistajuuden vaihto openstack-swiftlle ja palvelujen käynnistys *swift-init <palvelun_nimi> start*-komennolla.

- välityssolmu palvelimella
    - välityspalvelu *swift-init proxy-server start* komennolla
- tallennussolmu palvelimilta
    - container-updater-palvelu *swift-init container-updater start*
    - container-replicator-palvelu *swift-init container-replicator start*
    - container-auditor-palvelu *swift-init container-auditor start*
    - container-server-palvelu *swift-init container-server start*
    - container-sync-palvelu *swift-init container-sync start*
    - account-auditor-palvelu *swift-init account-auditor start*
    - account-server-palvelu *swift-init account-server start*
    - account-reaper-palvelu swift-init *account-reaper start*
    - account-replicator-palvelu *swift-init account-replicator start*
    - object-replicator-palvelu *swift-init object-replicator start*
    - object-auditor-palvelu *swift-init object-auditor start*
    - object-updater-palvelu *swift-init object-updater start*
    - object-server-palvelu *swift-init object-server start*

Komentojen suorituksen jälkeen on hyvä tarkistaa, että palvelut ovat todellakin käynnistyneet. Tämä tapahtuu komennolla *swift-init all status*, kuvassa 13 on esimerkkituloste komennosta StorageServ01-palvelimelta.

```
jakke@StorageServ01:~$ swift-init all status
container-updater running (7254 - /etc/swift/container-server.conf)
account-auditor running (1219 - /etc/swift/account-server.conf)
object-replicator running (7255 - /etc/swift/object-server.conf)
No proxy-server running
container-replicator running (7256 - /etc/swift/container-server.conf)
object-auditor running (1261 - /etc/swift/object-server.conf)
No object-expirer running
container-auditor running (1255 - /etc/swift/container-server.conf)
container-server running (7203 - /etc/swift/container-server.conf)
account-server running (7216 - /etc/swift/account-server.conf)
account-reaper running (1240 - /etc/swift/account-server.conf)
container-sync running (7257 - /etc/swift/container-server.conf)
account-replicator running (7258 - /etc/swift/account-server.conf)
object-updater running (7259 - /etc/swift/object-server.conf)
object-server running (7229 - /etc/swift/object-server.conf)
```

Kuva 13. Käynnissä olevat Swift-palvelut tallennussolmulla.

6.6 Ympäristön testaus

Ympäristön asennuksen jälkeen haluttiin testata ympäristön toimivuus. Ensimmäinen testi päätettiin suorittaa välityssolmupalvelimelta, jotta mahdolliset yhteysongelma asiakkaiden ja ympäristön välillä eivät vaikuttaisi testiin. Testaukseen käytettiin Swift-komentorivipohjaista asiakastyökalun, joka perustuu Swiftin natiivin *client.py*-kirjaston komentoihin. Asiakastyökalulla on mahdollista suorittaa seuraavia toimenpiteitä Swift objektitallennukseen:

- listata käyttäjän tallennusobjekteja
- luoda uusia säiliö-objekteja
- lisätä tallennusobjekteja käyttäjän säiliö-objekteihin
- ladata tallennusobjekteja
- poistaa tallennusobjekteja

- kerätä statiikkaa käyttäjän objekteista

- päivittää objektien metatietoa tili-, säiliö- ja tallennusobjekteissa.

Koska tässä vaiheessa tallennusjärjestelmään ei ollut vielä tallennettu mitään objekteja, testaus aloitettiin *stat*-komennolla, joka kertoo käyttäjästatiikan tallennusjärjestelmässä, kuten esimerkiksi tallennettujen objektien lukumäärän tili-, säiliö- ja tallennusobjekti kohtaiseksi. Komento on muotoa *swift --os-auth-url <identiteettipalvelun_osoite> --os-tenant-name <asiakkaan_nimi> --os-username <käyttäjätunnus> --os-password <salasana> stat*. Komennon parametreinä on annettava identiteettipalvelun osoite, asiakkaan nimi sekä käyttäjätunnus ja salasana. Koska testausta haluttiin helpottaa, luotiin konfiguraatiotiedosto *swift-openrc.sh,* johon nämä parametrit tallennettiin, konfiguraatiotiedosto on liitteenä (liite 8). Koska testausvaiheessa haluttiin komennon tulostavan meneillään olevat operaatiot komentokehotteeseen, stat-komento ajettiin testaustilassa *–debug*-valitsimella. Kuvissa 14 ja 15 tulosteet ensimmäisen *swift --debug stat*-komennon suorituksesta.


```
proxyserv:~ # swift --debug stat
REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent: python-keystoneclient"
DEBUG:keystoneclient.httpclient:REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json"
-H "User-Agent: python-keystoneclient"
REQ BODY: {"auth": {"tenantName": "service", "passwordCredentials": {"username": "swift", "password": "munOMAs414s4n4#"}}}

DEBUG:keystoneclient.httpclient:REQ BODY:
{"auth": {"tenantName": "service", "passwordCredentials": {"username": "swift", "password": "munOMAs414s4n4#"}}}

Starting new HTTP connection (1): authserv
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1): authserv
"POST /v2.0/tokens HTTP/1.1" 200 3206
DEBUG:requests.packages.urllib3.connectionpool:"POST /v2.0/tokens HTTP/1.1" 200 3206
RESP: [200] CaseInsensitiveDict({'date': 'Sat, 11 Jan 2014 19:33:14 GMT',
'vary': 'X-Auth-Token', 'content-length': '3206', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-01-11T19:33:14.551426", "expires": "2014-01-12T19:33:14Z",
"id": "MIIGHgYJKoZIhvcNAQcCoIIGDzCCBgsCAQExCTAHBgUrDgMCGjCCBHQGCSqGSIb3DQEHAaCCBGUEggRheyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNCOwMSOxMV
QxOTozMzoxNC41NTE0MjYiLCAiZXhwaXJlcyI6ICIyMDEOLTAxLTEyVDE5OjMzOjEOWiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJOZW5hbnQiOiB7ImR1c2NyaXBOaW9uIjogIlN3aWZOIENsaWV
udCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogIjYzZjIzZDRkNDE1MjQzZjVjNWQ5N2FjNTE5NzU2ODE1IiwgIm5hbWUiOiAiZGVtbyJ9fSwgInNlcnZpY2VDYXRhbG9nIjogW3siZW5kcG9pbnRz
jogW3siYWRtaW5VUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxLOFVVEhfcyIsICJyZWdpb24iOiAicmVnaW9uT251IiwgImludGVybmFsVVJMIjogImh0dHA6Ly9wcm94eXlcnY6ODA4MC
92MSIsICJpZCI6ICIyOTQxODNkOGY0ZGQ0OTlkOTBlZTUwZWMzZjg4NWYzMyIsICJwdWJsaWNVURwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxLOFVVEhfNjNmMjNkNGQOMTUyNDNmNWI1ZDk
3YWM1MTk3NTY4MTUifVOsICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAib2JqZWNOLXNOb3JlIiwgIm5hbWUiOiAic3dpZnQifSwgeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6ICJo
dHRwOi8vYXVOaHNlcnY6MzUzNTcvdjIuMCIsICJyZWdpb24iOiAicmVnaW9uT251IiwgImludGVybmFsVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAiLCAiaWQiOiAiMjkyZTNmNzdjO
TA5NGJmYjhiODk1MWNmNWY3NThkMGMiLCAicHVibGljVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAifVOsICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAiaWRlbnRpdHkiLC
AibmFtZSI6ICJrZXlzdG9uZSJ9XSwgInVzZXIiOiB7InVzZXJuYW1lIjogInRlc3QiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImQwMjRmNzE5ZjQ5MTQxMjdhOTg3OTdmYzAwYTNmNzUyIiw
gInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9LCB7Im5hbWUiOiAiTWVtYmVyIn1dLCAibmFtZSI6ICJOZXNOIn0sICJtZXRhZGFOYSI6IHsiaXNfYWRtaW4iOiAwLCAicm9sZXMiOiBbIjkxZWZj
NzIyOTIxMTQxNTc5N2ZjNWMyZWRjZGQwY2IxIiwgIjVhYTg3NjQ1NDAwYzRiNDBiZGE2ZjkyZGU2MGIwZTE2Il19fOxggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVVbnNld
DEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2VOMRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb2OCAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAkX7P3ousG+ETarSL6qQOW6Hc9ccy2O
gZpbttrIIqbrMopdi3ERM5Yphqudluj3ViOinJ2FbgxX5zXs92AkzieWOuKohWkQlkZACw4dDGg1Te9fDOfHwIwx2qZizBIH78x4rm8v8H3jfaagOBxHUbmLQfCqbq99O1nfbLFsOUjG9RBIwppdP
dENYPpmjkyLxCsZ8RsT7Q3rLeZ4O4dq6eS+AlKVTsTt+wMOxweElF+l8X6tYAhdEuMRaNHj5fUlGJclhpwR4EOGPTyF4idjuZDhpeLw4yrMS5AlCDMXb+xhMLzOrdEk9QRrwPm-OzIYps0Iu62u6C
r1SNQBLUjWzH4A==",
"tenant": {"description": "Swift Client", "enabled": true, "id": "63f23d4d415243f5b5d97ac519756815", "name": "service"}},
"serviceCatalog": [{"endpoints": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL": "http://proxyserv:8080/v1/AUTH_63f23d4d415243f5b5d97ac519756815"}],
"endpoints_links": [],
"type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL": "http://authserv:35357/v2.0", "region": "regionOne",
"internalURL": "http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c",
"publicURL": "http://authserv:5000/v2.0"}],
"endpoints_links": [], "type": "identity", name": "keystone"}],
"user": {"username": "swift", "roles_links": [], id": "d024f719f4914127a98797fc00a3f752", "roles": [{"name": "admin"}, {"name": "Member"}],
"name": "swift"}, "metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1", "5aa87645400c4b40bda6f92de60b0e16"]}}}
```


Kuva 14. swift –debug-stat-komennon tuloste SuSe välityssolmulta (osa 1).

```
DEBUG:keystoneclient.httpclient:RESP: [200] CaseInsensitiveDict({'date': 'Sat, 11 Jan 2014 19:33:14 GMT', 'vary':
'X-Auth-Token', 'content-length': '3206', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-01-11T19:33:14.551426",
"expires": "2014-01-12T19:33:14Z",
"id": "MIIGHgYJKoZIhvcNAQcCoIIGDzCCBgsCAQExCTAHBgUrDgMCGjCCBHQGCSqGSIb3DQEHAaCCBGUEggRheyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMSOxMV
QxOTozMzoxNC41NTEOMjYiLCAiZXhwaXJlcyI6ICIyMDEOLTAxLTEyVDE5OjMzOjEOWiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJOZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIlN3aWZ0IENsaWV
udCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogIjYzZjIzZDRkNDE1MjQzZjViNWQ5N2FjNTE5NzU2ODE1IiwgIm5hbWUiOiAiZGVtbyJ9fSwgInN1cnZpY2VDYXRhbG9nIjogW3siZW5kcG9pbnRz
IjogW3siYWRtaW5VUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxLOFVVEhfcyIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9wcm94eXNlcnY6ODA4M
C92MSIsICJpZCI6ICIyOTQxODNkOGYOZGQOOTlkOTB1ZTUwZWMzZjg4NWYzMyIsICJwdWJsaWNVURkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxLOFVVEhfNjNmMjNkNGQOMTUyNDNmNWI1ZD
k3YWM1MTk3NTY4MTUifVOsICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAib2JqZWNOLXNOb3JlIiwgIm5hbWUiOiAic3dpZnQifSwgeyJlbmRwb2ludHMiOiBbeyJhZG1pbiVSTCI6ICJ
odHRwOi8vYXVOaHNlcnY6MzUzNTcvdjIuMCIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAiLCAiaWQiOiAiMjkyZTNmNzdj
OTA5NGJmYjhiODk1MWNmNWY3NThkMGMiLCAicHVibGljVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAifVOsICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAiaWRlbnRpdHkiL
CAibmFtZSI6ICJrZXlzdG9uZSJ9XSwgInVzZXIiOiB7InVzZXJuYW1lIjogInRlc3QiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImQwMjRmNzE5ZjQ5MTQxMjdhOTg3OTdmYzAwYTNmNzUyIi
wgInJwbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9LCB7InNhbWUiOiAiTWVtYmVyIn1dLCAibmFtZSI6ICJOZXNOInOsICJtZXRhZGFOYSI6IHsiaXNfYWRtaW4iOiAwLCAicm9sZXMiOiBbIjkxZWZ
jNzIyOTIxMTQxNTc5N2ZjNWMyZWRjZGQwY2IxIiwgIjVhYTg3NjQ1NDAwYzRiNDBiZGE2ZjkyZGU2MGIwZTE2Il19fXOxggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVVbnN1
dDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2VOMRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb2OCAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAkX7P3ousG+ETarSL6qQOW6Hc9ccy2
OgZpbttrIIqbrMopdi3ERM5Yphqudluj3ViOinJ2FbgxX5zXs92akzieWOuKohWkQlkZACw4dDGg1Te9fDOfHwIwx2qZizBIH76x4rm8v6H3jfaagOBxHUbmLQfCqbq99O1nfbLFsOUjG9RBIwppd
PdENYPpmjkyLxCsZ8RsT7Q3rLeZ4O4dq6eS+AlKVTsTt+wMOxweElF+18X6tYAhdEuMRaNHj5fUlGJc1hpwR4EOGPTyF4idjuZDhpeLw4yrMS5AlCDMXb+xhMLzOrdEk9QRrwPm-OzIYps0Iu62u6
Cr1SNQBLUjWzH4A==",
"tenant": {"description": "Swift Client", "enabled": true, "id": "63f23d4d415243f5b5d97ac519756815", "name": "service"}},
"serviceCatalog": [{"endpoints": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL": "http://proxyserv:8080/v1/AUTH_63f23d4d415243f5b5d97ac519756815"}],
"endpoints_links": [],
"type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL": "http://authserv:35357/v2.0", "region": "regionOne",
"internalURL": "http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c",
"publicURL": "http://authserv:5000/v2.0"}],
"endpoints_links": [], "type": "identity", "name": "keystone"}],
"user": {"username": "swift", "roles_links": [], "id": "d024f719f4914127a98797fc00a3f752", "roles": [{"name": "admin"}, {"name": "Member"}],
"name": "swift"}, "metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1", "5aa87645400c4b40bda6f92de60b0e16"]}}}

DEBUG:iso8601.iso8601:Parsed 2014-01-12T19:33:14Z into
{'tz_sign': None, 'second_fraction': None, 'hour': u'19', 'tz_hour': None, 'month': u'01', 'timezone': u'Z', 'second': u'14', 'tz_minute': None,
'year': u'2014', 'separator': u'T', 'day': u'12', 'minute': u'33'} with default timezone <iso8601.iso8601.Utc object at 0xd87f90>
DEBUG:iso8601.iso8601:Got u'2014' for 'year' with default None
DEBUG:iso8601.iso8601:Got u'01' for 'month' with default None
DEBUG:iso8601.iso8601:Got u'12' for 'day' with default None
DEBUG:iso8601.iso8601:Got u'19' for 'hour' with default None
DEBUG:iso8601.iso8601:Got u'33' for 'minute' with default None
DEBUG:iso8601.iso8601:Got u'14' for 'second' with default None
```


Kuva 15. swift --debug stat-komennon tuloste SuSe välityssolmulta (osa 2).

Kuten edellä olevista kuvista 14 – 15 näkyy, asiakassovellus ottaa yhteyden Keystone-todennuspalveluun lähettäen asiakas-, käyttäjä- ja salasanatiedot tunnistuspalveluille ja pyytää todennuspalvelulta tarvittavaa tunnisteavainta sekä Swift-palvelun palvelunpäätepiste tietoja. Keystone palauttaa asiakassovellukselle tunnisteavaimen sekä tunnuksen käytettävissä olevien palvelujen päätepisteiden tiedot. Tuntemattomaksi jääneestä syystä komennon suoritus keskeytyy tämän jälkeen. Komennon tulosteen viimeisiksi riveiksi jää aika-asetusten toteaminen eikä tämän jälkeen ei tapahdu enää mitään. Tässä vaiheessa käynnistettiin ongelman selvitys ja tarkistettiin seuraavat asiat:

- NTP-palvelun asetukset ja palvelinten käyttämä aika
- lokitiedostot
- swift palvelujen käynnissä tila tallennus- ja välityssolmupalvelimilla
- konfiguraatiotiedostot tallennus- ja välityssolmupalvelimilla
- määrittelyt Keystone-palvelimella.

Kaikki palvelimet käyttivät oikeata NTP-palvelinta ja olivat oikeassa ajassa, lokitiedostoihin ei ollut kirjattu mitään merkintää mistä olisi voitu päätellä mistä, ongelma johtui.

Myös kaikki palvelut olivat käynnissä kaikilla palvelimilla ja konfiguraatiotiedostojen sisällöt olivat vaatimusten mukaisia. Vianselvitystä laajennettiin myös Internetin puolelle hakien hakukoneella ja eri tukifoorumeilta ratkaisua ongelmaan. Lisäksi ongelmatilanteesta avattiin eri tukifoorumeille kyselyjä, jos muut Swiftin käyttäjät olisivat törmänneet samaan ongelmaan. Tätä kirjoitettaessa, eli noin 8 viikkoa ensimmäisen SuSe OpenStack Swift toteutuksen testauksen jälkeen, asiaan ei ole saatu ainuttakaan ongelmatilanteen syytä valottavaa vastausta tukifoorumeilta. Tukifoorumien tutkinnan aikana, havaittiin kuitenkin, että useimmat Swift-toteutukset on asennettu Rackspacen suosittamalle Ubuntu Server 12.04 LTS-käyttöjärjestelmälle. Tästä syystä päätettiin vaihtoehtoisena tutkintalinjana asentaa Swift uusille virtuaalikoneille, joiden käyttöjärjestelmänä on tuo Ubuntu Server 12.04 LTS.

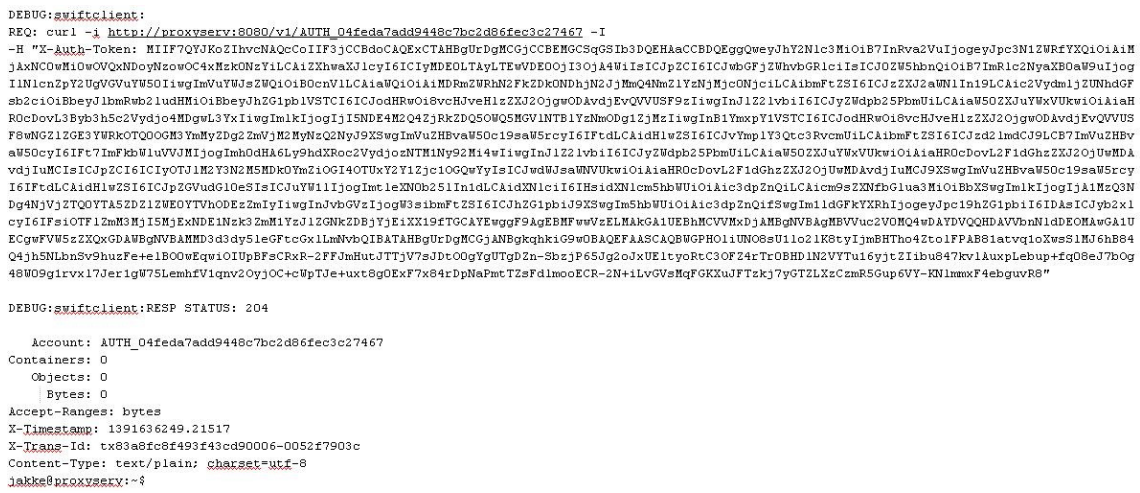
Ubuntu Serveriin asennettu OpenStack Swift-ympäristö vastasi täysin aikaisemmin esiteltyä SuSe-ympäristöä. Asennus ja konfiguraatio tehtiin myös täysin yhdenmukaisesti noudattaen aikaisemmin esiteltyä SuSe-toteutusta. Joidenkin tiedostojen sijainti eroaa johtuen eri Linux-jakeluversiosta, mutta suuria eroja ei ole. Kun ympäristö oli saatu pystytettyä, tehtiin testausta varten samat valmistelevat toimenpiteet kuin SuSe-ympäristössä ja ajettiin *swift --debug stat* komento.

```
jakke@proxyserv:~$ swift  --debug stat
REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent: python-keystoneclient"
DEBUG:keystoneclient.httpclient:
REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent: python-keystoneclient"
REQ BODY: {"auth": {"tenantName": "service", "passwordCredentials": {"username": "swift", "password": "OMAp4ssu1234#"}}}

DEBUG:keystoneclient.httpclient:
REQ BODY: {"auth": {"tenantName": "service", "passwordCredentials": {"username": "swift", "password": "OMAp4ssu1234#"}}}

INFO:urllib3.connectionpool:Starting new HTTP connection (1): authserv
DEBUG:urllib3.connectionpool:"POST /v2.0/tokens HTTP/1.1" 200 3089
RESP: [200] CaseInsensitiveDict({'date': 'Sun, 09 Feb 2014 14:27:08 GMT',
'vary': 'X-Auth-Token', 'content-length': '3089', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-09T14:27:08.139476",
"expires": "2014-02-10T14:27:08Z",
"id": "MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb3DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNCOwMiOwO
VQxNDoyNzowOC4xMzkONzYiLCAiZXhwaXJlcyI6ICIyMDEOLTAyLTEwVDEOOjI3OjA4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJOZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIlNlcnZpY2UgV
GVuYW50IiwgImVuYWJsZWQiOiBOcnVlLCAiaWQiOiAiMDRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjc0NjciLCAibmFtZSI6ICJzZXJ2aWNlIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJlb
mRwb2ludHMiOiBbeyJhZG1pblVSTCI6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL3Byb3h5c
2Vydjo4MDgwL3YxIiwgImlkIjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTB1YzNmODg1ZjMzIiwgInB1YmxpY1VSTCI6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF8wNGZlZGE3YWRkO
TQOOGM3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50cyI6W5sicyI6IFtdLCAidHlwZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBvaW50cyI6IFt7ImFkb
WluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50cyI6IFtdLCAidHlwZSI6I
CIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1Zjc1OGQwYyIsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50cyI6IFtdLCAidHlwZSI6I
CJpZGVudGl0eSIsICJuYW1lIjogImtleXN0b251In1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogIjA1MzQ3NDg4NjVjZTQOYTA5ZDZlZ
WEOYTVhODEzZmIyIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAic3dpZnQifSwgImlldGFkYXRhIjogeyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5M
jExNDE1Nzk3ZmM1YzJl2GNkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgNVBAgMBVVuc2VOMQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxGDAWBgNVB
AMMD3d3dy5leGFtcGx1LmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASCAQBWGPHOliUNO8sU1lo2lK8tyIjmBHTho4ZtolFPAB81atvq1oXwsS1MJ6hB84Q4jh5NLbnSv9huzFe+elBO0
wEqwiOIUpBFsCRxR-2FFJmHutJTTjV7sJDtOOgYgUTgDZn-SbzjP65Jg2oJxUEltyoRtC3OFZ4rTrOBHDlN2VYTu16yjtZIibu847kvlAuxpLebup+fqO8eJ7bOg48WO9g1rvx17Jer1gW75Lemh
fV1qnv2OyjOC+cWpTJe+uxt8gOExF7x84rDpNaPmtTZsFdlmooECR-2N+iLvGVsMqFGKXuJFTzkj7yGTZLXzCzmR5Gup6VY-KNlmmxF4ebguvR8",
"tenant": {"description": "Service Tenant", "enabled": true, "id": "04feda7add9448c7bc2d86fec3c27467", "name": "service"}},
"serviceCatalog": [{"endpoints": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL": "http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467"}],
"endpoints_links": [], "type": "object-store", "name": "swift"},
{"endpoints": [{"adminURL": "http://authserv:35357/v2.0", "region": "regionOne",
"internalURL": "http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c",
"publicURL": "http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "keystone"}],
"user": {"username": "swift", "roles_links": [], "id": "0534748865ce44a09d6eea4a5a813fb2",
"roles": [{"name": "admin"}], "name": "swift"}, "metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1"]}}}
```

Kuva 16. swift --debug stat-komennon tuloste SuSe-välityssolmulta (osa 1).

```
DEBUG:keystoneclient.httpclientRESP: [200] CaseInsensitiveDict({'date': 'Sun, 09 Feb 2014 14:27:08 GMT',
'vary': 'X-Auth-Token', 'content-length': '3089', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-09T14:27:08.139476",
"expires": "2014-02-10T14:27:08Z",
"id": "MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb3DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNCOwMiOwO
VQxNDoyNzowOC4xMzkONzYiLCAiZXhwaXJlcyI6ICIyMDEOLTAyLTEwVDEOOjI3OjA4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJOZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIlNlcnZpY2UgV
GVuYW50IiwgImVuYWJsZWQiOiBOcnVlLCAiaWQiOiAiMDRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjcONjciLCAibmFtZSI6ICJzZXJ2aWNlIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJlb
mRwb2ludHMiOiBbeyJhZG1pbiVSTCI6ICJodHRwOi8vcHJveHl6ZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50cyI6IFt
2Vydjo4MDgwL3YxIiwgImlkIjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTBlYzNmODg1ZjMzIiwgInB1YmxpY1VSTCI6ICJodHRwOi8vcHJveHl6ZXJ2OjgwODAvdjEvQVVUSF8wNGZlZGE3YWRkO
TQOOGM3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50cyI6IFtdLCAidHlwZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBvaW50cyI6IFt7ImFkb
WluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCIsICJpZCI6I
CIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1ZjclOGQwYyIsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50cyI6W3siaW5rcyI6IFtdLCAidHlwZSI6I
CJpZGVudGl0eSIsICJuYW1lIjogImtleXN0b25lInldLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogIjAiMzQ3NDg4NjVjZTQOYTA5ZDZlZ
WE0YTVhODEzZmIyIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAic3dpZnQifSwgImlkIjogIjc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5M
jExNDE1Nzk3ZmM1YzJlZGNkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgNVBAgMBVVuc2VOMQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxGDAWBgNVB
AMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASCAQBWGPHO1iUNO8sU1lo2lK8tyIjmBHTho4ZtolFPAB81atvq1oXwsS1MJ6hB84Q4jh5NLbn8v9huzFe+elBOO
wEqwiOIUpBFsCRxR-2FFJmHutJTTjV7sJDtOOgYgUTgDZn-SbzjP65Jg2oJxUE1tyoRtC3OFZ4rTrOBHD1N2VYTu16yjtZIibu847kvlAuxpLebup+fqO8eJ7bOg48WO9g1rvxl7Jer1gW75Lemh
fV1qnv2OyjOC+cWpTJe+uxt8gOExF7x84rDpNaPmtTZsFdlmooECR-2N+iLvGVsMqFGKXuJFTzkj7yGTZLXzCzmR5Gup6VY-KNlmmxF4ebguvR8",
"tenant": {"description": "Service Tenant", "enabled": true, "id": "04feda7add9448c7bc2d86fec3c27467", "name": "service"}},
"serviceCatalog": [{"endpoints": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL": "http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467"}],
"endpoints_links": [], "type": "object-store", "name": "swift"},
{"endpoints": [{"adminURL": "http://authserv:35357/v2.0", "region": "regionOne",
"internalURL": "http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c",
"publicURL": "http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "keystone"}],
"user": {"username": "swift", "roles_links": [], "id": "0534748865ce44a09d6eea4a5a813fb2",
"roles": [{"name": "admin"}], "name": "swift"}, "metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1"]}}}
```


Kuva 17. swift --debug stat-komennon tuloste SuSe-välityssolmulta (osa 2).



```
DEBUG:swiftclient:
REQ: curl -i http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467 -I
-H "X-Auth-Token: MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb3DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiM
jAxNCOwMiOwOVQxNDoyNzowOC4xMzkONzYiLCAiZXhwaXJlcyI6ICIyMDEOLTAyLTEwVDEOOjI3OjA4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJOZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjog
IlNlcnZpY2UgVGVuYW50IiwgImVuYWJsZWQiOiBOcnVlLCAiaWQiOiAiMDRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjcONjciLCAibmFtZSI6ICJzZXJ2aWNlIn19LCAic2VydmljZUNhdGF
sb2ciOiBbeyJlbmRwb2ludHMiOiBbeyJhZG1pbiVSTCI6ICJodHRwOi8vcHJveHl6ZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaH
ROcDovL3Byb3h5c2Vydjo4MDgwL3YxIiwgImlkIjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTBlYzNmODg1ZjMzIiwgInB1YmxpY1VSTCI6ICJodHRwOi8vcHJveHl6ZXJ2OjgwODAvdjEvQVVUS
F8wNGZlZGE3YWRkOTQOOGM3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50cyI6IFtdLCAidHlwZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBv
aW50cyI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvbiI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDA
vdjIuMCIsICJpZCI6ICIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1ZjclOGQwYyIsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVuZHBvaW50cyI6W3siaW5rcy
I6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0b25lInldLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogIjAiMzQ3N
Dg4NjVjZTQOYTA5ZDZlZWE0YTVhODEzZmIyIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm1hdGFkYXRhIjogeyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5MjExNDE1Nzk3ZmM1YzJlZGNkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgNVBAgMBVVuc2VOMQ4wDAYDVQQHDAVVbnNldDEOMAwGA1U
ECgwFVW5zZXQxGDAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0BAQEFAASCAQBWGPHO1iUNO8sU1lo2lK8tyIjmBHTho4ZtolFPAB81atvq1oXwsS1MJ6hB84
Q4jh5NLbn8v9huzFe+elBOOwEqwiOIUpBFsCRxR-2FFJmHutJTTjV7sJDtOOgYgUTgDZn-SbzjP65Jg2oJxUE1tyoRtC3OFZ4rTrOBHD1N2VYTu16yjtZIibu847kvlAuxpLebup+fqO8eJ7bOg
48WO9g1rvxl7Jer1gW75LemhfV1qnv2OyjOC+cWpTJe+uxt8gOExF7x84rDpNaPmtTZsFdlmooECR-2N+iLvGVsMqFGKXuJFTzkj7yGTZLXzCzmR5Gup6VY-KNlmmxF4ebguvR8"

DEBUG:swiftclient:RESP STATUS: 204

   Account: AUTH_04feda7add9448c7bc2d86fec3c27467
Containers: 0
   Objects: 0
     Bytes: 0
Accept-Ranges: bytes
X-Timestamp: 1391636249.21517
X-Trans-Id: tx83a8fc8f493f43cd90006-0052f7903c
Content-Type: text/plain; charset=utf-8
jakke@proxyserv:~$
```


Kuva 18. swift –debug stat komennon tuloste SuSe-välityssolmulta (osa 3).

Kuten komennon tulosteista otetuista kuvista 16 – 18 näkyy, jälleen asiakassovellus ottaa yhteyttä Keystoneen, saa vastaukseksi tunnisteavaimen ja palvelujen päätepistetiedot. Hyödyntäen tunnisteavainta ja Swiftin PublicURL:ia, asiakassovellus ottaa yhteyden itse Swift palvelun välityspalvelimeen ja saa siltä vastaukseksi kyseisen asiakkaan statiikan. Tätä vaihetta ei koskaan tapahtunut SuSe Swift-toteutuksessa. Selvittämättä jäi, johtuiko tämä SuSe:n Swift-jakelun asiakassovelluksen virheestä vai itse Swift-palvelun virheestä. Vianselvitykseen käytettiin kymmeniä tunteja, mutta saatavilla oleva tuki OpenStack Swiftlle oli sen verran heikkolaatuista, että ongelma ei selvinnyt.

Koska OpenStack Swiftin todettiin toimivan moitteettomasti Ubuntu Server 12.04 LTS:n tehdyssä asennuksessa, loput testauksesta tehtiin tähän versioon tehtyä asennusta käyttäen.

Seuraavaksi haluttiin testata palvelun käytettävyyttä ”julkisesta” verkosta. Tähän tarkoitukseen varatulle SuSe Linux-palvelimelle asennettiin Swiftin asiakassovellus. Asennus tapahtui http://docs.openstack.org/user-guide/content/install_clients.html-ohjeistusta seuraten. Ensimmäisenä toimenpiteenä asiakakoneelle on asennettava Python-tuki. Pythonia asennettaessa on huomattava, että Swiftin asiakkaat eivät tue Python 3.x versioita vaan on asennettava sitä vanhempi versio. Pythonin asennus asiakaskoneelle tapahtui yksinkertaisesti YaST-työkalulla, valittiin työkalulla sopiva Python-paketti ja asennettiin se. Tämän jälkeen, asennettiin ohjeistuksen mukaisesti Python setuptools-paketti. Tämä tapahtui helpoimmin https://pypi.python.org/pypi/setuptools#unix-based-systems-including-mac-os-x sivustolta saatavilla olevalla *ez_setup.py* Python-sovelluksella. Sovellus lataa ja asentaa oikean setuptools-version automaattisesti, sovelluksen lataus asiakaskoneelle tapahtui komennolla

wget https:bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py -O - | python

Kun setuptools oli asennettu, asiakaskoneelle asennettiin Python-pakettienhallintatyökalu *pip*-ohjeistuksen mukaisesti. Tämä tapahtui komennolla *zypper install python-pip*. Kun Swift-asiakassovelluksen asennusta valmistelevat toimenpiteet oli tehty, asennettiin itse asiakassovellus komennolla *pip install python-swiftclient* ja testauksen helpottamiseksi luotiin *swift-openrc.sh*-tiedosto. Tämän jälkeen testattiin asiakassovelluksen asennus swift --debug stat-komennolla ja havaittiin asennuksen toimivan. Lisäksi testattiin muita Swiftin komentorivi asiakassovellukselle saatavilla olevia komentoja ja havaittiin, että tiedostojen tallennus, päivitys ja poisto toimi moitteetta.

Koska palvelua haluttiin testata myös muilla alustoilla kuin pelkästään Linux-alustalla, saatavilla olevalla Windows Server 2008R2-palvelimelle asennettiin graafiset Swift-asiakasohjelmat

- CloudBerry Explorer for OpenStack
- Cyberduck

sekä OpenStackin komentorivipohjainen asiakassovellus ja Swiftin käyttöä testattiin näillä. Kuvakaappaukset testeistä ovat liitteenä (liite 9).

7 Päätelmät

Etäresurssipalveluna toteutettujen tallennusratkaisujen käyttö sekä niihin tallennetun tiedon määrä tulevat olemaan merkittävässä kasvussa tällä vuosikymmenellä. Yhä useammat kuluttajat tallentavat kaiken tietonsa etäresurssipalveluihin ja haluavat, että tuo tieto on nopeasti ja helposti saatavilla kaikilla päätelaitteilla. On siis helposti nähtävissä, että tämä sama muutos tiedontallennus ja -käsittely tavoissa tulee siirtymään yrityskäyttäjien tapoihin käsitellä ja tallentaa tietoa. Lisäksi yritysmaailman tehokkuusvaatimukset kiristyvät jatkuvasti, tietoa on pystyttävä tallentamaan yhä tehokkaammin ja sen on oltavissa saatavilla nopeasti ja helposti 24/7. Enää ei ole helposti hyväksyttävissä, että yrityskäyttäjien tiedostot olisivat saatamattomissa tallennusjärjestelmälle tehtävän huollon ajaksi. Myös laitteistojen käyttöasteiden halutaan olevan korkeat, joten enää ei nähdä järkeväksi investoida yksittäisiin palvelimiin, joihin tallennetaan vain yrityksen käyttäjien osajoukon tieto. Yritykset haluavat keskitettyjä ratkaisuja, joissa kaikkien yrityskäyttäjien tieto on tallennettu jaettuun tallennusjärjestelmään, joka mahdollistaa tiedon käsittelyn eri päätelaitteista. Koska palveluja toteuttavien laitteistojen näennäistäminen tarjoaa hyvän käyttöasteen fyysiselle laitteistolle, keskitetyt ympäristöt sekä ratkaisut, jotka on rakennettu näennäistetynlaitteiston avulla, tulevat saamaan yhä suurempaa osuutta näiden järjestelmien markkinoista.

Tilanne ei kuitenkaan ole tällä hetkellä täysin ongelmaton. Vuonna 2013 paljastuneet valtionlaitosten suorittamat salakuuntelut ja käyttäjien vakoilu nostavat varmasti myös vahvasti esille kysymykset, missä tallennettu tieto sijaitsee ja ketkä kaikki pääsevät tietoon käsiksi. Tämä tulee varmasti olemaan monille jaettuja tiedontallennusetäresurs-

sipalveluja tarjoaville yrityksille se avainkysymys, joka vaikuttaa siihen, tallentavatko yrityskäyttäjät kriittistä tietoa heidän palveluihinsa. Valitettavasti tilanne on tästä huolimatta osin riistäytynyt käsistä. Tietoturvasta joko tietämättömät tai siitä välittämättömät käyttäjät tallentavat yritysten tietoa julkisiin tallennuspalveluihin salaa työnantajaltaan. Eräissä verkkoseminaareissa joihin tutustuin tätä insinöörityötä varten, esitettiin jopa arvioita, että kolmannes yrityskäyttäjistä tallentaa työhönsä liittyvä tietoa julkisiin, eikä virallisesti työnantajansa tarjoamiin tallennuspalveluihin.

Edellä mainituista johtuen tiedontallennusetäresurssipalveluna tulee varmasti olemaan yksi lähivuosina eniten kehittyvistä ja laajenevista tallennusratkaisuista. Ja mikäli tarjolla on tai tarjolle tulee helposti toteutettavissa olevia ratkaisuja, jotka voidaan toteuttaa näennäistetyillä laitteistoilla, on täysin mahdollista, että yksityisten etäresurssipalveluna toteutettavien tallennusratkaisujen osuus markkinoista kasvaa.

Teknisesti valittavissa on jo nyt useita erilaisia kaupallisia ja ei-kaupallisia tapoja, joilla tämä voidaan toteuttaa. Kuten käytännön osuudesta on havaittavissa, on yksityisen etäresurssitallennuspalvelun toteutus avoimen lähdekoodin ohjelmistolla tehtävissä, vaikka toteuttajalla ei olekaan syvällistä osaamista OpenStack-pinon sovellusten toteutuksesta. Toteutuksessa on toistaiseksi vielä monia sudenkuoppia, suurimpina on ehdottomasti mainittava laadukkaan tuen puute sekä se, että käytettävyys ei ole kovinkaan korkealla tasolla ilman lisäpanostusta ratkaisun käytettävyyteen. Niinpä ratkaisua suunnitellessa on hyvä kartoittaa tarpeet, joihin ratkaisua haetaan, tuntea ympäristö, johon ratkaisua ollaan toteuttamassa ja selvittää käyttäjien vaatimukset. Tällöin nämä tuntevat valitun ratkaisun työkaluksi, jota on helppo ja järkevä käyttää. Suunnitteluvaiheessa olisi hyvä myös pohtia tulevaisuutta ja olla realistinen käsitys siitä, kuinka suureksi toteutettava ympäristö tulee muodostumaan. Tällöin vältetään monia ongelmia ympäristön laajentuessa. Varsinaista ratkaisua valittaessa ja sitä toteutettaessa pidän merkittävimpinä kysymyksinä juuri toteutuksen ylläpidon helppoutta, tietoturvan vaatimien ratkaisujen toteutuksen suunnittelua ja saatavilla olevaa tuen määrää sekä laatua.

Saavutetut tulokset

Insinöörityöprojektin tuloksena syntyi raportti, joka luo perustavan näkemyksen siitä, mikä on etäresurssipalvelu, mikä on etäresurssipalveluna toteutettava objektipohjainentallennus mitkä ovat toteutukseen liittyvät yksityiskohdat ja teknologiat. Dokumentti tarjoaa lukijalle yleiskuvan tallennusjärjestelmän eri komponenteista sekä teknologioista, jotka liittyvät tallennusjärjestelmän toteutukseen. Lisäksi työssä käsitellään objektitallennuksen teoria sillä syvyydellä, että lukija voi hyödyntää teoriaa eri objektitallennusratkaisuvaihtoehtoja tutkiessaan.

Lisäksi työn edetessä havaittiin, että on todellakin mahdollista toteuttaa objektipohjainen etäresurssipalvelu hyödyntäen näennäistettyä palvelinympäristöä avoimen lähdekoodin ratkaisulla ilman syvällisempää OpenStack-pinon tuntemusta.

Työn ulkopuolelle rajatut kehityskohteet

Työn rajauksessa pyrittiin kokonaisuuteen, jossa käsitellään etäresurssipalveluna toteutettavan tallennusratkaisun toteutukseen liittyvät käsitteet ja teknologia, sekä annetaan lukijalle käsitys näistä. Tästä syystä insinöörityön edetessä havaittiin, että kokonaisuus muodostui melko suureksi, eikä kaikkien teknologioiden yksityiskohtia, vaikutusta toteutukseen tai yksityiskohtia pystytty käsittelemään tai analysoimaan kovinkaan syvällisesti. Niinpä useimmat käsitteet ja teknologiat rajattiin esiteltäväksi vain hyvin yleisellä tasolla, jotta lukija saa peruskäsityksen niistä.

Koska työ toteutettiin näennäistetyillä laitteistolla varsinaisen fyysisen laitteiston puuttuessa, työhön muodostui rajauksesta johtuen merkittävä vastakkainasettelu lähdekoodin tarjoajan suositusten ja toteutuksen välille. Jotta työhön olisi saatu merkittävä tutkimuksellinen näkökohta, olisi työ pitänyt toteuttaa kahden eri ympäristön välisenä tutkimuksena. Toinen olisi ollut Rackspacen suositusten mukaisesti rakennettu fyysisistä palvelimista muodostuva laitteisto ja toinen olisi ollut VMware-virtuaalipalvelimilla ajettava ympäristö, joka olisi saanut levynsä iSCSI:lla oikeasta levyjärjestelmästä. Ympäristöjä olisi sitten verrattu suorituskyvyllisesti toisiinsa ja sitä kautta saatu näkemystä onko toteutusten välillä tieteellisesti merkittäviä suorituskyky eroja.

Lisäksi osa-alueet, jotka jouduttiin rajaamaan työn ulkopuolelle sen laajuudelle asetettujen rajoitusten vuoksi, ovat käytettävyys ja tietoturva. OpenStack Swift tukee natiivisti komentorivipohjaista asiakassovellusta, joka ei ole käytettävyydeltään mitenkään hyvä. Työn hyödynnettävyyden kannalta olisi ollut mielenkiintoista lisätä toteutukseen käytettävyyden optimointi, jossa asiakaskoneille olisi esimerkiksi ohjelmoitu sovellus, jolla tallennusjärjestelmää käytetään siten, että käyttäjäkokemus olisi vastannut normaalin paikallisen tallennuksen käyttöä. Tietoturvan kannalta tässä insinöörityössä toteutettu OpenStack Swift-asennus on sangen turvaton, yhteydet asiakaskoneiden ja tallennusympäristön välillä tapahtuvat salaamattomana http:n yli, samoin liikenne eri solmujen välillä ei ole salattua ollen alttiina salakuuntelulle. Lisäksi konfiguraatiotiedostot sisältävät selkokielisenä tekstinä olevia salasanoja, kuten esimerkiksi Swiftin jaettu valtuutusavain. Työn hyödynnettävyyden kannalta työtä voisi kehittää siten, että tietoliikenne eri solmujen sekä asiakkaiden ja järjestelmän välillä on salattu ipsec:iä, TSL:ää (Transport Layer Security) tai muuta tietoliikenteen salaavaa teknologiaa hyödyntäen. Tietoturvan kannalta olisi myös oleellista kyetä muuttamaan toteutusta siten, että ainakin kaikki tallennetut salasanat säilöttäisiin salatussa muodossa ja mikäli vain mahdollista, kaikki konfiguraatiotiedostot voitaisiin tallentaa muodossa, jossa niitä ei voida muokata.

Tulosten hyödynnettävyys

Insinöörityön tuloksena syntynyt raportti tarjoaa ensi katsauksen henkilölle, joka ei entuudestaan ole tutustunut tallennusratkaisujen teknologiaan. Työhön tutustuttaessa lukijalle pitäisi välittyä näkemys tallennusratkaisujen tärkeimmistä komponenteista ja siitä miten ne liittyvät toisiinsa. Mikäli lukija työhön tutustuttuaan on kiinnostunut toteuttamaan omaa tallennusratkaisuaan, lukijalla pitäisi olla riittävä peruskäsitys, miten eri komponentit ja teknologia vaikuttavat tallennusratkaisun toteutuksessa. Lisäksi lukijalla pitäisi olla työn luettuaan riittävä ymmärrys objektitallennuksen teoriasta, jotta hän pystyy käsittämään, miten eri komponentit objektitallennusjärjestelmässä liittyvät teoreettiseen malliin.

Lähteet

1 Gartner Says That Consumers Will Store More Than a Third of Their Digital Content in the Cloud by 2016. 2014. Verkkodokumentti. Garter, Inc. <http://www.gartner.com/newsroom/id/2060215>. Luettu 1.2.2014.

2 Arnold, Joe. 2013. Software defined storage with OpenStack Swift. 1st edition. San Francisco: SwiftStack, Inc.

3 McCarthy, John Speaking at the MIT Centennial in 1961. Teoksessa Abelson, Hal (edited). Architects of the Information Society, Thirty-Five Years of the Laboratory for Computer Science at MIT. Cambridge, MA: The MIT Press

4 The NIST Definition of Cloud Computing. 2011. Verkkodokumentti. National Institute of Standards and Technology. <http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf>. Luettu 4.9.2013.

5 FG Cloud Technical Report. 2013. verkkodokumentti. ITU. <http://www.itu.int/en/ITU-T/focusgroups/cloud/Documents/FG-coud-technical-report.zip>. Luettu 4.9.2013.

6 Brown, James - Amazon Web Services. 2013. Verkkoseminaari. Brighttalk, Inc. <https://www.brighttalk.com/webcast/9019/84577>. Esitetty 29.8.2013.

7 Cloud Cube Model: Selecting Cloud Formations for Secure Collaboration. 2009 Verkkodokumentti. Opengroup.org. <https://collaboration.opengroup.org/jericho/cloud_cube_model_v1.0.pdf>. Luettu 4.09.2013.

8 Somasundaram, Gnanasundaram & Alok, Shrivastava. 2012. Infromation Storage and management - Chapter 2.6. 2nd edition. Indianapolis: John Wiley & Sons.

9 Why Don’t HDDs Spin Faster than 15K RPM. 2012. Verkkodokumentti. The SSD Guy. <http://thessdguy.com/why-dont-hdds-spin-faster-than-15k-rpm/>. Updated 14 Nov 2012. Luettu 4.9.2013.

10 Solid State Storage Standards Explained. 2013. Verkkodokumentti. SNIA. <http://www.snia.org/forums/sssi/knowledge/standards>. Luettu 5.9.2013.

11 An Oracle White Paper - StorageTek T10000C Tape Drive Enterprise-Class Design for Maximum Reliability. 2011. Verkkodokumentti. Oracle, Inc. <http://www.oracle.com/us/products/servers-storage/storage/tape-storage/t10000c-reliability-wp-409919.pdf>. Luettu 25.10.13

12 Patterson, David A. & Gibson, Garth & Katz, Randy H. - A Case for Redundant Arrays of Inexpensive Disks (RAID). 1988. Verkkodokumentti. University of California Berkeley. <http://www.cs.cmu.edu/~garth/RAIDpaper/Patterson88.pdf>. Luettu 21.10.13

13 Common RAID Disk Data Format Specification - Version 2.0 Revision 19. 2009. verkkodokumentti. SNIA. <http://www.snia.org/sites/default/files/SNIA_DDF_Technical_Position_v2.0.pdf>. Luettu 22.10.13.

14 SATA technical Overview. 2013. Verkkosivut. The Serial ATA International Organization. <https://www.sata-io.org/technical-overview>. Luettu 3.11.2013.

15 SCSI Storage standards. Verkkosivu. T10.org. <http://www.t10.org/index.html>. Updated 2.2.2012. Luettu 3.11.2013.

16 SCSI Terms and Terminology. 2013. Verkkodokumentti. SCSITA. <http://www.scsita.org/terms-and-terminology.html>. Luettu 3.11.2013.

17 Somasundaram, Gnanasundaram & Alok, Shrivastava. 2012. Infromation Storage and management - Chapter 4.1.2. 2nd edition. Indianapolis: John Wiley & Sons.

18 Somasundaram, Gnanasundaram & Alok, Shrivastava. 2012. Infromation Storage and management - Chapter 2.8. 2nd edition. Indianapolis: John Wiley & Sons.

19 Somasundaram, Gnanasundaram & Alok, Shrivastava. 2012. Infromation Storage and management - Chapter 5. 2nd edition. Indianapolis: John Wiley & Sons.

20 Internet Small Computer Systems Interface (iSCSI). 2004. Verkkodokumentti. IETF.org. <http://tools.ietf.org/html/rfc3720>. Luettu 20.10.2013.

21 Fibre Channel Over TCP/IP (FCIP). 2004. Verkkodokumentti. IETF.org. <http://tools.ietf.org/html/rfc3821>. Luettu 18.11.2013.

22 NFS: Network File System Protocol Specification. 1989. Verkkodokumentti. IETF.org. <http://tools.ietf.org/html/rfc1094>. Luettu 20.11.2013

23 NFS Version 3 Protocol Specification. 1995. Verkkodokumentti. IETF.org. <http://tools.ietf.org/html/rfc1813>. Luettu 20.11.2013

24 Common Internet File System (CIFS) Protocol. 2013. verkkodokumentti. Microsoft. <http://msdn.microsoft.com/en-us/library/ee442092.aspx>. Luettu 20.11.2013

25 Object Storage: The Future Building Block for Storage Systems. 2013. Verkkodokumentti. IBM Haifa Research Laboratories. <http://webhdd.ru/library/files/PositionOSD.pdf>. Luettu 21.11.2013

26 Object-Based Storage. 2003. Verkkodokumentti. IEEE Communications Magazine. < http://www.storagevisions.com/White%20Papers/MesnierIEEE03.pdf>. Luettu 21.11.13

27 Du, David Hung-Chang. 2008. Recent Advancements and Future Challenges of Storage Systems. Proceedings of the IEEE Volume: 96 , Issue: 11. 11.2008, s. 1875 – 1886.

28 Principled Design of the Modern Web Architecture. 2002. Verkkodokumentti. University of California. <http://www.ics.uci.edu/~taylor/documents/2002-REST-TOIT.pdf>. Luettu 17.12.2013.

29 Uniform Resource Identifiers (URI): Generic Syntax. 1998. Verkkodokumentti. IETF.org. <http://tools.ietf.org/html/rfc2396>. Luettu 18.12.2013

30 SOAP Version 1.2 Part 1: Messaging Framework (Second Edition). 2007. Verkkodokumentti. W3.org. <http://www.w3.org/TR/soap12-part1/>. Luettu 18.12.2013

31 Swift 1.11.0.73.ge0c9211 Documentation. 2013. Verkkosivut. OpenStack.org. <http://docs.openstack.org/developer/swift/index.html#>. Luettu 1.12.2013

32 OpenStack Object Storage Manuals - Training guide. 2013. Verkkosivut. OpenStack.org. <http://docs.openstack.org/training-guides/content/module003-ch000-openstack-objstore.html>. Luettu 2.12.2013

33 OpenStack Swift Architecture. 2013. Verkkosivut. SwiftStack, Inc. <http://swiftstack.com/product/architecture/>. Luettu 3.12.2013

34 OpenStack ManualsOpenStack Installation Guide for openSUSE and SUSE Linux Enterprise Server - havana / Chapter 3. Configure the Identity Service. 2013. Verkkosivut. OpenStack.org. <http://docs.openstack.org/havana/install-guide/install/zypper/content/ch_keystone.html>. Luettu 15.12.2013

35 SUSE LINUX Administration Guide. 2013. Verkkosivut. Novell, Inc. <http://www.novell.com/documentation/suse91/suselinux-adminguide/html/>. Luettu 18.12.20.13

Liite 1. Määritellään käyttäjät, asiakkaat ja roolit

```
#Määritä  OS_SERVICE_TOKEN ja OS_SERVICE_ENDPOINT joilla määritellään
identiteettipalvelu tiedot.
export OS_SERVICE_TOKEN=T4lM3sCa71N6LyCA
export OS_SERVICE_ENDPOINT=http://authserv:35357/v2.0

##Luodaan asiakas sekä pääkäyttäjä keystonelle. Sekä käyttäjät ja asiakkaat Swift:iä
varten
#Asiakkaiden luonti

keystone tenant-create --name=admin --description="Admin Tenant"
+-------------+----------------------------------------------------+
|   Property  |                   Value                            |
+-------------+----------------------------------------------------+
| description |                   Admin Tenant                     |
|   enabled   |                   True                             |
|      id     | 4333601479d94e339ed4b7ec2228b704 |
|     name    |                   admin                            |
+-------------+----------------------------------------------------+

keystone tenant-create --name=service --description="Service Tenant"
+-------------+----------------------------------------------------+
|   Property  |                   Value                            |
+-------------+----------------------------------------------------+
| description |                   Service Tenant                   |
|   enabled   |                        True                        |
|      id     | 04feda7add9448c7bc2d86fec3c27467 |
|     name    |                   service                          |
+-------------+----------------------------------------------------+

keystone tenant-create --name=test --description="Test Tenant"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Test Tenant            |
|   enabled   |              True                |
|      id     | fb83bb1cf2ae4430ab5b02169f48d9a3 |
|     name    |              test                |
+-------------+----------------------------------+

keystone tenant-create --name=demo --description="Swift Client"
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |           Swift Client           |
|   enabled   |              True                |
|      id     | 63f23d4d415243f5b5d97ac519756815 |
|     name    |              demo                |
+-------------+----------------------------------+
```

#Luodaan keystone pääkäyttäjä

keystone user-create --name=admin --pass='EL1E$Ta!nS+ooM' --email=admin@os-site.local

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |       admin@os-site.local        |
| enabled  |               True               |
|    id    | d8332a64afdf4f2092e34e2cc79d8b2e |
|   name   |              admin               |
+----------+----------------------------------+
```

keystone user-create --name=swift --pass=OMAp4ssu1234# --email=swift.admin@os-site.local

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |    swift.admin@os-site.local     |
| enabled  |               True               |
|    id    | 0534748865ce44a09d6eea4a5a813fb2 |
|   name   |              swift               |
+----------+----------------------------------+
```

keystone user-create --name=test --pass=munOMAs4l4s4n4# --email=malli@os-site.local

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|  email   |       malli@os-site.local        |
| enabled  |               True               |
|    id    | d024f719f4914127a98797fc00a3f752 |
|   name   |              test                |
+----------+----------------------------------+
```

#Luodaan roolit

keystone role-create --name=admin

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 91efc7229211415797fc5c2edcdd0cb1 |
|   name   |              admin               |
+----------+----------------------------------+
```

keystone role-create --name=Member

```
+----------+----------------------------------+
| Property |              Value               |
+----------+----------------------------------+
|    id    | 5aa87645400c4b40bda6f92de60b0e16 |
|   name   |              Member              |
+----------+----------------------------------+
```

```
#Liitetään rooli ja asiakas käyttäjään
keystone user-role-add --user=d8332a64afdf4f2092e34e2cc79d8b2e --
tenant=4333601479d94e339ed4b7ec2228b704 --
role=91efc7229211415797fc5c2edcdd0cb1
keystone user-role-add --user=0534748865ce44a09d6eea4a5a813fb2 --
tenant=04feda7add9448c7bc2d86fec3c27467 --
role=91efc7229211415797fc5c2edcdd0cb1
keystone user-role-add --user=d024f719f4914127a98797fc00a3f752 --
tenant=63f23d4d415243f5b5d97ac519756815 --
role=91efc7229211415797fc5c2edcdd0cb1
keystone user-role-add --user=d024f719f4914127a98797fc00a3f752 --
tenant=63f23d4d415243f5b5d97ac519756815 --
role=5aa87645400c4b40bda6f92de60b0e16
```

Liite 2. Keystone- ja Swift-palvelujen rekisteröinti

##Määritetään palvelutyypit ja API:en päätepisteet
#Määritetään identiteetti palvelu
keystone service-create --name=keystone --type=identity --description="Keystone Identity Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Keystone Identity Service    |
|      id     | 8542a4faaa524a9dad715367d7623b92 |
|     name    |             keystone             |
|     type    |             identity             |
+-------------+----------------------------------+
```

#Määritetään Swift objektitallennuspalvelu
keystone service-create --name=swift --type=object-store --description="Swift Object Storage Service"

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
| description |     Swift Object Storage Service |
|      id     | a48897e2c2e44cde80e3c08657172e85 |
|     name    |             swift                |
|     type    |          object-store            |
+-------------+----------------------------------+
```

#Määritellään API päätepisteet Identity ja Object storage palveluille käyttäen edellisen komennon palauttamaa service ID:tä.
keystone endpoint-create \
--service-id=8542a4faaa524a9dad715367d7623b92 \
--publicurl=http://authserv:5000/v2.0 \
--internalurl=http://authserv:5000/v2.0 \
--adminurl=http://authserv:35357/v2.0

```
+-------------+----------------------------------+
|   Property  |              Value               |
+-------------+----------------------------------+
|   adminurl  |     http://authserv:35357/v2.0   |
|      id     | 593578b8fa8e4f5c86ddefdc6b66bd91 |
|  internalurl|     http://authserv:5000/v2.0    |
|   publicurl |     http://authserv:5000/v2.0    |
|    region   |           regionOne              |
|  service_id | 8542a4faaa524a9dad715367d7623b92 |
+-------------+----------------------------------+
```

```
keystone endpoint-create \
--service-id=a48897e2c2e44cde80e3c08657172e85 \
--publicurl="http://proxyserv:8080/v1/AUTH_\$(tenant_id)s" \
--internalurl="http://proxyserv:8080/v1" \
--adminurl="http://proxyserv:8080/v1/AUTH_\$(tenant_id)s"
+-------------+---------------------------------------------------------+
|   Property  |                          Value                          |
+-------------+---------------------------------------------------------+
|   adminurl  | http://proxyserv:8080/v1/AUTH_$(tenant_id)s    |
|      id     |       de78c7e0c4fb44bb87720baa8884e5d6        |
|  internalurl |          http://proxyserv:8080/v1                       |
|   publicurl   | http://proxyserv:8080/v1/AUTH_$(tenant_id)s   |
|    region   |                  regionOne                              |
|  service_id |       a48897e2c2e44cde80e3c08657172e85      |
+-------------+---------------------------------------------------------+


#Tarkistetaan identity palvelun toimivuus,
#Tyhjennä  OS_SERVICE_TOKEN ja OS_SERVICE_ENDPOINT environment variables:
unset OS_SERVICE_TOKEN
unset OS_SERVICE_ENDPOINT
#Pyydetään valtuutusavainta palvelullta käyttäen admin käyttäjää:
keystone --os-username=admin --os-password='EL1E$Ta!nS+ooM' \
--os-auth-url=http://authserv:35357/v2.0 token-get
+----------+-----------------------------------------------------------------------------------------------------+
| Property |                                                Value                                                |
+----------+-----------------------------------------------------------------------------------------------------+
| expires  | 2014-02-18T17:04:05Z                                                                                |
|    id    |
MIIC8QYJKoZIhvcNAQcCoIIC4jCCAt4CAQExCTAHBgUrDgMCGjCCAUcGCSqGSIb3
DQEHAaCCATgEggE0eyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAx
NC0wMi0xN1QxNzowNDowNS4zODk2NTIiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE4VD
E3OjA0OjA1WiIsICJpZCI6ICJwbGFjZWhvbGRlciJ9LCAic2VydmljZUNhdGFsb2ciOiBb
XSwgInVzZXIiOiB7InVzZXJuYW1lIjogImFkbWluIiwgInJvbGVzX2xpbmtzIjogW10sICJp
ZCI6ICJkODMzMmE2NGFmZGY0ZjIwOTJlMzRlMmNjNzlkOGIyZSIsICJyb2xlcyI6IFtdL
CAibmFtZSI6ICJhZG1pbiJ9LCAibWV0YWRhdGEiOiB7ImlzX2FkbWluIjogMCwgInJvb
GVzIjogW119fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQID
AVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0MRgwFgYD
VQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBB
QAEggEAp7dqhpC4h35I2VBaJFqjE6VdcXWhkpSVhEHJMDYMZ94MmMZGSCRJu3E
eEeqUcFj56LanYiKtpkoL6ztAZhHJeS3qsK+NYLgoRAVOpLWyTlf9xTbjoKkWFFPBDfv
PtlP36fVH4SwJnAFLEI5Z3CYRLCR4oOmvN7culpWjEXMBGVqNDB4Y2peCcBRmBf
Qmq350U37JHGEvcYVb0fcmAvzFr8q7D4giJcNgwaZK5TSP0lOm0ySgb7SL8cci7hqqf
6Xq947mB5p92sJ0rhAl7+AWBeqmIFkvcAEUS5-WZDMGzLZ-
RBookwkUaaOiTBJQO1+bfF8VXT+vabZiHQHLKurlg==                                                                |
| user_id  | d8332a64afdf4f2092e34e2cc79d8b2e                                                                     |
+----------+-----------------------------------------------------------------------------------------------------+
```

Liite 3. account-server.conf

```
##StorageServ01 account-server.conf mallina
[DEFAULT]
bind_ip = 172.16.1.2
bind_port = 6002
# bind_timeout = 30
# backlog = 4096
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = true
disable_fallocate = false
# Use an integer to override the number of pre-forked processes that will
# accept connections.
workers = 2
# Maximum concurrent requests per worker
# max_clients = 1024
# You can specify default log routing here if you want:
log_name = swift
log_facility = LOG_LOCAL0
log_level = DEBUG
# log_address = /var/log/swift
# comma separated list of functions to call to setup custom log handlers.
# functions get passed: conf, name, log_to_console, log_route, fmt, logger,
# adapted_logger
# log_custom_handlers =
# If set, log_udp_host will override log_address
# log_udp_host =
# log_udp_port = 514
# You can enable StatsD logging here:
# log_statsd_host = localhost
# log_statsd_port = 8125
# log_statsd_default_sample_rate = 1.0
# log_statsd_sample_rate_factor = 1.0
# log_statsd_metric_prefix =
# If you don't mind the extra disk space usage in overhead, you can turn this
# on to preallocate disk space with SQLite databases to decrease fragmentation.
# db_preallocation = off
# eventlet_debug = false
# You can set fallocate_reserve to the number of bytes you'd like fallocate to
# reserve, whether there is space for the given file size or not.
# fallocate_reserve = 0
[pipeline:main]
pipeline = healthcheck recon account-server
[app:account-server]
use = egg:swift#account
# You can override the default log routing for this app here:
# set log_name = account-server
# set log_facility = LOG_LOCAL0
# set log_level = INFO
# set log_requests = true
```

```
# set log_address = /dev/log
# auto_create_account_prefix = .
# Configure parameter for creating specific server
# To handle all verbs, including replication verbs, do not specify
# "replication_server" (this is the default). To only handle replication,
# set to a True value (e.g. "True" or "1"). To handle only non-replication
# verbs, set to "False". Unless you have a separate replication network, you
# should not specify any value for "replication_server".
# replication_server = false
[filter:healthcheck]
use = egg:swift#healthcheck
# An optional filesystem path, which if present, will cause the healthcheck
# URL to return "503 Service Unavailable" with a body of "DISABLED BY FILE"
# disable_path =
[filter:recon]
use = egg:swift#recon
recon_cache_path = /var/swift/recon
[account-replicator]
# You can override the default log routing for this app here (don't use set!):
# log_name = account-replicator
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
# vm_test_mode = no
# per_diff = 1000
# max_diffs = 100
# concurrency = 8
# interval = 30
# How long without an error before a node's error count is reset. This will
# also be how long before a node is reenabled after suppression is triggered.
# error_suppression_interval = 60
# How many errors can accumulate before a node is temporarily ignored.
# error_suppression_limit = 10
# node_timeout = 10
# conn_timeout = 0.5
# The replicator also performs reclamation
# reclaim_age = 604800
# Time in seconds to wait between replication passes
# run_pause = 30
recon_cache_path = /var/swift/recon
[account-auditor]
# You can override the default log routing for this app here (don't use set!):
# log_name = account-auditor
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
# Will audit each account at most once per interval
# interval = 1800
# log_facility = LOG_LOCAL0
# log_level = INFO
# accounts_per_second = 200
recon_cache_path = /var/swift/recon
[account-reaper]
```

```
# You can override the default log routing for this app here (don't use set!):
# log_name = account-reaper
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
# concurrency = 25
# interval = 3600
# node_timeout = 10
# conn_timeout = 0.5
# Normally, the reaper begins deleting account information for deleted accounts
# immediately; you can set this to delay its work however. The value is in
# seconds; 2592000 = 30 days for example.
# delay_reaping = 0
# If the account fails to be be reaped due to a persistent error, the
# account reaper will log a message such as:
# Account <name> has not been reaped since <date>
# You can search logs for this message if space is not being reclaimed
# after you delete account(s).
# Default is 2592000 seconds (30 days). This is in addition to any time
# requested by delay_reaping.
# reap_warn_after = 2592000
```

Liite 4. container-server.conf

```
[DEFAULT]
bind_ip = 172.16.1.2
bind_port = 6001
# bind_timeout = 30
# backlog = 4096
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = true
disable_fallocate = false
#
# Use an integer to override the number of pre-forked processes that will
# accept connections.
workers = 2
#
# Maximum concurrent requests per worker
# max_clients = 1024
#
# This is a comma separated list of hosts allowed in the X-Container-Sync-To
# field for containers. This is the old-style of using container sync. It is
# strongly recommended to use the new style of a separate
# container-sync-realms.conf -- see container-sync-realms.conf-sample
# allowed_sync_hosts = 127.0.0.1
#
# You can specify default log routing here if you want:
log_name = swift
log_facility = LOG_LOCAL0
log_level = DEBUG
# log_address = /var/log/swift
#
# comma separated list of functions to call to setup custom log handlers.
# functions get passed: conf, name, log_to_console, log_route, fmt, logger,
# adapted_logger
# log_custom_handlers =
#
# If set, log_udp_host will override log_address
# log_udp_host =
# log_udp_port = 514
#
# You can enable StatsD logging here:
# log_statsd_host = localhost
# log_statsd_port = 8125
# log_statsd_default_sample_rate = 1.0
# log_statsd_sample_rate_factor = 1.0
# log_statsd_metric_prefix =
#
# If you don't mind the extra disk space usage in overhead, you can turn this
# on to preallocate disk space with SQLite databases to decrease fragmentation.
```

```
# db_preallocation = off
#
# eventlet_debug = false
#
# You can set fallocate_reserve to the number of bytes you'd like fallocate to
# reserve, whether there is space for the given file size or not.
# fallocate_reserve = 0

[pipeline:main]
pipeline = healthcheck recon container-server

[app:container-server]
use = egg:swift#container
# You can override the default log routing for this app here:
# set log_name = container-server
# set log_facility = LOG_LOCAL0
# set log_level = INFO
# set log_requests = true
# set log_address = /dev/log
#
# node_timeout = 3
# conn_timeout = 0.5
# allow_versions = false
# auto_create_account_prefix = .
#
# Configure parameter for creating specific server
# To handle all verbs, including replication verbs, do not specify
# "replication_server" (this is the default). To only handle replication,
# set to a True value (e.g. "True" or "1"). To handle only non-replication
# verbs, set to "False". Unless you have a separate replication network, you
# should not specify any value for "replication_server".
# replication_server = false

[filter:healthcheck]
use = egg:swift#healthcheck
# An optional filesystem path, which if present, will cause the healthcheck
# URL to return "503 Service Unavailable" with a body of "DISABLED BY FILE"
# disable_path =

[filter:recon]
use = egg:swift#recon
recon_cache_path = /var/swift/recon

[container-replicator]
# You can override the default log routing for this app here (don't use set!):
# log_name = container-replicator
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
# vm_test_mode = no
# per_diff = 1000
# max_diffs = 100
```

```
# concurrency = 8
# interval = 30
# node_timeout = 10
# conn_timeout = 0.5
#
# The replicator also performs reclamation
# reclaim_age = 604800
#
# Time in seconds to wait between replication passes
# run_pause = 30
#
recon_cache_path = /var/swift/recon

[container-updater]
# You can override the default log routing for this app here (don't use set!):
# log_name = container-updater
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
# interval = 300
# concurrency = 4
# node_timeout = 3
# conn_timeout = 0.5
#
# slowdown will sleep that amount between containers
# slowdown = 0.01
#
# Seconds to suppress updating an account that has generated an error
# account_suppression_time = 60
#
recon_cache_path = /var/swift/recon

[container-auditor]
# You can override the default log routing for this app here (don't use set!):
# log_name = container-auditor
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
# Will audit each container at most once per interval
# interval = 1800
#
# containers_per_second = 200
recon_cache_path = /var/swift/recon

[container-sync]
# You can override the default log routing for this app here (don't use set!):
# log_name = container-sync
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
```

```
# If you need to use an HTTP Proxy, set it here; defaults to no proxy.
# You can also set this to a comma separated list of HTTP Proxies and they will
# be randomly used (simple load balancing).
# sync_proxy = http://10.1.1.1:8888,http://10.1.1.2:8888
#
# Will sync each container at most once per interval
# interval = 300
#
# Maximum amount of time to spend syncing each container per pass
# container_time = 60
```

Liite 5. object-server.conf

```
[DEFAULT]
bind_ip = 172.16.1.2
bind_port = 6000
# bind_timeout = 30
# backlog = 4096
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = true
disable_fallocate = false
# expiring_objects_container_divisor = 86400
#
# Use an integer to override the number of pre-forked processes that will
# accept connections.
workers = 2
#
# Maximum concurrent requests per worker
# max_clients = 1024
#
# You can specify default log routing here if you want:
log_name = swift
log_facility = LOG_LOCAL0
log_level = DEBUT
# log_address = /var/log/swift
#
# comma separated list of functions to call to setup custom log handlers.
# functions get passed: conf, name, log_to_console, log_route, fmt, logger,
# adapted_logger
# log_custom_handlers =
#
# If set, log_udp_host will override log_address
# log_udp_host =
# log_udp_port = 514
#
# You can enable StatsD logging here:
# log_statsd_host = localhost
# log_statsd_port = 8125
# log_statsd_default_sample_rate = 1.0
# log_statsd_sample_rate_factor = 1.0
# log_statsd_metric_prefix =
#
# eventlet_debug = false
#
# You can set fallocate_reserve to the number of bytes you'd like fallocate to
# reserve, whether there is space for the given file size or not.
# fallocate_reserve = 0
#
```

```
# Time to wait while attempting to connect to another backend node.
# conn_timeout = 0.5
# Time to wait while sending each chunk of data to another backend node.
# node_timeout = 3
# Time to wait while receiving each chunk of data from a client or another
# backend node.
# client_timeout = 60
#
# network_chunk_size = 65536
# disk_chunk_size = 65536

[pipeline:main]
pipeline = healthcheck recon object-server

[app:object-server]
use = egg:swift#object
# You can override the default log routing for this app here:
# set log_name = object-server
# set log_facility = LOG_LOCAL0
# set log_level = INFO
# set log_requests = true
# set log_address = /dev/log
#
# max_upload_time = 86400
# slow = 0
#
# Objects smaller than this are not evicted from the buffercache once read
# keep_cache_size = 5424880
#
# If true, objects for authenticated GET requests may be kept in buffer cache
# if small enough
# keep_cache_private = false
#
# on PUTs, sync data every n MB
# mb_per_sync = 512
#
# Comma separated list of headers that can be set in metadata on an object.
# This list is in addition to X-Object-Meta-* headers and cannot include
# Content-Type, etag, Content-Length, or deleted
# allowed_headers = Content-Disposition, Content-Encoding, X-Delete-At, X-Object-
Manifest, X-Static-Large-Object
#
# auto_create_account_prefix = .
#
# A value of 0 means "don't use thread pools". A reasonable starting point is
# 4.
# threads_per_disk = 0
#
# Configure parameter for creating specific server
# To handle all verbs, including replication verbs, do not specify
# "replication_server" (this is the default). To only handle replication,
# set to a True value (e.g. "True" or "1"). To handle only non-replication
# verbs, set to "False". Unless you have a separate replication network, you
```

```
# should not specify any value for "replication_server".
# replication_server = false
#
# Set to restrict the number of concurrent incoming REPLICATION requests
# Set to 0 for unlimited
# Note that REPLICATION is currently an ssync only item
# replication_concurrency = 4
#
# Restricts incoming REPLICATION requests to one per device,
# replication_currency above allowing. This can help control I/O to each
# device, but you may wish to set this to False to allow multiple REPLICATION
# requests (up to the above replication_concurrency setting) per device.
# replication_one_per_device = True
#
# Number of seconds to wait for an existing replication device lock before
# giving up.
# replication_lock_timeout = 15
#
# These next two settings control when the REPLICATION subrequest handler will
# abort an incoming REPLICATION attempt. An abort will occur if there are at
# least threshold number of failures and the value of failures / successes
# exceeds the ratio. The defaults of 100 and 1.0 means that at least 100
# failures have to occur and there have to be more failures than successes for
# an abort to occur.
# replication_failure_threshold = 100
# replication_failure_ratio = 1.0

[filter:healthcheck]
use = egg:swift#healthcheck
# An optional filesystem path, which if present, will cause the healthcheck
# URL to return "503 Service Unavailable" with a body of "DISABLED BY FILE"
# disable_path =

[filter:recon]
use = egg:swift#recon
recon_cache_path = /var/swift/recon
#recon_lock_path = /var/lock

[object-replicator]
# You can override the default log routing for this app here (don't use set!):
# log_name = object-replicator
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
# vm_test_mode = no
# daemonize = on
# run_pause = 30
# concurrency = 1
# stats_interval = 300
#
# The sync method to use; default is rsync but you can use ssync to try the
# EXPERIMENTAL all-swift-code-no-rsync-callouts method. Once verified as stable
```

```
# and nearly as efficient (or moreso) than rsync, we plan to deprecate rsync so
# we can move on with more features for replication.
# sync_method = rsync
#
# max duration of a partition rsync
# rsync_timeout = 900
#
# bandwith limit for rsync in kB/s. 0 means unlimited
# rsync_bwlimit = 0
#
# passed to rsync for io op timeout
# rsync_io_timeout = 30
#
# node_timeout = <whatever's in the DEFAULT section or 10>
# max duration of an http request; this is for REPLICATE finalization calls and
# so should be longer than node_timeout
# http_timeout = 60
#
# attempts to kill all workers if nothing replicates for lockup_timeout seconds
# lockup_timeout = 1800
#
# The replicator also performs reclamation
# reclaim_age = 604800
#
# ring_check_interval = 15
recon_cache_path = /var/swift/recon
#
# limits how long rsync error log lines are
# 0 means to log the entire line
# rsync_error_log_line_length = 0

[object-updater]
# You can override the default log routing for this app here (don't use set!):
# log_name = object-updater
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
# interval = 300
# concurrency = 1
# node_timeout = <whatever's in the DEFAULT section or 10>
# slowdown will sleep that amount between objects
# slowdown = 0.01
#
recon_cache_path = /var/swift/recon

[object-auditor]
# You can override the default log routing for this app here (don't use set!):
# log_name = object-auditor
# log_facility = LOG_LOCAL0
# log_level = INFO
# log_address = /dev/log
#
```

```
# files_per_second = 20
# bytes_per_second = 10000000
# log_time = 3600
# zero_byte_files_per_second = 50
recon_cache_path = /var/swift/recon

# Takes a comma separated list of ints. If set, the object auditor will
# increment a counter for every object whose size is <= to the given break
# points and report the result after a full scan.
# object_size_stats =
```

Liite 6. proxy-server.conf

```
[DEFAULT]
bind_port = 8080
user = swift

#Label used when logging
set log_name = swift
#Syslog log facility
set log_facility = LOG_LOCAL0
#Log level
set log_level = DEBUG

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = Member,admin,swiftoperator

[filter:authtoken]
paste.filter_factory = keystoneclient.middleware.auth_token:filter_factory

# Delaying the auth decision is required to support token-less
# usage for anonymous referrers ('.r:*').
delay_auth_decision = true

# cache directory for signing certificate
signing_dir = /var/cache/swift/keystone-signing

# auth_* settings refer to the Keystone server
auth_protocol = http
auth_host = authserv
auth_port = 35357
auth_uri = http://authserv:5000/

# the same admin_token as provided in keystone.conf
admin_token = T4lM3sCa71N6LyCA

# the service tenant and swift userid and password created in Keystone
admin_tenant_name = service
admin_user = swift
admin_password = OMAp4ssu1234#
cache = swift.cache
include_service_catalog = False
```

```
[filter:cache]
use = egg:swift#memcache
memcache_servers = 172.16.1.1:11211

[filter:catch_errors]
use = egg:swift#catch_errors

[filter:healthcheck]
use = egg:swift#healthcheck
```

Liite 7. Renkaiden luonti

```
#Luodaan account, container, and object renkaiden luontitiedostot
cd /etc/swift
sudo swift-ring-builder account.builder create 9 3 1
sudo swift-ring-builder container.builder create 9 3 1
sudo swift-ring-builder object.builder create 9 3 1

#Määritellään laitteet renkaisiin
#StoreServ01
sudo swift-ring-builder account.builder add z1-172.16.1.2:6002R172.16.2.2:6005/sdb1
100
sudo swift-ring-builder container.builder add z1-
172.16.1.2:6001R172.16.2.2:6005/sdb1 100
sudo swift-ring-builder object.builder add z1-172.16.1.2:6000R172.16.2.2:6005/sdb1
100
#StoreServ02
sudo swift-ring-builder account.builder add z2-172.16.1.3:6002R172.16.2.3:6005/sdb1
100
sudo swift-ring-builder container.builder add z2-
172.16.1.3:6001R172.16.2.3:6005/sdb1 100
sudo swift-ring-builder object.builder add z2-172.16.1.3:6000R172.16.2.3:6005/sdb1
100
#StoreServ03
sudo swift-ring-builder account.builder add z3-172.16.1.4:6002R172.16.2.4:6005/sdb1
100
sudo swift-ring-builder container.builder add z3-
172.16.1.4:6001R172.16.2.4:6005/sdb1 100
sudo swift-ring-builder object.builder add z3-172.16.1.4:6000R172.16.2.4:6005/sdb1
100
#StoreServ04
sudo swift-ring-builder account.builder add z4-172.16.1.5:6002R172.16.2.5:6005/sdb1
100
sudo swift-ring-builder container.builder add z4-
172.16.1.5:6001R172.16.2.5:6005/sdb1 100
sudo swift-ring-builder object.builder add z4-172.16.1.5:6000R172.16.2.5:6005/sdb1
100
#StoreServ05
sudo swift-ring-builder account.builder add z5-172.16.1.6:6002R172.16.2.6:6005/sdb1
100
sudo swift-ring-builder container.builder add z5-
172.16.1.6:6001R172.16.2.6:6005/sdb1 100
sudo swift-ring-builder object.builder add z5-
172.16.1.6:6000R172.16.2.6:6005/sdb1 100

#Tarkistetaan renkaiden luontitiedostojen sisältö
sudo swift-ring-builder account.builder
sudo swift-ring-builder container.builder
sudo swift-ring-builder object.builder
```

```
#Suoritetaan tasoitus
sudo swift-ring-builder account.builder rebalance
sudo swift-ring-builder container.builder rebalance
sudo swift-ring-builder object.builder rebalance

#Kopioidaan account.ring.gz, container.ring.gz, and object.ring.gz tiedostot jokaiselle Proxy and Storage nodelle /etc/swift.
sudo scp jakke@172.16.1.1:/etc/swift/*.gz /etc/swift/

#Vaihdetaan tiedoston omistava käyttäjä
sudo chown -R swift:swift /etc/swift
```

Liite 8. swift-openrc.sh

```
jakke@proxyserv:~$ cat swift-openrc.sh
export OS_USERNAME=swift
export OS_PASSWORD=OMAp4ssu1234#
export OS_TENANT_NAME=service
export OS_AUTH_URL=http://authserv:5000/v2.0
```

Liite 9. Testaus tulosteet ja kuvakaappaukset

##Testausta proxyserv palvelimelta
jakke@proxyserv:~$ source demo_testrc.sh
jakke@proxyserv:~$ swift --debubg stat
REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent: python-keystoneclient"
DEBUG:keystoneclient.httpclient:REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent: python-keystoneclient"
REQ BODY: {"auth": {"tenantName": "demo", "passwordCredentials": {"username": "test", "password": "munOMAs4l4s4n4#"}}}
DEBUG:keystoneclient.httpclient:REQ BODY: {"auth": {"tenantName": "demo", "passwordCredentials": {"username": "test", "password": "munOMAs4l4s4n4#"}}}
INFO:urllib3.connectionpool:Starting new HTTP connection (1): authserv
DEBUG:urllib3.connectionpool:"POST /v2.0/tokens HTTP/1.1" 200 3206
RESP: [200] CaseInsensitiveDict({'date': 'Mon, 17 Feb 2014 18:20:38 GMT', 'vary': 'X-Auth-Token', 'content-length': '3206', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-17T18:20:38.839797", "expires": "2014-02-18T18:20:38Z", "id":
"MIIGHgYJKoZIhvcNAQcCoIIGDzCCBgsCAQExCTAHBgUrDgMCGjCCBHQGCSqGSIb3DQEHAaCCBGUEggRheyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi0xN1QxODoyMDozOC44Mzk3OTciLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE4VDE4OjIwOjM4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIlN3aWZ0IENsaWVudCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogIjYzZjIzZDRkNDE1MjQzZjViNWQ5N2FjNTE5NzU2ODE1IiwgIm5hbWUiOiAiZGVtbyJ9fSwgInNlcnZpY2VDYXRhbG9nIjogW3siZW5kcG9pbnRzIjogW3siYWRtaW5VUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfcyIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9wcm94eXNlcnY6ODA4MC92MSIsICJpZCI6ICIyOTQxODNkOGY0ZGQ0OTlkOTBlZTUwZWMzZjg4NWYzMyIsICJwdWJsaWNVUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfNjNmMjNkNGQ0MTUyNDNmNWI1ZDk3YWM1MTk3NTY4MTUifV0sICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAib2JqZWN0LXN0b3JlIiwgIm5hbWUiOiAic3dpZnQifSwgeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6ICJodHRwOi8vYXV0aHNlcnY6MzUzNTcvdjIuMCIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAiLCAiaWQiOiAiMjkyZTNmNzdjOTA5NGJmYjhiODk1MWNmNWY3NThkMGMiLCAicHVibGljVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAifV0sICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAiaWRlbnRpdHkiLCAibmFtZSI6ICJrZXlzdG9uZSJ9XSwgInVzZXIiOiB7InVzZXJuYW1lIjogInRlc3QiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImQwMjRmNzE5ZjQ5MTQxMjdhOTg3OTdmYzAwYTNmNzUyIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9LCB7Im5hbWUiOiAiTWVtYmVyIn1dLCAibmFtZSI6ICJ0ZXN0In0sICJtZXRhZGF0YSI6IHsiaXNfYWRtaW4iOiAwLCAicm9sZXMiOiBbIjkxZWZjNzIyOTIxMTQxNTc5N2ZjNWMyZWRjZGQwY2IxIiwgIjVhYTg3NjQ1NDAwYzRiNDBiZGE2ZjkyZGU2MGIwZTE2Il19fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0MRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAil7vukBGruajljX1OuDxcHkYkSPJzpt-QkFD4SpFPgplj8hKoiuXlleajwcSasZelw9SQBwNRyWN-pGqLHOB9P6KnZt+7equ9xF-EhAR-tiDZxmzdcH5HBWortELx86J1mKV-LTZ3wmUoezQVceNF39X+01Xm4gsUdh57UdSe0jHDxemcTs-YASfNOyX913Pyt3HpmV+59waYGZhp+sSqObo8O8Qcaswpr5d3NbRcpuqk0m0eiPFRQjF7GrvDRXPBfBgIYwERZ0MrX5ltc1dnkMLm-

kAxNuiGkwnX85j9sYygdagIItRlUboxW7EPGnj9jsFkRFGmoGNheTlMe7iHw==", "tenant": {"description": "Swift Client", "enabled": true, "id": "63f23d4d415243f5b5d97ac519756815", "name": "demo"}}, "serviceCatalog": [{"endpoints": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne", "internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33", "publicURL": "http://proxyserv:8080/v1/AUTH_63f23d4d415243f5b5d97ac519756815"}], "endpoints_links": [], "type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL": "http://authserv:35357/v2.0", "region": "regionOne", "internalURL": "http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c", "publicURL": "http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "keystone"}], "user": {"username": "test", "roles_links": [], "id": "d024f719f4914127a98797fc00a3f752", "roles": [{"name": "admin"}, {"name": "Member"}], "name": "test"}, "metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1", "5aa87645400c4b40bda6f92de60b0e16"]}}}
DEBUG:keystoneclient.httpclient:RESP: [200] CaseInsensitiveDict({'date': 'Mon, 17 Feb 2014 18:20:38 GMT', 'vary': 'X-Auth-Token', 'content-length': '3206', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-17T18:20:38.839797", "expires": "2014-02-18T18:20:38Z", "id": "MIIGHgYJKoZIhvcNAQcCoIIGDzCCBgsCAQExCTAHBgUrDgMCGjCCBHQGCSqGSIb3DQEHAaCCBGUEggRheyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjAxNC0wMi0xN1QxODoyMDozOC44Mzk3OTciLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE4VDE4OjIwOjM4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaXB0aW9uIjogIlN3aWZ0IENsaWVudCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogIjYzZjIzZDRkNDE1MjQzZjViNWQ5N2FjNTE5NzU2ODE1IiwgIm5hbWUiOiAiZGVtbyJ9fSwgInNlcnZpY2VDYXRhbG9nIjogW3siZW5kcG9pbnRzIjogW3siYWRtaW5VcmwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfcyIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9wcm94eXNlcnY6ODA4MC92MSIsICJpZCI6ICIyOTQxODNkOGY0ZGQ0OTlkOTBlZTUwZWMzZjg4NWYzMyIsICJwdWJsaWNVUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfNjNmMjNkNGQ0MTUyNDNmNWI1ZDk3YWM1MTk3NTY4MTUifV0sICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAib2JqZWN0LXN0b3JlIiwgIm5hbWUiOiAic3dpZnQifSwgeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6ICJodHRwOi8vYXV0aHNlcnY6MzUzNTcvdjIuMCIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9wcmV4eXNlcnY6NTAwMC92Mi4wIiwgImlkIjogIjMjkyZTNmNzdjOTA5NGJmYjhiODk1MWNmNWY3NThkMGMiLCAicHVibGljVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAifV0sICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOiAiaWRlbnRpdHkiLCAibmFtZSI6ICJrZXlzdG9uZSJ9XSwgInVzZXIiOiB7InVzZXJuYW1lIjogInRlc3QiLCAicm9sZXNfbGlua3MiOiBbXSwgImlkIjogImQwMjRmNzE5ZjQ5MTQxMjdhOTg3OTdmYzAwYTNmNzUyIiwgInJvbGVzIjogW3sibmFtZSI6ICJhZG1pbiJ9LCB7Im5hbWUiOiAiTWVtYmVyIn1dLCAibmFtZSI6ICJ0ZXN0In0sICJtZXRhZGF0YSI6IHsiaXNfYWRtaW4iOiAwLCAicm9sZXMiOiBbIjkxZWZjNzIyOTIxMTQxNTc5N2ZjNWMyZWRjZGQwY2IxIiwgIjVhYTg3NjQ1NDAwYzRiNDBiZGE2ZjkyZGU2MGIwZTE2Il19fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDAYDVQQIDAVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0MRgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvcNAQEBBQAEggEAil7vukBGruajljX1OuDxcHkYkSPJzpt-QkFD4SpFPgplj8hKoiuXlleajwcSasZelw9SQBwNRyWN-pGqLHOB9P6KnZt+7equ9xF-EhAR-tiDZxmzdcH5HBWortELx86J1mKV-LTZ3wmUoezQVceNF39X+01Xm4gsUdh57UdSe0jHDxemcTs-YASfNOyX913Pyt3HpmV+59waYGZhp+sSqObo8O8Qcaswpr5d3NbRcpuqk0m0eiPFRQjF7GrvDRXPBfBgIYwERZ0MrX5ltc1dnkMLm-kAxNuiGkwnX85j9sYygdagIItRlUboxW7EPGnj9jsFkRFGmoGNheTlMe7iHw==", "ten-

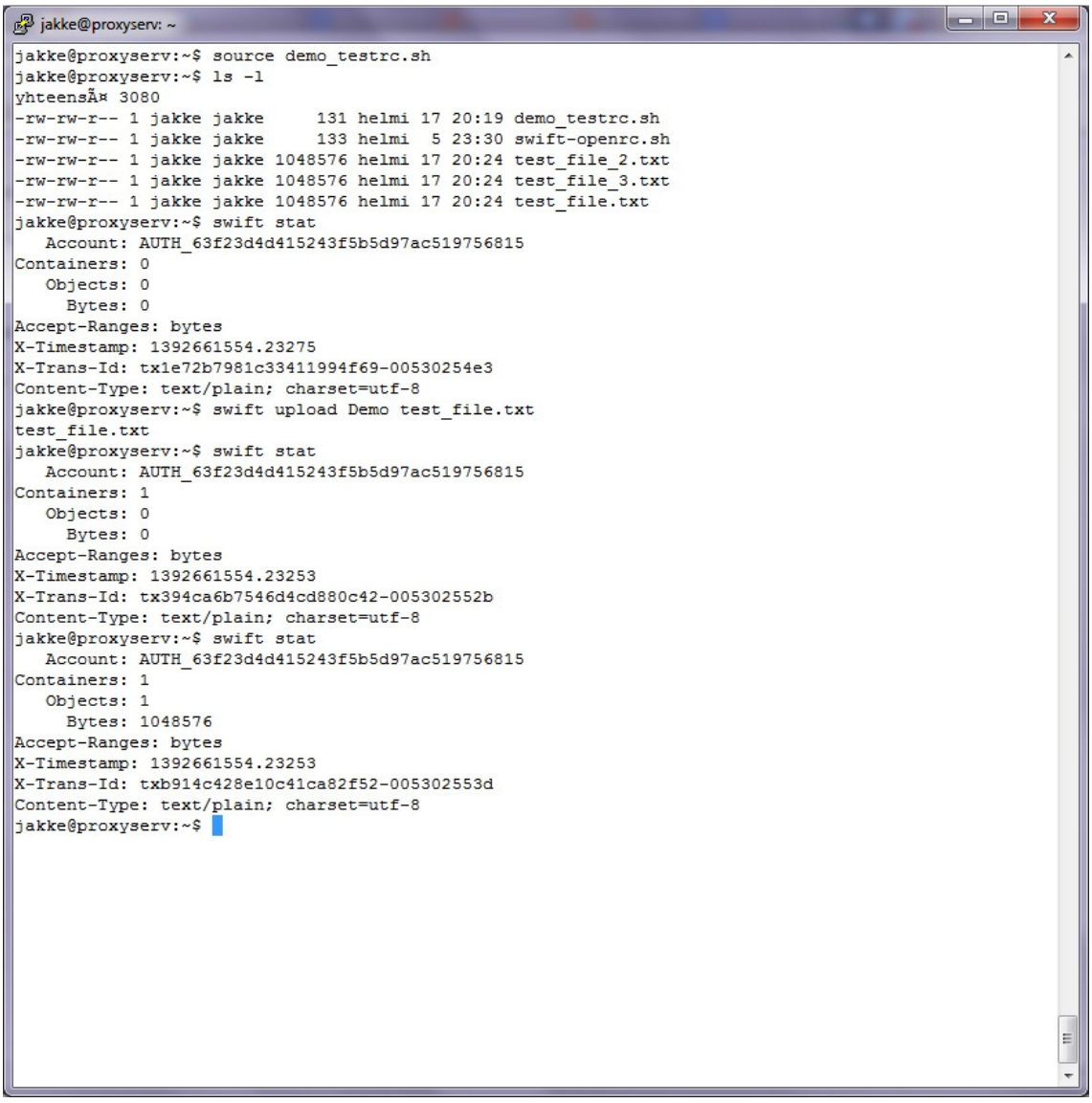
ant": {"description": "Swift Client", "enabled": true, "id":
"63f23d4d415243f5b5d97ac519756815", "name": "demo"}}, "serviceCatalog": [{"end-
points": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL":
"http://proxyserv:8080/v1/AUTH_63f23d4d415243f5b5d97ac519756815"}], "end-
points_links": [], "type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL":
"http://authserv:35357/v2.0", "region": "regionOne", "internalURL":
"http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c", "publicURL":
"http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "key-
stone"}], "user": {"username": "test", "roles_links": [], "id":
"d024f719f4914127a98797fc00a3f752", "roles": [{"name": "admin"}, {"name": "Mem-
ber"}], "name": "test"}, "metadata": {"is_admin": 0, "roles":
["91efc7229211415797fc5c2edcdd0cb1", "5aa87645400c4b40bda6f92de60b0e16"]}}}
DEBUG:swiftclient:REQ: curl -i
http://proxyserv:8080/v1/AUTH_63f23d4d415243f5b5d97ac519756815 -I -H "X-Auth-
Token:
MIIGHgYJKoZIhvcNAQcCoIIGDzCCBgsCAQExCTAHBgUrDgMCGjCCBHQGCSqGSI
b3DQEHAaCCBGUEggRheyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiM
jAxNC0wMi0xN1QxODoyMDozOC44Mzk3OTciLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE
4VDE4OjIwOjM4WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2NyaX
B0aW9uIjogIlN3aWZ0IENsaWVudCIsICJlbmFibGVkIjogdHJ1ZSwgImlkIjogIjYzZjIzZD
RkNDE1MjQzZjViNWQ5N2FjNTE5NzU2ODE1IiwgIm5hbWUiOiAiZGVtbyJ9fSwgInNlc
nZpY2VDYXRhbG9nIjogW3siZW5kcG9pbnRzIjogW3siYWRtaW5VUkwiOiAiaHR0cDov
L3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfcyIsICJyZWdpb24iOiAicmVnaW9uT25lIiwgIm
ludGVybmFsVVJMIjogImh0dHA6Ly9wcm94eXNlcnY6ODA4MC92MSIsICJpZCI6ICIyO
TQxODNkOGY0ZGQ0OTlkOTBlZTUwZWMzZjg4NWYzMyIsICJwdWJsaWNVUkwiOiAi
aHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxL0FVVEhfNjNmMjNkNGQ0MTUyNDNmNWI
1ZDk3YWM1MTk3NTY4MTUifV0sICJlbmRwb2ludHNfbGlua3MiOiBbXSwgInR5cGUiOi
Aib2JqZWN0LXN0b3JlIiwgIm5hbWUiOiAic3dpZnQifSwgeyJlbmRwb2ludHMiOiBbeyJh
ZG1pbiVSTCI6ICJodHRwOi8vYXV0aHNlcnY6MzUzNTcvdjIuMCIsICJyZWdpb24iOiAic
mVnaW9uT25lIiwgImludGVybmFsVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3Yy
LjAiLCAiaWQiOiAiMjkyZTNmNzdjOTA5NGJmYjhiODk1MWNmNWY3NThkMGMiLCAi
cHVibGljVVJMIjogImh0dHA6Ly9hdXRoc2Vydjo1MDAwL3YyLjAifV0sICJlbmRwb2ludH
NfbGlua3MiOiBbXSwgInR5cGUiOiAiaWRlbnRpdHkiLCAibmFtZSI6ICJrZXlzdG9uZSJ9
XSwgInVzZXIiOiB7InVzZXJuYW1lIjogInRlc3QiLCAicm9sZXNfbGlua3MiOiBbXSwgIml
kIjogImQwMjRmNzE5ZjQ5MTQxMjdhOTg3OTdmYzAwYTNmNzUyIiwgInJvbGVzIjogW
3sibmFtZSI6ICJhZG1pbiJ9LCB7Im5hbWUiOiAiTWVtYmVyIn1dLCAibmFtZSI6ICJ0ZX
N0In0sICJtZXRhZGF0YSI6IHsiaXNfYWRtaW4iOiAwLCAicm9sZXMiOiBbIjkxZWZjNzIy
OTIxMTQxNTc5N2ZjNWMyZWRjZGQwY2IxIiwgIjVhYTg3NjQ1NDAwYzRiNDBiZGE2Z
jkyZGU2MGIwZTE2Il19fX0xggGBMIIBfQIBATBcMFcxCzAJBgNVBAYTAlVTMQ4wDA
YDVQQIDAVVbnNldDEOMAwGA1UEBwwFVW5zZXQxDjAMBgNVBAoMBVVuc2V0M
RgwFgYDVQQDDA93d3cuZXhhbXBsZS5jb20CAQEwBwYFKw4DAhowDQYJKoZIhvc
NAQEBBQAEggEAil7vukBGruajljX1OuDxcHkYkSPJzpt-
QkFD4SpFPgpIj8hKoiuXlleajwcSasZelw9SQBwNRyWN-pGqLHOB9P6KnZt+7equ9xF-
EhAR-tiDZxmzdcH5HBWortELx86J1mKV-
LTZ3wmUoezQVceNF39X+01Xm4gsUdh57UdSe0jHDxemcTs-
YASfNOyX913Pyt3HpmV+59waYGZhp+sSqObo8O8Qcaswpr5d3NbRcpuqk0m0eiPFR
QjF7GrvDRXPBfBgIYwERZ0MrX5ltc1dnkMLm-
kAxNuiGkwnX85j9sYygdagIItRlUboxW7EPGnj9jsFkRFGmoGNheTlMe7iHw=="
DEBUG:swiftclient:RESP STATUS: 204
Account: AUTH_63f23d4d415243f5b5d97ac519756815
Containers: 0

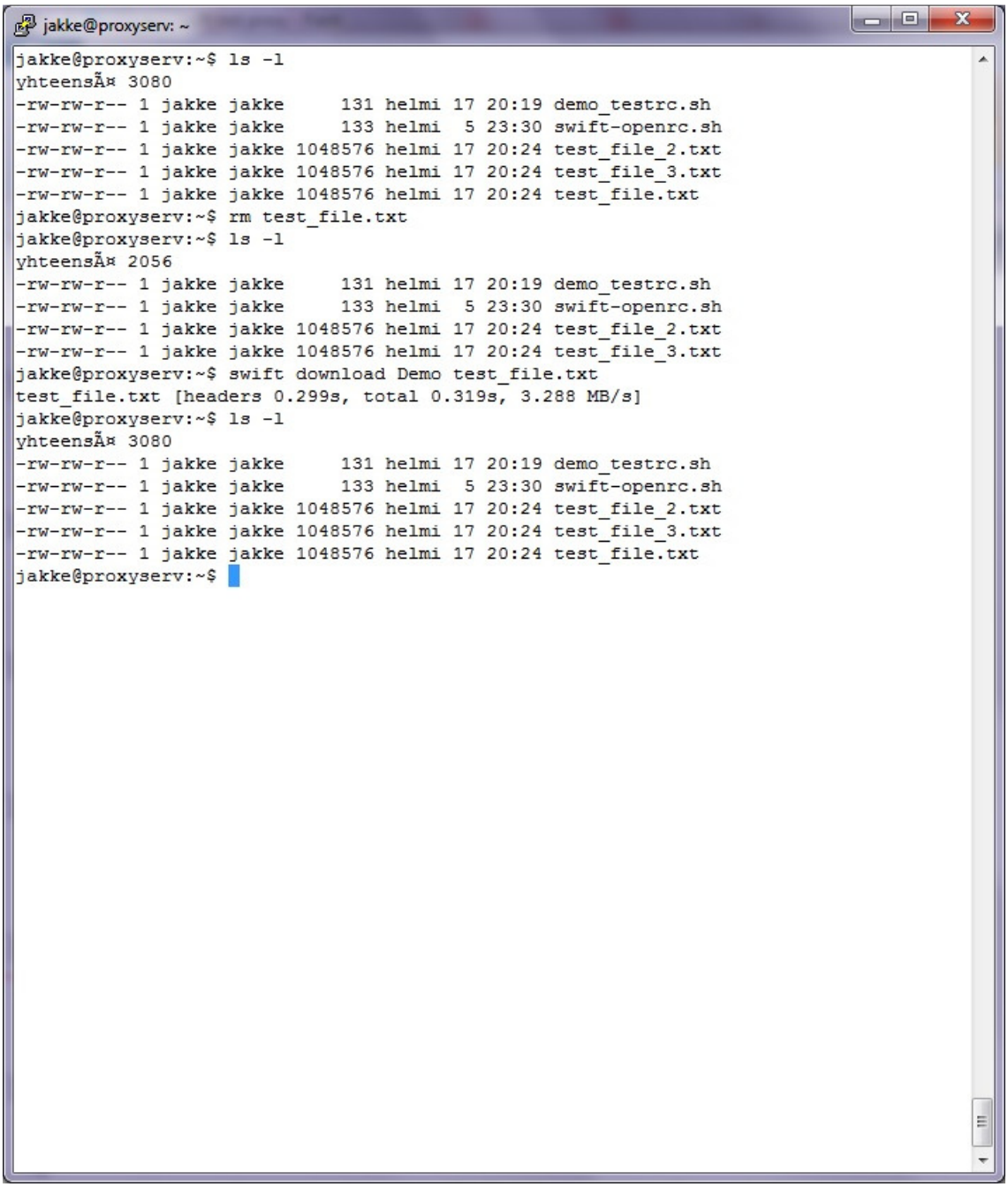
```
   Objects: 0
     Bytes: 0
Content-Type: text/plain; charset=utf-8
X-Timestamp: 1392661238.97235
X-Trans-Id: tx89cf5f9f4ff94a909f465-00530252f6
X-Put-Timestamp: 1392661238.97235
jakke@proxyserv:~$

## Testausta LinuxClient koneelta
LinuxClient:/home/root # source swift-openrc.sh
LinuxClient:/home/root # swift --debug stat
REQ: curl -i -X POST http://authserv:5000/v2.0/tokens -H "Content-Type: applica-
tion/json" -H "User-Agent: python-keystoneclient"
DEBUG:keystoneclient.httpclient:REQ: curl -i -X POST
http://authserv:5000/v2.0/tokens -H "Content-Type: application/json" -H "User-Agent:
python-keystoneclient"
REQ BODY: {"auth": {"tenantName": "service", "passwordCredentials": {"username":
"swift", "password": "OMAp4ssu1234#"}}}
DEBUG:keystoneclient.httpclient:REQ BODY: {"auth": {"tenantName": "service",
"passwordCredentials": {"username": "swift", "password": "OMAp4ssu1234#"}}}
Starting new HTTP connection (1): authserv
INFO:requests.packages.urllib3.connectionpool:Starting new HTTP connection (1):
authserv
"POST /v2.0/tokens HTTP/1.1" 200 3089
DEBUG:requests.packages.urllib3.connectionpool:"POST /v2.0/tokens HTTP/1.1" 200
3089
RESP: [200] CaseInsensitiveDict({'date': 'Mon, 17 Feb 2014 18:41:37 GMT', 'vary': 'X-
Auth-Token', 'content-length': '3089', 'content-type': 'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-17T18:41:37.543481", "ex-
pires": "2014-02-18T18:41:37Z", "id":
"MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb
3DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMj
AxNC0wMi0xN1QxODo0MTozNy41NDM0ODEiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE
4VDE4OjQxOjM3WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2Ny
aXB0aW9uIjogIlNlcnZpY2UgVGVuYW50IiwgImVuYWJsZWQiOiB0cnVlLCAiaWQiOiAi
MDRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjc0NjciLCAibmFtZSI6ICJzZXJ2aW
NlIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6I
CJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25
PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxIiwgIml
kIjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTBlYzNmODg1ZjMzIiwgInB1YmxpY1VST
CI6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF8wNGZlZGE3YWRkOTQ0O
GM3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAid
HlwZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBvaW50c
yI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvb
iI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUw
MDAvdjIuMCIsICJpZCI6ICIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1Zjc1OGQwYy
IsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVu
ZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0
b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOi
BbXSwgImlkIjogIjA1MzQ3NDg4NjVjZTQ0YTA5ZDZlZWE0YTVhODEzZmIyIiwgInJvbGVzI
jogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAic3dpZnQifSwgIm1ldGFkYXRhIjog
eyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5MjExNDE1Nzk3ZmM1YzJlZG
NkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgN
```

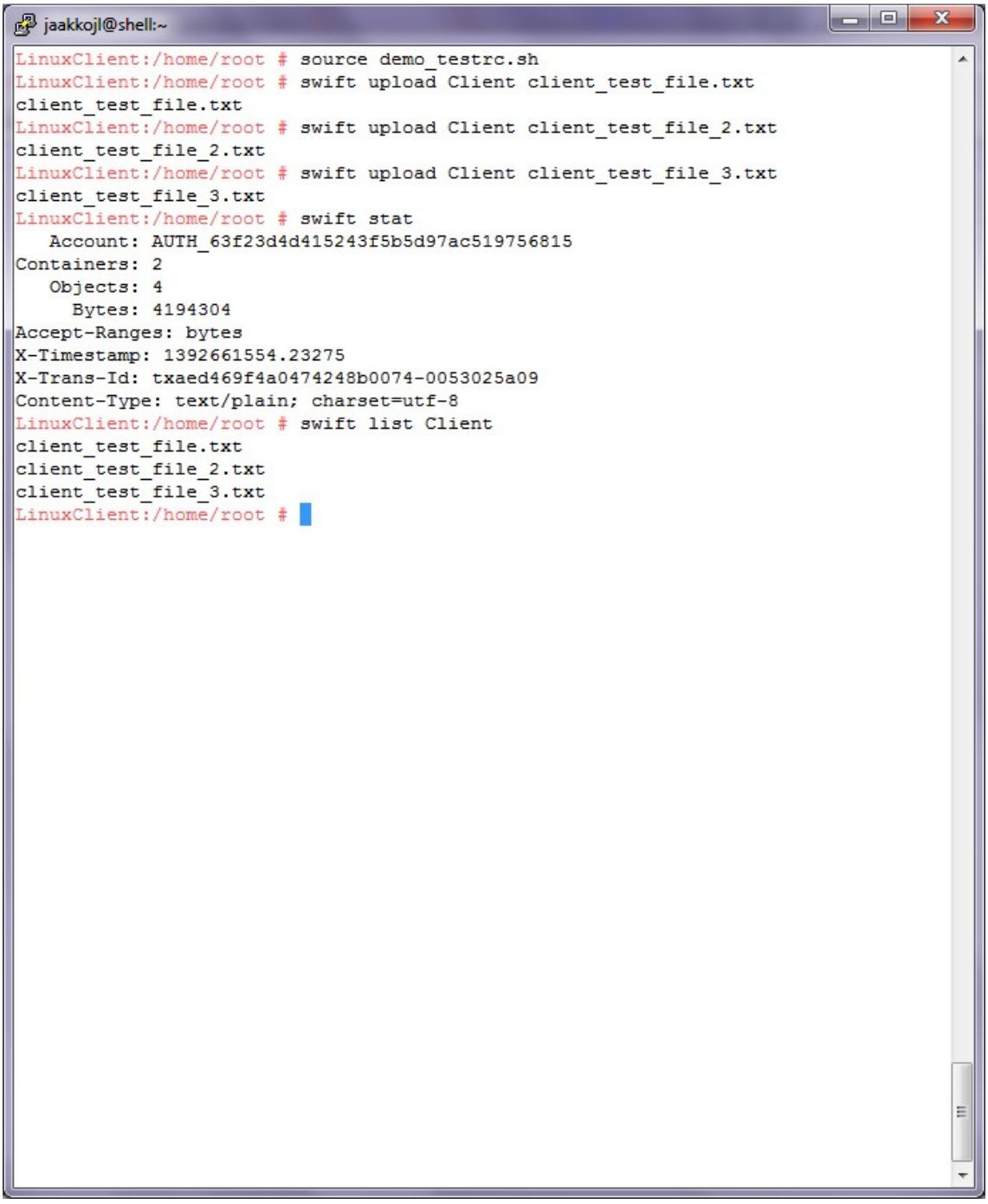
VBAgMBVVuc2V0MQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxG
DAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0B
AQEFAASCAQCsqwKzb1tDuldu1UP9V3aT9vE4IvrQOkx1kBRdukGzAHjhrWh7nB9su
RQPo7fCzlcE+HHQpgcTI1baaTShI716vn9MRgEVumdpdZcVuYjPzjQ4dl5lr0-
qQwuDx3NKs3X4JfCtzkjXKIJroocQvNi3KauL8fpK8NjOZoPlUoLFs9BwMGrp2OxuCNX
l3+zyOqk5BHdj6vxnqTx8tX4IsfDnhoiJvJsRmG0Tbrk9slGCkU8IH8GOV5nvDScmbS+j
mpXe6PqhUjcJXtB26VYCAZam-
BkkDz99Fe0YSx0di6QscM8q0IZnJBhpP6a6QRRyJ5gHJSWSpwTQ0tcYaQrdZML9",
"tenant": {"description": "Service Tenant", "enabled": true, "id":
"04feda7add9448c7bc2d86fec3c27467", "name": "service"}}, "serviceCatalog": [{"end-
points": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL":
"http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467"}], "end-
points_links": [], "type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL":
"http://authserv:35357/v2.0", "region": "regionOne", "internalURL":
"http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c", "publicURL":
"http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "key-
stone"}], "user": {"username": "swift", "roles_links": [], "id":
"0534748865ce44a09d6eea4a5a813fb2", "roles": [{"name": "admin"}], "name": "swift"},
"metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1"]}}}
DEBUG:keystoneclient.httpclient:RESP: [200] CaseInsensitiveDict({'date': 'Mon, 17 Feb
2014 18:41:37 GMT', 'vary': 'X-Auth-Token', 'content-length': '3089', 'content-type':
'application/json'})
RESP BODY: {"access": {"token": {"issued_at": "2014-02-17T18:41:37.543481", "ex-
pires": "2014-02-18T18:41:37Z", "id":
"MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb
3DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMj
AxNC0wMi0xN1QxODo0MTozNy41NDM0ODEiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE
4VDE4OjQxOjM3WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2Ny
aXB0aW9uIjogIlNlcnZpY2UgVGVuYW50IiwgImVuYWJsZWQiOiB0cnVlLCAiaWQiOiAi
MDRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjc0NjciLCAibmFtZSI6ICJzZXJ2aW
NlIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJhbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6I
CJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25
PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxIiwgIml
kIjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTBlYzNmODg1ZjMzIiwgInB1YmxpY1VST
CI6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF8wNGZlZGE3YWRkOTQ0O
GM3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAid
HlwZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBvaW50c
yI6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvb
iI6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUw
MDAvdjIuMCIsICJpZCI6ICIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1Zjc1OGQwYy
IsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVu
ZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0
b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOi
BbXSwgImlkIjogIjA1MzQ3NDg4NjVjZTQ0YTA5ZDZlZWE0YTVhODEzZmIyIiwgInJvbGVzI
jogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAic3dpZnQifSwgIm1ldGFkYXRhIjog
eyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5MjExNDE1Nzk3ZmM1YzJlZG
NkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgN
VBAgMBVVuc2V0MQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxG
DAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0B
AQEFAASCAQCsqwKzb1tDuldu1UP9V3aT9vE4IvrQOkx1kBRdukGzAHjhrWh7nB9su
RQPo7fCzlcE+HHQpgcTI1baaTShI716vn9MRgEVumdpdZcVuYjPzjQ4dl5lr0-

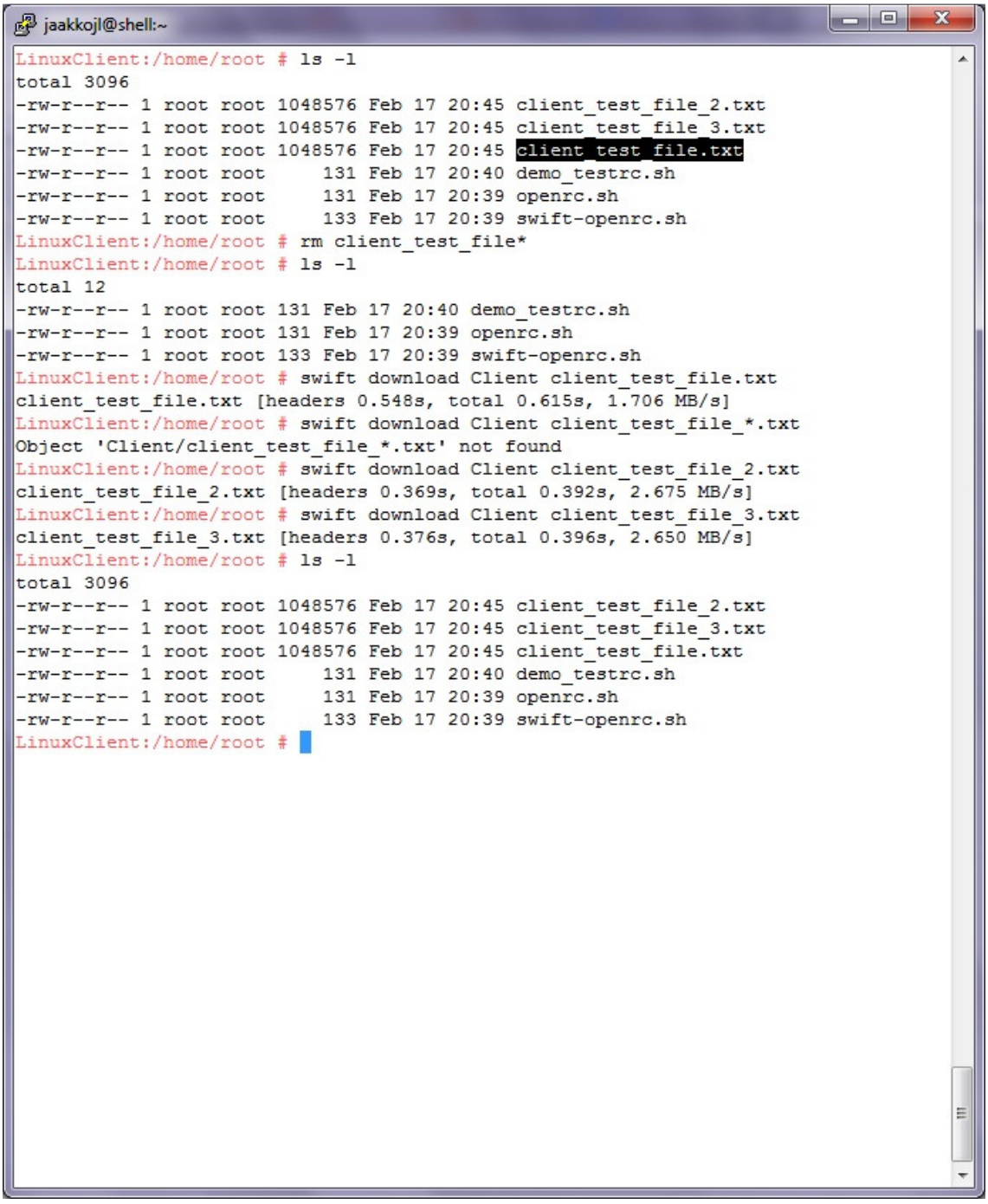
qQwuDx3NKs3X4JfCtzkjXKIJroocQvNi3KauL8fpK8NjOZoPlUoLFs9BwMGrp2OxuCNX
l3+zyOqk5BHdj6vxnqTx8tX4IsfDnhoiJvJsRmG0Tbrk9slGCkU8IH8GOV5nvDScmbS+j
mpXe6PqhUjcJXtB26VYCAZam-
BkkDz99Fe0YSx0di6QscM8q0IZnJBhpP6a6QRRyJ5gHJSWSpwTQ0tcYaQrdZML9",
"tenant": {"description": "Service Tenant", "enabled": true, "id":
"04feda7add9448c7bc2d86fec3c27467", "name": "service"}}, "serviceCatalog": [{"end-
points": [{"adminURL": "http://proxyserv:8080/v1/AUTH_s", "region": "regionOne",
"internalURL": "http://proxyserv:8080/v1", "id": "294183d8f4dd499d90ee50ec3f885f33",
"publicURL":
"http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467"}], "end-
points_links": [], "type": "object-store", "name": "swift"}, {"endpoints": [{"adminURL":
"http://authserv:35357/v2.0", "region": "regionOne", "internalURL":
"http://authserv:5000/v2.0", "id": "292e3f77c9094bfb8b8951cf5f758d0c", "publicURL":
"http://authserv:5000/v2.0"}], "endpoints_links": [], "type": "identity", "name": "key-
stone"}], "user": {"username": "swift", "roles_links": [], "id":
"0534748865ce44a09d6eea4a5a813fb2", "roles": [{"name": "admin"}], "name": "swift"},
"metadata": {"is_admin": 0, "roles": ["91efc7229211415797fc5c2edcdd0cb1"]}}}
DEBUG:iso8601.iso8601:Parsed 2014-02-18T18:41:37Z into {'tz_sign': None,
'second_fraction': None, 'hour': u'18', 'tz_hour': None, 'month': u'02', 'timezone': u'Z',
'second': u'37', 'tz_minute': None, 'year': u'2014', 'separator': u'T', 'day': u'18', 'minute':
u'41'} with default timezone <iso8601.iso8601.Utc object at 0xdff290>
DEBUG:iso8601.iso8601:Got u'2014' for 'year' with default None
DEBUG:iso8601.iso8601:Got u'02' for 'month' with default None
DEBUG:iso8601.iso8601:Got u'18' for 'day' with default None
DEBUG:iso8601.iso8601:Got u'18' for 'hour' with default None
DEBUG:iso8601.iso8601:Got u'41' for 'minute' with default None
DEBUG:iso8601.iso8601:Got u'37' for 'second' with default None
DEBUG:swiftclient:REQ: curl -i
http://proxyserv:8080/v1/AUTH_04feda7add9448c7bc2d86fec3c27467 -I -H "X-Auth-
Token:
MIIF7QYJKoZIhvcNAQcCoIIF3jCCBdoCAQExCTAHBgUrDgMCGjCCBEMGCSqGSIb3
DQEHAaCCBDQEggQweyJhY2Nlc3MiOiB7InRva2VuIjogeyJpc3N1ZWRfYXQiOiAiMjA
xNC0wMi0xN1QxODo0MTozNy41NDM0ODEiLCAiZXhwaXJlcyI6ICIyMDE0LTAyLTE4
VDE4OjQxOjM3WiIsICJpZCI6ICJwbGFjZWhvbGRlciIsICJ0ZW5hbnQiOiB7ImRlc2Nya
XB0aW9uIjogIlNlcnZpY2UgVGVuYW50IiwgImVuYWJsZWQiOiB0cnVlLCAiaWQiOiAiM
DRmZWRhN2FkZDk0NDhjN2JjMmQ4NmZlYzNjMjc0NjciLCAibmFtZSI6ICJzZXJ2aWN
lIn19LCAic2VydmljZUNhdGFsb2ciOiBbeyJlbmRwb2ludHMiOiBbeyJhZG1pblVSTCI6IC
JodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF9zIiwgInJlZ2lvbiI6ICJyZWdpb25P
bmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL3Byb3h5c2Vydjo4MDgwL3YxIiwgImlk
IjogIjI5NDE4M2Q4ZjRkZDQ5OWQ5MGVlNTBlYzNmODg1ZjMzIiwgInB1YmxpY1VSTC
I6ICJodHRwOi8vcHJveHlzZXJ2OjgwODAvdjEvQVVUSF8wNGZlZGE3YWRkOTQ0OG
M3YmMyZDg2ZmVjM2MyNzQ2NyJ9XSwgImVuZHBvaW50c19saW5rcyI6IFtdLCAidHl
wZSI6ICJvYmplY3Qtc3RvcmUiLCAibmFtZSI6ICJzd2lmdCJ9LCB7ImVuZHBvaW50cyI
6IFt7ImFkbWluVVJMIjogImh0dHA6Ly9hdXRoc2VydjozNTM1Ny92Mi4wIiwgInJlZ2lvbiI
6ICJyZWdpb25PbmUiLCAiaW50ZXJuYWxVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUw
MDAvdjIuMCIsICJpZCI6ICIyOTJlM2Y3N2M5MDk0YmZiOGI4OTUxY2Y1Zjc1OGQwYy
IsICJwdWJsaWNVUkwiOiAiaHR0cDovL2F1dGhzZXJ2OjUwMDAvdjIuMCJ9XSwgImVu
ZHBvaW50c19saW5rcyI6IFtdLCAidHlwZSI6ICJpZGVudGl0eSIsICJuYW1lIjogImtleXN0
b25lIn1dLCAidXNlciI6IHsidXNlcm5hbWUiOiAic3dpZnQiLCAicm9sZXNfbGlua3MiOi
BbXSwgImlkIjogIjA1MzQ3NDg4NjVjZTQ0YTA5ZDZlZWE0YTVhODEzZmIyIiwgInJvbGVzI
jogW3sibmFtZSI6ICJhZG1pbiJ9XSwgIm5hbWUiOiAic3dpZnQifSwgIm1ldGFkYXRhIjog
eyJpc19hZG1pbiI6IDAsICJyb2xlcyI6IFsiOTFlZmM3MjI5MjExNDE1Nzk3ZmM1YzJlZG
NkZDBjYjEiXX19fTGCAYEwggF9AgEBMFwwVzELMAkGA1UEBhMCVVMxDjAMBgN

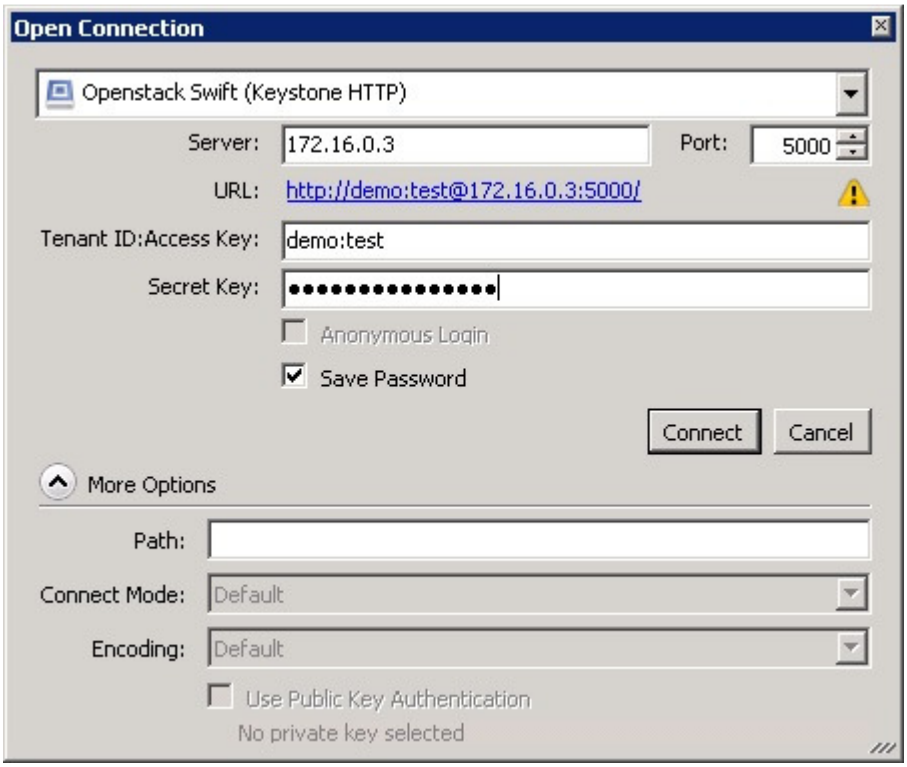
VBAgMBVVuc2V0MQ4wDAYDVQQHDAVVbnNldDEOMAwGA1UECgwFVW5zZXQxG
DAWBgNVBAMMD3d3dy5leGFtcGxlLmNvbQIBATAHBgUrDgMCGjANBgkqhkiG9w0B
AQEFAASCAQCsqwKzb1tDuldu1UP9V3aT9vE4IvrQOkx1kBRdukGzAHjhrWh7nB9su
RQPo7fCzlcE+HHQpgcTI1baaTShI716vn9MRgEVumdpdZcVuYjPzjQ4dl5lr0-
qQwuDx3NKs3X4JfCtzkjXKIJroocQvNi3KauL8fpK8NjOZoPlUoLFs9BwMGrp2OxuCNX
l3+zyOqk5BHdj6vxnqTx8tX4IsfDnhoiJvJsRmG0Tbrk9slGCkU8IH8GOV5nvDScmbS+j
mpXe6PqhUjcJXtB26VYCAZam-
BkkDz99Fe0YSx0di6QscM8q0IZnJBhpP6a6QRRyJ5gHJSWSpwTQ0tcYaQrdZML9"
DEBUG:swiftclient:RESP STATUS: 204
  Account: AUTH_04feda7add9448c7bc2d86fec3c27467
Containers: 3
  Objects: 10
    Bytes: 29067002
Accept-Ranges: bytes
X-Timestamp: 1391636249.21517
X-Trans-Id: tx835859c3dee748c099d50-00530257e1
Content-Type: text/plain; charset=utf-8
LinuxClient:/home/root #

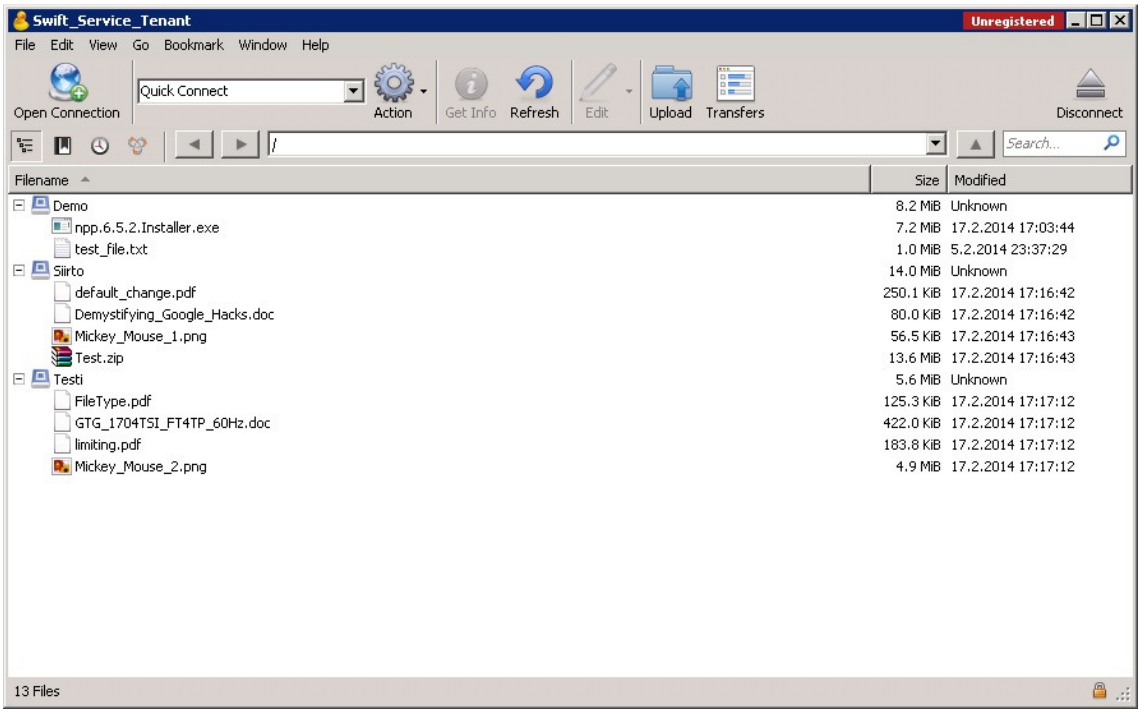

jakke@proxyserv: ~
jakke@proxyserv:~$ source demo_testrc.sh
jakke@proxyserv:~$ ls -l
yhteensÃ¤ 3080
-rw-rw-r-- 1 jakke jakke   131 helmi 17 20:19 demo_testrc.sh
-rw-rw-r-- 1 jakke jakke   133 helmi 5 23:30 swift-openrc.sh
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_2.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_3.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file.txt
jakke@proxyserv:~$ swift stat
Account: AUTH_63f23d4d415243f5b5d97ac519756815
Containers: 0
Objects: 0
Bytes: 0
Accept-Ranges: bytes
X-Timestamp: 1392661554.23275
X-Trans-Id: tx1e72b7981c33411994f69-00530254e3
Content-Type: text/plain; charset=utf-8
jakke@proxyserv:~$ swift upload Demo test_file.txt
test_file.txt
jakke@proxyserv:~$ swift stat
Account: AUTH_63f23d4d415243f5b5d97ac519756815
Containers: 1
Objects: 0
Bytes: 0
Accept-Ranges: bytes
X-Timestamp: 1392661554.23253
X-Trans-Id: tx394ca6b7546d4cd880c42-005302552b
Content-Type: text/plain; charset=utf-8
jakke@proxyserv:~$ swift stat
Account: AUTH_63f23d4d415243f5b5d97ac519756815
Containers: 1
Objects: 1
Bytes: 1048576
Accept-Ranges: bytes
X-Timestamp: 1392661554.23253
X-Trans-Id: txb914c428e10c41ca82f52-005302553d
Content-Type: text/plain; charset=utf-8
jakke@proxyserv:~$


jakke@proxyserv: ~
jakke@proxyserv:~$ ls -l
yhteensÃ¤ 3080
-rw-rw-r-- 1 jakke jakke   131 helmi 17 20:19 demo_testrc.sh
-rw-rw-r-- 1 jakke jakke   133 helmi 5 23:30 swift-openrc.sh
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_2.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_3.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file.txt
jakke@proxyserv:~$ rm test_file.txt
jakke@proxyserv:~$ ls -l
yhteensÃ¤ 2056
-rw-rw-r-- 1 jakke jakke   131 helmi 17 20:19 demo_testrc.sh
-rw-rw-r-- 1 jakke jakke   133 helmi 5 23:30 swift-openrc.sh
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_2.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_3.txt
jakke@proxyserv:~$ swift download Demo test_file.txt
test_file.txt [headers 0.299s, total 0.319s, 3.288 MB/s]
jakke@proxyserv:~$ ls -l
yhteensÃ¤ 3080
-rw-rw-r-- 1 jakke jakke   131 helmi 17 20:19 demo_testrc.sh
-rw-rw-r-- 1 jakke jakke   133 helmi 5 23:30 swift-openrc.sh
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_2.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file_3.txt
-rw-rw-r-- 1 jakke jakke 1048576 helmi 17 20:24 test_file.txt
jakke@proxyserv:~$


jaakkojl@shell:~
LinuxClient:/home/root # source demo_testrc.sh
LinuxClient:/home/root # swift upload Client client_test_file.txt
client_test_file.txt
LinuxClient:/home/root # swift upload Client client_test_file_2.txt
client_test_file_2.txt
LinuxClient:/home/root # swift upload Client client_test_file_3.txt
client_test_file_3.txt
LinuxClient:/home/root # swift stat
Account: AUTH_63f23d4d415243f5b5d97ac519756815
Containers: 2
Objects: 4
Bytes: 4194304
Accept-Ranges: bytes
X-Timestamp: 1392661554.23275
X-Trans-Id: txaed469f4a0474248b0074-0053025a09
Content-Type: text/plain; charset=utf-8
LinuxClient:/home/root # swift list Client
client_test_file.txt
client_test_file_2.txt
client_test_file_3.txt
LinuxClient:/home/root #


jaakkojl@shell:~
LinuxClient:/home/root # ls -l
total 3096
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file_2.txt
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file_3.txt
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file.txt
-rw-r--r-- 1 root root 131 Feb 17 20:40 demo_testrc.sh
-rw-r--r-- 1 root root 131 Feb 17 20:39 openrc.sh
-rw-r--r-- 1 root root 133 Feb 17 20:39 swift-openrc.sh
LinuxClient:/home/root # rm client_test_file*
LinuxClient:/home/root # ls -l
total 12
-rw-r--r-- 1 root root 131 Feb 17 20:40 demo_testrc.sh
-rw-r--r-- 1 root root 131 Feb 17 20:39 openrc.sh
-rw-r--r-- 1 root root 133 Feb 17 20:39 swift-openrc.sh
LinuxClient:/home/root # swift download Client client_test_file.txt
client_test_file.txt [headers 0.548s, total 0.615s, 1.706 MB/s]
LinuxClient:/home/root # swift download Client client_test_file_*.txt
Object 'Client/client_test_file_*.txt' not found
LinuxClient:/home/root # swift download Client client_test_file_2.txt
client_test_file_2.txt [headers 0.369s, total 0.392s, 2.675 MB/s]
LinuxClient:/home/root # swift download Client client_test_file_3.txt
client_test_file_3.txt [headers 0.376s, total 0.396s, 2.650 MB/s]
LinuxClient:/home/root # ls -l
total 3096
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file_2.txt
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file_3.txt
-rw-r--r-- 1 root root 1048576 Feb 17 20:45 client_test_file.txt
-rw-r--r-- 1 root root 131 Feb 17 20:40 demo_testrc.sh
-rw-r--r-- 1 root root 131 Feb 17 20:39 openrc.sh
-rw-r--r-- 1 root root 133 Feb 17 20:39 swift-openrc.sh
LinuxClient:/home/root #


Open Connection
Openstack Swift (Keystone HTTP)
Server:
172.16.0.3
Port:
5000
URL:
http://demo:test@172.16.0.3:5000/
Tenant ID:Access Key:
demo:test
Secret Key:
Anonymous Login
Save Password
Connect
Cancel
More Options
Path:
Connect Mode:
Default
Encoding:
Default
Use Public Key Authentication
No private key selected


Swift_Service_Tenant
Unregistered
File Edit View Go Bookmark Window Help
Open Connection
Quick Connect
Action
Get Info
Refresh
Edit
Upload
Transfers
Disconnect
Search...
Filename
Size
Modified
Demo 8.2 MiB Unknown
npp.6.5.2.Installer.exe 7.2 MiB 17.2.2014 17:03:44
test_file.txt 1.0 MiB 5.2.2014 23:37:29
Siirto 14.0 MiB Unknown
default_change.pdf 250.1 KiB 17.2.2014 17:16:42
Demystifying_Google_Hacks.doc 80.0 KiB 17.2.2014 17:16:42
Mickey_Mouse_1.png 56.5 KiB 17.2.2014 17:16:43
Test.zip 13.6 MiB 17.2.2014 17:16:43
Testi 5.6 MiB Unknown
FileType.pdf 125.3 KiB 17.2.2014 17:17:12
GTG_1704TSI_FT4TP_60Hz.doc 422.0 KiB 17.2.2014 17:17:12
limiting.pdf 183.8 KiB 17.2.2014 17:17:12
Mickey_Mouse_2.png 4.9 MiB 17.2.2014 17:17:12
13 Files