Haaga-Helia
ammattikorkeakoulu Oy

OpenStack ja pilvipalvelut

Jaakko Laitinen

Opinnäytetyö
Tietojenkäsittelyn -
koulutusohjelma
2016

Tekijä(t)
Jaakko Laitinen

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
OpenStack ja pilvipalvelut

Sivu- ja liitesivumäärä
34 + 10

Opinnäytetyö ”OpenStack ja pilvipalvelut” käynnistettiin syksyllä 2016 tekijän tarpeesta valmistua. Aihe valikoitui edellämainitun mukaiseksi, koska pilvipalvelut ovat erittäin ajankohtainen asia, joiden käyttö kasvaa kiihtyvää tahtia globaalissa mittakaavassa. Työssä käsitellään yksityiskohtaisemmin OpenStackia, joka on yksi tämän hetken käytetyimmistä pilvipalvelualustoista.

Projektin tavoitteena oli perehtyä erilaisiin pilvipalvelumalleihin ja niiden toimintaan, Openstackiin ja suorittaa OpenStackin asennus käytännössä. Asennuksen suorittaminen käytännössä mahdollisti tarkemman perehtymisen pilvipalvelun toimintaan ja –käyttöönottoon. Kaikki asennuksessa tehdyt toimenpiteet dokumentoitiin tarkasti, jotta asiasta kiinnostuneiden olisi helpompi tutustua OpenStackin käyttöönottoprosessiin.

Opinnäytetyön johdannossa käsitellään taustoja ja toteutusta. Teoriaosuudessa käydään läpi erilaisia pilvipalvelutyyppejä ja pilvipalveluiden mahdollisia riskejä. Pilvipalvelualusta OpenStackiin perehdytään käymällä läpi sen pääkomponentit, jotka otetaan käyttöön asentamalla. Nämä komponentit ovat Nova, Keystone, Glance, Cinder ja Neutron.

Teoriaosuuden jälkeen esitellään OpenStack asennusympäristö. Tämän jälkeen käydään yksityiskohtaisesti läpi kaikki asennusprosessin vaiheet ja kerrotaan tehdyistä toimenpiteistä. Asennuksen valmistuttua ympäristön toiminta varmistetaan ja sitä testataan käytännössä.

Lopuksi käsitellään projektin onnistumista ja tekijän mielipiteitä OpenStackista.

Asiasanat
Pilvipalvelut Linux OpenStack

Sisällys

KÄSITTEIDEN MÄÄRITTELY

Software as a Service	Tapa toimittaa ohjelmisto verkon yli palveluna
Platform as a Service	Tapa ulkoistaa palvelualusta ja käyttää sitä verkon yli
Infrastructure as a Service	Tapa ulkoistaa palvelimet pilvipalveluihin
API	Ohejlmointirajapinta, jonka avulla ohjelmalle voi tehdä pyyntöjä.
API-Endpoint	Osoite mistä apiin pääsee käsiksi.
NTP	Network Time Protocol.
Chrony	Ohjelma NTP:n käyttöön.
Node	Nimitys asennettavaan järjestelmään kuuluvalle tietokoneelle.
Mariadb	Asennuksessa käytetty tietokanta.
Rabbit messagequeue	Ohelma, jolla OpenStackin moduulit viestivät keskenään.
Token	Paketoitu OpenStackin autentikointitieto.
Memcached	Ohjelma jota käytetään tokeneiden säilyttämiseen.
Apache	Avoimen lähdekoodin HTTP-palvelinohjelma
mod_wsgi	Apache liitännäinen, joka mahdollistaa Pythonin käytön.
User	OpenStackin käyttäjäidentiteetti, minkä voi liittää projekteihin. Voi olla esimerkiksi palvelu tai OpenStackia käyttävä ihminen.
Role	Määrittää käyttäjän roolin OpenStack-ympäristössä.
Service	OpenStackin palvelu, kuten esimerkiksi Compute
Nova	OpenStackin Compute-palvelu, jolla muunmuassa ajetaan virtuaalikoneita.
Keystone	OpenStackin autentikointipalvelu
Glance	OpenStackin levykuvapalvelu
Cinder	OpenStackin tietovarastona käytetty palvelu
Neutron	OpenStackin verkkoja hoitava palvelu
Dashboard	OpenStackin graaffinen käyttöliittymä
Nano	Yksi Linuxin tekstieditoreista.
Root	Unix-järjestelmän pääkäyttjä

1 Johdanto

Teknologian nopean kehityksen vuoksi, tietoteknisten palveluiden ylläpitämiseen tarvittavan laitteiston määrä kasvaa kiihtyvällä tahdilla. Kehityksen myötä syntyneiden pilvipalveluiden avulla on kuitenkin mahdollista ulkoistaa tämä ylläpitäminen toisten tahojen vastuulle. Pilvipalvelut mahdollistavat tiedon tallentamisen, sovellusten tarjoamisen ja jopa kokonaisten palvelinympäristöjen käyttämisen internetin yli. Niin yksityiset ihmiset, kuin suuret yritykset voivat hyöyä näistä pilvipalveluiden tarjoamista lukuisista mahdollisuuksista. Yksi tämän hetken merkittävimmistä alustoista pilvipalveluiden tarjoamiseen on avoimen lähdekoodin OpenStack, johon tässä opinnäytetyössä perehdytään.

Opinnäytetyö ”Pilvipalvelut ja OpenStack” käynnistettiin syksyllä 2016 tekijän tarpeesta valmistua. Aihevalinta muotoutui lopulta edellämainittuun, sillä tekjiä oli kiinnostunut erittäin ajankohtaisesta pilvipalvelualusta OpenStackista. Motivaattoreina toimivat myöskin mahdollisuudet ammatillisen osaamisen lisäämiseen käytännön asennustoimenpiteitä tekemällä.

Opinnäytetyön alussa käsitellän pilvipalveluja ja niiden erillaisia tyyppejä. Lisäksi perehdytään niiden tietoturvaan ja mahdollisiin riskeihin. Tämän jälkeen syvennytään tarkemmin yksittäiseen pilvipalvelualusta OpenStackiin. Projektin päätavoitteena on tutkia OpenStackin käyttöönottoa ja toimintaa suorittamalla pilvipalveluympäristön käytännön asennus. Tavoiteltu lopputulos on toimiva kolmen tietokoneen OpenStack-ympäristö, joka sisältää OpenStackin päämoduuleita. Tämän ympäristön tulee olla kykenevä virtuaalikoneiden ajamiseen, jotta IaaS-pilvipalvelumallin simuloiminen olisi mahdollista. Lopuksi pohditaan asennusprosessin onnistumista ja analysoidaan OpenStackin toimintaa.

2 Pilvipalvelut ja niiden tyypit

Pilvipalveluilla tarkoitetaan erilaisia internetin yli käytettäviä palveluita. Nämä palvelut voivat olla joko sovelluksia, tai näiden sovelluksien hostaamiseen käytettäviä palvelinkeskuksissa sijaitsevia palvelimia ja näiden varusohjelmistoja (Armburst, Fox, Griffith, Joseph, Katz, Konwinski, Lee, Patterson, Rabikin, Stoica, Zaharia 2010, 50). Tämä mahdollistaa esimerkiksi sovelluksien ajamisen internetyhteyden yli selaimen avulla palveluntarjoajan serveriltä. Sovelluksien hallinasta ja toimivuudesta vastaa palveluntarjoaja, joten käyttäjän ei tarvitse käyttää resursseja sovelluksien ylläpitoon ja päivittämiseen, mikä voi olla työlästä ja vaativaa. Pilvipalveluiden suurin etu onkin ehdottomasti niiden helppo saavutettavuus ja -käytettävyys. Pilvessä sijaitseviin resursseihin pääsee käsiksi lähes mistä tahansa missä on käytettävissä internetyhteys. Myös tiedon tallentaminen pilveen voi olla järkevä ratkaisu niin yksityiselle henkilölle, kuin isolle yritykselle. Tallennettavan tiedon määrä kasvaa koko ajan, joten fyysistä tallennusmediaa joutuu hankkimaan jatkuvasti lisää. Pilvipalvelua käyttämällä palveluntarjoaja hoitaa tämän ongelman puolestasi ja itse päätät vain kuinka paljon tilaa tarvitset.

Pilvipalvelun käyttäjän näkökulmasta pilvipalvelun tekniikka jää yleensä mysteeriksi, sillä pilven tekninen toteutus ei ole näkyvissä. Käyttäjä ei siis välttämättä edes tiedä missä palvelua ylläpitävät palvelimet sijaitsevat. Lisäksi palvelimien laskentateho on virtualisoitu, joten alla olevaan rautaan ei pääse käsiksi mitenkään.

2.1 Pilvipalveluiden tyypit ja luokat

Pilvipalvelut voidaan jaoitella karkeasti julkiseen- ja yksityiseen pilveen. Yksityinen pilvi toimii yksityisessä verkossa eli vaikkapa yrityksen lähiverkossa. Tämä tarkoittaa sitä, että sen käyttöön ei välttämättä tarvita internet-yhteyttä. On myös mahdollista, että yksityistä pilveä hostataan palveluntarjoajan konesaleista, mutta tällöin pilven täytyy olla varattu vain yhden organisaation käyttöön (Techtarget 2015). Yksityistä pilveä käyttävällä organisaatiolla on julkisen pilven käyttöön verrattuna paremmat hallintamahdollisuudet, sekä merkittävästi parempi tietoturva. Yhdistäminen pilveen tapahtuu tässä tapauksessa yleensä vpn-yhteyden avulla.

Julkisella pilvellä taas tarkoitetaan palveluntarjoajan ylläpitämää ympäristöä, jossa pilvipalvelut tarjotaan virtualisoidussa ympäristössä. Näitä julkisen pilven tarjoamia palveluita pääsee käyttämään kuka tahansa internetin yli (Interroute 2016). Yksityisen -ja julkisen pilven välimaastosta löytyy vielä muitakin vaihtoehtoja, kuten hybridipilvi ja yhteisöllinen pilvi.

Hybridipilvi on pilvi, joka yhdistelee yksityisen- ja julkisen pilven ominaisuuksia. Nämä julkiset- ja yksityiset ominaisuudet yhdistetään, mikä mahdollistaa joustavamman pilvipalveluiden käytön ja antaa enemmän mahdollisuuksia tiedon sijoittamiseen useisiin paikkoihin. Yhteisöllinen pilvi on periaatteessa samanlainen, kuin yksityinen pilvi, mutta se on jaettu useamman oraganisaation käyttöön. Pilvipalvelut voidaan jaoitella näistä pääluokista vielä eteenpäin alaluokkiin, jotka ovat Saas(Software as a Service), PaaS(Platform as a Service) ja Iaas (Infrastracture as a Service). Jatkuvasti kehittyvien pilvipalveluteknologioiden vuoksi on myös kehitetty termi XaaS(Anything as a Service) kuvaamaan kaikkia luokkia (Apperenda 2016).

2.2 Software as a Service

SaaS on tapa jolla sovelluksien toimittaminen internetin yli on mahdollista. Gartner määrittelee tämän sovelluksena, joka omistetaan, toimitetaan ja jota ylläpidetään etäältä palveluntarjoajan toimesta (Gartner 2016). Näitä sovelluksia voidaan yleensä käyttää internet-selaimen avulla palveluntarjoajan serveriltä. Joskus myös palveluntarjoaja tarjoaa kevyen sovelluksen, jolla muita sovelluksia voidaan käyttää etänä servereiltä. Käyttämällä SaaS-sovelluksia on mahdollista ulkoistaa ohjelmistojen asennus, päivitys ja ylläpito ulkopuoliselle taholle. Myöskään sovellusten käyttämien ei vaadi laitteistolta merkittäviä, tehoja sillä käytännössä ohjelmia käytetään etäyhteyden avulla. Yleensä näistä palveluista maksetaan kiinteää tilausmaksua tai laskutus tulee käyttömäärien perusteella.

2.3 Platform as a Service

PaaS mahdollistaa sovelluksien kehittämiseen ja pyörittämiseen tarvittavan palvelualustan ulkoistamisen. PaaS on siis pilvipalvelupohjainen tietojenkäsittelyympäristö, joka tukee sovellusten nopeaa kehittämistä, –ajamista, sekä –ylläpitämistä (Joshi 2014). Ulkoistamalla palvelualustan, käyttäjä pääsee suoraan keskittymään ohjelmistokehitykseen ilman, että joutuisi miettimään monimutkaista ohlemistoinfrastruktuuria.

2.4 Infrastructure as a Service

IaaS-malli mahdollistaa IT-infrastruktuurin ulkoistamisen palveluntarjoajalle, joka tuottaa tämän tarvittavan infrastruktuurin virtuaalisena. Käytännössä tämä siis tarkoittaa muunmuassa virtuaalisia palvelimia ja tallennustilaa (Gartner 2016). IaaS-malli tarjoaa merkittävää etua yrityksille, sillä palvelinsalien ylläpitämiseen ja päivittämiseen tarvitaan paljon resursseja teknologian kehittyessä, sekä datan määrän kasvaessa jatkuvasti. Yksi esimerkki IaaS-alustasta on OpenStack, jota käsitellään pian tarkemmin.

2.5 Pilvipalveluiden riskit

Vaikka pilvipalvelut tarjoavat paljon mahdollisuuksia, mukana tulee myös useita mahdollisia riskitekijöitä. Käytännössä nämä riskitekijät ovat identtisiä uhkien kanssa, jotka ovat kohdistuneet perinteisesti yksityisen ihmisen tietokoneeseen tai yrityksen it-infrastruktuuriin. Yhdessä pilvipalvelussa saattaa vaan olla tallennettuna useiden yritysten ja lukuisten yksityisten ihmisten tiedot, joten ne ovat erittäin houkuttelevia kohteita verkkorikollisille (Rashid 2016). Yleensä tietovarkauden tapahtuessa sen vakavuuden ja mahdolliset vahingot määrittävät se, miten tärkeää tietoa varastettu data sisältää. Varastetut lomakuvat saattavat aiheuttaa yksityisyyden menetyksen tunnetta, mutta varsinaiset vahingot ovat hyvin pieniä. Jos taas esimerkiksi varastetut tiedot ovat yrityssalaisuuksia, menetykset yritykselle voivat olla merkittävät.

Myös tunnusten kaappaaminen ja datan fyysinen sijainti ovat merkittäviä riskejä (Järvinen 2016, 14). Pilvipalveluihin tallennetut resurssit ovat usein pelkästään käyttäjätunnuksen takana, joten esimerkiksi heikko salasana vaarantaa kaiken pilvipalvelussa olevan tiedon. Tähän liittyvät ongelmat on kuitenkin helppo korjata ottamalla käyttöön monitasoinen autentikaatoijärjestelmä, vaatimalla esimerkiksi normaalin salasanan lisäksi kertakäyttöinen tekstiviestillä saatava koodi. Asiakas ei välttämättä tiedä missä pilvipalveluntarjoajan palvelimet sijaitsevat, joten myös tiedon sijainti on mysteeri. Tämän takia myös tietoturva on siis ulkoistettu palveluntarjoajan vastuulle, mikä on mahdollinen riskitekijä.

Datan säilyttämiseen liittyy myös laillisia ongelmia. Pilvipalveluiden käyttäjillä voi olla lainsäädännöllisiä velvoitteita siitä miten ja missä tietoja säilytetään. Esimerkiksi Euroopan Unionin sisällä toimiva firma ei välttämättä voi säilyttää henkilötietoja EU-alueen ulkopuolella (Salo 2013, 107). Salon mukaan pilvipalveluihin liittyviä ongelmakohtia liioitellaan kuitenkin merkittävästi. Yleensä pilvipalveluissa tapahtuvat tietovuodot päätyvät uutisotsikoihin nopeasti, joka edistää negatiivista suhtautumista pilvipalveluja kohtaan. Pilvipalveluita käyttävien yritysten kannattaa kuitenkin arvioida, kuinka kriittistä materiaalia pilveen laitetaan, koska riskejä kuitenkin on. Esimerkiksi Microsoftin Azure-pilvipalvelu tallentaa kaiken tiedon kolmenkertaisesti, joten pysyvästi tietoja menettää harvoin. Näitäkin tapauksia kuitenkin on missä isotkin pilvipalveluntarjoajat ovat aiheutuneet ongelmiin ja hävittäneet pysyvästi dataa (Blodget 2011).

Yksi iso mahdollinen ongelma on myös saavutettavuus pilveen. Kun yhteys pilveen katkeaa, sen sisällä oleviin resursseihin ei päästä enää käsiksi.

Jopa pieni katkos voi tulla yritykselle erittäin kalliksi, jos se ei esimerkiksi pysty käyttämään tarvitsemiansa pilvessä hostattuja sovelluksia. Tämän takia harkittaessa resurssien siirtämistä pilvipalveluun, on hyvä myöskin miettiä onko mahdollisiin katkoihin varaa.

3 Openstack ja mikä se on

OpenStack ilmainen avoimen lähdekoodin virtualisointijärjestelmä ja alusta pilvipalveluille. Se on toteutettu niin sanotulla modulaarisella rakenteella, eli kaikki erilaiset toiminnot on keskitetty oman moduulinsa sisälle. Tämän ansiosta otettaessa OpenStackia käyttöön, voidaan valita tarvittavat moduulit sen mukaan mihin käyttötarkotikseen järjestelmä tulee. Näitä moduuleja on yhteensä kymmeniä erilaisia. OpenStackin käyttö ja asentaminen tapahtuu lähinnä Linuxin komentoriviä käyttämällä, mutta sen resurssien hallinta on myös mahdollista graaffisen käyttöliittymän avulla. OpenStackin virallisten sivujen mukaan sen yleisin käyttömuoto on IaaS, eli sitä käytetään lähinnä virtuaalipalvelimien ylläpitämiseen (OpenStack Foundation 2016).

Jokaisella OpenStackin moduulilla on oma tietokantansa, minne tallenetaan moduuliin liittyvät tiedot ja asetukset. Se miten tietokanta täytetään, riippuu konfiguroinnista joka hoidetaan konfiguraatiotiedostojen avulla. Jokainen OpenStackin moduuli sisältää valmiin ohjelmointirajapinnan eli API:n, jonka kautta moduulien hallitseminen ja käyttö on mahdollista.

OpenStack syntyi 2010 Rackspacen ja NASAn yhteistyön tuloksena. Ensimmäiset kolme vuotta kehityksestä vastasi Rackspace useiden yhteistyökumppanien kanssa, mutta vuonna 2013 perustettiin itsenäinen voittoa tavoittelematon järjestö OpenStack Foundation, joka otti kehityksen vastuulleen. Nykyään OpenStackin kehityksen rahoittamiseen osallistuu satoja teknologiayhtiöitä, kuten esimerksiksi Intel ja Ibm (OpenStack Foundation 2016).

OpenStack-yhteistö kehittää järjestelmäänsä noin kuuden kuukauden mittaisissa sykleissä ja yhden syklin päätyttyä julkaistaan aina uusi versio. Versio pysyy yleensä tuettuna vielä seuraavan version tuotannossa olemisen ajan. Tämän jälkeen sen käyttöön ei saa enää tukea yhteistöltä. Ensimmäiset muihin pilvipalvelualustoihin verrattuna kilpailukykyiset versiot olivat vuonna 2013 ilmestyneet Grizzly ja Havana. Tällä hetkellä tuotannossa oleva versio on Mitaka ja kehitysasteella ovat vielä Newton ja Ocata. Alla olevassa tauukossa 1. näkyvät kaikki OpenStackin versiot.

Taulukko 1: OpensStackin versiohistoria

Julkaisuvuosi	Nimi
2010	Austin
2011	Bexar, Cactus, Diablo
2012	Essex, Folsom
2013	Grizzly, Havana
2014	Icehouse, Juno
2015	Kilo
2016	Liberty, Mitaka

3.1 Levykuvapalvelu Glance

OpenStackin levykuvapalvelu Glancen pääasiallinen tarkoitus on levykuvien hallinta. OpenStackin käyttäjät voivat tämän avulla luoda ja hallita levykuvia. Näitä levykuvia voidaan skäyttää virtuaalikoneiden ajamiseen. Glancea on myös mahdollista käyttää varmuuskopioiden ottamiseen omista virtuaalikoneista. Glanceen tallenetut levykuvat on mahdollista rajoittaa tiettyjen projektien tai käyttäjien käyttöön.

3.2 Identiteettipalvelu Keystone

Keystonea käytetään lähinnä autentikoinnissa OpenStack-järjestelmässä. Se tukee useita erilaisia autentikointimuotoja, kuten tavallista käyttäjäntunnusta ja salasanaa, tokenpohjaisia järjestelmiä, sekä AWS(Amazon web Services)- kirjautumisia (OpenStack Foundation 2016). Kun jokin OpenStackin palvelu tai käyttäjä koittaa autentikoitua järjestelmään, Keystone varmistaa, että tällä on oikeudet autentikointipyynnön tekemiseen. Jos oikeudet löytyvät, Keystone antaa valtuudet autentikaation suorittamiseen.

3.3 Verkkopalvelu Neutron

Verkkopalvelu Neutron mahdollistaa verkkoyhteyksien luomisen ja hallinnan OpenStack-järjestelmässä. Se tukee sekä staattisten ip-osoitteiden ja DHCP:n käyttöä. Neutronin käyttäjät voivat luoda omia OpenStackin sisäisiä verkkoja, yhdistää eri virtuaalikoneita näihin verkkoihin, sekä hallita näiden verkkoliikennettä.
Neutron koostuu useasta eri komponentista, kuten Neutron-palvelimesta, Linux-bridge agenteista ja messagequeue-palvelusta.

Palvelimen kautta päästään käsiksi Neutronin ohjelmointirajapintaan, agenttien avulla hoidetaan verkkojen sekä aliverkkojen toimintaa ja messagequeue välittää tietoa muiden komponenttien välillä.

3.4 Computepalvelu Nova

Nova on OpenStackin compute-palvelu, jota käytetään pääasiassa virtuaalikoneiden ajamiseen ja ylläpitämiseen. Se on hyvin virheenkestävä ja skaalautuu erinomaisesti isoihinkin toteutuksiin erilaisilla alustoilla. OpenStackin käyttämän viestintäarkkitehtuurin ansiosta(messagequeue), Novan komponentteja pystytään tarvittaessa ajamaan useille eri servereillä. Se tukee myöskin useita eri virtualisointiteknologioita, kuten QEMU ja kvm.

3.5 Graafinen käyttöliittymä Dashboard

OpenStackin graaffinen käyttöliittymä Dashboard tarjoaa järjestelmän käyttäjille graafisen ympäristön pilvipalvelun resurssien hallintaan. Sen rakenne mahdollistaa ulkopuolisten palvelujen, kuten järjestelmän monitorointityökalujen käyttämisen. Dasboard on myös mahdollista brändätä palveluntarjoajille ja muille kaupallisille toimijoille, jotka haluavat käyttää sitä. Komentoriviltä hallitsemisen ohella, Dashboard on toinen tapa jolla OpenStack-resursseja käytetään.

3.6 Openstack vs muut alustat

OpenStackin suosio pilvipalvelualustana kasvaa jatkuvasti. STT-infossa julkaistussa tiedotteessa kerrotaan SUSE:n toimeksiannosta tehdystä kansainvälisestä kyselystä, jonka mukaan 81% suurista yrityksistä käyttää tai aikoo käyttää OpenStack-pohjaisia ratkaisuja omissa yksityisissä pilvissään(Suse Suomi, 2016). OpenStackin suosiota varjostavat kuitenkin käyttöönottoon liittyvät haasteet, jotka johtuvat OpenStack-osaajien puutteesta, sekä käyttöönoton haastavuudesta. Lisäksi kyselyn mukaan noin joka toinen yrityksistä oli epäonnistunut OpenStackin käyttöönotossa, mikä on hälyttävän suuri lukema.

Toinen merkittävä IaaS palvelualusta OpenStackin lisäksi on Citrixin OpenCloud. OpenCloudin etuna on helpompi käyttöönotto, mutta toisaalta tämän myötä se ei skaalaudu yhtä hyvin erilaisille alustoille. OpenStackin kehityksessä on mukana paljon isoja teknologiafirmoja, joka auttaa sen yhteensopivuudessa useiden alustojen kanssa (Lando 2014).

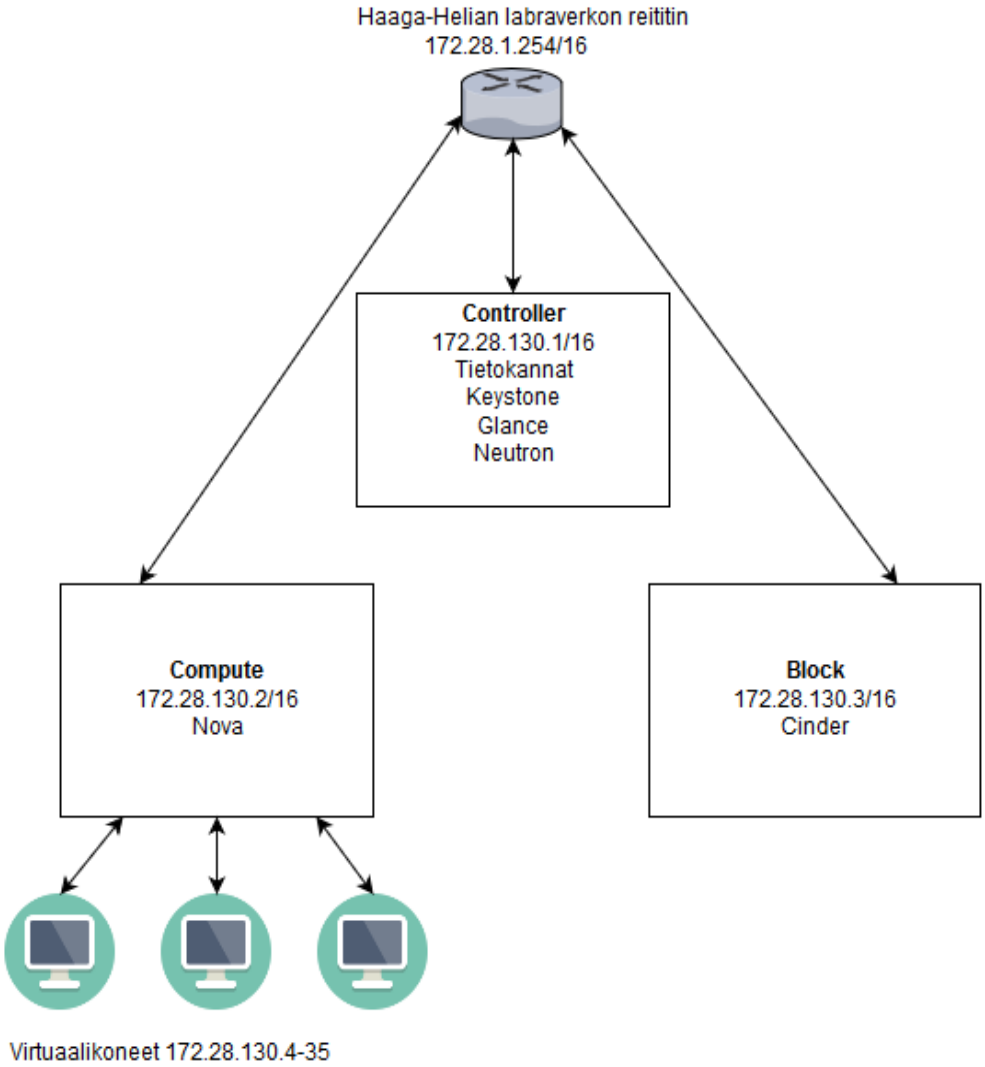
3.7 OpenStackin testiasennus

Tässä osassa suoritetaan OpenStack Mitakan testiasennus OpenStack-säätiön tarjoamia asennusohjeita ja konfiguraatiotiedostoja mallina käyttäen (OpenStack Foundation 2016). Tavoitteena on saada aikaiseksi toimiva OpenStack-testiympäristö, jossa pystyttäisiin ajamaan virtuaalikoneita. Testiympäristössä käytetään kolmea tietokonetta eli nodea, joista jokaiseen asennetaan tarvittavat Openstack-komponentit. Asennus voitaisiin myös tehdä yhteen tietokoneeseen ns. All in one-asennuksena, mutta käyttämällä useaa nodea pystytään imitoimaan paremmin oikeaa OpenStack-ympäristöä, mitä esimerkiksi yritys voisi käyttää. Myös OpenStackin virtuaalikoneille asentaminen olisi mahdollista, mutta useampaa fyysistä tietokonetta käyttämällä saadaan myöskin enemmän resursseja käyttöön.

Tässä asennusraportissa käytettävät salasanat merkataan kursivoidusti esim. *salasana.* Kaikki toimenpiteet suoritetaan root-oikeuksilla. Pitkät konfiguraatiotiedostot ovat liitteinä, joihin viitataan tekstissä rivinumeroilla. Nämä rivinumerot eivät kuulu oikeisiin konfiguraatiotiedostoihin, vaan ne on lisätty tähän työhön selkeyttämisen vuoksi. Asennuksessa käytettävät konfiguraatiotiedostot on kerrottu niiden sijainnin perusteella. Esimerkiksi Linux bridge agentin konfiguraatiotiedosto merkataan tavalla: /etc/neutron/plugins/m12/linuxbridge_agent.ini, jossa /etc//etc/neutron/plugins/m12/ tarkoittaa tiedoston sijaintia ja linuxbridge_agent.ini nimeä. Asennuksessa esitellyt konfiguraatiotiedostot ovat OpenStackin pakettien mukana tulleita oletuskonfiguraatiotiedostoja. Niiden pituuden takia koko konfiguraatiotiedostoja ei esitetä. Sen sijaan esitetään niihin tehdyt muutokset ja lisäykset.

3.7.1 Ympäristö

Ympäristönä käytetään kolmea tietokonetta eli nodea, joissa on Intel i5-prosessorit, 8-gigatavua keskusmuistia ja 500-gigatavun kovalevyt. Tietokoneisiin on asennettu Ubuntu 16.04 LTS-käyttöjärjestelmät ja asennettava Openstack on uusin vakaa versio nimeltään Mitaka. Nodeihin on laitettu kiinteät ip-osoitteet 172.28.130.1-3 ja reititykseen käytetään Haaga-Helian labraverkon reititintä 172.28.1.254. Nodejen nimiksi on laitettu Controller, Compute ja Block. Nodejen /etc/hosts-tiedostoon on laitettu muiden nodejen nimet ja ip-osoitteet nimenselvitystä varten. Kuvassa 1. näkyvät kaikki nodet ja moduulit, jotka niihin tullaan asentamaan.



Kuva 1. Asennusympäristö

3.7.2 Alkutoimenpiteet

OpenStack käyttää ntp:tä (network time protocol) palveluiden synkronoimiseen eri nodejen välillä. Sen käyttö ei ole pakollista, mutta parantaa toimivuutta merkittävästi, joten asennetaan kaikkiin nodeihin ntp-palvelu chrony. Chronyn avulla pystymme synkronoimaan nodejen kellot ntp-serverin kanssa, jolloin saamme kaikkiin nodeihin täsmällisen ajan (Chrony 2016). Asennus suoritetaan kaikkiin nodeihin komennolla: apt-get install chrony.

Tämän jälkeen muokataan Chronyn konfiguraatiotiedostoa /etc/chrony/chrony.conf. Oikeassa OpenStackin tuotantoversiossa Controller-node synkronoisi ajan ntp-serveriltä ja muut nodet synkronoisivat sen jälkeen Controller-nodesta, mutta tässä testiasennuksessa kaikki nodet synkronoivat ajan suoraan ntp-serveriltä. Lisätään tähän konfiguraatiotiedostoon ntp-serveri lisäämällä parametri ”server ntp1.funet.fi iburst”.

Parametri iburstia käytettäessä ntp-palvelu mittaa käynnistyessä ajan nopeasti neljä kertaa säännöllisten mittausten sijasta. Tällä saavutetaan nopeampi synkronointi ntp-serverin kanssa

Seuraavaksi otetaan OpenStackin pakettivarastot käyttöön kaikille nodeille, jotta saadaan Openstackin paketit ladattua. Tehdään tämä ajamalla komennot: “apt-get install software-properties-common” ja “add-apt-repository cloud-archive:mitaka”. Lisäksi asennetaan ja päivitetään nämä tarvittavat paketit komennolla ”apt-get update && apt-get dist-upgrade”. Tämän jälkeen asennetaan vielä OpenStack-klientti komennolla: ”apt-get install python-openstackclient”

Suurin osa OpenStackin palveluista käyttää sql-tietokantaa tiedon tallentamiseen, joten asennetaan sellainen seuraavaksi. Tässä testiasennuksessa sql-tietokantana käytetään Mariadb:tä sen helppouden takia, mutta esimerkiksi MySql:n tai Postgres-sql:än käyttö on tarvittaessa mahdollista (OpenStack Foundation 2016). Tietokanta laitetaan tässä asennuksessa Controller-nodelle, mutta periaatteessa se voisi sijaita millä tahansa nodeista. Asennetaan samalla PyMySql-klinetti tietokantojen käyttämistä varten. Nämä saadaan asennettua komennolla: ”apt-get install mariadb-server python-pymysql”.

Määritetään seuraavalla sivulla Mariadb:n konfiguraatiotiedostoon /etc/mysql/my.cnf kohdan [mysqld] alle bind-osoitteeksi Controller-noden ip-osoite parametrilla ”bind-address 172.28.130.1”, jotta muutkin nodet pääsevät käyttämään tietokantaa. Mariadb:n bind-osoite on oletuksena localhost, joka sopii hyvin yhden noden ratkaisuun. Lisäksi määritetään kirjaimistoksi utf-8 lisäämällä ”collation-server = utf8_general_c” ja ”character-set-server = utf8”.
Viimeisenä laitetaan mariadb käyttämään innodb:tä, joka on Mariadb:n oletustietokantamottoria yksinkertaisempi versio. Tämä tehdään parametrilla ”default-storage-engine = innodb”. Käynnistetään tämän jälkeen serveri uudestaan komennolla ”service mysql restart” ja ajetaan mysql:n asennuksen varmistuskripti ”mysql_secure_installation” (Liite 1), jossa määritellään asennukseen liittyviä tietoja, kuten root-käyttäjän salasana.

1. my.cnf
2. [mysqld]
3. bind-address 172.28.130.1
4. ”collation-server = utf8_general_c
5. character-set-server = utf8
6. default-storage-engine = innodb

Seuraavaksi asennetaan Controller-nodelle niin sanottu messagequeue, joka käsittelee käyttäjän tekemiä pyyntöjä ja ohjaa ne sille osalle OpenStackia jolle ne kuuluvat. Asennetaan se komennolla ”apt-get install rabbitmq-server”. Lisätään tänne käyttäjä nimeltä openstack ja annetaan sille salasana komennolla ”rabbitmqctl add_user openstack *salasana”* Lisäksi määritellään tälle luodulle käyttäjälle konfiguraatio-, luku- ja kirjoitusoikeudet messagequeue-palveluun parametrilla: rabbitmqctl set_permissions openstack ".*" ".*" ".*"

Viimeiseksi asennetaan Controller-nodelle Memcached, jossa säilytetään OpenStackin identiteettipalvelua käyttävien muiden palveluiden kirjautumistokeneita. Asennus tapahtuu komennolla ”apt-get install memcached python-memcache”. Lisäksi editoidaan sen konfiguraatiotiedostoa /etc/memcached.conf. Tiedostossa määritellän Controller-noden ip-osoite, jotta muut nodet saavat siihen yhteyden. Tehdään tämä lisäämällä parametri ”-l 172.28.130.3 ”

3.7.3 Keystone-asennus

Ensimmäisenä moduulina asennetaan identiteettipavelu Keystone. Keystonen hoitaa autentikoinnin OpenStackissa, kuten esimerkiksi tarjoaa moduuleille yhteydet tietokantoihin. Lisäksi se sisältää muunmuassa tilit kaikille käyttäjille, OpenStackin projektit ja palvelut. Luodaan ensin identiteettipalvelulle tietokanta, jotta tiedot voidaan tallentaa jonnekkin. Kirjaudutaan sisään sql-clientillämme root-käyttäjänä ”mysql -u root –p” ja tehdään tietokanta keystone sql-komennolla ”CREATE database keystone;” . Lisäksi määritellään salasana, jolla kirjautumalla saadaan kaikki oikeudet tähän tietokantaan sekä localhostista että muilta koneilta seuraavasti:

GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY 'salasana'; sekä
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY 'salasana';

Seuraavaksi asennetaan Keystone ja Apache-serveri mod_wsgi-moduulin kanssa, joita käytetään muiden nodejen ja käyttäjien identiteettipalvelulle tekemien pyyntöjen hallitsemiseen. Asennetaan nämä komennolla ”apt-get install keystone apache2 libapache2-mod-wsgi”. Pyynnöt tulevat porttien 5000 ja 35357 kautta, joita keystone kuuntelee oletuksena, joka on turhaa sillä pyynnöt käsitellään tässä asennuksessa Apachen avulla. Tämän takia Keystone-palvelu disabloidaan ensin manuaalisesti komennolla ”echo "manual" > /etc/init/keystone.override”.

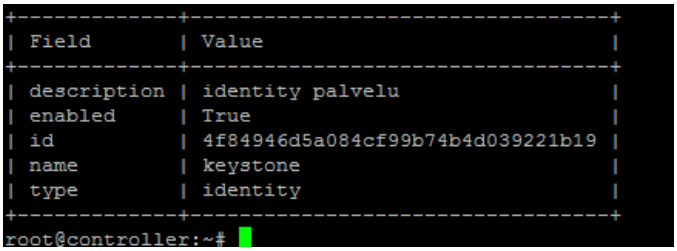
Lisätään myös Apachen konfiguraatiotiedostoon /etc/apache2/apache2.conf serverin nimi-parametri nimenselvitystä varten ”ServerName controller”.
Lisäksi konfiguroidaan mod_wsgi-moduulin (Liite 2) kaksi virtuaalista hostia, joista toinen toimii portissa 5000 adminien- ja toinen portissa 35357 normaalien käyttäjien pyyntöjen vastaanottamiseen. Ensimmäimen hosti konfiguroidaan liitteen riveillä 3-14 ja toinen 15-28. Tämän jälkeen nämä virtuaaliset hostit voidaan laittaa päälle komennolla:
”ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled”.

Muokataan Keystonen konfiguraatiotiedostoa /etc/keystone/keystone.conf ja lisätään sinne kohdan [DEFAULT] alle konfiguraatiovaiheessa kirjautumiseen käytettävä admintokeni parametrilla ”admin_token = 51b6256b7c”. Admintokeni voi olla satunnainen merkkijono, mutta tietoturvan takia sen on hyvä olla suhteellisen pitkä. Määritellään yhteys myös aikaisemmin luotuun tietokantaan kohdan [database] alle, jotta keystone pääsee siihen käsiksi. Tämä tehdään lisäämällä ”parametri connection = mysql+pymysql://keystone:salasana@controller/keystone”. Lisäksi määritetään keystone käyttämään fernettiä, joka on turvallinen keino API-tokenien lähettämiseen salattuna. (Dolph 2016). Tämän tehdään lisäämällä kohdan [token] alle parametri ”provider = fernet”.

Tämän jälkeen voidaan alustaa tietokanta ja fernet-avaimet, sekä täyttää tietokanta keystonen apin-tiedoilla. Tämä tehdään komennoilla:
”keystone-manage fernet_setup --keystone-user keystone --keystone-group keystone”, sekä
”su -s /bin/sh -c "keystone-manage db_sync" keystone”

Nyt Keystone on valmis käytettäväksi, joten seuraavaksi luodaan käyttäjät ja palvelut OpenStackia varten. Ensimmäisenä ladataan keystonen konfiguraatioon laittettu admintokeni komennolla ”export OS_TOKEN=51b6256b7c”, api-versio komennolla ”export OS_IDENTITY_API_VERSION=3” ja Apache-serverin portti komennolla ”export OS_URL=http://controller:35357/v3”, jotta voidaan kirjautua sisään.

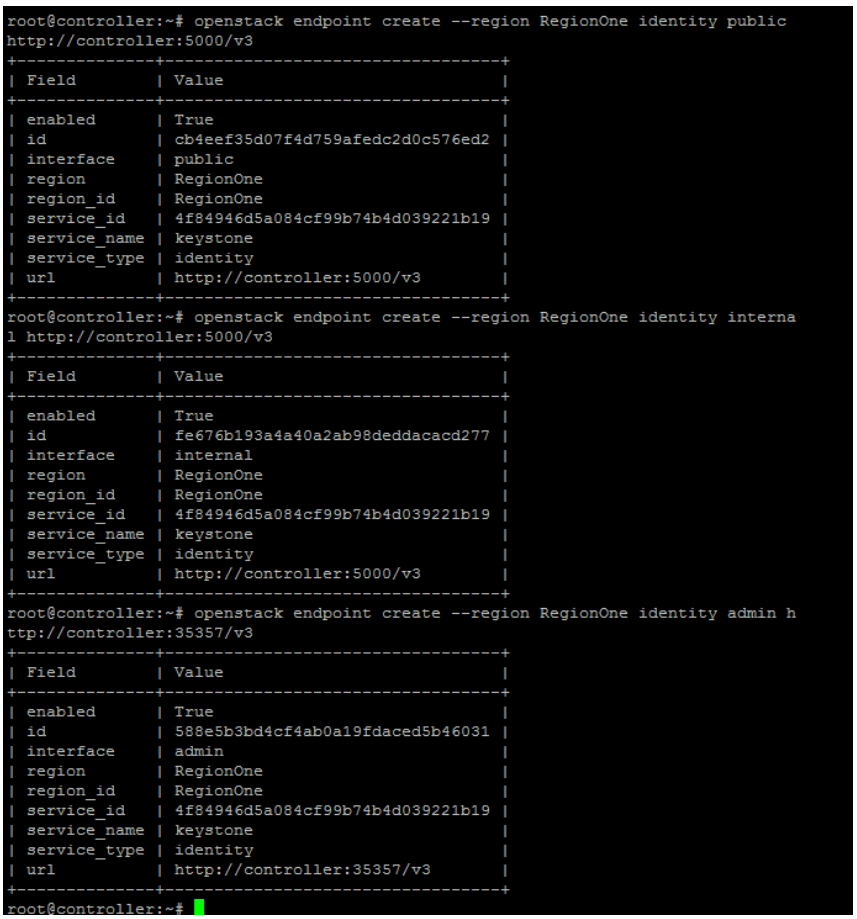
Kaikille OpenStackin moduuleille täytyy ennen käyttöönottoa luoda palvelu, joten luodaan ensimmäisenä keystonelle omansa komennolla: ”openstack service create --name keystone --description "identity palvelu" identity”. Alla olevassa kvassa 2. näkyy OpenStackin palauttamat tiedot onnistuneesti luodusta palvelusta. Onnistumisen näkee siitä, että palautetut tiedot ovat vastaavia annettujen parametrien kanssa ja lisäksi palvelu on saanut id:n(kuvassa 2. rivillä id).



Kuva 2. palvelun luonti keystonelle.

Tämän jälkeen tehdään kolme päätepistettä (Kuva 3.), joista OpenStackin apiin pääsee käsiksi. Yksi päätepisteistä on adminkäyttöön, josta OpenStackia voidaan hallita täysin oikeuksin. Lisäksi on yksi julkinen päätepiste esimerkiksi ulkoverkosta kirjautuville käyttäjille, jotta he voisivat hallita omia pilvipalveluitaan. Viimeisenä on sisäinen päätepiste jota käytetään OpenStack-klusterissa olevien nodejen yhteyksiin.

Alla olevassa kuvassa ensimmäisellä rivillä olevalla komennolla tehdään julkinen päätepiste käyttäen komentoa ”openstack endpoint create”. Parametrilla ”–region” määritellään käytetyksi regioniksi ”RegionOne”, mikä on OpenStackin oletusregion. Lisäksi parametrilla ”identity” määritellään tämä päätepiste julkiseen käyttöön. Viimeisenä komentoon tulee osoite(kuvan toinen rivi) mistä päätepisteeseen saa yhteyden. Onnistuneen päätepisteen luomisen jälkeen, OpenStack palauttaa tiedot tästä päätepisteessä(Kuvassa 3. annettujen komentojen alla). Kuvassa 3. olevissa toisessa ja kolmannessa komennossa luodaan vastaavalla tavalla sisäinen- ja admin päätepiste.



Kuva 3: Keystonen Api-endpointit

Seuraavaksi tehdään toimialue kaikille OpenStack-asennukseen tuleville projekteille käyttäjille ja ryhmille. Domain tehdään alla olevassa kuvassa 4. komennolla ”openstack domain create”. Parametria ”- -description” ei ole pakko käyttää, mutta sen avulla voi lisätä kuvauksen luotuun domainiin. Viimeisenä parametrina annetaan nimi tälle domainille, joka on tässä asennuksessa default. Komennon alla näkyy OpenStackin palauttamat tiedot onnistuneen komennon jälkeen. Kuvassa 4. näkyvässä toisessa komennossa ”openstack project create” luodaan projekti adminille. Esimerkiksi adminin luomat virtuaalikoneet tulevat kuulumaan tähän projektiin kunhan asennuksessa päästään tähän vaiheeseen. Projekti liitetään äsken luotuun defaultdomainiin parametrilla ”- -domain default”. Kuvassa 4. näkyvässä kolmannessa komennossa tehdään adminkäyttäjä default domainiin openstack user create-komennolla. Kahdessa viimeisessä komennossa luodaan ensiksi adminrooli openstack role create komennolla ja lisätään adminkäyttäjä adminprojektiin adminrooliin. Nyt meillä on siis projekti adminille, johon adminilla on täydet käyttöoikeudet. Näitä projekteja voitaisiin luoda tarvittaessa useampiakin, kuten vaikka oma projekti jollekin ulkopuoliselle käyttäjälle.

```
root@controller:~# openstack domain create --description "oletusdomain" default
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | oletusdomain                     |
| enabled     | True                             |
| id          | 5026e5d1ff4145b1b57255903ab1494d |
| name        | default                          |
+-------------+----------------------------------+
root@controller:~#
root@controller:~# openstack project create --domain default --description "admin-proggis" admin
+-------------+----------------------------------+
| Field       | Value                            |
+-------------+----------------------------------+
| description | admin-proggis                    |
| domain_id   | 5026e5d1ff4145b1b57255903ab1494d |
| enabled     | True                             |
| id          | 6014bf918b17448f90947524b6e1f15d |
| is_domain   | False                            |
| name        | admin                            |
| parent_id   | 5026e5d1ff4145b1b57255903ab1494d |
+-------------+----------------------------------+
root@controller:~# openstack user create --domain default --password-prompt admin
User Password:
Repeat User Password:
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | 5026e5d1ff4145b1b57255903ab1494d |
| enabled   | True                             |
| id        | 93e9c367771c4233b46f7dc9fb66916b |
| name      | admin                            |
+-----------+----------------------------------+
root@controller:~# openstack role create admin
+-----------+----------------------------------+
| Field     | Value                            |
+-----------+----------------------------------+
| domain_id | None                             |
| id        | d0b7e6c1e19943fabc0b055fcdffe0d8 |
| name      | admin                            |
+-----------+----------------------------------+
root@controller:~# openstack role add --project admin --user admin admin
root@controller:~#
```

Kuva 4. Domainin, projektin ja adminin luonti

Tämän jälkeen tehdään default domainiin kuuluva service-projekti johon kaikki asennettavat moduulit, kuten esimerkiksi Glance tulevat kuulumaan. Projekti tehdään komennolla “openstack project create --domain default --description "Service Project" service”.

Nyt projektit ja käyttäjät on luotu ja OpenStackin käyttö on mahdollista ilman aikaisemmin luotua väliaikaista admintokenia. Tämä on erittäin työlästä, sillä komennot ovat erittäin pitkiä lukuisten parametrien takia. Esimerkiksi pyydettässä testiksi autentikointitokeni adminkäyttäjällä komennolla: ”openstack --os-auth-url http://controller:35357/v3 --os-project-domain-name default --os-user-domain-name default --os-project-name admin --os-username admin token issue”. Varsinainen komento on ”openstack token issue”, mutta väliin tarvitaan lukuisia parametreja, kuten autentikoiniosoite, domainin nimi ja käyttäjätunnukset.

Jotta jatkossa jokaista toimenpidettä varten ei tarvitsisi kirjoittaa näin pitkää pitkää listaa parametrejä, luodaan adminkäyttäjän projektiin kirjautumista varten skripti joka lataa kirjautumistiedot automaattisesti. Määritetään se lataamaan default-projekti adminkäyttäjällä Apachen adminporttia käyttäen. Luodaan tekstitiedosto admin-openrc, jonne lisätään kirjautumisparametrit.

```
export OS_PROJECT_DOMAIN_NAME=default
export OS_USER_DOMAIN_NAME=default
export OS_PROJECT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=salasana
export OS_AUTH_URL=http://controller:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Tämän jälkeen äsken tehty autentikointitokenin pyyntö voidaan tehdä paljon yksinkertaisemmin lataamalla ensin tämä skripti komennolla ”. admin-openrc” jonka jälkeen voidaan ajaa ”openstack token issue”-komento. Keystone on tämän jälkeen asennettu, joten voidaan siirtyä muiden moduulien asennukseen.

3.7.4 Glance-asennus

Seuraavaksi asennetaa levykuvapalvelu Glance, jotta voitaisiin tallentaa levukuvia virtuaalikoneiden käynnistämistä tai niiden varmuuskopioita varten. Image-palvelu voidaan asentaa mille tahansa nodeista, mutta tässä testiasennuksessa se laitetaan Controller-nodelle.
Se tulee tallentamaan levykuvat Controller-noden lokaaliin tiedostojärjestelmään, mutta muutkin vaihtoehdot ovat mahdollisia. Jos OpenStack-ympäristö sisältää esimerkiksi objekti-varasto Swiftin, Glancen levukuvat voidaan tallentaa tänne (Openstack Foundation, 2016). Aloitetaan asennus samalla tavalla kuin Keystonen kanssa kirjautumalla sisään sql-clientilä rootkäyttäjänä komennolla ”mysql -u root -p ” ja luomalla tietokanta nimeltään glance: ”CREATE DATABASE glance”. Laitetaan jälleen oikeudet tietokantaan salasanan taakse komennoilla:

GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY '*salasana'*;
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY '*salasana*';

Tämän jälkeen ladataan aikaisemmin tehdyt admin-tunnukset ajamalla skripti ”. admin-openrc” ja luodaan Glancea varten OpenStackiin käyttäjä komennolla: ”openstack user create --domain default --password-prompt glance”, sekä laitetaan tämä käyttäjä aikaisemmin tehtyyn serviceprojektiin adminiksi komennolla: ”openstack role add --project service --user glance admin”. Tämän lisäksi tehdään uusi glance-niminen palvelu, jonka tyypiksi laitetaan image käyttötarkoituksen mukaisesti: ”openstack service create --name glance --description "levykuvat" image”. Lisäksi lisätään julkinen, sisäinen ja admin api-endpoint samalla tavalla kuin Keystonen kanssa toimittiin komennoilla:
”openstack endpoint create --region RegionOne image public http://controller:9292”, ”openstack endpoint create --region RegionOne image internal http://controller:9292” ja “openstack endpoint create --region RegionOne image admin http://controller:9292”

Asennetaan glancen paketit ja editoidaan Glancen apin konfiguraatiotiedostoa /etc/glance/glance-api.conf (Liite 3). Rivillä kaksi määritetään ensiksi yhteys tietokantaan. Riveillä 4-6 määritetään missä osoitteissa normaalikäyttäjät ja adminit voivat autentikoida. Rivit 7-12 määrittävät aikaisemmin tehdyt projektien ja domainen nimet, sekä kirjautumistunnuksen ja salasanan. Rivit 13-14 määrittävät keystonen käyttöön. Viimeiseksi riveillä 15-18 määritetään, miten ja minne glance tallentaa levykuvat.

Samat konfiguraatiot rivejä 15-18 lukukuunottamatta kirjoitetaan myös /etc/glance/glance-registry.conf tiedostoon.Tämän jälkeen Glancen asennus on valmis ja tietokanta voidaan täyttää ajamalla skripti ”su -s /bin/sh -c "glance-manage db_sync" glance”.

3.7.5 Nova-asennus

Nova hoitaa virtuaalikoneiden ylläpitämisen OpenStack-järjestelmässä, joten sitä voidaan pitää tärkeimpänä moduulina. Se kannattaa asentaa omalle compute-nodelleen, sillä virtuaalikoneiden ajaminen vie paljon resursseja. Tarvittavien virtuaalikoneiden määrän ylittäessä yhden Compute-noden kapasiteetin, OpenStack-järjestelmään pystytään liittämään uusia vastaavia compute-nodeja lähes rajattomasti.

Asennus alkaa muiden moduulien kanssa samalla tavalla, eli ensiksi luodaan Controller-nodella Novaa varten tietokannat. Muista moduuleista poiketen Nova tarvitsee kaksi tietokantaa, sillä sen api täyttyy laittaa omaan tietokantaan esimerkiksi virtuaalikoneiden indeksointia varten (Openstack Foundation 2016). Luodaan siis tietokannat nova ja nova_api SQL-lausella ”CREATE DATABASE nova;” ja ”CREATE DATABASE nova_ap”i. Tämän jälkeen laitetaan taas oikeudet molempiin tietokantoihin salasanojen taakse GRANT ALL PRIVILEGES-komennolla.

GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'localhost' IDENTIFIED BY '*salasana*';
GRANT ALL PRIVILEGES ON nova_api.* TO 'nova'@'%' IDENTIFIED BY '*salasana*';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY '*salasana*'
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY ''*salasana*'';

Seuraavaksi ladataan jälleen OpenStackin admintunnukset komennolla ”. admin-openrc” ja luodaan nova niminen käyttäjä default domainiin: ”user create-komennolla: openstack user create --domain default --password-prompt nova”. Service-projekti tulee sisältämään käyttäjän jokaiselle moduulille, joten laitetaan tämä käyttäjä adminiksi sinne role add-komennolla: ”openstack role add --project service --user nova admin”. Tämän jälkeen luodaan palvelu nimeltä nova service create-komennolla, ja laitetaan sen tyypiksi käyttötarkoituksen mukaan compute: ”openstack service create --name nova --description "compute" compute”.

Tehdään Novaakin varten julkinen, sisäinen ja admin api-endpoint, jotta muut palvelut saisivat siihen yhteyden controller-noden portissa 8774. Alla olevassa kuvassa 5. näkyvät OpenStackin palauttamat tiedot, kun api-endpointit on luotu onnistuneesti ”openstack endpoint create-komennolla”.

```
root@controller:~# openstack endpoint create --region RegionOne compute public http://controller:
8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 299a8b413f9348bbb2e85d8bb78a7c63        |
| interface    | public                                  |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | fe5e94d3c77b445590d3c962fccf55ab        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
root@controller:~#
root@controller:~# openstack endpoint create --region RegionOne compute internal http://controlle
r:8774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | 88ee53c24dd746faa3a175de6d20d1fd        |
| interface    | internal                                |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | fe5e94d3c77b445590d3c962fccf55ab        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
root@controller:~# openstack endpoint create --region RegionOne compute admin http://controller:8
774/v2.1/%\(tenant_id\)s
+--------------+-----------------------------------------+
| Field        | Value                                   |
+--------------+-----------------------------------------+
| enabled      | True                                    |
| id           | c9e3e3a85d7f4eee97cd890f2b6538a8        |
| interface    | admin                                   |
| region       | RegionOne                               |
| region_id    | RegionOne                               |
| service_id   | fe5e94d3c77b445590d3c962fccf55ab        |
| service_name | nova                                    |
| service_type | compute                                 |
| url          | http://controller:8774/v2.1/%(tenant_id)s |
+--------------+-----------------------------------------+
root@controller:~#
```

kuva 5. Novan api-endpointtien luomista.

Seuraavaksi asennetaan Controller-nodelle Novan käyttämiä palveluita. Nova-conductor toimii computenoden ja tietokannan välissä lisäten OpenStackin turvallisuutta. Jos mahdollinen hakkeri onnistuisi saamaan esimerkiksi virtuaalikoneiden kautta compute-noden haltuunsa, hän voisi tehdä tietokannassa lähes mitä tahansa. Conductorin api kuitenkin asettaa rajoituksia mahdollisille toimenpiteille, mikä lisää turvallisuutta (Cloudystuffhappens 2013). Nova-consoleauth ja Nova-novncproxy mahdollistavat yhdessä konsoliyhteyden ottamisen virtuaalikoneisiin OpenStackin graafisen käyttöliittymän kautta (Redhat 2016). Ne eivät siis ole välttämättömiä komponentteja, mutta lisäävät käyttömukavuutta merkittävästi. Nova-api mahdollistaa pyyntöjen tekemiseen Novaan muiden palveluiden toimesta ja nova-scheduler määrittää esimerkiksi, että missä virtuaalikone käynnistetään. Asennetaan nämä palvelut apt-get install-komennolla: “apt-get install nova-conductor nova-consoleauth nova-api nova-novncproxy nova-scheduler”.

Tämän jälkeen konfiguroidaan Novan Controller-nodella sijaitsevat osat tiedostoon nova.conf (liite 4). Tehdään seuraavaksi Novan konfiguraatiot: Liitteen 4 rivillä kaksi määritetään käytössä olevat api:t. Rivillä kolme otetaan käyttöön moduulien väliseen viestintäää varten asennettu rabbit messagequeue. Rivit 4-6 määrittelevät autentikoinnin keystonen kautta, ip-osoitteen missä konfiguraatiotiedosto sijaitsee ja verkkomoduuli neutronin käytön. Riveillä yhdeksän ja kymmenen määritetään yhteys nova ja nova_api tietokantaan. Rivit 13-16 sisältävät autentikointitunnukset rabbit messagequeueen. Riveillä 17-25 määritellään missä osoitteessa autentikointi tapahtuu admineille ja normaalikäyttäjille, ja että autentikointi hoidetaan käyttäjätunnuksella ja salasanalla. Rivillä 30 on määritetty yhteys levykuvapalvelu Glanceen.

Tämän jälkeen Controller-nodelle on tehty tarvittavat toimenpiteet, joten Novan tietokannat voidaan täyttää komennoilla: *su -s /bin/sh -c "nova-manage api_db sync" nova* ja *su -s /bin/sh -c "nova-manage db sync" nova* ja siirtyä varsinaisen moduulin asentamiseen Compute-nodelle.

Asennetaan Compute-nodelle Novan paketit komennolla ”apt-get install nova-compute”. Novan Compute-nodella sijaitseva osa konfiguroidaan vastaavaan nova.conf tiedostoon, joka sijaitsee samassa /etc/nova hakemistossa kuin Controller-nodella oleva nova.conf. Konfiguraatio on myöskin lähes identtinen Controllerin nova-konfiguraatioon verrattuna, mutta asennuksen aikana täytyy olla tarkkana, että konfiguraatiot suoritetaan oikeaan paikkaan. Esimerkiksi ip-osoite on tällä kertaa 172.28.130.2, koska olemme Compute-nodella. Lisäksi käytössä olevia apeja tai yhteyksiä tietokantoihin ei tarvitse määrittää. Nyt Compute-node on asennettu ja valmis virtuaalikoneiden ajamiseen.

3.7.6 Neutron-asennus

Jos Novalla ajattavien virtuaalikoneiden haluaa olevan verkossa, täytyy asentaa OpenStackin verkkomoduuli nimeltään Neutron. Suurin osa sitä asennetaan Controller-nodelle, mutta myös Compute-nodelle asennetaan niin kutsuttu Linux bridge-agentti, joka tarvitaan jokaisella datapaketteja käsittelevällä nodella (Redhat 2016). Tämä siis tarkoittaa, että jokaisella OpenStack-ympäristön Compute-nodella ja verkkoja hallitsevalla nodella täytyy olla Linux bridge-agentti. Kuten aikaisempienkin moduulien kohdalla, Neutronia varten on luotu tietokanta nimeltään Neutron ja OpenStackiin on lisätty Neutron niminen käyttäjä jolla on adminit serviceprojektiin. Näiden lisäksi on luotu Neutron-palvelu ja apiendpointit.

Neutron tarjoaa mahdollisuuden kahden eri verkkoratkaisun käyttämiseen. Tässä asennuksessa käyttetään yksinkertaisempaa provider-ratkaisua, joka mahdollistaa virtuaalikoneiden liittämisen ainoastaan ulkoiseen verkkoon. Vaihtoehto kaksi mahdollistaisi yksityisten, OpenStackin sisäisten verkkojen luomisen, jotka voisivat käyttää virtuaalisia reitittimiä muihin verkkoihin yhdistämistä varten. Tälle ei kuitenkaan ole tässä testiasennuksessa tarvetta. Konfiguroidaan Neutron konfiguraatiotiedososssa neutron.conf (Liite 5.) Neutronin konfiguraatiossa ei sinäänsä ole juuri mitään uutta Novan konfigurointiin verrattuna. Kaikki yhteydet määritellään samalla tavalla lukuunottamatta liitteen rivejä 24-32, joissa määritetään yhteys novaan. Isoimmat erot tulevat riveillä 2-3, joissa otetaan käyttöön niin sanottu Modular Layer 2-plugin ja disabloidaan kaikki muut pluginit käytöstä.

Modular Layer 2 on viitekehys, joka mahdollistaa OSI-mallin tasolla kaksi sijaitsevien verkkoteknologioiden, kuten kytkimien käyttämistä OpenStackin verkkopalvelun kanssa. (OpenStack Foundation, 2016). Konfiguroidaan siis ML2-plugin, jotta virtuaalikoneet voisivat ottaa yhteyttä ulkoverkkoon ja kommunikoida keskenään. Avataan tiedosto /etc/neutronplugins/m12/m12_conf.ini tiedosto ja lisätään sinne kohdan [m12] alle ”type:drivers = flat, vlan”. Tämä ottaa käyttöön flatverkot, jotka tarkoittavat verkkoja joissa kaikki koneet ovat suoraan kytkimessä kiinni, sekä VLAN-verkot eli virtuaaliset lähiverkot, jotka on eristetty isommasta verkosta. Lisätään myös kohta ”tenant_network_types =”, joka ottaa pois muut verkkotyypit käytöstä, sillä käytämme yksinkertaista provider-ratkaisua. Määritetään myös verkko käyttämään Linux bridge-agentteja kommunikointiin lisäämällä parametri ”mechanism_drivers = linuxbridge”. Lisäksi määritetään [m12_type_flat] kohdan alle käyttämämme provider-verkkotyyppi parametrilla ”flat_networks = provider”.

Tämän jälkeen päästään konfiguroimaan Linux bridge agentin, joten avataan konfiguraatiotiedosto /etc/neutron/plugins/m12/linuxbridge_agent.ini. Lisätään tänne kohdan [linux_bridge] alle parametri ”physical_interface_mappings = provider:eno1”, joka määrittää Linux bridge agentin käyttämään fyysisellä tietokoneella olevaa verkkokorttia, mikä sattuu olemaan nimeltään eno1 Ubuntu serverin nimeämiskäytännön takia. Kannattaa myös ottaa OpenStackin sisäinen palomuuri käyttöön virtuaalikoneita varten, joten laitetaan kohdan [securitygroups] alle ”parametri enable_security_group = True”, sekä lisätään palomuurin ajuri ”firewall_driver = neutron.agent.linux.iptables_firewall.IptablesFirewallDriver”.

OpenStack tarvitsee dhcp:tä, jotta virtuaalikoneet saisivat ip-osoitteet automaattisesti. Tämän takia konfiguroidaan OpenStackin dhcp-agent, joka mahdollistaa sen. Avataan konfiguraatiotiedosto /etc/neutron/dhcp_agent.ini ja lisätään kohdan [DEFAULT] alle ensimmäisenä Linux bridge agentin ajuri parametrilla ”interface_driver = neutron.agent.linux.interface.BridgeInterfaceDriver”. Lisäksi otetaan käyttöön Neutronin dhcp-ajuri lisäämällä parametri ”dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq”. Lisäksi mahdollistetaan virtuaalikoneiden pääsy OpenStackin metadataan parametrilla ”enable_isolated_metadata = True”.

Jos esimerkiksi virtuaalikone haluaa tietää oman julkisen ip-osoitteensa tai ssh-avaimensa, se hakee tämän tiedon OpenStackin metadatasta. Tämän takia on hyvin tärkeää, että metadata otetaan käyttöön OpenStack-järjestelmässä.

Otetaan siis näitä tietoja hallitseva metadata-agentti käyttöön tiedostosta /etc/neutron/metadaga_agent.ini. Lisätään tänne sen hostina toimiva Controller-node parametrilla ”nova_metadata_ip = controller”. Lisäksi laitetaan nämä metadatatiedot salatuiksi lisäämällä ”metadata_proxy_shared_secret = *salasana”.* Myös virutaalikoneita ajavan Nova-moduulin on päästävä käsikis Neutronin tietoihin, joten lisätään nämä autentikaatiotiedot sen konfiguraatiotiedostoon nova.conf (Liite 2), jota muokattiin jo aiemmin Novan asennuksen yhteydessä. Nämä tiedot näkyvät liitteen 2 riveillä 34-42. Lisäksi otetaan tämä metadata käyttöön riveillä 43-44. Rivillä 44 määriteltävä salasana on sama mikä laitetaan metadata-agentin konfiguraatioon. Nyt voidaan siirtää Neutronin tiedot tietokantaan OpenStackin valmiilla skripteillä ajamalla komento:
”su -s /bin/sh -c "neutron-db-manage --config-file /etc/neutron/neutron.conf --config-file /etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron”.

Myös Compute-node tarvitsee oman Linux bridge agenttinsa, joten asennetaan se sinne komennolla ”apt-get install neutron-linuxbridge-agent”. Mahdollistetaan ensiksi yhteys Neutroniin lisäämällä parametreja tiedostoon /etc/neutron/neutron.conf. Tehdyt lisäykset vastaavat aiemmin Controller-noden vastaavan tiedostoon tehtyjä hyvin suurilta osin(Liite 5). Tänne lisättävät rivit ovat liitteessä olevat rivit 3-4 ja 10-23. Näissä määritetään autentikointi messagequeue-palveluun ja Keystoneen. Tämän jälkeen konfiguroidaan Linux bridge agentti identtisellä tavalla Controller-noden vastaavan kanssa. Lisätään myös Novan-konfiguraatiotiedostoon samat Neutronin autentikaatiotiedot, kuin Controller-noden Nova-konfiguraatiossa(Liite 3, rivit 33-42). Tässä vaiheessa Neutronin asennus on valmis ja asennetussa OpenStack-ympäristössä on teoriassa mahdollista ajaa virtuaalikoneita.

3.7.7 Cinder-asennus

Jotta virtuaalikoneemme saisivat pysyvää tallennustilaa, asennetaan block storage moduuli Cinder. Tässä testiasennuksessa se tullaan asentamaan omalle nodelleen nimeltään Block. Asennus on aloitettu samalla tavalla, kuin muiden moduuleiden kanssa luomalla Cinder niminen tietokanta, jonka käyttö onnistuu annettun salasanan avulla. Tämän lisäksi on tehty käyttäjä Cinderiä varten, ja tälle käyttäjälle on annettu admin-rooli Service-projektiin, jossa kaikki moduulit sijaitsevat. OpenStackiimme on myös lisätty palvelu nimeltään Cinder ja tehty julkinen, yksityinen ja admin api-endpoint.

Asennetaan Controller-nodelle Cinderin api sekä Cinder-scheduler-palvelu, joka ylläpitää Cinderiä hallitsemalla käytössä olevia resursseja (Sparkymycloud 2016) komennolla: ”apt-get install cinder-api cinder-scheduler”. Tämän jälkeen konfiguroidaan Cinder tiedostosta cinder.conf (Liite 6). Konfiguraatioprosessi on samantapainen muiden moduulien kanssa. Määritetään yhteydet tietokantaan, Keystoneen sekä messagequeue-palveluun. Konfiguroidaan myös Nova käyttämään Cinderiä (Liite 2, rivit 45-46) Tämän jälkeen voimme täyttää Cinderin tietokannan komennolla: ”su -s /bin/sh -c "cinder-manage db sync" cinder”.

Seuraavaksi konfiguroidaan Cinder toimimaan Block-nodella. Luodaan ensin LVM-osio Block-noden kovalevylle: ”pvcreate /dev/sdb4”. LVM-osio mahdollistaa joustavamman ja automaattisen osioinnin, jonka avulla järjestelmä pystyy jatkuvasti muuttuvien vaatimusten tasalla ja tekee laajentamisesta helppoa. Tehdään lisäksi lvm-ryhmä jonne Cinder tallentaa nämä loogiset levyt ajamalla komento: ”vgcreate cinder-volumes /dev/sdb4”.

Asennetaan Cinderin paketit komennolla ”apt-get install cinder-volume”. Tämän jälkeen avataan tiedosto cinder.conf (Liite 7) konfigurointia varten. Konfiguraatio on hyvin samantapainen Controller-nodella tehdyn Cinderkonfiguraation kanssa. Eroina ovat liitteen riveillä 18-22 tehdyt lvm-konfiguraatiot. Riveillä 19-20 otetaan käyttöön Cinderin lvm-ajuri sekä määritetään aiemmin tekemämme lvm-ryhmä. Riveillä 21-22 otetaan käyttöön iSCSI eli internet small computer system interface, joka mahdollistaa virtuaalikoneiden datan lähetyksen tcp-protokollalla verkon ylitse Cinderissä sijaitsevaan iSCSI-pohjaiseen varastoon (Techtarget 2016). Cinderin asennus on tämän jälkeen valmis.

3.7.8 Horizon-asennus

Viimeisenä asennetaan graafinen käyttöliittymä OpenStack-järjestelmälle. Tämän asentaminen ei olisi välttämätöntä sillä OpenStackia pystyy käyttämään täysin komentorivin varassa. Graafinen käyttöliittymä on kuitenkin hyvin nopea asentaa ja se lisää käyttömukavuutta, sekä havainnollistaa hyvin miten järjestelmä toimii. Dashboardin toiminta vaatii alussa asennetun Apache-serverin asentamista.

Asennetaan ensimmäisenä Dashboardin paketit komennolla ”apt-get install openstack-dashboard”. Dashboard konfiguroidaan tiedostosta local-settings.py (Liite 8). Kahdella ensimmäisellä rivillä määritellään, että Dashboardia ylläpidetään Controller-nodella ja, että sinne pääsee yhdistämään kaikista muista verkon tietokoneista. Riveillä 3-6 määritetään Dashboard käyttämään memcachedia, joka asennettiin alussa kirjaitumistokeneiden säilytystä varten. Rivillä 9 määritellään käyttämiemme moduulien apiversiot. Rivit 10-11 määrittelevät, että dashboardin kautta luodut käyttäjät kuuluvat Default domainiin ja että heidän käyttäjätyyppi on normaali user. Lisäksi riveillä 12-20 otetaan pois käytöstä OpenStackin edistyneempiä verkko-ominaisuuksia sillä tässä asennuksessa käytetään yksinkertaisempaa provider-ratkaisua. Tämän konfiguroinnin jälkeen Dashboard on valmis käyttöön.

3.7.9 OpenStackin toimivuuden testaus

Nyt kun kaikki tarvittavat moduulit on asennettu, ympäristön toimintaa voidaan testata ja katsoa saadaanko virtuaalikone käynnistettyä. Yritetään ensin laittaa Glanceen käyttöjärjestelmäimage sisään. Tätä varten on ladattu niin sanottu Ubuntu Cloud levykuva. Nämä Cloud levykuvat ovat valmiiksi esiasennettuja levykuvia, jotka on muokattu erityisesti pilvipalvelualustoja, kuten OpenStackia varten. Alla olevassa kuvassa 6. image nimeltään ubuntu tehdään ensimmäisellä rivillä ”openstack image create”-komennolla. Parametrilla ”--file xenial-server…” määritetään, että image tehdään aikaisemmin ladatusta Ubuntu Cloud levykuvasta. Seuraavaksi määritetään levykuvan tyyppi kohdassa ”—disk-format qcow2”. Qcow2 on levykuvatyyppi QEMU-hypervisorilla toimiville virtuaalialustoille.

Lisäksi määritetään parametrilla ”--container-format bare” ettei levykuvan mukana ole mitään metadataa. Viimeisempänä komennon osana ”--public” määrittää, että levykuva on julkinen eli kaikkien OpenStack-ympäristömme sisäisten projektien käytettävissä. Komennon alapuolella näkyy OpenStackin palauttamaa informaatiota juuri luodusta imagesta.

```
root@controller:~# openstack image create "ubuntu" --file xenial-server-cloudimg-amd64-dis
k1.img --disk-format qcow2 --container-format bare --public
+------------------+------------------------------------------------------+
| Field            | Value                                                |
+------------------+------------------------------------------------------+
| checksum         | 35fd929b17d6c3532d04aa66e3d6ab7e                     |
| container_format | bare                                                 |
| created_at       | 2016-12-02T15:22:11Z                                 |
| disk_format      | qcow2                                                |
| file             | /v2/images/59212eb1-8c11-40c3-90c7-43de0707e6a8/file |
| id               | 59212eb1-8c11-40c3-90c7-43de0707e6a8                 |
| min_disk         | 0                                                    |
| min_ram          | 0                                                    |
| name             | ubuntu                                               |
| owner            | 6014bf918b17448f90947524b6e1f15d                     |
| protected        | False                                                |
| schema           | /v2/schemas/image                                    |
| size             | 315228160                                            |
| status           | active                                               |
| tags             |                                                      |
| updated_at       | 2016-12-02T15:22:12Z                                 |
| virtual_size     | None                                                 |
| visibility       | public                                               |
+------------------+------------------------------------------------------+
root@controller:~#
```

Kuva 6. Openstackin imagen luomista

Novan toimivuuden voi tarkistaa yksinkertaisesti katsomalla ovatko kaikki asennetut palvelut päällä. Tämä onnistuu ajamalla komento ”openstack computer service list”. Alla olevassa kuvassa 7. Novan palvelut näkyy, että kaikki kolme asennettua Controllerilla sijaitsevaa Novan palvelua, sekä Computella sijaitseva nova-compute ovat päällä. Tästä kertoo kohdassa status näkyvä enabled ja kohdassa state näkyvä up.

```
root@controller:~# openstack compute service list
+----+------------------+------------+----------+---------+-------+----------------------------+
| Id | Binary           | Host       | Zone     | Status  | State | Updated At                 |
+----+------------------+------------+----------+---------+-------+----------------------------+
|  7 | nova-consoleauth | controller | internal | enabled | up    | 2016-10-25T12:37:55.000000 |
|  8 | nova-scheduler   | controller | internal | enabled | up    | 2016-10-25T12:38:05.000000 |
|  9 | nova-conductor   | controller | internal | enabled | up    | 2016-10-25T12:37:57.000000 |
| 12 | nova-compute     | compute    | nova     | enabled | up    | 2016-10-25T12:38:02.000000 |
+----+------------------+------------+----------+---------+-------+----------------------------+
```

Kuva 7. Novan palvelut

Seuraavaksi testataan, että verkon tekeminen Neutronilla virtuaalikoneita varten on mahdollista. Luodaan ensiksi verkko nimeltään Provider. Tämä tapahtuu komennolla: “neutron net-create --shared --provider:physical_network provider --provider:network_type flat provider”

Parametri –shared määrittää, että verkko on kaikkien OpenStackissa olevien projektien käytössä. Lisäksi parametrissa ”–provider:physical_network provider” määritetään verkon kytkeytyvän eno1 verkkokorttiin, joka konfiguroitiin OpenStackin käyttöön aiemmin Neutronin asennuksessa.

Tämän jälkeen täytyy luoda vielä aliverkko minne virtuaalikoneet tulevat kuulumaan. Alla olevassa kuvassa 8. aliverkon luominen tehdään ensimmäisellä rivillä komennolla ”neutron subnet-create” Parametrilla –allocation-pool määritetään mistä ip-osoiteavaruudesta verkko tulee jakamaan osoitteita. Lisäksi määritetään nimi, käytetty dns-palvelu, reititin ja verkko. Jos verkon luominen onnistuu, OpenStack palauttaa tiedot kyseisestä verkosta.

```
root@controller:~# neutron subnet-create --name provider --allocation-pool start
=172.28.130.5,end=172.28.130.30 --dns-nameserver 8.8.8.8 --gateway 172.28.1.254
provider 172.28.175.0/16
Created a new subnet:
+-------------------+-----------------------------------------------------+
| Field             | Value                                               |
+-------------------+-----------------------------------------------------+
| allocation_pools  | {"start": "172.28.130.5", "end": "172.28.130.30"}   |
| cidr              | 172.28.0.0/16                                       |
| created_at        | 2016-11-07T15:38:44                                 |
| description       |                                                     |
| dns_nameservers   | 8.8.8.8                                             |
| enable_dhcp       | True                                                |
| gateway_ip        | 172.28.1.254                                        |
| host_routes       |                                                     |
| id                | e8e50cd5-643c-4287-a0ac-ca5f7f887de6                |
| ip_version        | 4                                                   |
| ipv6_address_mode |                                                     |
| ipv6_ra_mode      |                                                     |
| name              | provider                                            |
| network_id        | 57322c0b-4d60-4349-922d-6d164d1d806c                |
| subnetpool_id     |                                                     |
| tenant_id         | 6014bf918b17448f90947524b6e1f15d                    |
| updated_at        | 2016-11-07T15:38:44                                 |
+-------------------+-----------------------------------------------------+
root@controller:~#
```

Kuva 8. Aliverkon luominen

Myös Cinderin toiminnan voi tarkistaa katsomalla ovatko asennetut palvelut päällä. Tämä tapahtuu alla olevassa kuvassa 9. komennolla: ”cinder service-list”

```
root@controller:~# cinder service-list
+------------------+------------+------+---------+-------+----------------------------+-----------------+
|      Binary      |    Host    | Zone |  Status | State |         Updated_at         | Disabled Reason |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
| cinder-scheduler | controller | nova | enabled |   up  | 2016-12-02T15:14:35.000000 |        -        |
|  cinder-volume   | block@lvm  | nova | enabled |   up  | 2016-12-02T15:14:31.000000 |        -        |
+------------------+------------+------+---------+-------+----------------------------+-----------------+
root@controller:~#
```

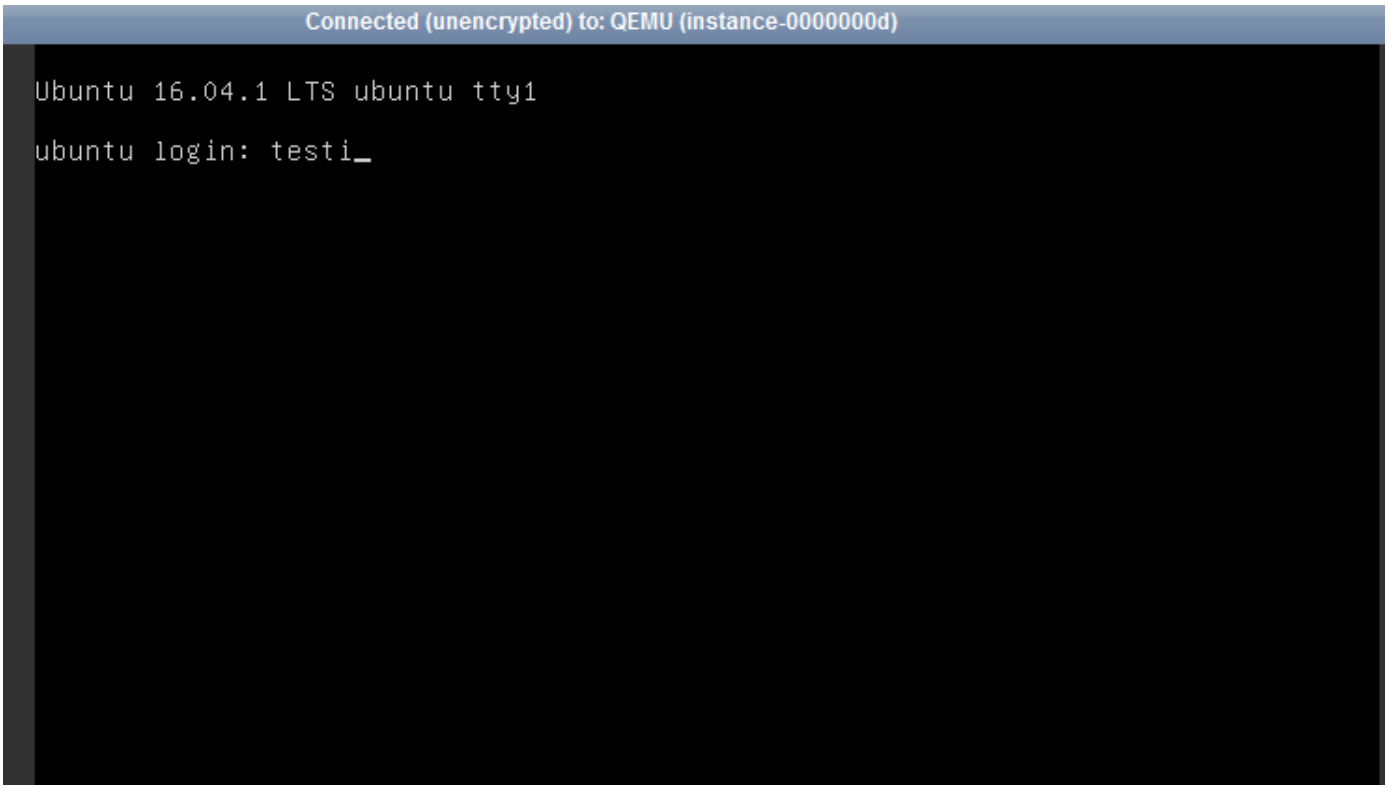
Kuva 9. Cinderpalvelut

Kaikki asennetut palvelut näyttävät toimivan, joten nyt voidaan siirtyä virtuaalikoneen käynnistykseen. Käytetty ubuntun levykuva tukee julkisen avaimen salauksella toimivaa autentikaatiota, joten sen ottaminen käyttöön kannattaa. Suoritetaan ensin komento: ssh-keygen -q -N "", joka luo avainparin. Tämän jälkeen komennolla “openstack keypair create --public-key ~/.ssh/id_rsa.pub mykey” luodaan mykey-niminen avainpari, äsken ssh-keygenillä tehdyn ssh-avainparin perusteella.

Liitten 9. ensimmäisellä rivillä esitetään virtuaalikoneen nimeltä ubuntutesti käynnistys komennolla ”openstack server create”. Ensimmäinen parametri ”–flavor” määrittää kuinka paljon resursseja virtuaalikoneelle annetaan. Tässä käytetään OpenStackin oletusflavoria ”m1.medium”, joka antaa virtuaalikoneelle käyttöön kaksi prosessoriydintä, 40-gigaa kovalevytilaa ja 4096-megaa keskusmuista. ”--image" määrittää käytetyn levykuvan, joka on tässä tapauksessa aiemmin luotu ubuntu-image. Tämän jälkeen otetaan käyttöön aliverkko paremetrilla ”--nic net-id=..". Perässä oleva pitkä merkkijono on aiemmin luodun verkon satunnaisesti generoitu id. Virtuaalikone lisätään vielä oletuksena käytettyyn palomuuriryhmään: ”--security-group default”. Viimeisenä otetaan vielä käyttöön avainpari mykey: ”—key-name mykey”.

Oikein tehdyn käynnistyskomennon jälkeen OpenStack palauttaa pitkän listan tietoja käynnistettävästä koneesta (Liiteessä 9. ensimmäisen suoritetun komennon alla). Tärkein tieto on ”status BUILD”, joka kertoo, että virtuaalikone on käynnistymässä. Virtuaalikoneen käynnistymistä voi seurata kuvassa toisena ajetulla komennolla ”openstack server list”. OpenStackin palauttamissa tiedoissa näkyy, että virtuaalikone on saanut ID:n, mutta se on vielä BUILD-tilassa. Lisäksi kohdassa Networks näkyy, että se on saanut ip-osoitteen DHCP:llä halutusta ip-osoiteavaruudesta.

Suoritetaan vielä sama komento uusiksi hetken päästä jolloin status on muuttunut aktiiviseksi. Tämä tarkoittaa, että virtuaalikone on käynnistynyt ja toimiva. Tämän jälkeen on hyvä testata graafinen käyttöliittymä Dashboardin toiminta yhdistämällä sieltä konsoliyhteys virtuaalikoneeseen. Dashboardiin pääsee käsiksi osoitteesta 172.28.130.1/horizon. Kirjaudutaan sisään Defaultprojektiin admintunnuksilla ja valitaan valikosta haluttu projekti ja välilehti instances. Tässä välilehdessä näkyvät kaikki projektiin kuuluvat virtuaalikoneet. Valitaan äsken tehty ubuntutesti ja siirrytään välilehteen Console. Tämä käynnistää automaattisesti konsoliyhteyden virtuaalikoneeseen. Alla olevassa kuvassa 10. näkyy yhdistetty konsoliyhteys virtuaalikoneeseen.



Kuva 10. Konsoliyhteys virtuaalikoneeseen

4 Pohdinta

OpenStackin testiasennus oli mielenkiintoinen, mutta työläs prosessi. OpenStackin modulaarinen rakenne on sen paras ominaisuus, koska se mahdollistaa erittäin laajan muokattavuuden ja yhteensopivuuden eri alustojen kanssa. Toisaalta tämä on myöskin OpenStackin kirous, sillä tämän vuoksi kymmenet konfiguraatiotiedostot sisältävät tuhansia mahdollisia parametrejä, mikä tekee OpenStackin asentamisen erittäin monimutkaiseksi. Ongelman sattuessa komentorivi täyttyy kymmenien rivien mittaisista kryptisistä virheviesteistä, jotka sisältävät rivi riviltä missä osassa OpenStackin-koodia virhe on sattunut. Virheenselvitys kannattaakin aloittaa lokitiedostoista, mutta mistä niistä? Jokaisessa moduulissa on yleensä useita lokitiedostoja, jotka sisältävät virheiden lisäksi kaikki moduulissa tapahtuneet asiat. Tämän vuoksi oikean virheen löytäminen oikeasta lokista on erittäin työlästä. Virhekohdan löytyminen ei vielä ratkaise ongelmaa, vaan virheestä täytyy etsiä tietoa internetistä, tai kysyä apua OpenStack-yhteisön irc-kanavalta. OpenStack-yhteisö on onneksi erittäin avulias ja verkosta löytyy paljon keskustelua useista eri virhetilanteista. Loppujen lopuksi ongelmat selvisivät aina.

Tämänkin asennuksen aikana tuli useita ongelmatilanteita, joista suurin osa johtui onneksi yksinkertaisista kirjoitusvirheistä. Esimerkiksi Mariadb:n asennuksen kanssa tuli oikeitakin ongelmia. Mariadb:n konfiguraatiotiedosto my.cnf sisältää parametrin ”!includedir /etc/my.cnf.d”. Tämä tarkoittaa, että konfiguraatiotiedoston pitäisi ottaa huomioon kaikki tiedostot tästä kansiosta. Jostain syystä Mariadb ei tehnyt tätä vaan käytti oletuskonfiguraatioita haluttujen sijasta. Tämän takia tietokannan merkistöksi tuli ”utf8mb4”, jota OpenStack ei pystynyt käyttämään. Ongelma korjaantui luomalla tietokannat uudestaan ja konfiguroimalla Mariadb suoraan my.cnf-tiedostoon. Kyseinen bugi oli onneksi dokumentoitu, joten apu löytyi nopeasti (Launchpad, 2016).

Onnistuneen asennuksen jälkeen OpenStackin käyttö on erittäin helppoa. Virtuaalikoneiden käynnistäminen komentoriviltä, sekä graafisesta käyttöliittymästä on erittäin nopeaa. Virtuaalikoneen saa päälle komentoriviltä yhdellä komennolla muutamassa sekunissa ja graafisessa käyttöliittymässä halutut ominaisuudet saa valittua muutamalla klikkauksella. Konsolihteys on erittäin toimiva OpenCloudin vastaavaan verrattuna. OpenCloudin konsoliyhteys on kokemuksieni mukaan erittäin hidas ja pätkivä. OpenStackin konsoliyhteys ei katkea uudelleenkäynnistettäessä virtuaalikonetta, kun taas OpenCloudissa yhteys kokemuksieni mukaan katkeaa joka kerta.

Kokemuksieni perusteella asennuksen suorittaminen virtuaalikoneelle olisi ollut järkevämpi ratkaisu. Projektin asennusvaiheen eteneminen kärsi huomattavasti, sillä tietokoneet joissa asennus suoritettiin, olivat arkisin yleensä varattuna 8-16 opetuksen aikaan. Asennuksen suoritus vaati neljän tietokoneen yhtäaikaista käyttöä(kolme nodea + yksi kone asennukseen), joten esimerkiksi työskentely opetuksen aikana ei ollut mahdollista. Virtuaalikonetta käyttämällä olisin voinut tehdä projektin kotona ja työskennellä aina silloin, kun haluan. Toisaalta oli hyvä kokemus saada aikaiseksi toimiva useamman noden ympäristö, koska tälläisen tekeminen on monimutkainen prosessi.

Kokemuksieni perustella voin sanoa OpenStackin olevan tarpeeksi vakaa ja toimiva yrityksien tuotannossa olevien pilvipalveluiden käyttöön. Testauksen aikana virtuaalikoneet eivät kaatuneet kertaakaan ja niiden ylläpitäminen oli helppoa. OpenStackin laajat dokumentaatiot ja asennusohjeet mahdollistavat asennuksen suorittamisen monimutkaiseenkin ympäristöön.

Projekti oli oppimiskokemuksena erittäin hyödyllinen. Opin merkittävästi OpenStackin rakenteesta ja siitä miten se konfiguroidaan. Olen myöskin käyttänyt koko elämäni ajan pääsiallisesti Windowsia, joten projekti toimi myös hyvänä keinona kehittää Linuxin käyttöä. Asennuksen suorittaminen pelkältä komentoriviltä opetti minulle paljon linuxkomentojen hyödyntämisestä.

Projektia olisi mahdollista jatkaa opinnäytetyön jälkeenkin laajentamalla sitä uusien moduulien avulla. Esimerkiksi asentamalla Heat-moduuli olisi mahdollista laajentaa OpenStack-ympäristöni tukemaan sovellusten jakamista pilven kautta. Lisäksi olisi mielenkiintoista asentaa toinen compute-node, jotta voisin tutkia miten OpenStack yhdistää nodejen resurssit virtuaalikoneiden ajamista varten.

Lähteet

Armburst, M., Fox, A., Griffith, R., Joseph, A., Katz, R., Konwinski, A., Gunho, L., Pattersson, D., Rabikin, A., Stoica, I. & Zaharia. M. 2010. A view of cloud computing. Luettavissa: https://dl.acm.org/citation.cfm?doid=1721654.1721672. Luettu 12.11.2016.

Techtarget 2015. Definiton: private cloud (internal cloud or corporate cloud). Luettavissa: http://searchcloudcomputing.techtarget.com/definition/private-cloud. Luettu 2.12.2016

Interroute 2016. What Is a Public Cloud. Luettavissa: http://www.interoute.com/cloud-article/what-public-cloud. Luettu 2.12.2016

Apperenda 2016. IaaS, PaaS, SaaS (Explained and Compared). Luettavissa: https://apprenda.com/library/paas/iaas-paas-saas-explained-compared/ Luettu 2.12.2016

Gatner 2016. Software as a Service(Saas). Luettavissa: https://www.gartner.com/it-glossary/software-as-a-service-saas/. Luettu 3.12.2016

Joshi, S. 2016. What is a platform as a Service (PaaS)?. Luettavissa: https://www.ibm.com/blogs/cloud-computing/2014/02/what-is-platform-as-a-service-paas/ Luettu 3.12.2016

Gartner 2016. Infrastructure as a Service(Iaas). Luettavissa: https://www.gartner.com/it-glossary/infrastructure-as-a-service-iaas/ Luettu 3.12.2016

Rashid, F. The dirty dozen: 12 cloud security threats. Luettavissa: http://www.infoworld.com/article/3041078/security/the-dirty-dozen-12-cloud-security-threats.html. Luettu 4.12.2016

Järvinen, T. 2011. Pilviteknologia ja tietoturva. Luettavissa: https://wiki.aalto.fi/download/attachments/58941866/pilvipalvelut_ja_tietoturva_tj_2011_public.pdf?version=1&modificationDate=1320931339000&api=v2. Luettu 4.12.2016

Salo, I. 2014. Big data ja pilvipalvelut. Docendo.

Blodget, H. 2011. Amazon's Cloud Crash Disaster Permanently Destroyed Many Customers' Data. Luettavissa:
http://www.businessinsider.com/amazon-lost-data-2011-4?r=US&IR=T&IR=T
Luettu 4.12.2016

OpenStack Foundation 2016. Luettavissa: https://www.openstack.org/
Luettu 15.11.2016

OpenStack Foundation 2016. Keystone, the OpenStack Identity Service. Luettavissa:
http://docs.openstack.org/developer/keystone/. Luettu 5.12.2016

Suse Suomi 2016. Kyselytutkimus: Valtaosa suurista yrityksistä investoinut tai investoimassa OpenStack-pohjaisiin yksityisen pilven ratkaisuihin. Luettavissa:
https://www.sttinfo.fi/tiedote/kyselytutkimus-valtaosa-suurista-yrityksista-investoinut-tai-investoimassa-openstack-pohjaisiin-yksityisen-pilven-ratkaisuihin?publisherId=31369299&releaseId=39768436. Luettu 5.12.2016

Lando, G. 2014. A Game of Stacks : OpenStack vs. CloudStack.
Luettavissa: https://www.getfilecloud.com/blog/2014/02/a-game-of-stacks-openstack-vs-cloudstack/ Luettu 5.12.2016

OpenStack Foundation 2016. OpenStack Installation Guide for Ubuntu. Luettavissa:
http://docs.openstack.org/mitaka/install-guide-ubuntu/. Luettu: 1.10.2016

Chrony 2016. Introduction. Luettavissa: https://chrony.tuxfamily.org/ Luettu: 5.10.2016

OpenStack Foundation 2016. Glance documentation. Luettavissa
http://docs.openstack.org/developer/glance/configuring.html. Luettu 11.11.2016

Dolph, M. 2016. OpenStack Keystone Fernet tokens. Luettavissa:
http://dolphm.com/openstack-keystone-fernet-tokens/. Luettu 15.11.2016

OpenStack Foundation 2016. Nova documentation. Luettavissa:
http://docs.openstack.org/developer/nova/cells.html. Luettu 3.12.2016

Cloudy Stuff Happens 2016. Understanding nova-conductor in OpenStack Nova. Luettavissa: https://cloudystuffhappens.blogspot.fi/2013/04/understanding-nova-conductor-in.html. Luettu 4.12.2016

Redhat 2016. Getting Started Guide Luettavissa: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/2/html/Getting_Started_Guide/ch13s02.html. Luettu 5.12.2016

Redhat 2016. Configuring the Linux Bridge Plug-in Agent. Luettavissa: https://access.redhat.com/documentation/en-US/Red_Hat_Enterprise_Linux_OpenStack_Platform/3/html/Installation_and_Configuration_Guide/Configuring_the_Linux_Bridge_Plug-in.html. Luettu 6.12.2016

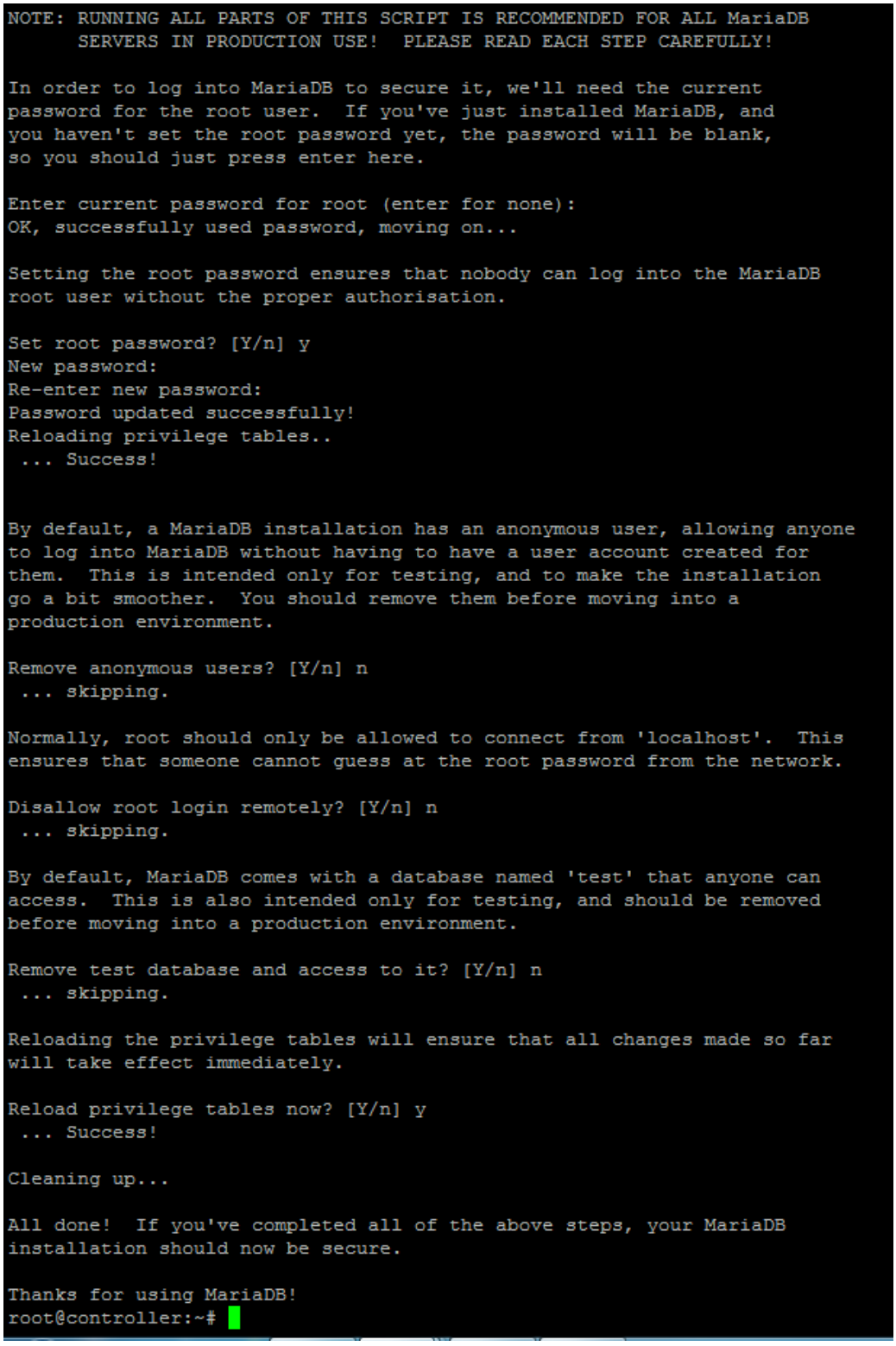
OpenStack Foundation 2016. Neutron/ML2. Luettavissa: https://wiki.openstack.org/wiki/Neutron/ML2 Luettu: 6.12.2016

Techtarget 2016. Definition: iSCSI-initator. Luettavissa: http://searchstorage.techtarget.com/definition/iSCSI-initiator. Luettu 6.12.2016

Launchpad 2016. Newton on 16.04 Xenial and mysql's new default utf8mb4 charset. Luettavissa: https://bugs.launchpad.net/openstack-manuals/+bug/1575688. Luettu 6.12.2016

Liitteet

Liite 1. Mariadb:n mysql_secure_installation skripti



```
NOTE: RUNNING ALL PARTS OF THIS SCRIPT IS RECOMMENDED FOR ALL MariaDB
      SERVERS IN PRODUCTION USE!  PLEASE READ EACH STEP CAREFULLY!

In order to log into MariaDB to secure it, we'll need the current
password for the root user.  If you've just installed MariaDB, and
you haven't set the root password yet, the password will be blank,
so you should just press enter here.

Enter current password for root (enter for none):
OK, successfully used password, moving on...

Setting the root password ensures that nobody can log into the MariaDB
root user without the proper authorisation.

Set root password? [Y/n] y
New password:
Re-enter new password:
Password updated successfully!
Reloading privilege tables..
 ... Success!


By default, a MariaDB installation has an anonymous user, allowing anyone
to log into MariaDB without having to have a user account created for
them.  This is intended only for testing, and to make the installation
go a bit smoother.  You should remove them before moving into a
production environment.

Remove anonymous users? [Y/n] n
 ... skipping.

Normally, root should only be allowed to connect from 'localhost'.  This
ensures that someone cannot guess at the root password from the network.

Disallow root login remotely? [Y/n] n
 ... skipping.

By default, MariaDB comes with a database named 'test' that anyone can
access.  This is also intended only for testing, and should be removed
before moving into a production environment.

Remove test database and access to it? [Y/n] n
 ... skipping.

Reloading the privilege tables will ensure that all changes made so far
will take effect immediately.

Reload privilege tables now? [Y/n] y
 ... Success!

Cleaning up...

All done!  If you've completed all of the above steps, your MariaDB
installation should now be secure.

Thanks for using MariaDB!
root@controller:~#
```

Liite 2. Mod_wsgi-konfiguraatio /etc/apache2/sites-available/wsgi-keystone.conf

```
1.  Listen 5000
2.  Listen 3535
3.  <VirtualHost *:5000>
4.    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
      group=keystone display-name=%{GROUP}
5.   WSGIProcessGroup keystone-public
6.  WSGIScriptAlias / /usr/bin/keystone-wsgi-public
7.  WSGIApplicationGroup %{GLOBAL}
8.   WSGIPassAuthorization On
9.   ErrorLogFormat "%{cu}t %M"
10.  ErrorLog /var/log/apache2/keystone.log
11.  CustomLog /var/log/apache2/keystone_access.log combined
12.  <Directory /usr/bin>
13.  Require all granted
14.  </Directory>
15. </VirtualHost>
16. <VirtualHost *:35357>
17.  WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
      group=keystone display-name=%{GROUP}
18.  WSGIProcessGroup keystone-admin
19.  WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
20.   WSGIApplicationGroup %{GLOBAL}
21.  WSGIPassAuthorization On
22. ErrorLogFormat "%{cu}t %M"
23. ErrorLog /var/log/apache2/keystone.log
24.  CustomLog /var/log/apache2/keystone_access.log combined
25.   <Directory /usr/bin>
26. Require all granted
27.  </Directory>
28. </VirtualHost>
```

Liite 3. Glancen api konfiguraatio /etc/glance/glance-api.conf

```
1. [database]
2. connection = mysql+pymysql://glance:openstack@controller/glance
3. [keystone_authtoken]
4. auth_uri = http://controller:5000
5. auth_url = http://controller:35357
6. memcached_servers = controller:11211
7. auth_type = password
8. project_domain_name = default
9. user_domain_name = default
10. project_name = service
11. username = glance
12. password =  salasana
13. [paste_deploy]
14. flavor = keystone
15. [glance_store]
16.  stores = file,http
17. default_store = file
18. filesystem_store_datadir = /var/lib/glance/images/
```

Liite 4. Controller-noden novakonfiguraatio /etc/nova/nova.conf

```
[DEFAULT]
enabled_apis = osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
 my_ip = 172.28.130.1
use_neutron = True
firewall_driver = nova.virt.firewall.NoopFirewallDriver
 [api_database]
connection = mysql+pymysql://nova:openstack@controller/nova_api
[database]
connection = mysql+pymysql://nova:openstack@controller/nova
[oslo_messaging_rabbit]
rabbit_host = controller
 rabbit_userid = openstack
rabbit_password = salasana
[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
memcached_servers = controller:11211
auth_type = password
project_domain_name = default
user_domain_name = default
project_name = service
username = nova
password = salasana
[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip
[glance]
api_servers = http://controller:9292
[oslo_concurrency]
lock_path = /var/lib/nova/tmp

[neutron]
url = http://controller:9696
auth_url = http://controller:35357
auth_type = password
project_domain_name = default
```

38. user_domain_name = default
39. region_name = RegionOne
40. project_name = service
41. username = neutron
42. password = *salasana*
43. service_metadata_proxy = True
44. metadata_proxy_shared_secret = *salasana*
45. [cinder]
46. os_region_name = RegionOne

Liite 5. Controller-noden Neutronkonfiguraatio /etc/neutron/neutron.conf

1. [DEFAULT]
2. core_plugin = ml2
3. service_plugins =
4. rpc_backend = rabbit
5. auth_strategy = keystone
6. notify_nova_on_port_status_changes = True
7. notify_nova_on_port_data_changes = True
8. [database]
9. connection = mysql+pymysql://neutron:openstack@controller/neutron
10. [oslo_messaging_rabbit]
11. rabbit_host = controller
12. rabbit_userid = openstack
13. rabbit_password = *salasana*
14. [keystone_authtoken]
15. auth_uri = http://controller:5000
16. auth_url = http://controller:35357
17. memcached_servers = controller:11211
18. auth_type = password
19. project_domain_name = default
20. user_domain_name = default
21. project_name = service
22. username = neutron
23. password = *salasana*
24. [nova]
25. auth_url = http://controller:35357
26. auth_type = password
27. project_domain_name = default
28. user_domain_name = default
29. region_name = RegionOne
30. project_name = service
31. username = nova
32. password = *salasana*

Liite 6. Controller-noden Cinderkonfiguraatio /etc/cinder/cinder.conf

1. [DEFAULT]
2. auth_strategy = keystone
3. rpc_backend = rabbit
4. my_ip = 172.28.130.1
5. [database]
6. connection = mysql+pymysql://cinder:openstack@controller/cinder
7. [keystone_authtoken]
8. auth_uri = http://controller:5000
9. auth_url = http://controller:35357
10. memcached_servers = controller:11211
11. auth_type = password
12. project_domain_name = default
13. user_domain_name = default
14. project_name = service
15. username = cinder
16. password = *salasana*
17. [oslo_concurrency]
18. lock_path = /var/lib/cinder/tmp
19. [oslo_messaging_rabbit]
20. rabbit_host = controller
21. rabbit_userid = openstack
22. rabbit_password = *salasana*

Liite 7. Block-noden cinderkonfiguraatio /etc/cinder/cinder.conf

1. [DEFAULT]
2. rpc_backend = rabbit
3. auth_strategy = keystone
4. my_ip = 172.28.130.3
5. enabled_backends = lvm
6. glance_api_servers = http://controller:9292
7. connection = mysql+pymysql://cinder:openstack@controller/cinder
8. [keystone_authtoken]
9. auth_uri = http://controller:5000
10. auth_url = http://controller:35357
11. memcached_servers = controller:11211
12. auth_type = password
13. project_domain_name = default
14. user_domain_name = default
15. project_name = service
16. username = cinder
17. password = *salainen*
18. [lvm]
19. volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
20. volume_group = cinder-volumes
21. iscsi_protocol = iscsi
22. iscsi_helper = tgtadm
23. [oslo_concurrency] lock_path = /var/lib/cinder/tmp
24. [oslo_messaging_rabbit]
25. rabbit_host = controller
26. rabbit_userid = openstack
27. rabbit_password = *salainen*

Liite 8. Dashboardin konfiguraatio /etc/openstack-dashboard/local-settings.py

```
1. OPENSTACK_HOST = "controller"
2. ALLOWED_HOSTS = ['*',]
3. SESSION_ENGINE = 'django.contrib.sessions.backends.cache'
4. CACHES = {
5. 'default': {
6. 'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache', 'LO-
   CATION': 'controller:11211',}}
7. OPENSTACK_KEYSTONE_URL = "http://%s:5000/v3" % OPENSTACK_HOST
8. OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
9. OPENSTACK_API_VERSIONS = { "identity": 3, "image": 2, "volume": 2,}
10. OPENSTACK_KEYSTONE_DEFAULT_DOMAIN = "default"
11. OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
12. OPENSTACK_NEUTRON_NETWORK = {
13. 'enable_router': False,
14. 'enable_quotas': False,
15. enable_distributed_router': False,
16. 'enable_ha_router': False,
17. 'enable_lb': False,
18. 'enable_firewall': False,
19. 'enable_vpn': False,
20. 'enable_fip_topology_check': False,}
```

Liite 9. Virtuaalikoneen käynnistystä

```
root@controller:~# openstack server create --flavor m1.medium --image ubuntu --nic net-id=573
22c0b-4d60-4349-922d-6d164d1d806c --security-group default --key-name mykey ubuntutesti
+--------------------------------------+-----------------------------------------------+
| Field                                | Value                                         |
+--------------------------------------+-----------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                        |
| OS-EXT-AZ:availability_zone          |                                               |
| OS-EXT-SRV-ATTR:host                 | None                                          |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                          |
| OS-EXT-SRV-ATTR:instance_name        | instance-0000000d                             |
| OS-EXT-STS:power_state               | 0                                             |
| OS-EXT-STS:task_state                | scheduling                                    |
| OS-EXT-STS:vm_state                  | building                                      |
| OS-SRV-USG:launched_at               | None                                          |
| OS-SRV-USG:terminated_at             | None                                          |
| accessIPv4                           |                                               |
| accessIPv6                           |                                               |
| addresses                            |                                               |
| adminPass                            | VaA3Qz2R6Jri                                  |
| config_drive                         |                                               |
| created                              | 2016-12-02T15:27:55Z                          |
| flavor                               | m1.medium (3)                                 |
| hostId                               |                                               |
| id                                   | 7ec8cdbd-48a9-4102-aeec-250cff8b399a          |
| image                                | ubuntu (59212eb1-8c11-40c3-90c7-43de0707e6a8) |
| key_name                             | mykey                                         |
| name                                 | ubuntutesti                                   |
| os-extended-volumes:volumes_attached | []                                            |
| progress                             | 0                                             |
| project_id                           | 6014bf918b17448f90947524b6e1f15d              |
| properties                           |                                               |
| security_groups                      | [{u'name': u'default'}]                       |
| status                               | BUILD                                         |
| updated                              | 2016-12-02T15:27:56Z                          |
| user_id                              | 93e9c367771c4233b46f7dc9fb66916b              |
+--------------------------------------+-----------------------------------------------+
root@controller:~# openstack server list
+--------------------------------------+-------------+--------+------------------------+
| ID                                   | Name        | Status | Networks               |
+--------------------------------------+-------------+--------+------------------------+
| 7ec8cdbd-48a9-4102-aeec-250cff8b399a | ubuntutesti | BUILD  | provider=172.28.130.18 |
+--------------------------------------+-------------+--------+------------------------+
root@controller:~# openstack server list
+--------------------------------------+-------------+--------+------------------------+
| ID                                   | Name        | Status | Networks               |
+--------------------------------------+-------------+--------+------------------------+
| 7ec8cdbd-48a9-4102-aeec-250cff8b399a | ubuntutesti | ACTIVE | provider=172.28.130.18 |
+--------------------------------------+-------------+--------+------------------------+
root@controller:~#
```