Pilvipalvelut ja OpenStack Cloud Software

HAMK
HÄMEEN AMMATTIKORKEAKOULU

Ammattikorkeakoulun opinnäytetyö

Tietojenkäsittelyn koulutusohjelma

Visamäki, 5.6.2013

Pasi Kuivasaari

HAMK
HÄMEEN AMMATTIKORKEAKOULU

TIIVISTELMÄ

Hämeenlinna
Tietojenkäsittelyn koulutusohjelma
Systeemityö

Tekijä Pasi Kuivasaari **Vuosi** 2013

Työn nimi Pilvipalvelut ja OpenStack Cloud Software

TIIVISTELMÄ

Opinnäytetyön toimeksiantajana oli Hämeen ammattikorkeakoulu. Työn aloitushetkellä toimeksiantajalla ei ollut tietoa OpenStackista. Aihe oli hyödyllinen, koska sitä voisi hyödyntää tulevaisuudessa tietojenkäsittelyn opintojaksoilla ja myös muissa opinnäytetöissä. Aihe on myös hyödyllinen monille yrityksille ja yhteisöille.

Opinnäytetyön tarkoitus oli tutkia OpenStackia, sen eri komponentteja, asennusta ja käyttöä. Työ oli enimmäkseen käytäntöön perustuva, mutta työssä käytiin läpi myös käytännön osalta tärkeä teoriaosuus. Työssä käytettiin vain sähköisiä materiaaleja.

Teoriaosuudessa käsiteltiin pilvipalveluja yleisesti. Pilvipalveluista käytiin läpi pilvipalvelumallit ja -tyypit. Teoriaosuus sisälsi myös pilvipalveluiden historian ja tietoturvan. OpenStackista yleisen tiedon lisäksi käytiin läpi eri komponentit, arkkitehtuuri ja järjestelmävaatimukset.

Käytännön osuudessa käytiin läpi OpenStackin asennus skriptien avulla testiympäristöön sekä sen testaus ja käyttö. Työstä löytyi erilaisia huomioita liittyen asennukseen ja komponentteihin, havainnot ja huomiot perustuvat käytännön testaukseen. Testauksen perusteella OpenStack on kattava ja monipuolinen kokonaisuus, josta yritykset tai yhteisöt voivat rakentaa oman pilvialustan omalla tavallaan. Testausten perusteella OpenStackin asennus ja konfigurointi oli vaikeaa, mutta tämän jälkeinen käyttö kuitenkin suhteellisen helppoa.

Avainsanat Pilvipalvelu, OpenStack, virtualisointi, infrastruktuuri

Sivut 35 s. + liitteet 15 s.

Hämeenlinna
Degree Programme in Business Information Technology
System work

Author Pasi Kuivasaari **Year** 2013

Subject of Bachelor's thesis Cloud Services and OpenStack Cloud Software

ABSTRACT

This thesis was commissioned by HAMK University of Applied Sciences. In the beginning the commissioner did not have any knowledge of OpenStack. The topic was useful because it could be used later in business information technology courses and in other theses. The topic is very useful for various companies and communities.

The goal of this thesis was to research OpenStack, its various components, installation and usage. This thesis was mostly practice-based but the work went through also the theoretical part which is important as for the practice, too. Only electronic sources were used.

The theory part handled cloud services generally. Cloud services included cloud service models and types. The theory part also included the history of cloud services and data security. OpenStack part went through not only the general information but also the various components, architecture and the system requirements.

The practical part included testing and use of OpenStack after installing it to the test environment with the scripts. A variety of considerations relating to the installation and components were found. Findings and observations are based on practical testing. Testing shows that OpenStack is a comprehensive and versatile solution, which provides companies and communities to build their own cloud platform in their own way. Based on the tests OpenStack installation and configuration was difficult but after that relatively easy to use.

Keywords Cloud service, OpenStack, virtualization, infrastructure

Pages 35 p. + appendices 15 p.

SISÄLLYS

KÄSITTEIDEN MÄÄRITTELY

Apache HTTP Server	http-palvelinohjelma, joka perustuu avoimeen lähdekoodiin.
Apache License	Avoimen lähdekoodin lisenssi.
API	Application programming interface eli ohjelmointirajapinta.
CirrOS	Minimaalinen käyttöjärjestelmä, joka on tarkoitettu ajettavaksi pilvessä.
CLI	Command Line Interpreter eli komentorivitulkki.
CSA	Cloud Security Alliance.
DHCP	Dynamic Host Configuration Protocol on IP-osoitteiden jakamiseen käytetty verkkoprotokolla.
GPL	General Public License on yleisimmin käytetty avoimen lähdekoodin lisenssi.
Intranet	Tietyn ryhmän käyttöön eristetty lähiverkko.
LXC	LinuX Containers on käyttöjärjestelmätasolla toimiva virtualisointi menetelmä.
mod_wsgi,	Apache-liitännäinen, joka mahdollistaa python-ohjelmointikielen käyttämisen web-sovelluksissa.
MySQL	Oraclen omistama relaatiotietokantaohjelmisto.
NIC	Network Interface Card eli verkkosovitin.
Pilvipalvelut	Tietoteknisiä palveluita, joita tarjotaan internetissä.
Rackspace	Palveluntarjoaja, joka omistaa OpenStackin.
Red Hat	Tunnettu yhtiö, jonka liiketoimintamalli perustuu avoimeen lähdekoodiin.
Root	Unix-pohjaisten käyttöjärjestelmien pääkäyttäjä.
SAS	Serial Attached SCSI

SATA	Serial AT Attachment
SMB	Server Message Block on verkkoprotokolla, jota käytetään tiedostojen jakamiseen verkossa.
SQlite	Relaatiotietokantaohjelmisto, joka on toteutettu pienenä C-kirjastona.
SSD	Solid-state drive
sudo	Unix-ohjelma, jolla voidaan ajaa komentoja toisen käyttäjän oikeuksilla. Usein superuser tai root.
Web 2.0	World Wide Webin toinen vaihe.
VMware ESX	VMwaren yrityksille suunnattu virtualisointi-tuote.
VMware ESXi	Ilmainen ja ominaisuuksiltaan karsittu versio VMware ESX:stä
VPN	Virtual Private Networkin avulla voidaan yhdistää yksityiset verkot julkisen verkon yli.

1 JOHDANTO

Pilvipalvelut ovat tärkeässä ja koko ajan yleistyvässä osassa teknologian kehitystä. Pilvipalvelut auttavat säästämään laite- ja ohjelmistohankinnoissa, päivityksissä ja ylläpidossa. Pilvipalveluita käytetään tiedostojen ja tietojen tallentamiseen sekä yrityksissä että yksityisten henkilöiden toimesta. Pilvipalveluita käytetään myös alustoina, joiden päällä ajetaan sovelluksia. Jotta tiedot kulkevat mukana aina pilvipalveluissa, tarvitaan erilaisia tuotteita ja tekniikoita tätä varten. OpenStack tarjoaa yhden ratkaisun ja toteutuksen pilvipalveluun ja sen käyttöön.

Opinnäytetyön toimeksiantajan on Hämeen ammattikorkeakoulu (myöhemmin HAMK). HAMK on ammattikorkeakoulu Kanta-Hämeessä, Etelä-Pirkanmaalla sekä Uudellamaalla. HAMK tarjoaa koulutusta ammattikorkeakoulututkintoon ja ylempään ammattikorkeakoulututkintoon. HAMKilla ja toiseen asteen koulutusta tarjoavalla Hämeen ammattiinstituutilla on henkilöstöä noin 800 ja opiskelijoita noin 8 000. HAMKissa on tarjolla 27 koulutusohjelmaa kuudelta koulutusalalta.

Opinnäytetyön tarkoituksena on perehtyä pilvipalveluihin, avoimen lähdekoodin IaaS-pilviympäristöön OpenStackiin, OpenStackin asennukseen ja asennuksen eri osa-alueisiin. Pilvipalveluista käydään läpi erilaiset pilvipalvelumallit ja pilvityypit. Pilvipalveluiden historia, kehitys ja siihen liittyvä tietoturva käsitellään lyhyesti. OpenStackista käsitellään komponentit ja arkkitehtuuri. OpenStackin laitteisto ja järjestelmävaatimukset ovat keskeisessä osassa käytännön testausta. Käytännön testaus ja asennus käydään läpi suoraviivaisella kaavalla.

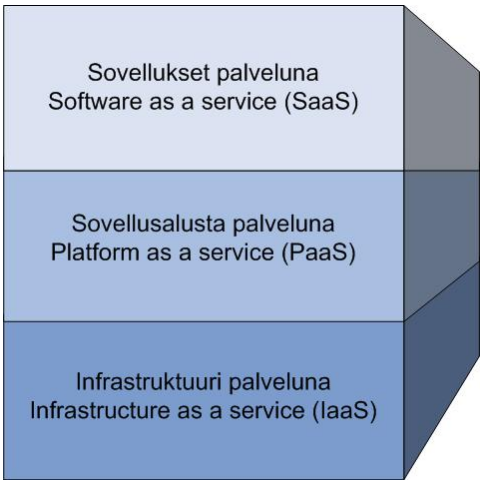
Materiaalin avulla pyritään selvittämään kattavasti ja selkeästi pilvipalveluiden sekä OpenStackin toiminta ja tarkoitus. Käytännön testaus teorian ohella monipuolistaa materiaalia sekä tekee siitä käytännön läheistä. Opinnäytetyö etenee perusasioista kohti käytännön testaamista. Käytännön testaus sisältää asennuksia, huomioita asennuksista sekä ominaisuuksien testaamista.

2 PILVIPALVELU

Pilvipalvelu (cloud service) tarkoittaa pilvessä tarjottavaa palvelua. Pilvipalveluilla on kolme pääluokkaa, joista kerrotaan kohta lisää. Tieteellisemmin voidaan puhua pilvilaskennasta. Pilvilaskenta tulee englanninkielen sanoista cloud computing. Kaikki pilvipalvelut käyttävät pilvilaskentaa ja pilvilaskenta tarkoittaa internetissä tapahtuvaa laskennan käyttöä ja kehitystä. Yleisesti pilvipalvelun etuina voidaan pitää kustannustehokkuutta, palvelun laadun parantumista, IT:n joustavuuden lisäystä ja riskien pienenemistä. Kuitenkin pilvipalveluilla on omat haittapuolensa sekä vaikeutensa, yksi suurimmista kysymyksistä on tietoturva. (Pilvilaskenta - Cloud Computing 2013; What is Cloud Computing? 2013; Pilvipalvelut n.d.)

2.1 Pilvipalvelumallit

Pilvipalvelulla on kolme erilaista palvelumallia. Pilvipalveluiden tarjoajat tekevät jaon näihin kolmeen malliin tarjotessaan palveluita asiakkailleen. Nämä kolme mallia ovat Infrastructure as a Service eli IaaS , Platform as a Service eli PaaS ja Software as a Service eli SaaS. Usein palveluista käytetään vain lyhenteitä IaaS, PaaS ja SaaS. Kuvassa 1 pilvipalvelumallit ovat palveluarkkitehtuurin mukaisessa järjestyksessä alhaalta alkaen. (Pilvipalvelumallit 2010; Cloud Computing Service Models 2013.)



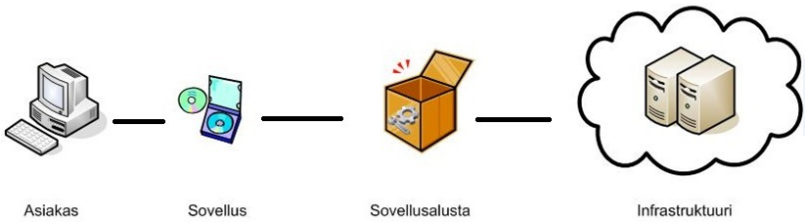
Kuva 1. Pilvipalvelumallien palveluarkkitehtuuri (Pilvipalvelumallit 2010.)

2.1.1 IaaS

IaaS eli Infrastructure as a Service, tarkoittaa suomeksi infrastruktuuria palveluna. IaaS luo pohjan kaikille muille pilvipalveluille, kuten kuvasta 2 nähdään. Kun asiakas ostaa palveluntarjoajalta infrastruktuurin palveluna (IaaS), hän saa palveluntarjoajalta laitteiston resursseja käyttöönsä. IaaSin kohdalla tarjottavat resurssit ovat usein virtualisoituja ja skaalautuminen sekä ylläpito ovat automatisoituja. IaaS-pilvipalvelumallia tarjotaan usein ”maksat mitä käytät” periaatteella. Käyttöön perustuva laskutus takaa sen, että kustannukset pysyvät mahdollisimman alhaisina, eikä minkäänlaisia etukäteissitoumuksia tarvitse tehdä. IaaS-pilvipalvelut tarjoavat usein

myös muita resursseja, kuten palomuureja, IP-osoitteita, virtuaalisia verkkoja, ohjelmistopaketteja, kuormantasaaja ja levykuvakirjastoja. (Pilvilaskenta - Cloud Computing 2013; What is Cloud Computing? 2013; Cloud Computing Service Models. 2013.)

IaaS-palveluntarjoajia ovat esimerkiksi Amazon Elastic Compute Cloud eli EC2, Rackspace Open Cloud, Google Compute Engine, HP Cloud ja Oracle Infrastructure as a Service. (Cloud Computing Service Models 2013.)

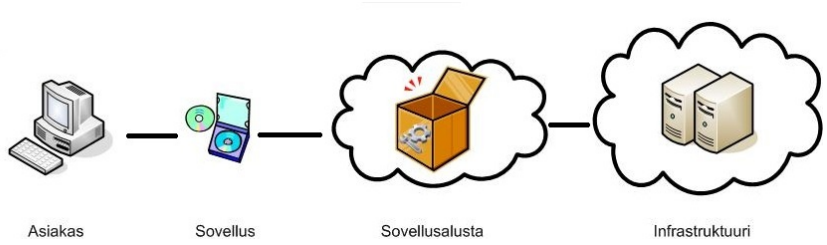


Kuva 2. IaaS pilvipalvelumalli. (Pilvipalvelumallit 2010.)

2.1.2 PaaS

PaaS eli Platform as a Service tarkoittaa suomeksi sovellusalustaa palveluna. PaaS-mallia hyödyntäen palveluntarjoaja toimittaa sovellusalustan asiakkaalle. Kuvasta 3 nähdään, että sovellusalusta toimii infrastruktuurin päällä. PaaS tarkoittaa palvelualustan ulkoistamista. Sovellusalusta sisältää tyypillisesti käyttöjärjestelmän, joillekin ohjelmointikielille ajo-alustan, tietokannan ja web-palvelimen. Ohjelmien kehittäjät voivat kehittää, testata sekä ajaa omia ohjelmiaan helposti PaaS-ympäristössä ilman erillistä rautaa ja ohjelmistotasoja. PaaS tarjoaa myös mahdollisuuden hyödyntää resursseja käyttöön perustuvalla laskutuskaavalla. Esimerkkinä voidaan käyttää tallennuskapasiteettia, joka kasvaa käytön mukaan, tällöin kapasiteettia ei tarvitse varata etukäteen ja palvelualusta on helposti laajennettavissa. (Pilvipalvelumallit 2010; Cloud Computing Service Models. 2013.)

PaaS-palveluntarjoajia ovat esimerkiksi Google App Engine, Windows Azure Cloud Services, OrangeScape, AWS Elastic Beanstalk, Cloud Foundry, Heroku, Force.com, OpenShift ja EngineYard. (Cloud Computing Service Models 2013.)

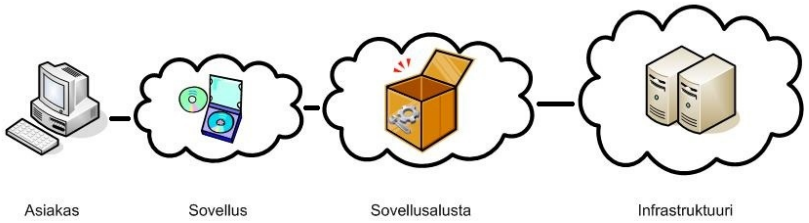


Kuva 3. PaaS pilvipalvelumalli. (Pilvipalvelumallit 2010.)

2.1.3 SaaS

SaaS eli Software as a Service tarkoittaa suomeksi sovellusta palveluna. SaaS-mallia käytettäessä palveluntarjoaja tarjoaa sovelluksen asiakkaan käyttöön, jolloin asiakkaan ei tarvitse huolehtia sovelluksen ylläpidosta, päivittämisestä tai asennuksesta. Asiakas maksaa sovelluksesta esimerkiksi käyttäjäkohtaisen, konekohtaisen tai aikaperusteisen maksun, jolloin hinta on skaalautuva ja säädeltävissä käyttäjien mukaan. Asiakas ajaa sovelluksen sovellusalustan päällä kuvan 4 mukaisesti. (Pilvipalvelumallit 2010; Cloud Computing Service Models 2013.)

SaaS palveluntarjoajia ovat esimerkiksi Google Apps, F-Secure, Microsoft Office 365, Onlive Marketo, GT Nexus, Marketo, Casengo, TradeCard. (Cloud Computing Service Models 2013.)



Kuva 4. SaaS pilvipalvelumalli. (Pilvipalvelumallit 2010.)

2.2 Pilvipalvelutyypit

Pilvipalvelutyyppejä on neljää erilaista. Nämä neljä eroavat toisistaan pääasiassa sen perusteella kenellä palveluun on pääsy ja kuka palvelua ylläpitää sekä kuka palvelun omistaa. Pilvipalvelun tyypit ovat julkinen pilvi (public cloud), yhteisöllinen pilvi (community cloud), yksityinen pilvi (private cloud) ja hybridi pilvi (hybrid cloud). (Pilvipalvelu n.d.)

2.2.1 Julkinen pilvi

Julkinen pilvi on internetissä suoritettavien tietoteknisten palveluiden allas. Palveluntarjoajat käyttävät tyypillisesti mallia, jossa käyttäjä maksaa käytön mukaan palveluistaan. Julkisella pilvellä on monia etuja. Näistä eduista merkittävimpiä ovat maksu vain käytettävistä resursseista, käyttöönoton nopeus, kapasiteetin nopea skaalautuvuus, kaikkien palveluluiden toimitus käyttövalmiina, vikasietoisuus, turvallisuus ja helppo hallittavuus. Julkisen pilven vaihtoehdot ovat jaettu julkinen pilvi ja omistettu julkinen pilvi. Usein julkiset pilvet ovat ilmaisia ennalta määrättyihin resursseihin asti, minkä jälkeen on mahdollista ostaa resursseja lisää. Esimerkkejä julkisen pilven palveluntarjoajista ovat Google, Amazon AWS ja Microsoft. Kaikki kolme esimerkkiä operoivat omia isoja infrastruktuurejaan ja palveluihin on mahdollista päästä käsiksi vain internetin välityksellä. (Loeffler 2011.)

2.2.2 Yksityinen pilvi

Yksityinen pilvi on palvelu, joka nimensä mukaisesti toimii yksityisenä. Palvelut voidaan toimittaa yrityksen sisäisen intranetin kautta. Yksityinen pilvi toimii yrityksen oman palomuurin sisäpuolella, jolloin yrityksen ulkopuoliset yhteydet tarvitsevat esimerkiksi VPN-yhteyden päästäkseen käsiksi itse pilveen niin kuin muihinkin yrityksen sisäisiin verkkoympäristöihin. (Pilvipalvelut n.d.)

Yksityisen pilven rakentaminen ja suunnittelu on kuitenkin melko haastavaa ja tarvitsee paljon suunnittelua, kouluttautumista ja paneutumista erinäisiin asioihin. Yrityksen täytyy uudelleen arvioida omat resurssinsa huolella. Kun palvelu tehdään oikein, se voi parantaa liiketoimintaa. Jokainen askel projektissa nostaa esiin uusia ongelmia tietoturvan kannalta ja tältä osalta on hyvä huomioida ongelmista jokainen. Yksityisen pilven rakentaminen on kallista, mutta saattaa kuitenkin tuoda huomattavia ja varteenotettavia etuja. Yksityisen pilven vaihtoehdot ovat itse isännöity yksityinen pilvi, isännöity yksityinen pilvi ja sovellettava yksityinen pilvi. (Loeffler 2011.)

Taulukossa 1 vertaillaan yksityistä ja julkista pilveä. Taulukosta nähdään nopeasti esimerkiksi kummankin pilvityypin kustannukset. Taulukon avulla voidaan vertailla myös muitakin ominaisuuksia, kuten yksityisyyttä.

Taulukko 1. Yksityisen ja julkisen pilven vertailu.

	Julkinen pilvi	**Yksityinen pilvi**
Alustavat kustannukset	Tyypillisesti nolla	Tyypillisesti korkeat
Käyttökustannukset	Ennalta arvattavissa	Ennalta arvaamaton
Kustomointi	Ei mahdollista	Mahdollista
Yksityisyys	Ei	Kyllä
Kertakirjautuminen	Ei mahdollista	Mahdollista
Skaalautuvuus	Helppoa määritellyissä rajoissa	Vaativaa mutta ilman rajoituksia

Taulukosta 2 voidaan nähdä, kuinka yksityisen ja julkisen pilven kaikki eri vaihtoehdot eroavat toisistaan. Taulukosta nähdään esimerkiksi isännöidäänkö pilveä ulkoisesti vai sisäisesti. Jos yritys tai yksityishenkilö on käyttöönottamassa pilvipalvelua, voidaan tämän taulukon avulla vertailla nopeasti muutamia pilvien ominaisuuksia.

Taulukko 2. Julkisen ja yksityisen pilven vaihtoehdot.

Käyttöönoton tyyppi	**Isännöinnin sijainti**	**Jaettu vai omistettu**	**Arkkitehtuurin hallinta**	**Skaalautuvuus**	**Investointi**
Jaettu julkinen pilvi	Ulkoinen	Jaettu	Palveluntarjoaja tai myyjä	Minimaalisesti skaalautuva	Käytön mukaan maksettava
Omistettu julkinen pilvi	Ulkoinen	Osittain tai kokonaan omistettu	Palveluntarjoaja tai myyjä	Sopimuksen mukaisesti skaalautuva	Käytön mukaan maksettava
Itse isännöity yksityinen pilvi	Sisäinen	Kokonaan omistettu	Käyttäjä	Pääomasijoituksen mukaisesti skaalautuva	Rakenna pilvi, jaa palvelut
Isännöity yksityinen pilvi	Ulkoinen	Kokonaan omistettu	Käyttäjä	Pääomasijoituksen tai sopimuksen mukaan skaalautuva	Vaihtelee sopimuksen mukaan, voi olla vaikutusta pääomaan
Sovellettava yksityinen pilvi	Sisäinen	Kokonaan omistettu	Palveluntarjoaja	Tarjonnan mukaan skaalautuva	Vaihtelee sopimuksen mukaan, voi olla vaikutusta pääomaan

2.2.3 Yhteisöllinen pilvi

Yhteisöllinen pilvi on usean vuokralaisen infrastruktuuri, joka on samanaikaisesti käytettävissä monella eri organisaatiolla. Pilven käyttäjillä tai vuokralaisilla on yhteiset tavoitteet tai huolenaiheet. Tavoitteena voi olla esimerkiksi tietoturvan parantaminen. Yhteisöllisen pilven hallinta voi olla sisäisesti hoidettava tai kolmansien osapuolien hoidettavana. Yhteisöllinen pilvi voidaan isännöidä sisäisillä resursseilla tai ulkoistaa. Pilven kustannukset jaetaan useammille käyttäjille kuin yksityisessä pilvessä, mutta kuitenkin vähemmälle kuin julkisen pilven osalla. Yhteisöllisen pilven tavoit-

teena on saada organisaatiot ymmärtämään ja hyödyntämään julkisen pilven hyödyt yksityisemmin ja turvallisemmin. (Rouse 2012; Ames 2012.)

2.2.4 Hybridipilvi

Hybridipilvi koostuu kahdesta tai useammasta pilvestä. Nämä pilvet voivat olla yksityisiä, yhteisöllisiä tai julkisia pilviä. Kaksi pilveä toimivat yksilöllisinä kokonaisuuksina, mutta ovat sidottuna toisiinsa tarjoten monen palvelun edut. Yhdistämällä tavallisen yksityisen pilven julkiseen pilveen voidaan hallita odottamattomia muutoksia työkuormassa. Verrattuna muihin pilvityyppeihin haittapuolena hybridipilvessä on monen eri turvallisuusrajapinnan pakollinen ylläpito ja samalla huolenpito siitä, että kaikki tarvittavat liiketoiminnan osa-alueet keskustelevat keskenään pilven avulla. (Ames 2012.)

Taulukko 3. Esimerkkitilanteita, joissa hybridipilvi on erinomainen vaihtoehto

	Tilanne
1.	Yritys haluaa käyttää SaaS-sovellusta mutta on huolissaan tietoturvasta.
2.	Yritys tarjoaa palveluita, jotka on räätälöity eri osa-alueille. On mahdollista käyttää julkista pilveä asiakasyhteyksiin, mutta pitää asiakkaiden tiedot turvattuna yksityisessä pilvessä.
3.	Yritys voi tarjota julkisen pilven asiakkailleen ja samalla sisäinen IT voi käyttää yksityistä pilveä.

2.3 Historia

Pilvilaskenta on kehittynyt monen eri vaiheen lävitse. Vaiheet sisältävät ruudukko- ja hyödyllisyyslaskennan, sovellus palvelutuotannon ja ohjelmiston palveluna. Idean ”tähtienväliseen tietokone verkkoon” (intergalactic computer network) esitteli kuusikymmentä luvulla J.C.R. Lickliderin. Hän oli vastuussa ARPANET (Advanced Research Projects Agency Network) kehityksen mahdollistamisesta. Hänen näkemyksensä oli, että jokainen maapallolla liitetään yhteen, jolloin kaikilla on mahdollisuus päästä käsiksi mihin tahansa sivuun ja ohjelmaan mistä tahansa maapallolla. Monet asiantuntijat yhdistävät käsitteen pilvilaskenta tiedemieheen nimeltä John McCarthy, joka ehdotti että laskenta toimitetaan yhteishyödylliseksi. (Mohamed n.d, Maynard 2013.)

Kuusikymmentä luvun jälkeen pilvilaskenta on kehittynyt monien mutkien kautta nykytilanteeseen. Kaikkein viimeaikaisin evoluutio on Web 2.0. Koska internet alkoi tarjota suurempaa kaistanleveyttä vasta yhdeksänkymmentäluvulla, kehittyi pilvilaskenta nykymuotoonsa vasta myöhään. (Mohamed n.d; Maynard 2013.)

Yksi ensimmäisistä välietapeista pilvilaskennan kannalta oli Salesforce.com:in saapuminen vuonna 1999. Salesforce.com tarjosi yritysovelluksia yksinkertaisen web-sivun välityksellä. Seuraava kehitysaskel oli Amazon Web Services vuonna 2002. Se tarjosi pilvi-pohjaisia palvelui-

ta, kuten tietovaraston, laskentaa ja jopa ihmisälykkyyttä Amazon Mechanical Turk:in kautta. Sitten 2006 vuonna Amazon käynnisti Elastic Compute cloud (EC2) palvelun kaupallisena web-palveluna, jonka avulla pienyritykset ja yksityishenkilöt voivat vuokrata tietokoneita ja ajaa omia ohjelmiaan niissä. Amazon EC2/S3 oli ensimmäinen laajalti saatavilla oleva pilvilaskenta infrastruktuuripalvelu. (Mohamed n.d; History of Cloud Computing; Maynard 2013.)

Seuraava suuri välietappi tuli vuonna 2009, kun Web 2.0 otti suuren askeleen, koska Google sekä muut palveluntarjoajat alkoivat tarjota selainpohjaisia yrityssovelluksia palveluidensa kautta, esimerkkinä Google Apps. Merkittävimpiä edistäjiä pilvilaskennalle voidaan pitää teknologian jättiyrityksiä, kuten Google ja Microsoft. Muita avaintekijöitä, jotka ovat mahdollistaneet pilvilaskennan virtualisointiteknologian kehityksen, ovat nopeat kehittyneet laajakaistayhteydet ja yhteensopivuutta edistävät ohjelmistostandardit. Tällä hetkellä on mahdollista toimittaa lähes mikä tahansa palvelu tai ohjelma pilvessä. On myös selvää, että pilvilaskenta voi tuoda valtavaa hyötyä IT-käyttäjille. Kuitenkin monet IT-johtajat ovat sitä mieltä, että heidän täytyy jatkaa sisäisten IT-palveluiden hoitamista sekä samalla oppia, kuinka suojata, hallita ja valvoa kasvavaa valikoimaa pilvessä asuvia ulkoisia resursseja. (Mohamed n.d; History of Cloud Computing n.d; Maynard 2013.)

2.4 Tietoturva

Pilvipalveluiden tietoturva on yksi yleisemmistä puheenaiheista vuonna 2013. Usein organisaation ensimmäinen askel pilvipalvelun käyttöönotossa on määritellä tarkasti, missä suurimmat pilvipalveluun liittyvät uhat sijaitsevat. Tätä varten CSA eli Cloud Security Alliance on määrittänyt yhdeksän korkeinta uhkaa vuodelle 2013. CSA:n logo on kuvassa 5. (Samson 2013.)

Kuva 5. Cloud Security Alliancen logo.

Ensimmäisenä listalla on tietomurrot ja esimerkkinä tämän uhan suuruudesta CSA on esittänyt tutkimusdokumentin, kuinka virtuaalikone voisi käyttää sivukanavan ajoitustietoa poimiakseen täydellisen yksityisen kryptograafisen avaimen, joka on käytössä toisella virtuaalikoneella samaisella serverillä. Ilkeän hakkerin ei välttämättä tarvitsisi nähdä tällaista vaivaa saavuttaakseen tämänkaltaisen tehtävän. Jos monen asiakkaan pilvipalvelun tietokanta ei ole suunniteltu kunnolla, niin yksikin pieni virhe asiakkaan ohjelmassa voisi sallia hyökkääjälle muidenkin asiakkaiden tiedot. Tämänkaltaisen tietovuodon välttämiseksi voitaisiin salata tiedot salausavaimella. Kuitenkin on otettava huomioon, että jos salaus-avaimen hukkaa, tarkoittaa se myös kaiken tiedon hukkaamista. Jos haluaa pitää offli-

ne-varmuuskopiot kaikesta tiedosta välttääkseen tiedon häviämisen, altistuu silloinkin enemmän tietomurroille. (Samson 2013.)

Toiseksi suurin uhka pilvipalveluympäristössä on tiedon häviäminen. Ilkeä hakkeri voisi poistaa kohteensa tietoja ilkeyttään. On kuitenkin mahdollisuus, että tiedot katoaisivat myös huolimattoman palveluntarjoajan toimesta tai muun katastrofin kautta, kuten tulipalon, tulvan, tai muun luonnonkatastrofin. Tietojen menetys ei ole ongelmallista ainoastaan asiakkaan näkökulmasta. Pilven ylläpitäjä voi joutua myös viranomaisten haaviin, jos ei noudata ennalta sovittuja sääntöjä tai mahdollisesti lakeja koskien esimerkiksi yksityisyyden suojaa. (Samson 2013.)

Kolmanneksi suurin turvallisuusuhka pilviympäristössä on käyttäjätunnusten kaappaus. Pilvipalvelu tuo uuden maiseman tälle osa-alueelle. Jos hyökkääjä saa pääsyn käyttäjätunnukseen, hän voi vakoilla käyttäjän toimintaa ja liiketoimia, manipuloida tietoa, väärentää tietoa sekä ohjata käyttäjän mahdolliset omat asiakkaat väärennetyille tai saastuneille sivustoille. Käyttäjän tunnuksista ja palveluista saattaa tulla hyökkääjän oma perusta. Tämänkaltaisen hyökkäyksen estämiseksi organisaatioiden täytyisi kieltää täysin käyttäjätietojen ja tunnusten jako käyttäjien välillä. Organisaatioiden olisi myös suositeltavaa käyttää vahvaa kaksipuolista suojaustekniikkaa aina kun se on mahdollista. (Samson 2013.)

Neljäs uhka listalta on epävarmat rajapinnat ja muut Application Programming Interfacet eli API:t. IT-järjestelmänvalvojat luottavat pilvipalveluiden rajapintoihin, hallintaan ja seurantaan. API:t ovat integroitavissa tietoturvaan ja yleisiin pilvipalveluihin. Tätä kautta organisaatiot ja kolmannet osapuolet voivat rakentaa omia rajapintoja ja sisäistää lisä-osia palveluihin. Tämä luo uusia monimutkaisia kerroksellisia API:a. Se myös kasvattaa riskejä, koska organisaation täytyy mahdollisesti luovuttaa tietoja kolmansille osapuolille. Ainoa neuvo ja kehotus organisaatioille on, että he ymmärtävät turvallisuuteen liittyvät vaikutukset käytettäessä hallintasovelluksia ja mahdollisia seuranta, sekä tarkkailusovelluksia. (Samson 2013.)

Viidenneksi suurin uhka listan mukaan on palvelunestohyökkäykset (Denial-of-Service eli DoS tai Distributed Denial-of-Service eli DDos). Palvelunestohyökkäykset ovat olleet suuri uhka internetissä jo monia vuosia, mutta siitä on tullut vielä suurempi uhka, kun jotkut palvelut ovat täysin riippuvaisia palveluiden toiminnasta ympärivuorokautisesti. Palvelunestohyökkäkset voivat olla myös erittäin kalliita niille asiakkaille, jotka ostavat palveluitaan käytön mukaisesti. Hyökkääjän ei välttämättä tarvitse edes saada palvelua alas palvelunestohyökkäyksellä vaan pienempikin palvelunestohyökkäys saattaa esimerkiksi viedä niin paljon prosessoritehoa ja sitä kautta tulla niin kalliiksi, että asiakkaan on pakko ajaa itse palvelu alas. (Samson 2013.)

Seuraavana listalta löytyy haitallinen työntekijä, jolla voi olla pääsy järjestelmiin. Tätä kautta käyttäjä pääsee käyttämään järjestelmiä vääriin tarkoituksiin. Seitsemäntenä listalta löytyy pilven väärinkäyttö. Väärinkäyttäjä saattaa käyttää pilven laskentatehoa esimerkiksi salasanan kryptauksen

purkuun, jos salasana olisi liian vaikea purkaa normaalin koneen laskentateholla. Kahdeksas turvallisuusuhka on tietotaidon riittämättömyys pilvipalveluiden ylläpitäjillä. Järjestelmänvalvojien täytyy olla aina täysin perillä siitä, mitä he pilven hallintajärjestelmillä tekevät. Viimeisenä eli yhdeksäntenä listalta löytyy normaali pilvipalveluissa yleisesti käytetty jakamisteknologia. Pilvipalvelut, erityisesti IaaS käyttävät palveluita usein skaalautuvina, jolloin esimerkiksi prosessorin välimuistiin saattaa jäädä sinne kuulumatonta tietoa ja tästä syntyy yksi heikkous sekä turvallisuusuhka. (Samson 2013.)

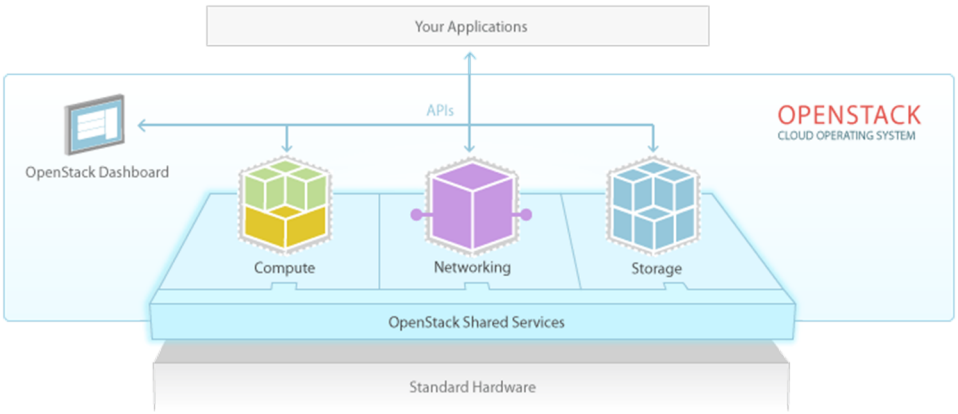
3 OPENSTACK

OpenStack on avoimen lähdekoodin ohjelmisto yksityisen ja julkisen pilven rakennukseen. Kuvassa 6 on OpenStackin logo. OpenStack on Rackspacen ja Nasan yhteisesti perustama, mutta kehitykseen on kuitenkin liittynyt mukaan jo yli 100 eri organisaatiota. OpenStack on pilvikäyttöjärjestelmä, jolla voidaan hallita suuria laskenta-altaita, varastointipalveluita ja kaikkia tietokeskuksen verkkoresursseja. Kaikkia palveluita hallitaan OpenStackin kojelaudan (dashboard) avulla, mikä toimii normaalina web-käyttöliittymänä. OpenStackin yleisimpiä käyttäjiä ovat yritykset, palveluntarjoajat, SMB:t (Server Message Block), tutkijat ja globaalit datakeskukset. OpenStackin avoimen lähdekoodin lisensointi perustuu Apache 2.0-standardiin, mikä tarkoittaa sitä, että koko ohjelmiston koodi on avoin kaikille. Kuka tahansa voi ajaa, kehittää tai muuttaa sitä lisenssin rajoissa. Apache 2.0 -lisensointimalli on sopiva myös kaupalliseen käyttöön toisin kuin GPLv3. Apache 2.0 -lisenssimalli ei vaadi oman muokatun lähdekoodin avointa jakamista. Tämän lisenssi-mallin hyödyntämisestä on tuttu esimerkiksi Red Hat. (OpenStack: The Open Source Cloud Operating System n.d; OpenStack Community Welcome Guide 2013.)

Kuva 6. OpenStackin logo.

OpenStack koostuu kolmesta suuresta osasta, jotka ovat Compute, Storage ja Network. OpenStack Compute säännöstelee ja hallitsee suuria virtuaalikoneista koostuvia verkkoja. OpenStack Computen koodinimi on Nova. OpenStack Storage toimii OpenStackin skaalautuvana tietovarastona. OpenStack Storage koostuu kahdesta eri osasta, jotka ovat Object storage (koodinimi Swift) ja Block storage (koodinimi Cinder). OpenStack Network on API:n avulla toimiva skaalautuva verkko ja IP-hallintajärjestelmä. Lisäksi OpenStack sisältää useita jaettuja palveluita, jotka toimivat kolmen pääelementin Computen, Storagen ja Networking välillä yhdistäen nämä yhdeksi kokonaisuudeksi kuvan 7 mukaisesti. Nämä jaetut palvelut pitävät sisällään muun muassa identiteettipalvelut, levykuvien hallinnan ja web-käyttöliittymän. OpenStack dashboard tarjoaa graafisen käyttöliittymän järjestelmänvalvojille ja käyttäjille päästäkseen käsiksi pilven säännöstelyn hallintaan ja automatisoituihin pilvipohjaisiin

resursseihin sekä muihin hallintaa tarvitseviin alueisiin. OpenStackilla on tällä hetkellä käytössä kuuden kuukauden julkaisusykli. (OpenStack: The Open Source Cloud Operating System n.d; OpenStack Community Welcome Guide 2013.)



Kuva 7. OpenStackin kolme osaa. (OpenStack Community Welcome Guide 2013.)

3.1 Komponentit

OpenStackin tämänhetkinen julkaisuversio Grizzly sisältää seitsemän eri ydinkomponenttia, jotka ovat koodinimiltään Nova, Glance, Swift, Horizon, Keystone, Quantum ja Cinder. Näistä komponenteista Quantum sekä Cinder ovat uusia, eivätkä sisältyneet toiseksi edelliseen versioon nimeltään Essex. Diablo, joka oli Essex-versiota aikaisempi julkaisu, tämä julkaisu ei sisältänyt kuin kolme komponenttia Novan, Glacen ja Swiftin. Seuraava versio Havana on suunniteltu julkaistavaksi lokakuussa 2013. OpenStackin vanhemmista versioista tällä hetkellä yhteisöllisesti tuettuna ovat enää Folsom, Essex ja Diablo. Turvallisuuden kannalta tuettuja versioita ovat vain uusin versio Grizzly sekä vanhempi Folsom. Kaikki OpenStackin komponentit ovat erittäin monipuolisesti konfiguroitavissa ja tämä on yksi syy, jonka takia OpenStack on niin monipuolinen järjestelmä ja käyttötarkoituksia löytyy todella paljon, mutta haittapuolena on myös monimutkaisuus. Taulukossa 4 on esitetty OpenStackin julkaisut. (Releases n.d.)

Taulukko 4. OpenStack julkaisuhistoria.

Julkaisun nimi	**Julkaisu päivämäärä**	**Sisältyvien komponenttien koodinimet**
Austin	21.10.2010	Nova, Swift
Bexar	03.02.2011	Nova, Glance, Swift
Cactus	15.04.2011	Nova, Glance, Swift
Diablo	22.09.2011	Nova, Glance, Swift
Essex	05.04.2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27.09.2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzly	04.04.2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17.10.2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder

3.1.1 Object Store

Object Storen koodinimi on Swift. Swiftin avulla voidaan tallentaa tai noutaa tiedostoja mutta ei kiinnittää hakemistoja, kuten esimerkiksi tiedostopalvelimissa. Useat yritykset tarjoavat kaupallisia varastointipalveluita perustuen Swiftiin. Näistä voidaan ottaa esimerkkinä Internap sekä Rackspace, josta Swift sai alkunsa. Swiftiä käyttävät myös monet suuret yritykset tallentaakseen omat yksityiset ja sisäiset tietonsa. Swiftin arkkitehtuuri on hyvin jakautettu, jotta pystytään estämään pienikin vika. Swift sisältää neljä eri komponenttia. Ensimmäinen komponentti on välityspalvelin, joka hyväksyy saapuvat pyynnöt OpenStack Object API:n kautta tai pelkkänä http-pyyntönä. Toisena komponenttina on käyttäjätilipalvelin, jolla hallitaan pääsyä kansioihin ja tiedostoihin. Kolmantena komponenttina tulee Container eli sisältöpalvelin, jolla hallitaan kansioiden sisältöä. Neljäntenä komponenttina on objektipalvelin, joka sisältää itse tiedostot eli storage nodet. Swift sisältää myös muita ajoitettuja prosesseja, jotka pitävät esimerkiksi huolta suurista tiedostomääristä. Swift voi myös tarjota staattisia web-sivuja http:n välityksellä. Autentikointi on hallittu toisen komponentin kautta eli Keystone-komponentin kautta. (Pepple 2012; Welcome to Swift's documentation! 2013.)

3.1.2 Image

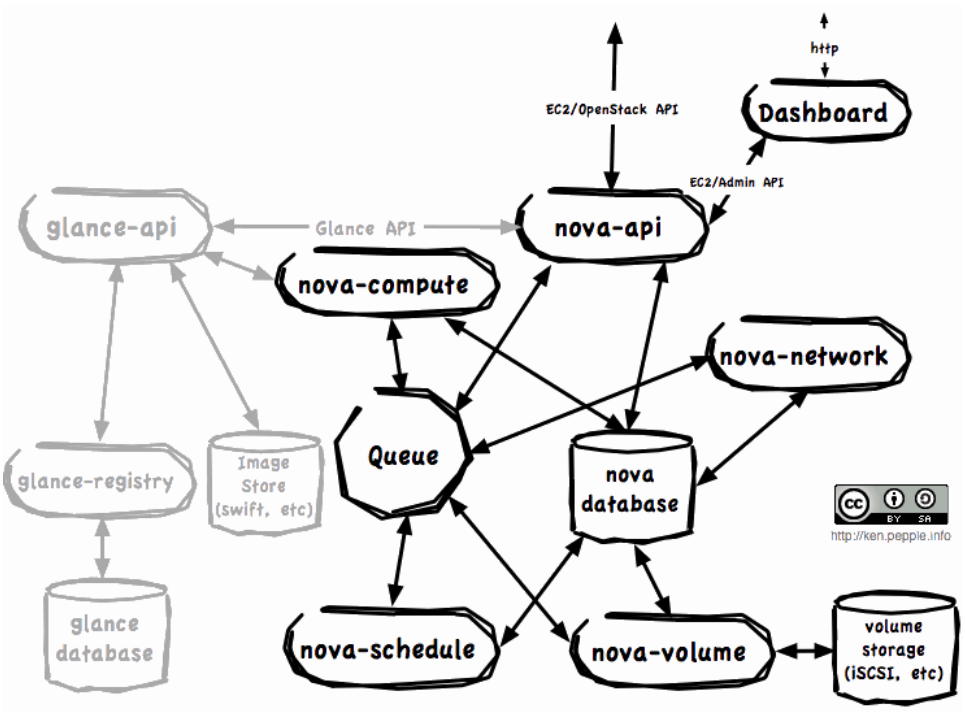
OpenStack levykuvapalvelu käyttää koodinimeä Glance. Glance sisältää neljä pääosaa tai toimintoa. Glance-api hyväksyy Image API:n pyynnöt levykuvien hakemista ja varastointia varten. Glance-registry varastoi, prosessoi ja noutaa metatietoa levykuvista, kuten levykuvan koko ja tyyppi. Glance sisältää tietokannan, johon levykuvien metatieto varastoidaan. Tietokannan tyypin, jota käytetään, voi itse määrittää. Useimmin käytössä on MySQL tai SQLite. Glance sisältää varastointisäilön itse levykuville.

Swiftiin verrattuna Glance tukee myös normaaleja levyjärjestelmiä, kuten RADOS-järjestelmiä, Amazonin S3:a ja http:ta. Kuitenkin jotkut näistä palveluista toimivat vain luettuina järjestelminä, joten kirjoitus ei onnistu. Glance sisältää myös muita pienempiä prosesseja, joiden avulla Glance tukee esimerkiksi levyn välimuistiin kirjoitusta. (Pepple 2012; Welcome to Glance's documentation! 2013.)

3.1.3 Compute

Computen koodinimi on Nova. Nova tarjoaa virtuaalisia palvelimia vaatimusten mukaisesti. Esimerkiksi NASA sekä Libre käyttävät novaa sisäisessä ympäristössään, kun taas Rackspace sekä HP tarjoavat kaupallisia palveluita perustuen Novaan. Nova on kaikista monimutkaisin komponentti koko OpenStack-ympäristössä. Nova sisältää todella suuren määrän yhteisesti toimivia prosesseja. Näiden prosessien avulla loppukäyttäjän API-pyynnöt ohjautuvat virtuaalikoneille. Nova-api hyväksyy ja vastaa loppukäyttäjän API-pyyntöihin. Se tukee OpenStack computen omaa APIa, Amazonin EC2 APIa ja Admin APIa. Nova-compute prosessi luo ja lopettaa virtuaalikoneiden instanssit valvojan APIn kautta. Valvoja API voi olla esimerkiksi XenAPI XenServerille tai XCP:lle, libvirt KVM:lle tai QEMU:lle ja VMwareAPI VMwarelle ja näin edespäin.

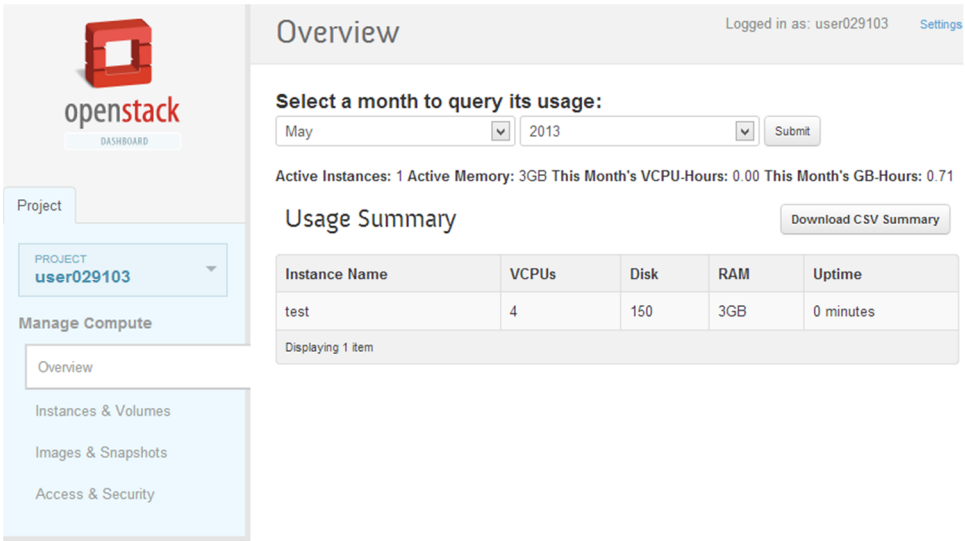
Nova-volume hallitsee pysyvien asemien luomista, liittämistä ja irroittamista. Nova-volume voi käyttää asemia eri toimittajilta, kuten iSCSI tai Rados Block Device. OpenStack Cinder komponentti tulee ajan mittaan korvaamaan kokonaan nova-volumen toiminnan. Prosessina nova-network on hyvin samanlainen toiminnaltaan kuin nova-compute ja nova-volume. Se hyväksyy verkkotehtävät omasta jonostaan ja suorittaa ne verkon mukauttamiseksi. Esimerkiksi siltausliitännän muuttaminen tai IP-taulukoiden muuttaminen ovat nova-networkin tehtäviä. Nova-networkin toiminta on lähes yhtäläinen Quantum-komponentin kanssa. Nova-schedule prosessi on yksinkertaisin osa Nova-komponenttia. Nova-schedule käsittelee virtuaalikoneinstanssin pyynnöt ja lähettää ne osaan, jossa pyynnöt kuuluu suorittaa. Nova käyttää myös tietokantaa, johon se tallentaa tiedot esimerkiksi instanssien aloitus- ja suoritusajoista. Teoreettisesti Nova tukee kaikkia tietokantoja, mutta tällä hetkellä eniten käytössä olevat tietokannat ovat sqlite3, MySQL ja PostgreSQL. Nova sisältää myös konsolipalveluita, joiden avulla loppukäyttäjät pääsevät käsiksi virtuaali-instansseihin konsolin kautta. Nova on vuorovaikutuksessa kaikkien muiden Openstakcin komponenttien kanssa. Kuvassa kahdeksan on kuvattu Nova-komponentin monimutkainen arkkitehtuuri. (Pepple 2012; Pepple 2011; Welcome to Nova's developer documentation! 2013.)



Kuva 8. Nova komponentin monimutkainen arkkitehtuuri. (Pepple 2011.)

3.1.4 Dashboard

OpenStack Dashboard tarjoaa web-käyttöliittymän OpenStackin palveluiden hallintaan sekä järjestelmänvalvojille että loppukäyttäjille. Dashboardin tai kojelaudan koodinimi on Horizon. Kuvassa 9 näkyy yksinkertainen yleisnäkymä Horizonista loppukäyttäjänä. (Pepple 2012; Horizon: The OpenStack Dashboard Project 2013.)



Kuva 9. OpenStack Horizon.

Niin kuin useimpien web-käyttöliittymien osalla, myös Horizonin arkkitehtuuri on yksinkertainen. Horizon hyödyntää myös omaa tietokantaansa, mutta itse tietoa tallennetaan erittäin vähän. Horizon hakee suurimman

osan tiedosta muiden komponenttien tietokannoista. Horizon on usein ajettu mod_wsgi Apache -komponentin välityksellä. Verkon kannalta Horizonin täytyy olla käytettävissä asiakkaille sekä vuorovaikutuksessa kaikkien muiden komponenttinen julkisiin API:hin. Eli, jos haluaa käyttää järjestelmänvalvojan toimintoja, Horizon vaatii myös yhteyden Admin API:in. Nämä toiminnot eivät kuitenkaan pitäisi olla asiakkaan käytettävissä missään tapauksessa. (Pepple 2012; Horizon: The OpenStack Dashboard Project 2013.)

3.1.5 Identity

OpenStack Identityn koodinimi on Keystone. Keystone tarjoaa tunnistautumispalvelut ja valtuutuspalvelut kaikille OpenStackin palveluille. Keystone tarjoaa myös luettelon kaikista kyseisen OpenStackin palveluista. Keystone käsittelee API-pyyntöjä tarjoten konfiguroitavan katalogin, tunnistautumistokenit ja identiteettipalvelut. Kaikki keystonen ominaisuudet ovat liitettävissä esimerkiksi LDAPiin tai KVS-varastoihin. Useimmiten käytetään tällaista kustomointia ja liittämistä käyttäjän valmiiseen tunnistautumispalveluun. (Pepple 2012; Welcome to Keystone, the OpenStack Identity Service! 2013.)

3.1.6 Network

OpenStack Network käyttää koodinimeä Quantum. Quantum on uusi komponentti Folsom-julkaisussa. Quantum tarjoaa verkkoyhteyksiä palveluna (NaaS) OpenStackin palveluiden ja verkko-laitteiden välillä. Palvelu antaa käyttäjille mahdollisuuden luoda omia verkkoympäristöjä ja liittää rajapintoja näihin jo luotuihin ympäristöihin. Quantum-server hyväksyy API-pyynnöt ja ohjaa nämä oikealle Quantum-liitännäiselle toimintaa varten. Quantumin liitännäiset suorittavat itse komennot tai toiminnot, kuten porttien ohjauksen, verkon luonnin tai IP-osoitemuutokset. Nämä liitännäiset eroavat laitteiston tarjoajan ja teknologioiden osalta riippuen siitä, mitkä ovat käytössä kyseisessä pilvessä. Quantum tukee seuraavia liitännäisiä: Cisco virtuaaliset ja fyysiset kytkimet, Nicira NVP, NEC OpenFlow, Open vSwitch, Linux siltaus, Ryu Network käyttöjärjestelmä, DHCP ja L3. Quantum on vuorovaikutuksessa lähinnä vain Novan komponentin kanssa, josta se tarjoaa verkon ja yhteydet instansseille ja niiden välille. (Pepple 2012; Welcome to Quantum's developer documentation! 2013.)

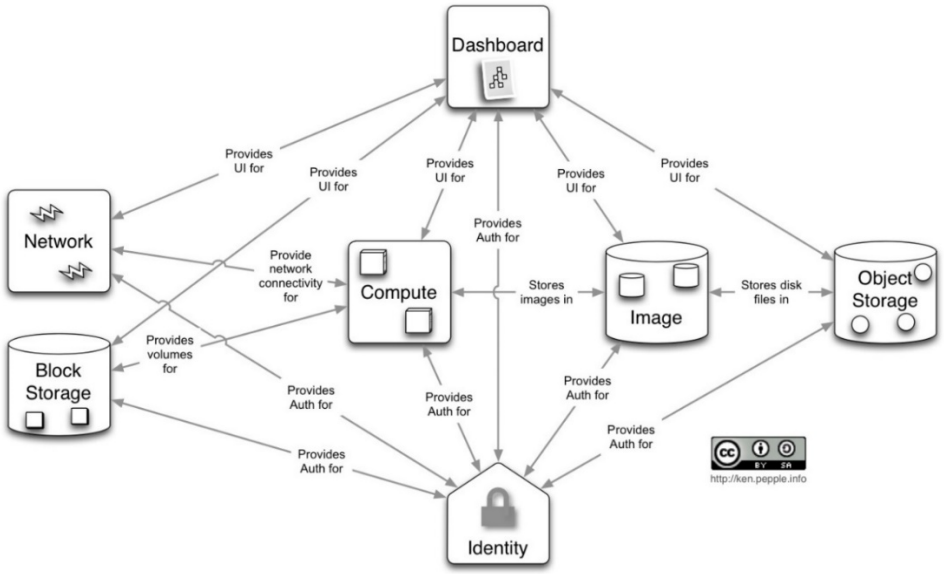
3.1.7 Block Storage

OpenStackin lohkovarasto eli block storage käyttää koodinimeä Cinder. Samoin kuin Quantum, Cinder on uusi komponentti Folsom-julkaisussa. Cinder tarjoaa pysyvän lohkovaraston vierailija-virtuaalikoneille. Cinder projekti sai alkunsa Novan sisäisestä osasta nova-volumesta. Cinder APIn avulla voidaan muokata ja hallita asemia, levyjä sekä levykuvia. Cinder-api hyväksyy API-pyynnöt ja ohjaa ne toimintoa varten cinder-volumeen. Käskyn saatuaan cinder-volume lukee tai kirjoittaa tietokantaan lohkon tilan, jolloin muut prosessit voivat olla vuorovaikutuksessa tämän kanssa

tarjoten lohkon resursseista. Cinder on vuorovaikutuksessa pääasiallisesti Nova komponentin kanssa, tarjoten levyjä novan virtuaalikoneille tai instansseille. (Pepple 2012; Cinder 2013.)

3.2 Käsitteellinen arkkitehtuuri

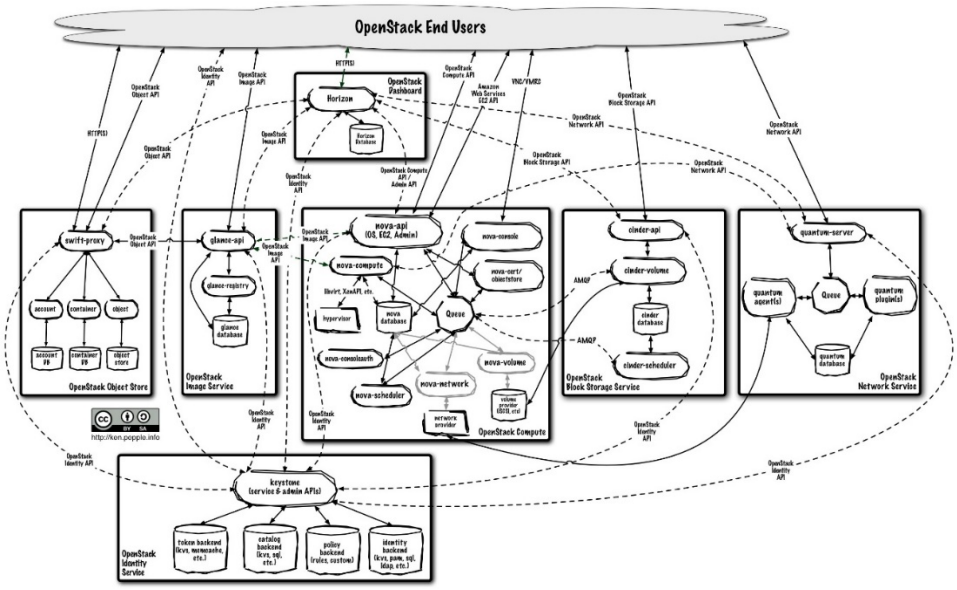
OpenStack kokonaisuutena on suunniteltu ja tarkoitettu toimittamaan massiivisesti skaalautuvana pilvikäyttöjärjestelmänä. Kokonaisen infrastuktuurin palveluna (IaaS) toiminnallisesti mahdollistaakseen jokainen palvelu ja komponentti on suunniteltu toimimaan yhteisesti. Jotta toiminnallisuus olisi sulavaa, kokonaisuus käyttää julkisia ohjelmointirajapintoja (API). Julkiset ohjelmointirajapinnat mahdollistavat sen että palvelut pääsevät käsiksi toisiinsa. Lisäksi nämä rajapinnat mahdollistavat minkä tahansa palvelun vaihtamisen toiseen palveluun, joka hyödyntää samoja rajapintoja. Kuvassa 10 näkyy, kuinka komponentit kättelevät toisiaan, esimerkiksi Keystone (Identity) tarjoaa todennuksen kaikille komponenteille. Kuva 10 esittää yksinkertaistetumman näkökulman arkkitehtuurista, jossa kaikki komponentit ovat käytössä. Kuvassa ei kuitenkaan näy muuta kuin itse operaattorin näkökulma, se ei kuvaa, kuinka asiakkaat käyttävät pilveä. Asiakas voi esimerkiksi päästä käsiksi suoraan lohkovarastoon. (Conceptual Architecture 2013.)



Kuva 10. OpenStack komponenttien suhteet. (Pepple 2012.)

3.3 Looginen arkkitehtuuri

Looginen arkkitehtuuri on paljon monimutkaisempi kuin käsitteellinen arkkitehtuuri. Kuva 11 (Liite 1) esittää yleisimmän arkkitehtuurin OpenStack pilvestä. OpenStack tukee kuitenkin monenlaista teknologiaa, joten kuva 11 ei esitä ainoaa mahdollista arkkitehtuuria. Kuva 11 esittää loppukäyttäjien vuorovaikutuksen web-rajapinnan kanssa tai suoraan eri palveluiden kanssa niiden APIen avulla. Yksittäiset palvelut ovat vuorovaikutuksessa toistensa kanssa niiden julkisten APIen kautta, paitsi silloin kun vaaditaan järjestelmänvalvojan oikeudet. (Logical Architecture 2013.)



Kuva 11. OpenStack käyttäjän suhteet. (Pepple 2012.)

Kuva 11 ei esitä tarkkoja eikä varsinkaan tyhjentäviä tietoja arkkitehtuurista, mutta tavoite on antaa merkityksellinen näkökulma. Järjestelmänvalvojan on ymmärrettävä kokonaiskuva ja arkkitehtuuri hyvin, jotta pilvialusta on mahdollista suunnitella, asentaa ja konfiguroida toimivaksi kokonaisuudeksi. Loogisessa arkkitehtuurissa moduulit on organisoitu toiminta-alueittain tyyppien mukaisesti omiin luokkiinsa. Taulukko 5 esittää tyypit, joiden mukaan moduulit on organisoitu. (Logical Architecture 2013.)

Taulukko 5. Organisointi tyypit.

Tyypit	**Lisätiedot**
Palveluprosessi	Ajaa prosessia Linux-alustan avulla, joka toimii usein palveluna.
Skripti	Tapahtuman tapahtuessa ulkopuolinen moduuli ajaa skripitin.
Asiakas	Asiakas luo Python-sidoksen palvelussa.
CLI	Komentorivitulkki toimittaa komennot esimerkiksi OpenStack Computelle.

3.4 Kilpailu ja muut palvelut

Tällä hetkellä pilvipalvelun tarjoajia ja eri järjestelmiä löytyy useita. Amazonin AWS-palvelu on asiakasmäärissä ylivoimainen voittaja ja tämän takia myös markkinajohtaja. AWS perustuu kuitenkin täysin suljettuun lähdekoodiin, joten sitä on vaikea verrata OpenStackiin. OpenStack on tällä hetkellä ainoa varteenotettava avoimen lähdekoodin järjestelmä, joka on verrattavissa Amazonin, Microsoftin tai Googlen vastaaviin järjestelmiin. Rackspace tarjoaa OpenStack-palvelua kaupallisesti samoin kuin Amazon omaa AWS-palveluaan. Myös Google, Microsoft ja useat muut tarjoavat

omia palveluitaan. Kuvassa 12 on jaoteltu palveluntarjoajia lohkoihin toteuttamiskyvyn ja vision täydellisyyden mukaan lokakuun 2012 tilanteen mukaisesti. Kuvassa esiintyvät seuraavat palvelut: Amazon Web Services, Terremark, Savvis, CSC, Dimension Data, Bluelock, Joyent, GoGrid, SoftLayer, OVH, Fujitsu, Dell, Tier 3, Rackspace ja VirtuStream. (Butler 2012.)

Kuva 12. Palveluntarjoajia jaoteltuna lohkoihin.

OpenStackin arkkitehtuuri on hyvin samanlainen kuin AWS:n. Tämän johdosta AWS ja OpenStack ovat jokseenkin yhteensopivia toistensa kanssa työkalujen ja API:en osalta. Nova on käsitteellisesti yhtäläinen AWS EC2:n kanssa. On myös olemassa monia tapoja, joilla tarjotaan Novalle yhteensopivuus AWS EC2 API:n kanssa. Swift on käsitteellisesti samankaltainen AWS S3:n kanssa ja Glance tarjoaa monia samoja ominaisuuksia kuin AWS AMI -katalogi. Cinder taas puolestaan tarjoaa samankaltaisia lohkopalveluja kuin AWS EBS-palvelu. (Pepple 2012.)

3.5 Laitteisto- ja järjestelmävaatimukset

OpenStackin komponentit on tarkoitettu ajettavaksi normaaleilla laitteistoilla. Taulukossa 6 on lueteltu vähimmäisjärjestelmävaatimukset tuotantoympäristö-käytössä. Jos verrataan muihin kilpailijoihin, esimerkiksi Rackspace käyttää jopa 96 GB RAM-muistia yhdelle compute nodelle. Vaatimuksissa täytyy erityisesti huomioida käytettävä prosessori. Jotta OpenStackissa saadaan virtualisointi toimimaan, on käytettävä x86-arkkitehtuurilla varustettua prosessoria. Virtualisointiin sopivan AMD-prosessorin tunnistaa SVM- tai AMD-V -päätteestä. Intelin vastaavan prosessorin tunnistaa VT-merkinnästä (Virtualization Technology). LXC:tä

käytettäessä VT merkintöjä ei tarvita. (Compute and Image System Requirements 2013.)

Taulukko 6. OpenStack laitteistosuositukset.

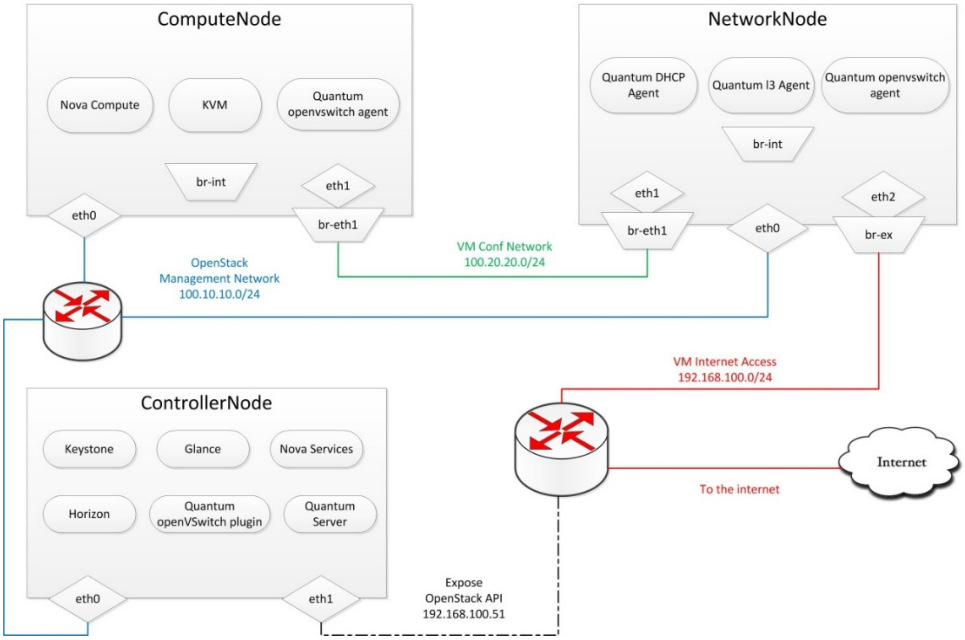
Palvelin	**Suositeltu laitteisto**	**Muut huomiot**
Cloud Controller node (Sisältää network, volume, API, ajastus ja levykuvapalvelut.)	Prosessori: 64-bit x86 Muisti: 12 GB RAM Levytila: 30 GB (SATA, SAS tai SSD) Massamuisti: 2 * 2 TB levyä compute noden asemia varten Verkko: 1 * 1 Gbps verkkokortti (NIC)	Kaksi verkkokorttia olisi suositeltavaa mutta ei pakollista. Neljä ytiminen palvelin ja 12 GB RAM-muistia olisi enemmän kuin riittävä cloud controller nodelle.
Compute nodet (ajavat virtuaali-instansseja)	Prosessori: 64-bit x86 Muisti: 32 GB RAM Levytila: 30 GB (SATA) Verkko: 2 * 1 Gbps verkkokorttia (NIC)	2 GB muistilla voi ajaa yhtä m1.small instanssia tai kolmea m1.tiny instanssia ilman memory swappia, joten 2 GB muistia compute nodelle olisi minimivaatimus testiympäristössä.

Vaikka jotkut OpenStackin osa-alueet saattavatkin toimia monilla käyttöjärjestelmillä, on tällä hetkellä vain tietyt 64-bit Linux-käyttöjärjestelmät täysin tuettuja tuotantoympäristöön. OpenStackin asennuspaketit löytyvät seuraaville jakeluversioille: CentOs, Debian, Fedora, RHEL, openSUSE, SLES ja Ubuntu. Paketit ovat ylläpidettyjä yhteisön toimesta. OpenStack vaatii pääsyn myös PostgreSQL- tai MySQL-tietokantaan. Jos tietokantaa ei löydy ennestään, voi sen asentaa OpenStackin pakettien yhteydessä. OpenStack-palvelut on asennettava suoraan root-käyttäjällä tai normaalina käyttäjänä, jolla on sudo-oikeudet. Monet OpenStackin palvelut vaativat kohdallaan olevan ajan, kuten esimerkiksi ajoitetut tehtävät ja automaattiset päivitykset. Tämän takia on suositeltavaa asentaa NTP tai muu ajan synkronointipalvelu. (Compute and Image System Requirements 2013.)

4 ASENNUS

Tuotantoympäristöön asennettaessa OpenStack vaatii vähintään kaksi palvelinta, mutta kolme palvelinta olisi vielä parempi. OpenStack pystytään asentamaan monella eri tavalla toimivaksi kokonaisuudeksi. Tämän takia asennuksen suunnittelu, varsinkin tuotantoympäristöön asennettaessa, on erittäin tärkeää. Kuva 13 esittää tuotantoympäristöön soveltuvan verkkoarkkitehtuurin. Computenode, Controllernode ja Networknode ovat kaikki

asennettu erillisille alustoilleen. Tällainen asennus vaatii erinomaista tietoa asennuskohteen verkkoympäristöstä ja palomuurisäännöistä. Huomattava on kuitenkin, että kuva 13 esittää vain esimerkin ympäristöstä, johon OpenStack on mahdollista asentaa. OpenStack on tarkoitettu täysin oman pilven rakennukseen ja tämä tarkoittaa sitä, että jokainen omiin tarpeisiin suunniteltu arkkitehtuuri näyttäisi erilaiselta.



Kuva 13. Tuotantoympäristöön soveltuva verkkoarkkitehtuuri. (Msekni 2013)

Testikäyttöön OpenStackin voi asentaa vain yhdelle palvelimelle. Tämä on hyvä tapa, jos käytettävissä on vain rajalliset järjestelmäresurssit. Testikäyttö tarkoittaa sitä, että haluaa vain testata palvelun toimintaa, eikä sitä ole tarkoitus ylläpitää kauempaa aikaa. Asennuksen avuksi internetistä löytyy erilaisia skriptejä. Näitä skriptejä hyödyntämällä jokaista komponenttia ja pienintä asetusta ei tarvitse käydä läpi samalla tavalla kuin tuotantoympäristöön asennettaessa. (Campbell 2012.)

4.1 Asennusvaiheet

OpenStack asennettiin testikäyttöön seuraavilla ohjeilla. Ainoat vaatimukset tämän ohjeen seuraamiseen on, että testijärjestelmästä löytyy vähintään 8 GB-muistia, neljällä ytimellä varustettu prosessori, kaksi kiintolevyä ja yksi tai kaksi verkkokorttia sekä puhdas asennus Ubuntu 12.04.2 LTS 64-bit palvelin -käyttöjärjestelmästä. Täytyy muistaa myös, että asennuksessa on käytetty OpenStackin Essex-versiota. Tästä versiosta puuttuu komponentit Quantum ja Cinder, kun verrataan tämänhetkiseen uusimpaan versioon Grizzlyyn. Ohjeessa käytettyjen skriptien lähdekoodit löytyvät työn liitteistä 2-9.

4.1.1 Käytettävien skriptien lataus

Ensimmäiseksi kirjautumisen jälkeen on hyvä päivittää juuri asennettu käyttöjärjestelmä. Päivityksen jälkeen ladataan ja asennetaan git.

```
sudo su
apt-get update
apt-get install git
```

Asennuksessa käytettävien StackGeek-skriptien kopiointi palvelimelle Github-sivuston kautta tapahtuu seuraavalla skriptillä:

```
git clone
http://github.com/StackGeek/OpenStackgeek.git
cd OpenStackgeek/essex
```

4.1.2 Perusskriptien asennus

Ennen skriptien ajamista olisi hyvä tietää ja ymmärtää, mitä ne tekevät. Skriptin sisällön voi tarkistaa esimerkiksi cat-tiedostonlukijalla. Kaikki käytettävät skriptit löytyvät työn liitteistä.

```
cat ./OpenStack_base_1.sh
```

Seuraava komento ajaa skriptin:

```
./OpenStack_base_1.sh
```

Verkkomääritykset konfiguroidaan manuaalisesti seuraavalla tavalla:

nano /etc/network/interfaces

Asetuksen täytyisi näyttää esimerkiksi tältä:

```
auto eth0
iface eth0 inet static
  address 10.0.1.20
  network 10.0.1.0
  netmask 255.255.255.0
  broadcast 10.0.1.255
  gateway 10.0.1.1
  dns-nameservers 8.8.8.8

auto eth1
```

Seuraavaksi suoritetaan verkon uudelleenkäynnistys, jotta määritykset astuvat voimaan. Tämän jälkeen suoritetaan toinen perus skripti:

```
/etc/init.d/networking restart
./OpenStack_base_2.sh
```

Toisen skriptin jälkeen Novalle täytyy luoda looginen asema snapshotteja sekä Novan omia asemia varten. Tämä tapahtuu fdisk-levytyökalun avulla. Fdiskiä käytettäessä täytyy olla varovainen, ettei sotke jo olemassa olevia asemia. Asema luodaan seuraavasti fdisk-työkalun avulla:

```
fdisk /dev/sdb

Device contains neither a valid DOS partition
table, nor Sun, SGI or OSF disklabel
Building a new DOS disklabel with disk identifier
0xb39fe7af.
Changes will remain in memory only, until you de-
cide to write them.
After that, of course, the previous content won't
be recoverable.

Warning: invalid flag 0x0000 of partition table 4
will be corrected by w(rite)

Command (m for help): n
Partition type:
   p   primary (0 primary, 0 extended, 4 free)
   e   extended
Select (default p): p
Partition number (1-4, default 1): 1
First sector (2048-62914559, default 2048):
Using default value 2048
Last sector, +sectors or +size{K,M,G} (2048-
62914559, default 62914559):
Using default value 62914559

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.
```

Tämän jälkeen ajetaan vielä nämä komennot, jotta uusi levy saadaan näkyviin:

```
pvcreate -ff /dev/sdb1

vgcreate nova-volumes /dev/sdb1
```

4.1.3 Tietokannan asennus

Tietokannan asennuksessa voidaan käyttää myös skriptiä. Tällä tavalla pienempiä säätöjä ei tarvitse tehdä manuaalisesti. Skripti asentaa MySQL-tietokannan ja luo esimerkiksi Novalle, Keystonelle ja Glancelle tarvittavat oikeudet tietokantaan sekä kysyy käyttäjältä käytettävät salasanat tietokantaan.

Skripti ajetaan samanlailla kuin edellä mainitutkin:

```
./OpenStack_mysql.sh
```

Skriptin suorituksen jälkeen on suositeltavaa testata tarvittavien tunnusten toiminta tietokannassa. Tietokantaan voi kirjautua seuraavilla komennoilla:

```
mysql -u root –pqwerty1
mysql -u nova –pqwerty1 nova
mysql -u keystone –pqwerty1 keystone
mysql -u glance –pqwerty1 glance
```

4.1.4 Keystone-asennus

Keystonen asennusskriptistä löytyy monia komentoja liittyen käyttöoikeuksiin. Skripit ajetaan samalla tavalla kuin aikaisemmatkin:

```
./OpenStack_keystone.sh
```

Skripti kysyy OpenStack-palveluille käytettävää salasanaa sekä salasanaa, joka syötettiin aiemmin MySQL-asennuksen aikana. Tämän jälkeen keystonelle on mahdollista lähettää kyselyitä, mutta kyselyä ennen täytyy kohdentaa stackrc-tiedosto. Keystonen pitäisi palauttaa seuraavalla komennolla lista käyttäjistä:

```
. ./stackrc
keystone user-list
```

4.1.5 Glance-asennus

Seuraavaksi ajetaan Glancen asennusskripti. Tämä asentaa normaalisti glancen ja sen eri osat, kuten glance-apin ja glance-clientin. Skripti ajetaan vastaavalla komennolla kuin aikaisemmin:

```
./OpenStack_glance.sh
```

Asennus saattaa palauttaa virheen SADeprecationWarning, mutta tämän voi kuitenkin huoletta ohittaa. Skripti lataa myös glanceen valmiin levykuvan, joten tässä saattaa kestää hetken aikaa riippuen yhteyden nopeudesta. Seuraavalla komennolla voidaan listata glancen sisältämät levykuvat:

```
glance index
```

4.1.6 Nova-asennus

Seuraavaksi asennetaan nova, joka on OpenStackin tärkein osa. Nova toimii computen lisäksi myös network-komponenttina Essex julkaisussa.

Asennus aloitetaan edelleen skriptiä hyödyntäen:

```
./OpenStack_nova.sh
```

Koska nova-skripti ei ole kovinkaan hienostunut eikä käytä oletusasetuksia, täytyy IP-määritykset tehdä itse virheettömästi. Lähes heti skriptin ajamisen jälkeen kysytään IP-määritykset, jotka täytyy itse määrittää omaan verkkoon sopivaksi. Tässä esimerkki skriptin palauttamasta kyselystä:

```
################################################
The IP address for eth0 is probably 10.0.1.35.
Keep in mind you need an eth1 for this to work.
################################################
Enter the primary ethernet interface IP:
10.0.1.35
Enter the fixed network (eg. 10.0.2.32/27):
10.0.2.32/27
Enter the fixed starting IP (eg. 10.0.2.33):
10.0.2.33
################################################
The floating range can be a subset of your cur-
rent network.  Configure your DHCP server
to block out the range before you choose it here.
An example would be 10.0.1.224-255
################################################
Enter the floating network (eg. 10.0.1.224/27):
10.0.1.224/27
Enter the floating netowrk size (eg. 32): 32
```

Fixed network on joukko IP-osoitteita, jotka ovat paikallisia compute nodeille. Jos haluaisi tai tarvitsisi käyttää isompaa verkkoa, voisi käyttää fixed network -osoitteena esimerkiksi 10.0.4.0/24 ja aloitusosoitteena 10.0.4.1 -osoitetta. Floating network on joukko osoitteita, jotka voidaan asettaa ajettaville instansseille. Esimerkiksi on mahdollista pystyttää webpalvelin ja reitittää ulkoinen IP tämän kautta tarjoten internet-sivuja. On myös mahdollista reitittää ulkoisia IP-osoitteita suoraan OpenStack instansseille, jos palomuuri, reititin ja muu verkko sallisi tämän. Nova-skripti on mahdollista ajaa uudelleen, jos haluaa muuttaa IP-määrityksiä. Kun novan asennus on suoritettu, on mahdollista saada lista glancen levykuvista myös novan kautta:

```
nova image-list
```

4.1.7 Horizon-asennus

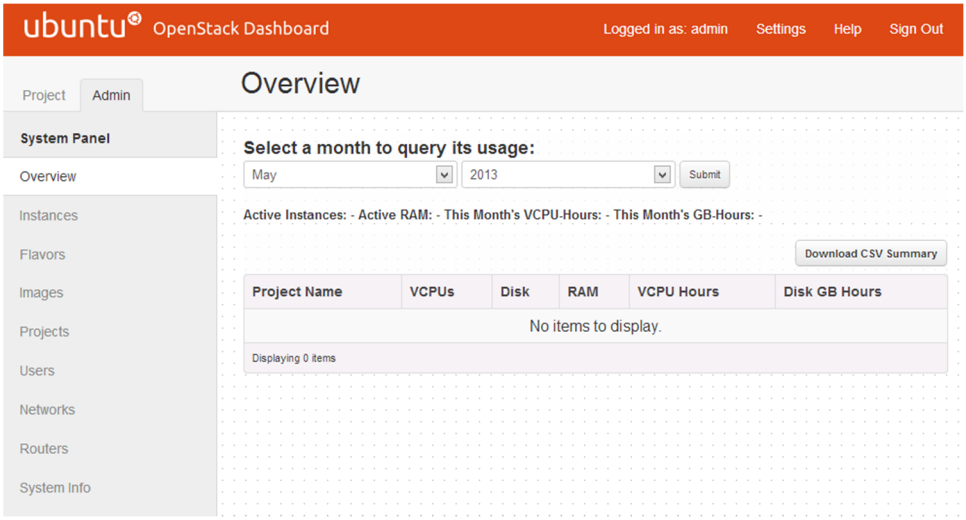
Horizonin asennusskripti asentaa horizonin ja siihen kuuluvat muut osat normaalisti sekä tekee muita pienempiä säätöjä. Asennus käynnistää apachen uudelleen, jotta horizon saadaan toiminaan asennuksen jälkeen. Skripi ajetaan samalla kaavalla kuin edellisetkin:

```
./OpenStack_horizon.sh
```

Kun asennus on valmis, skripti ilmoittaa URL-osoitteen, jolla pääsee dashboardiin käsiksi. Kirjautuminen onnistuu admin-tunnuksella ja aiemmin määritetyllä salasanalla. Jos salasana on unohtunut tähän mennessä, saa sen grep-työkalulla kätevästi näkyviin:

```
env |grep OS_PASSWORD
```

Kirjautumisen jälkeen OpenStack asennus on valmis ja instansseja pystyy sekä luomaan että ajamaan. Kuva 14 esittää näkymän heti sisäänkirjautumisen jälkeen järjestelmänvalvojan tunnuksella.



Kuva 14. OpenStack kirjautumisen jälkeinen järjestelmänvalvojan näkymä.

5 TESTAUS JA KÄYTTÖ

OpenStackia on mahdollista testata monella eri tavalla, kuten esimerkiksi internetistä löytyvillä palveluilla. Näillä palveluilla voi testata OpenStackia loppukäyttäjän osalta. Kuitenkin, jos halutaan hallita järjestelmää muutenkin, on usein ainoa vaihtoehto asentaa OpenStack kokonaan omaan käyttöön.

5.1 Testiympäristöt

Työssä käytettiin kolmea eri testiympäristöä, jotka ovat VMware ESXi 4.1, Oracle VirtualBox 4.2 ja TryStack. Työssä ei ollut tarkoitus asentaa OpenStackia kuin testikäyttöön eli yhtäläistä järjestelmää kuin tuotannossa käytetään ei ollut järkevää alkaa rakentaa. Tuotantojärjestelmää suunnitellessa olisi mennyt jo aivan liian kauan, eikä siitä olisi saatu kuitenkaan testikäyttöön minkäänlaista merkittävää hyötyä. Testiympäristöistä ylivoimaisesti paras työssä tapahtuviin testauksiin oli Oraclen VirtualBox.

5.1.1 VMware ESXi 4.1

HAMKilta käyttöön saatu VMware ESXi 4.1-työtila olisi ollut täysin pätevä ympäristö OpenStackin testailuun, jos sillä olisi pystynyt ajamaan 64-bittisiä käyttöjärjestelmiä. Kuten jo aikaisemmin on mainittu, OpenStack ei ole täysin tuettu 32-bittisillä käyttöjärjestelmillä ja tämän takia syntyikin monta ongelmaa. 32-bittisyydestä johtuvien ongelmien selvittelyyn kuluikin todella aikaa paljon. OpenStack Dashboard työkaluineen toimi kuitenkin asennusten jälkeen täysin moitteetta, mutta instanssien ajaminen ei toiminut ollenkaan. Ongelmia tuotti myös vain yksi DHCP-alue, joka oli varattu käyttöön, kun asennustapa, jota käytettiin, vaati kaksi DHCP-aluetta.

5.1.2 Oracle VirtualBox

Oracle VirtualBox on ilmainen työpöytävirtualisointiin tarkoitettu ohjelma. Ohjelman käyttö on helppoa. Ainoa ongelma OpenStackin testauksen kannalta tulee työpöytäkoneen tehoista, pääasiassa muistin määrästä sekä siitä, tukeeko prosessori virtualisointiteknologiaa. Nämä määräävät, voiko OpenStackia ajaa tällaisella ohjelmalla. Ubuntu 12.04 sekä OpenStack -asennus onnistui ohjeiden mukaan ja skriptien avulla pääasiassa ongelmitta VirtualBoxin virtuaalikoneelle. Normaalissa kotiverkossa ei myöskään tarvitse miettiä IP-määrityksiä niin paljoa. Palomuurit ja reitittimet ovat myös täysin omassa hallinnassa, joka helpottaa käyttöä erittäin paljon.

Testikokoonpano sisälsi 16 GB muistia sekä Intel 3770k prosessorin. Prosessori vaikutti yllätykseksi olevan enemmän kuin riittävä tällaiseen testaukseen. Kun vertaillaan asennusta skriptien avulla VirtualBoxilla pyörivään Ubuntu-palvelinkäyttöjärjestelmään, se kesti noin 10 minuuttia, kun taas ESXi 4.1 palvelimella pyörivälle Ubuntulle, jossa 12 GB muistia ja hieman vanhempi AMD prosessori, kesti asennus noin 60 minuuttia. VirtualBoxista voidaan määrittää virtuaalikoneen ensisijainen verkkokortti siltaavaksi, jolloin oma reititin näkee tämän normaalina tietokoneena ja jakelee tälle IP-osoitteen normaalisti. Tällöin on helppo asettaa esimerkiksi omaan reitittimeen ulkopuolelta tulevat http-pyynnöt reitittymään portin 80 kautta virtuaalikoneeseen. Näin virtuaalikone toimii normaalina palvelimena, johon pääsee käsiksi ulkoverkostakin. Tällaisessa tilanteessa on kuitenkin muistettava, että kuka tahansa pääsee käsiksi palvelimeen avatun portin kautta, joten salasanojen täytyy olla tarpeeksi turvallisia.

5.1.3 TryStack

TryStack on ilmainen palvelu, joka tarjoaa käyttäjille mahdollisuuden testata OpenStackia hiekkalaatikko-tyylisesti, mutta vain loppukäyttäjän osalta. TryStack mahdollistaa yhden instanssin ajamisen ja se onkin tarkoitettu lähinnä tällä kyseisellä instanssilla ajettavien ohjelmien testaukseen. TryStackin kaikki instanssit pyyhitään 24-tunnin jälkeen instanssin käynnistyksestä. Tämä ehkäisee palvelun väärinkäyttöä. TryStack palveluun ei myöskään voi lisätä omia levykuvia. Tällä hetkellä palvelu sisältää kolme valmiiksi asetettua levykuvaa, jotka ovat Ubuntu 12.10, Fedora17

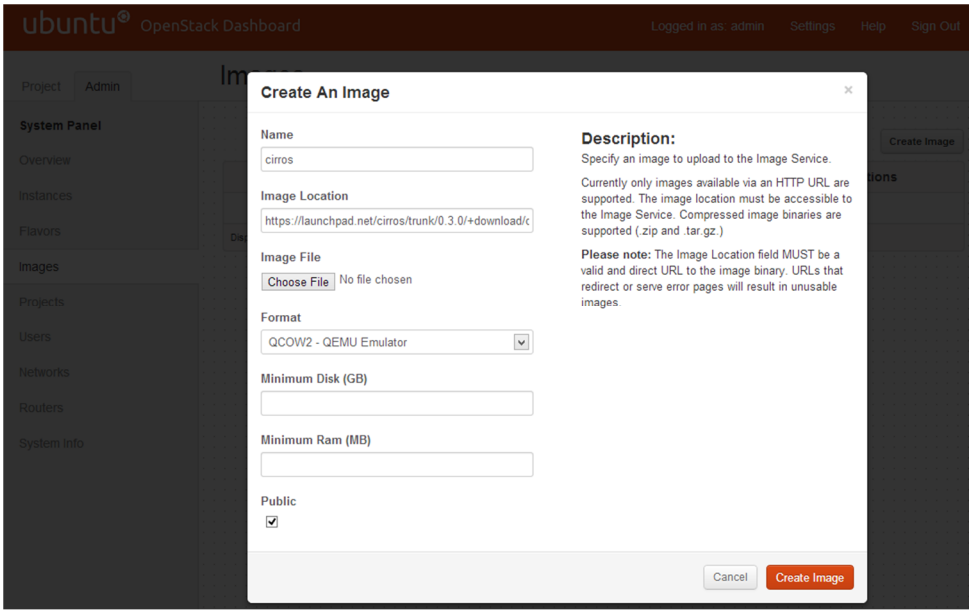
ja Fedora18. Palvelu on kuitenkin täysin ilmainen ja antaa hyvän kuvan OpenStackista loppukäyttäjän osalta. Palveluun voi tutustua osoitteessa http://trystack.org/. Kuva 9 esittää normaalin TryStack käyttäjän näkymän.

5.2 Testauskohteet

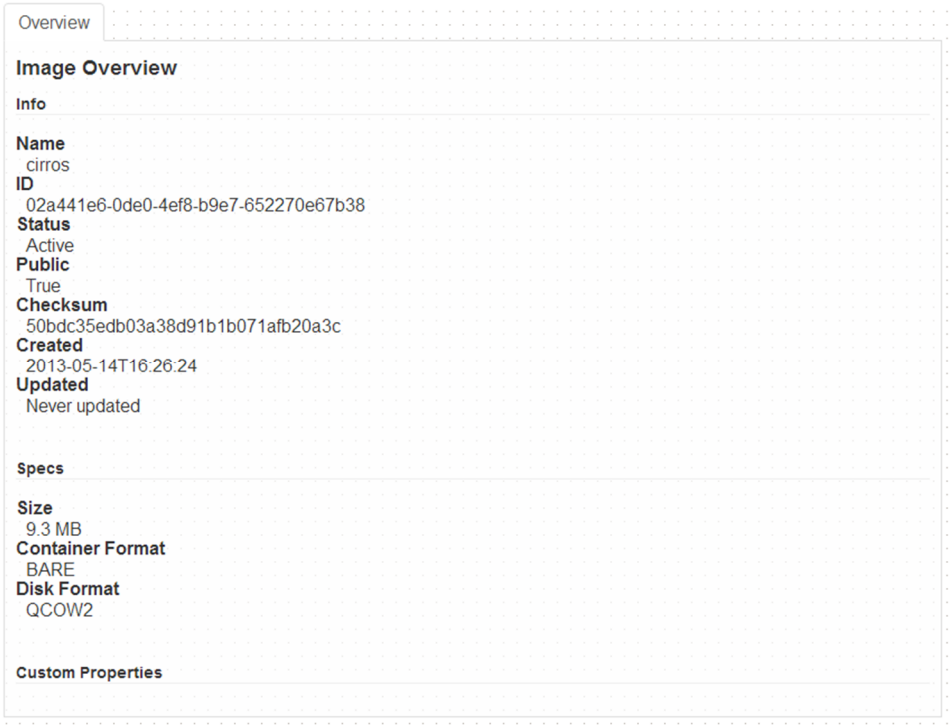
Työssä on valittu testauskohteiksi OpenStackin asennuksen jälkeen oleellisimmat asiat, eli levykuvien lisääminen Glanceen sekä OpenStack instanssin luominen. Nämä testitapaukset ovat suoritettu VirtualBox testiympäristössä.

5.2.1 Levykuvan lisääminen Glanceen

Käytännössä lähes kaikki asennuksen ja konfiguroinnin jälkeen tapahtuu OpenStack-dashboardin avulla. Esimerkiksi levykuvat voidaan lisätä Glanceen suoraan Dashboardin kautta. Heti asennuksen ja kirjautumisen jälkeen voidaan valita System Panel -valikosta Images ja tämän jälkeen yläreunasta Create Image. Seuraavaksi keskelle selainikkunaa aukeaa kuvan 15 mukaisesti uusi valikko, josta määritetään levykuvalle nimi sekä muut asetukset. Levykuvan voi ladata suoraan työaseman tiedostoista tai osoittaa URL-osoite johonkin levykuvaan, jolloin tämä ladataan suoraan internetistä. Kuvassa 15 on syötetty Image Location -kohtaan CirrOS-levykuvan URL-latausosoite. Latauksen jälkeen on mahdollista tarkistaa levykuvan tarkemmat tiedot kuvan 16 mukaisesti. Tiedoista on hyvä tarkastaa esimerkiksi levykuvan Checksum-tiedon vastaavuus alkuperäiseen levykuvaan. Näin voidaan varmistua, että levykuva on täysin sama, joka oli tarkoitus ladata. CirrOS on minimaalinen käyttöjärjestelmä, joka on tarkoitettu ajettavaksi pilvessä. CirrOS vie levytilaa todella vähän, joten se on erittäin hyvä vaihtoehto ajettavaksi juuri tällaisessa testiympäristössä.



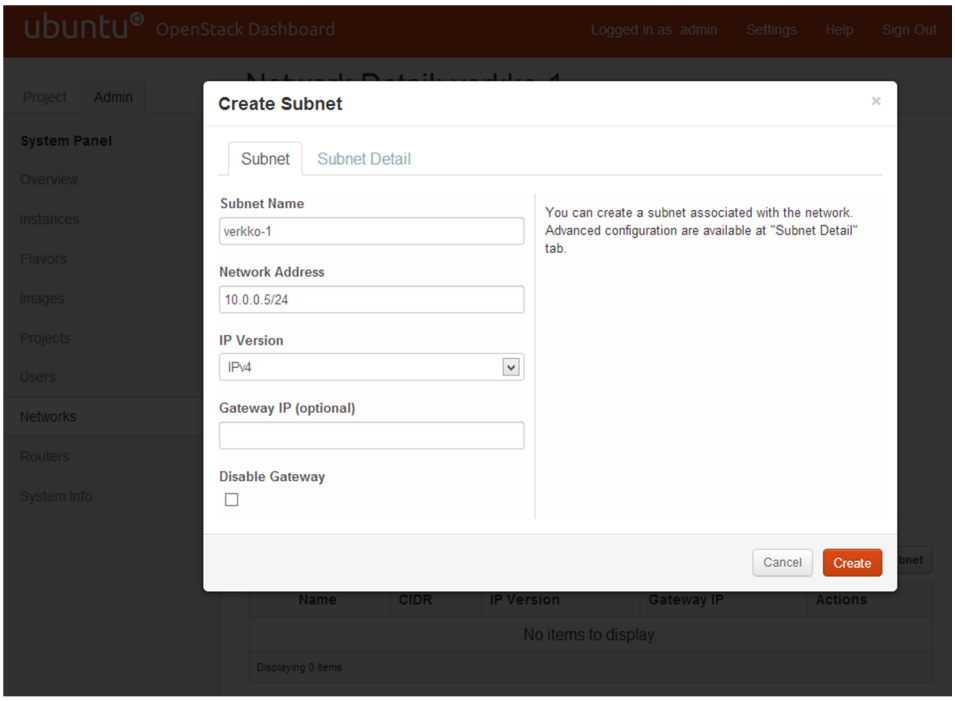
Kuva 15. Uuden levykuvan lisäysasetukset.



Kuva 16. Tarkat tiedot levykuvasta.

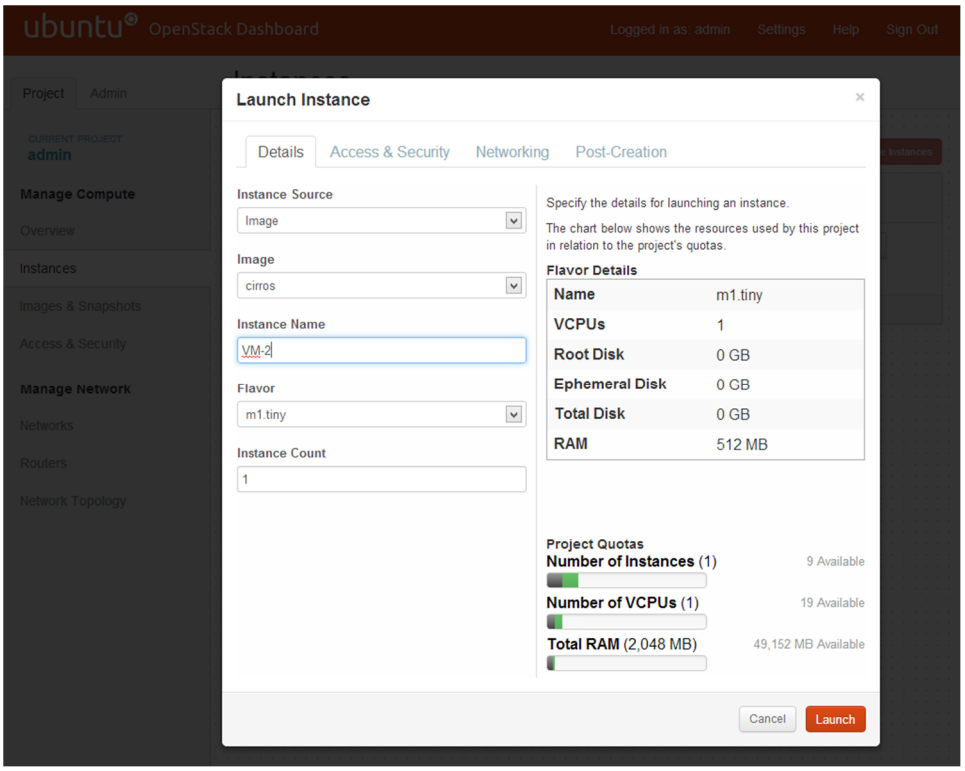
5.2.2 OpenStack-instanssin luominen

Ennen kuin instansseja voidaan luoda, täytyy näille luoda verkko ja verkolle aliverkko. Verkko luodaan System Panel -valikon takaa löytyvästä Networks-kohdasta. Verkon luonti on helppoa, koska ei tarvitse kuin klikata Create Network, jonka jälkeen annetaan verkolle nimi ja osoitetaan se jollekin projektille. Projekti voi olla esimerkiksi admin-käyttäjälle tai jollekin toiselle käyttäjälle aiemmin luotu projekti. Verkkoa luodessa voidaan myös määrittää, onko verkko jaettu- tai ulkoinen verkko. Testauksessa käytämme vain sisäistä verkkoa. Kun verkko on luotu, täytyy sille luoda aliverkko, jonka kautta instansseille jaetaan IP-osoitteet. Kuva 17 esittää, kuinka aliverkko luodaan. Tässä tapauksessa aliverkon osoitteeksi annetaan 10.0.0.5/24. Subnet Detail -välilehdeltä voidaan määrittää tarkemmat asetukset aliverkolle, kuten esimerkiksi DHCP-ominaisuus päälle.



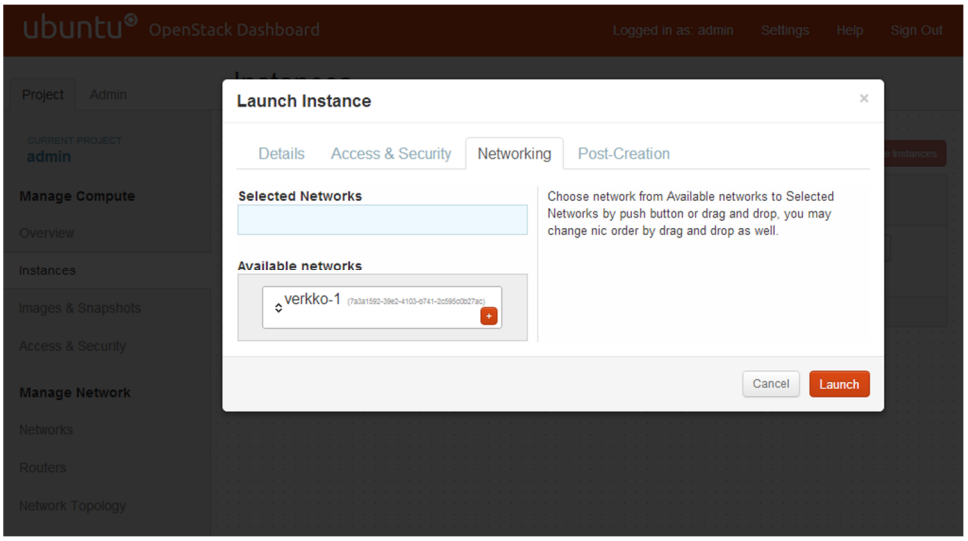
Kuva 17. Aliverkko.

Seuraavaksi voidaan luoda itse instanssi. Instanssi luodaan projektivälilehdeltä valitsemalla Manage Compute -valikosta Instances ja klikkaamalla Launch Instance. Tämän jälkeen kuvan 18 mukainen asetusvalikko avautuu keskelle selainikkunaa. Asetusvalikosta voidaan määrittää, luodaanko instanssi levykuvasta vai snapshotista. Instansseja on mahdollista luoda samalla kertaa niin monta kuin halutaan, mutta tietenkin järjestelmäresurssien puitteissa.



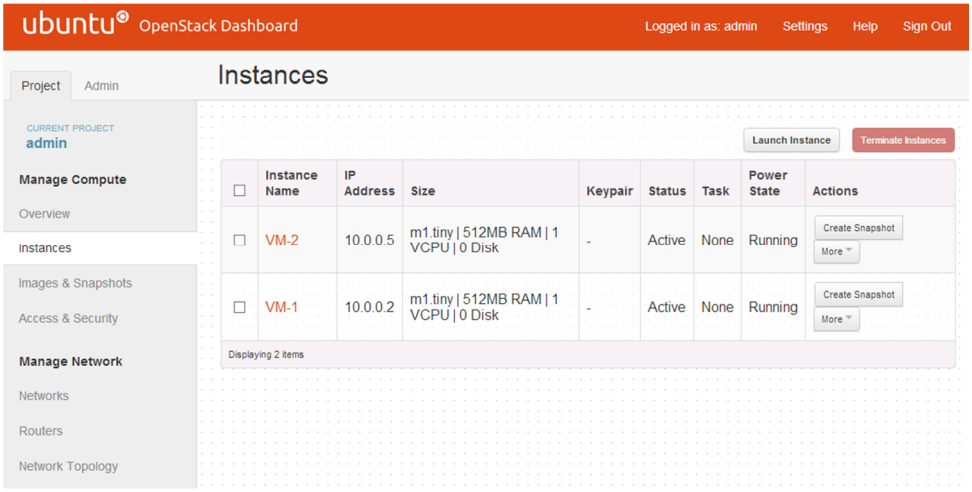
Kuva 18. Launch Instance -päävalikko.

Ennen kuin instanssi voidaan käynnistää, on sille määritettävä aiemmin luotu verkko. Verkko määritetään Networking-välilehdeltä raahaamalla haluttu verkko Selected Networks -alueelle. Kuva 19 esittää verkon valinnan välilehden. Jos verkkoja olisi enemmän kuin yksi, voisi näitä myös valita enemmän instansseille käyttöön. Verkot voitaisiin myös laittaa siihen järjestykseen kuin halutaan.



Kuva 19. Launch Instance -verkkovalikko.

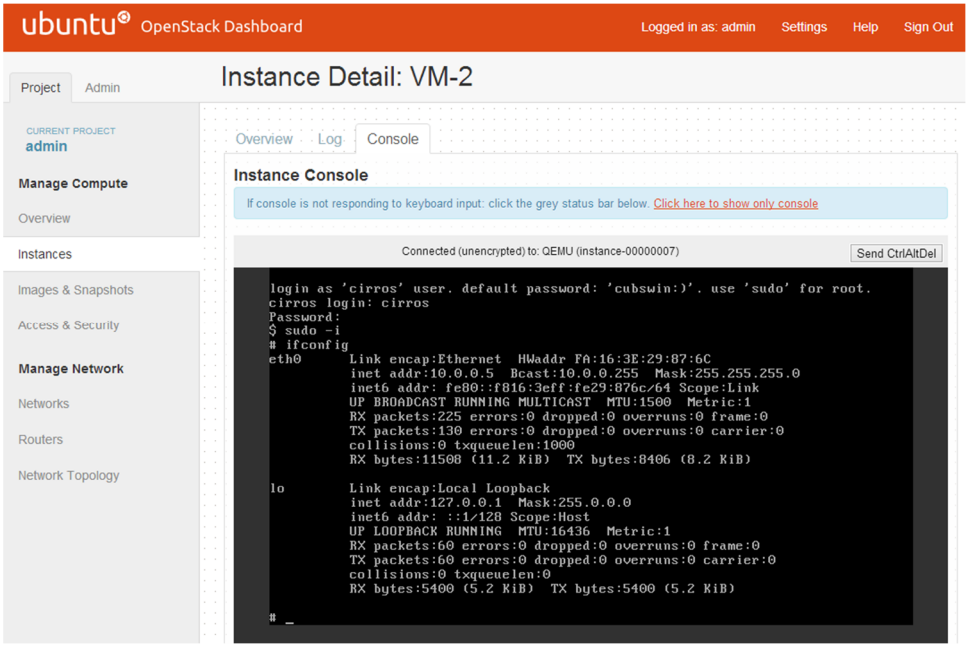
Pakollisten määritysten jälkeen instanssi voidaan käynnistää. CirrOS levykuvaa käytettäessä instanssin käynnistys kestää vain noin yhden minuutin. Kun instanssi on käynnissä, näkyy Instances-valikossa kuvan 20 mukainen yhteenveto käynnissä olevista instansseista.



Kuva 20. Käynnissä olevat instanssit.

Instanssista näkee tarkemmat tiedot klikkaamalla instanssin nimeä. Tämän valikon takaa löytyy myös välilehti nimeltä Console. Console-välilehdeltä aukeaa nimensä mukaisesti konsoli kyseiseen virtuaalikoneeseen, mistä tätä virtuaalikonetta voi hallita. Kuvassa 21 on avattu instanssin VM-2 konsoli ja syötetty tälle käyttäjänimi sekä salasana. CirrOS-käyttöjärjestelmä asentuu itsestään käynnistyksen yhteydessä ja kertoo lopuksi käyttäjäni-

men ja salasanan komentorivillään. Kuvassa 21 on myös ajettu ifconfig-komento, joka näyttää järjestelmän verkkoasetukset.



Kuva 21. Instanssin VM-2 -konsolivälilehti.

5.3 Huomiot

Huomioitavaa on, että vaikka OpenStack on saatu toimimaan testijärjestelmällä, se ei välttämättä toimi samoilla konfiguraatioilla toisessa järjestelmässä. Vaikein osuus OpenStackin asennuksessa on ongelmien selvittely. Tällaisessa järjestelmässä ongelmat voivat johtua erittäin monesta asiasta, ja vaikka jokin ongelma on pieni, se saattaa vaikuttaa koko järjestelmän toimintaan. Linux-käyttöjärjestelmät eivät anna myöskään aina täysin ongelmaan viittaavaa virheilmoitusta, jolloin käyttäjällä on oltava itsellään tietämystä siitä, mitä on tekemässä. Hyvä kokonaiskuva sekä tietämys Linux-järjestelmistä, testiympäristöstä ja verkkoympäristöistä auttavat paljon. Heti kun OpenStack toimii ja ensimmäinen instanssi saadaan käynnistettyä, on helppo perehtyä kaikkiin muihin OpenStackin ominaisuuksiin. Kunnolla asennettua OpenStack-järjestelmää ei tarvitse välttämättä hallita kuin Dashboardin avulla. Dashboard sisältää myös hyvät ohjeet useimpaan sisältämäänsä työkaluun.

6 YHTEENVETO

Pilvipalvelu alkaa olla jo normaali käytäntö ja ratkaisu useimmissa yrityksissä. Pilvipalvelulla saavutetaan monia hyötyjä niin yksityisessä kuin julkisessakin muodossa. Pilvipalvelulla on kuitenkin arveluttavat puolensa, kuten tietoturva. Tietoturva täytyy ottaa erittäin tarkasti huomioon pilvipalveluiden yhteydessä.

Opinnäytetyössä esitelty OpenStack on kattava kokonaisuus eri komponentteja, palveluja ja tekniikoita. OpenStackin avulla voidaan luoda täy-

sin omanlainen pilvialusta. OpenStack on tarkoitettu juuri oman pilven luomiseen, eli tarkoituksena ei ole vain asentaa ohjelmia oletuksena vaan myös kehittää näitä eteenpäin. OpenStackia voidaan testata yhdellä koneella, mutta OpenStackin tuotantoon perustuva asennus vaatii jo vähintään 2 konetta ja enemmän suunnittelua kokonaisuuden osalta.

Opinnäytetyössä saavutettiin tavoitteet ja OpenStack saatiin asennettua testiympäristöön täysin toimivaksi kokonaisuudeksi. OpenStackia testattiin tärkeimpien osa-alueiden kohdalta. Työn avulla voisi jo suunnitella tuotantoympäristöönkin soveltuvan OpenStackin asennuksen, jos käytössä olevat järjestelmäresurssit vain sallivat tämän.

LÄHTEET

Ames, J. 2012. Types of Cloud Computing: Private, Public and Hybrid Clouds. Viitattu 20.3.2013. http://blog.appcore.com/blog/bid/167543/Types-of-Cloud-Computing-Private-Public-and-Hybrid-Clouds

Butler, B. 2012. Gartner's IaaS Magic Quadrant: a who's who of cloud market. Viitattu 8.5.2013. http://www.networkworld.com/news/2012/110712-gartner-magic-quadrant-264058.html?page=1

Campbell, K. 2012. Installing OpenStack on Ubuntu 12.04 LTS in 10 Minutes. Viitattu 10.5.2013. http://www.stackgeek.com/guides/gettingstarted.html

Cinder. 2013. Viitattu, 6.5.2013. https://wiki.OpenStack.org/wiki/Cinder

Cloud Computing Service Models. 2013. Viitattu 4.2.2013. http://imuhandis.com/cloud-computing-service-models/

Compute and Image System Requirements. 2013. Viitattu 8.5.2013. http://docs.OpenStack.org/trunk/OpenStack-compute/admin/content/compute-system-requirements.html

Conceptual Architecture. 2013. Viitattu, 7.5.2013. http://docs.OpenStack.org/trunk/OpenStack-compute/admin/content/conceptual-architecture.html

History of Cloud Computing. n.d. Viitattu 1.4.2013. http://www.eci.com/cloudforum/cloud-computing-history.html

Horizon: The OpenStack Dashboard Project. 2013. Viitattu, 5.5.2013. http://docs.OpenStack.org/developer/horizon/

Loeffler, B. 2011. Cloud Computing: What is Infrastructure as a Service. Viitattu 1.3.2013. http://technet.microsoft.com/en-us/magazine/hh509051.aspx

Logical Architecture. 2013. Viitattu 7.5.2013. http://docs.OpenStack.org/trunk/OpenStack-compute/admin/content/logical-architecture.html

Maynard, J. 2013. Cloud Storage – the First 50 Years. Viitattu 1.4.2013. http://www.mosaicarchive.com/2013/01/03/cloud-storage-the-first-50-years/

Mohamed, A. n.d. A history of cloud computing. Viitattu 1.4.2013. http://www.computerweekly.com/feature/A-history-of-cloud-computing

Msekni, B. 2013. OpenStack Folsom Install guide. Viitattu 9.5.2013. http://www.stackgeek.com/blog/bilelmsekni/guides/OpenStack-folsom-install-guide

OpenStack Community Welcome Guide. 2013. Viitattu 1.5.2013. http://www.OpenStack.org/assets/welcome-guide/ OpenStackWelcomeGuide.pdf

OpenStack: The Open Source Cloud Operating System. n.d. Viitattu 1.5.2013.
http://www.OpenStack.org/software/

Pepple, K. 2012. OpenStack Folsom Architecture. Viitattu 3.5.2013. http://ken.pepple.info/OpenStack/2012/09/25/OpenStack-folsom-architecture/

Pepple, K. 2011. OpenStack Nova Architecture. Viitattu. 3.5.2013. http://ken.pepple.info/OpenStack/2011/04/22/OpenStack-nova-architecture/

Pilvilaskenta - Cloud Computing. 2013. Viitattu 4.2.2013.
http://pilvilaskenta.wikispaces.com/Pilvilaskenta+-+Cloud+Computing

Pilvipalvelu. n.d. Viitattu 1.4.2013. http://suomisanakirja.fi/pilvipalvelu

Pilvipalvelumallit. 2010. Viitattu 11.2.2013.
http://fi.laovirtualisointi.wikia.com/wiki/Pilvipalvelumallit

Pilvipalvelut. n.d. Viitattu 4.2.2013.
http://www-05.ibm.com/fi/solutions/cloud/

Releases. n.d. Viitattu 2.5.2013.
https://wiki.OpenStack.org/wiki/Releases

Rouse, M. 2012. Community Cloud. Viitattu 20.3.2013.
http://searchcloudstorage.techtarget.com/definition/community-cloud

Samson, T. 2013. 9 top threats to cloud computing security. Viitattu 28.4.2013.
http://www.infoworld.com/t/cloud-security/9-top-threats-cloud-computing-security-213428?page=0,0

Welcome to Glance's documentation! 2013. Viitattu, 4.5.2013.
http://docs.OpenStack.org/developer/glance/

Welcome to Keystone, the OpenStack Identity Service! 2013. Viitattu, 5.5.2013. http://docs.OpenStack.org/developer/keystone/

Welcome to Nova's developer documentation! 2013. Viitattu, 5.5.2013. http://docs.OpenStack.org/developer/nova/

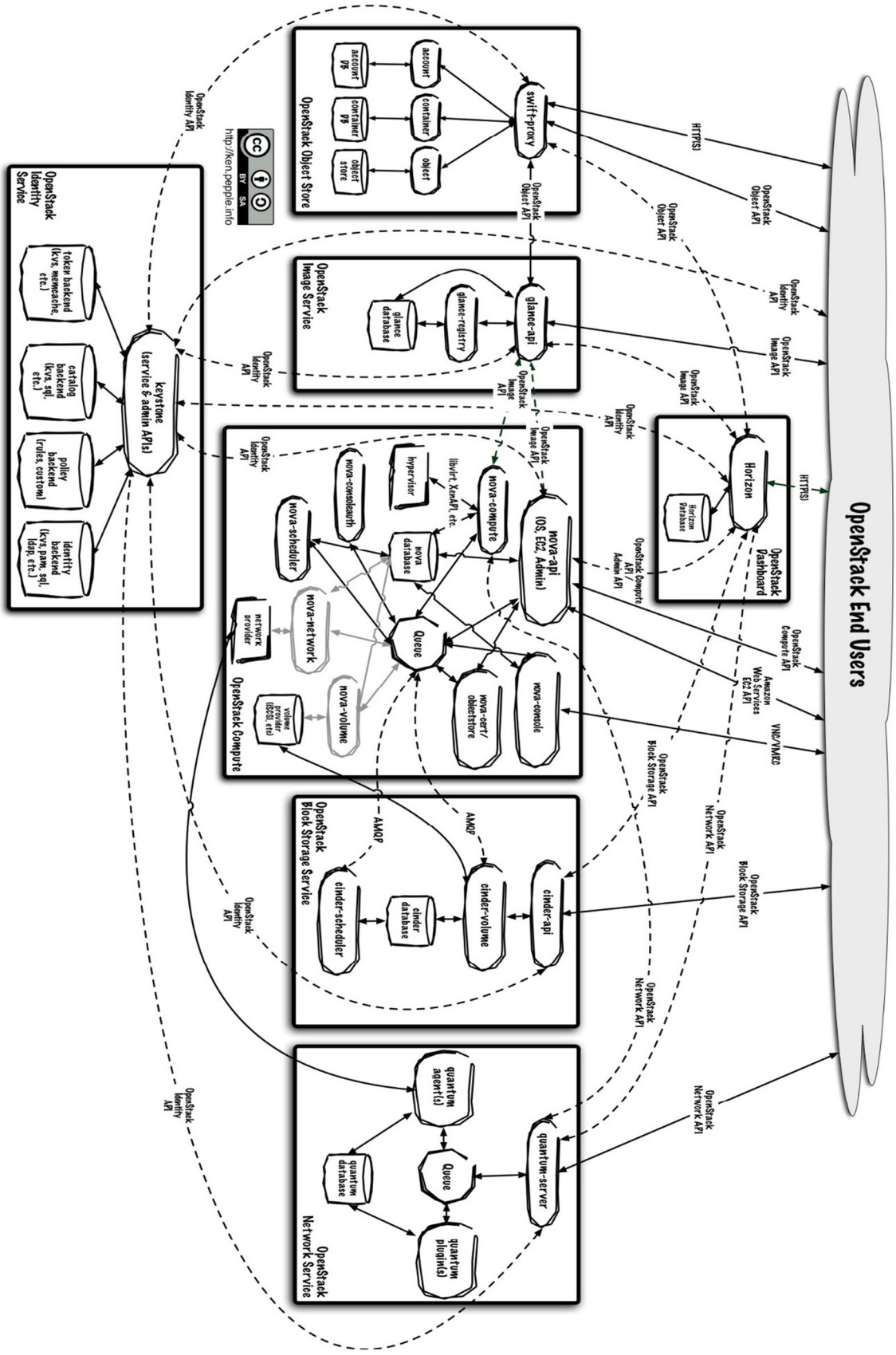
Welcome to Quantum's developer documentation! 2013. Viitattu, 5.5.2013. http://docs.OpenStack.org/developer/quantum/

Welcome to Swift's documentation! 2013. Viitattu, 4.5.2013. http://docs.OpenStack.org/developer/swift/

What is Cloud Computing? 2013. Viitattu 4.2.2013. http://aws.amazon.com/what-is-cloud-computing/

Liite 1

openstack-logical-arch-folsom.jpg



Liite 2

OpenStack_base_1.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# install time server
apt-get install ntp
service ntp restart

# modify timeserver configuration
sed -e "
/^server ntp.ubuntu.com/i server 127.127.1.0
/^server ntp.ubuntu.com/i fudge 127.127.1.0 stratum 10
/^server ntp.ubuntu.com/s/^.*$/server ntp.ubutu.com iburst/;
" -i /etc/ntp.conf

# install tgt
apt-get install tgt
service tgt start

# openiscsi-client
apt-get install open-iscsi open-iscsi-utils

# turn on forwarding
echo 1 > /proc/sys/net/ipv4/ip_forward

echo
"###############################################################################
Go edit your /etc/network/interfaces file to look something like this:

auto eth0
iface eth0 inet static
 address 10.0.1.20
 network 10.0.1.0
 netmask 255.255.255.0
 broadcast 10.0.1.255
 gateway 10.0.1.1
 dns-nameservers 8.8.8.8
auto eth1

After you are done, do a '/etc/init.d/networking restart', then run
'./OpenStack_base_2.sh'

###############################################################################"

exit
```

Liite 3

OpenStack_base_2.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# bridge stuff
apt-get install bridge-utils

# rabbit food
apt-get install rabbitmq-server memcached python-memcache

# kvm
apt-get install kvm libvirt-bin

echo
"##############################################################################################
You'll need a LVM for 'nova-volumes'.  This assumes you have an empty disk spinning
at /dev/sdb:

fdisk /dev/sdb

Create a new partition by hitting 'n' then 'p'.  Use the defaults.  Type 't' then
'8e' to set the
partition to the LVM type.  Hit 'w' to write and exit.

Next, do a:

pvcreate -ff /dev/sdb1
vgcreate nova-volumes /dev/sdb1

NOTE: You should use whatever device handle your system has.  Be careful!

When you are done, run './OpenStack_mysql.sh'

##############################################################################################
"
exit
```

Liite 4

OpenStack_glance.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# get glance
apt-get install glance glance-api glance-client glance-common glance-registry python-
glance

. ./stackrc
password=$SERVICE_PASSWORD

# edit glance api conf files
if [ -f /etc/glance/glance-api-paste.ini.orig ]
then
   echo
"################################################################################
#############"
   echo "Not changing config files.  If you want to edit, they are in /etc/glance/"
   echo
"################################################################################
#############"
else
   # copy before editing
   cp /etc/glance/glance-api-paste.ini /etc/glance/glance-api-paste.ini.orig
   cp /etc/glance/glance-registry-paste.ini /etc/glance/glance-registry-
paste.ini.orig
   cp /etc/glance/glance-registry.conf /etc/glance/glance-registry.conf.orig
   sed -e "
   /^sql_connection =.*$/s/^.*$/sql_connection =
mysql:\/\/glance:$password@127.0.0.1\/glance/
   " -i /etc/glance/glance-registry.conf

   sed -e "
   s,%SERVICE_TENANT_NAME%,admin,g;
   s,%SERVICE_USER%,admin,g;
   s,%SERVICE_PASSWORD%,$password,g;
   " -i /etc/glance/glance-registry-paste.ini

   sed -e "
   s,%SERVICE_TENANT_NAME%,admin,g;
   s,%SERVICE_USER%,admin,g;
   s,%SERVICE_PASSWORD%,$password,g;
   " -i /etc/glance/glance-api-paste.ini

# do not unindent!
```

```
echo "
[paste_deploy]
flavor = keystone
" >> /etc/glance/glance-api.conf

# do not unindent!
echo "
[paste_deploy]
flavor = keystone
" >> /etc/glance/glance-registry.conf

   echo
"########################################################################################
#############"
   echo "Backups of configs for glance are in /etc/glance/"
   echo
"########################################################################################
#############"
fi

# create db tables and restart
glance-manage version_control 0
glance-manage db_sync
sleep 4
service glance-api restart
service glance-registry restart
sleep 4

# add ubuntu image
if [ -f images/ubuntu-12.04-server-cloudimg-amd64-disk1.img ]
then
  glance add name="Ubuntu 12.04 LTS" is_public=true container_format=ovf
disk_format=qcow2 < images/ubuntu-12.04-server-cloudimg-amd64-disk1.img
else
  wget http://stackgeek.s3.amazonaws.com/ubuntu-12.04-server-cloudimg-amd64-disk1.img
  mv ubuntu-12.04-server-cloudimg-amd64-disk1.img images
  glance add name="Ubuntu 12.04 LTS" is_public=true container_format=ovf
disk_format=qcow2 < images/ubuntu-12.04-server-cloudimg-amd64-disk1.img
fi

sleep 4
glance index

echo
"########################################################################################
#############"
echo "You can now run './OpenStack_nova.sh' to set up Nova."
echo
"########################################################################################
#############"
```

Liite 5

OpenStack_horizon.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# get horizon
apt-get install libapache2-mod-wsgi OpenStack-dashboard

# restart apache
service apache2 restart

. ./stackrc
password=$SERVICE_PASSWORD
host_ip=$(/sbin/ifconfig eth0| sed -n 's/.*inet *addr:\([0-9\.]*\).*/\1/p')

echo
"###############################################################################"
echo "The horizon dashboard should be at http://$host_ip/.  Login with ad-
min/$password"
echo
"###############################################################################"
```

Liite 6

OpenStack_keystone.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# get keystone
apt-get install keystone python-keystone python-keystoneclient

read -p "Enter a token for the OpenStack services to auth with keystone: " token
read -p "Enter the password you used for the MySQL users (nova, glance, keystone): "
password
read -p "Enter the email address for service accounts (nova, glance, keystone): "
email

# set up env variables for testing
cat > stackrc <<EOF
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=$password
export OS_AUTH_URL="http://127.0.0.1:5000/v2.0/"
export ADMIN_PASSWORD=$password
export SERVICE_PASSWORD=$password
export SERVICE_TOKEN=$token
export SERVICE_ENDPOINT="http://127.0.0.1:35357/v2.0"
export SERVICE_TENANT_NAME=service
EOF

. ./stackrc

# edit keystone conf file to use templates and mysql
cp /etc/keystone/keystone.conf /etc/keystone/keystone.conf.orig
sed -e "
/^admin_token = ADMIN/s/^.*$/admin_token = $token/
/^driver = keystone.catalog.backends.sql.Catalog/d
/^\[catalog\]/a driver = keystone.catalog.backends.templated.TemplatedCatalog
/^\[catalog\]/a template_file = /etc/keystone/default_catalog.templates
/^connection =.*$/s/^.*$/connection =
mysql:\/\/keystone:$password@127.0.0.1\/keystone/
" -i /etc/keystone/keystone.conf

# create db tables and restart
keystone-manage db_sync
service keystone restart


# sleep a bit before we whack on it
```

```
sleep 5
ADMIN_PASSWORD=$password
SERVICE_PASSWORD=$password
export SERVICE_TOKEN=$token
export SERVICE_ENDPOINT="http://localhost:35357/v2.0"
SERVICE_TENANT_NAME="service"

function get_id () {
    echo `$@ | awk '/ id / { print $4 }'`
}
# Tenants
ADMIN_TENANT=$(get_id keystone tenant-create --name=admin)
SERVICE_TENANT=$(get_id keystone tenant-create --name=$SERVICE_TENANT_NAME)
DEMO_TENANT=$(get_id keystone tenant-create --name=demo)
INVIS_TENANT=$(get_id keystone tenant-create --name=invisible_to_admin)
# Users
ADMIN_USER=$(get_id keystone user-create --name=admin \
                                         --pass="$ADMIN_PASSWORD" \
                                         --email=$email)
DEMO_USER=$(get_id keystone user-create --name=demo \
                                        --pass="$ADMIN_PASSWORD" \
                                        --email=$email)
# Roles
ADMIN_ROLE=$(get_id keystone role-create --name=admin)
KEYSTONEADMIN_ROLE=$(get_id keystone role-create --name=KeystoneAdmin)
KEYSTONESERVICE_ROLE=$(get_id keystone role-create --name=KeystoneServiceAdmin)
# ANOTHER_ROLE demonstrates that an arbitrary role may be created and used
# TODO(sleepsonthefloor): show how this can be used for rbac in the future!
ANOTHER_ROLE=$(get_id keystone role-create --name=anotherrole)

# Add Roles to Users in Tenants
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id
$ADMIN_TENANT
keystone user-role-add --user $ADMIN_USER --role $ADMIN_ROLE --tenant_id $DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $ANOTHER_ROLE --tenant_id
$DEMO_TENANT

# TODO(termie): these two might be dubious
keystone user-role-add --user $ADMIN_USER --role $KEYSTONEADMIN_ROLE --tenant_id
$ADMIN_TENANT
keystone user-role-add --user $ADMIN_USER --role $KEYSTONESERVICE_ROLE --tenant_id
$ADMIN_TENANT

# The Member role is used by Horizon and Swift so we need to keep it:
MEMBER_ROLE=$(get_id keystone role-create --name=Member)
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id $DEMO_TENANT
keystone user-role-add --user $DEMO_USER --role $MEMBER_ROLE --tenant_id
$INVIS_TENANT

# Configure service users/roles
NOVA_USER=$(get_id keystone user-create --name=nova \
                                        --pass="$SERVICE_PASSWORD" \
                                        --tenant_id $SERVICE_TENANT \
```

```
                                        --email=$email)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $NOVA_USER \
                       --role $ADMIN_ROLE

GLANCE_USER=$(get_id keystone user-create --name=glance \
                                          --pass="$SERVICE_PASSWORD" \
                                          --tenant_id $SERVICE_TENANT \
                                          --email=$email)
keystone user-role-add --tenant_id $SERVICE_TENANT \
                       --user $GLANCE_USER \
                       --role $ADMIN_ROLE

if [[ "$ENABLED_SERVICES" =~ "swift" ]]; then
    SWIFT_USER=$(get_id keystone user-create --name=swift \
                                             --pass="$SERVICE_PASSWORD" \
                                             --tenant_id $SERVICE_TENANT \
                                             --email=$email)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $SWIFT_USER \
                           --role $ADMIN_ROLE
    # Nova needs ResellerAdmin role to download images when accessing
    # swift through the s3 api. The admin role in swift allows a user
    # to act as an admin for their tenant, but ResellerAdmin is needed
    # for a user to act as any tenant. The name of this role is also
    # configurable in swift-proxy.conf
    RESELLER_ROLE=$(get_id keystone role-create --name=ResellerAdmin)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $NOVA_USER \
                           --role $RESELLER_ROLE
fi

if [[ "$ENABLED_SERVICES" =~ "quantum" ]]; then
    QUANTUM_USER=$(get_id keystone user-create --name=quantum \
                                               --pass="$SERVICE_PASSWORD" \
                                               --tenant_id $SERVICE_TENANT \
                                               --email=$email)
    keystone user-role-add --tenant_id $SERVICE_TENANT \
                           --user $QUANTUM_USER \
                           --role $ADMIN_ROLE
fi

echo
"##########################################################################"
echo "Time to test keystone.  Do a '. ./stackrc' then a 'keystone user-list'."
echo "Assuming you get a user list back, go on to install glance with
'./OpenStack_glance.sh'."
echo
"##########################################################################"
```

Liite 7

OpenStack_mysql.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

read -p "Enter a password to be used for the OpenStack services to talk to MySQL (us-
ers nova, glance, keystone): " service_pass

echo
"#############################################################################
###"
echo "Setting up MySQL now.  You will be prompted for an admin password by the setup
process."
echo
"#############################################################################
###"
echo ""

# mysql
apt-get install -y mysql-server python-mysqldb

# make mysql listen on 0.0.0.0
sudo sed -i '/^bind-address/s/127.0.0.1/0.0.0.0/g' /etc/mysql/my.cnf

# restart
service mysql restart

# wait for restart
sleep 4

echo
"#############################################################################
###"
echo "Creating OpenStack databases and users.  Use your database password when
prompted."
echo ""
echo "Run './OpenStack_keystone.sh' when the script exits."
echo
"#############################################################################
###"

mysql -u root -p <<EOF
CREATE DATABASE nova;
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'%' IDENTIFIED BY '$service_pass';
GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' IDENTIFIED BY '$service_pass';
CREATE DATABASE glance;
```

```
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'%' IDENTIFIED BY '$service_pass';
GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' IDENTIFIED BY
'$service_pass';
CREATE DATABASE keystone;
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '$service_pass';
GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY
'$service_pass';
EOF

echo "Done creating users and databases!"
```

Liite 8

OpenStack_nova.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

host_ip=$(/sbin/ifconfig eth0| sed -n 's/.*inet *addr:\([0-9\.]*\).*/\1/p')
echo
"################################################################################
#########################"
echo "The IP address for eth0 is probably $host_ip".  Keep in mind you need an eth1
for this to work.
echo
"################################################################################
#########################"
read -p "Enter the primary ethernet interface IP: " host_ip_entry
read -p "Enter the fixed network (eg. 10.0.2.32/27): " fixed_range
read -p "Enter the fixed starting IP (eg. 10.0.2.33): " fixed_start
echo
"################################################################################
###"
echo "The floating range can be a subset of your current network.  Configure your
DHCP server"
echo "to block out the range before you choose it here.  An example would be
10.0.1.224-255"
echo
"################################################################################
###"
read -p "Enter the floating network (eg. 10.0.1.224/27): " floating_range
read -p "Enter the floating netowrk size (eg. 32): " floating_size

# get nova
apt-get install nova-api nova-cert nova-common nova-compute nova-compute-kvm nova-doc
nova-network nova-objectstore nova-scheduler nova-vncproxy nova-volume python-nova
python-novaclient

. ./stackrc
password=$SERVICE_PASSWORD

# hack up the nova paste file
sed -e "
s,%SERVICE_TENANT_NAME%,admin,g;
s,%SERVICE_USER%,admin,g;
s,%SERVICE_PASSWORD%,$password,g;
" -i /etc/nova/api-paste.ini

# write out a new nova file
```

```
echo "
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/var/lock/nova
--allow_admin_api=true
--use_deprecated_auth=false
--auth_strategy=keystone
--scheduler_driver=nova.scheduler.simple.SimpleScheduler
--s3_host=$host_ip_entry
--ec2_host=$host_ip_entry
--rabbit_host=$host_ip_entry
--cc_host=$host_ip_entry
--nova_url=http://$host_ip_entry:8774/v1.1/
--routing_source_ip=$host_ip_entry
--glance_api_servers=$host_ip_entry:9292
--image_service=nova.image.glance.GlanceImageService
--iscsi_ip_prefix=192.168.22
--sql_connection=mysql://nova:$password@127.0.0.1/nova
--ec2_url=http://$host_ip_entry:8773/services/Cloud
--keystone_ec2_url=http://$host_ip_entry:5000/v2.0/ec2tokens
--api_paste_config=/etc/nova/api-paste.ini
--libvirt_type=kvm
--libvirt_use_virtio_for_bridges=true
--start_guests_on_host_boot=true
--resume_guests_state_on_host_boot=true
--vnc_enabled=true
--vncproxy_url=http://$host_ip_entry:6080
--vnc_console_proxy_url=http://$host_ip_entry:6080
# network specific settings
--network_manager=nova.network.manager.FlatDHCPManager
--public_interface=eth0
--flat_interface=eth1
--flat_network_bridge=br100
--fixed_range=$fixed_range
--floating_range=$floating_range
--network_size=$floating_size
--flat_network_dhcp_start=$fixed_start
--flat_injected=False
--force_dhcp_release
--iscsi_helper=tgtadm
--connection_type=libvirt
--root_helper=sudo nova-rootwrap
--verbose
" > /etc/nova/nova.conf

# sync db
nova-manage db sync

# restart nova
./OpenStack_restart_nova.sh
```

```
# no clue why we have to do this when it's in the config?
nova-manage network create private --fixed_range_v4=$fixed_range --num_networks=1 --
bridge=br100 --bridge_interface=eth1 --network_size=$fixed_size
nova-manage floating create --ip_range=$floating_range

# do we need this?
chown -R nova:nova /etc/nova/

echo
"##############################################################################################
###"
echo "'nova list' and a 'nova image-list' to test.  Do './OpenStack_horizon.sh'
next."
echo
"##############################################################################################
###"
```

Liite 9

OpenStack_restart_nova.sh

```
#!/bin/bash

# Make sure only root can run our script
if [ "$(id -u)" != "0" ]; then
   echo "You need to be 'root' dude." 1>&2
   exit 1
fi

# stop and start nova
for a in libvirt-bin nova-network nova-compute nova-api nova-objectstore nova-
scheduler nova-volume nova-vncproxy; do service "$a" stop; done
for a in libvirt-bin nova-network nova-compute nova-api nova-objectstore nova-
scheduler nova-volume nova-vncproxy; do service "$a" start; done
```