BACHELOROPPGAVE:

AutoStack - Automatisk og adaptiv utrulling av Openstack

FORFATTERE:

Geir André Tufte	11HBDRA
Kjetil André Finsrud	11HBIDATA
Anders Godtland Noem	11HBDRA

Dato:

19. Mai 2014

SAMMENDRAG

Tittel:	AutoStack	Dato : 19.05.2014
Deltaker(e)/	Geir André Tufte Kjetil André Finsrud Anders Godtland Noem	
Veileder(e):	Hanno Langweg	
Evt. oppdragsgiver:	Erik Hjelmås	
Stikkord/nøkkelord (3-5 stk)	Automatisering, Openstack, Overvåkning, Nettsky	
Antall sider: 190	Antall vedlegg: 7	Publiseringsavtale inngått: Ja

Kort beskrivelse av master/bacheloroppgaven:

Automatisering er blitt et stadig mer fremtredende tema i informatikkens verden, og er med på å spare verdifull tid, samt forenkle kompliserte prosesser. Samtidig har den raskt voksende bruken av nettskyer til leveranse av infrastruktur til bedrifter, ført til et utgangspunkt for en oppgave som benytter seg av begge disse konseptene.

Denne oppgaven omhandler en automatisert og adaptiv utrulling av skyteknologien Openstack, med tilhørende overvåkning. Rapporten beskriver arbeidsprosessen vedrørende oppsett av server for utrulling, samt de serverne som utgjør skyløsningen. Løsningen realiseres ved å benytte verktøy for automatisk utrulling av operativsystem, konfigurasjonsstyring og skripting. Resultatet er et fungerende multinode-oppsett av Openstack, hvor bruk av løsningen eksemplifiseres ved å rulle ut et fiktivt scenario.

ABSTRACT

Title:	AutoStack	Date : 19.05.2014
Participants/	Geir André Tufte Kjetil André Finsrud Anders Godtland Noem	
Supervisor(s)	Hanno Langweg	
Employer:	Erik Hjelmås	
Keywords (3-5)	Automation, Openstack, Monitoring, Cloud	
Number of pages: 190	Number of appendix: 7	Availability: Open

Short description of the bachelor thesis:

Automation has become an increasingly prominent subject in the world of information technology, and helps to save valuable time and simplify complex processes. Meanwhile, the rapidly growing use of cloud computing for provisioning of infrastructure to businesses, led to a starting point for a task that uses both of these concepts.

This paper discusses an automated and adaptive deployment of the cloud project Openstack, accompanied with monitoring. The report describes the work process associated to the setup of server deployment, and the servers that make the cloud solution. The solution is achieved using tools for automatic operating system deployment, configuration managment and scripting. The result is a functioning multi-node setup of Openstack, where the use of the solution is exemplified by deploying a fictional scenario.

Forord

Hensikten med denne rapporten er å videre utforske skyteknologien Openstack, en teknologi som det er blitt forsket på ved Høgskolen i Gjøvik de siste årene. Tidligere oppgaver har omhandlet implementasjon av skyløsning, og nå er det aktuelt å se på hvordan denne implementasjonen kan automatiseres. Deler av oppgaven er også tenkt å kunne implementeres i skolens nåværende skyløsning, SkyHiGh. Det har vært en stor motivasjon for gruppen å kunne introduseres til skyteknologien Openstack ifm. bacheloroppgaven, samt få god kjennskap til flere verktøy som i dag brukes mye i systemadministrasjon. Det gleder oss også at deler av oppgaven vår vil være relevant ved utvidelser av SkyHiGh.

Denne rapporten omhandler vår arbeidsprosess med å utvikle en automatisert installasjon av et Openstack-multinode-miljø, med overvåkning og en medfølgende automatisk utrulling av et scenario.

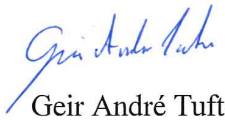
Gruppen ønsker å rette en stor takk til oppdragsgiver Erik Hjelmås for god dialog og hjelp under arbeidet med denne oppgaven, samt til veileder Hanno Langweg for faglig innsikt og konstruktiv kritikk. Videre har IT-tjenesten ved HiG og førsteamanuensis Kyrre Begnum bidratt med å løse problemer innenfor nettverk og Openstack underveis, hjelp som har vært til nytte for prosjektets fremdrift.

I tillegg har gruppen fra familie, venner og medstudenter fått verdifull tilbakemelding på dokumentasjon, samt sosial avkobling og støtte underveis. Dette har vært avgjørende for fullføringen av oppgaven.

Gjøvik 19.05.2014

Kjetil André Finsrud

Anders Godtland Noem



Geir André Tufte

Innhold

1 Introduksjon 1

1.1 Bakgrunn . . . 2

1.2 Oppgavebeskrivelse . . . 3

1.3 Målgrupper . . . 3

1.3.1 Prosjektet . . . 3

1.3.2 Rapporten . . . 3

1.4 Formål . . . 4

1.5 Avgrensning . . . 4

1.6 Rapportens innhold . . . 4

1.7 Studentenes faglige bakgrunn . . . 5

1.8 Roller . . . 5

1.9 Arbeidsrammer . . . 6

2 Kravspesifikasjon 7

2.1 Utforming av kravspesifikasjon . . . 8

2.2 Use Case . . . 8

2.3 Operasjonelle krav . . . 9

3 Design 10

3.1 Utforming av løsningen . . . 11

3.2 Alternative løsninger . . . 11

4 Teoretisk grunnlag 12

4.1 Skytjenester . . . 13

4.2 Openstack . . . 13

4.2.1 Nova . . . 14
4.2.2 Glance . . . 15
4.2.3 Neutron . . . 15
4.2.4 Keystone . . . 16
4.2.5 RabbitMQ . . . 16
4.3 Hvorfor Openstack . . . 17
4.4 Alternativer til Openstack . . . 17
4.5 Cobbler . . . 18
4.6 Puppet . . . 18
4.6.1 Deployment . . . 19
4.6.2 Configuration language and resource abstraction . . . 19
4.6.3 Transaction . . . 20
4.7 Overvåkning . . . 21
4.7.1 Munin . . . 21
4.7.2 Nagios . . . 22
5 Implementering 23
5.1 Oppsett av Newman . . . 24
5.1.1 Cobbler . . . 25
5.1.2 Nettverk . . . 29
5.1.3 Puppet . . . 30
5.1.4 Git . . . 32
5.2 Oppsett av installasjonservere . . . 33
5.2.1 Puppet . . . 33
5.3 Manuelt oppsett av OpenStack . . . 34
5.3.1 Fysisk oppsett . . . 34
5.3.2 Kontrollernode . . . 35
5.3.3 Nettverksnode . . . 44
5.3.4 Computenode . . . 47

5.3.5 Test av installasjon . . . 49
5.4 Utrulling av manuell Openstack installasjon . . . 52
5.4.1 Klargjøring . . . 52
5.4.2 Utføring . . . 52
5.5 Automatisk installasjon av Openstack . . . 54
5.5.1 Openstack-komponenter . . . 55
5.5.2 Annen programvare . . . 56
5.5.3 Overvåkning . . . 57
5.6 Gjennomføring/Utviklingsprosess . . . 58
5.6.1 Informasjonstilegning . . . 58
5.6.2 Valg av utviklingsmodell . . . 59
5.6.3 Kommentar til Gantt-skjema . . . 60
5.6.4 Kommentar til brukerveiledning . . . 60
5.7 Utrulling av scenario . . . 61
5.7.1 Jungle-scenario . . . 61
6 Testing 63
6.1 Metodikk . . . 64
6.2 Cobbler . . . 64
6.3 Puppet . . . 65
6.4 Manuell installasjon av Openstack . . . 66
6.5 Automatisk installasjon av Openstack . . . 67
6.6 Overvåkning . . . 68
6.7 Maskinvare . . . 68
7 Diskusjon 69
7.1 Tilgjengelig dokumentasjon . . . 70
7.2 Komplikasjoner ved automatisering . . . 70
7.3 Bruk av åpen kildekode . . . 71
7.4 Inspirasjon og kilder . . . 71

7.5 Videreutvikling og robusthet 71
7.6 Interessenter 72
7.7 Utrulling av skyen 72
7.8 Utrulling av overvåkning 73
8 Konklusjon 74
8.1 Resultat 75
8.2 Læringsutbytte 76
8.3 Forbedringer 76
8.4 Videreutvikling 77
8.5 Egenevaluering 77
Bibliografi 79
A Terminologi 83
B Kode 87
C Brukerveiledning 114
D Prosjektplan 119
E E-post 140
F Logg og status 153
G Prosjektavtale 188

Figurer

4.1 Openstack-databaser 14

4.2 Puppet layers 19

5.1 Cobbler check utskrift 26

5.2 Openstack-oppsett. 34

5.3 Utskrift av *glance image-list*. 40

5.4 Horizon, hypervisors 44

5.5 Virtualiseringssjekk 47

5.6 Statussjekk for Nova-tjenester 49

5.7 Openstack-roller 50

5.8 Munin oversikt. 57

5.9 Openstack utrullingsprosess 58

5.10 Nettverkstopologi Jungle scenario 62

Kapittel 1

Introduksjon

1.1 Bakgrunn

Virtuelle arbeidsmiljøer har i nyere tid fått en fremtredende rolle i hverdagslivet til studenter, IT-selskaper og andre arbeidstakere i et bredt spekter av yrkesretninger. Ved Høgskolen i Gjøvik er det et behov for å kunne tilby dataressurser til studenter i forbindelse med prosjekter, bachelor/masteroppgaver og liknende i virtuelle miljøer. I dag benyttes høgskolens egen skyløsning, SkyHiGh, for å dekke disse behovene. I denne oppgaven søkes det å oppnå forståelse rundt hvordan en slik løsning kan settes opp for automatisk konfigurasjon. En automatisert utrulling av en Openstack-skyløsning vil kunne utvides til å tilby spesialtilpassede scenarier, som består av virtuelle maskiner som rulles ut i Openstack-miljøet. Det er tidligere blitt gjennomført bacheloroppgaver innenfor ulike områder av Openstack ved Høgskolen i Gjøvik, noe som har gitt et grunnlag for videre forskning.

Aktører hos forskningssenteret ved høgskolen, Center for Cyber- and Information Security(CCIS)[1], har belyst at det pr. dags dato ikke eksisterer noen enkel metode for å bedrive IT-teknisk eksperimentering på. Det er derfor ønskelig at tilpassede scenarier kan opprettes og konfigureres etter de spesifikke behovene som eksisterer. I dag eksisterer en løsning[2] hvor tjenestetilbyderen oppretter virtuelle labmiljøer for å kunne utføre trening og eksperimenter.

Denne bacheloroppgaven har dermed en relevans med bakgrunn i et behov for å kunne tilby en helautomatisert utrulling av Openstack, med opplegg for at allerede eksisterende labmiljøer/scenarier skal kunne implementeres. Som tilbydere av en slik løsning er det ønskelig at det på én dag skal kunne rulles ut et miljø med ønskelige scenarier, og gjøres tilgjengelig for bruk. Som en del av dette vil det også kunne tilbys resirkulering av utdaterte servere og annen maskinvare som ikke lenger er i drift, men som fortsatt kan tjene en hensikt for Openstack.

Ved prosjektets oppstart eksisterer det liknende løsninger som tar for seg de samme verktøyene i bruk for utrulling av Openstack som denne oppgaven. Eksempelvis har Cisco[3] etablert seg som en bidragsyter med sin egen versjon av Havana-utgivelsen, og virker å følge utviklingen til Openstack. Det er altså åpenbart at det er interesse for temaene denne oppgaven tar for seg, noe som gir et sammenlikningsgrunnlag.

1.2 Oppgavebeskrivelse

Oppgaven omhandler hovedsaklig hvor automatisert en utrulling av Openstack-skyløsningen kan gjøres. Dette innebærer distribusjon av operativsystem og alle de nødvendige konfigurasjonene som sørger for at komponentene Openstack består av vil etableres, konfigureres og tilpasses automatisk. I tillegg er det ønskelig å oppnå en detaljert overvåkning av disse nodene som en del av automatiseringen, i form av verktøyene Munin og Nagios. Høgskolen i Gjøvik får gjennom sine nettverk opptil flere ganger årlig donert servere fra regionale bedrifter. I denne sammenheng, vil oppgaven også innebære det å sette inn disse serverne i et Openstack-miljø, samt sette opp automatisk konfigurasjon og implementering av disse som en ressurs i skyløsningen.

1.3 Målgrupper

1.3.1 Prosjektet

Dette prosjektets målgruppe er først og fremst oppdragsgiver, Erik Hjelmås, hos IMT-avdelingen ved Høgskolen i Gjøvik, med bakgrunn i at det er ønskelig å skape en lettvint måte å rulle ut løsningen på i ulike undervisnings,- og forskningscenarier. Videre er det vist interesse fra eksterne aktører, noe som tyder på at denne løsningen også kan være interessant for andre å benytte seg av.

1.3.2 Rapporten

Rapportens målgruppe i denne sammenhengen er i all hovedsak oppdragsgiver Erik Hjelmås, samt interessenter ved IMT -avdelingen på Høgskolen, men det er også ønskelig å nå ut til en så stor brukergruppe som mulig. Dersom en våger å tenke utenfor denne boksen, vil det være spennende å kunne bidra til forbedringer av de eksisterende løsningene innenfor Openstack og Puppet, slik at disse blir tatt i bruk av de større brukergruppene på disse områdene. Det bør bemerkes at det forventes en teknologisk forståelse innenfor høyere utdanning i informatikk for å kunne bearbeide denne rapporten.

1.4 Formål

Denne oppgaven har fokus på automatisk detektering og utrulling av Openstack, til bruk i brukertilpassede scenarier. Som en del av dette regnes minimeringen av det manuelle arbeidet som kreves for konfigurering av ny maskinvare i SkyHiGh, automatisk implementering av ulike typer maskinvare, og adaptiv ressurstildeling. Dette skal føre med seg flere og bedre tjenesteleveranser til Høgskolen i Gjøvik sine ansatte og studenter, samt øke HiGs bidrag til Open-source-miljøet.

For gruppens medlemmer tilbyr dette prosjektet en rekke læringsgoder, deriblant oppnåelse av kunnskap innenfor skyteknologier, virtualisering, automatisering, overvåkning, konfigurasjonsstyring osv. I tillegg vil det være et stort læringsutbytte i forhold til det å samarbeide i et større prosjektarbeid.

1.5 Avgrensning

Det vil fokuseres på å automatisere installasjonen av de Openstack-komponentene beskrevet i oppgavebeskrivelsen, hvor det eneste kravet er at serverne som skal ta del i løsningen møter systemkravene for en Openstack installasjon. Skyløsningen som rulles ut skal kun legge til rette for at spesifikke scenarier bestående av prekonfigurerte VM'er kan implementeres i skyløsningen. Det vil ikke bli lagt vekt på noen form for lagring, backup, gjenoppretting eller ytelsestesting da dette faller utenfor formålet med denne oppgaven, og fordi enkelte av disse emnene vil bli berørt av andre bacheloroppgaver. Videre søkes det å implementere overvåking av systemet, og der skal det benyttes programvaren Munin og Nagios.

1.6 Rapportens innhold

I denne rapporten søkes det å fokusere mest mulig på automatiseringen av Openstack, samtidig som andre nødvendige områder vil bli dekket, for at leseren skal kunne oppnå forståelse og se sammenhengen i innholdet. Strukturen på hoveddelen av rapporten, implementeringsdelen, er bygd opp på en slik måte at leseren først blir kjent med de verktøyene som legger til grunn for at automatiseringen av Openstack kan skje. Deretter introduseres Openstack-komponentene i form av en manuell installasjon, etterfulgt av arbeidet med automatisering av installasjonen.

1.7 Studentenes faglige bakgrunn

Prosjektgruppens medlemmer kommer fra to ulike studieretninger innenfor informatikk ved Høgskolen i Gjøvik. Geir André Tufte og Anders Godtland Noem studerer drift av nettverk og datasystemer, mens Kjetil André Finsrud studerer ingeniørfag data. Felles for gruppen er at alle har tilegnet seg kunnskaper innenfor programmering, databaser, systemutvikling osv. Dette har gitt en felles plattform som er relevant for løsning av denne prosjektoppgaven. Videre har driftsstudentene opparbeidet seg kunnskaper innenfor emner som blant annet virtualisering, konfigurasjonsstyring og nettverk. Samtidig har Kjetil ingeniørfaglige emner i bakgrunn, i tillegg til en fagskolebakgrunn med liknende fagområder som de driftsstudiet legger vekt på.

I dette prosjektet er det nødvendig med tilegning av ny kunnskap innenfor flere områder. Openstack består av ulike komponenter med mange små detaljer, og vil naturlig nok være den delen av prosjektet det settes av mest tid til å tilegne seg kunnskap om. Utover dette distribueres de andre verktøyene til dels mellom gruppemedlemmene, slik at kunnskapstilegningen i forhold til tidsforbruk gjøres mest mulig effektivt. De andre verktøyene omfatter hovedsakelig Cobbler[1] og Puppet[2]. Her benyttes Cobbler til å foreta en automatisk installasjon av operativsystem, og Puppet til orkestrering. Fokus vil altså hovedsakelig være på Openstack, men det er samtidig viktig at alle gruppemedlemmer har forståelse for de ulike verktøyene i bruk, og sammenhengen dem imellom.

1.8 Roller

Oppdragsgiver for denne bacheloroppgaven er førsteamanuensis Erik Hjelmås ved Høgskolen i Gjøvik. Erik innhar høy kompetanse innenfor temaene i oppgaven, og han vil derfor kunne være til god hjelp ved eventuelle tekniske spørsmål som gruppen måtte ha. Som oppdragsgiver vil han i tillegg ha fokus på å styre oppgaven i retning av det ønskelige sluttproduktet. Veileder for gruppen er førsteamanuensis Hanno Langweg. Han har tidligere vært veileder for flere bachelorgrupper, deriblant en bacheloroppgave innen Openstack, og innehar derfor god kompetanse på prosjektarbeid og det teoretiske arbeidet med dette. Gruppen vil dra stor nytte av å kunne rette spørsmål om selve rapporten og arbeidet med den til han. Som veileder vil han ha fokus på hvordan gruppen best mulig kan nå det produktet oppdragsgiver ønsker.

[1] http://www.cobblerd.org
[2] http://puppetlabs.com

Prosjektleder og kontaktperson for gruppen er Geir André Tufte. Han vil også ha ansvar for hjemmesiden. Andre ansvarsforhold og roller vil bli fordelt utover i prosjektperioden etter hvert som behovet oppstår.

1.9 Arbeidsrammer

Arbeidet med dette prosjektet vil i hovedsak foregå på et tilegnet grupperom på høgskolen. Rommet inneholder et serverskap med de nødvendige fysiske komponentene som skal benyttes under oppsett, utvikling, konfigurering og testing av den automatiserte utrullingen, fra start til slutt. I forbindelse med informasjonsinnhentingen, vil gruppen ha tilgang til dette via Internett, oppdragsgiver og tidligere bacheloroppgaver. Dermed er det i utgangspunktet ikke behov for å oppsøke eksterne aktører.

En typisk arbeidsdag vil variere fordi gruppens medlemmer har ulike valgfag på ulike tidspunkt, og en ellers varierende timeplan. Det vil likevel avtales faste tidspunkter for møter og samarbeid fra uke til uke. Gruppen kan også arbeide hjemme når dette er nødvendig, men søker å begrense dette til det minimale.

Andre rammer prosjektgruppen må forholde seg til gjelder de som går på HiG sitt regelverk for gjennomføring av bacheloroppgaver. Dette innebærer krav til dokumentasjon, møter, overholding av frister, oppfølging av kontrakter osv. I tillegg gjelder regelverket gruppen har satt internt gjennom hele perioden.

Kapittel 2

Kravspesifikasjon

2.1 Utforming av kravspesifikasjon

Kravene som stilles i forbindelse med denne løsningen baserer seg i hovedsak på å etablere en fungerende automatisk utrulling av Openstack med adaptiv ressurstildeling, noe som går på bekostning av fokus på områder som ytelse og robusthet. Videre vil kravene rundt funksjonalitet for løsningen kunne variere, da det vil være ulike ønsker fra kundens side for oppsettet. Et eksempel på dette kan være at noen ønsker et testscenario som kun opererer internt i bedriftens nettverk, mens det i andre tilfeller vil være ønskelig å nå ut på Internett fra de virtuelle maskinene.

2.2 Use Case

I Use Case-modellen, som finnes i Prosjektplanen(s.119), er det vurdert ulike scenarier, hovedsakelig i forbindelse med forespørsler om utrulling, samt administrering av Openstack. Når det gjelder forespørsler fra kunden om dette, er prosessen rimelig enkel. Et poeng verdt å merke seg her er at dersom gruppen ikke har mulighet til å etterkomme kundens ønsker, det være seg grunnet tidsmangel, problemer rundt lokasjon, eller utfordringer i forbindelse med tilgjengelige ressurser, vil det fortsatt være mulig for kunden å benytte denne løsningen selv. Siden dette prosjektet opererer i Open-source, vil det også gjøres tilgjengelig for hvem som helst til å ta i bruk. Utfordringen for kunden vil da være at de selv må kompensere for kunnskapen gruppen besitter, eventuelt tilegne seg denne for å kunne implementere løsningen, samt utnytte den til dets fulle potensiale.

Når det gjelder administrasjon av Openstack, er dette et steg i prosessen hvor gruppen som leverandør har levert og rullet ut løsningen, og fra dette punktet er det kundens administratorer som står for videre drift. For at de skal kunne administrere Openstack, vil de ha behov for all relevant dokumentasjon. Dette innebærer eventuelle brukernavn og passord for tilgang til web-grensesnittet, systemressurser, oppsett av serverroller, nettverkskonfigurasjon osv. Den nevnte dokumentasjonen krever faglig innsikt fra brukerne av systemet, og dette må tas i betraktning fra kundens side før løsningen implementeres. I etterkant av at AutoStack har levert produktet, ligger forholdene til rette for kunden. Det er gode muligheter for videreutvikling av denne løsningen, at som ved siden av tilpasning til eget system, kan være meget aktuelt å se nærmere på etter utrullingen er fullført og eventuelt tatt i bruk.

2.3 Operasjonelle krav

Det stilles en rekke krav til den automatiske utrullingen av Openstack, og viktigheten rundt disse vil variere en del. Grunnet omfanget og vanskelighetsgraden som følger med oppgaven, vil områder som ytelse og robusthet bli fokusert på i mindre grad. Dette vil si at en fungerende løsning prioriteres fremfor en robust løsning.

Når det gjelder bruk av løsningen, skal den kunne benyttes i brukertilpassede scenarier, med nødvendig dokumentasjon tilgjengelig. I tillegg til den fullstendige løsningen som leveres og gjøres tilgjengelig på Internett, genereres det en brukerveiledning for dette systemet. Slik skal kunden lett kunne benytte seg av det uten å måtte bruke veldig mye tid og ressurser på informasjonstilegning. En del av tanken er at kunden skal spare mye tid på at Autostack implementerer denne løsningen for dem. Dermed sees det på som en del av pakken, at kunden lettvint skal få en grunnleggende forståelse for hvordan systemet fungerer.

Krav som stilles for ytelse, både i forbindelse med utrulling og bruk, har gruppen avsatt lite tid og ressurser, da dette ikke har vært fokusområde for oppgaven. De kravene som stilles vil gjerne være knyttet til utstyret det kjøres på, og ikke løsningen i seg selv. Dette er dermed et område kunden vil ha ansvaret for, og gjerne har tilrettelagt for i forkant av implementering ut ifra egne ønsker.

Sikkerhet i Openstack kan fordeles på flere områder. Mye er avhengig av hvordan løsningen er tenkt i bruk. Dersom tilgang ut mot Internett ikke er ønskelig, vil en naturlig sikkerhet allerede eksistere da systemet blir som en sandkasse å regne. Det er samtidig meget situasjonsavhengig, med tanke på at det i mange av tilfellene er aktuelt å implementere Openstack for å bedrive nettopp sikkerhetstesting, og funksjonaliteten i systemet vil tilpasses deretter.

Videre følger Keystone som en grunnpilar innenfor Openstack, en autentisering,- og identifiseringstjeneste. I tillegg til Openstack er det i henhold til oppgavebeskrivelsen implementert overvåkning i form av verktøyet Munin. Dette vil tilby overvåking av de serverne som utgjør løsningen, i form av trendgrafer som måler hovedsakelig ytelse. Sikring av løsningen utover dette faller innenfor kundens ansvarsområde.

Kapittel 3

Design

3.1 Utforming av løsningen

For dette prosjektet ble det fra oppdragsgiver definert et ønsket design/format for hvordan løsningen skulle settes opp, og videre funksjon fysisk og logisk sett. Som vist i Prosjektplan(s.119), er Newman hovedserveren i løsningen og er, sammen med en switch, den komponenten som benyttes for å implementere løsningen. Newman er nøkkelen i denne løsningen, og er den komponenten som står for utrullingen av Openstack med tilhørende rollefordeling mot kundens eksisterende noder. Newman har samtidig rollen som ruter under utrullingen og ruter trafikk mellom skyløsningen og skolens nett.

I designet vist i figur 5.2(s.34) er løsningen satt opp med de serverne gruppen har fått tildelt for dette prosjektet. I et scenario for utrulling hos kunde vil dette tilpasses det utstyret kunden har tilgjengelig eller ønsker å benytte, med forbehold om at kravene om en kontroller-, nettverks- og compute-node oppfylles, slik at løsningen har de nødvendige komponentene.

3.2 Alternative løsninger

I utgangspunktet er Openstack tilbøyelig for å kunne implementeres på flere ulike måter. Det vil for eksempel ikke være nødvendig med en egen nettverksnode, og dette er noe som vil passe bedre i mange situasjoner. I dette prosjektet er det ikke spesifisert konkret i oppgavebeskrivelsen, men det ble, etter diskusjon med oppdragsgiver, klart at det er ønskelig med de nodene et Openstack-multinode-miljø består av. Det vil si én kontrollernode, én nettverksnode og et varierende antall computenoder ut i fra gitt scenario. Dette vil gi en ryddig og oversiktlig løsning, som også er godt tilpasset de guidene som benyttes under prosjektet. Det er altså mulig å implementere alternativer løsninger, men dette er ikke blitt vurdert.

Kapittel 4

Teoretisk grunnlag

4.1 Skytjenester

Nettskytjenester omhandler leveranse av tjenester fra en ekstern lokasjon som er ukjent for brukeren og tjenesteleverandøren. En samling av fysiske servere tildeles ressurser, kapasitet, og eventuelt programvare, for å kunne tilby disse tjenestene ut mot en brukergruppe, som da kun trenger å forholde seg til de virtuelle maskinene. En av de store fordelene med bruk av skytjenester er at de er designet slik at kunden betaler for det faktiske forbruket, og dersom kundens behov endrer seg, skal dette på en lettvint måte kunne skaleres i form av f.eks. økt båndbredde, tilgjengelig minne osv.

Skytjenestene kan deles opp i følgende kategorier:

- Infrastructure as a Service (IaaS) - Tilgang til virtuelle maskiner i skyen.
- Platform as a Service (PaaS) - Det leveres et oppsett med en infrastruktur bestående av maskinvare, operativsystemer, web-server osv.
- Software as a Service (SaaS) - Kunden betaler for programvaren i skyen og kan se bort fra infrastrukturen da denne skal forbli uendret.

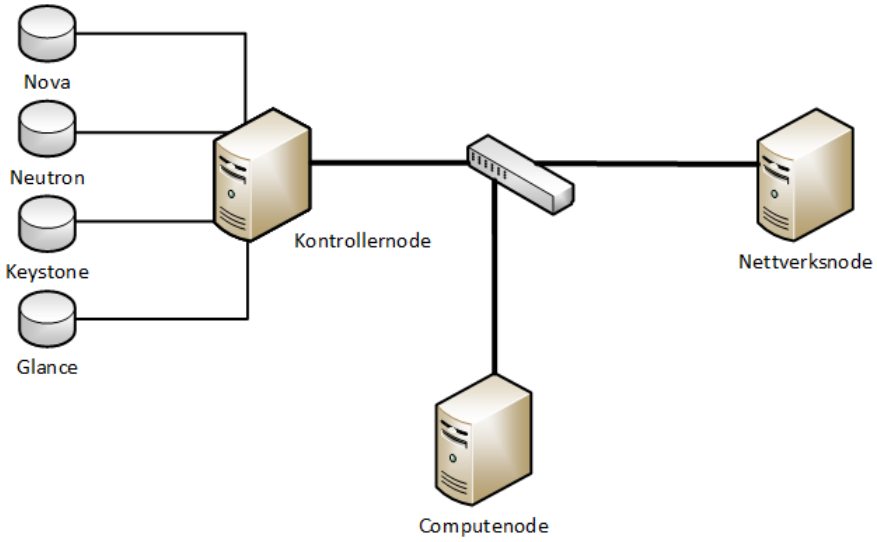
I dette prosjektet er det IaaS som er den aktuelle metoden for levering av skytjenesten.

4.2 Openstack

Openstack er et Open-source-skyløsningsprosjekt som har til hensikt å gi organisasjoner muligheten til å opprette skyløsninger som kan kjøres på vanlig maskinvare, og tilby Infrastructure as a Service(IaaS). Dette er tydeligvis noe som vekker interesse hos IT-industrien, da Openstack regnes som det raskest voksende Open-source-miljøet i verden, og har fått en tilhengergruppe med kjente navn som CERN og PayPal. Openstack-prosjektet opererer i dag med utgivelsessykluser på 6 måneder med fastsatte milepæler.

I dette prosjektet er det aktuelt å benytte komponentene Nova, Glance, Keystone og Neutron, i tillegg til RabbitMQ for å håndtere kommunikasjon. Sentrale Openstack-komponenter, som Swift og Cinder, vil ikke tas hensyn til, da områdene lagring og volumhåndtering faller utenfor oppgavebeskrivelsen. For å tilby et webgrensesnitt i Openstack benyttes tjenesten *Horizon*. I dette prosjektet regnes ikke det som et behov i utgangspunktet, men det kan bli aktuelt ved eventuell testing for en kunde-/brukergruppe.

Ved oppsett av disse Openstack-komponentene, vil det opprettes en MySQL-database med databaser for Nova, Neutron, Keystone og Glance.



Figur 4.1: Openstack-databaser

4.2.1 Nova

Nova er kjent som kjernemodulen i Openstack. Den tar seg av oppretting og administrering av de virtuelle maskinene, og er skrevet i programmeringsspråket Python. Nova innfører en rekke begreper for å beskrive de ulike elementene i bruk:

- Instance : En virtuell instans (datamaskin).
- Flavor : Et sett med maskinvarekonfigurasjoner for den valgte instansen (CPU-kjerner, RAM, diskplass et cetera).
- Security Group: Et sett med brannmurregler for den valgte instansen. Er i utgangspunktet satt til *implicit deny all*, som vil si at det ikke tillater trafikk fra andre noder.
- Keypair: Et SSH-nøkkelpar bestående av en privat og en offentlig nøkkel som benyttes i forbindelse med pålogging. Ved oppstart benyttes den ene nøkkelen, slik at brukeren kan benytte sin egen og få tilgang.
- Nova-api: Gir et grensesnitt ut mot brukerne. Dette kjøres vanligvis på kontrollernoden.
- Nova-network: Håndterer nettverksforespørsler i et køsystem.

- Nova-scheduler: Sørger for at forespørsler i køsystemet videresendes til riktig node i nettverket.
- Nova-compute: Ansvarlig for oppretting og terminering av virtuelle instanser.
- Nova-volume: Styrer den fysiske lagringsplassen til instansene.

4.2.2 Glance

Glance er modulen som har jobben med å administrere de virtuelle imagene i Openstack. Disse kan lagres på ulike måter, det være seg i enkle filsystemer eller objektorienterte lagringssystemer. Den består av tjenestene glance-api og glance-registry. Førstnevnte tar i mot API-kommandoer som omhandler images, mens glance-registry håndterer metadata.

Glance baserer seg på å følge en rekke retningslinjer:

- Komponentbasert arkitektur.
- Høy tilgjengelighet og skalerbarhet.
- Isolerte prosesser.
- Gode gjennopprettingsfunksjoner.
- Åpne standarder.

4.2.3 Neutron

Neutron er et prosjekt innenfor Openstack designet for å tilby *Networking as a Service*(NaaS). Dens oppgave er å levere dette mellom ulike grensesnitt på tvers av Openstack-tjenester. Tidligere var nettverksdelen en delkomponent av Nova, men med dette fulgte en del mangler på teknologi i forhold til design, samtidig som det å iverksette mer avanserte nettverkstjenester var umulig.

Neutron skal tilby en plugin-funksjon som gir *Tenants* egenskapen å opprette private nettverk, samt kontrollere IP-addressering. Som et resultat av API-utvidelsene i Neutron, gis organisasjoner muligheten til å kontrollere sentrale temaer som sikkerhet og kompatibilitetsutfordringer. Videre er utfordringene med lettvint å kunne rulle ut avanserte nettverkstjenester som brannmur og VPN, løst ved hjelp av Neutron.

4.2.4 Keystone

Keystone er en identifiserings- og autentiseringstjeneste i Openstack, og benyttes av alle de andre modulene i spill. Det opprettes en database for å lagre informasjon om brukere, prosjekter, privilegier osv, og det er dette som bygger opp Keystone slik den er kjent i dag. En bruker kan autentiseres ved hjelp av brukernavn og passord eller brukernavn og API-Nøkkel.

Konsepter :

- User: En bruker eller en representasjon av en bruker.
- Token: Et sett med privilegier (tekststreng) som håndterer ulike brukeres tilganger/rettigheter.
- Tenant: Et prosjekt. Altså en sammenknytning av alle instanser, IP-adresser, sikkerhetsregler et cetera.
- Service: Alle tjenester registreres i Keystone. Hver tjeneste har en bruker knyttet opp mot seg.
- Endpoint: URL til en API. Altså, informasjon om hvordan en tjeneste kan aksesseres.
- Role: En bruker er knyttet til et prosjekt med en bestemt rolle. Her har hver rolle et sett med privilegier, knyttet opp mot en token.

4.2.5 RabbitMQ

RabbitMQ(Messaging Queue) er en kommunikasjonstjeneste som benyttes i Openstack. Den er ikke en del av Openstack-løsningen, men brukes for at de ulike Openstack-komponentene skal kunne kommunisere med hverandre. Den benytter seg av asynkrone kall, slik at man slipper tjenester som henger mens de venter på hverandre. RabbitMQ er skrevet i programmeringsspråket *Erlang*.

4.3 Hvorfor Openstack

De nevnte alternativer bygger på mange styrker som gjør dem til attraktive alternativer i en skyløsning. Så hva gjør Openstack til det beste alternativet?

Det mange vil peke på som den største fordelen ved Openstack, er at Linux-verdenen har gitt sin støtte til dette prosjektet ved at det nå kan benyttes av alle de store Linux-distribusjonene[5] (Ubuntu, RedHat et cetera). I tillegg blir det støttet av flere av tungvekterne i bransjen, der i blant Cisco, Dell og IBM. Det er altså sendt et tydelig signal om at Open-source-miljøene benytter sine ressurser på Openstack. Dette vil mange påstå kommer som en følge av at Openstack har lyttet til sine brukermiljøer, og bygd en plattform som ikke er låst fast til en spesiell leverandør. En av de viktigste begrunnelsene for valget av Openstack bygger på det faktum at HiG allerede benytter seg av den på flere områder, eksempelvis i undervisningssammenheng, i faget Database- og applikasjonsdrift[6]

4.4 Alternativer til Openstack

I forbindelse med at skolen benytter seg av Openstack til eksisterende løsninger, og ønsker å videreutvikle seg på dette området, ble det naturlig nok definert i oppgavebeskrivelsen at Openstack skulle benyttes. Det kan allikevel være interessant å se på hvilke andre alternativer som finnes innenfor skyløsninger[7].

- **Eucalyptus** er en Open-source-tjeneste for private skyløsninger. Den har stort fokus på skalerbarhet/tilpasningsdyktighet, og har sterk støtte fra Amazon Web Services.
- **Cloudstack** er underlagt Apache Software Foundation, og er en skytjeneste for oppretting, utrulling og vedlikehold av infrastruktur(IaaS). En av fordelene ved å bruke Cloudstack er støtten til de mest kjente hypervisorene, deriblant KVM og VMware. Samtidig som Cloudstack er en konkurrent til Openstack, samarbeider disse på flere områder, og er på denne måten et kjerneeksempel på sunn konkurranse.
- **Joyent** tilbyr programvare for IaaS og PaaS, og er designet for større virksomheter. Det markedsføres som et produkt som leverer lik ytelse med halvparten av servermengden, med dobbel sikkerhet og til en tredjedel av kostnadene Openstack fører med seg. Dette regnes som et forsøk på å fremheve Openstacks mangler på produksjonsområdet.

4.5 Cobbler

Cobbler er en Linux-installasjonsserver som tilbyr automatisert utrulling av operativsystemer. Applikasjonen legger vekt på bruk av lettvinte kommandoer for å få utført de ønskede konfigurasjonene, bruk av *PXE*, virtuelle-, eller mediabaserte installasjoner, samt reinstallasjoner. Cobbler er skrevet i Python, og er en forholdsvis liten applikasjon som streber etter å være lettvint å benytte i både store og små sammenhenger. Fokus på effektive behandlinger av både enkle og avanserte oppgaver, gjør den til en tidsbesparende løsning ved de aller fleste bruksområdene.

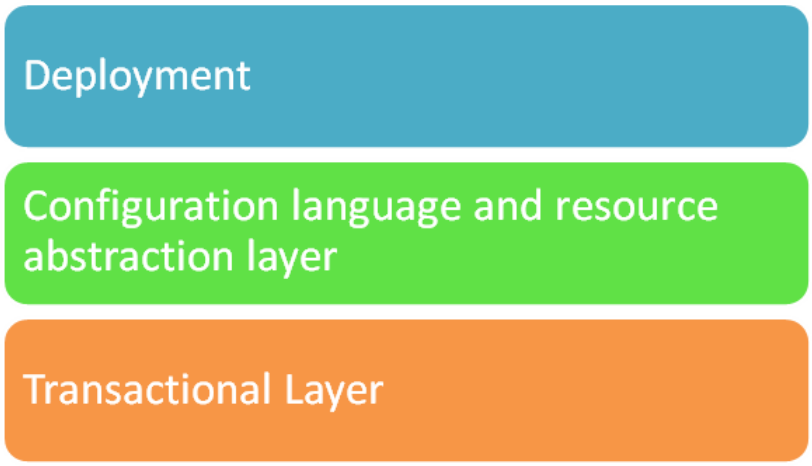
En egenskap gruppen ønsker å benytte seg av i dette prosjektet, er at Cobbler har støtte for å fungere sammen med konfigurasjonsstyringsverktøy, i dette tilfellet Puppet. Gruppen benytter seg av Cobblers konfigurasjonsfiler for å utføre de nødvendige innstillingene før applikasjonen tas i bruk. Alternativer til Cobbler vurderes ikke, da det er spesifisert fra oppdragsgiver at det er det ønskelige verktøyet for denne løsningen, selv om dette ikke er nevnt i oppgavebeskrivelsen.

4.6 Puppet

Puppet er et verktøy for konfigurasjonsstyring. Det er basert på programmeringsspråket Ruby, og kan benyttes i forbindelse med konfigurasjoner på ulike operativsystemplattformer. Eksempler på slike konfigurasjoner er installasjon av pakker, oppgraderinger og vedlikehold av disse. Den optimale løsningen for å rulle ut en konfigurasjon, er ved bruk av Puppet-moduler. En modul inneholder prosedyrer for hvordan en spesifikk applikasjon eller en konfigurasjon skal installeres og vedlikeholdes. Dette gir grunnlag for en infrastruktur hvor samme konfigurasjon kan rulles ut flere ganger uten å endre resultatet etter første utrulling. Moduler er også skrevet på en slik måte at de vil ta høyde for plattformforskjeller i infrastrukturen under utrulling. I denne oppgaven vil det benyttes Puppet-moduler så langt det går for å dra nytte av disse egenskapene. Det skrives da såkalte manifester som inneholder de ønskene som er blitt satt for systemet.

Innenfor konfigurasjonsstyring eksisterer det flere alternativer til Puppet. Et eksempel på dette er CFEngine[14], som flere av gruppens medlemmer har erfaring med fra tidligere. Likevel er gruppen låst til valget av Puppet i henhold til oppgavebeskrivelsen, delvis i forbindelse med at det er et verktøy som benyttes av samarbeidspartnere til oppdragsgiver.

Puppet består av 3 lag: Deployment, configuration language and resource abstraction layer, samt transactional layer.



Figur 4.2: Puppet layers

4.6.1 Deployment

Det er flere måter å sette opp Puppet på, men det er vanligvis, og i dette tilfellet, aktuelt med en klient/server-modell. Her omtales serveren som *Puppet-master*, mens programvaren, som kjøres på klienten, er en *agent*. Vertsmaskinen som agenten kjøres på kalles en *node*. Det fungerer videre slik at de ønskede nodekonfigurasjonene defineres på mesteren, som lar agenten koble seg til via SSL, slik at den enkelt kan hente ned sine respektive konfigurasjoner. Dersom innstillingene allerede er tredd i kraft, vil heller ingen resulterende hendelser forekomme.

Et alternativ til denne metoden er å kjøre en såkalt *standalone* løsning. Dette vil si at Puppet kjøres kun på en lokal maskin og utfører alle konfigurasjoner her. Det er da ingen Puppet-master å forholde seg til, og Puppet kjøres for å utføre konfigurasjonene.

Normal praksis innebærer å kjøre Puppet som en daemon, og sette den til periodevis å sjekke opp mot serveren om dens konfigurasjoner er oppdaterte, eller om det finnes nye konfigurasjoner tilgjengelige for utrulling.

4.6.2 Configuration language and resource abstraction

Puppet er et deklarativt programmeringsspråk, som vil si at det programmeres med tanke på hva det ønskelige resultatet er, og ikke på hvordan man skal komme seg dit. Man kan da for eksempel si at pakken *Vim* skal være installert, og videre forvente at Puppet håndterer utførelsen av dette, da dette ikke er interessant å vite detaljene rundt for den som installerer programmet.

I Puppet defineres såkalte ressurser for å håndtere de ønskede konfigurasjonene. En ressurs består av en *type*(f.eks. en pakke, tjeneste et cetera), *tittel*(navnet på ressursen) samt attributter som benyttes for å definere den ønskede tilstanden til ressursene.
For å se mer i dybden på eksempelet med Vim, kan man tenke seg de ulike faktorene som tas i betraktning ved skripting av installasjonsprosessen. Da må det opprettes tilkobling til noden, utføres en sjekk på om programmet er installert, eventuelt legge inn kommandoer for å installere programmet, samt rapportere resultatene av installasjonen. Hvis man her velger å benytte seg av Puppet, trengs det kun å sette en ressurs som vist under:

```
package{'vim':
    ensure => present,
}
```

Skulle det her være ønskelig å avinstallere Vim, er det kun nødvendig å skifte attributtverdien fra *present* til *absent*. Dermed vil agenten avinstallere Vim neste gang den henter konfigurasjoner fra mesteren.

I forbindelse med at en agent henter ut konfigurasjoner fra mesteren, er Puppet avhengig av å vite hvilket OS/plattform agenten kjører på. Her benyttes et bibliotek som kalles *Facter*, som returnerer informasjonen til mesteren, slik at Puppet benytter riktig installasjonsmetode. Hver ressurstype har en rekke leverandører, og ved hjelp av Facter forstår Puppet for eksempel at hvis Ubuntu er operativsystemet i bruk, skal *apt* benyttes for å installere pakker.

En fordel med Puppet er at den er idempotent, som vil si at ved bruk av de samme parameterene i konfigurasjonene, vil alltid de samme resultatene returneres. Dette betyr at den ikke endrer miljøet med mindre det er nødvendig.

4.6.3 Transaction

Dette er den delen av Puppet som tar hånd om prosesseringen av konfigurasjonen. Dette innebærer sending av konfigurasjonen til agenten, dens bruk på agenten, samt tilbakerapportering til mesteren.

Det første Puppet gjør er å analysere konfigurasjonen som er blitt produsert, og forsøker å finne ut hvordan denne skal benyttes på agenten. Det genereres så en ressursoversikt som viser de ulike relasjonene. På denne måten kan Puppet finne ut en fornuftig rekkefølge å implementere konfigurasjonen på. Videre sendes ressursene som en katalog til hver av de aktuelle agentene. På bakgrunn av dette, sendes en rapport tilbake til Puppet master. Det er verdt å merke seg at Puppet ikke er fullt transaksjonell, da man ikke kan gå tilbake på overføringer som er blitt utført. Allikevel har man muligheten til å kjøre endringene i en såkalt *no-operation* mode, slik at man får testet konfigurasjonen i form av tekst til skjerm, uten at dette påvirker produksjonsmiljøet.

4.7 Overvåkning

I forbindelse med den automatiserte implementeringen av Openstack, er det ønskelig fra oppdragsgiver at det følger med oppsatt overvåkning i form av verktøyene Munin og Nagios. Disse er kjent for å komplimentere hverandre godt, da Munins registrering av hendelser, samt Nagios sine *real-time* funksjonaliteter dekker de områdene rundt overvåkning som er nødvendige i et driftsscenario.

Overvåkningssystemer har i nyere tid fått en meget viktig rolle, da det stadig dukker opp nye trusler som kan gjøre enorm skade på en infrastruktur dersom disse ikke blir tatt hånd om umiddelbart. Samtidig tas det i bruk stadig mer komplekse og avanserte systemer som stiller store krav til kontinuerlig oversikt og kontroll.

Tanken gruppen går ut i fra for denne oppgaven, er å først implementere Munin med dens aktuelle noder, da den sees på som den minst komplekse av de to verktøyene. Videre vil Nagios legges til dersom dette er mulig. Det vil da tas utgangspunkt i eksisterende Puppet-moduler for disse verktøyene, men det vil likevel være aktuelt å gjøre egne tilpasninger for at disse skal fungere.

4.7.1 Munin

Munin er et Open-source nettverksbasert monitoreringsverktøy. Det tilbyr monitorering og medfølgende tjenester for alarmering. Monitoreringen gjelder både for fysisk utstyr, som switcher, servere, aksesspunkter etc, samt ulike applikasjoner og tjenester som kjører i infrastrukturen. Munin alarmerer både når noe har gått galt, og når noe er tilbake til fungerende status. Verktøyet er skrevet i programmeringsspråket Perl, og benytter seg av RRDtool (round robin database), som er et verktøy for å generere grafer. Aksessering av Munin foregår gjennom et web-grensesnitt.

Munin er bygget på en mester/node-arkitektur, hvor mesteren med jevne mellomrom kontakter nodene for å innhente informasjon fra dem. Denne informasjonen, det være seg disk- eller minneforbruk, tjenester som er utilgjengelige eller liknende, blir så lagret av mesteren i RRD-filer. Videre oppdateres grafene jevnlig, og kan benyttes til å se trender i infrastrukturen og forutse eventuelle svakheter.

En av de store fordelene med Munin er antallet plugins, som gjør at man ved en helt vanlig installasjon får grafer tilgjengelig innenfor mange forskjellige områder for den som bedriver overvåkningen. Et annet kjennetegn ved Munin er at det er et såkalt trend-overvåkningsverktøy, som vil si at det fokuseres på registrering av hendelser over tid, samt årsaker til ytelsessvikt og liknende.

4.7.2 Nagios

Nagios er et Open-source-verktøy for monitorering av real-time applikasjoner, nettverk og infrastruktur. Som Munin, tilbyr den monitorering og alarmeringstjenester for nettverksutstyr som servere, switcher, printere, applikasjoner og tjenester. Dens egenskaper til å gi beskjed når noe ikke fungerer som det skal, samt når feilen er blitt rettet opp, gjør Nagios til et mektig og nødvendig verktøy for en systemadministrator.

På hjemmesiden til Nagios[1] proklameres det for at Nagios har egenskapene til å hjelpe en organisasjon til å identifisere, samt håndtere problemer som oppstår før de påvirker kritiske sektorer. Et eksempel på dette kan være at det er i ferd med å oppstå plassmangel i lagringssektoren, og ansvarlige har dermed muligheten til å håndtere dette problemet før systemet overbelastes og tjenester blir utilgjengelige. Dette blir sett på som en av de viktigste grunnene til at Nagios har etablert seg som en tungvekter i IT-sektoren.

[1] http://www.nagios.org

Kapittel 5

Implementering

Serverne, som benyttes i løsningen, har blitt oppkalt etter karakterer og navn på skuespillere fra TV-programmet Seinfeld. Den ene av de fem serverne gruppen har til rådighet har fått rollen som hovedserver og har navnet Newman. Med hovedserver menes det serveren som vil utføre utrullingen av Openstack-miljøet på de resterende fire serverne, som har navnene Jerry, Bubbleboy, Bania og Bookman. Jerry er den kraftigeste serveren blant disse fem og har derfor fått rollen som kontrollernode i Openstack-miljøet. I et multinodeoppsett av Openstack krever en dedikert nettverksnode tre nettverkskort, noe kun Bania har. Derfor har Bania fått rollen som nettverksnode. Bubbleboy og Bookman har fått rollen som computenoder. Alle serverne som skal settes opp som Openstack-noder vil omtales som installasjonservere.

Dette kapitellet beskriver oppsettet av hovedserveren, Newman, og videre det arbeidet som ble gjort for å komme frem til sluttproduktet. Kapitellet består hovedsaklig av følgende seksjoner:

1. Oppsett av Newman med verktøy for utrulling av operativsystem og konfigurasjon på installasjonservere.
2. Manuell installasjon av Openstack.
3. Utrulling av Openstack basert på den manuelle installasjonen.
4. Automatisk installasjon av Openstack.
5. Utrulling av scenario.

I siste punkt som omhandler utrulling av scenario, ble det hentet inspirasjon fra den tidligere bacheloroppgaven, *Jungle*[8], skrevet ved Høgskolen i Gjøvik, våren 2011. Jungle omhandler utvikling av et rammeverk for penetrasjonstesting og læring. Rammeverket består av virtuelle maskiner, konfigurert for å gi brukere av systemet mulighet til å finne potensielle sårbarheter ved konfigurasjonen i en kontrollert omgivelse. Eksempler fra Jungle benyttes i denne oppgaven for å vise hvordan den utrullede skyløsningen kan huse et slik scenario.

5.1 Oppsett av Newman

Newman, som står for utrullingen av løsningen, har verktøyene Cobbler og Puppet installert. Cobbler er som nevnt et verktøy for å automatisere installasjon av operativsystem på en eller flere maskiner. Cobbler kan konfigureres til å serve flere operativsystemer slik at en maskin som kobler seg til Cobbler-serveren, kan velge i en liste med operativsystemer. Automatisering av selve installasjonen gjøres ved å benytte en svarfil som har listet alle svarene som blir stilt under en installasjon, noe som vanligvis ville krevd manuell interaksjon.

Puppet er et verktøy for å administrere og konfigurere maskiner i en infrastruktur. Dette gjøres ved å sette opp Newman som Puppet-master og la de resterende serverne (Puppet-agenter) koble seg til masteren for å hente ned sin konfigurasjon. Hvilken konfigurasjon serverne henter ned fra Puppet-master bestemmes av hvilket vertsnavn de har. Vertsnavnet kan settes på to forskjellige måter avhengig av hvilken informasjon som er tilgjengelig før utrullingsprosessen starter:

1. Det foreligger ikke noen informasjon om serverne i forkant av utrullingen, og servernes tildelte IP-adresse bestemme hvilket vertsnavn som settes.
2. Det finnes tilstrekkelig informasjon om servernes maskinvare, slik at det i forkant av en utrulling kan tildeles roller til best egnede servere.

Med roller menes det Openstack-noder som kontrollernode, nettverksnode og computenode som hver har hver sine systemkrav.

Newman har to harddisker på 250GB, hver satt opp i RAID 1+0. Dette for å ha en redundans på den serveren som er mest kritisk. Cobbler og Puppet er begge tilgjengelige på Debian-plattformen, og med dette er Newman satt opp med Ubuntu Server 13.10. Ubuntu er også den Linux distribusjonen gruppen er mest komfortable med. I dette kapitellet vil det henvises til den essensielle konfigurasjonen av Newman. Fullstendig konfigurasjon finnes i vedlegg.

5.1.1 Cobbler

Cobbler lastes ned og installeres med *Advanced Packaging Tool* (APT) på Newman. For enklere å kunne holde oversikt over Cobbler-installasjonen og dens konfigurasjon, installeres også web-grensesnittet til Cobbler.

```
apt-get install cobbler cobbler-web
```

Før Cobbler settes opp til å rulle ut ønskede operativsystem, kan den nåværende installasjonen verifiseres ved å eksekvere følgende kommando:

```
cobbler check
```

Check-funksjonen forteller oss hvilke avhengigheter Cobbler har for å kjøre optimalt. Nedenfor vises en utskrift av denne kommandoen:

```
root@newman:~# cobbler check
The following are potential configuration items that you may want to fix:

1 : some network boot-loaders are missing from /var/lib/cobbler/loaders, you may ru
n 'cobbler get-loaders' to download them, or, if you only want to handle x86/x86_64
 netbooting, you may ensure that you have installed a *recent* version of the sysli
nux package installed and can ignore this message entirely.  Files in this director
y, should you want to support all architectures, should include pxelinux.0, menu.c3
2, elilo.efi, and yaboot. The 'cobbler get-loaders' command is the easiest way to r
esolve these requirements.
2 : debmirror package is not installed, it will be required to manage debian deploy
ments and repositories

Restart cobblerd and then run 'cobbler sync' to apply changes.
```

Figur 5.1: Cobbler check utskrift

Sjekkfunksjonens utskrift informerer om at det er to mangler som bør løses. I utskriften er det også beskrevet hvordan disse manglene skal løses.

```
cobbler get-loaders
```

```
apt-get install debmirror
```

For å kunne administrere Cobbler via web-grensesnittet, opprettes det en bruker *autostack* i Cobbler sin brukerkonfigurasjon.

```
htdigest /etc/cobbler/users.digest "Cobbler" autostack
```

I web-grensenittet kan mye av den nødvendige konfigurasjonen gjøres, men for at det på et senere tidspunkt kan være aktuelt å lage en prosedyre for automatisk oppsett av en server lik Newman, gjøres all konfigurasjon i kommandolinje. Web-grensesnittet blir her kun benyttet for å verifisere den konfigurasjonen som gjøres i kommandolinje.

Av operativsystem som skal rulles ut med Cobbler, er *Ubuntu Server 13.10* valgt. Samtidig som dette ble anbefalt av oppdragsgiver, er også dette som tidligere nevnt den Linux distribusjonen gruppen er mest komfortabel med. Dette valget vil ikke ha noen konsekvenser for utvikling av løsningen da Openstack er tilgjengelig for Ubuntu. Image-fil for Ubuntu Server 13.10 lastes ned, deretter blir den mountet på Nemwan. Når dette er gjort kan operativsystemet importeres til Cobbler.

```
wget http://www.ubuntu.com/<filbane-image-fil>
```

```
mount -o loop ubuntu_serv13.10.iso /mnt/iso
```

```
cobbler import --name=ubuntu-server13.10 --path=/mnt --breed=ubuntu
```

En automatisk installasjon av Ubuntu krever at maskinen som skal installeres mottar en svarfil (også kjent som preseed) som inneholder svar på alle spørsmålene som stilles under installasjonen av operativsystemet. En slik svarfil kommer som standard når operativsystemer importeres til Cobbler. Det tas utgangspunkt i svarfilen som opprettes under importeringen. Denne svarfilen inneholder de fleste svar som trengs under installasjonen, men må tilpassses med noen ekstra svar som er nødvendige for å automatisere installasjonen på de serverne gruppen har til rådighet. Tilpasning av denne filen vil i praksis si at det legges til ekstra svar for å eksempelvis bestemme hvilket nettverkskort som skal benyttes under installasjonen, hvordan disker skal formateres og hva som eventuelt skal skje på slutten av installasjonen.

Svarfilen har ingen funksjon før den knyttes opp mot en Cobbler profil. Når Ubuntu ble importert tidligere, ble det automatisk opprettet en profil, og den editerte svarfilen må nå knyttes til denne profilen.

```
cobbler profile edit --name=ubuntu-server13.10 --kickstart=/var/lib/cobbler/kickstarts/ \
ubuntu-server.openstack.preseed
```

I denne løsningen benyttes kun servere. Skulle det senere bli aktuelt å rulle ut løsningen på flere typer maskinvare, kan dette løses ved å ha flere typer svarfiler tilpasset ulik maskinvare. Svarfilen benyttet i denne oppgaven finnes i vedlegg Preseed(s.90).

I konfigurasjonsfilen til Cobbler må det defineres hvilken maskin som skal være Cobbler-server og hvilken maskin som skal håndtere PXE forespørsler (next_server). Newman vil ha begge disse rollene, og de aktuelle parametrene i konfigurasjonsfilen ble derfor endret.

```
/etc/cobbler/settings:
  ↪ server: 10.0.0.254
  ↪ next_server: 10.0.0.254
```

Når Cobbler skal rulle ut et operativsystem på en maskin, må maskinen boote fra nettverket (PXE). Maskinen er avhengig av å få tildelt en IP-adresse under PXE for å kunne kontakte Newman. Det er derfor behov for at Newman settes opp som en DHCP-server, hvor Cobbler videre konfigureres til å styre DHCP-konfigurasjonen.

```
apt-get install isc-dhcp-server
```

```
/etc/cobbler/settings:
  ↪ manage_dhcp: 1
```

Newman konfigureres til å dele ut IP-adresser i 10.0.0.0/24 nettet hvor Newman selv har IP-adressen 10.0.0.254. Det er i dette nettet Newman skal utføre hele utrullingen av Openstack.

For at endringene gjort i konfigurasjonsfilen til Cobbler skal tre i kraft, må de synkroniseres. Etter hver synkronisering må også Cobbler-tjenesten restartes.

cobbler sync

service cobbler restart

Slik løsningen er satt opp, kreves det at Newman har to nettverkskort, som i systemet er navngitt *eth0* og *eth1*. Nettverkskortet eth0 står tilkoblet skolen sitt offentlig nettverk og mottar dynamisk IP-adresse. Nettverkskortet eth1 står tilkoblet switchen hvor resten av serverne som er del av løsningen står tilkoblet. Newman skal kun tilby dynamisk IP-adresse til disse serverne og det er derfor viktig at Newman kun svarer på DHCP forespørsler på eth1.

/etc/default/isc-dhcp-server:

```
↪ INTERFACES='eth1'
```

Selv om Newman i dette tilfellet står tilkoblet nettverket til høgskolen, vil ikke dette være en begrensening eller forutsetning i løsningen. Kravet er at Newman har Internettilgang på eth0.

For å kunne ta automatisering av løsningen ett steg videre, ble det undersøkt muligheter for å automatisk velge at en server skal boote fra nettverket kun én gang. Etter at serveren har bootet fra nettverket og er ferdig med utrullingen av operativsystem, vil serveren foreta en reboot før det nyinstallerte operativsystemet startes. Det er derfor viktig at serveren ikke på nytt velger å boote fra nettverket. Da vil hele installasjonsprosessen skje igjen og forbli i en evig PXE loop. Cobbler har en metode for å kunne forhindre PXE loops for maskiner som har PXE boot satt som første prioritet i boot-rekkefølge i BIOS. For at valget om PXE boot skal automatiseres, kreves det likevel en manuell jobb ved å verifisere at PXE boot er satt som første prioritet i BIOS, noe som vil bryte med intensjonen om en automatisert løsning. For å kunne si at løsningen er fullt automatisert, vil en slik endring av boot-rekkefølgen i BIOS være en forutsetning.

5.1.2 Nettverk

En forutsetning for at utrulling av operativsystem med Cobbler skal fungere etter intensjon, må serverne som det rulles ut på ha tilgang til Internett. Dette observeres ved å analysere nettverkstrafikken mellom Newman og en server under en installasjon, hvor trafikkdata viser at serveren prøver å hente informasjon fra Internett. Denne observasjonen gjør det nødvendig å sette Newman opp som en ruter for å sende trafikk mellom det eksterne og interne nettverket.

For å sette opp ruting-funksjonalitet, konfigureres Network Address Translation (NAT) på Newman. Dette gjøres ved å tillate trafikk som ankommer det eksterne nettverket blir endret før det sendes videre ut på interne nettverket. Det settes også opp at trafikk som skal til en server bak NAT-nettverket kun skal rutes hvis trafikken har startet fra innsiden av NAT-nettverket.

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
iptables -A FORWARD -i eth0 -o eth1 -m state --state RELATED,ESTABLISHED -j ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth0 -j ACCEPT
```

For at de definerte ruting-reglene skal ha noen funksjon, må IP-versjon 4 videresending aktiveres. Dette gjøres ved å endre følgende konfigurasjonsfiler:

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

/etc/sysctl.conf:

```
↪ net.ipv4.ip_ forward=1
```

Denne konfigurasjonen vil ikke tre i kraft før kernel-parameterne oppdateres.

```
sysctl -p /etc/sysctl.conf
```

Rutingtabellen definert på Newman vil bli slettet hver gang serveren restarter. For å ikke miste rutingtabellen ved restart, tas det backup av rutingtabellen, og videre eksekveres et skript som gjenoppretter denne tabellen hver gang Newman restartes, basert på backupen.

```
iptables-save > /etc/iptables.rules
```

/etc/init.d/iptables.rules.sh:

```
↪ iptables-restore < /etc/iptables.rules
```

5.1.3 Puppet

Når serverne tilknyttet Newman er ferdig installert med operativsystem, skal de på nytt kontakte Newman for å hente ned sin respektive nodekonfigurasjon. Til dette benyttes orkestreringsverktøyet Puppet. Newman har den såkalte rollen som Puppet-master.

```
apt-get install puppetmaster
```

Med Puppet-installasjonen følger en standard konfigurasjon. Denne konfigurasjonen erstattes med en mye mer omfattende konfigurasjon, generert med Puppet.

```
puppet master --genconfig > /etc/puppet/puppet.conf
```

Konfigurasjonsfilen *puppet.conf* består hovedsaklig av tre seksjoner; *agent*, *master* og *main*. Agent-seksjonen bestemmer hvordan Puppet-agenter skal forholde seg til Puppet-masteren. I denne oppgavens tilfelle vil det være hvilket Puppet-miljø agentene skal laste ned sin nodekonfigurasjon fra. Master-seksjonen definerer Puppet-masterens lokasjoner for konfigurasjonsfiler, loggføring, et cetera. Under main-seksjonen defineres alle variable som skal gjelde globalt som eksempelvis hvilken server i infrastrukturen som skal være Puppet-master.

/etc/puppet/puppet.conf:

```
↪ [main]
↪ server=newman.localdomain
```

Servernavnet, spesifisert ovenfor, krever at det finnes et oppslag i *hosts*-filen på Newman, slik at det er en IP-adresse tilknyttet *newman.localdomain*. Det legges også på et alias, *newman*, slik at IP-adressen er tilknyttet vertsnavnet.

/etc/hosts:

```
↪ 10.0.0.254 newman.localdomain newman
```

Puppet-agenter identifiserer seg selv ovenfor Puppet-master ved å sende SSL sertifikater som blir generert første gang Puppet kjøres på agentene. Denne formen for autentisering brukes til å verifisere at Puppet-agentene snakker med riktig Puppet-master og omvendt. Newman er satt til å automatisk signere alle innkommende sertifikatforespørsler fra Puppet-agenter som kobler seg til. Dette for å understøtte automatisering av Puppet-oppsettet.

For å skille utvikling fra produksjon, blir egne *environments* (*miljøer*) definert i Puppet. Dette tillater agenter å kunne konfigureres i henhold til et gitt miljø. Dette for å kunne teste ny nodekonfigurasjon før den nye konfigurasjonen implementeres i produksjonsmiljøet. Utviklingsmiljøet, *development*. opprettes på Puppet-master.

cd /etc/puppet

mkdir -p environments/development

Når utviklingsmiljøet opprettes, skal det ta utgangspunkt i produksjonsmiljøet. Etter at utviklingsmiljøet er klonet (av produksjonsmiljøet), kan videre konfigurasjon foregå i utviklingsmiljøet. Når gruppen er fornøyd med en utviklet komponent, kan arbeidet eksporteres til produksjonsmiljøet for å oppdatere den faktiske konfigurasjonen agentene forholder seg til. Det som i praksis eksporteres, er de Puppet-modulene som er benyttet for å utvikle den nye komponenten, og manifestet (site.pp) som styrer nodekonfigurasjonen i infrastrukturen.

Puppet-miljøene må defineres i Puppet-konfigurasjonsfilen der det spesifiseres hvor modulene til hvert miljø er lokalisert, samt hvor nodekonfigurasjonen, site.pp, ligger.

/etc/puppet/puppet.conf:

```
↪ [production]
↪ modulepath = $confdir/modules
↪ manifest = $confdir/manifests/site.pp
↪ [development]
↪ modulepath = $confdir/environments/development/modules
↪ manifest = $confdir/environments/development/manifests/site.pp
```

Hvert miljø vil ha sitt eget site.pp manifest som definerer hvordan noder skal konfigureres i det aktuelle miljøet.

5.1.4 Git

Git er et versjonskontrollverktøy som hjelper gruppen med å holde orden på de skriptene og modulene som ble utviklet i løpet av prosjektetperioden. Det finnes flere alternativer til versjonskontroll, men Git ble valgt fordi det er distribuert og det finnes mange relevante Puppet-manifester tilgjengelig på nettstedet GitHub[1] som kan lastes ned med dette verktøyet. Git installeres og konfigureres med følgende kommandoer:

```
apt-get install git
```

```
git config --global user.name "Autostack"
```

Løsningen vil bestå av mange lokasjoner med konfigurasjonsfiler det vil være aktuelt å versjonskontrollere. Det opprettes med dette et Git-*repository* på hjemmeområdet til root-brukeren i mappen *autostack*.

```
cd /root/autostack
```

```
git init
```

Det opprettes en mappe for både Cobbler og Puppet i dette repositoriet for å holde oversikt over konfigurasjonsfilene. Når filer som benyttes i løsningen endres eller opprettes, kopieres de til dette repositoriet som videre *pushes* til et Git-repository lokalisert på GitHub. Alle essensielle konfigurasjonsfiler til Cobbler og Puppet kopieres inn i det lokale repositoriet og deretter eksekveres følgende kommandoer:

```
git add *
```

```
git commit -a --author="AutoStack <>" -m "Comment"
```

```
git remote add origin ssh://github.com/gtufte/autostack.git
```

```
git push origin master
```

Github-repositoriet *autostack* er opprettet på GitHub-brukeren til Geir André Tufte, men repositoriet er delt med resten av gruppemedlemene slik at alle kan endre innholdet i repositoriet. Ved å signere nye endringer som blir pushet til repositoriet med *author* parametret, vil det være enkelt å se hvem som har utført endringen og når endringen ble utført.

[1] www.github.com

5.2 Oppsett av installasjonservere

I dette avsnittet vil alle serverne som skal ta del i utrullingen (eksl. Newman) bli omtalt som installasjonserverne.

5.2.1 Puppet

Som nevnt tidligere skal installasjonserverne kontakte Newman på nytt for å hente ned sin respektive nodekonfigurasjon, etter at operativsystem er blitt installert. Dette gjøres ved å avslutningsvis under utrulling av operativsystem, laste over et bash-skript til installasjonserveren. Skriptet gjøres eksekverbart og satt til å kjøre ved neste oppstart. Dette skriptet bestemmer blant annet hvordan de forskjellige Openstack-rollene tildeles. Rollene bestemmes for to forskjellige situasjoner:

1. Hvilke servere som er best egnet til de forskjellige Openstack-rollene er ukjent, og det er rekkefølgen for når Newman oppdager serveren i nettet, som bestemmer rolletildelingen. Dette gjøres ved å la serveren, som får den første IP-adressen (10.0.0.1) i DHCP-*scoopet*, bli en Openstack-kontrollernode. Serveren, som får den andre IP-adressen i scoopet, blir en Openstack-nettverksnode. Alle andre servere som kontakter Newman etter dette vil bli computenoder. Rolletildelingen koker derfor ned til rekkefølgen for når serverne startes.
2. Maskinvaren på serverne er kjent, og det er blitt gjort en vurdering av hvilke servere som egner seg best til de forskjellige Openstack-rollene. Når rollene på forhånd er tildelt, kan skriptet endres med et lite grep som endrer hvordan tildelingen skjer. Istedenfor å bestemme roller på bakgrund av IP-adresse, bestemmes rollen nå på bakgrund av MAC-adresser. Dette gjør at rekkefølgen for når serverne starter eller kontakter Newman er irrelevant.

I tillegg til at skriptet bestemmer hvilke roller hver enkelt server skal ha, installerer også skriptet Puppet på serveren. Deretter konfigureres Puppet slik at serveren vet hvilken maskin i nettet som er Puppet-master, og avslutningsvis starter nedhenting av nodekonfigurasjon. Til å hente ut IP og MAC-adresseverdier fra serverne under rolletildelingen benyttes verktøyet *Facter*. Dette verktøyet følger med når Puppet blir installert, noe Puppet bruker aktivt til konfigurasjon av noder.

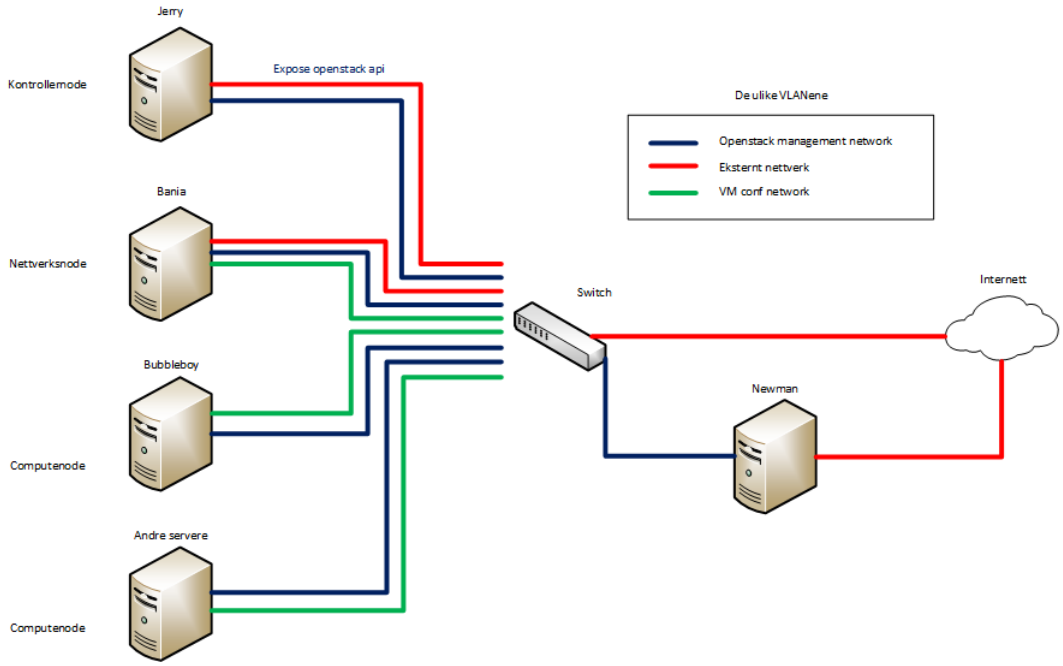
Skriptet som er blitt benyttet til å initialisere installasjon av Puppet, og som igangsetter resten av Openstack-installasjonen, finnes i vedlegg Newman startup script(s.92).

5.3 Manuelt oppsett av OpenStack

For at gruppen enklere skal kunne automatisere installasjonen av et Openstack multinode-miljø, er det en fordel å først foreta en manuell installasjon for å bli kjent med de aktuelle Openstack-komponentene. Ved den manuelle installasjonen av Openstack, skal det etableres et Openstack multinode-miljø. Et slikt miljø består av én kontrollernode, én nettverksnode og én eller flere computenoder. Det benyttes en installasjonsguide som tar for seg et slikt oppsett [15], og i tillegg benyttes en guide som ble utarbeidet når Høgskolen i Gjøvik sin egen skyløsing, SkyHiGh, ble satt opp.

5.3.1 Fysisk oppsett

Det fysiske oppsettet av Openstack kan variere i forhold til de ulike scenariene det implementeres i, hvor det skal tilpasses brukere og det utstyret man har tilgjengelig. I skissen nedenfor vises oppsettet som er satt opp manuelt i denne oppgaven, og som benyttes til implementeringen som følger.



Figur 5.2: Openstack-oppsett.

Det stilles krav til informasjon og utstyr fra kunden, da det er tenkt at AutoStack, som tilbyr denne løsningen, kun skal bidra med en server til utrulling av et Openstack multinode-miljø, samt en switch som er konfigurert i henhold til de kravene et Openstack multinode-miljø stiller. Følgende informasjon kreves fra kundens side:

- Minimum tre servere for å dekke Openstack-rollene kontroller, nettverk og compute. Som et absolutt minimum krever nettverksnoden tre nettverkskort, computenoden krever to, og kontrollernoden krever ett nettverkskort.
- Hvis kunden på forhånd har vurdert hvilke servere som egner seg best til de forskjellige Openstack-rollene, må MAC-adresser til kontroller,- og nettverksnode fremskaffes.
- Hvis skyløsningen skal være tilgjengelig fra utsiden av nettverket, må Expose-IP til kontrollernoden fremskaffes. Dette vil i praksis si at kunden stiller med en IP-adresse som tilhører deres eksterne nettverk, som settes på kontrollernoden. Dette øker igjen kontrollernodens krav til antall nettverkskort fra ett til to.
- Newman krever tilgang til Internett under utrullingen, noe kunden må kunne fremskaffe. Når skyløsningen skal tas i bruk, blir det opp til kunden å avgjøre om virtuelle maskiner som opprettes skal ha tilgang til Internett eller ikke. Løsningen fungerer uten Internettilgang.

Andre krav som for kunde kan være fordelaktig, er kundens egne brukernavn og passord som gir tilgang til løsningen. Dette er ikke tatt med som en krav da det er fullt mulig å rulle ut løsningen med standard brukernavn og passord som kan overleveres til kunden etter utrullingen. Dette er også verdier som kan endres på et senere tidspunkt.

5.3.2 Kontrollernode

En Openstack-kontrollernode i et multinode-miljø kan bestå av flere Openstack-komponenteter. I denne oppgaven vil kontrollernoden ha komponentene Keystone, Nova, Neutron og Glance. Annen programvare som RabbitMQ og MySQL kommer i tilegg for å støtte Openstack-komponentene. Ved endring av konfigurasjonsfiler, vil det i de fleste tilfeller kreve en omstart av den aktuelle tjenesten, noe det ikke vises til her, men som antas at blir utført.

Meldingskøapplikasjonen, RabbitMQ, og MySQL installeres og et egendefinert passord settes for meldingskøens standardbruker.

```
apt-get install python-mysqldb mysql-server rabbitmq-server
```

```
rabbitmqctl change_password guest <password>
```

Før installasjon og konfigurasjon av Keystone påbegynnes, opprettes MySQL-databasen samt databasebrukeren Keystone er avhengig av.

```
mysql> CREATE DATABASE keystone;
```

```
mysql> GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' \
    IDENTIFIED BY '<password>';
```

Keystone installeres og deretter konfigureres Keystone til å bruke databasen som ble opprettet i forrige steg med tilhørende autorisasjonsdetaljer. Her settes også verdien *admin token* som benyttes til å etablere Keystone-tjenesten.

```
/etc/keystone/keystone.conf:
  ↪ connection=mysql://keystone:<password>@10.0.0.1/keystone
  ↪ admin_token = ADMIN
```

Til å sette opp identitets-komponenten, Keystone, settes midlertidig følgende miljøvariabler:

```
export OS_SERVICE_TOKEN=ADMIN
```

```
export OS_SERVICE_ENDPOINT=http://10.0.0.1:35357/v2.0
```

Disse miljøvariablene benyttes kun i starten for å sette opp det administrative rundt Keystone-tjenesten. Når tjenesten er etablert, skal disse verdiene nullstilles og videre konfigurasjon skal skje med administrator-brukeren.

Med Keystone opprettes alle Openstack-brukere, prosjekter(tenants), brukerroller, tjenester, et cetera. Dette vil i praksis si at Keystone databasen fylles opp med informasjon. Videre følger opprettelsen av administrator og service-prosjektene.

```
keystone tenant-create --name=admin --description="Admin Tenant"
```

```
keystone tenant-create --name=service --description="Service Tenant"
```

Administrator-prosjektet har den overordnede styringen på Openstack-installasjonen hvor informasjon om alle instanser, nettverk, rutere og prosjekter finnes, samt informasjon om hvilke computenoder som er tilkoblet kontrolleren. Service-prosjektet har som oppgave å huse de andre Openstack-tjenestene som i denne oppgaven er Glance, Nova og Neutron. Dette prosjektet vil inneholde informasjon om hvilken IP-adresse og port hver tjeneste lytter på, slik at API-kall mot tjenesten har et endepunkt (endpoint).

Brukeren som skal benyttes til videre konfigurasjon av Openstack, opprettes. Denne brukeren benyttes også til andre Openstack-operasjoner som eksempelvis opprettelse, endring eller sletting av prosjekter, instanser, et cetera. Brukeren får administratorrollen som gir privilegier til å gjøre endringer i administratorprosjektet.

```
keystone user-create --name=admin --pass=<password>
```

```
keystone role-create --name=admin
```

```
keystone user-role-add --user=admin --tenant=admin --role=admin
```

Keystone-tjenesten opprettes og tilhørende endepunkt som beskriver hvor denne tjenesten eksisterer og hvilken port det lyttes på spesifiseres.

```
keystone service-create --name=keystone --type=identity --description="Keystone \
    Identity Service"
```

Kommandoen ovenfor genererer en verdi, KEYSTONE_ID, som blir brukt til å opprette endepunktet. Denne verdien benyttes for å unikt definere tjenesten med navnet *keystone*.

```
keystone endpoint-create \
    --service-id=<KEYSTONE_ID> \
    --publicurl=http://10.0.0.1:5000/v2.0 \
    --internalurl=http://10.0.0.1:5000/v2.0 \
    --adminurl=http://10.0.0.1:35357/v2.0
```

Endepunktene er i praksis et grensesnitt til Keystone API'et. Bak kulissene benyttes dette grensesnittet av de andre Openstack-tjenesten til å utføre operasjoner. Keystone tjenesten som lytter på port 5000 benyttes til eksempelvis kommunikasjon mellom de forskjellige Openstack-tjenestene. Port 35357 er forbeholdt mer administrative oppgaver som passord bytte og tildeling av brukerroller.

Når alle kommandoene beskrevet frem til nå er kjørt, er det initielle oppsettet av identitetstjenesten ferdig og miljøvariablene som ble satt tidligere nullstilles.

```
unset OS_SERVICE_TOKEN OS_SERVICE_ENDPOINT
```

Openstack-kommandoer har i de fleste tilfeller intensjon om å endre, legge til, eller hente ut data fra en database. Slike operasjoner må autoriseres, og når de benyttede miljøvariablene ble nullstilt i forrige steg, ble videre konfigurasjon av Openstack gjort ved å anvende det som til hittil er opprettet med Keystone. Til å verifisere at stegene ovenfor har blitt riktig utført, kan følgende kommandoer eksekveres hvor autorisasjonsparameterne er lagt ved.

```
keystone --os-username=admin --os-password=<password> \
    --os-auth-url=http://10.0.0.1:5000/v2.0 token-get
```

Denne kommandoen skal returnere autorisasjonsnøkkelen som benyttes når API-kall kjøres mot Openstack. For å slippe å legge ved slike parametre hver gang en kommando eksekveres, eksporteres autorisasjonsparameterne som miljøvariabler og vil bli benyttet hver gang en Openstack-kommando kjøres. Miljøvariabler som blir satt er kun aktive per sesjon (ssh), og må settes på nytt hver gang en ny sesjon startes. For å raskt oppta arbeidet med konfigurasjon av Openstack ved en ny sesjon, opprettes det et skript som setter disse miljøvariablene. Se vedlegg openstackAuth.sh(s.88).

Glance er neste Openstack-komponent som installeres og på samme måte som Keystone, bruker Glance også en database.

```
mysql> CREATE DATABASE glance;
```

```
mysql> GRANT ALL PRIVILEGES ON glance.* TO 'glance'@'localhost' \
    IDENTIFIED BY '<password>';
```

Glance består av flere konfigurasonsfiler der det må spesifiseres hvor API-kall skal sendes og hvor data om installerte image ligger. Autorisasjon for disse funksjonene må også defineres.

```
/etc/glance/glance-{ api-paste.ini,registry-paste.ini }:
    ↪ auth_host = 10.0.0.1
    ↪ auth_port = 35357
    ↪ auth_protocol = http
    ↪ admin_tenant_name = service
    ↪ admin_user = glance
    ↪ admin_password = <password>
```

For at Glance skal velge å autentisere seg mot Keystone, må dette spesifiseres i følgende filer:

```
/etc/glance/glance-{ api.conf, registry.conf }:
  ↪ connection=mysql://glance:<password>@10.0.0.1/glance
  ↪ flavor = keystone
```

For å integrere Glance i Openstack-installasjonen, legges tjenesten til på samme måte som tidligere hvor en bruker for tjenesten opprettes, samt rolletildeling, selve tjenesten, og endpunkt for Glance-API'et.

```
keystone user-create --name=glance --pass=<password>
```

```
keystone user-role-add --user=glance --tenant=service --role=admin
```

```
keystone service-create  --name=glance --type=image  --description="Glance \
Image Service"
```

```
keystone endpoint-create --service-id=<GLANCE_ID> \
--publicurl=http://10.0.0.1:9292 \
--internalurl=http://10.0.0.1:9292 \
--adminurl=http://10.0.0.1:9292
```

Glance-komponentens konfigurasjon ble verifisert ved å laste ned og importere et image til Glance. Image av CirrOS benyttes her for å verifisere installasjonen av Glance, som senere skal brukes til videre testing når virtuelle maskiner settes opp.

```
wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-i386-disk.img
```

```
glance image-create --name="CirrOS 0.3.1" --is-public true --container-format \
bare --disk-format qcow2 < cirros-0.3.1-i386-disk.img
```

Selve verifiseringen bekreftes ved å liste ut alle aktive image som til nå er lagt til med Glance. Figur 5.3 viser resultatet av en slik liste hvor inspeksjon av statusfeltet gir indikasjon på om installasjonen av Glance har vært vellykket eller ikke. Hvis status for imaget er *active*, er komponenten riktig integrert og installasjon av neste Openstack-komponent kan starte.

På samme måte som med Keystone og Glance, opprettes MySQL-databasen for Neutron og brukeren som skal ha tilgang til den.

```
mysql> CREATE DATABASE neutron;
```

```
root@controller:~# glance image-list
+--------------------------------------+-------------+-------------+------------------+----------+--------+
| ID                                   | Name        | Disk Format | Container Format | Size     | Status |
+--------------------------------------+-------------+-------------+------------------+----------+--------+
| bf8c6fd1-dd87-4319-82ca-88680648fa26 | CirrOS 0.3.1 | qcow2       | bare             | 12251136 | active |
+--------------------------------------+-------------+-------------+------------------+----------+--------+
```

Figur 5.3: Utskrift av *glance image-list*.

```
mysql> GRANT ALL PRIVILEGES ON neutron.* TO 'neutron'@'localhost' \
    IDENTIFIED BY '<password>';
```

Sammen med installasjon av Neutron, installeres også Open-vSwitch som Neutron benytter til administrasjon av virtuelle nettverk. Open-vSwitch bruker Neutron-databasen og tilgang til denne må defineres i filen vist nedenfor. I samme fil spesifiseres også hva slags nettverkstype Open-vSwitch skal behandle, VLAN eller tunnelering, samt hvilke VLAN som støttes. I denne oppgaven brukes VLAN, hvor den fysiske switchen i løsningen er satt opp til å håndtere VLANene 2000 til og med 2099.

/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini:

```
↪ connection=mysql://neutron:<password>@10.0.0.1/neutron
↪ tenant_network_type=vlan
↪ network_vlan_ranges = physnet1:2000:2099
```

Videre blir konfigurasjonsfiler tilhørende Neutron endret. Innstillingene for databasetilkobling settes her på samme måte som ved Glance, hvor forskjellen her er autorisasjon mot Neutron databasen.

/etc/neutron/api-paste.ini:

```
↪ paste.filter_factory= \
   keystone.middleware.auth_token:filter_factory
↪ auth_host = 10.0.0.1
↪ auth_port = 35357
↪ auth_protocol = http
↪ admin_tenant_name = service
↪ admin_user = neutron
↪ admin_password = <password>
```

Autorisasjon mot meldingskøen (RabbiMQ) spesifiseres for at Neutron kan kommunisere med de andre Openstack-komponentene.

```
/etc/neutron.conf:
  ↪ rabbit_password=<password>
  ↪ rabbit_host=10.0.0.1
```

Likt som for Keystone og Glance, opprettes brukeren, tjenesten, endepunktene, samt rolletildeling for Neutron.

```
keystone user-create --name=neutron --pass=<password>
```

```
keystone user-role-add --user=neutron --tenant=service --role=admin
```

```
keystone service-create --name=neutron --type=network --description= \
"OpenStack Networking Service"
```

```
keystone endpoint-create --service-id <NEUTRON_ID> \
--publicurl http://10.0.0.1:9696 \
--adminurl http://10.0.0.1:9696 \
--internalurl http://10.0.0.1:9696
```

Siste Openstack-komponenten som installeres på kontrollernoden er Nova. Databaseavhengigheter kreves også for denne komponenten.

```
mysql> CREATE DATABASE nova;
```

```
mysql> GRANT ALL PRIVILEGES ON nova.* TO 'nova'@'localhost' \
    IDENTIFIED BY '<password>';
```

Denne komponenten består av flere deler hvor hver del har sin spesifikke funksjon. Nova kan også settes opp til å styre nettverket i Openstack-miljøet, men da det i denne oppgaven benyttes en dedikert nettverksnode, installeres ikke denne funksjonaliteten. Det er fortsatt iht. installasjonsguide nødvendig å aktivere videresending av IP-versjon4 trafikk.

```
/etc/sysctl.conf:
  ↪ net.ipv4.ip_forward=1
```

Autentisering mot Keystone API defineres på samme måte som tidligere, men hvor det her spesifiseres autorisasjon til databasen opprettet for Nova.

/etc/nova/api-paste.ini:

```
↪ [filter:authtoken]
↪ paste.filter_factory = \
    keystone.middleware.auth_token:filter_factory
↪ auth_host = 10.0.0.1
↪ auth_port = 35357
↪ auth_protocol = http
↪ admin_tenant_name = service
↪ admin_user = nova
↪ admin_password = <password>
↪ signing_dirname = /tmp/keystone-signing-nova
```

Nova komponenten står i senter for mye av funksjonaliteten i Openstack og består av flere og store konfigurasjonsfiler. Det vil derfor her kun vises til de mest essensielle innstillingene for Nova. For en fullstendig oversikt over konfigurasjonsfilene, henvises det til installasjonsguiden, Folsom Guide [15].

Innledningsvis spesifiseres autorisasjon mot meldingskøen for at Nova skal kunne kommunisere med de andre Openstack-komponentene.

/etc/nova/nova.conf:

```
↪ rabbit_password=<password>
↪ rabbit_host=10.0.0.1
```

For å senere kunne få konsolltilgang til instanser som opprettes i skyen, aktiveres VNC og deretter spesifiseres det hvor konsollet skal gjøres tilgjengelig fra. Dette er til fordel når web-grensenittet Openstack-Horizon installeres.

/etc/nova/nova.conf:

```
↪ vnc_enabled=true
↪ novncproxy_base_url=http://10.0.0.1:6080/vnc_auto.html
↪ novncproxy_port=6080
↪ vncserver_proxyclient_address=10.0.0.1
↪ vncserver_listen=0.0.0.0
```

Med en dedikert nettverksnode i Openstack-miljøet, er det viktig at Nova kjenner til dette og vet hvilken Neutron-API-versjon som benyttes.

/etc/nova/nova.conf:

```
↪ network_api_class=nova.network.neutronv2.api.API
↪ neutron_url=http://10.0.0.1:9696
↪ neutron_auth_strategy=keystone
↪ neutron_admin_tenant_name=service
↪ neutron_admin_username=neutron
↪ neutron_admin_password=<password>
↪ neutron_admin_auth_url=http://10.0.0.1:35357/v2.0
```

Avslutningsvis for denne konfigurasjonsfilen må også Nova få kjennskap til hvor databasen Nova benytter er lokalisert, med tilhørende autorisasjon.

/etc/nova/nova.conf:

```
↪ sql_connection=mysql://nova:<password>@10.0.0.1/nova
```

For Nova, på samme måte som med Glance og Neutron, opprettes brukeren, tjenesten, endepunktet, samt rolletildeling for komponenten.

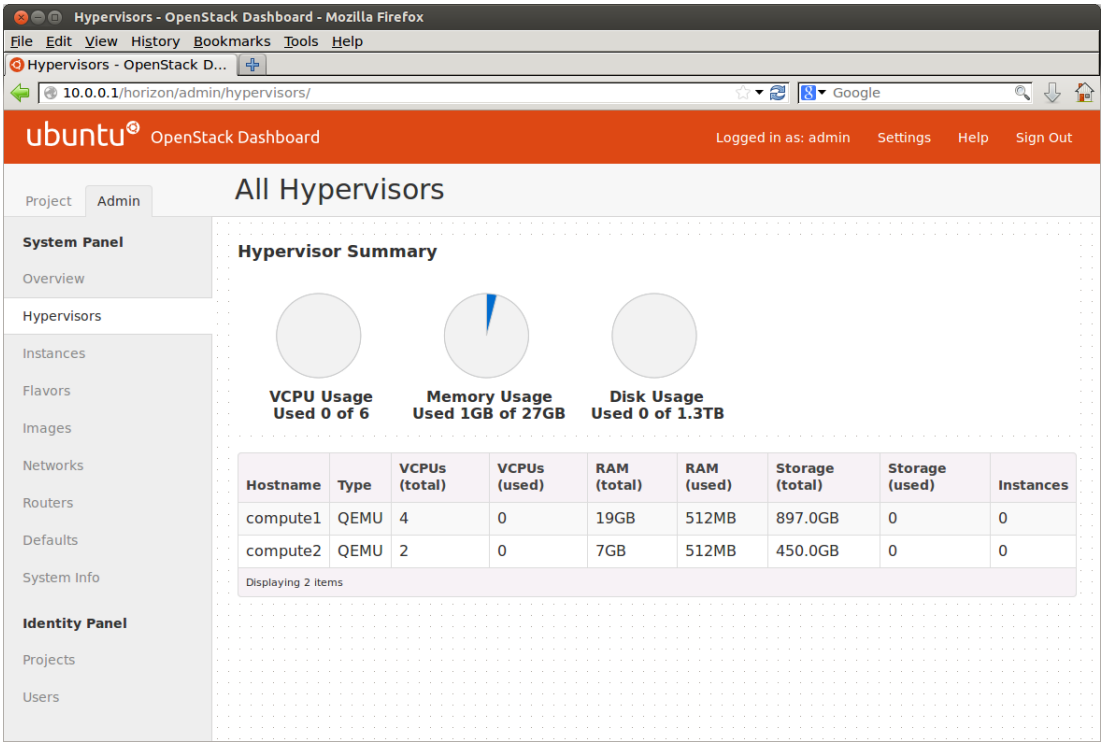
```
keystone user-create --name=nova --pass=<password>
```

```
keystone user-role-add --user=nova --tenant=service --role=admin
```

```
keystone service-create --name=nova --type=compute --description="Nova \
Compute service"
```

```
keystone endpoint-create --service-id=<NOVA_ID> \
  --publicurl=http://10.0.0.1:8774/v2/%\(tenant_id\)s \
  --internalurl=http://10.0.0.1:8774/v2/%\(tenant_id\)s \
  --adminurl=http://10.0.0.1:8774/v2/%\(tenant_id\)s
```

For å enklere kunne få oversikt over konfigurasjonen samt testing av løsningen, installeres Openstack-komponenten Horizon som gir et grafisk grensesnitt til Openstack-installasjonen via nettleser. Når Horizon er installert, vil web-grensesnittet være tilgjengelig på http://10.0.0.1/horizon. Brukernavn og passord kan etter behov opprettes der det gis tilgang til spesifikke prosjekter, noe som blir aktuelt under scenarie-utrullingen. Som en midlertidig løsning, kan administratorbrukeren benyttes. Figur 5.4 viser et eksempel på web-grensesnittet hvor en liste med alle computenoder som er tilkoblet kontrolleren og hvilke ressurser de har disponibelt.



Figur 5.4: Horizon, hypervisors

Figuren viser riktignok tilstanden slik den er når computenodene er konfigurert, noe de ikke er på dette tidspunktet i installasjonen.

5.3.3 Nettverksnode

Nettverksnoden består kun av den éne Openstack-komponenten, Neutron, som baserer seg på å tilby nettverk som en tjeneste. Denne noden fungerer som en ruter for instanser som lever i skyen og gir instanser en forbindelse ut mot det eksterne nettverket i form av Network Address Translation (NAT). På innsiden av nettet fungerer noden som en DHCP-server og tilbyr et internt nettverk for instanser. For å tilby en slik tjeneste er Neutron avhengig av Open-vSwitch.

Avhengigheten installeres og første bru, *br-int*, opprettes. Det er en integrasjonsbru som benyttes for å kommunisere med instanser.

```
ovs-vsctl add-br br-int
```

Videre opprettes broen *br-eth1* som har en forbindelse til det fysiske nettverkskortet eth1 som er tilkoblet det interne nettverket.

```
ovs-vsctl add-br br-eth1
```

```
ovs-vsctl add-port br-eth1 eth1
```

For å rute trafikk fra det interne nettet ut på det eksterne, opprettes broen *br-ex*. Den har en forbindelse mot det fysiske nettverkskortet eth2, som er tilkoblet det eksterne nettverket. Dette åpner for instansers tilgang til eksempelvis Internett, noe som er en fordel når en instans skal konfigureres med diverse tjenester.

```
ovs-vsctl add-br br-ex
```

```
ovs-vsctl add-port br-ex eth2
```

Som nevnt tidligere, er instansers tilgang ut mot det eksterne nettet basert på NAT, og med dette er ikke instansene synlige fra det eksterne nettet. For å gjøre en instans synlig fra utsiden, opererer Openstack med *flytende-IP-adresser*, som kan tildeles instanser etter behov. En flytende-IP-adresse er i praksis en adresse som tilhører det eksterne nettverket. For at nettverksnoden skal kunne koordinere trafikken til og fra instanser, må videresending av IP-versjon 4 pakker aktiveres.

```
/etc/sysctl.conf:
  ↪ net.ipv4.ip_forward=1
```

Installasjon av Neutron kommer her i form av flere pakker som er nødvendig for å tilby tjenesten beskrevet ovenfor. Dette inkluderer Netron-tjenesten, DHCP-funksjonalitet, ruting og en plugin for at Neutron skal kunne administrere Open-vSwitch.

```
apt-get install neutron-server neutron-dhcp-agent \
neutron-l3-agent neutron-plugin-openvswitch-agent
```

Konfigurasjonen av Neutron her er relativt lik konfigurasjon av samme komponent på kontrollernoden, hvor nettverksnoden forholder seg til kontrollernoden for autorisasjon.

/etc/neutron/api-paste.ini:

```
↪ [filter:authtoken]
↪ paste.filter_factory = \
   keystone.middleware.auth_token:filter_factory
↪ auth_host = 10.0.0.1
↪ auth_port = 35357
↪ auth_protocol = http
↪ admin_tenant_name = service
↪ admin_user = neutron
↪ admin_password = <password>
↪ signing_dirname = /tmp/keystone-signing-nova
```

Open-vSwitch plugin benytter MySQL-databasen lokalisert på kontrollernoden, og det er nødvendig å spesifisere tilgang til databasen i følgende konfigurasjonsfil.

/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini:

```
↪ sql_connection=mysql://neutron:<password>@10.0.0.1/neutron
```

Konfigurasjonen for ruting-funksjonen blir satt i filen vist nedenfor med autorisasjon mot kontrollernoden. Her spesifiseres også hvilken region ruting-funksjonaliteten på denne nettverksnoden skal gjelde. Region-parameteret er brukt i sammenhenger hvor Openstack-installasjoner er plassert på forskjellige geografiske lokasjoner. Selv om denne Openstack-installasjonen vil foregå i én region, er dette fortsatt et nødvendig parameter.

/etc/neutron/l3_agent.ini:

```
↪ auth_url = http://10.0.0.1:35357/v2.0
↪ auth_region = regionOne
↪ admin_tenant_name = service
↪ admin_user = neutron
↪ admin_password = <password>
↪ metadata_ip = 10.0.0.253
↪ metadata_port = 8775
```

Openstack-kommunikasjon skjer som nevnt tidligere vha. av en meldingskø, installert på kontrollernoden. Derfor konfigureres Neutron på nettverksnoden til å bruke denne meldingskøen.

/etc/neutron/neutron.conf:

```
↪ rabbit_host = 10.0.0.1
↪ rabbit_password=<password>
```

5.3.4 Computenode

Konfigurasjonen som vist i dette avsnittet vil være lik for alle computenoder som hektes på løsningen. Denne seksjonen beskriver kun det essensielle ved konfigurasjon av en computenode. Innledningsvis før konfigurasjonen kan starte, må det først verifiseres om maskinen støtter kjernebasert-virtualisering (KVM). Verktøy for å verifisere dette installeres og sjekk kjøres.

```
apt-get install cpu-checker
```

```
kvm-ok
```

Kommandoen ovenfor vil generere en melding om KVM er støttet eller ikke. Er ikke KVM støttet, vil dette i de fleste tilfeller være at virtualisering er deaktivert i BIOS. Figur 5.5 viser denne kommandoens utskrift på computenoden satt opp i denne installasjonen.

```
root@compute1:~# kvm-ok
INFO: /dev/kvm exists
KVM acceleration can be used
```

Figur 5.5: Virtualiseringssjekk

En computenode kan huse flere instanser så lenge noden har ressurser tilgjengelig. Instansene kan være av forskjellige operativsystem, tilhøre forskjellige nettverk og operere på ulike VLAN. Dette gjør det nødvendig her, som på nettverksnoden, å installere Open-vSwitch med de forskjellige bruene for å realisere et slikt nettverksoppsett. På computenodene trengs det kun å opprette integrasjonbruen, og bruen som knytter instansene til det interne nettverket.

```
ovs-vsctl add-br br-int
```

```
ovs-vsctl add-br br-eth1
```

```
ovs-vsctl add-port br-eth1 eth1
```

Videre spesifiseres Open-vSwitch plugin tilgang til Neutron-databasen samt autorisasjon mot meldingskøen på kontrollernoden.

/etc/neutronplugins/openvswitch/ovs_neutron_plugin.ini:

```
↪ sql_connection=mysql://neutron:<password>@10.0.0.1/neutron
```

/etc/neutron/neutron.conf:

```
↪ rabbit_host=10.0.0.1
↪ rabbit_password=<password>
```

På lik måte som for kontrollernoden og nettverksnoden, må videresending av IP-versjon 4 trafikk aktiveres.

/etc/sysctl.conf:

```
↪ net.ipv4.ip_forward=1
```

Virtualiseringskomponenten Nova installeres på computenoden, og autorisasjon mot den respektive databasen på kontrolleren defineres.

/etc/nova/api-paste.ini:

```
↪ [filter:authtoken]
↪ paste.filter_factory = \
    keystone.middleware.auth_token:filter_factory
↪ auth_host = 10.0.0.1
↪ auth_port = 35357
↪ auth_protocol = http
↪ admin_tenant_name = service
↪ admin_user = nova
↪ admin_password = <password>
↪ signing_dirname = /tmp/keystone-signing-nova
```

Til å verifisere delen av Openstack-installasjonen som definerer forholdet mellom kontrollernoden og computenodene, kan følgende kommando eksekveres på eksempelvis kontrollernoden:

```
nova-manage service list
```

Denne kommandoen vil liste ut status på de enkelte Nova-komponentene som kjører på kontrolleren, samt status for computenoder som er tilkoblet kontrollernoden. Figur 5.6 viser en utskrift av denne kommandoen kjørt.

```
root@controller:~# nova-manage service list
Binary           Host                                 Zone             Status     State
nova-consoleauth controller                           internal         enabled    :-)
nova-scheduler   controller                           internal         enabled    :-)
nova-conductor   controller                           internal         enabled    :-)
nova-cert        controller                           internal         enabled    :-)
nova-compute     compute1                             nova             enabled    :-)
```

Figur 5.6: Statussjekk for Nova-tjenester

Store deler av Openstack-installasjonen kan verifiseres med denne kommandoen. Med store deler menes først og fremst Keystone og Nova. Indikasjonen på at en Nova-komponent eller en computenode er riktig tilkoblet, gjøres ved å inspisere tilstandsfeltet i utskriften. Smilefjes indikerer et riktig oppsett.

5.3.5 Test av installasjon

På dette tidspunktet er Openstack-installasjonen ferdig (iht. guide) og videre konfigurasjon er til for å teste skyløsningen. Først opprettes et prosjekt som skal huse noen få testinstanser.

```
keystone tenant-create --name testProject
```

Det er mulig å gi den eksisterende administratorbrukeren tilgang til dette prosjektet, men det er mer naturlig at det opprettes en egen bruker som skal kunne administrere prosjektet.

```
keystone user-create --name=testUser --pass=<password> \
  --tenant-id <testProject ID>
```

En standard installasjon av Openstack kommer med to roller, *admin* og *member*. Disse rollene bestemmer hvilket rettighetsnivå brukere har i de ulike prosjektene, der admin-rollen gir fulle rettigheter til alle prosjekter, mens member kun gir tilgang til spesifikke prosjekter. Figur 5.7 viser listen med roller som eksisterer i denne Openstack-installasjonen.

```
root@controller:~# keystone role-list
+----------------------------------+----------+
|                id                |   name   |
+----------------------------------+----------+
| 9fe2ff9ee4384b1894a90878d3e92bab | _member_ |
| 5d917a7b190c4d3793c1c89027f10dff |  admin   |
+----------------------------------+----------+
```

Figur 5.7: Openstack-roller

Det er her ønskelig at den opprettede brukeren kun har medlemstilgang til prosjektet. Med dette tildeles *testUser* rollen som *member* i prosjektet *testProject*.

```
keystone user-role-add \
  --tenant-id <testProject ID> \
  --user-id <testUser ID> \
  --role-id <memberRole ID>
```

Nettverket instansene skal benytte i prosjektet opprettes, her med navnet *netTestProject* og definert i VLAN 2000.

```
neutron net-create \
  --tenant-id <testProject ID> netTestProject \
  --provider:network_type vlan --provider:physical_network physnet1 \
  --provider:segmentation_id 2000
```

Det opprettes et sub-nett i nettverket netTestProject som vil være de IP-adressene som tildeles instansene i prosjektet.

```
neutron subnet-create --tenant-id <testProject ID> netTestProject 50.50.1.0/24
```

En ruter for prosjektet legges til og en port på ruteren settes opp til å være tilkoblet sub-nettet i netTestProject, her med navnet *routerTestProject*.

```
neutron router-create --tenant-id <testProject ID> routerTestProject
```

```
neutron router-interface-add <routerTestProject ID> <netTestProject ID>
```

Videre opprettes nettverket *extNet*, som skal gi instansene i skyen et grensesnitt ut mot det eksterne nettverket. Merk at dette nettet eksisterer i service-prosjektet.

```
neutron net-create \
  --tenant-id <serviceProject ID> extNet \
  --router:external=True
```

På samme måte som for det interne nettverket, opprettes et sub-nett for det eksterne nettverket. I dette sub-nettet er det viktig at IP-adressene har en tilhørighet til det faktiske eksterne nettverket. I denne oppgavens tilfelle er det skolenettet som er det eksterne nettverket. IP-adressene som skal assosieres med utsiden, er reserverte adresser i skolenettet som gruppen har fått tildelt i forbindelse med oppgaven.

```
neutron subnet-create \
  --tenant-id <serviceProject ID> \
  --allocation-pool start=128.39.141.217,end=128.39.141.218 \
  --gateway 128.39.140.1 extNet 128.39.140.0/23 \
  --enable_dhcp=False
```

Ruteren opprettet tidligere må tilkobles det eksterne nettet, hvor ruteren automatisk setter første IP-adressen i nettet som gateway, her 128.39.141.217.

```
neutron router-gateway-set <routerTestProject ID> <extNet ID>
```

Computenodene bruker en metadata-tjeneste for å hente instans-spesifikk informasjon, som serves fra kontrollernoden via det eksterne nettverket. Det er derfor her nødvendig på kontrollernoden å legge til følgende rute.

```
route add -net 50.50.1.0/24 gw 128.39.141.217
```

I installasjonen av Openstack som vist frem til nå, kan instanser opprettes og tilgang til instanser er mulig gjennom Horizon. Instansene mottar ingen IP-adresse fra nettverksnoden (DHCP) og tilgang til Internett mangler, som det forventes at instansene skal ha. For å få bedre oversikt over sammenhengen mellom Openstack-komponentene og en mulighet til å løse nettverksproblemet nevnt ovenfor, ble denne installasjonen av Openstack utført flere ganger. Når gruppen hadde fått bedre forståelse for Openstack, var nettverksproblemet fortsatt ikke løst, men et godt grunnlag ble lagt for arbeidet med å automatisere installasjonen.

5.4 Utrulling av manuell Openstack installasjon

For å bli bedre kjent med Puppet, ble det utført en utrulling av den manuelle Openstack installasjonen. Denne installasjonen tar utgangspunkt i de installerte programmene samt konfigurasjonsfiler fra den manuelle Openstack-installasjonen. Intensjonen var ved denne utførelsen å rulle ut alle de aktuelle Openstack-nodene, men når gruppen følte seg trygge på bruken av Puppet etter at kontrollernoden var rullet ut, ble dette arbeidet avsluttet. Dette for å komme tidligere i gang med å bruke Openstack-modulene til Puppet som ble benyttet i utviklingen av sluttproduktet.

5.4.1 Klargjøring

Alle konfigurasjonsfilene beskrevet i den manuelle Openstack-installasjonen kopieres over til Puppet-masteren (Newman) slik at disse igjen kan kopieres tilbake til den aktuelle noden under utrullingen. For enkelhets skyld dumpes databasene, som ble fylt opp i Openstack-installasjonen, til egne filer som også kopieres til Newman. Database-dumpene skal kopieres over til kontrollernoden som skal utføre databasegjenoppretting basert på disse dumpene. Før testingen påbegynnes, verifiseres det at Puppet er installert på noden, og at den er koblet opp mot Newman.

5.4.2 Utføring

Under utvikling av denne installasjonen, jobbet gruppen i utviklermiljøet som ble opprettet tidligere. Kontrollernoden som skal utføre databasegjenoppretting må ha MySQL installert. MySQL kan installeres på flere måter, men gruppen velger å benytte MySQL-modulen utviklet av Puppetlabs. Denne modulen kan lastes ned med Puppet sitt eget verktøy for integrasjon av nye moduler.

```
puppet module install puppetlabs-mysql --environment=development
```

For utrullingen av databasedumpene og konfigurasjonsfilene, opprettes en egen modul, *autostack-controller*.

```
puppet module generate autostack-controller
```

Denne kommandoen vil generere de fleste standard filer og mapper en modul bør bestå av. Manifestfilene som inneholder detaljene for hvordan noden skal konfigureres må fortsatt opprettes. Gruppen benytter den filstrukturen *Pro-Puppet* [17] anbefaler, som består av følgende filer:

```
cd ..modules/autostack-controller/manifests
```

```
touch install.pp, config.pp, service.pp
```

Ved å bruke denne filstrukturen kan administrasjonen av modulen forenkles ved at det lages ryddige og oversiktlige manifest. Manifestet *install.pp* definerer hvilke pakker som skal installeres, *config.pp* styrer konfigurasjonsfiler og eventuelle kommandoer som skal kjøres, og *service.pp* forsikrer at aktuelle tjenester er i en bestemt tilstand. For å oppnå en plattformuavhengig modul, kan manifestet *params.pp* introduseres. Dette manifestet inneholder plattformspesifikke verdier som navn på installasjonspakker, navn på tjenester, filbaner til hvor program installeres, et cetera. Dette for at Puppet-master skal kunne tilby ulike sett med operasjoner avhengig av hvilken platform en Puppet-agent kjører på, slik at lik nodekonfigurasjon kan oppnås på flere ulike typer plattformer. Da det kun benyttes én plattform (Debian Ubuntu) i denne løsningen, vil manifestet params.pp utelukkes for å spare tid til utviklingsarbeidet. For å aktivere de aktuelle manifestene, må de inkluderes i filen *init.pp* som ble opprettet da autostack-controller-modulen ble opprettet. Avslutningsvis må det defineres hvordan nodene skal konfigureres. Dette gjøres i manifestet site.pp som styrer konfigurasjonen for hele utviklingsmiljøet. Nedenfor vises hvordan MySQL og den egenkomponerte modulen legges til i nodekonfigurasjonen for kontrollernoden.

```
node 'controller' {
   include mysql::server
   include autostack-controller
 }
```

Som en forsikring om at de essensielle delene av kontrollernoden har blitt rullet ut riktig, verifiseres installasjonen ved å eksekvere kommandoer som er avhengige av at databasen er riktig satt opp og konfigurasjonen er korrekt. Keystone funksjonaliteten kan verifiseres [18] ved å kjøre følgende kommandoer:

```
keystone token-get
```

```
keystone user-list
```

Disse kommandoene bekrefter at miljø-variabler er riktig satt og at admin-brukeren i Keystone-databasen har autorisasjon til å utføre administrative oppgaver. Da disse kommandoene ga forventet resultat, ble arbeidet med å rulle ut denne manuelle installasjonen av Openstack, avsluttet.

5.5 Automatisk installasjon av Openstack

I løpet av dette arbeidet ble Openstack sin installasjonsguide[19] for Havana-utgivelsen fulgt. Under utviklingen av løsningen som ruller ut Openstack, har gruppen jobbet i utviklingsmiljøet opprettet på Puppet-master. Når en ny del av løsningen er blitt ferdig utviklet, er den blitt implementert i produksjonsmiljøet. Utviklingen har bestått av først å finne ut hvilken programvare som skal rulles ut, for deretter å undersøke i Puppetlabs[2] om det finnes moduler for denne programvaren. I stor grad finnes det moduler for den programvaren en Openstack-installasjon skal bestå av, men flere av disse modulene har mangler. Med mangler menes det at det ikke finnes noen metode i modulen for å sette enkelte parametere som er essensielle for Openstack-installasjonen og løsningen i sin helhet for øvrig.

I tilfeller hvor det er nødvendig å sette parametere ingen Puppet-modul kan tilby, er det blitt benyttet et bash-skript (deploy-skript) som eksekveres vha. Puppet. I denne løsningen har hver Openstack-node sitt eget deploy-skript som kjører mot slutten av utrullingen for å sette parametere som ikke blir satt med Puppet.

Under utviklingen av manifestet, som skal rulle ut Openstack, har gruppen jobbet i denne sekvensen:

1. Undersøke hvilken funksjonalitet som trengs og se etter tilhørende Puppet-modul.
2. Laste ned og installere modul i utviklingsmiljøet.
3. Legge til modulens funksjonalitet i site.pp (manifestet som styrer nodekonfigurasjonen).
4. Teste ny funksjonalitet på nodene og gjøre eventuelle tilpasninger i site.pp.
5. Mangler som oppdages ved Puppet-modulen legges til nodens deploy-skript
6. Når ønsket funksjonalitet er i orden, implementere løsningen i produksjonsmiljøet.

[2] https://forge.puppetlabs.com/puppetlabs

Når ny funksjonalitet skal legges til i site.pp, tas det utgangspunkt i dokumentasjonen for den spesifikke modulen som beskriver hvordan modulen skal brukes og hvordan enkelte standard parametere defineres. Denne dokumentasjonen følger med som en tekstfil når ny modul installeres, men er også tilgjengelig på Puppetlabs sine hjemmesider. Når modulene ikke er tilstrekkelig dokumentert, vil inspeksjon av manifestene for modulene gi hint om hvilke parametere som kan endres.

5.5.1 Openstack-komponenter

Denne oppgaven fokuserer hovedsakelig på Openstack-komponentene Keystone, Glance, Neutron og Nova. Da løsningen skal kunne brukes av personer uten Openstack kompetanse, er også komponenten Horizon lagt ved for enkel administrasjon av skyen. For alle de komponentene som løsningen bygger på, finnes en tilhørende modul i Puppet, utviklet av Puppetlabs. Puppet-moduler installeres på følgende måte:

```
puppet module install <modul navn> --environment development
```

Den første Openstack-komponenten som installeres er Keystone som kun installeres på kontrollernoden. Keystone-modulen som benyttes er utviklet av Puppetlabs, og er avhengig av modulene Apache, inifile, MySQL og stdlib. Avhengighetene installeres automatisk når Keystone-modulen installeres. Disse modulene kan hver for seg benyttes separat, men slik Openstack-modulene fungerer, kan eksempelvis MySQL-parametere defineres ved å sende parametere til Keystone-modulen. Dette er parametere kun relevant for Keystone, så det vil også være nødvendig å definere MySQL-modulen separat for å garantere at MySQL er installert og brukernavn og passord blir satt for tilgang til MySQL-konsollet. MySQL-relevante parametere for de resterende Openstack-modulene på kontrolleren settes i den aktuelle modulen. De parameterne som definerer Keystone-komponenten, er i stor grad lik de parameterne som er definert under den manuelle installasjonen av Openstack.

Glance-modulen lastes ned og installeres på lik måte som Keystone-modulen, og er også kun installert på kontrollernoden. Denne modulen har ingen medfølgende metode for å legge til image for virtuelle maskiner, derfor legges dette til i deploy-skriptet som laster ned et image for Ubuntu Server 12.04 LTS og Cirros, som deretter importeres til Glance.

I et multinode-oppsett av Openstack med en dedikert nettverksnode er det fortsatt nødvendig å installere Neutron-komponentene på kontrolleroden. Dette for å kunne fortelle computenodene, som forholder seg til kontrolleren, hvordan nettverket styres. Multinode-oppsettet denne oppgaven bygger på, er det en dedikert nettverksnode som håndterer nettverkstraffikken til og fra instanser i skyen. Nettverksnoden står som DHCP-server for instansene og kan, med konfigurasjonen som rulles ut, skille mellom 100 nettverk basert på VLAN-tagging. Alle 100 VLANene går gjennom samme fysiske nettverkskort hvor ethernet-headeren på utgående pakker blir tagget med den respektive VLAN-iden.

Virtualiseringstjenesten, Nova, har mange undermoduler med hver sin funksjon. Dette er funksjoner som bestemmer hvilken computenode som skal huse instanser som opprettes, console-tilgang på instanser, et cetera. Nova-komponenten kan også i enkelte Openstack-oppsett styre nettverkskonfigurasjonen, noe som ikke vil være relevant for denne oppgaven, og derfor må Nova konfigureres til å bruke den dedikerte nettverksnoden.

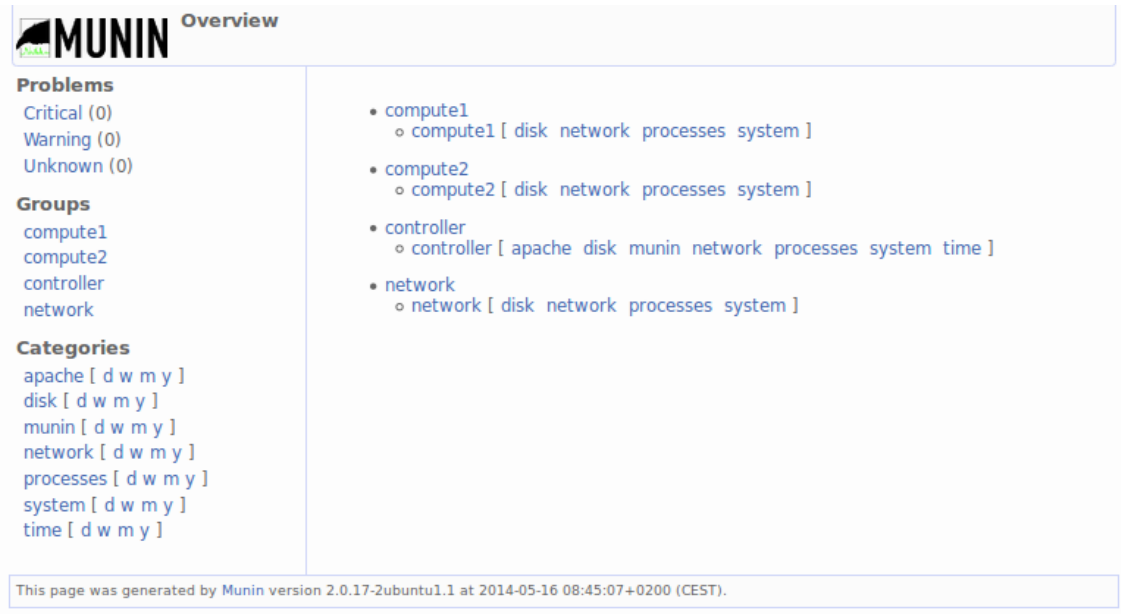
5.5.2 Annen programvare

Openstack-tjenestene som kjører på de ulike serverne i løsningen er avhengig av være synkronisert, og med dette installeres NTP på alle nodene. En NTP-modul, utviklet av Puppetlabs, lastes ned og implementeres i site.pp. Kontrolleren settes opp til å hente data fra NTP-servere utenfor skolenettet, mens resterende noder vil hente data fra kontrollernoden.

I denne løsningen benyttes RabbitMQ som meldingskø-server som også er standard for Openstack-tjenestene. Det finnes også her en RabbitMQ-modul utviklet av Puppetlabs som lastes ned og installeres. Modulen benyttes til å forsikre om at RabbitMQ-server er installert. Puppetlabs RabbitMQ-modul har ingen metode for å sette passordet for standard-brukeren tilknyttet RabbitMQ. Kommandoen for å sette dette passordet legges til i deploy-skriptet.

5.5.3 Overvåkning

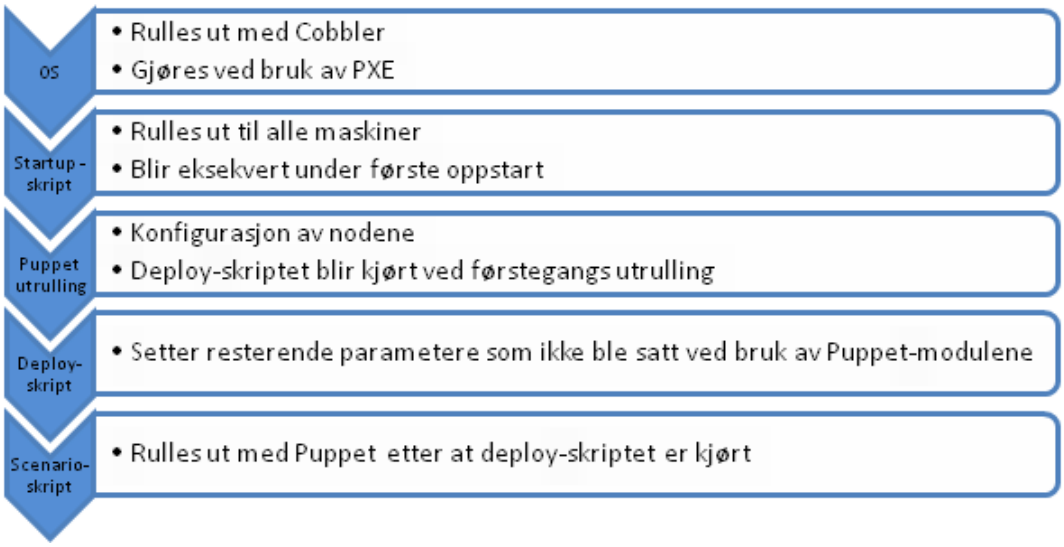
For overvåkningen rulles Munin ut på alle nodene, hvor kontrollernoden har fått tildelt rollen som Munin-master. Det vil si at kontrolleren henter inn data fra de andre nodene i nettet for å kunne lage trend-grafer basert på aktiviteten på nodene. Munin-masteren vil også overvåke seg selv. Det finnes Puppet-moduler for Munin, men disse modulene løser ikke utfordringen som oppstår når Munin-masteren regelmessig skal lete etter nye noder som dukker opp i nettet. Munin installeres derfor på enklest mulig måte med Puppet sin type-reference *package*, hvor resten av Munin-master konfigurasjon gjøres med et bash-skript. Skriptet som oppdager nye Munin-noder i nettet, ligger innledningsvis på Puppet-masteren og kopieres over til Munin-master under utrulling. Det opprettes en rutine (crontab) som eksekverer skriptet hvert femte minutt for å oppdage nye noder. For å oppdage nye noder, er verktøyet Nmap benyttet, hvor det tas utgangspunktet i en ping-sweep som sjekker om det finnes noder i nettet som hittil ikke er registrert i konfigurasjonsfilen til Munin-master. Figur 5.8 viser web-grensesnittet på Munin-master og oversikt over de tilkoblede nodene.



Figur 5.8: Munin oversikt.

5.6 Gjennomføring/Utviklingsprosess

Nedenfor følger en figur som visualiserer prosessen for utrulling av Openstack, med komponentene som benyttes for dette. Her beskrives sammenhengen mellom de ulike skriptene og hvilken rekkefølge som gjelder for implementering av disse.



Figur 5.9: Openstack utrullingsprosess

5.6.1 Informasjonstilegning

En essensiell del av et prosjektarbeid, hvor ingen av gruppens medlemmer har kjennskap til skyløsningen som skal tas i bruk, er at informasjonstilegningen gjøres effektivt. For gruppen var Cobbler og Puppet delvis kjente områder, mens Openstack var den ukjente, og samtidig mest omfattende. For de to førstnevnte, delte gruppen seg opp slik at noen fokuserte på Cobbler, mens andre på Puppet. Slik ble det effektivt tidsforbruk frem mot Openstack. Da den skulle påbegynnes, ble det arbeidet mye på en manuell installasjon av Openstack, men også her delte gruppen seg opp slik at noen arbeidet i et virtuelt miljø, mens andre testet direkte på den fysiske maskinvaren som benyttes i den endelige løsningen. Dokumentasjonen benyttet for Openstack varierte underveis, da gruppen erfarte at guiden som ble benyttet i første omgang ikke hadde tilstrekkelig dokumentasjon til å kunne benyttes. I de fleste sammenhenger ble diverse guider og nettressurser tatt i bruk for informasjonstilegning, mens i tilfellet med Puppet ble boken Pro Puppet[17] benyttet.

Først da den automatiske utrullingen av Openstack var helt i sluttfasen, så gruppen nærmere på implementering av overvåkning. Det ble tidlig tatt en vurdering av om det kun ville være tid nok til å implementere én av disse, og ut fra gruppens eksisterende kunnskaper om Munin, falt valget på den. Det var altså nødvendig med svært lite ressursbruk på informasjonstilegning rundt Munin, og fokuset gikk på å skripte en funksjonalitet som integrerte det inn i den eksisterende løsningen.

5.6.2 Valg av utviklingsmodell

Valget av utviklingsmodell for denne oppgaven ble vurdert slik at det var hensiktsmessig å benytte en kombinasjon av egenskapene til flere modeller, for å få en så konstruktiv arbeidsprosess som mulig. Med tanke på at dette prosjektet var todelt; én implementeringfase av Cobbler, Puppet og Openstack, samt én fase for skripting og rapportskriving. Disse fasene krever en såpass ulik tilnærming at det ville lønne seg å benytte flere utviklingsmodeller. Det ble dermed benyttet en sekvensiell/inkrementell modell for de tre verktøyene Cobbler, Puppet og Openstack, mens for den andre delen ville det være mange uforutsigbarhet som krevde kortere planleggingsperioder, noe Scrum passet best til. I tillegg til disse to modellene var gruppen avhengig av å kunne gå tilbake og endre på systemkonfigurasjoner underveis, noe som ga plass for modellen *samtidige-aktiviteter*.

I startfasen av prosjektet ble det lagt forholdsvis mye vekt på å arbeide etter de valgte modellene, men ettersom gruppen kom godt i gang med oppgaven, ble det en naturlig flyt i arbeidsmetodikken som førte til at det ble et visst avvik fra planen. Mye av grunnen til dette skyldtes antageligvis at gruppen til daglig samles for arbeid med prosjektet, og dermed ble det unaturlig å følge modellene sporadisk med fokus på møter og planlegging.

Ut fra gruppens valg av utviklingsmodeller, ble det definert faste møteformer under prosjektets periode. I perioden med sekvensiell/inkrementell modell ville det være et fast møte hver fredag morgen, mens det under Scrum ville være daglige møter på 15 minutter. Disse planene er for så vidt blitt fulgt, men på bakgrunn av at gruppen stort sett har vært fysisk samlet under arbeidet, ble møteformatet mer flytende slik at diskusjoner og vurderinger ble tatt fortløpende.

5.6.3 Kommentar til Gantt-skjema

Gantt-skjemaet beskrevet i prosjektplanen har vært utgangspunktet for gruppens planlegging av tidsforbruk under denne prosjektperioden. Tidlig i prosjektet ble det lagt mye vekt på denne tidsplanen, slik at det ikke skulle dukke opp uønskede overraskelser underveis. Den første delen av skjemaet som omtaler prosjektplanen er den som stemmer best overens med hvordan perioden faktisk artet seg. Utarbeidelsen av en prosjektplan innebar mange like faktorer uavhengig av gruppe og oppgaveform, og var forholdsvis grei å forutse på forhånd, spesielt med tanke på at det gjaldt innenfor en kort periode.

Utviklingsfasen ble initiert med verktøyene Cobbler og Puppet, som etter planen skulle være etablert som fungerende løsninger i løpet av få dager. Implementering av disse verktøyene gikk iht. planen, og gruppen var foran tidsskjema da den manuelle installasjonen av Openstack ble påbegynt. Ettersom dette var den modulen det var knyttet mest usikkerhet til, var det satt av omkring dobbelt så lang tid til den som til de andre, og dette viste seg også å være høyst nødvendig. Grunnet de nevnte utfordringene med guiden om ble benyttet i forbindelse med den manuelle installasjonen, viste dette seg å være en langt mer tidkrevende modul enn antatt. Dermed kom gruppen forholdsvis sent i gang med skriptingen ut fra Gantt-skjemaet. På dette tidspunkt ble det gjort fortløpende vurderinger av hvor langt gruppen hadde mulighet til å strekke seg for å bli ferdig med ulike problemområder før neste fase måtte påbegynnes. Dermed ble Gantt-skjemaet fra dette punktet gitt lite oppmerksomhet.

Rapporten har fått oppmerksomhet hele veien under prosjektets varighet. Gruppen hadde likevel gjort en feilvurdering av nødvendig tidsforbruk på denne i forkant og ble dermed nødt til å gå videre fra utviklingsfasen tidligere enn antatt for å fullføre dokumentasjonen.

5.6.4 Kommentar til brukerveiledning

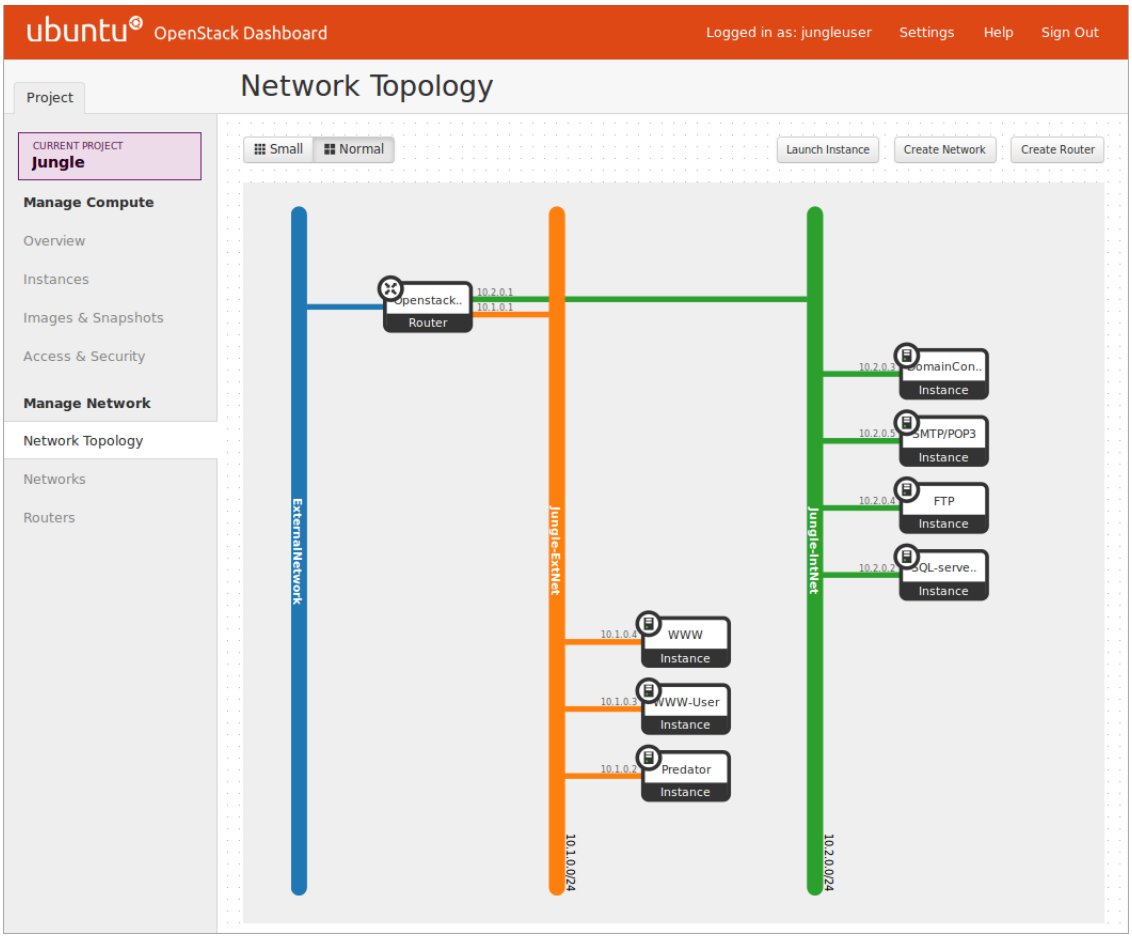
Som vist i vedlegg er det blitt generert en Brukerveiledning(s.114) som søker å assistere brukere av Openstack med å løse de vanligste oppgavene de møter på. Dette inkluderer blant annet innlogging, opprettelse av instanser, oversikt og liknende. Hensikten med denne brukerveiledningen er ikke å tilby hjelp for administratorer av Openstack, da de er ment å inneha den nødvendige kunnskapen på disse områdene.

5.7 Utrulling av scenario

Til demonstrasjon og testing av løsningen, ble det opprettet et bash-skript som ruller ut et scenario i skyen, basert på et tema hentet fra bacheloroppgaven *Jungle*. Da dette kun er en demonstrasjon og et eksempel på hvordan løsningen kan brukes, vil ikke de faktiske instansene som utgjør scenariet være satt opp med den programvaren beskrevet i infrastrukturen.

5.7.1 Jungle-scenario

Scenariet består av to nettverk, et internt og et eksternt. Det interne nettverket inneholder fire servere med rollene domenekontroller, SQL-server, FTP og SMTP/POP3. På utsiden i det eksterne nettet, står en web-server, en arbeidsmaskin og en *Predator*. Predatoren er en server utstyrt med verktøy for å angripe og finne svakheter i infrastrukturen. Intensjonen med dette er å kunne sikkerhetsteste en infrastrukturen slik at potensielle sårbarheter kan oppdages i et testmiljø. Figur 5.10 viser nettverkstopologien for det utrullede scenariet slik det presenteres i web-grensesnittet Horizon.



Figur 5.10: Nettverkstopologi Jungle scenario

Dette oppsettet kan videre tilpasses etter behov hvor det kan introduseres nye instanser, nettverk, et cetera. Løsningen vil også kunne huse andre scenarier til bruk i utdanningssammenhenger og annen eksperimentelt arbeid. Bash-skriptet benyttet til utrulling av scenario er tilgjengleig som vedlegg scenario.sh(s.113).

Kapittel 6

Testing

I et prosjekt som dette, hvor det utvikles en løsning innenfor en kompleks skyløsning, vil testing være en essensiell komponent helt fra planleggingsfasen og frem til at løsningen fungerer som ønskelig. Fasen med den manuelle installasjonen av Openstack var kanskje den delen med mest kunnskapstilegnelse, og gruppen fikk her verdifull forståelse av de ulike komponentenes funksjonalitet, noe som var en fordel for testingen i automatiseringsfasen. I forkant av dette prosjektet var det utfordrende å forutse hvilke problemområder som kunne oppstå, da gruppens medlemmer var ukjente med kompleksiteten rundt Openstack fra tidligere. Signalene gitt fra oppdragsgiver tydet på at Cobbler og Puppet ville være forholdsvis greie å implementere, mens Openstack var den ukjente faktoren. Dette viste seg for så vidt å stemme, da testing og feilsøking her har vært meget tidkrevende og tålmodighetsprøvende. I dette kapittelet vil det fokuseres på testing i forbindelse med oppsett av Cobbler, Puppet og utrulling av skyløsningen.

6.1 Metodikk

Målet med testing underveis i oppgaven er kvalitetsikring av konfigurasjonen som rulles ut. Dette innebærer å regelmessig utføre en utrulling av løsningen frem til det punktet der siste endring ble gjort, observere resultatet, gjøre eventuelle tilpasninger for nodekonfigurasjonen, og videre utvikle løsningen for å komme nærmere sluttproduktet. Ved å teste løsningen på denne måten, kan de aktuelle komponentene systemet bygger på verifiseres, frem til det punktet utviklingen er kommet.

6.2 Cobbler

Arbeidet relatert til testing av Cobbler, omhandlet tilpassing av konfigurasjonfiler for å kunne ha muligheten til å rulle ut operativsystemer automatisk. Under testing av utrulling, stoppet den automatiske installasjonen på ulike punkter, og gikk deretter over til å bli en manuell installasjon. Dette skyltdes flere ting, eksempelvis mangel på Internettforbindelse for installasjonsserverne under utrullingen. Dette problemet ble oppdaget ved å analysere nettverkstrafikken mellom Newman og en server under utrulling, hvor det kommer frem at serveren prøver å hente ned informasjon fra adresser på Internett. Her ble løsningen å sette opp Newman som en ruter for serverne. Utrullingen stoppet også av at svarfilen som Cobbler benytter under installasjonen, ikke var riktig konfigurert til å inneholde svar på alle nødvendige punkter under installasjonen.

6.3 Puppet

For å klargjøre installasjonsserverne til konfigurasjonsutrulling, ble den innebygde Cobbler-funksjonen for å installere Puppet, testet. Installasjon av denne funksjonen er iht. Cobbler en enkel oppgave, hvor følgende parameter ble satt i konfigurasjonsfilen til Cobbler;

/etc/cobbler/settings:

```
↪ puppet_auto_setup: 1
↪ sign_puppet_certs_automatically: 1
↪ puppet_server: 'newman'
↪ puppet_version: 3
```

Dette oppsettet skal også gjøre at Newman autosignerer innkommende sertifikatforespørsler (som er måten Puppet autentiserer maskiner på) fra Puppet-agenter. Da serverne etter endt operativsystemutrulling ikke var satt opp med Puppet, og Newman ikke hadde noen loggfiler over hva som eventuelt hadde gått galt, ble det nødvendig å komme opp med en alternativ løsning på problemet. Her ble et bash-skript innført, som lastes over til serverne under installasjonen, og eksekverer alle nødvendig kommandoer for å klargjøre serveren for konfigurasjonsutrulling.

Testing av Puppet omhandler først og fremst arbeidet med å sjekke at miljøet er korrekt satt opp, dvs. at nodene har en forbindelse med Puppet-master, og at nodene får hentet ned sine respektive nodekonfigurasjoner. For å verifisere dette, kan et lite program som eksempelvis Vim legges til i nodekonfigurasjonen. Hvis Vim blir installert, kan det bekrefte at konfigurasjonen er riktig satt opp for noden.

Utrulling av nodekonfigurasjon med Puppet kan skje på flere måter. Nodene kan hente ned konfigurasjoner fra bestemte Puppet-miljø, avhengig av hva som ønskes testet. Det er også mulig i Puppet å kjøre en ikke-operasjonell utrulling av en konfigurasjon. En ikke-operasjonell utrulling med Puppet viser hvordan en node ville ha blitt konfigurert med den nåværende nodekonfigurasjonen, i form av en beskrivende utskrift til skjerm. Til å begynne med benyttet gruppen en ikke-operasjonell tilnærming til utrulling av Openstack-komponenteter. Da en slik utrulling er relativt omfattende, ble det veldig mye utskrift å analysere, noe som fort ble uoversiktlig og gjorde det vanskelig å verifisere konfigurasjonen.

6.4 Manuell installasjon av Openstack

Under den manuelle installasjonen av Openstack, hvor intensjonen var å bli kjent med Openstack-komponentene oppgaven består av, jobbet gruppen parallelt. Her jobbet en del på de fysiske serverne tildelt i forbindelse med oppgaven, mens en annen del jobbet lokalt i et virtualisert miljø. Dette tillot gruppen å jobbe mer effektivt ved at hvert medlem hver for seg kunne få forståelse av Openstack-komponentene. Denne arbeidsmetoden ble benyttet i mindre grad under utviklingen av den automatiserte installasjonen av Openstack. Det var i denne fasen viktigere for gruppen å oppnå enighet rundt implementering av de enkelte komponentene.

For å installere Openstack, ble det hovedsakelig benyttet en guide som beskriver installasjon av *Folsom*[15]-utgivelsen av Openstack. Det ble også benyttet en guide[16] som baserer seg på en installasjon oppdragsgiver har gjort av Openstack, skrevet av oppdragsgiver selv. Da denne guiden ikke var dokumentert tilstrekkelig, ble svært få deler av den benyttet. Når Openstack var installert iht. Folsom guiden, møtte gruppen problemer med at instanser som ble opprettet i skyen, ikke fikk tildelt IP-adresse automatisk. For å feilsøke problemet, ble det satt opp pakkesniffing på nettverksnoden og computenoden for å se om DHCP-trafikk ble utvekslet. Samtidig som nettverkstrafikk ble analysert, ble også loggfiler tilhørende Openstack på de aktuelle nodene undersøkt. Loggfilene ga i noen grad feilmeldinger som kunne relateres til problemet, hvor disse feilmeldingene ble søkt etter løsninger for på Internett. Ved å teste flere forslag til løsninger på problemet, ble konfigurasjonen på serverne over tid et resultat som ikke lenger kunne sammenliknes med installasjonsguiden. Det ble med dette, opptil flere ganger, nødvendig å starte helt på nytt med Openstack-installasjonen for å komme tilbake til en konfigurasjon som ikke var preget av mye testing. Dette var bare en fordel, ettersom gruppen ble mer kjent med installasjon av de aktuelle Openstack-komponentene.

6.5 Automatisk installasjon av Openstack

Ved å gå bort fra ikke-operasjonell utrulling med Puppet, ble Openstack-komponentene testet ved å bli rullet ut på normalt vis. Dette gjorde det enkelt å verifisere at komponentene ble installert med riktig konfigurasjon. Hver utrullingstest fører ofte til konfigurasjonsendring på noden, som er ønskelig i de fleste tilfeller, men over tid vil nodens konfigurasjon være et resultat av tester som har gått bra og tester som ikke har gått bra. Det var derfor ofte nødvendig å finne ut hvordan nodekonfigurasjonen egentlig var og hvordan det utrullede systemet var i forhold til installasjonguiden som ble fulgt. Basert på det tidligere arbeidet i oppgaven, trengtes det kun å restarte de aktuelle serverne og boote fra nettverket og, i løpet av 20 - 60 minutter (avhengig av hvilken server som ble installert), være tilbake på en konfigurasjon kun bestående av det som ble definert i site.pp.

I serverskapet, hvor all maskinvare benyttet i denne løsningen står, finnes det også annen maskinvare benyttet i bacheloroppgaven *Backup Storage Recovery Manager (BSRMGR)*. Bachelorgruppen bak BSRMGR har ved flere anledninger hatt behov for å formatere serverne deres og installere operativsystem på nytt. De har derfor benyttet arbeidet til AutoStack vedrørende utrulling av operativsystem til å raskt få serverne tilbake til en ønsket tilstand. Slik løsningen i AutoStack er satt opp, vil Newman oppdage disse serverne som ny maskinvare til skyløsningen og, uten intensjon, implementere serverne til BSRMGR som computenoder i løsningen. Så lenge Newman er tilkoblet nettet og kontrollernode, samt nettverksnode er rullet ut, vil alle etterfølgende servere som kobler seg til få rollen som computenoder.

Likt som for den manuelle installasjonen av Openstack, møtte gruppen problemer med nettverket i Openstack. Da det ble fulgt en annen installasjonsguide enn den brukt under den manuelle installasjonen, var det andre problemer rundt nettverket som oppstod. Problemet her var hovedsakelig at instanser ikke hadde tilgang ut mot Internett, hvor NAT-nettverket i Openstack ikke fungerte etter intensjon. Switchen som ble benyttet, ble levert gruppen ferdig konfigurert med de VLAN som trengtes i løsningen, som var i tråd med skolenettet switchen skulle inn i. Det ble fra dette tidspunktet antatt at det tildelte nettverkspunktet switchen fikk fra IT-tjenesten, var en trunk-port. Da det tildelte punktet ikke var en trunk-port, ble konfigurasjon av enkelte VLAN på switchen irrelevant. Da dette ble oppdaget, ble nettverkskortet tilknyttet det eksterne nettverket på nettverksnoden, koblet til riktig nettverk. Dette løste problemene instansene hadde med mangel på tilgang ut mot Internett.

6.6 Overvåkning

Implementering av overvåkning ble sett på som en underordnet komponent i dette prosjektet, og dermed avsatt lite ressurser av gruppen. Under implementering av Munin oppstod det først og fremst utfordringer i forbindelse med dynamisk tilegning av noder i konfigurasjonsfilen til Munin, munin.conf.

Det var da nødvendig for gruppen å finne en metode for å innhente den nødvendige informasjonen fra nodene i nettverket, og legge denne til med riktige verdier i konfigurasjonsfilen. Dette var ikke noe en Puppet-modul ville kunne tilby, så en løsning med skript ble sett på som den mest fornuftige idéen. Det ble vurdert ulike løsninger for dette, før det ble konkludert med at Nmap (Network Mapper) hadde de nødvendige egenskapene for prosessen med innhenting av nodeverdier. Utfordringen herfra bestod i å bygge et skript rundt en Nmap-kommando som ville løse denne problemstillingen.

Gruppen påbegynte skriptingprosessen med å søke etter ulike eksisterende skript, og fant et med mange av de ønskede egenskapene. Det ble herfra forsøkt ulike tilpasninger til gruppens scenario, før det til slutt ble konkludert med at en egen løsning fra bunnen av ville være det beste alternativet. Fra dette punktet ble resten av tiden lagt i å produsere et tilfredsstillende skript, noe som krevde store ressurser fra gruppens side. Et godt og velfungerende skript ble generert før den avsatte tiden gikk ut, og overvåkning med Munin fungerte som ønsket.

6.7 Maskinvare

Av de tildelte serverne har det oppstått problemer med diverse maskinvarekomponenter. Serveren *Kramer* fikk aldri delta i løsningen da den hadde en defekt raidkontroller, samtidig som harddisker ikke ble detektert under utrulling av operativsystem, noe som forhindret installasjonen. Serveren *Nostrand* deltok som en computenode under store deler av utviklingen, men begynte å oppføre seg rart når serveren ikke lenger mottok DHCP fra Newman under utrulling av operativsystem. Dette skyldtes at det integrerte nettverkskortet på serveren var defekt. Det ble satt inn flere nettverkskort i Nostrand slik at den fortsatt kunne bli brukt som en computenode. Det skulle imidlertid vise seg at det kun var integrerte nettverkskort som tillot booting fra nettverk på denne serveren, og derfor hadde installasjon av nye eksterne nettverkskort ingen funksjon. Nostrand ble med dette tatt ut av løsningen. Med to servere ute av løsningen, var det kun fire servere igjen (inkludert Newman), som er det absolutte minimum antall servere som trengs. Det var ønskelig med minst to computenoder, og derfor ble en ny server, *Bookman*, introdusert til løsningen.

Kapittel 7

Diskusjon

7.1 Tilgjengelig dokumentasjon

Siden denne oppgaven tar for seg en komplisert automatiseringsprosess som gjerne må tilpasses et gitt scenario, var det fra begynnelsen av klart at det ville oppstå utfordringer med graden av tilgjengelig dokumentasjon. Verktøyene Puppet og Cobbler hadde dog en tilfredsstillende mengde informasjon som gruppen kunne utnytte. Puppet har dessuten på sine nettsider tilgjengelig moduler for Openstack, som ga et glimrende utgangspunkt for automatisering av løsningen. Likevel følger det en rekke komplikasjoner med dette scenariet, og det er her brorparten av oppgaven ligger.

7.2 Komplikasjoner ved automatisering

En automatisert utrulling av Openstack kan i stor grad oppnås ved hjelp av tilgjengelige moduler innenfor Puppet, som utfører dette på en oversiktlig og plattformuavhengig måte. Likevel er det enkelte problemområder som ikke dekkes, samtidig som spesialtilpasninger til det enkelte scenariet også må tas hensyn til. Det faktum at gruppen har benyttet seg av bash-skript for å få på plass disse elementene, vil av mange bli sett på som en uryddig funksjonalitet som helst bør unngås. Valg av bash-skript går på bekostning av løsningens robusthet ved at den ikke blir plattformuavhengig. I praksis vil dette si at en Puppet-modul vil kunne håndtere utrulling av løsningen på flere ulike type platformer, mens et bash-skript kun vil være gjeldende for den spesifikke platformen skriptet er utviklet til. Utvikling av Puppet-moduler innebærer et langt mer tidkrevende arbeid sammenliknet med utvikling av et bash-skript. Gruppen prioriterte med dette å bruke bash-skript fremfor egenkomponerte Puppet-moduler for å komme i mål med utrulling av en fungerende løsning.

Arbeidet med å gjøre utrulling av løsningen plattformuavhengig, innebærer å oppnå samme konfigurasjon deploy-skriptet står for, men på flere platformer. Hvordan programvare installeres og hvor program plasseres i filstrukturen, varierer fra plattform til plattform. Ut i fra hvilken plattform en Puppet-agent kjører på, kan et spesifikt sett med operasjoner tilpasses den aktuelle plattformen. Ved installasjon av eksempelvis Openstack-komponenten, Keystone, på Debian plattform, er navnet på installasjonspakken *keystone* og installeres med *Advanced Packagin Tool* (apt). På RedHat plattform heter installasjonspakken *openstack-keystone* og installeres med *Yellowdog Updater, Modified* (yum). Det er slike ulikheter det må tas hensyn til for å oppnå en plattformuavhengig utrulling av løsningen.

7.3 Bruk av åpen kildekode

Openstack er en løsning basert på åpen kildekode, som gjerne fører med seg en rekke fordeler og ulemper. En erfaring gruppen har gjort seg er at det ved innhenting av data eksisterer veldig mange moduler det er mulig å benytte seg av, og at det derfor kan være utfordrende å finne den beste løsningen for dette scenariet. Likevel er fordelen ved at man slipper lisensiering og sparer potensielt store kostnader, god nok begrunnelse til å benytte seg av en slik løsning.

7.4 Inspirasjon og kilder

I startfasen av dette prosjektet var det aktuelt å benytte seg av installasjonsguider i forbindelse med det manuelle oppsettet av Openstack, slik at gruppen kunne oppnå bedre forståelse av skyløsningen som er grunnlaget for oppgaven. I ettertid viste det seg at med de guidene som ble benyttet i startfasen, etter anbefaling fra oppdragsgiver, hadde en del feil og dårlig dokumenterte løsninger. Etter hvert gikk gruppen over til en nyere Havana-guide[19] fra Openstack sine hjemmesider, som fungerte mye bedre. Det er likevel blitt lagt merke til at de fleste tilgjengelige kildene for installasjon av Openstack er for dårlig dokumenterte til å kunne anbefales, noe som er tydeliggjort i tilbakemeldingene fra brukere i kommentarfeltene under de nevnte guidene.

7.5 Videreutvikling og robusthet

Siden denne oppgaven er såpass omfattende å løse i seg selv, går dette på bekostning av potensiell tilleggsfunksjonalitet, og ikke minst robusthet. Det blir lagt vekt på at løsningen skal kunne rulles ut etter de gitte premissene uten at det vil oppstå problemer, men det vil alltid være usikkerhet knyttet til denne løsningen da det foreløpig ikke er blitt lagt ned nok ressurser i det. Robusthet og tilleggsfunksjonalitet er punkter gruppen mener det vil være interessant å utforske dersom det blir aktuelt å videreutvikle løsningen.

Det ble på et relativt tidlig tidspunkt anbefalt av oppdragsgiver å rulle ut operativsystemet Ubuntu Server 13.10 med Cobbler. Dette vil si at videre utvikling av løsningen hadde fokus på å støtte denne plattformen, noe som forsvarer bruken av bash-skript til å fullføre de deler av konfigurasjonen som ikke lot seg gjøre på en enkel måte med Puppet. Gruppen prioriterte å utvikle en fungerende løsning til Ubuntu-plattformen, fremfor en halvferdig løsning som ville vært plattformuavhengig. Gruppen ser fortsatt at plattformspesifikke skript bør avskaffes ved en eventuell videreutvikling av oppgaven, da skriptene ikke støtter en idempotent konfigurasjonsutrulling på samme måte som Puppet.

7.6 Interessenter

Verktøyene som ble benyttet for denne oppgaven, henholdsvis Openstack, Puppet, Cobbler og Munin, var alle definert fra oppdragsgivers side enten i form av oppgavebeskrivelsen eller muntlig anbefaling. Det har derfor vært få diskusjoner fra gruppens side rundt alternativer som kunne passet bedre inn, eller fungert bedre i samspill med hverandre. Puppet har fungert som et meget godt alternativ, da dette verktøyet tilbyr moduler til bruk for Openstack, noe gruppen også har benyttet seg av, og som antagelig har ført til en brukbar tidsbesparelse. Cobbler for utrulling av operativsystem har også utført jobben som var påkrevd. Munin er et verktøy gruppen hadde god kjennskap til fra tidligere, og er et komfortabelt alternativ for implementering av overvåkning. For eventuelle interessenter kan det spekuleres i hvor stor betydning valgene av disse verktøyene har. En begrunnelse er at alle disse er populære løsninger med mye innebygd funksjonalitet, samtidig som de baserer seg på åpen kildekode. Disse attributtene bør ha stor drakraft for å tiltrekke seg en størst mulig kundegruppe.

7.7 Utrulling av skyen

Gjennom denne prosjektperioden har det vært flere diskusjoner rundt scenarier for den automatiserte utrullingen av skyløsningen. Som nevnt i introduksjonskapittelet, er det vist interesse for lettvinte metoder for å bedrive IT-teknisk eksperimentering, og denne løsningen vil være grunnlaget for et effektivt hjelpemiddel i den sammenheng. Fordi gruppen etter endt prosjektperiode i utgangspunktet ikke vil arbeide videre med denne løsningen, genereres en brukerdokumentasjon for løsningen slik at den skal kunne benyttes av andre fagpersoner med tilstrekkelig kunnskap uten at det skal være nødvendig med fordypning i denne rapporten.

7.8 Utrulling av overvåkning

Implementering av overvåkning i dette prosjektet bestod i oppgavebeskrivelsen av verktøyene Munin og Nagios. Disse skulle inkluderes som en del av den automatiske utrullingen. Gruppen bestemte seg tidlig for at hovedfokus for oppgaven skulle være å komme i mål med automatisk utrulling av Openstack, og dermed ble det ikke satt fokus på overvåkningsløsningene før i sluttfasen. Med tanke på hvor ressurskrevende hoveddelen av oppgaven var, ble det svært lite tid til implementering av overvåkning. Under automatiseringen av Munin-implementeringen, fant gruppen tidlig ut at det ikke ville bli tilstrekkelig tid til Nagios. En optimal løsning ville imidlertid innebære Nagios som en komponent, med tanke på hvordan dette verktøyet utfyller områder på real-time overvåkning som Munin ikke dekker spesielt godt. Med bakgrunn i at det ikke eksisterte noen passende moduler for dynamisk tilegning av noder til Munin-mesteren, lå arbeidet her i å utarbeide et skript som utførte dette. Gruppen er fornøyd med løsningen som ble produsert, selv om det såfremt det er mulig bør benyttes eksisterende Puppet-moduler. Dette gjelder forsåvidt for hele prosjektet, og ville på generell basis skapt et ryddigere resultat.

Kapittel 8

Konklusjon

8.1 Resultat

Resultatet av denne oppgaven er en fullt funksjonell sky-løsning (Openstack) som er blitt implementert på en gruppe servere gjennom en automatiseringsprosess. Automatiseringen bygger på verktøyene Cobbler og Puppet som er installert på hovedserveren, Newman. Denne serveren står for utrullingen av Openstack med dets avhengigheter, samt overvåkning av serverne som deltar i løsningen.

Løsningen kan benyttes i flere sammenhenger. Servere som er tatt ut av drift, av en eller annen årsak, kan nå gjenbrukes og tas inn i en skyløsning hvor test-miljøer kan etableres, eller andre eksperimentelle scenarier kan opprettes. Løsningen kan også benyttes der det er behov for å kunne tilby maskinkraft, eksempelvis i en utdanningssammenheng hvor enkelte emner eller studentoppgaver trenger en eller flere servere for å utføre tester eller annen eksperimentell virksomhet.

Det faktum at ny maskinvare kan legges til dynamisk, noe som for øvrig vil være aktuelt å benytte seg av i mange scenarier, gjør dette til en meget fleksibel løsning som vil kunne spare brukerne for unødvendig tids- og ressursbruk. Det er altså mulig å først rulle ut løsningen etter ønsket oppsett, for deretter å koble til nye fysiske noder i etterkant som automatisk får roller som computenoder i løsningen. Dette forutsetter at Newman fortsatt er koblet til nettverket.

Som en understilt komponent av Openstack-løsningen, var det ønskelig fra oppdragsgiver med implementering av overvåkning i form av Munin og Nagios. Dette har resultert i en automatisering av førstnevnte komponent, med dets Mester/Node-arkitektur. Bruk av Puppet og skript for tilpasning med dynamisk tilegning av nye noder, gjør dette til en fullt fungerende overvåkningsløsning som en del av denne oppgaven. Det ble foretatt en vurdering før gruppen begynte på overvåkningsdelen av prosjektet, hvor gruppen ble enige om at det kun ville være tilstrekkelig tid til å implementere ett av verktøyene. Dermed ble Munin valgt på bakgrunn av gruppens kjennskap og trygghet til den.

8.2 Læringsutbytte

Dette prosjektet har bestått av ukjente verktøy, aller mest Openstack, som har stilt store krav til effektiv kunnskapstilegning og forståelse for å kunne oppnå en tilfredsstillende løsning. Av andre verktøy som ble benyttet i løpet av prosjektperioden, og som det var nødvendig å sette oss godt inn i, er Cobbler, Puppet, ShareLaTeX og Git. Det har vært en meget krevende periode med svært mange små problemområder som har bremset opp for utviklingen underveis. Samtidig har gruppen sørget for å fordele de aktuelle gjøremålene og oppnådd god ekspertise på hovedområdene for oppgaven, Openstack og automatisering ved hjelp av Puppet. Videre har den oppnådde kunnskapen ved å skrive en større prosjektoppgave vært til nytte for gruppen. Det har gitt erfaring om prosjektplanlegging, vurderinger som må tas underveis, tilpasninger, bruk av nødløsninger, veiledning samt rapportskriving.

8.3 Forbedringer

Gruppen ser på oppgaven som svært omfattende, hvor det har vært behov for å tilegne seg mye ny kunnskap. Denne tilegningen av kunnskap kunne allerede ha startet fra det tidspunktet oppgaven ble tildelt gruppen. Selv om det ble lagt noe arbeid i oppsett av Cobbler før oppgavens start i januar, kunne gruppen hatt bedre utbytte av oppgaven hvis dette arbeidet hadde startet tidligere.

Det ble brukt mye tid på å følge forskjellige installasjonsguider for Openstack. En fordel ville vært å forholdt seg til kun én guide, og helst da guiden som ble benyttet under den automatiserte installasjonen av Openstack[19], som har ført frem til det produktet gruppen sitter igjen med.

For å effektivisere arbeidet rundt testing av løsningen underveis i utviklingen, kunne virtualisering vært et alternativ. Ved å benytte de fordelene eksempelvis snapshots har, kan testutrulling av løsningen starte fra en bestemt servertilstand. Dette ville kortet ned tiden det tar å teste en ny nodekonfigurasjon betraktelig, da utrullingen av operativsystem kunne blitt sløyfet. Virtualiseringsverktøyet, *VirtualBox*, innehar de nødvendig egenskapene for å støtte en slik form for testing, samtidig som verktøyet er Open-source og tilgjengelig på flere plattformer. Et annet alternativ til virtualisering for å effektivt utføre tester, er *Vagrant*, som også er Open-source, og støtter bruk av Puppet-manifester til testing av nodekonfigurasjon.

8.4 Videreutvikling

Hadde løsningen vært mer robust, ville den også kunne blitt benyttet på flere typer plattformer. Første steg for å gjøre løsningen mer robust er å opprette Puppet-moduler (evt. utvide eksisterende moduler) for å erstatte konfigurasjonen deploy-skriptene står for. En slik videreutvikling vil også stille nye krav til installasjonen av Cobbler, hvor det må tilbys flere alternativer til ulike operativsystem som kan rulles ut. I denne sammenheng vil det også være aktuelt å vurdere utrulling av den niende (og nyeste) Openstack-utgivelsen, Icehouse, på de nyeste operativsystemutgivelsene, eksempelvis Ubuntu Server 14.04.

Det ville også vært høyst aktuelt å gjøre hele automatiseringsprosessen tilgjengelig fra Internett, slik at en Openstack-installasjon kan planlegges og rulles ut fra en ekstern lokasjon.

Et annet aspekt, som vil være interessant å se nærmere på, er alternativer til Cobbler og Puppet. Mange av de aktuelle brukerne av en slik løsning vil ha ulike ønsker og forutsetninger for hvilke verktøy de ønsker å benytte i sin bedrift, og dermed vil man ved å tilby flere alternativer kunne nå ut til et bredere publikum.

Ettersom løsningen ikke inneholder mulighet for overvåking med Nagios, er det aktuelt å implementere dette ved en videreutvikling. Eksisterende løsning med Munin tilbyr overvåkning med hovedfokus på tendensmålinger, og Nagios vil bidra til å gjøre overvåkningen mer komplett. Det er i løsningen brukt skript for å få Munin til å operere etter ønsket funksjonalitet. Dette er ikke en optimal løsning, og det vil derfor være ideelt å generere egne Puppet-moduler som erstatning.

8.5 Egenevaluering

AutoStack som gruppe er veldig fornøyd med den tildelte oppgaven og finner den veldig relevant både for generelle interesser og som en god erfaring til arbeidslivet. Store deler av arbeidet har foregått i Linux-server terminal, noe som føltes uvant i begynnelsen, men veldig naturlig etter hvert som løsningen utviklet seg.

I henhold til prosjektplanen har gruppen forholdt seg til en gruppeleder, med flytende rollefordeling under arbeidet. Det ble tidlig klargjort hvilke styrker de ulike medlemmene hadde innenfor de ulike verktøyene og metodene i bruk, og det ble under hele prosessen fokusert på å utnytte disse best mulig. Veileder har også blitt benyttet aktivt, ved å ha ukentlige møter for kartlegging av fremdrift og planlegging av videre arbeid.

I løpet av prosjektperioden har det dukket opp en del tekniske spørsmål om Openstack og om nettverk. Disse spørsmålene har vi fått besvart av vår oppdragsgiver Eirik Hjelmås, som har erfaring fra tidligere bacheloroppgaver om Openstack, samt HiGs SkyHiGh. Vi har ved problemer underveis også fått konfigurasjonsfiler og spørringer gjort i SkyHiGh for å sammenligne våre filer/spørringer med deres Openstack-løsning. Dette har vært til stor nytte for å finne manglende og/eller feil konfigurasjon under uviklingen.

Bibliografi

[1] Christian Meyer.
CCIS - Center for Cyber- and Information Security Tilgjengelig fra:
https://norsis.no/2013/06/center-for-cyber-and-information-security 2

[2] Mikael Wedlin, Teodor Sommestad.
CRATE - Cyber Range And Training Environment.
Tilgjengelig fra: http://www.foi.se/en/Our-Knowledge/Information-Security-and-Communication/Information-Security/Lab-resources/CRATE/ 2

[3] OpenStack:Project Info Tilgjengelig fra:
http://docwiki.cisco.com/wiki/OpenStack:Project_Info 2

[4] Biblioteket ved Høgskolen i Gjøvik - Vancouver Siteringsstil
Tilgjengelig fra: www.hig.no/biblioteket/oppgaveskriving/vancouver 2012

[5] Sean Michael Kerner
OpenStack Wins the Open Source Cloud
Tilgjengelig fra: http://www.internetnews.com/blog/skerner/openstack-wins-the-open-source-cloud.html 2012 17

[6] Database- og applikasjonsdrift
Tilgjengelig fra: http://www.hig.no/studiehaandbok/studiehaandboeker/2013_2014/emner/avdeling_for_informatikk_og_medieteknikk/imt3441_database_og_applikasjonsdrift 17

[7] Matt Weinberger
Three Open Source-Based Cloud Alternatives to OpenStack
Tilgjengelig fra: http://siliconangle.com/blog/2012/04/24/three-open-source-based-cloud-alternatives-to-openstack/ 17

[8] Gaute Borgenholt, Trond Isak Alseth, Otto Trodal Øksnevad
Jungle
24

[9] Cobbler Preconfiguration File
Tilgjengelig fra:
https://help.ubuntu.com/10.04/installation-guide/example-preseed.txt

[10] Mandar Shinde
How to configure Ubuntu as a router
Tilgjengelig fra: http://www.yourownlinux.com/2013/07/how-to-configure-ubuntu-as-router.html

[11] Riccardo Magrini
UBUNTU SERVER:"HOW TO ENABLE IP FORWARDING IN UBUNTU 12.04LTS SERVER EDITION"
Tilgjengelig fra: http://ideasnet.wordpress.com/2013/05/29/ubuntu-server-how-to-enable-ip-forwarding-in-ubuntu-12-04lts-\server-edition

[12] Installing Puppet: Pre-Install Tasks
Tilgjengelig fra: http://docs.puppetlabs.com/guides/installation.html

[13] 10 Ways to Generate a Random Password from the Command Line
Tilgjengelig fra: http://www.howtogeek.com/howto/30184/10-ways-to-generate-a-random-password-from-the-command-line/

[14] CFengine Tilgjengelig fra: https://cfengine.com/ 18

[15] Roy Sowa, Stephen Gran, Dennis E. Miyoshi, Marco Consonni, Houssern Medhioub, Djamal Zeghlache
OpenStack Folsom Install Guide
Tilgjengelig fra:
https://github.com/mseknibilel/OpenStack-Folsom-Install-guide 34, 42, 66

[16] Erik Hjelmås
SkyHiGh
2014 66

[17] Spencer Krum, William Van Hevelingen, Ben Kero, James Turnbull, Jeffrey McCune
Pro Puppet
2nd Edition
2011 53, 58

[18] Verify the Identity Service installation Tilgjengelig fra: http://docs.openstack.org/havana/install-guide/install/apt/content/keystone-verify.html 53

[19] Havana Openstack installation guide for Ubuntu 12.04LTS Tilgjengelig fra: docs.openstack.org/havana/install-guide/install/apt/content 54, 71, 76

[20] Cobbler - Linux install and update server [Internett].
Ingen dato [sitert 5 mars 2014].
Tilgjengelig fra: http://www.cobblerd.org 84

[21] Wikipedia-brukere; Wikipedia - The Free Encyclopedi: Puppet (software) [Internett].
c2014 [Oppdatert 1. mars 2014 19:00 UTC, sitert 5 mars 2014 09:20 UTC]
Tilgjengelig fra: http://en.wikipedia.org/w/index.php?title=Puppet_(software)&oldid=597703130 84

[22] Wikipedia-brukere; Wikipedia - The Free Encyclopedi: RAID [Internett].
c2014 [Oppdatert 9 september 2013 11:52 UTC , sitert 5 mars 2014 09:52 UTC]
Tilgjengelig fra: no.wikipedia.org/w/index.php?title=RAID&oldid=12646465 84

[23] Dan Wendland
OpenStack Quantum Intro (OS Meetup 3-26-12)
Tilgjengelig fra:
www.slideshare.net/danwent/openstack-quantum-intro-os-meetup-32612

[24] Lars Erik Pedersen, Jon Arne Westgaard, Hallvard Westman.
SkyHiGh ADM
Tilgjengelig fra: http://brage.bibsys.no/hig/bitstream/URN:NBN:no-bibsys_brage_30474/3/LEPedersen_JAWWestgaard_HAWestman.pdf

[25] Tom Fifield
Introduction to OpenStack
Tilgjengelig fra:
http://www.linuxjournal.com/content/introduction-openstack

[26] Nova Image-list error
Tilgjengelig fra: http://stackoverflow.com/questions/19740109/nova-image-listerror-unauthorized-http-401

[27] Glance's Documentation
Tilgjengelig fra: docs.openstack.org/developer/glance/#concepts

[28] Puppet - Server orchestration.
Ingen dato [sitert 6 mars 2014].
Tilgjengelig fra: http://puppetlabs.com

[29] Scott Chacon
Pro Git
Tilgjengelig fra: https://github.s3.amazonaws.com/media/progit.en.pdf

[30] Wikipedia contributors, Wikipedia, The Free Encyclopedia: Munin (software)
c2014 [Oppdatert 21 Mars 2014 21:44 UTC , sitert 3 April 2014 13:02 UTC]
Tilgjengelig fra: http://en.wikipedia.org/w/index.php?title=Munin_(software)&oldid=600654436

[31] Wikipedia contributors, Wikipedia, The Free Encyclopedia: Nagios
c2014 [Oppdatert 01 April 2014 22:22 UTC , sitert 3 April 2014 13:45 UTC]
Tilgjengelig fra:
http://en.wikipedia.org/w/index.php?title=Nagios&oldid=602330184

[32] Puppet Konfigurasjonsverktøy
Tilgjengelig fra: http://www.visualisere.no/main/diverse/puppet-konfigurasjonsverktoy.html

[33] Nagios - About
Tilgjengelig fra: http://www.nagios.org/about

Tillegg A

Terminologi

- Cobbler: En Linux installasjonsserver som gjør det mulig å raskt sette opp nettverksinstallasjoner, og automatiserer mesteparten av prosessen. [20]
- Puppet-(master/agent): Open-source konfigurasjonsstyringsverktøy fra PuppetLabs. [21]
- Operativsystem: Er den grunnleggende programvaren som styrer og tildeler ressurser for andre programmer og tjenester på maskinen.
- Linux: Et Open-source operativsystem.
- Ubuntu: Et gratis Linux basert operativsystem.
- RAID: Redundant Array of Inexpensive Disks (redundant matrise av billige disker), gjør at man kan få økt ytelse og/eller redundans på datalageret ved bruk av fler enn en harddisk. [22]
- Advanced Packaging Tool (APT): Pakkebehandlingsverktøy for Linux. Håndterer nedlastning, konfigurasjon og installasjon av programvare.
- ISO-fil: Er en fil med et bilde av en CD eller DVD.
- Image: En diskfil med et installert virtuell maskin.
- Preboot Execution Environment (PXE): booting av datamaskiner ved hjelp av nettverksgrensesnitt.
- Preseed-fil: En fil som inneholder de svarene som blir forespurt under installasjon av et operativsystem
- Network Address Translation (NAT): En måte å endre IP-adresser på ettersom de passerer en ruter eller brannmur.
- IP-forwarding: En prosess for å fastslå hvilken rute en IP-pakke skal videresendes.
- Kernel: Programmet som håndterer I/O-forespørsler fra programvare, og oversetter de til instruksjoner til en datamaskins fysiske komponenter.
- Host fil: Fil for å koble IP-adresser opp mot en node.
- Transmission Control Protocol (TCP): Nettverksprotokoll for overføring av informasjon.
- Sertifikat: En lisens som opplyser om gitte tillatelser.
- Eksekverbar fil: En fil som inneholder et program med instruksjoner til utførelse.

- Media Access Control (MAC)-adresse: En unik identifikator for utstyr som kan kobles til et nettverk. Også kjent som en enhets fysiske adresse
- Expose-IP: Ekstern IP-adresse, tilgjengelig ut mot Internett.
- Virtual Local Area Network (VLAN): Segmentering av fysiske nettverk opp i flere logiske.
- Vlan bridge-utils: Koble sammen flere ethernet til et virtuelt grensesnitt.
- Openstack: Et Open-source skyløsningsprosjekt
- Keystone: Identifiserings,- og autentiseringsmodulen til Openstack
- Nova: Modulen som administrerer de virtuelle datamaskinene i Openstack.
- Neutron: Modulen som håndterer nettverket i Openstack.
- Glance: Modulen som administrerer de virtuelle imagene i Openstack.
- Horizon: Modulen som tilbyr et webgrensesnitt i Openstack
- RabbitMQ (Messaging Queue): Kommunikasjonstjenesten som benyttes av Openstack.
- MySQL: SQL-basert databaseadministrasjonssystem.
- Database: Strukturert samling av relaterte data.
- Miljøvariable: Dynamisk navngitte variable som påvirker hvordan prosesser kjøres.
- Tenant: Et prosjekt i Openstack.
- Endpoint: Informasjon om hvordan vi aksesserer en tjeneste.
- Openvswitch: Virtuell, flerlags switch, lisensiert under Apache.
- Memcached: Distribuert objekt caching system.
- Kernel-based Virtual Machine (KVM): Linux kjerneinfrastruktur for å støtte virtualisering.
- Live migrering: Flytte virtuelle maskiner mellom fysisk maskinvare mens de forblir kjørende.
- Pakkesniffing: Monitorering og analysering av nettverkstrafikk.

- Application Programming Interface (API): Grensesnitt for å benytte eksisterende tjenester i egenutviklede løsninger.
- Openstack-node: En maskin som består av en eller flere Openstack-komponenter.
- Openstack-modul: Moduler tilhørende orkestreringsverktøyet Puppet for å automatisere installasjon av en eller flere Openstack-komponenter.
- Openstack-komponent: En spesifik tjeneste Openstack er bygd opp av. Eksempelvis Keystone.
- Deploy-skript: Skript for utrulling av løsning der puppetmoduler ikke strekker til
- Idempotent: En operasjon som kan utføres gjentatte ganger uten å videre endre resultatet hver gang operasjonen utføres.
- Ping-sweep: En metode for å finne IP-adresser til aktive hosts i nettverket.
- VM: En virtuell maskin.
- Orkestrering: Et begrep brukt i konfigurasjonsstyringssystemer som omhandler distribusjon av konfigurasjon i en infrastruktur.
- Snapshot: En lagret tilstand av en virtuell maskin i form at et image.

Tillegg B

Kode

openstackAuth.sh

C:\My Documents\openstackAuth.sh 12. mai 2014 12:28

```
#!/bin/bash

export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=<password>
export OS_AUTH_URL="http://10.0.0.1:5000/v2.0/"

#END
```

-1-

Preseed

C:\My Documents\preseed.txt 19. mai 2014 18:12

```
# Ubuntu Server Quick Install
# by Dustin Kirkland <kirkland@ubuntu.com>
#  * Documentation: http://bit.ly/uquick-doc

d-i     debian-installer/locale string en_US.UTF-8
d-i     debian-installer/splash boolean false
d-i     console-setup/ask_detect        boolean false
d-i     console-setup/layoutcode        string us
d-i     console-setup/variantcode       string
### NETWORK
d-i     netcfg/choose_interface select auto
d-i     netcfg/get_nameservers  string
d-i     netcfg/get_ipaddress    string
d-i     netcfg/get_netmask      string 255.255.255.0
d-i     netcfg/get_gateway      string
d-i     netcfg/confirm_static   boolean true

d-i     clock-setup/utc boolean true
### PARTITIONING
#d-i    partman-auto/disk string /dev/cciss/c0d0
d-i     partman-auto/method string regular
d-i     partman-lvm/device_remove_lvm boolean true
d-i     partman-lvm/confirm boolean true
d-i     partman/confirm_write_new_label boolean true
d-i     partman/choose_partition         select Finish partitioning and write changes to disk
d-i     partman/confirm boolean true
d-i     partman/confirm_nooverwrite boolean true
d-i     partman/default_filesystem string ext3

d-i     clock-setup/utc boolean true
d-i     clock-setup/ntp boolean true
d-i     clock-setup/ntp-server  string ntp.ubuntu.com
d-i     base-installer/kernel/image     string linux-server
d-i     passwd/root-login       boolean false
d-i     passwd/make-user        boolean true
d-i     passwd/user-fullname    string ubuntu
d-i     passwd/username string ubuntu
d-i     passwd/user-password-crypted    password
$6$.1eHH0iY$ArGzKX2YeQ3G6U.mlOO3A.NaL22Ewgz8Fi4qqz.Ns7EMKjEJRIW2Pm/TikDptZpuu7I92frytmk5YeL.9f
RY4.
d-i     passwd/user-uid string
d-i     user-setup/allow-password-weak  boolean false
d-i     user-setup/encrypt-home boolean false
d-i     passwd/user-default-groups      string adm cdrom dialout lpadmin plugdev sambashare
d-i     apt-setup/services-select       multiselect security
d-i     apt-setup/security_host string security.ubuntu.com
d-i     apt-setup/security_path string /ubuntu
d-i     debian-installer/allow_unauthenticated  string false
d-i     pkgsel/upgrade  select safe-upgrade
d-i     pkgsel/language-packs   multiselect
d-i     pkgsel/update-policy    select none
d-i     pkgsel/updatedb boolean true
d-i     grub-installer/skip     boolean false
d-i     lilo-installer/skip     boolean false
d-i     grub-installer/only_debian      boolean true
d-i     grub-installer/with_other_os    boolean true
d-i     finish-install/keep-consoles    boolean false
```

Preseed

C:\My Documents\preseed.txt **19. mai 2014 18:12**

```
d-i     finish-install/reboot_in_progress       note
d-i     cdrom-detect/eject      boolean true
d-i     debian-installer/exit/halt      boolean false
d-i     debian-installer/exit/poweroff  boolean false
d-i     pkgsel/include string openssh-server
d-i     preseed/late_command string wget http://10.0.0.254/script.sh /; cp /script.sh
/target/etc/init.d; chmod +x /target/etc/init.d/script.sh; in-target update-rc.d script.sh
defaults
```

Newman startup script

C:\My Documents\script.sh 19. mai 2014 17:51

```
#!/bin/sh

apt-get update
apt-get upgrade -y
apt-get dist-upgrade -y
apt-get install puppet -y

ROLES_ASSIGNED=0
MAC_CTRL=""
MAC_NET=""
ENVIRONMENT="production"

if [ "$ROLES_ASSIGNED" -eq 0 ]; then
  IP_OCT4=$(facter ipaddress_eth0 | cut -d "." -f 4)
  if [ "$IP_OCT4" -eq 1 ]; then
    hostname controller
    echo controller > /etc/hostname
  elif [ "$IP_OCT4" -eq 2 ]; then
    hostname network
    echo network > /etc/hostname
  else
    hostname compute$(expr $IP_OCT4 - 2)
    echo compute$(expr $IP_OCT4 - 2) > /etc/hostname
  fi
elif [ "$ROLES_ASSIGNED" -eq 1 ]; then
  #Jerry mac
  #MAC_CTRL="00:19:bb:28:d3:38"
  #bania = network
  #MAC_NET="00:25:b3:21:b9:0e"
  #this machines eth0 mac address
  MAC_LOCAL=$(facter macaddress_eth0)
  # Having enough information about the servers hardware, the different
  # Openstack roles are assign based on the servers mac-address.
  if [ "$MAC_CTRL" = "$MAC_LOCAL" ]; then
    hostname controller
    echo controller > /etc/hostname
  elif [ "$MAC_NET" = "$MAC_LOCAL" ]; then
    hostname network
    echo network > /etc/hostname
  # Those servers that didn't meet the Openstack controller/network criterias
  # is assinged as compute nodes.
  else
    IP_OCT4=$(facter ipaddress_eth0 | cut -d "." -f 4)
    hostname compute$(expr $IP_OCT4 - 2)
    echo compute$(expr $IP_OCT4 - 2) > /etc/hostname
  fi
fi

# When Puppet is installed, the nessecary configuration for the
# Puppet-agents is set so that they are able to contact the Puppet-master
echo "10.0.0.254 newman.localdomain newman" >> /etc/hosts
sed -i '2iserver=newman.localdomain' /etc/puppet/puppet.conf
echo "[agent]" >> /etc/puppet/puppet.conf
echo "environment = $ENVIRONMENT" >> /etc/puppet/puppet.conf
##############################
service puppet restart
puppet agent --enable
```

Newman startup script

C:\My Documents\script.sh 19. mai 2014 17:51

```
# Not all parameters are set in the first Puppet-run,
# therefor several runs is set up
puppet agent -t
puppet agent -t
puppet agent -t

# This script should only run once, and is therefor deleted after first boot.
rm -f /etc/init.d/script.sh
```

site.pp

C:\My Documents\site.pp 18. mai 2014 16:24

```
node controller {

  ############
  ### NTP ###
  ############

  # To synchronize services acrosss multiple machines, NTP is installed.
  # The controller fetches time data from the Internet, while
  # network and compute fetches time data from controller.

  class { '::ntp':
    servers => [ 'ntp1.corp.com', 'ntp2.corp.com' ],
  }

  ##############
  ### MYSQL ###
  ##############

  # Most of the Openstack services require a database to store information.
  # For this configuration, MySQL is the chosen database.

  class { '::mysql::server':
    config_hash => { 'root_password' => 'SRVPW',
                     'bind_address'  => '10.0.0.1' }
  }

  #################
  ### KEYSTONE ###
  #################

  # Puppetlabs-keystone module ensures that the Openstack-komponent
  # keystone is present and that the database connection for keystone is set.

  class { 'keystone':
    verbose        => True,
    catalog_type   => 'sql',
    admin_token    => 'ADMIN',
    sql_connection => 'mysql://keystone:SRVPW@10.0.0.1/keystone',
  }

  # Installs the service user endpoint.
  class { 'keystone::endpoint':
    public_address   => '10.0.0.1',
    admin_address    => '10.0.0.1',
    internal_address => '10.0.0.1',
    region           => 'regionOne',
  }

  # Creates the keystone MySQL user, database and tables.
  class { 'keystone::db::mysql':
    password      => 'SRVPW',
    allowed_hosts => '%',
    require       => Class['::mysql::server'],
  }

  # The Admin-role credentials is set.
  class { 'keystone::roles::admin':
```

site.pp


```
  email        => 'example@example.com',
    password     => 'SRVPW',
  }

  ###############
  ### GLANCE ###
  ###############

  # Configuration of the api file for Glance.
  # Ensures Glance is installed.
  class { 'glance::api':
    verbose           => true,
    keystone_tenant   => 'services',
    keystone_user     => 'glance',
    keystone_password => 'SRVPW',
    sql_connection    => 'mysql://glance:SRVPW@10.0.0.1/glance',
    auth_host         => '10.0.0.1',
    pipeline          => 'keystone',
  }

  # Configuration of the registry file for Glance.
  class { 'glance::registry':
    verbose           => true,
    keystone_tenant   => 'services',
    keystone_user     => 'glance',
    keystone_password => 'SRVPW',
    sql_connection    => 'mysql://glance:SRVPW@10.0.0.1/glance',
    auth_host         => '10.0.0.1',
  }

  # Creates the Glance MySQL user, database and tables.
  class { 'glance::db::mysql':
    password      => 'SRVPW',
    allowed_hosts => '%',
  }

  # Configures authentication for Glance to the identity service.
  class { 'glance::keystone::auth':
    password         => 'SRVPW',
    email            => 'example@example.com',
    public_address   => '10.0.0.1',
    admin_address    => '10.0.0.1',
    internal_address => '10.0.0.1',
    region           => 'regionOne',
  }

  #############
  ### NOVA ###
  #############

  # Install and configure Nova
  class { 'nova':
    sql_connection      => false,
    database_connection => 'mysql://nova:SRVPW@10.0.0.1/nova',
    rpc_backend         => 'nova.rpc.impl_kombu',
  }
```

site.pp


```
# Install and configure nova-api
class { 'nova::api':
  enabled           => true,
  auth_strategy     => 'keystone',
  admin_password    => 'SRVPW',
  auth_host         => '10.0.0.1',
  neutron_metadata_proxy_shared_secret => 'METADATA_PASS',
  auth_port         => '35357',
  auth_protocol     => 'http',
  auth_uri          => 'http://10.0.0.1:5000/v2.0',
  admin_tenant_name => 'services',
  admin_user        => 'nova',
}

# Install Nova components.
class { 'nova::rabbitmq': }
class { 'nova::scheduler':    enabled => true }
class { 'nova::cert':         enabled => true }
class { 'nova::vncproxy':     enabled => true }
class { 'nova::consoleauth':  enabled => true }
class { 'nova::conductor':    enabled => true }
$novaComponents = [ 'nova-ajax-console-proxy',
                    'nova-doc' ]
package { $novaComponents:    ensure  => 'present' }

# Creates the Nova MySQL user, database and tables.
class { 'nova::db::mysql':
  user          => 'nova',
  password      => 'SRVPW',
  dbname        => 'nova',
  allowed_hosts => '%',
}

# Configures Nova to use Neutron
class { 'nova::network::neutron':
  neutron_admin_password => 'SRVPW',
  neutron_url            => 'http://10.0.0.1:9696',
  neutron_admin_auth_url => 'http://10.0.0.1:35357/v2.0',
  neutron_region_name    => 'regionOne',
}

# Configures authentication for Nova to the identity service.
class { 'nova::keystone::auth':
  password         => 'SRVPW',
  email            => 'example@example.com',
  public_address   => '10.0.0.1',
  admin_address    => '10.0.0.1',
  internal_address => '10.0.0.1',
  region           => 'regionOne',
}

################
### Neutron ###
################

# Installs and configures the Neutron service.
class { '::neutron':
```

site.pp


```
    enabled         => true,
    bind_host       => '0.0.0.0',
    rabbit_host     => '10.0.0.1',
    rabbit_user     => 'guest',
    rabbit_password => 'SRVPW',
    rabbit_hosts    => false,
    verbose         => false,
    debug           => false,
  }

  # Configures database connection for the Neutron service.
  class { 'neutron::server':
    auth_host       => '10.0.0.1',
    auth_password   => 'SRVPW',
    sql_connection  => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
    connection     => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
   }

  # Installation and configuration of the OVS-plugin for Neutron.
  class { 'neutron::plugins::ovs':
    tenant_network_type => 'vlan',
    network_vlan_ranges => 'physnet1:2000:2099',
  }

  # Configuring the firewall_driver for Neutron agents.
  class { 'neutron::agents::ovs':
    firewall_driver =>
    'neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver',
  }

  # Creates the Neutron MySQL user, database and tables.
  class { 'neutron::db::mysql':
    password      => 'SRVPW',
    allowed_hosts => '%',
  }

  # Configures authentication for Neutron to the identity service.
  class { 'neutron::keystone::auth':
    password            => 'SRVPW',
    email               => 'example@example.com',
    public_address      => '10.0.0.1',
    admin_address       => '10.0.0.1',
    internal_address    => '10.0.0.1',
    region              => 'regionOne',
  }

  ########################
  #### deploy script ####
  ########################

  # The controller deploy script is copied to the controller
  file {'/etc/puppet/deployController.sh':
    ensure      => directory,
    owner       => 'root',
    group       => 'root',
    mode        => '0755',
    source      => 'puppet:///modules/nova/deployController.sh',
```

site.pp


```
    # Listing requirements to ensure that the script is not
    # copied and executed before depended service are in place.
    require     => Class[ '::neutron',
                          'nova',
                          'neutron::server',
                          'nova::api',
                          'glance::api',
                          'nova::rabbitmq' ],
  }

  # The deploy script ensures;
  # 1. That various Openstack service parameters are set which
  #    there exists no method for in the modules
  # 2. Static IP is set based on the dynamically assigned IP
  # 3. Glance images is downloaded and imported
  # 4. Installation of the Openstack Horizon component with
  #    its dependencies
  # 5. Installation and configuration of Munin
  exec { 'deployController':
    command     => '/etc/puppet/deployController.sh',
    #creates can be used to check for a spesific file before
    #the exec is run. If the files exists, the exec wont run
    creates     => '/var/tmp/deploy.sh.lock',
    path        => '/usr/bin:/usr/sbin:/bin:/sbin',
    # Execution of the script is dependent on the script beeing copied first.
    require     => File['/etc/puppet/deployController.sh'],
  }

  ####################
  #### Scenario1 #####
  ####################

  file {'/etc/puppet/scenario.sh':
    ensure      => directory,
    owner       => 'root',
    group       => 'root',
    mode        => '0755',
    source      => 'puppet:///modules/nova/scenario.sh',
    # Listing requirements to ensure that the script is not
    # copied and executed before depended service are in place.
    require     => Exec['deployController'],
  }

  #################
  #### MUNIN #####
  #################

  # The munin script handles the discovery of new munin-nodes.
  file {'/etc/puppet/munin.sh':
    ensure      => directory,
    owner       => 'root',
    group       => 'root',
    mode        => '0755',
    source      => 'puppet:///modules/nova/munin.sh',
    require     => Exec['deployController'],
  }
}
```

site.pp

C:\My Documents\site.pp 18. mai 2014 16:24

```
node network {

  # To synchronize services acrosss multiple machines, NTP is installed.
  # The network node fetches time data from the controller in first instance.

  class { '::ntp':
     servers => [ '10.0.0.1', 'ntp1.corp.com', 'ntp2.corp.com' ],
  }

  # Installation of the Neutron service with configuration
  # of message-queue server.
  class { '::neutron':
    enabled         => true,
    bind_host       => '0.0.0.0',
    rabbit_host     => '10.0.0.1',
    rabbit_user     => 'guest',
    rabbit_password => 'SRVPW',
    verbose         => true,
    debug           => true,
  }

  # Configures authentication and database connection for the Neutron service.
  class { 'neutron::server':
    auth_host       => '10.0.0.1',
    auth_password   => 'SRVPW',
    sql_connection  => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
    connection      => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
  }

  # Enable the Open VSwitch plugin server with Neutron.
  class { 'neutron::plugins::ovs':
    # Enable vlan-tagging
    tenant_network_type => 'vlan',
    # Available vlans is 2000 up to 2099.
    network_vlan_ranges => 'physnet1:2000:2099',
  }

  # Installation and configuration of the metadata agent for Neutron.
  class { 'neutron::agents::metadata':
    auth_password => 'SRVPW',
    auth_url      => 'http://10.0.0.1:5000/v2.0',
    auth_region   => 'regionOne',
    auth_user     => 'neutron',
    shared_secret => 'METADATA_PASS',
    metadata_ip   => '10.0.0.1',
  }

  # Installation of the dhcp, l3 and ovs agent for Neutron.
  class { 'neutron::agents::dhcp': }
  class { 'neutron::agents::l3': }
  class { 'neutron::agents::ovs':
    firewall_driver =>
    'neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver',
  }

  # The network deploy script is copied to the network
```

site.pp


```
  file {'/etc/puppet/deployNetwork.sh':
    ensure      => directory,
    owner       => 'root',
    group       => 'root',
    mode        => '0755',
    source      => 'puppet:///modules/nova/deployNetwork.sh',
    # OVS specific parameters is set in the deploy script so it is required
    # that the OVS plugin is installed before the script is executed.
    require     => Class['neutron::plugins::ovs'],
  }

  # The deploy script ensures;
  # 1. That various Openstack service parameters are set which
  #    there exists no method for in the modules
  # 2. Network configuration is set for the interfaces;
  #        Openstack-mgmt, VM-Configuration and VM-Internet access.
  # 3. OVS bridges and ports are created.
  # 4. IPv4 forwarding is enabled.
  exec { 'deployNetwork':
    command     => '/etc/puppet/deployNetwork.sh',
    # 'creates' checks to se if deploy.sh.lock exists on the Puppet-agent
    # before executing deployNetwork.sh. If the file exist, the script will not run.
    creates     => '/var/tmp/deploy.sh.lock',
    path        => '/usr/bin:/usr/sbin:/bin:/sbin',
    require     => File['/etc/puppet/deployNetwork.sh'],
    # The configuration of interfaces requires the networking service to be restarted.
    notify      => Service['networking'],
  }
  service { 'networking':
    hasrestart  => true,
  }

  #################
  #### MUNIN #####
  #################

  # Ensure that munin-node is installed.
  package { 'munin-node':
    ensure      => 'present',
  }

  # Configure the munin-node to send munin-data to the controller.
  exec { 'configure munin-node.conf':
    command     => "sed -i 's/allow \\^127\\\\.0\\\\.0\\\\.1\\$/allow
    \\^10\\\\.0\\\\.0\\\\.1\\$/g' /etc/munin/munin-node.conf",
    path        => ['/bin'],
    user        => "root",
    # For this exec to run, it is required that munin-node is installed first.
    require     => Package['munin-node'],
    # If the exec runs, the munin-configuration have been altered and the munin-node service
    have to be restarted.
    notify      => Service['munin-node'],
  }
  service { 'munin-node':
    hasrestart  => true,
  }
}
```

site.pp

C:\My Documents\site.pp 18. mai 2014 16:24

```
node /^compute\d+$/ {

  ##############
  #### NTP ####
  ##############

  # To synchronize services acrosss multiple machines, NTP is installed.
  # The compute node fetches time data from the controller in first instance.

    class { '::ntp':
    servers => [ '10.0.0.1', 'ntp1.corp.com', 'ntp2.corp.com' ],
  }

  # Installation and configuration of the Nova service.
  class { 'nova':
    database_connection => 'mysql://nova:SRVPW@10.0.0.1/nova',
    rabbit_userid       => 'guest',
    rabbit_password     => 'SRVPW',
    image_service       => 'nova.image.glance.GlanceImageService',
    glance_api_servers  => '10.0.0.1:9292',
    verbose             => false,
    rabbit_host         => '10.0.0.1',
    rpc_backend         => 'nova.rpc.impl_kombu',
  }

  # Installation and authentication configuration for the nova-api.
  class { 'nova::api':
    enabled             => true,
    admin_password      => 'SRVPW',
    auth_host           => '10.0.0.1',
    sync_db             => 'false',
  }

  # Installation and configuration for the nova compute service.
  class { 'nova::compute':
    enabled                             => true,
    ensure_package                      => 'present',
    neutron_enabled                     => true,
   vnc_enabled                         => true,
    vncproxy_host                       => '10.0.0.1',
    vncserver_proxyclient_address       => "$::ipaddress_eth0",
  }

  # Configuration of VNC-access to VM's
  class { 'nova::compute::libvirt':
    vncserver_listen  => '0.0.0.0',
  }

  # Configuring nova to use the dedicated network node.
  class { 'nova::network::neutron':
    neutron_admin_password => 'SRVPW',
    neutron_url            => 'http://10.0.0.1:9696',
    neutron_admin_auth_url => 'http://10.0.0.1:35357/v2.0',
    neutron_region_name    => 'regionOne',
  }
```

site.pp


```
# Specifying libvirt driver
class { 'nova::compute::neutron':
  libvirt_vif_driver  => 'nova.virt.libvirt.vif.LibvirtGenericVIFDriver',
}

# Installation of the neutron service with message-queue and plugin configuration.
class { '::neutron':
  enabled         => false,
  rabbit_host     => '10.0.0.1',
  rabbit_user     => 'guest',
  rabbit_password => 'SRVPW',
  core_plugin     => 'neutron.plugins.openvswitch.ovs_neutron_plugin.OVSNeutronPluginV2',
}

# Configuration of the neutron database connection and authentication to the controller.
class { 'neutron::server':
  enabled         => 'false',
  auth_host       => '10.0.0.1',
  auth_password   => 'SRVPW',
  sql_connection  => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
  connection      => 'mysql://neutron:SRVPW@10.0.0.1/neutron',
}

# Configures neutron to use vlan-tagging where available vlans is 2000 to 2099.
class { 'neutron::plugins::ovs':
  tenant_network_type => 'vlan',
  network_vlan_ranges => 'physnet1:2000:2099',
}

#  Firewall driver configuration for the neutron OVS plugin.
class { 'neutron::agents::ovs':
   firewall_driver =>
   'neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver',
 }

#################
#### MUNIN #####
#################

# Ensure that munin-node is installed.
package { 'munin-node':
  ensure      => 'present',
}

# Configure the munin-node to send munin-data to the controller.
exec { 'configure munin-node.conf':
  command     => "sed -i 's/allow \\^127\\\\.0\\\\.0\\\\.1\\$/allow
  \\^10\\\\.0\\\\.0\\\\.1\\$/g' /etc/munin/munin-node.conf",
  path        => ['/bin'],
  user        => "root",
  # For this exec to run, it is required that munin-node is installed first.
  require     => Package['munin-node'],
  # If the exec runs, the munin-configuration have been altered and the munin-node service
  have to be restarted.
  notify      => Service['munin-node'],
}
service { 'munin-node':
```

site.pp


```
    hasrestart  => true,
  }

  ########################
  #### deploy script ####
  ########################

  # The compute deploy script is copied to the compute nodes.
  file {'/etc/puppet/deployCompute.sh':
    ensure      => directory,
    owner       => 'root',
    group       => 'root',
    mode        => '0755',
    source      => 'puppet:///modules/nova/deployCompute.sh',
   # Listing requirements to ensure that the script is not
   # copied and executed before depended service are in place.
    require     => Class[ '::neutron',
                          'nova::api',
                          'nova',
                          'neutron::plugins::ovs' ],
  }

  # The deploy script ensures;
  # 1. That various Openstack service parameters are set which
  #    there exists no method for in the modules
  # 2. Network configuration is set for the interfaces;
  #        Openstack-mgmt and VM-Configuration.
  # 3. OVS bridges and ports are created.
  # 4. IPv4 forwarding is enabled.
  exec { 'deployCompute':
    command     => '/etc/puppet/deployCompute.sh',
    # 'creates' checks to se if deploy.sh.lock exists on the Puppet-agent
    # before executing deployCompute.sh. If the file exist, the script will not run.
    creates     => '/var/tmp/deploy.sh.lock',
    path        => '/usr/bin:/usr/sbin:/bin:/sbin',
    require     => File['/etc/puppet/deployCompute.sh'],
    # The configuration of interfaces requires the networking service to be restarted.
    notify      => Service['networking'],
  }
  service { 'networking':
    hasrestart  => true,
  }
}
```

deployController.sh

C:\My Documents\deployController.sh 18. mai 2014 16:10

```
#!/bin/bash

# Creates the lock file, to prevent this script to run more than once
echo "Do not delete this file unless you know what you are doing!" >> /var/tmp/deploy.sh.lock

# Retrives the IP for the controller
ipAddr=$(facter ipaddress)

# Adding networking support
cd /etc/network
echo "auto lo" > interfaces
echo "iface lo inet loopback" >> interfaces
echo "auto eth0" >> interfaces
echo "iface eth0 inet static" >> interfaces
echo "address $ipAddr" >> interfaces
echo "dns-nameservers 8.8.8.8" >> interfaces
echo "gateway 10.0.0.254" >> interfaces
echo "netmask 255.255.255.0" >> interfaces

# Setting rabbitMQ guest-password and enabling web-interface for the
# rabbitMQ-server for debuging purposes.
rabbitmqctl change_password guest SRVPW
# Webinterface available at http://localhost:55672
/usr/lib/rabbitmq/bin/rabbitmq-plugins enable rabbitmq_management
service rabbitmq-server restart

# Removes nova.sqlite if it exists, to prevent nova from mistaking it with the sql DB
if [ -f /var/lib/nova/nova.sqlite ]; then
   sudo rm /var/lib/nova/nova.sqlite
fi

# Removes glance.sqlite if it exists, to prevent glance from mistaking it with the sql DB
if [ -f /var/lib/glance/glance.sqlite ]; then
   sudo rm /var/lib/glance/glance.sqlite
fi

cd /etc/nova
# Adds my_ip to the default section in nova.conf
# This is done by making the change in a temp file and replace the original
sed '/\[DEFAULT\]/a my_ip=10.0.0.1' ./nova.conf > ./tmpFile1
sed '/^\[DEFAULT\]/a linuxnet_interface_driver =
nova.network.linux_net.LinuxOVSInterfaceDriver' ./tmpFile1 > ./tmpFile2
# Adds rabbit credentials to nova.conf
sed '/\[DEFAULT\]/a rabbit_userid=guest' ./tmpFile2 > ./tmpFile1
sed '/\[DEFAULT\]/a rabbit_password=SRVPW' ./tmpFile1 > ./tmpFile2
sed '/\[DEFAULT\]/a rabbit_host=10.0.0.1' ./tmpFile2 > ./tmpFile1
sed '/\[DEFAULT\]/a rabbit_port=5672' ./tmpFile1 > ./tmpFile2
rm ./nova.conf
mv ./tmpFile2 ./nova.conf
rm ./tmp*
# Adds authentication parameters to api-paste.ini
sed '/\[filter:authtoken\]/a auth_host=10.0.0.1' ./api-paste.ini > ./tmpFile1
sed '/\[filter:authtoken\]/a admin_user=nova' ./tmpFile1 > ./tmpFile2
sed '/\[filter:authtoken\]/a admin_tenant_name=services' ./tmpFile2 > ./tmpFile1
sed '/\[filter:authtoken\]/a admin_password=SRVPW' ./tmpFile1 > ./tmpFile2
rm ./api-paste.ini
mv ./tmpFile2 ./api-paste.ini
```

deployController.sh

C:\My Documents\deployController.sh 18. mai 2014 16:10

```
rm ./tmp*

# Adding auth_url to the neutron.conf file
cd /etc/neutron
sed '/^\[keystone_authtoken\]/a auth_url = http:\/\/10.0.0.1:35357\/v2.0' ./neutron.conf > ./
tmpFile
rm ./neutron.conf
mv ./tmpFile ./neutron.conf

# Configuring the config files for Glance
cd /etc/glance
# Adds authentication parameters to glance-registry-paste.ini
sed '/^\[filter:authtoken\]/a auth_host=10.0.0.1' ./glance-registry-paste.ini > ./tmpFile1
sed '/^\[filter:authtoken\]/a admin_user=glance' ./tmpFile1 > ./tmpFile2
sed '/^\[filter:authtoken\]/a admin_tenant_name=services' ./tmpFile2 > ./tmpFile1
sed '/^\[filter:authtoken\]/a admin_password=SRVPW' ./tmpFile1 > ./tmpFile2
rm ./glance-registry-paste.ini
mv ./tmpFile2 ./glance-registry-paste.ini
# Adds authentication parameters to glance-api-paste.ini
sed '/^\[filter:authtoken\]/a auth_host=10.0.0.1' ./glance-api-paste.ini > ./tmpFile1
sed '/^\[filter:authtoken\]/a admin_user=glance' ./tmpFile1 > ./tmpFile2
sed '/^\[filter:authtoken\]/a admin_tenant_name=services' ./tmpFile2 > ./tmpFile1
sed '/^\[filter:authtoken\]/a admin_password=SRVPW' ./tmpFile1 > ./tmpFile2
rm ./glance-api-paste.ini
mv ./tmpFile2 ./glance-api-paste.ini
rm ./tmp*

# Adding database connection for neutron ovs plugin
cd /etc/neutron/plugins/openvswitch
echo "[database]" >> ./ovs_neutron_plugin.ini
echo "connection = mysql://neutron:SRVPW@10.0.0.1/neutron" >> ./ovs_neutron_plugin.ini
service neutron-plugin-openvswitch-agent restart

# Downloads a CirrOS 0.3.1 and a Ubuntu 12.04 image to folder images
# and adds the images to glance

# Authenication credentials
export OS_TENANT_NAME=openstack
export OS_USERNAME=admin
export OS_PASSWORD=SRVPW
export OS_AUTH_URL="http://10.0.0.1:5000/v2.0/"
# Creates the folder images at /var/tmp/
cd /var/tmp/
mkdir images
cd images
# Downloads images
wget http://cdn.download.cirros-cloud.net/0.3.1/cirros-0.3.1-i386-disk.img
wget http://cloud-images.ubuntu.com/releases/12.04.2/release/
ubuntu-12.04-server-cloudimg-i386-disk1.img
# Adds images to glance
glance image-create --name="CirrOS 0.3.1" --disk-format=qcow2 --container-format=bare --
is-public=true < cirros-0.3.1-i386-disk.img
glance image-create --name="Ubuntu 12.04" --disk-format=qcow2 --container-format=bare --
is-public=true < ubuntu-12.04-server-cloudimg-i386-disk1.img

# Installing packages for horizon
apt-get install memcached libapache2-mod-wsgi libapache2-mod-wsgi apache2 openstack-dashboard
```

deployController.sh


```
 -y

# Enables the package wsgi and openstack-dashboard
a2enmod wsgi
a2enconf openstack-dashboard

# Changes the value of OPENSTACK_KEYSTONE_DEFAULT_ROLE from Member to _member_
cd /etc/openstack-dashboard
sed -i 's/OPENSTACK_KEYSTONE_DEFAULT_ROLE = "Member"/OPENSTACK_KEYSTONE_DEFAULT_ROLE =
"_member_"/g' /etc/openstack-dashboard/local_settings.py

# Installing munin and NMAP
apt-get install munin nmap -y
# Configuring controller as the munin master
cd /etc/munin
sed -i 's/localhost.localdomain/controller/g' munin.conf
sed '/^<Directory \/var\/cache\/munin\/www>/a Require all granted' apache.conf > tmpFile1
sed '/^<Directory \/var\/cache\/munin\/www>/a Allow from all' tmpFile1 > apache.conf
# Restart of services
service apache2 restart
service munin-node restart
# Cleanup
rm tmpFile1

# Run munin scipt every 5 minutes, to add new nodes for monitoring
echo "*/5 * * * * root /etc/puppet/munin.sh" >> /etc/crontab
# Run the deployment of scenario script
echo "*/5 * * * * root /etc/puppet/scenario.sh" >> /etc/crontab

#END
```

deployNetwork.sh

C:\My Documents\deployNetwork.sh 18. mai 2014 16:13

```
#!/bin/bash

echo "Do not delete this file unless you know what you are doing!" >> /var/tmp/deploy.sh.lock

# Retrives the IP for the network node
ipAddr=$(facter ipaddress)

# Configuring additional networking support
cd /etc/network
echo "auto lo" > interfaces
echo "iface lo inet loopback" >> interfaces
echo "auto eth0" >> interfaces
echo "iface eth0 inet static" >> interfaces
echo "address $ipAddr" >> interfaces
echo "netmask 255.255.255.0" >> interfaces
echo "gateway 10.10.10.254" >> interfaces
echo "dns-nameservers 8.8.8.8" >> interfaces
echo "auto eth1" >> interfaces
echo "iface eth1 inet manual" >> interfaces
echo "up ifconfig \$IFACE 0.0.0.0 up" >> interfaces
echo "down ifconfig \$IFACE down" >> interfaces
echo "auto eth2" >> interfaces
echo "iface eth2 inet manual" >> interfaces
echo "up ifconfig \$IFACE 0.0.0.0 up" >> interfaces
echo "up ip link set \$IFACE promisc on" >> interfaces
echo "down ip link set \$IFACE promisc off" >> interfaces
echo "down ifconfig \$IFACE down" >> interfaces

# Adding and configuration of the bridges for the OpenStack network-node
ovs-vsctl add-br br-ex
ovs-vsctl add-port br-ex eth2
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1

# Adding bridging for the openvswitch plugin
cd /etc/neutron/plugins/openvswitch
sed '/^\[OVS\]/a bridge_mappings = physnet1:br-eth1' ./ovs_neutron_plugin.ini > ./tmpFile
rm ./ovs_neutron_plugin.ini
mv ./tmpFile ./ovs_neutron_plugin.ini

# Adding database connection for neutron ovs plugin
echo "[database]" >> ./ovs_neutron_plugin.ini
echo "connection = mysql://neutron:SRVPW@10.0.0.1/neutron" >> ./ovs_neutron_plugin.ini
service neutron-plugin-openvswitch-agent restart

# Enabling ipv4 forwarding
sed -i 's/#net.ipv4.ip\_forward\=1/net.ipv4.ip\_forward\=1/g' /etc/sysctl.conf
sed -i 's/#net.ipv4.conf.all.rp\_filter\=1/net.ipv4.conf.all.rp\_filter\=0/g' /etc/sysctl.conf
sed -i 's/#net.ipv4.conf.default.rp\_filter\=1/net.ipv4.conf.default.rp\_filter=0/g' /etc/
sysctl.conf
sysctl -p

#END
```

deployCompute.sh

C:\My Documents\deployCompute.sh 18. mai 2014 16:13

```
#!/bin/bash

# Ensures the script to only run once
echo "Do not delete this file unless you know what you are doing!" >> /var/tmp/deploy.sh.lock

# Retrives the IP for the compute node by using facter
ipAddr=$(facter ipaddress)

# Configuring additional networking support
cd /etc/network
echo "auto lo" > interfaces
echo "iface lo inet loopback" >> interfaces
echo "auto eth0" >> interfaces
echo "iface eth0 inet static" >> interfaces
echo "address $ipAddr" >> interfaces
echo "netmask 255.255.255.0" >> interfaces
echo "gateway 10.10.10.254" >> interfaces
echo "dns-nameservers 8.8.8.8" >> interfaces
echo "auto eth1" >> interfaces
echo "iface eth1 inet manual" >> interfaces
echo "up ifconfig \$IFACE 0.0.0.0 up" >> interfaces
echo "down ifconfig \$IFACE down" >> interfaces

# Disable packet destination filtering
sed -i 's/#net.ipv4.conf.all.rp\_filter\=1/net.ipv4.conf.all.rp\_filter\=0/g' /etc/sysctl.conf
sed -i 's/#net.ipv4.conf.default.rp\_filter\=1/net.ipv4.conf.default.rp\_filter=0/g' /etc/
sysctl.conf
sysctl -p

# Adding database connection for neutron ovs plugin
cd /etc/neutron/plugins/openvswitch
echo "[database]" >> ./ovs_neutron_plugin.ini
echo "connection = mysql://neutron:SRVPW@10.0.0.1/neutron" >> ./ovs_neutron_plugin.ini
sed '/^\[OVS\]/a bridge_mappings = physnet1:br-eth1' ./ovs_neutron_plugin.ini > ./tmpFile1
rm ./ovs_neutron_plugin.ini
mv ./tmpFile1 ./ovs_neutron_plugin.ini
service neutron-plugin-openvswitch-agent restart

# Adding integration bridge
ovs-vsctl add-br br-int

# Configuring OVS to use VLANs
ovs-vsctl add-br br-eth1
ovs-vsctl add-port br-eth1 eth1

# Adding auth_url to the neutron.conf file
cd /etc/neutron
sed '/^\[keystone_authtoken\]/a auth_url = http:\/\/10.0.0.1:35357\/v2.0' ./neutron.conf > ./
tmpFile
rm ./neutron.conf
mv ./tmpFile ./neutron.conf

# Adds the my_ip value, driver, glance_host and rabbit parameters in nova.conf
cd /etc/nova
sed '/^\[DEFAULT\]/a my_ip='$ipAddr'' ./nova.conf > ./tmpFile1
sed '/^\[DEFAULT\]/a linuxnet_interface_driver =
nova.network.linux_net.LinuxOVSInterfaceDriver' ./tmpFile1 > ./tmpFile2
```

deployCompute.sh

C:\My Documents\deployCompute.sh 18. mai 2014 16:13

```
sed '/^\[DEFAULT\]/a rabbit_userid=guest' ./tmpFile2 > ./tmpFile1
sed '/^\[DEFAULT\]/a rabbit_password=SRVPW' ./tmpFile1 > ./tmpFile2
sed '/^\[DEFAULT\]/a rabbit_host=10.0.0.1' ./tmpFile2 > ./tmpFile1
sed '/^\[DEFAULT\]/a rabbit_port=5672' ./tmpFile1 > ./tmpFile2
sed '/^\[DEFAULT\]/a glance_host=10.0.0.1' ./tmpFile2 > ./tmpFile1
rm ./nova.conf
mv ./tmpFile1 ./nova.conf
rm ./tmp*

# Adds authenication credentials to api-paste.ini
sed '/\[filter:authtoken\]/a auth_host=10.0.0.1' ./api-paste.ini > ./tmpFile1
sed '/\[filter:authtoken\]/a admin_user=nova' ./tmpFile1 > ./tmpFile2
sed '/\[filter:authtoken\]/a admin_tenant_name=services' ./tmpFile2 > ./tmpFile1
sed '/\[filter:authtoken\]/a admin_password=SRVPW' ./tmpFile1 > ./tmpFile2
sed '/\[filter:authtoken\]/a auth_port=35357' ./tmpFile2 > ./tmpFile1
sed '/\[filter:authtoken\]/a admin_protocol=http' ./tmpFile1 > ./tmpFile2
rm ./api-paste.ini
mv ./tmpFile2 ./api-paste.ini
rm ./tmp*

# Install the python-guestfs package require for the compute nodes
apt-get install python-guestfs -y

#END
```

munin.sh

C:\My Documents\munin.sh 18. mai 2014 16:15

```
#!/bin/bash

cd /etc/munin/
# Creates a temporary file
touch munin.tmp2

# Default location for munin.conf
MUNIN_CONFIG="/etc/munin/munin.conf"

# The config remains unchanged until otherwise proven
CONFIG_UPDATE=0

# Ping sweeps the network with IPs from 10.0.0.1 to 10.0.0.253
# and adds the results into munin.tmp2
scan() {
   nmap -sP 10.0.0.1-253 > munin.tmp2
# Calls the parse function
   parse;
}

parse() {
   # Extracts only the IP address from munin.tmp2 and stores it in munin.tmp
   cat munin.tmp2 | grep -E -o '(10\.0\.0\.(25[0-3]|2[0-4][0-9]|[01]?[0-9][0-9]?))' /etc/
   munin/munin.tmp2 > munin.tmp
   rm munin.tmp2
   # For each IP address in munin.tmp:
   cat munin.tmp | while read ip; do
      echo $ip > ipAddr
      # Extracts the last octet from the IP address
      IP_OCT4=$(awk -F'[.]' '{print $4}' ipAddr)
      grep $ip $MUNIN_CONFIG
      if [ "$?" != "0" ]; then
         # If the last octet is 2 then this node will be named network
         if [ "$IP_OCT4" == "2" ]; then
            echo "[network]" >> $MUNIN_CONFIG
            echo -e "address $ip\n" >> $MUNIN_CONFIG
            # munin.conf changed and the service needs to be restarted
            CONFIG_UPDATE=1
         # If the last octet is greater than 2 then this node will be named named compute(1
         or greater)
         elif [ "$IP_OCT4" -gt "2" ]; then
            # The last octet is decreased by 2 to have the hostname for the compute nodes to
            be
            # in a logical and sequential way
            IP_OCT4=$(expr $IP_OCT4 - 2)
            echo "[compute$IP_OCT4]" >> $MUNIN_CONFIG
            echo -e "address $ip\n" >> $MUNIN_CONFIG
            # munin.conf changed and the service needs to be restarted
            CONFIG_UPDATE=1
         fi
      fi
   done
   # If the CONFIG_UPDATE has been set to true above, then it will do a restart of the munin
   service
   if [ "$CONFIG_UPDATE" != "0" ]; then
      service munin restart
   fi
```

munin.sh

C:\My Documents\munin.sh 18. mai 2014 16:15

```
}
# Calls the scan function
scan;
# Removes temporary files
rm /etc/munin/munin.tmp
rm /etc/munin/ipAddr
exit 0

#END
```

scenario.sh

C:\My Documents\scenario.sh 19. mai 2014 17:48

```
#!/bin/bash

# Check if the scenario already have been rolled out
if [ -f "/var/tmp/jungleScenario" ]; then
  exit 0;
fi

export OS_TENANT_NAME=openstack
export OS_USERNAME=admin
export OS_PASSWORD=SRVPW
export OS_AUTH_URL="http://10.0.0.1:5000/v2.0/"

COMPUTE_NODES_REQ="2"
# Count the number of compute nodes that is up and ready
COMPUTE_NODES=$(nova-manage service list | grep compute | grep ":-)" | wc | awk '{print $1}')

if [ "$COMPUTE_NODES" -lt "$COMPUTE_NODES_REQ" ]; then
  exit 0;
fi


# Openstack mgmt parameters

EXT_NET="ExternalNetwork"
FLOAT_IP_START="128.39.141.217"
FLOAT_IP_END="128.39.141.218"
EXT_GW="128.39.140.1"
EXT_CIDR="128.39.140.0/23"
ROUTER="OpenstackRouter"

# Jungle scenario paramteres

TENANT_NAME="Jungle"
TENANT_NET1="Jungle-ExtNet"
TENANT_NET2="Jungle-IntNet"
SUBNET_NET1="10.1.0.0/24"
SUBNET_NET2="10.2.0.0/24"
SUBNET_NET1_GW="10.1.0.1"
SUBNET_NET2_GW="10.2.0.1"
DNS_SERVERS="128.39.32.2 8.8.8.8 8.8.8.7"
USER_NAME="jungleuser"
USER_PASSWORD="password"

#Creating the base networks for neutron

neutron net-create $EXT_NET -- --router:external=True --provider:network_type vlan --provider
:physical_network physnet1 --provider:segmentation_id 2
neutron subnet-create $EXT_NET --allocation-pool start=$FLOAT_IP_START,end=$FLOAT_IP_END --
gateway=$EXT_GW --enable_dhcp=False $EXT_CIDR
keystone tenant-create --name $TENANT_NAME
ID_TENANT=$(keystone tenant-list | grep $TENANT_NAME | awk '{print $2;}')
neutron router-create $ROUTER --tenant-id $ID_TENANT
ID_ROUTER=$(neutron router-list | grep $ROUTER | awk '{print $2;}')
ID_EXT_NET=$(neutron net-list | grep $EXT_NET | awk '{print $2;}')
neutron router-gateway-set $ID_ROUTER $ID_EXT_NET
neutron net-create --tenant-id $ID_TENANT $TENANT_NET1 --provider:network_type vlan --
provider:physical_network physnet1 --provider:segmentation_id 2000
```

scenario.sh

C:\My Documents\scenario.sh 19. mai 2014 17:48

```
neutron subnet-create --tenant-id $ID_TENANT $TENANT_NET1 $SUBNET_NET1 --gateway
$SUBNET_NET1_GW --dns_nameservers list=true $DNS_SERVERS
neutron net-create --tenant-id $ID_TENANT $TENANT_NET2 --provider:network_type vlan --
provider:physical_network physnet1 --provider:segmentation_id 2001
neutron subnet-create --tenant-id $ID_TENANT $TENANT_NET2 $SUBNET_NET2 --gateway
$SUBNET_NET2_GW --dns_nameservers list=true $DNS_SERVERS
ID_SUBNET_NET1=$(neutron subnet-list | grep $SUBNET_NET1 | awk '{print $2;}')
ID_SUBNET_NET2=$(neutron subnet-list | grep $SUBNET_NET2 | awk '{print $2;}')
neutron router-interface-add $ID_ROUTER $ID_SUBNET_NET1
neutron router-interface-add $ID_ROUTER $ID_SUBNET_NET2

# Administrativ options for the tenant

keystone user-create --name $USER_NAME --tenant_id $ID_TENANT --pass $USER_PASSWORD

SSH_KEYNAME="ctrlKey"

ssh-keygen -f /root/.ssh/id_rsa -P ""

nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0

FLAVOR_NAME="m1.smaller"
IMAGE_CIRROS=$(nova image-list | grep CirrOS | awk '{print $2;}')
IMAGE_UBUNTU=$(nova image-list | grep Ubuntu | awk '{print $2;}')
nova flavor-create $FLAVOR_NAME auto 1024 7 1

unset OS_TENANT_NAME OS_USERNAME OS_PASSWORD
export OS_TENANT_NAME=$TENANT_NAME
export OS_USERNAME=$USER_NAME
export OS_PASSWORD=$USER_PASSWORD

ID_NET1=$(neutron net-list | grep $TENANT_NET1 | awk '{print $2;}')
ID_NET2=$(neutron net-list | grep $TENANT_NET2 | awk '{print $2;}')
MANAGER="DomainController"

# Launch the instances making up the scenario

nova keypair-add --pub_key /root/.ssh/id_rsa.pub $SSH_KEYNAME
nova boot --image $IMAGE_UBUNTU --flavor $FLAVOR_NAME --key_name $SSH_KEYNAME --nic net-id=
$ID_NET1 Predator
nova boot --image $IMAGE_CIRROS --flavor m1.tiny --key_name $SSH_KEYNAME --nic net-id=
$ID_NET1 WWW-User
nova boot --image $IMAGE_UBUNTU --flavor $FLAVOR_NAME --key_name $SSH_KEYNAME --nic net-id=
$ID_NET1 WWW
nova boot --image $IMAGE_CIRROS --flavor m1.tiny --key_name $SSH_KEYNAME --nic net-id=
$ID_NET2 SQL-server
nova boot --image $IMAGE_CIRROS --flavor m1.tiny --key_name $SSH_KEYNAME --nic net-id=
$ID_NET2 FTP
nova boot --image $IMAGE_CIRROS --flavor m1.tiny --key_name $SSH_KEYNAME --nic net-id=
$ID_NET2 SMTP/POP3
nova boot --image $IMAGE_CIRROS --flavor m1.tiny --key_name $SSH_KEYNAME --nic net-id=
$ID_NET2 DomainController

# Assing floating IP

nova floating-ip-create $EXT_NET
```

scenario.sh

C:\My Documents\scenario.sh 19. mai 2014 17:48

```
nova list | grep $MANAGER | grep ACTIVE
while [ $? != 0 ]
do
  echo "Too soon for assigning floating IP!"
  sleep 5
  nova list | grep $MANAGER | grep ACTIVE
done
nova add-floating-ip $MANAGER $FLOAT_IP_END

# Stop this scenario being rolled out more than once

touch /var/tmp/jungleScenario
```

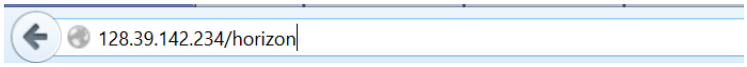
Tillegg C

Brukerveiledning

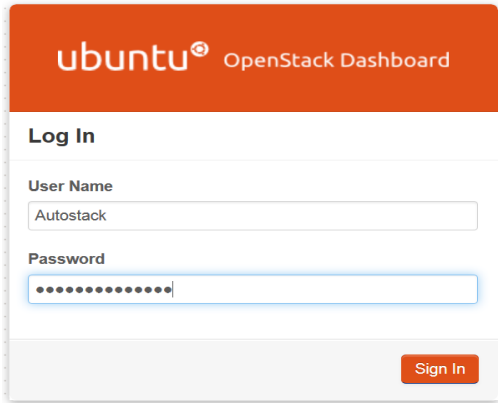
Brukerveiledning Autostack

Hvordan logger jeg inn?

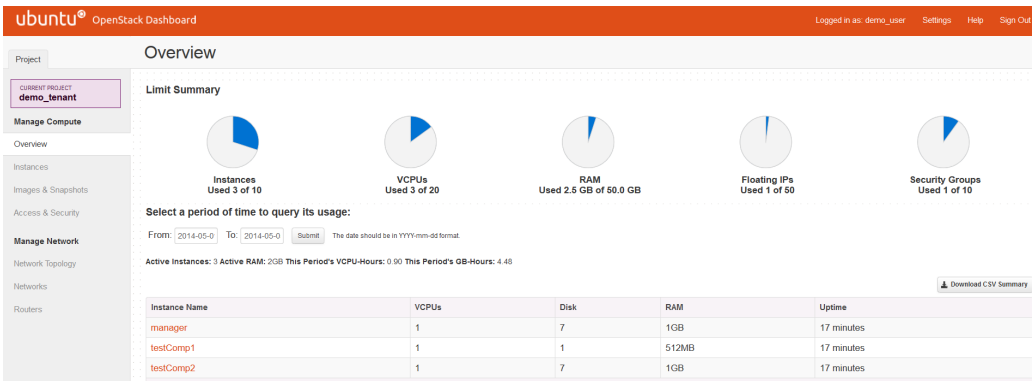
1. Administrator tilbyr IP-adresse eller vertsnavn til OpenStack-kontrollpanelet, samt gyldig brukernavn og passord.
2. Løsningen aksesseres via en nettleser, f.eks. Firefox. Hvor bruker skriver inn IP-adresse eller vertsnavn i adressefeltet.



3. I det påfølgende innloggingsvinduet skriver inn brukernavn og passord som mottatt fra adminstrator.

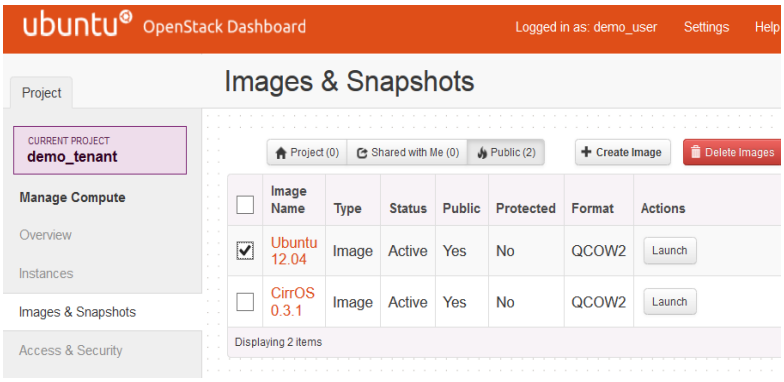


4. Etter vellykket innlogging vil kontrollpanelet til OpenStack se ut omtrent som vist nedenfor:



Hvordan oppretter jeg instanser?

1. Logg inn på OpenStack som vist tidligere i guiden.
2. Velg aktuelt prosjekt, klikk så på **Images & Snapshot**
   Panelet viser her de imagene som er blitt lastet opp og er tilgjengelige til bruk i dette prosjektet
3. Velg et image, og trykk **Launch**

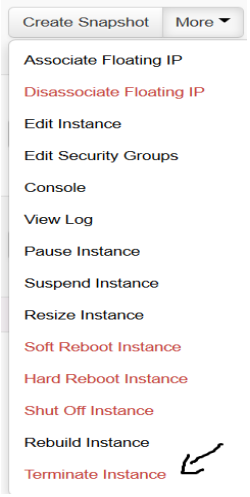


4. I dialogboksen som dukker opp, er det en rekke verdier som settes for den instansen som skal opprettes.

Availability Zone	Settes som default til aktuell sone. Kan her benytte 'nova'
Instance Name	Navn på den virtuelle maskinen
Flavor	Spesifiser størrelsen på instansen
Instance Count	Er default satt til 1. Dersom det er ønskelig å opprette flere instanser, spesifiseres dette her.
Instance Boot Source	Er default satt til **boot from image.**Her er det muligheter for å spesifisere andre boot-ønsker.
Image Name	Dersom det ble valgt image boot tidligere, får man nå opp **Image Name**
Keypair	Her spesifiseres et nøkkelpar. Unødvendig hvis det benyttes statisk root-passord, eller statisk nøkkelsett.
Security Groups	Aktivere sikkerhetsgruppe for instansen(dersom det ikke er definert, kan en default sikkerhetsgruppe tilegnes.
Selected Networks	Dersom det er ønskelig å legge til et nettverk, klikker man på + tegnet under **Available Networks.**
Customization Scripts	Her er det mulig å legge inn kustomiserte script som kjøres etter en instans er blitt startet opp.

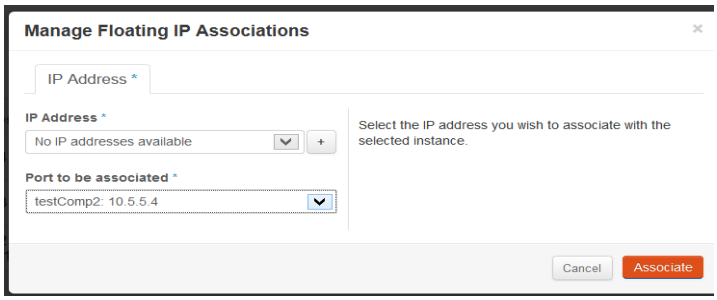
Hvordan sletter jeg instanser?

1. Logg inn på OpenStack som vist tidligere i guiden
2. Velg et prosjekt, og klikk **Instances**
3. Ved siden av ønsket instance, trykk på **More**. Ved rullegardinlisten som dukker opp, velg **Terminate Instance**. Her finnes også en rekke andre valgmuligheter for en instans.



Hvordan tilegner jeg floating IP?

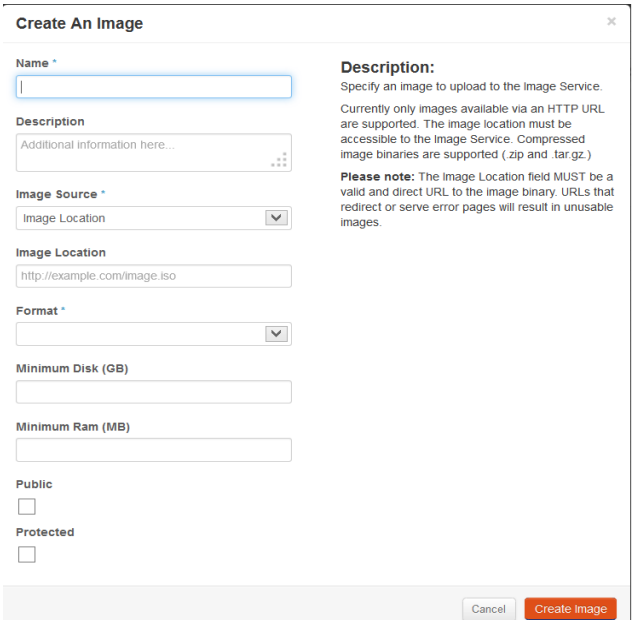
1. Logg på OpenStack som vist tidligere i guiden
2. Velg et prosjekt, og klikk på **Instances**
3. Ved siden av ønsket instance, trykk på **More**
4. Ved rullegardinlisten som dukker opp, velg **Associate Floating IP**



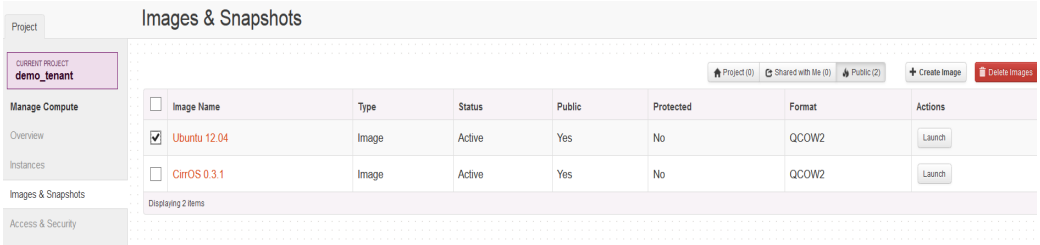
5. I vinduet som dukker opp, velges hvilken IP-adresse som skal kobles opp mot en gitt instans. Klikk så **Associate.**

Hvordan kan jeg opprette og slette et image?

1. Logg på OpenStack som vist tidligere i guiden
2. Under **Project**, velg **Images & Snapshots.**
3. Klikk **Create Image**



4. Her får vi opp en rekke alternativer for det imaget vi ønsker å opprette. Etter at de ønskede verdiene er skrevet inn, klikk **Create Image.** Deretter kan det gå litt tid før imaget blir lastet opp og får status **Active.**
5. Ønsker vi å slette et image det ikke lenger er behov for, går vi under **Project** inn på **Images & Snapshots.** Her markerer vi det/de imagene vi ønsker å slette, og klikker **Delete images** helt til høyre i kontrollpanelet.



Tillegg D

Prosjektplan

Prosjektplan

AutoStack

Kjetil André Finsrud
Anders Godtland Noem
Geir André Tufte

Januar 2014

Innhold

1 MÅL OG RAMMER **3**
1.1 Bakgrunn . . . 3
1.2 Prosjektmål (Effektmål og Resultatmål) . . . 4

2 OMFANG **5**
2.1 Avgrensning . . . 5
2.2 Oppgavebeskrivelse . . . 5

3 PROSJEKTORGANISERING **6**
3.1 Studentenes faglige bakgrunn . . . 6
3.2 Ansvarsforhold og roller . . . 6
3.3 Rutiner og regler i gruppa . . . 6
3.4 Verktøy . . . 7

4 PLANLEGGING, OPPFØLGING OG RAPPORTERING **8**
4.1 Hovedinndeling av prosjektet . . . 8
4.2 Plan for statusmøter og beslutningspunkter . . . 9
4.3 Ressursbehov . . . 9

5 ORGANISERING AV KVALITETSSIKRING **10**
5.1 Dokumentasjon, standardbruk og kildekode . . . 10
5.2 Risikoanalyse (identifisere, analysere, tiltak, oppfølging) . . . 10

6 PLAN FOR GJENNOMFØRING **12**

7 Kravspesifikasjon **13**
7.1 Use Case Diagram . . . 13
7.2 Use Case . . . 14
7.3 Kommentar til Use Case Diagram . . . 15
7.4 Operasjonelle krav . . . 15

8 Design **17**
8.1 Deployment view . . . 17
8.2 Kommentar til deployment view . . . 17

Vedlegg **19**
A Grupperegler . . . 19
B Gantt-skjema . . . 20

1 MÅL OG RAMMER

1.1 Bakgrunn

Ved Høgskolen i Gjøvik er det behov for å kunne tilby dataressurser til studenter ifm. bachelor/masteroppgaver, prosjekter etc. og ansatte ifm. emner hvor eksempelvis jobbing i virtuelle miljøer er aktuelt. Skyløsningen til høgskolen, SkyHiGh, tilbyr denne tjenesten i dag og dekker flere av disse behovene. I denne oppgaven vil vi se på hvordan en tilsvarende skyløsning kan settes opp og konfigureres automatisk. En slik automatisert utrulling av en OpenStack skyløsning kan utvides til å kunne tilby spesialtilpassede scenarier bestående av virtuelle maskiner som rulles ut i OpenStack-miljøet.

Enkelte aktører ved forskingssenteret, Center for Cyber- and Information Secuity (CCIS)[3], har belyst at de per dags dato ikke har noen enkel måte å bedrive IT-teknisk trening og/eller eksperimentering på. Her er det ønskelig at tilpassede scenarier kan opprettes og konfigureres etter spesifikke behov. En slik tjeneste finnes allerede i dag[1] hvor tjenestetilbyderen har opprettet lab-miljøer for å kunne utføre trening og eksperimenter.

Bacheloroppgaven har med dette en relevans med et reelt behov hvor vi kan tilby en heltautomatisert utrulling av et OpenStack-miljø hvor eksisterende lab-miljøer/scenarier kan implementeres. Vi som tilbydere av en slik løsning kan på én dag rulle ut et slikt miljø hvor ønskede scenarier er tilgjengelig. En slik løsning vil også kunne tilby resirkulering av utdaterte servere og maskinvare som ikke lenger er i bruk men som nå kan implementeres som en del av OpenStack-miljøet.

1.2 Prosjektmål (Effektmål og Resultatmål)

Effektmål

- HiG ønsker å oppnå en automatisk detektering og utrulling av OpenStack på sin SkyHiGh skyløsning.
- Minimere manuelt arbeid som kreves for å konfigurere ny maskinvare i SkyHiGh.
- HiG ønsker å tilby enda flere og bedre tjenester til skolens ansatte og studenter.
- Øke HiGs bidrag til open-source miljøet.
- Automatisert utrulling av OpenStack-miljøer til bruk i tilpassede scenarier gitt av bruker.

Resultatmål

- Automatisk implementasjon av donerte servere og annen maskinvare til bruk i et OpenStack-miljø.
- Automatisert utrulling av et OpenStack-miljø.
- Adaptiv ressurstildeling.

2 OMFANG

2.1 Avgrensning

Prosjektet søker å oppnå en automatisert prosess med tilegning av ressurser ut i fra den maskinvaren som er tilgjengelig, og videre de punktene beskrevet i oppgavebeskrivelsen. Vi vil ikke legge fokus på lagring, backup og recovery i OpenStack, da disse emnefeltene dekkes av en annen bachelorgruppe. Videre vil vi i implementasjon av overvåkningen være bundet til Munin og Nagios.

2.2 Oppgavebeskrivelse

Oppgaven omhandler hvor automatisert en utrulling av en OpenStack skyløsning kan gjøres. Dette innebærer distribusjon av operativsystem og systemkonfigurasjon slik at alle nodene OpenStack består av etableres og konfigureres automatisk. Detaljert overvåking av disse nodene skal også rulles ut som del av automatiseringen.

Høgskolen i Gjøvik får opptil flere ganger i året donert servere fra regionale bedrifter. Oppgaven vil med dette også innebære hvordan disse serverene raskt kan settes inn et OpenStack-miljø og automatisk konfigureres og bli implementert som en ressurs i skyløsningen.

3 PROSJEKTORGANISERING

3.1 Studentenes faglige bakgrunn

Studentene i dette prosjektet kommer fra to ulike studieretninger innenfor IT. Geir André Tufte og Anders Godtland Noem studerer drift av nettverk og datasystemer. Kjetil André Finsrud studerer ingeniørfag data.
Alle har tilegnet seg kunnskaper innenfor emner som programmering, databaser og systemutvikling, og har her en felles plattform som kommer til nytte i prosjektet. Videre har studentene ved driftsstudiet tilegnet seg kunnskaper innenfor ulike områder av systemadministrasjon, mens Kjetil har en fagskolebakgrunn med liknende fagområder som de driftsstudiet legger vekt på.

3.2 Ansvarsforhold og roller

Oppdragsgiver i dette prosjektet er førsteamanuensis Erik Hjelmås. Han er den som sitter med ekspertkompetansen på området, og vil i utgangspunktet være den viktigste informasjonskilden under prosjektperioden når vi møter på utfordringer.
Veileder er førsteamanuensis Hanno Langweg. Han har erfaring fra tidligere med bacheloroppgaver i OpenStack, og vil derfor i tillegg til generell veiledning under prosjektperioden være en ypperlig inspirasjonskilde.
Prosjektleder for gruppen er Geir André Tufte. Han vil i tillegg være kontaktperson.
Videre fordeles oppgaver dynamisk gjennom prosjektperioden, slik at alle gruppens medlemmer oppnår kunnskaper og erfaringer innenfor de ulike modulene, samtidig som det legges fokus på effektivitet i arbeidet.

3.3 Rutiner og regler i gruppa

Vi har i gruppereglene fastsatt et minimum på 25 timer i uka med jobbing på prosjektet. Det er planlagt at mesteparten av tiden skal brukes sammen på skolen, men dette vil til tider være vanskelig ettersom vi alle tre har forskjellige fag ved siden av bacheloroppgaven. Lørdager og søndager er i utgangspunktet fridager men vil bli benyttet om nødvendig. Det er viktig å påpeke at minimumsantallet timer i uken ikke reflekterer det faktiske antallet timer vi vil arbeide i en gjennomsnittlig uke. Dette gir oss fleksibilitet i de ukene hvor det kreves ekstra innsats på valgfag eller andre situasjoner som kan dukke opp. Vi regner med at timeantallet kommer til å overstige 30 timer i uken. Vårt primære arbeidssted vil være serverrom/grupperom, K202, ettersom vi har alt utstyret vårt der. Hele gruppereglementet finnes i vedlegg A.

3.4 Verktøy

Gruppen har valgt å benytte seg av ShareLaTeX som hovedkilde for dokumentasjon av prosjektet. Her er dataene sikret, mens Dropbox gir oss sikkerhetskopiering av alle data vi benytter oss av i prosjektet, som ikke direkte benyttes i ShareLaTeX-dokumentene. I tillegg har gruppen konfigurert SVN for å ha et alternativ til for konfigurasjonsstyring.

For kommunikasjon innad i gruppen, samt med veileder, vil Skype i stor grad være det som benyttes. Gruppen internt møtes i stor grad daglig, og vi ser derfor ikke behov for ytterligere kommunikasjonsmidler under samarbeidet.

Virtualbox og VMware benyttes lokalt på våre laptoper til installasjon og testing av programvare, før det settes i produksjon på serverene. Vi prøver å holde det lokale virtuelle miljøet så likt produksjonsmiljøet som mulig.

For dokumentasjon, koding/scripting, modellering og liknende, benyttes i all hovedsak MS Office Project 2010, MS Office Excel, Vim, Putty og Notepad++.

4 PLANLEGGING, OPPFØLGING OG RAPPORTERING

4.1 Hovedinndeling av prosjektet

Det er blitt vurdert flere systemutviklingsmodeller for utvikling av denne løsningen. Siden ingen modell gir oss alle kriteriene vi trenger, har vi valgt å kombinere egenskaper til flere modeller. En slik tilpassing av utviklingsmodell vil gi økt effektivitet og fleksibilitet.

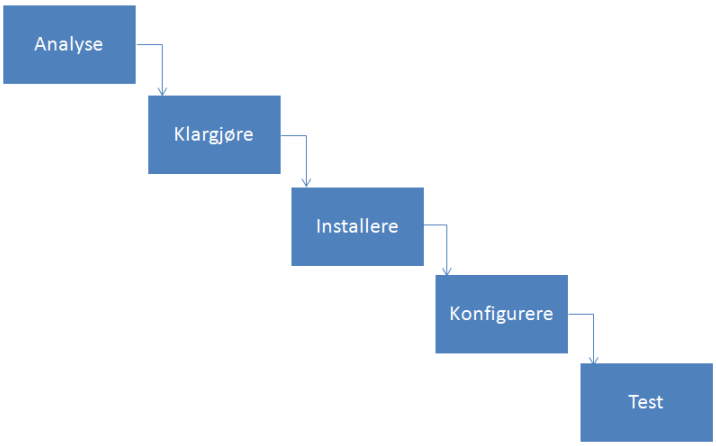
Utviklingen av oppgaven må tillate oss å kunne gå tilbake på tidligere arbeid og utføre endringer på områder som har vist seg å ikke være tilfredsstillende, eller moduler som er inkompatible i forhold til andre deler av løsningen. Vi ønsker å fortløpende dokumentere arbeidet som gjøres, noe utviklingsmodellen vi velger må tillate.

Programvaren som skal brukes til oppgaven er kjent, men det kreves at vi bruker tid til å undersøke i dybden hvordan hver av disse skal installeres samt testes. Det vil bli vanskelig å sette et tidsestimat på implementasjon av disse programvarene, derfor vil en modell som ikke har et fast bestemt arbeidsintervall være fordelaktig. En modulbasert tilnærming på denne delen av oppgaven virker naturlig.

Scripting vil også bli en stor del av oppgaven, og er en fase som starter etter at programvaren er implementert. Denne fasen vil bestå av mye uforutsigbarhet og vil kreve kortsiktig planlegging.

Med dette har vi valgt å bruke en sekvensiell/inkrementell utviklingsmodell under programvareimplementasjonen. Under skriptingen har vi valgt å bruke Scrum hvor vi setter opp korte sprinter for lettere kunne håndtere uforutsigbarheter. Parallellt med de modellene som er valgt, vil vi også benytte utviklingsmodellen samtidige-aktiviteter. Dette for å kunne gå tilbake på tidligere arbeid for å videre tilpasse systemkonfigurasjonen.

Figur 1: Utviklingsmodell til AutoStack



Figur 2: Nærmere beskrivelse av vår sekvensielle inkrementelle modell

4.2 Plan for statusmøter og beslutningspunkter

I løpet av hele oppgaveperioden kommer vi til å ha jevnlige møter med både veileder og oppdragsgiver. Vi kommer til å ha gruppemøte på fredags morgen hver uke under den perioden vi bruker sekvensiell inkrementell modell. I den siste delen av prosjektet hvor vi skal bruke Scrum kommer vi til å ha 15 minutters daglig møte på morgenen (daily scrum) for å gå igjennom hva som er gjort, hva som skal gjøres og hva som gjenstår i iterasjonen. Sprint review vil forekomme etter hver iterasjon.

4.3 Ressursbehov

Denne oppgaven vil kreve ressurser som i liten eller ingen grad er knyttet opp mot økonomiske ytelser. Maskinvare og programvare som benyttes er enten bevilget av skole/oppdragsgiver, eller open-source. I forbindelse med serverne, stiller ikke oppgaven noen spesielle krav til disse, da fokuset er på å oppnå en så automatisk utrulling som mulig, og ikke på ytelsesaspektet.

Programvare: Cobbler, Puppet, OpenStack, Munin, Nagios.

Maskinare: 6 Servere bevilget av oppdragsgiver, hvorav én av disse fungerer som puppetmaster og cobblerserver, og resterende servere vil settes opp med OpenStack.

5 ORGANISERING AV KVALITETSSIKRING

5.1 Dokumentasjon, standardbruk og kildekode

Dokumentasjon gjennom hele prosjektperioden vil hovedsakelig foregå i LaTeX, og vi har valgt å benytte oss av ShareLaTeX som gjør at vi kan arbeide samtidig på et dokument gjennom en nettleser. Utgangspunktet vi arbeider ut i fra er at alt skal dokumenteres, dette innebærer også kommentering av koding/scripting. Ved kildehenvisning benyttes Vancouver-stilen.
Vi har i utarbeidelsen av prosjektplanen benyttet oss av SkyHiGh ADM[2] til inspirasjon, og vurdering av hva som burde være med av innhold i rapporten.

5.2 Risikoanalyse (identifisere, analysere, tiltak, oppfølging)

Sannsynlighetsgraden deles opp i 3 deler: Lite sannsynlig, sannsynlig, og veldig sannsynlig.

Punktet 'Mangel på kompetanse' henviser til spisskompetanse på områdene vi imøtekommer i dette prosjektet. Oppgaven krever at vi må tilegne oss ny kompetanse fortløpende, noe som kan bli svært tidkrevende da vi ikke har tilstrekkelig kompetanse på disse områdene. Krever dette punktet for mye ressurser av oss, kan dette påvirke andre, viktige deler av prosjektet.

Risikoanalyse				
	Beskrivelse	**Sannsynlighet**	**Konsekvens**	**Tiltak**
1	Sykdom	Sannsynlig	Redusert bemanning	Ekstra innsats
2	Mangel på kompetanse	Sannsynlig	Dårligere løsninger/ ustandardiserte løsninger	Godt forarbeid
3	Maskinvare ryker	Lite sannsynlig	Tap av data og utstyr	Backup av data
4	Gruppemedlem slutter/avskjediges	Lite sannsynlig	Reduserte ressurser	Tilpassning av oppgave, ekstra innsats
5	Uforutsigbare tidsestimater	Veldig sannsynlig	Holder ikke interne tidsfrister	Godt forarbeid
6	Programvare inkompatibilitet	Sannsynlig	Avvik fra opprinnelig plan	Godt forarbeid
7	Tap av data	Lite sannsynlig	Bortkastet tid	Backup og gode rutiner

Tabell 1: Risikoanalyse

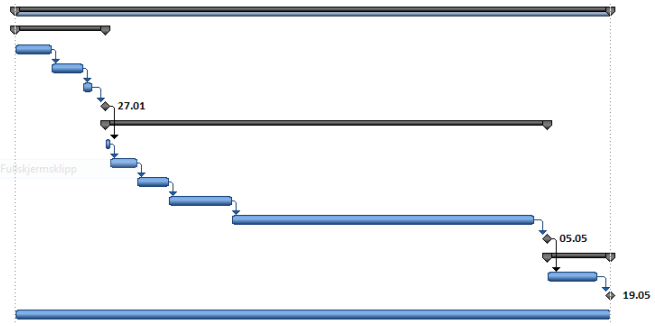
Ved å gjøre et godt forarbeid med å undersøke de aktuelle programvarene vi skal benytte i oppgaven, kan vi redusere sannsynligheten for at det oppstår problematikk rundt nødvendig kompetanse. Et slikt forarbeid vil også til en viss grad gi økt forståelse for omfanget ved installasjon og implementasjon av aktuell programvare, som gjør at vi bedre kan sette riktige tidsestimater.

Maskinvare som ryker vil alltid være en risikofaktor, men vi har ikke ressurser til å sikre oss mot dette. De tiltakene vi kan gjøre her er å sikre oss mot datatap. Vi sørger for å lagre mest mulig av dataene i skyløsninger, og vil derfor merke lite problemer på dette området.

Gruppemedlemmer som slutter eller blir avskjediget, ser vi på som lite sannsynlig. Gruppens medlemmer kjenner hverandre og har en god kjemi fra tidligere, som gjør at vi er godt sikret mot dette. Skulle problemer likevel oppstå, vil det fortsatt være mulig å fullføre prosjektet etter noen justeringer.

6 PLAN FOR GJENNOMFØRING

Bacheloroppgave	**94 days**	**Wed 08.01.14**	**Mon 19.05.14**
Forprosjekt	**14 days**	**Wed 08.01.14**	**Mon 27.01.14**
Prosjektplanlegging	6 days	Wed 08.01.14	Wed 15.01.14
Kravspesifisering	5 days	Thu 16.01.14	Wed 22.01.14
Designdokument	2 days	Thu 23.01.14	Fri 24.01.14
Ferdigstilt forprosjekt	1 day	Mon 27.01.14	Mon 27.01.14
Utvikling	**70 days**	**Tue 28.01.14**	**Mon 05.05.14**
Oppsett av hardware	1 day	Tue 28.01.14	Tue 28.01.14
Installere Cobbler	4 days	Wed 29.01.14	Mon 03.02.14
Installere Puppet	5 days	Tue 04.02.14	Mon 10.02.14
Installere OpenStack	10 days	Tue 11.02.14	Mon 24.02.14
Scripting	49 days	Tue 25.02.14	Fri 02.05.14
Ferdigstilt produkt	1 day	Mon 05.05.14	Mon 05.05.14
Rapport	**10 days**	**Tue 06.05.14**	**Mon 19.05.14**
Rapportskriving	9 days	Tue 06.05.14	Fri 16.05.14
Ferdigstilt bacheloroppgave	1 day	Mon 19.05.14	Mon 19.05.14
Utarbeidelse av rapport	94 days	Wed 08.01.14	Mon 19.05.14



Figur 3: Ganttskjema (for større versjon se vedlegg B)

I dette prosjektet vil effektiv tidsforbruk være essensielt for at gruppen skal besvare oppgaven på en best mulig måte. Grunnlaget for vår vurdering her er i stor grad basert på tidligere erfaringer, valg av utviklingsmodell, samt beskrivelsen av oppgaven vi er blitt tildelt. Vi har lagt inn ekstra tid i hver del av prosjektet for få et lite buffer mot overskridelse av tidsskjemaet vi har.

De tre initielle modulene tar for seg implementasjon av programvare vi skal benytte. Det finnes mye god dokumentasjon og veiledninger tilgjengelig på internett for denne programvare. Vi ser derfor ikke på denne delen av oppgaven som veldig utfordrende.

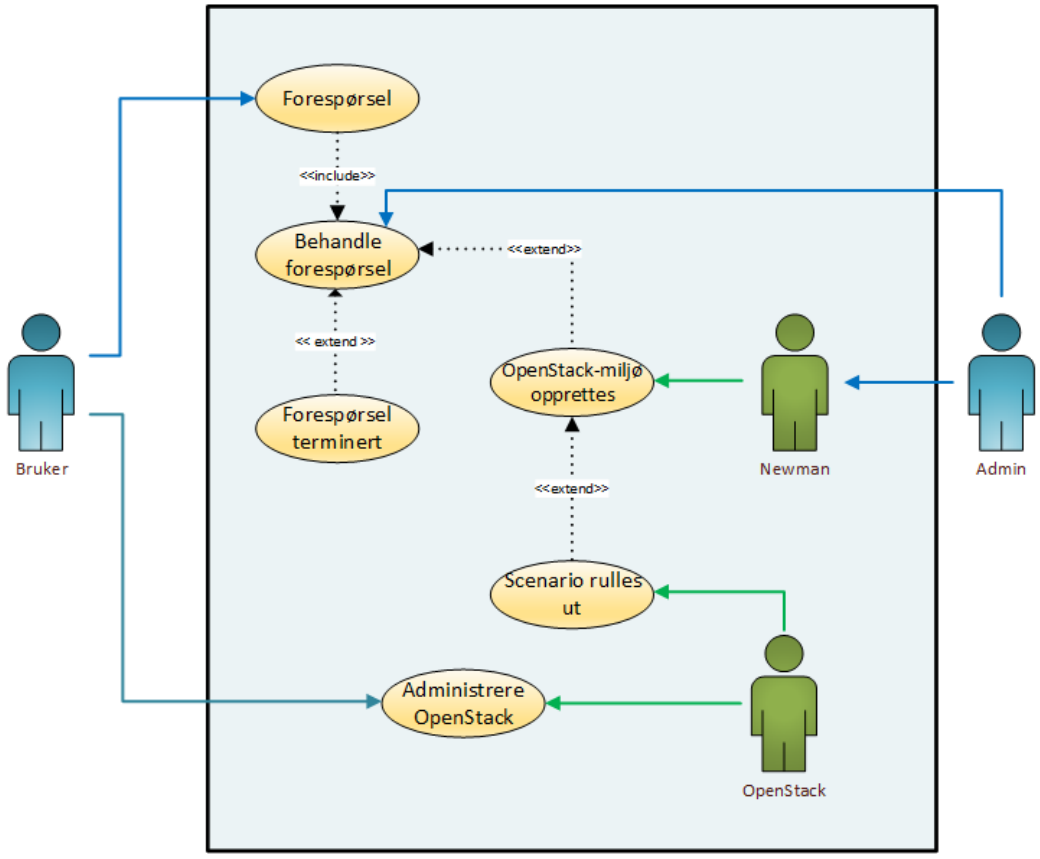
Av disse ser vi på implementasjon av OpenStack som et litt usikkert område, samtidig som den er mer omfattende, og setter derfor av litt lengre tid her enn de andre. Det ligger allikevel også her mye god dokumentasjon tilgjengelig, og vi ser ingen større problemer med denne heller.

Den påfølgende fasen med scripting er det store usikkerhetsmomentet i dette prosjektet. Her vet vi lite om hvor mange utfordringer vi kommer til å møte på, samtidig som vi har lite erfaring på området. Det er derfor blitt avsatt veldig mye tid her i forhold til de andre modulene. Fullfører vi denne fasen godt innenfor tidsfristen kan det her være aktuelt å forsøke å legge til ekstra funksjonalitet.

Vi har satt av de ti siste dagene til rapportskriving. Dokumentasjon og rapportskriving kommer til å være aktuelt fra første dag og videre utover i prosjektet. En periode på slutten vil hovedsakelig benyttes for å sy sammen rapporten og finpussing.

7 Kravspesifikasjon

7.1 Use Case Diagram



Figur 4: Use case diagram

7.2 Use Case

Use Case	Forespørsel
Aktør	Bruker
Hensikt	Opprette kontakt med Admin
Beskrivelse	Lar brukere sende forespørsler om å få satt opp et testmiljø.

Use Case	Behandle forespørsel
Aktør	Admin
Hensikt	Vurdere om forespørselen skal godtas eller avslås.
Beskrivelse	Admin tar stilling til om denne forespørselen er mulig, og eventuelt ønskelig å godta.

Use Case	Scenario rulles ut
Aktør	Admin, Newman, OpenStack
Hensikt	Implementere et spesifikt scenario.
Beskrivelse	Basert på eksisterende lab-miljøer med intensjon om trening/testing/eksperimentering, rulles et slikt scenario ut i OpenStack-miljøet.

Use Case	Administrere OpenStack
Aktør	Bruker, OpenStack
Hensikt	Administrering av det opprettede OpenStack-miljøet.
Beskrivelse	Brukeren av systemet administrerer OpenStack-miljøet etter eget ønske. Brukeren får overlevert all nødvendig dokumentasjon for å kunne utføre dette.

7.3 Kommentar til Use Case Diagram

Dette diagrammet viser hvordan denne løsningen hovedsakelig har to brukergrupper. 'Bruker' er en betegnelse på de som vil være aktuelle til å benytte seg av løsningen.

'Behandle forespørsel' har to utfall. Enten har Admin mulighet til å levere løsningen og prosessen 'OpenStack-miljø opprettes' iverksettes. Det andre utfallet kan være at Admin ikke er istand til å levere det brukeren førespør, som vil føre til at forespørselen termineres.

- Løsningen skal kunne rulle ut et OpenStack-miljø.
- Løsningen skal kunne rulle ut forhåndsdefinerte virtuelle miljøer (scenarier).
- Løsningen skal kunne detektere maskinvare som blir installert, og skal automatisk implementere dette som en del av OpenStack-miljøet.

7.4 Operasjonelle krav

Brukervennlighet

OpenStack-miljøet som rulles ut skal for en bruker komme i to varianter. Første variant vil være et rent OpenStack-miljø klargjort for at brukeren selv kan bestemme oppsett og konfigurasjon. Andre variant vil være lik den første, men med prekonfigurerte virtuelle maskiner som utgjør et spesifikt scenario hvor brukeren selv kan tilpasse scenariet etter eget behov.

Bruken av systemet vil i stor grad være avhengig av kompetansen som ligger hos brukergruppen. Løsningen vil bestå av kun open-source programvare og har med det mye dokumentasjon tilgjengelig på internett.

Deler av løsningen kan gjenbrukes i HiG sin skyløsning, SkyHiGh, hvor maskinvare som blir donert til høgskolen kan implementeres i skyløsningen, automatisk, med få grep.

Ytelse

Vi har i dette prosjektet få eller ingen krav til ytelse, da hovedfokus vil være på å kunne sette opp systemet automatisk. Allikevel vil det i fremtiden være aktuelt å definere krav her, da dette har mer å si for utnyttelse av systemet, og vi har en brukergruppe som vil komme til å stille krav opp mot dette. Oppdragsgiver ønsker at vi i et tenkt scenario skal komme til en bedrift, koble opp vår Newman server og rulle ut det virtuelle OpenStack-miljøet i løpet av én arbeidsdag.

Ved å benytte de deler av oppgaven som blir relevant for HiG sin skyløsning, SkyHiGh, vil donert maskinvare som høgskolen får, raskt kunne bli implementert i skyløsningen og dermed øke ytelsen i form av større ressurskapasiteter.

Sikkerhet

Det vil være en naturlig sikkerhet i systemet brukeren benytter seg av, ettersom et virtuelt miljø blir som en sandkasse å regne. Det er videre ønskelig med muligheter for å nå ut til internett, men dette forutsetter at man klarer å sikre dataflyten i testmiljøet, slik at det ikke er mulig å sende for eksempelvis malware ut på nettet, avhengig av scenario.

Systemet vårt vil også benytte Keystone, som følger med som en autentisering,- og identifiseringstjeneste i OpenStack.

Dokumentasjon

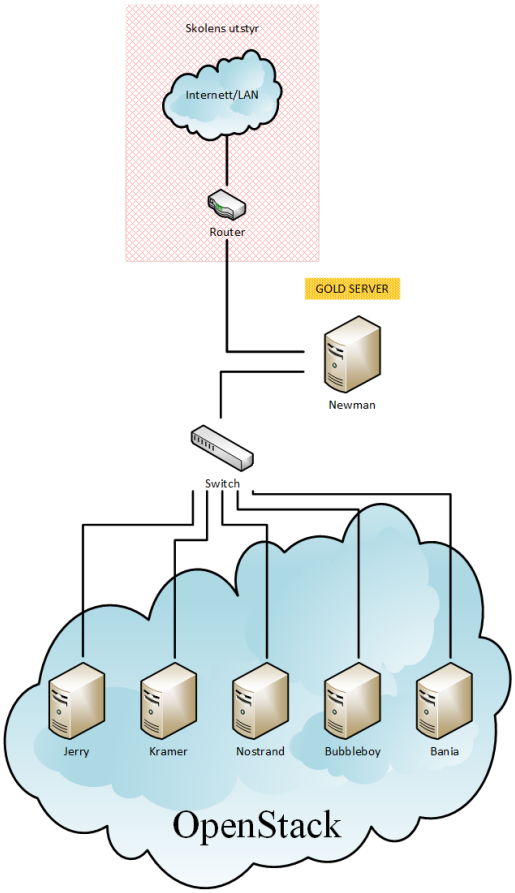
Løsningen, det være seg OpenStack-miljø eller et scenario implementert i et OpenStack-miljø, må dokumenteres på en slik måte at brukeren av løsningen på en lettvint måte skal kunne finne svar på spørsmål rundt systemkonfigurasjon. Vi vil i forbindelse med dette prosjektet stå for dokumentasjon av løsninger, feil som oppstår, og annen relevant informasjon som dukker opp underveis. Etter prosjektets slutt fraskriver vi oss alt ansvar for videre vedlikehold.

Tilgjengelighet

Et nøkkelpoeng med denne løsningen er at den skal være tilgjengelig for den brukergruppen som har bestilt løsningen for sitt miljø, og kun der. Dette bør løses greit ut i fra systemet som er tilrettelagt hos kunden før oppsett av løsningen.

8 Design

8.1 Deployment view



Figur 5: Deployment view

8.2 Kommentar til deployment view

Vi kommer til å ha 6 servere og 1 switch + kabler som skal utgjøre det vi kommer til å bruke av maskinvare i løpet av prosjektperioden. Serverene Jerry, Kramer, Nostrand, Bubbleboy og Bania skal utgjøre OpenStack-miljøet vårt som danner en skyløsning. Newman vil være gold-server, som vil si at den skal stå for utrulling av hele løsningen. Den vil også fungere som en router for vårt lokale nett ut mot skolens offentlige nett og internett. Newman vil også detektere maskinvare og implementere disse i OpenStack-miljøet. Switchen vil være den som kobler sammen alle våre servere sammen i et lokalt nett. Internett, skolens router og skolens lokale nettverk vil ikke være et fokusområde for dette prosjektet.

Referanser

[1] Mikael Wedlin, Teodor Sommestad.
CRATE - Cyber Range And Training Environment.
http:
//www.foi.se/en/Our-Knowledge/Information-Security-and-Communication/
Information-Security/Lab-resources/CRATE/

[2] Lars Erik Pedersen, Jon Arne Westgaard, Hallvard Westman.
SkyHiGh ADM. http://brage.bibsys.no/hig/bitstream/URN:NBN:
no-bibsys_brage_30474/3/LEPedersen_JAWWestgaard_HAWestman.pdf

[3] Christian Meyer.
CCIS - Center for Cyber- and Information Security
https://norsis.no/2013/06/center-for-cyber-and-information-security

Vedlegg

A Grupperegler

Grupperegler

1. Hvis uenigheter oppstår, løses disse i første omgang ved demokratisk avstemning. Dersom dette ikke fører frem, benyttes veileder samt oppdragsgiver for å finne en tilfredsstillende løsning.
2. Geir André Tufte er prosjektleder gjennom hele oppgaven.
3. Antall arbeidstimer per uke settes til et minimum på 25 timer. Unntak fra dette avtales innad i gruppen.
4. Et hvert gruppemedlem kan signere på vegne av gruppen, såfremt minst ett av de andre medlemmene er informert og har samtykket.
5. Kostnader og utgifter med bacheloroppgaven deles likt på gruppemedlemene.
6. Ved sykdom skal dette meldes fra så fort som mulig, helst med forventet estimat på når friskmeldt.
7. Alle er pliktige til å møte til avtalte tidspunkter. Skulle det være behov for fritak avtales dette med resten av gruppen så fort som mulig.
8. Sabotasje er definert som et alvorlig brudd på gruppereglene.
9. Ved gjentatte og/eller alvorlig brudd på gruppereglene beskrevet ovenfor vil det bli innkalt til et møte med veileder for en eventuell utkastelse.

Geir André Tufte

Geir André Tufte Gjøvik 24/1 - 14

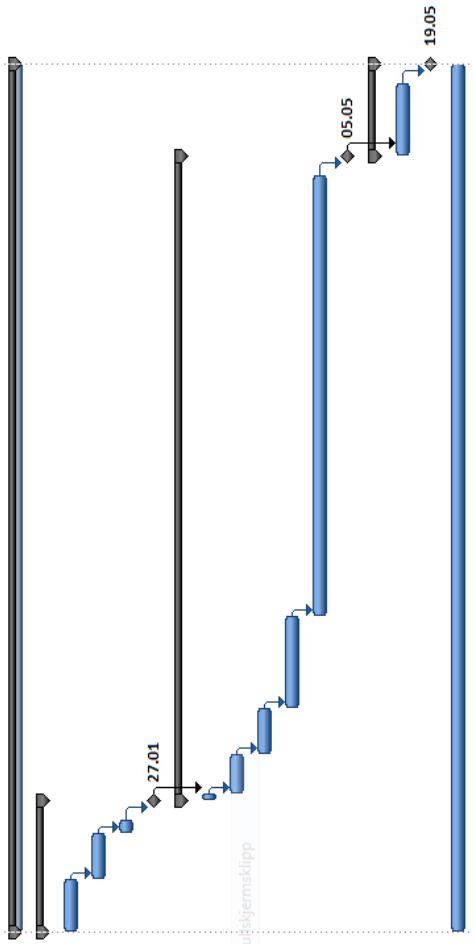
Anders Godtland Noem

Anders Noem Gjøvik 24/1-14

Kjetil André Finsrud

K.A. Finsrud Gjøvik 24/1 - 14

B Gantt-skjema



Bacheloroppgave	94 days	Wed 08.01.14	Mon 19.05.14
Forprosjekt	14 days	Wed 08.01.14	Mon 27.01.14
Prosjektplanlegging	6 days	Wed 08.01.14	Wed 15.01.14
Kravspesifisering	5 days	Thu 16.01.14	Wed 22.01.14
Designdokument	2 days	Thu 23.01.14	Fri 24.01.14
Ferdigstilt forprosjekt	1 day	Mon 27.01.14	Mon 27.01.14
Utvikling	70 days	Tue 28.01.14	Mon 05.05.14
Oppsett av hardware	1 day	Tue 28.01.14	Tue 28.01.14
Installere Cobbler	4 days	Wed 29.01.14	Mon 03.02.14
Installere Puppet	5 days	Tue 04.02.14	Mon 10.02.14
Installere OpenStack	10 days	Tue 11.02.14	Mon 24.02.14
Scripting	49 days	Tue 25.02.14	Fri 02.05.14
Ferdigstilt produkt	1 day	Mon 05.05.14	Mon 05.05.14
Rapport	10 days	Tue 06.05.14	Mon 19.05.14
Rapportskriving	9 days	Tue 06.05.14	Fri 16.05.14
Ferdigstilt bacheloroppgave	1 day	Mon 19.05.14	Mon 19.05.14
Utarbeidelse av rapport	94 days	Wed 08.01.14	Mon 19.05.14

Tillegg E

E-post

IP adresser

Current Folder: **INBOX** **Sign Out**

Compose Addresses Folders Options Search Help Filters Gjovik University College

Search Results | Unread | Delete Forward | Forward as Attachment | Reply | Reply All

Subject: IP adresser
From: Erik Hjelmås <erikh@hig.no>
Date: Fri, February 7, 2014 12:29
To: "Geir Andre Tufte" <geir.tufte@hig.no> (more)
Priority: Normal
Create Filter: Automatically | From | To | Subject
Options: View Full Header | View Printable Version | Download this as a file

```
Mens vi er på emnet IP-addresser, jeg har reservert to IP adresser til
dere, det er

128.39.141.217 (seinfeld.hig.no)
128.39.141.218 (ikke noe dns navn)

dvs controllern i openstack må ha en public ip, det er .217, mens dere
kan bruke .218 som aksess til det interne nettet av VMer (dvs siden vi
har så få IP adresser må dere gjemme private ip adresser til VMene bak .218)


/Erik
```

Openstack Neutron

Gmail by Google

Kjetil André Finsrud <kafinsrud@gmail.com>

OpenStack Neutron

6 e-poster

Geir Andre <geirtufte@gmail.com> 26. februar 2014 kl. 08.52
Til: Erik Hjelmås <erikh@hig.no>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Hei

Vi får nå launchet instanser og de får IP-addresse, hvertfall ifølge Horizon. Men, vi får ikke pinget maskinene og når vi går inn på consollet har ikke maskinen noen IP-addresse. Når vi verifiserer nettverkskonfigurasjonen på instansen og restarter networking, får ikke instansen svar på dhcp-req. Det virker som neutron ikke svarer på dhcp-reqs men vi har ikke konfigurert noe i dhcp-konfigurasjonsfilen på neutron heller (iht. folsome-guide).

Har du muligheten til å sende oss skyHiGh sin dhcp_agent.ini på neutron?

Jeg tror også vi kunne trengt litt hjelp til selve nettverksoppsettet, så når du har tid, må du gjerne komme innom k202.

Mvh
Geir A.T.

Erik Hjelmås <erikh@hig.no> 26. februar 2014 kl. 09.27
Til: Geir Andre <geirtufte@gmail.com>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Glimrende jobba,

Tror ikke jeg rekker se på dette før i ettermiddag men kanskje

E

-------- Opprinnelig melding --------
Fra: Geir Andre
Dato:26.02.2014 08:52 (GMT+01:00)
Til: Erik Hjelmås
Ko: andersnoem90@gmail.com,kafinsrud@gmail.com
Emne: OpenStack Neutron

[Sitert tekst skjult]

Geir Andre <geirtufte@gmail.com> 3. mars 2014 kl. 08.12
Til: Erik Hjelmås <erikh@hig.no>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

God morgen!

Kunne du sendt meg dhcp_agent.ini filen på skyHiGh sin neutron, når du får tid?
Vi kunne også trengt litt hjelp med nettverksoppsettet.

Mvh
Geir
[Sitert tekst skjult]

Erik Hjelmås <erikh@hig.no> 3. mars 2014 kl. 09.31
Til: Geir Andre <geirtufte@gmail.com>, andersnoem90@gmail.com, kafinsrud@gmail.com

Openstack Neutron

foresten, de to andre filene i Neutron mappa som kanskje også er interesssante er:

root@steinbrenner:/etc/neutron# cat metadata_agent.ini | egrep -v '^#'

[DEFAULT]
debug = True

auth_url = http://chiles:5000/v2.0
auth_region = regionOne
admin_tenant_name = service
admin_user = neutron
admin_password = X
nova_metadata_ip = chiles
metadata_proxy_shared_secret = X

nova_metadata_port = 8775

root@steinbrenner:/etc/neutron# cat l3_agent.ini | egrep -v '^#'

[DEFAULT]

interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
use_namespaces = True

external_network_bridge = br-ex

/Erik

PS! om jeg ikke har oppgitt det før så er altså i SkyHiGh:
chiles er contoller
steinbrenner er network
compute01, 02, 03, 04 er computenodene

Den 03. mars 2014 09:21, skrev Erik Hjelmås:

> Den 03. mars 2014 08:12, skrev Geir Andre:
>
> > God morgen!
> >
> > Kunne du sendt meg dhcp_agent.ini filen på skyHiGh sin neutron, når du får tid?
>
> er faktisk bare fire ikke-bortkommenterte linjer i denne fila:
>
> root@steinbrenner:/etc/neutron# cat dhcp_agent.ini | egrep -v '^#'
>
> [DEFAULT]
>
> interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver
> use_namespaces = True
>
> dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq

Openstack Neutron

> Vi kunne også trengt litt hjelp med nettverksoppsettet.

jeg regner med å ha tid mellom 9 og 13 imorgen tirsdag

/Erik
[Sitert tekst skjult]

Geir Andre <geirtufte@gmail.com> 3. mars 2014 kl. 12.16
Til: Erik Hjelmås <erikh@hig.no>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Innholdet i disse filene ser veldig riktig ut og vi har oppdatert våre filer nå. Nå har vi problemer med servicen **openvswitch-switch**.

Har du muligheten til å paste innholdet i;
/etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
på deres Neutron, når du får tid?

Geir
[Sitert tekst skjult]

Erik Hjelmås <erikh@hig.no> 4. mars 2014 kl. 05.53
Til: andersnoem90@gmail.com, kafinsrud@gmail.com
Kopi: Geir <geirtufte@hotmail.com>

sorry glemte CC dere

-------- Opprinnelig melding --------
Emne: Re: OpenStack Neutron
Dato: Tue, 04 Mar 2014 05:51:32 +0100
Fra: Erik Hjelmås <erikh@hig.no>
Til: Geir Andre <geirtufte@gmail.com>

Den 03. mars 2014 12:16, skrev Geir Andre:
> Innholdet i disse filene ser veldig riktig ut og vi har oppdatert våre
> filer nå. Nå har vi problemer med servicen *openvswitch-switch*.
>
> Har du muligheten til å paste innholdet i;
> /etc/neutron/plugins/openvswitch/ovs_neutron_plugin.ini
> på deres Neutron, når du får tid?

```
# cat ovs_neutron_plugin.ini | egrep -v '^#'
[ovs]

tenant_network_type = vlan
network_vlan_ranges = default:2000:2099
bridge_mappings = default:br-em2


[agent]

[securitygroup]
firewall_driver = neutron.agent.firewall.NoopFirewallDriver

[database]
connection = mysql://neutron:X@10.10.10.51/neutron
```

Openstack Neutron

/E
[Sitert tekst skjult]

Utrulling av Munin

Kjetil André Finsrud <kafinsrud@gmail.com>

Fwd: SV: Utrulling av Munin

1 e-post

Geir Andre <geirtufte@gmail.com> 12. mai 2014 kl. 09.32
Til: "kafinsrud@gmail.com" <kafinsrud@gmail.com>, "andersnoem90@gmail.com" <andersnoem90@gmail.com>

-------- Original Message --------
Subject:SV: Utrulling av Munin
Date: Tue, 1 Apr 2014 16:46:32 +0200
From: Erik Hjelmås <erikh@hig.no>
To: Geir Andre <geirtufte@gmail.com>

Bare nedprioriter munin, fokuser på OpenStack puppet modulene

E

-------- Opprinnelig melding --------
Fra: Geir Andre
Dato:01.04.2014 14:56 (GMT+01:00)
Til: Erik Hjelmås
Emne: Utrulling av Munin

Hei!

Vi trenger å vite i hvilken grad Munin skal rulles ut i bacheloroppgaven.
Er det tanken at munin skal rulles ut på virtuelle maskiner som
opprettes i OpenStack, eller er det mening av det skal rulles ut på de
fysiske serverene som hoster OpenStack, eller begge deler?

Geir

Siste Openstack-finish

Kjetil André Finsrud <kafinsrud@gmail.com>

Siste openstack-finish

5 e-poster

Geir Andre <geirtufte@gmail.com> 11. april 2014 kl. 10.35
Til: Erik Hjelmås <erikh@hig.no>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Hei

Vi har siste uken hatt fullt fokus på å gå gjennom Openstack sin egen dokumentasjon for å installere Openstack. Vi blei ferdig med å automatisere denne prosessen i går kveld og fikk testet litt med å kjøre opp noen CirrOS VM'er. VM'en kommer opp og vi får console-tilgang og ser i loggen at VMen sender ut dhcp-discovers men den får ikke noe svar. I selve Openstack oppsettet, ser vi at VMen har fått tildelt en ip-addresse, men leveransen av ip-addressen går ikke i orden. Vi mistenker at dette kan være nettverkskonfigurasjonen på kontrollernoden og nettverksnoden.

Eksempelvis her i pkt. 3;
http://docs.openstack.org/havana/install-guide/install/apt/content/install-neutron.install-plug-in.ovs.vlan.html
Det vi ikke skjønner er hvilken ip-addresse bridge interface skal ha, fordi det fysiske interfacet har jo aldri hatt noen ip-addresse.
Samme gjelder for br-ex interfacet, se pkt. 5 her;
http://docs.openstack.org/havana/install-guide/install/apt/content/install-neutron.install-plug-in.ovs.html

Hvis du ikke umiddelbart kan svare på dette, kunne du fortsatt ha sendt nettverkskonfigen som blir brukt på kontroller,nettverk og èn compute node i skyhigh? Jeg tror det kan løse mye av problemet vårt.

Mvh
Geir

Erik Hjelmås <erikh@hig.no> 14. april 2014 kl. 09.23
Til: Geir Andre <geirtufte@gmail.com>
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Den 11. april 2014 10:35, skrev Geir Andre:
[Sitert tekst skjult]
sorry, jeg rekker ikke annet enn å sende dere configen, si ifra hvis det ikke hjelper, jeg tror jeg skal ha noe ledig tid mot slutten av uka (tors/fre)

/Erik

config.txt
8K

Erik Hjelmås <erikh@hig.no> 14. april 2014 kl. 09.25
Til: Kyrre Matthias Begnum <kyrre.begnum@hig.no>
Kopi: Geir Andre <geirtufte@gmail.com>, "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Kyrre, se de to spm om br-ex interfaces nedenfor, har du umiddelbart svaret på det de spør om?

jeg har sendt dem alle relevante config filer,

/Erik

Den 11. april 2014 10:35, skrev Geir Andre:

Siste Openstack-finish

> Hei
> [Sitert tekst skjult]

Geir Andre <geirtufte@gmail.com> 17. april 2014 kl. 12.51
Til: Erik Hjelmås <erikh@hig.no>
Kopi: kyrre.begnum@hig.no, "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Hei!

Takk for mailen! Det holdt lenge det!

Vi har nå fått til å rulle ut Openstack med nettverk-support men det er et par småting vi fortsatt ikke har fått til/testet nok/trenger hjelp til. Det vi ikke har fått til enda er å sette opp en VM slik at den er tilgjengelig ut mot Internett, men her vi usikkre på hvordan det eksterne nettverket skal konfigureres og hvilke IP-addresser som skal settes hvor. Vi har testet med noen random eksterne nett til nå og tildeling av floating-ip fungerer. Hvis du hadde hatt 10minutter til å se på dette sammen med oss etter påske, hadde det vært fint!

Vi er ferdig med å rulle ut munin-nodene og jobber nå med hvordan munin-master regelmessig kan sjekke etter nye hosts i nettet. Vi vil nedprioritere Nagios for å rekke å skrive så mye som mulig på rapporten frem til 1 mai da Hanno hadde muligheten til å lese den da. Vi får se om vi legger til Nagios etter 1 mai, avhengig av hvordan vi ligger an med rapporten.

Har du noen innspill?

Kyrre:
Du kan se bort fra den mailen du fikk ang. interface br-ex :)

Mvh
Geir
[Sitert tekst skjult]

Erik Hjelmås <erikh@hig.no> 17. april 2014 kl. 15.17
Til: Geir Andre <geirtufte@gmail.com>
Kopi: Kyrre Matthias Begnum <kyrre.begnum@hig.no>, "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Den 17. april 2014 12:51, skrev Geir Andre:
> Hei!
>
> Takk for mailen! Det holdt lenge det!

flott! bra jobba!

jeg har 10min enten tirs 22/4 eller fredag 25/4, så da snakes vi,

/Erik
[Sitert tekst skjult]

VM i-net access

Kjetil André Finsrud <kafinsrud@gmail.com>

VM i-net access

2 e-poster

Geir Andre <geirtufte@gmail.com> 28. april 2014 kl. 09.21
Til: Erik Hjelmås <erikh@hig.no>, kyrre.begnum@hig.no
Kopi: "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Hei!

Under boot av en CirrOS instans får vi denne meldingen under nettverksconf;

```
Starting network...
udhcpc (v1.20.1) started
Sending discover...
Sending select for 10.5.5.2...
Lease of 10.5.5.2 obtained, lease time 120
deleting routers
route: SIOCDELRT: No such process
adding dns 128.39.32.2
adding dns 8.8.8.7
adding dns 8.8.8.8
cirros-ds 'net' up at 2.86
checking http://169.254.169.254/2009-04-04/instance-id
successful after 1/20 tries: up 2.89. iid=i-00000001
found datasource (ec2, net)
cirros-apply-net already run per instance
check-version already run per instance
Starting dropbear sshd: OK
userdata already run per instance
=== network info ===
if-info: lo,up,127.0.0.1,8,::1
if-info: eth0,up,10.5.5.2,24,fe80::f816:3eff:fec9:3d5d
ip-route:default via 10.5.5.1 dev eth0
ip-route:10.5.5.0/24 dev eth0  src 10.5.5.2
=== datasource: ec2 net ===
instance-id: i-00000001
name: N/A
availability-zone: nova
local-hostname: test.novalocal
launch-index: 0
=== cirros: current=0.3.1 uptime=5.51 ===
```

Sjekket opp hvilke DNS instanser i Appdrift har (128.39.32.2). La derfor til denne DNS serveren for instanser i nettverket vårt også.
> cat /etc/resolv.conf
search openstacklocal
nameserver 128.39.32.2
nameserver 8.8.8.8
nameserver 8.8.8.7

Fra instansen får jeg pinget neutron-router 10.5.5.1 og jeg får pinget eksternt interface på neutron-routeren 128.39.141.217. Men, jeg kommer fortsatt ikke ut på nett.

Satt opp tcpdump på br-ex på nettverksnoden mens jeg pinget www.vg.no. Resultatet av dump;

09:03:19.029019 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:20.028466 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:21.028456 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:22.029124 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:23.028455 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:24.028467 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28
09:03:25.029351 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28

VM i-net access

09:03:26.028459 ARP, Request who-has 128.39.140.1 tell 128.39.141.217, length 28

Kommando kjørt på nettverksnoden;
> ip a | grep state
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
3: eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
4: eth2: <BROADCAST,MULTICAST,PROMISC,UP,LOWER_UP> mtu 1500 qdisc mq state UP qlen 1000
5: eth3: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN qlen 1000
6: ovs-system: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN
7: br-int: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
8: br-ex: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
9: br-eth1: <BROADCAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN
10: phy-br-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000
11: int-br-eth1: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state UP qlen 1000

Skal ovs-system ha state DOWN og br-int, br-ex og br-eth1 ha state UNKNOWN?

Hvis det er relevant, kunne jeg fått en output fra **ip a | grep state** fra SkyHiGh's nettverksnode?

For pingingen er følgende regel lagt til i SecGroup rules;
Ingress IPv4 ICMP - 0.0.0.0/0 (CIDR)

Noen idéer?

Mvh
Geir

Kyrre Matthias Begnum <kyrre.begnum@hig.no> 28. april 2014 kl. 10.13
Til: Geir Andre <geirtufte@gmail.com>
Kopi: Erik Hjelmås <erikh@hig.no>, "andersnoem90@gmail.com" <andersnoem90@gmail.com>, "kafinsrud@gmail.com" <kafinsrud@gmail.com>

Hei, jeg tror man må sjekke på den andre enden. Her blir jo arp sendt ut, men får ikke svar, er dere sikre på et den porten fungerer? Dere kan jo evt. koble en laptop til den porten og gi den 217 adressen. Så kan dere se om dere får pinget? Det ser ut som om ting er riktige i openstack.
[Sitert tekst skjult]

Great Success

Kjetil André Finsrud <kafinsrud@gmail.com>

Fwd: Re: Great Success!

1 e-post

Geir Andre <geirtufte@gmail.com> 12. mai 2014 kl. 09.33
Til: "kafinsrud@gmail.com" <kafinsrud@gmail.com>, "andersnoem90@gmail.com" <andersnoem90@gmail.com>

-------- Original Message --------
Subject: Re: Great Success!
Date: Wed, 30 Apr 2014 21:47:59 +0200
From: Erik Hjelmås <erikh@hig.no>
To: Geir Andre <geirtufte@gmail.com>, Kyrre Matthias Begnum <kyrre.begnum@hig.no>

kjempebra! gjetter på at det er en god følelse nå :)

/E

Den 30. april 2014 19:53, skrev Geir Andre:
> Nettverksnoden (br-ex) har vært tilkoblet VLAN 10 på switchen som står i
> racket. Dette VLAN'et server tydligvis ikke det nettet vi trodde var
> 128.39.140.0/23. Dette ble oppdaget da jeg fikk en mistanke om at det
> kanskje var to forskjellige nett og satt opp expose-api-interfacet på
> kontrolleren til å motta DHCP og fikk da en 128.39.*142*.0/24 addresse.
> Newman står jo tilkoblet det sistenevnte nettet og der har det ikke vært
> noe problem, så vi har antatt at feilen har ligget hos oss. Etter denne
> oppdagelsen, dro vi en lang TP fra br-ex på nettverksnoden til switchen
> vi har stående på bordet (k202) som vi vet server 128.39.140.0/23 nettet
> og det fikset problemet.
>
> Dette betyr egentlig at oppsettet vårt har fungert i lang tid og at det
> kun er denne lille bagatellen som har vært årsaken :)
>
> Gleder meg til å fortelle resten av gruppen om resultatet i morgen, det
> får bli en overaskelse! :)
>
> I morgen vil vi teste å sette opp en VM med apache og assosiere en
> floating-ip for å hoste en random web-side, for deretter å begynne med
> automatisert utrulling av VM'er.
>
> Geir
>
> On 04/30/2014 06:44 PM, Erik Hjelmås wrote:
>> knallbra jobba!
>>
>> hva fiksa det?
>>
>> erik
>>
>>
>>
>>
>> Sendt fra Galaxy Tab
>>
>>
>> -------- Opprinnelig melding --------
>> Fra: Geir Andre
>> Dato:30.04.2014 16:42 (GMT+01:00)
>> Til: Erik Hjelmås ,Kyrre Matthias Begnum
>> Emne: Great Success!

Great Success


```
>>
>> Hei
>>
>> Får nå pinget Internett fra VM'er.
>>
>> Takk for hjelp :)
>>
>> Erik: Si ifra hvis du fortsatt vil ha tilgang til systemet.
>>
>> /Geir
>
```

Tillegg F

Logg og status

Statusrapporter

Statusrapport

Autostack

Statusrapporter

5. Februar

Fremdriftsplan

Oppsettet av server-skap har tatt noe lenger tid en forventet da vi har vært to grupper som har hatt behov for å installere hvert sitt oppsett, bestående av flere servere. Det har også være feilsøking på èn av serverene som har defekt raid-kontroller og er med det tatt ut av løsningen.

Iht. planene lagt i forprosjektet, skal vi på dette tidspunktet være i gang med installasjon av Puppet og arbeidet rundt Cobbler være avsluttet. Med dette ligger vi foran det som er planlagt og er godt igang med arbeidet som egentlig skulle startet på et senere tidspunkt.

Under arbeid

Når en maskin har fått rullet ut et operativsystem, er det ønskelig at maskinen kontakter Puppet-master for videre konfigurasjon. Det er i denne sammenheng arbeidet med hvordan en maskin skal koble seg til Puppet-master etter operativsysteminstallasjonen. Enkelte verktøy i Cobbler er blitt prøvd for å løse dette problemet, men uten hell. Vi har vurdert et alternativ som baserer seg på å laste opp et bash-skript til en maskin under operativsystem utrulling som eksekveres og klargjør maskinen for Puppet.

Arbeid fremover

Det vil bli lagt fokus på startup-skriptet som skal benyttes for klargjøring av maskiner for Puppet. Det er også på dette tidspunktet høyst relevant å sette seg inn i bruken av Puppet for å komme igang og bruke dette verktøyet så fort så mulig. Det kan bli nødvendig å ta ibruk andre ressurser en oppdragsgiver og veileder for å få støtte til det vi fremover skal jobbe med.

Gruppen vil jobbe parallelt fremover hvor noen jobber med oppsettet av Puppet, mens andre starter med å lese seg opp på Openstack og hvordan de aktuelle Openstack-komponentene relevante for oppgaven henger sammen. Dette for å klargjøre for den manuelle installasjonen av Openstack.

Utfordringer

Er det mulig å automatisere den manuelle jobben hvor man før en operativsystem ut-rulling må velge å boote maskinen fra nettverket? Hvis booting fra nettverket er satt som førsteprioritet, vil dette forårsake en loop av operativsystem utrullinger. Er det da en måte for å kun boote fra nettverket èn gang for deretter å alltid boote fra harddisk?

Det er blitt forespurt om oppgaven kan tas til det punkt hvor løsningen kan legges ut som en tjeneste på nett hvor en fullautomatisert installasjon av Openstack kan gjøres over Internett. Dette er arbeid som kan påbegynnes først når de store essensielle komponetene av oppgaven er ferdig.

Annet

Statusrapporter

5. Mars

Fremdriftsplan

Ut ifra gantt skjema skal vi nå være ferdig med å installere Cobbler, Puppet og OpenStack manuelt, og vi skal nå være i startfasen av automatiseringen. Ettersom vi har hatt en god del problemer underveis siden forrige statusrapport så ligger vi nå litt etter skjema. Vi ser ikke på dette som noe stort problem slik situasjonen er nå.

Under arbeid

For øyeblikket arbeides det fortsatt med den manuelle installasjonen av OpenStack, men vi regner med å være ferdig i løpet av uken. Vi har også begynt å gjøre oss kjent med Puppet og enkelte av modulene som vi muligens kommer til å benytte

Arbeid fremover

Vi anslår at vi kan klare å være ferdig med den manuelle installasjonen i løpet av denne uken, og vi vil derfor komme i gang med å automatisere utrullingen av OpenStack med Puppet allerede i neste uke. Vi kommer også til å sette opp og lære oss å bruke Git. Arbeidet med rapporten vil fortsatt foregå parallelt med utviklingen slik som før

Utfordringer

Lære oss å bruke Git og sette oss godt inn i hvordan Puppet og dets moduler fungerer. Ettersom dette er helt nytt for gruppen vil det være usikkert hvor omfattende dette vil være. Vi kan også se for oss at det kan by på problemer med overgangen fra manuell til automatisk utrulling.

Annet

Statusrapporter

2. April

Fremdriftsplan

Godt ute i skript-fasen av oppgaven har vi begynt på automatiseringsprosessen av Openstack installasjonen. Dette innebærer å rulle ut en konfigurasjon på serverene med Puppet, teste konfigurasjonen og hvis testen gikk bra, utvide konfigurasjonen som rulles ut som til slutte skal bli den endelige installasjonen av Openstack. For de gangene testen ikke går bra, må skriptet som styrer server-konfigurasjonen endres til ønsket resultat oppnås.

Under arbeid

For å forenkle automatiseringsprosessen er det blitt vurdert flere Puppet-moduler for å rulle ut Openstack. Disse modulene har som regel hver sin funksjon og er tilpasset et spesifikt Openstack-miljø, være seg et multinode-miljø eller en enklere løsningen bestående av kun èn server. Det er derfor blitt brukt litt tid på å finne ut hvilke moduler som vil passe best for det multinode-miljøet vi skal rulle ut.

Gruppen jobber også med å avdekke hvilken informasjons som trengs fra en kunde før en utrullingsprossess kan starte. Dette er informasjon som mac-addresser på serverene og ip-addresser som skal settes på serverene.

Arbeid fremover

Det vil være høyt fokus på å få til Openstack installasjonen hvor alle komponentene fungerer. Gruppen har møtt motgang under nettverksoppsettet i Openstack og vil med dette prioritere å få til denne delen før arbeid med tillegsfunksjonalitet som eksempelvis overvåkning iverksettes.

Utfordringer

Hvordan kan vi gjøre en Openstack installasjon så enkel som mulig for en kunde? Med forenkling menes i hvor stor grad kan vi redusere mengden informasjonen vi trenger på forhånd fra kunden. For at det skal bli enklere for kunden, vil løsningen som utvikles være mer kompleks med en høyere dynamisk tilnærming til utrullingen hvor informasjonen vi er avhengig av oppdages underveis i utrullingen.

Annet

Statusrapporter

30. April

Fremdriftsplan

Vi har nå kommet til den siste delen av prosjektet der det skal være fokus på rapportskriving. Frem til nå så har vi fullført automatiseringen av OpenStack og utrullingen av Munin. OpenStack løsningen vår mangler egenskapen å kunne pinge ut av det lokale nettverket, men feilsøking fortsettes foreløpig. Vi har også valgt å sløyfe Nagios helt fra oppgaven. Med tanke på rapporten ligger vi i rute til å få levert den til fristen 19. Mai.

Under arbeid

Vi har nå mest fokus på rapporten og kommer til å levere et utkast av denne til Hanno i morgen 1. Mai. Det er fortsatt aktivitet på feilsøkingen med å få pinget ut fra OpenStack.

Arbeid fremover

Arbeidet fremover vil primært være rapporten. Vi kommer i første omgang til å ha størts fokus på å produsere mer tekst og rettskrive/fin pusse denne, før vi kommer til å gå over formatering og pynting av rapporten i sin helhet.

Utfordringer

Vår største utfordring vil være å holde motivasjonen oppe under rapportskrivingen. Av tekniske utfordringer har vi fortsatt det med å kunne pinge ut av lokalnettet, dette er noe som i værste fall ikke vil bli implementert i vår endelige løsning.

Annet

Logg

		Geir		Anders		Kjetil	Totalt
Dato	Timer	Beskrivelse	Timer	Beskrivelse	Timer	Beskrivelse	pr uke
7.1.	2	Puppet, møte	2	Puppet, møte	2	Puppet, møte	
8.1.	6	Lynkurs, adm, webside	2	Lynkurs	6	Lynkurs, adm	
9.1.	8	SU-modell, forprosjekt planlegging	8	SU-modell, forprosjekt planlegging	6	SU-modell, forprosjekt planlegging	
10.1.	4	Møte, Prosjektplan, loggføring	8	Møte, SU-modell, prosjektplan	9	Møte, SU-modell, prosjektplan, +div retting	
11.1.							
12.1.	5	Oppgavebeskrivelse	5	Verktøy, ressursbehov	5	Div LaTeX, prosjektavtale	
Totalt	25		25		28		78
13.1.	2	Adm, Latex	2	Diskusjoner, prosjektavtale, div.	2	Div LaTeX, forside	
14.1.	9	Use case, latex, HW, dokumentasj., div.	7	Gantt dok., HW, opplesing, dokument	9	Use case, Gantt, LaTeX, lest skyhig inst. guide, rutiner og regler	
15.1.	8	Møte, Rack, Newman(Raid + Ubuntu), Cobbler	3	Møte, opplæring i Puppet	0	Fridag	
16.1.	5	Newman(Ubuntu), Rack, Forprosjekt	6	Use case beskrivelse/modeller	6	Use case, Deployment view	
17.1.	5	Kravspek, Cobbler, dokumentasjon	6	Ikke-funksjonelle krav, Puppet	6	Deployment txt, LaTeX, Kravspek, div	
18.1.							
19.1.	6	Cobbler PXE-boot + Webside	4	Opplesning Cobbler, OpenStack-Moduler	4	Retting/pynting forprosjekt	
Totalt	35		28		27		90
		Cobbler på Newman + dokumentasjon			3	cobbler, deployment view retting	
20.1.	7						
21.1.	7	Cobbler-testing	9	Teori OpenStack, Puppet testing	7	Puppet teori + VM	
22.1.	6	Cobbler, utrulling, bakgrunn	2	Teori, div	5	Møte, puppet, utrulling OS	
		Dokumentasjon av cobbler	6	Skifting av konsoll, Puppet/Passenger	6	Byttet server monitor, Puppet, div	
23.1.	2						
		Dokumentasjon av cobbler, forprosjekt.	7	Korrigering av forprosjekt, Puppet teori	6	Forprosjekt, Puppet teori	
24.1.	3						
		Ferdigstilt dokumentasjon på cobbler.					
25.1.	1						
26.1.	3	Ferdigstilling av prosjektplan.	3	Ferdigstilling av prosjektplan.	3	Ferdigstilling av prosjektplan.	
Totalt	29		27		30		86
27.1.	4	Use case, forprosjekt	5	Use case, forprosjekt, Puppet dok.	4	Use case, forprosjekt	
28.1.	8	Utrulling, puppet preseed	5	Utrulling, puppet	8	Utrulling, puppet	
29.1.	9	Puppet preseed TS, møte	4	Puppet troubleshooting	7	Puppet, møte	
30.1.	6	Puppet preseed TS	4	Puppet troubleshooting	1	Puppet	
31.1.	6	Puppet, preseed	9	Dokumentasjon, Teori moduler	10	Puppet, cyberdawn, div	
1.2.							
2.2.							
Totalt	33		27		30		90
3.2.	8	Puppet, hostname, Web-update	1	Dokumentasjon	6	Puppet, hostname	
4.2.	5	Hostname[MAC], dokumentasjon	8	Dokumentasjon, OpenStack	8	Teori OpenStack, LaTeX	
5.2.	8	Dokumentasjon, møte, hostname[MAC]	2	Teori OpenStack inst, møte	8	Teori OpenStack, LaTeX, møte	
6.2.	7	Dokumentasjon, Openstack	8	OpenStack installasjon	5	OpenStack	
7.2.	5	OpenStack Nova	8	OpenStack installasjon	7	OpenStack Keystone, dokumentasjon	
8.2.							
9.2.	6	Openstack installasjon/feilsøking	6	Openstack installasjon/feilsøking, dokumentasjon	6	Openstack installasjon/feilsøking, dokumentasjon	
Totalt	39		33		40		112
10.2.	4	OpenStack Neutron og Compute	4	OpensStack Neutron og Compute	4	Openstack Neutron og Compute	
11.2.	7	OpenStack, Nova og Neutron (fresh-install)	4	OpensStack Folsom-Guide	7	OpensStack Folsom-Guide	
12.2.	6	OpenStack, Neutron og Compute + TS	4	Puppet+OpenStack-classes, cont. Inst.	6	OpensStack Folsom-Guide	
13.2.	4	OpenStack Neutron TS	7	OpensStack Folsom-Guide, div. Teori	4	OpensStack Folsom-Guide	
14.2.	3	Møte, OpenStack Horizon TS	6	Møte, Teori, dokumentasjon	4	Møte, Teori	
15.2.			2	Teori, dokumentasjon			
16.2.			1	FOI			
Totalt	24		28		25		77
17.2.	5	Nytt oppsett av OpenStack	4	OpensStack feilsøking og testing	5	OpenStack feilsøking og testing	

Logg

		Ferdigstille OpenStack + Horizon testing	4	OpensStack VM, feilsøking	5	OpenStack feilsøking og testing	
18.2.	8						
		Horizon, møte, Puppet-klargjøring	3	Møte, teori manifests	5	OpenStack feilsøking og testing	
19.2.	8						
			0	SKIDAG	2	Openstack feilsøking	
20.2.							
			4	Openstack feilsøking	4	Openstack feilsøking	
21.2.							
			5	OpensStack feilsøking/test(connection			
22.2.				issues)			
23.2.	7	OpenStack Installasjon	7	OpenStack Installasjon	5	OpenStack Installasjon	
Totalt	28		27		26		81
24.2.	2	Ferdigstille OpenStack + Neutron TS.	6	Ferdigstilling og feilsøking OpenStack	3	Ferdigstilling og feilsøking OpenStack	
25.2.	7	Ferdigstilling og feilsøking OpenStack	7	Ferdigstilling og feilsøking OpenStack	7	Ferdigstilling og feilsøking OpenStack	
26.2.	7	Lærer puppet.	4	Dokumentasjon, LaTeX	5	OpenStack, dokumentasjon, LaTeX	
27.2.	4	Lærer puppet, satt opp Git (basics).	5	Dokumentasjon, LaTeX	5	LaTeX, dokumentasjon	
28.2.			3	Dokumentasjon, LaTeX	5	LaTeX, dokumentasjon	
1.3.							
		Puppet environments, git, nova-					
2.3.	5	utrulling.					
Totalt	25		25		25		75
3.3.	4	Puppet modul mysql.	4	Dokumentasjon, LaTeX	5	LaTeX, dokumentasjon	
4.3.	8	Puppet utrulling av Controller-node.	5	Dokumentasjon, LaTeX, Teori	8	LaTeX, dokumentasjon	
5.3.	5	Møte, Dokumentasjon, Teori, Lynkurs	3	Teori, Lynkurs	6	LaTeX, dokumentasjon, lynkurs2	
6.3.	4	Dokumentasjon.	6	Puppetmoduler	4	Puppetmoduler	
7.3.	2	Dokumentasjon	6	Puppetmoduler	3	Puppetmoduler	
8.3.							
9.3.	1	Oppdatert hjemmeside	2	Puppetmoduler			
Totalt	24		26		26		76
10.3.	1	Dokumentasjon ferdigstilling	3	Puppetmoduler	2	Puppetmoduler	
11.3.	6	Puppet modul testing	4	Puppetmoduler, teori	6	Puppetmoduler, teori	
12.3.	7	Møte, PuppetModule Keystone	2	Puppetmoduler, Møte	7	Puppetmodul Neutron, møte	
13.3.	4	PuppetModule Keystone og Glance	0	VALGFAG	6	Puppetmodul Nova	
14.3.	1	PuppetModule Nova og Neutron	5	Puppetmoduler, feilsøking	4	Puppetmoduler, feilsøking	
15.3.			5	Git			
16.3.	3	Git intro + github	6	Puppetmoduler, feilsøking	5	Puppetmoduler, feilsøking	
Totalt	22		25		30		77
17.3.	6	Git oppsett på newman	3	(Jasså)Git		Fridag	
18.3.	8	Git, startupscript, Horizon,	6	Puppet feilsøking, bearbeiding rapport	7	Puppetmoduler, feilsøking	
19.3.	7	Horizon, startupscript	2	Git	6	Puppetmoduler, feilsøking	
20.3.	4	Puppet Horizon/Keystone TS	8	Dokumentasjon rapport	4	Dokumentasjon rapport	
21.3.	1	Puppet Keystone TS	7	Dokumentasjon, rapport	6	Dokumentasjon rapport	
22.3.							
23.3.	4	Keystone/Nova/Nova-Compute/OS-instanser	4	Dokumentasjon, rapport	4	Dokumentasjon rapport	
Totalt	30		30		27		87
24.3.	6	TS – får ikke opprettet instanser	4	Dokumentasjon	5	Dokumentasjon rapport	
25.3.	6	TS – Instanser opprettes men uten nettverk. Kabling	6	Korrigering av rapport	5	Dokumentasjon rapport, puppet static IP	
26.3.	6	TS – NetverksNode.	2	Static IP, Møte	5	Puppet dynamic => static IP, møte	
27.3.	4	TS – NettverksNode, ControllerNode	7	Static IP	6	Static IP	
28.3.	4	Dokumentasjon	5	Static IP, dokumentasjon	5	Static IP, dokumentasjon	
29.3.							
30.3.	2	Dokumentasjon	4	Dokumentasjon	3	LaTeX, dokumentasjon	
Totalt	28		28		29		85
31.3.	6	Dokumentasjon	6	Static IP, Munin	5	Static IP, Munin	
1.4.	6	Munin, Rack-servere.	4	Munin, rapport	8	Munin, fikse "defekte" servere/HDD/NIC	

Logg

2.4.	7	Network-node puppet config	2	Neutron feilsøking	7	Hosts script, Neutron puppet modul feilsøking
3.4.	6	Network-node puppet config, script	7	Neutron feilsøking, dokumentasjon	5	Neutron puppet manifest
4.4.	3	Neutron config script[Bridge&Ovs-Plugin]	5	div. Feilsøking/rapportformatering	5	Neutron/nova feilsøking
5.4.						
6.4.	2	Neutron config script[Bridge&Ovs-Plugin]	4	Neutron, rapportformatering	4	Neutron, LaTeX, div
Totalt	30		28		34	92
7.4.	6	Verifisering av installasjon på Controller og Neutron	6	Controller, script	6	Controller, script
8.4.	7	Verifisering av Installasjon på Controller og Compute	5	div. moduler+scripting, rapport	6	Controller, Compute moduler+script
9.4.	7	Verifisering av installasjon på Controller og Compute	2	Verifisering av installasjon på Controller og Compute	6	Verifisering av installasjon på Controller og Compute
10.4.	6	Verifisering av hele openstack-install + testing	8	Verifisering av installasjon på Controller og Compute	6	Verifisering av installasjon på Controller og Compute
11.4.	1	Møte, feilsøking nettverk	7	Møte, feilsøking nettverk, horizon	4	Møte, script/site.pp for horizon
12.4.						
13.4.	2	Ubuntu img up! Feilsøking nettverk			2	verifisering av script/site.pp
Totalt	29		28		30	87
14.4.	8	Nettverk up! Automatisering av nettverk			8	Nettverk, horizon automatisering, diverse
15.4.	8	Automatisering av nettverk	3	Rapport(diskusjon)	8	Nettverk, rapport (diverse), prøvd å fikse server
16.4.	5	Automatisering av nettverk	5	Munin	5	Munin
17.4.	4	Dokumentasjon	6	Munin	4	Munin ping sweep
18.4.	5	Dokumentasjon, Munin-script, MuninMaster	5	Munin, dokumentasjon	5	Munin-script, dokumentasjon
19.4.						
20.4.	3	Munin-finish og Openstack-finish	6	Rapport(notater, dokumentasjon, format)	3	Rydde i script, dokumentasjon
Totalt	33		25		33	91
21.4.	4	Dokumentasjon	6	Dokumentasjon	6	Dokumentasjon, rydde/kommentere kode
22.4.	4	Dokumentasjon, kommentere kode	4	Dokumentasjon, LaTeX	6	Dokumentasjon, LaTeX, kommentering av kode
23.4.	6	Møte, dokumentasjon.	2	Møte, dokumentasjon	5	Møte, dokumentasjon
24.4.	6	Dokumentasjon	6	Dokumentasjon	4	Dokumentasjon
25.4.	3	Feilsøking VM Internett tilgang	5	Dokumentasjon, planlegging	4	Dokumentasjon
26.4.			2	Dokumentasjon		
27.4.	4	Feilsøking VM Internett tilgang	4	Dokumentasjon	3	Dokumentasjon
Totalt	27		29		28	84
28.4.	4	Feilsøking VM Internett tilgang, Dok.	4	Dokumentasjon	4	Dokumentasjon
29.4.	6	Feilsøking VM Internett tilgang, Dok.	6	Dokumentasjon	5	Dokumentasjon
30.4.	6	VM internett tilgang fixed! Dokumentasjon	3	Møte, dokumentasjon	5	Møte, Dokumentasjon
1.5.	6	Auto. Oppsett av scenarie	8	Dokumentasjon	6	Dokumentasjon
2.5.	2	Auto. Oppsett av scenarie	2	Dokumentasjon	5	Dokumentasjon
3.5.						
4.5.			2	Dokumentasjon		
Totalt	24		25		25	74
5.5.	9	Finpuss Auto-Scenarie. Dokumentasjon.	8	Brukerveiledning, Dokumentasjon	10	Dokumentasjon, LaTeX
6.5.	7	Dokumentasjon	6	Dokumentasjon	9	Dokumentasjon, LaTeX
7.5.	7	Openstack-dybdeforståelse, Dokum.	5	Dokumentasjon, LaTeX	8	Dokumentasjon, LaTeX
8.5.	7	Dokumentasjon, retting	8	Dokumentasjon	4	Dokumentasjon, LaTeX
9.5.	6	Dokumentasjon, retting	4	Dokumentasjon, retting av innhold	2	Dokumentasjon, LaTeX
10.5.						
11.5.			4	Retting	5	Retting, LaTeX
Totalt	36		35		38	109
12.5.	3	Dokumentasjon, retting	2	Retting	6	Retting
13.5.	5	Dokumentasjon, retting	2	Retting	7	Retting
14.5.	7	Dokumentasjon, Jungle-scenario, Møte	2	Retting, Møte	4	Retting, Møte
15.5.	9	Dokumentasjon, Jungle-scenario.	3	Retting	4	Retting
16.5.	5	Dokumentasjon, retting	9	Retting	5	Retting

Logg

17.5.	4	Dokumentasjon, retting	5	Retting	3	Retting			
18.5.	8	Dokumentasjon, retting	7	Retting	6	Retting			
Totalt	41		30		35				106
19.5.	11	Retting	11	Retting	11	Retting			
20.5.									
21.5.									
22.5.									
23.5.									
24.5.									
25.5.									
Totalt	11		11		11				33
SUM:	573	Geir	540	Anders	577	Kjetil	SUM:	1690	

Ukelogg

Ukelogg AutoStack

January 2014

Ukelogg

Uke2

Deltatt på lynkurs i prosjektstyring med Tom Røise. Dokumenter som omhandler rettningslinjer for bacheloroppgaven er lest. Administrative verktøy som skal brukes under bacheloroppgaven er innstallert og tilpasset, eksemplevis SVN, ShareLatex og andre relevante programmer for modellering/skjema.

Lest på tidligere bacheloroppgaver innenfor OpenStack, samt teknisk dokumentasjon rundt SkyHigh.

Gruppen jobbet med å finne hvilke utviklingsmodeller som passer best og hvordan disse kan kombineres for å gi oss de nødvendige karakteristikker vi er avhengig av under arbeidet med bacheloroppgaven. Hatt møte med Tom Røise i forbindelse med valg av systemutviklingsmodell. Har etter diskusjoner valgt en modell bestående av hovedsakelig Inkrementell Sekvensiell modell, samt Scrum(disse i hver sin fase av utviklingsprosessen).

Har begynt å arbeide på prosjektplanen, med blant annet: Gantt-skjema,
risikostyring, roller, avgrensning og som nevnt systemutviklingsmodell. Mange av disse punktene skal vi gå tilbake og redigere/forbedre på.

Uke3

Denne uken ble det fokusert på kravspesifikasjonsdelen av forprosjektet. Use Case har blitt viet mye tid og ressurser for å få et klargjort utgangspunkt før selve prosjektet påbegynnes.

Samtidig har opplesning på sentrale temaer vært i fokus, med Puppet, Cobbler og OpenStack i fokus. Vi er i ferd med å oppnå god kjennskap til de to førstnevnte, slik at vi har fått en bedre forståelse av hvordan oppgaven skal løses. OpenStack med dets moduler er fortsatt litt fremmed.

Alle serverne vi har fått tildelt er blitt montert i serverracket og Newman er satt opp med Ubuntu server 13.10 hvor diskene er satt i raid 1+0 for redundans.

Websiden er lagd og publisert på tildelt område.

Godt igang med Cobbler og har startet med testing av operativsystem utrulling.

Ukelogg

Uke4

Denne uken er det gått med mye tid på å lese teori innenfor Puppet og OpenStack, for å få en bredere plattform til å installere disse.

Vi har skiftet konsoll på rackskapet, med en del utfordringer som oppstod underveis.

Det har gått med mye tid på ferdigstilling av forprosjektet. Vi har funnet en del områder det har vært behov for omjusteringer på.

Vi har også rullet ut Ubuntu på flere servere. Med dette har vi forbedret automatiseringen av installasjonen.

Det har vært fremskritt på Cobbler-fronten med oppsett på Newman, testing, utrulling, og ferdigstilling av dokumentasjon.

Uke5

Ferdigstilt og levert inn forprosjekt på mandag

Har brukt mye tid på å få installert puppet på slutten av en operativsystem ut-rulling. I samme prosess med å installere puppet, skal puppet agenten kontakte puppet masteren. Vi har hatt flere tilnærminger for å få til dette. Vi håper på å løse problemet med å bruke et bash-script som kjøres på clienten ved første oppstart etter OS ut-rulling.
Er i gang med dokumentasjon av teori på de tre modulene vi arbeider på (Cobbler, Puppet, OpenStack). Knyttet til dette er uthenting av informasjon fra ulike kilder og vurderinger av hva som er fornuftig å ha med.

Uke6

Tidlig denne uken ble vi ferdig med å sette opp Puppet slik at det er klart for å rulle ut OpenStack. Før vi starter med å rulle ut OpenStack med Puppet har vi satt av to uker til å installere OpenStack manuelt for å få kjennskap til modulene OpenStack består av.

Vi har jobbet parallellt denne uken hvor vi har dokumentert arbeidet med Puppet samtidig som vi har startet installasjon av OpenStack. Mot slutten av uken har alle vært involvert i OpenStack installasjon.

Vi har kommet godt igang med å installere OpenStack-moduler og er ferdig med å sette opp Nova og snart er Neutron oppe også.

Neste uke vil vi ferdigstille Neutron og starte med å sette oppe en Compute-node. Vi vil nok komme til å bruke litt tid på å sy sammen disse modulene til å fungere sammen når alle modulene er installert.

Ukelogg

Uke7

Denne uken har mye tid blitt benyttet i forbindelse med manuell installasjon av OpenStack. Vi delte oss opp, slik at to jobbet på de fysiske serverne, mens en opprettet sitt eget virtuelle miljø.

Vi utførte installasjonsprosessen delvis flere ganger, ved bruk av både skyHigh,- og OpenStack-folsom installasjonsguider, i tillegg til en kombinasjon av disse. På denne måten vil vi finne den beste metoden for oss.

Vi støtte på noen utfordringer underveis, men de fleste av disse var greit
overkommelige med litt feilsøking. På slutten av uken stoppet det litt opp med oppsett av VM'er.

Vi benyttet litt tid på slutten av uken til dokumentasjon, samt diverse teori.

Uke8

Vi har fortsatt å jobbe delt denne uken, med manuell installasjon av OpenStack. Vi har møtt på en rekke små problemer underveis, noe som har ført til at fremdriften har vært relativt lav.

Det har blitt benyttet ulike installasjonsveiledninger slik at vi ville få den beste konfigurasjonen, og har til slutt endt opp med å stort sett følge OpenStack-folsom-guiden.

Underveis i det manuelle oppsettet, endelige versjon, har vi valgt å dokumentere de konfigurasjonene vi utførte som i utgangspunktet ikke var en del av guiden, i tillegg til at vi forsøkte å nevne de fleste av utfordringene vi møtte på, samt hvordan vi løste de.

I tillegg har Geir i sitt virtuelle miljø satt opp webgrensesnittet Horizon, som kan være til hjelp for forståelse og utførelse av oppgaven.

Uke9

Vi begynte denne uken med ferdigstilling av den manuelle installasjonen av OpenStack. Vi er så godt som ferdig med dette, men opplever noen mindre nettverksproblemer som vi trenger litt hjelp med før vi kan si at vi er helt i mål.

I mellomtiden har vi delt oss opp slik at to har arbeidet på dokumentasjon av den manuelle implementasjonen, mens en har sett nærmere på Puppet i forbindelse med de kommende manifestene som skal settes opp. Når det gjelder dokumentasjonen har det gått med litt tid på LaTeX-formateringen slik at vi får den beste layouten til rapporten.

Før vi ruller ut OpenStack med de modulene som allerede eksisterer i Puppet, skal vi først rulle ut en statisk konfigurasjon av OpenStack. Det vil i praksis bety at vi installerer de pakkene som skal installeres for deretter å kopiere over de aktuelle konfigurasjonsfilene vi brukte under den manuelle installasjonen av OpenStack. Dette er også en god måte å bli kjent med Puppet på før vi starter utrulling med de relativt store og komplekse OpenStack modulene til Puppet.

Ukelogg

Uke10

Vi begynte denne uken med dokumentasjon av OpenStack, samt på andre områder hvor vi følte behov for det. Ved å forsøke å dokumentere på de områdene vi kan underveis, vil det være lettere i sluttfasen å kunne rette på ting, spare oss for skippertak, samt utbedre ut i fra den kunnskapen vi har opparbeidet oss.

Var på lynkurs 2 på Onsdag, og fikk her verdifull informasjon rundt rapportskriving.

Vi har jobbet mye parallelt denne uken hvor noen har studert ulike Puppet-moduler som skal brukes til ut-rullingen, mens andre har jobbet med å rulle ut den manuelle installasjonen av Openstack med Puppet.

Uke11

Denne uken satte vi oss dypere inn i puppetmodulene for OpenStack, og arbeidet med å få et teoretisk grunnlag for å kunne begynne arbeidet med testing av manifester. Det var en del usikkerhet i starten rundt hvilke kilder vi helst burde benytte oss av og som passet best, men dette løste seg ganske greit etter ukens møte hvor vi ble veiledet inn på riktig spor. Mot slutten av uken har vi fått til å rulle ut flere av de aktuelle Openstack-komponentene med Puppet.
For å ha backup og versjonskontroll på manifester, konfigurasjonsfiler og skript vi lager, har vi begynt å lære oss Git, hvordan det skal brukes og settes opp

Underveis i disse prosessene har vi dokumentert litt rundt teori vi har funnet, metoder vi har testet ut, samt problemområder vi møtte på og hvordan vi løste disse.

Uke12

I løpet av denne uka har vi kjørt to parallelle løp, der Geir har jobbet med Puppet manifestene for automatisk utrulling av OpenStack, samtidig som Kjetil og Anders har jobbet med rapportskriving og pynting på den. Modulene keystone, nova og glance er så godt som ferdige. Det samme gjelder for compute noden. Neutron er lagt inn men mangler å få verifisert installasjonen, og eventuelle tilpassinger som må gjøres gjenstår. Det er også blitt gjort forsøk på å få lagt inn horizon men så langt uten hell. Derfor vil arbeidet inntil videre fortsette å være kommandolinjebasert.

På onsdag så sendte vi det vi hadde av rapport så langt til Hanno og Erik, for å få litt feedback på det som er gjort. Både med tanke på oppsett/struktur og selve innholdet i teksten. Regner med å få tilbakemelding i løpet av neste uke.

Ukelogg

Uke13

Vi har i den uka jobbet videre med OpenStack modulene i Puppet, men status er den samme som forrige uke da vi er så og si ferdig med Keystone, Nova, Glance og har igjen å få Neutron opp å gå. Vi har også brukt tid på å lage en Puppet kommando som setter statisk IP på de forskjellige nettverkskortene, dette er vi blitt ferdig med og det fungerer på samtlige noder.
På onsdag så fikk vi tilbakemelding fra Hanno på det som ble sendt inn i forrige uke. Så vi har derfor brukt en del tid på å forbedre rapporten ut ifra denne tilbakemeldingen. Uten om dette har vi fått produsert mer tekst i rapporten ettersom vi har skrevet mer på implementasjonsdelen, lagt til innholdet i filer inn i appendiks, skrevet litt om terminologibruk og de ulike skytjenestene som finnes.

Uke14

På starten av uken ble det skrevet mer på rapporten samtidig som det var fokus på nettverkskonfigurasjon på Puppet-agentene samt overvåkning. Slik løsningen er nå vil hovedserveren (Newman) som ruller ut løsningen "snakke" med maskiner i nettet vha. den dynamiske ip-addressen de får tildelt. Vi må ha en løsningen for når Newman er ferdig med å rulle ut løsningen og maskinene i nettet ikke lenger har en DHCP. Dette er løst ved at hver Puppet-agent setter sin tildelte ip-addresse som en statisk IP-addresse.

For overvåkning, skal Newman rulle ut Munin og Nagios på alle nodene. Vi jobbet for å få ferdig Munin-nodene først som er veldig lett å installere og konfigurere. Det er derimot utfordrende å sette opp Munin-masteren som skal samle inn data og at Munin-masteren dynamisk skal legge til Munin-noder etterhvert som de dukker opp i nettverket. Dette høres ut som en standard funksjon et slikt verktøy bør ha, men det er ikke tilfelle med Munin. Hvordan dette skal løses kan bli veldig tidkrevende og vil bestå av "stygge" løsninger. Utrulling av Overvåkning nedprioriteres inntil videre for å fremover rette fullt fokus på å få til Openstack installasjonen.

Siste halvdel av uken føler gruppen at vi er på riktig spor med utrullingen av Openstack. Måten gruppen har jobbet på har vært å ta for seg èn og èn Openstack node, rulle ut den foreløpige konfigurasjonen med Puppet og sjekke iht. Openstack dokumentasjonen at ønskede parameter er satt. Hvis disse parameterene ikke har vært satt eller har vært feil, har vi oppdatert filen site.pp på Newman for å rette opp i feilene. Site.pp er manifestet på Puppet-master som styrer konfigurasjonen på Puppet-agentene i nettet. Dette høres forholdsvis ut som en enkel jobb, men det er ikke alltid at de Openstack-Puppet-modulene vi bruker har metoder for å sette de parameterene vi trenger fordi modulene ganske enkelt ikke er ferdig utviklet. Alle parametere som ikke modulene vi bruker har muligheten til å sette, løser vi med å bruke Puppet til å kjøre egenkomponerte bash-skript på agentene.

Ukelogg

Uke15

Etter at Openstack er blitt rullet iht. Openstack sin egen dokumentasjon har det vært nødvendig å i etterkant av utrullingen verifisere installasjonen. Dette har innebært å oppdatere manifestet som styrer node-konfigurasjonen slik at alle parametere som ikke skulle ha blitt satt, blir satt.

Mot slutten av uken er det meste at Openstack installasjonen rulle ut hvor det fortsatt er noen problemer med nettverket på virtuelle maskiner som opprettes. Dette har kokt ned til at det er kun problemer med DHCP'en i nettverket og de virtuelle maskinene ikke mottar IP-adresse. Nettverk mellom de virtuelle maskinene fungerer når IP-adressen settes manuelt.

Som en ekstrafunksjon til oppgaven, har også web-grensesnittet for Openstack, Horizon, blitt implementert i automatiseringsprosessen. Denne Openstack -komponenten er ikke strengt tatt nødvendig, men siden implementasjon av Horizon går raskt, valgte vi å ta den med. Dette web-grensesnittet har også sine fordeler hvor konfigurasjon av nettverk og VM'er kan plasseres i en GUI hvor det er enkelt å holde oversikt på oppsettet.

Uke16

Denne uken skulle avrunde det tekniske/utviklingsfasen for dette prosjektet, slik at vi får tid til bearbeiding av rapporten frem mot innlevering til veileder, og videre endelig sluttrapport 19. Mai. På begynnelsen av uken ble det lagt ned mye tid til å løse de siste utfordringene i forbindelse med nettverk og automatisering av dette.

Etter dette delte gruppen seg opp, slik at det ble arbeidet parallelt med automatisering, dokumentasjon og Munin. I forbindelse med implementering av overvåkning ble det tidlig klart at vi ikke ville få tid til å sette opp Nagios, men Munin ble ferdig i siste liten med de ønskede innstillingene.

Det ble påbegynt arbeide med dokumentasjon og ellers bearbeiding av rapporten. Antageligvis vil vi trenge tiden fremover får å komme i mål med rapportskrivningen, men skulle vi føle oss ferdig med dette i god tid vil det være muligheter for å se nærmere på noen av de funksjonalitetene vi ikke fikk tid til å implementere i utgangspunktet, eventuelt forbedring av eksisterende løsning. Det er på flere områder muligheter for å gjøre løsningen penere/mer robust.

Uke17

I denne perioden har dokumentasjon og generell bearbeiding av rapport tatt hovedfokus for gruppen. Det ble tidlig denne uken generert en oversikt over de områdene som gjenstår, og gruppen har ut i fra det bearbeidet disse punktene i en hensiktsmessig rekkefølge. Blant punktene som har vært i fokus denne uken finner man statusrapport, konklusjon, kravspesifikasjon, og ellers generell bearbeiding av de ulike tekstlige seksjonene.

Endelig har VM'er opprettet i skyløsningen fått internett-tilgang. Det ble raskt testet å sette opp en VM som webserver som servet en tilfeldig nettside. Når denne siden var tilgjengelig fra skolenettet, hadde vi fått verifisert at oppsettet fungerte som det skulle. Web-serveren vil være tilgjengelig fra internett vha. VPN inn til skolenettet. Fokuset mot slutten av uken var å få på plass et script som automatiserer utrulling av et standard-nettverk med noen VM'er hvor èn av VM'ene skal få tildelt en ekstern-ip-adresse.

Ukelogg

Uke18

I nest siste fulle uke av prosjektarbeidet ble det arbeidet i all hovedsak på rapporten. Dette innebærer retting av tekst ut i fra tilbakemeldinger vi har mottatt, formatering av innhold, og forbedring av innhold der vi ser dette har vært nødvendig
Det ble generert en brukerveiledning som tjener til hensikt å tilby sluttbrukere av OpenStack hjelp med noen av de grunnleggende funksjonene man finner i webgrensesnittet.

Uke19

Gruppen ser ikke behov for å skrive mer logg her ifra og ut prosjektperioden, ettersom vi nå kommer til å ha fullt fokus på rapportskriving.

Møtelogg

Møtelogg AutoStack

Mai 2014

1

Møtelogg

7 Januar - 2014

Deltagere

Erik Hjelmås
Anders Godtland Noem
Kjetil André Finsrud
Geir André Tufte

Tid og Sted

Sittegruppe utenfor kontoret til Erik, K-bygget.
30 minutter.

Referat

1. Erik har reservert følgende servere til bacheloroppgaven;
   Newman, Jerry, Kramer, Nostrand, Bubbleboy, Bania.
2. Newman vil fungere som PuppetMaster/CobblerServer/OpenStack
3. Hvem av OpenStackgruppene skal ta hvilke skap?
4. Openstack Havannah er versjonen av Openstack vi skal bruke i denne oppgaven.
5. Openstack.org har en steg for steg guide til hvordan implementere openstack.
6. Under forprosjektet; Ha fokus på hva vi skal løse. Sensor vil se på OpenStack oppgaven som er gitt som utgangspunkt i vurderingen. Legge vekt på å skrive en god og solid kravspesifikasjon til forprosjektet!

Møtelogg

15 Januar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Ikke bruk for mye tid på forprosjektet! Finpussing på den vil ikke gi noen uttelling.
2. Det er selve jobben som lager grunnlaget for en god sluttrapport. Kom igang med arbeidet.
3. Bakgrunnen for prosjektet, Eriks generelle interesse for automatisering. Tilfeldigvis passer denne oppgaven også sammen med CCIS hvor det er kommet frem at spesifikke øvingscenarier er noe flere bedrifter/
   organisasjoner i dag ikke får utført på noe god måte.
4. I utarbeidelse av kravspesifikasjon, ta utgangspunkt i at det er en
   bruker/organisasjon.
5. Newman skal kjøre ubuntu server 13.10.
6. Alle serverne rullet ut skal kjøre ubuntu server 13.10
7. Husk å skrive opp referanser! (t.o.m. skyhigh.pdf)
8. Vi har fått 3stk disker som kan settes inn i servere som per i dag ikke har noen diskplass. Dette for å kunne komme igang å teste cobbler og utrulling av OS.
9. Ang. overvåkningsmoduler. Rull ut munin OVERALT.
10. Oppsettet med virtuelle maskiner; anbefalt å kikke på tidligere bacheloroppgave Jungel for idéer.
11. Vi bør rulle ut noen OS med cobbler i første omgang, evt. også innstallere puppet og få dette til å fungere. Deretter installere OpenStack iht. guide på utrullet server(e) og gå gjennom installasjonsprosessen først manuelt. Deretter kan jobben med å automatisering starte.
12. Newman trenger ikke å ha OpenStack installert - Trenger kun å initielt rulle ut OS med cobbler og deretter kjøre puppet-manifester.
13. Vi kan se på automatiseringsprosessen som todelt. Første del vil fokusere på å rulle ut et OpenStack-miljø (automatisk). Andre del vil være å kunne tilby spesifikke scenarier som blir automatisk implementert i OpenStack-miljøet.
14. Møter fremover er satt til ONSDAGER klokka 11:30, møterom MY07.

Møtelogg

22 Januar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Nå som cobbler er mer eller mindre satt opp blir neste utfordring å få innstallert puppet-agent som et punkt i preseed-fila som benyttes under den automatiske installasjonen.

2. Det er først ved bruk av puppet det vil bli aktuellt å skille på typer hardware som skal settes opp i forbindelse med OpenStack utrullingen. I Cobbler kjører vi bare en generell fil som bør gjelde for alle typer hardware.

3. OpenStack har et særegent oppsett for hvordan de forskjellige nodene skal settes opp. Derfor må det bli slik at den første serverene Newman kontakter under en OpenStack utrulling, må bli kontroller-noden, deretter netverksnoden, deretter compute noder, etc.

4. Er det mulig å gjøre PXE-booting 100% automatisert? Dvs. kan man koble en server til nettet (bak Newman), starte serveren og derfra skal alt skje automatisk? Dette vil innebære at første gang serveren booter skal den velge PXE-boot og etter at innstallasjonen er ferdig skal den velge å boote fra disk (viss ikke vil en ny PXE-boot skje og ny OS-utrulling starte). Selv om dette kanskje ikke er mulig er det verdt å legge ned litt tid til å undersøke og skrive en paragraf eller to i sluttrapporten for å vise at det er blitt vurdert.

5. Svenske FOI har en tjeneste som i noen grad kan likne det vi skal lage. De tilbyr virtuelle scenariebaserte miljøer som er basert på predefinerte lab-miljøer. Disse lab-miljøene kan være tilgjengelig for oss også, noe som kan være verdt å undersøke etterhvert som det blir aktuellt å kunne opprette scenarier.

6. ProPuppet sec.ed. er en god bok ("-Den beste") for å lære seg puppet (kan være verdt å se på, men ikke nødvendig ift. oppgaven).

7. Keystone, Nova, Glance og Neutron er de modulene av OpenStack vi skal ha fokus på. Neutron som er nettverksmodulen er en veldig fleksibel modul noe som er bra, men som også kan øke kompleksiteten. Tips: Start rolig med Neutron, få til det enkleste først!

8. Under skriving av sluttrapporten, dokumenter det som går galt og det som er blitt testet! Viktig å vise troubleshooting ferdigheter.

Møtelogg

29 Januar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Vi kan se på oppgaven vår som en utviklingsoppgave og skal derfor bruke uviklingsmalen for sluttrapporten.
2. Kunne det på sikt vært mulig å hatt f.eks en nettside autostack.org og rulle ut openstack derifra?
3. Vi må finne ut hvilken informasjon vi trenger før vi starter en utrulling (eksempelvis mac-adresser). Med dette kan er det mulig å kunne på forhånd opprette hvordan utrullingen skal se. Hvis slik informasjon ikke er tilgjengelig, må vi kunne ha en definert sekvens som ruller ut bestemte noder på bestemte tidspunkt.
4. Under puppet installasjonen trengs kun *hosts* fila og endres slik at en initiell oppkobling mot puppetmaster kan utføres, deretter kan hvilken som helst konfigurasjon rulles ut.
5. Vi kan bli flinkere til å ta ibruk personer til å hjelpe oss med oppgaven.
6. Vi går bort fra å bruke Cobbler sin innebygde puppet konfigurasjon og bruker heller kickstarter skript.

Møtelogg

5 Februar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. I sluttrapporten har vi kapitellet *Implementasjon* etterfulgt av *Testing*. Da det er flere emner som egentlig tilhører testfasen blir det nødvendig å skrive om disse emnene i implementasjonsfasen da dette vil gi den mest naturlig sekvensielle flyten i rapporten.

2. Compute-nodene som tar del i OpenStack-løsningen kan ha nøkkelbasert authentisering mot kontroller-noden (Nova). Dette for å slippe passord.

3. Det er viktig at vi drøfter hvorfor vi tar de valgene vi tar. Dette eksemplevis mtp. first-boot scriptet vi har valgt fremfor annen integrert Cobbler funksjonalitet.

4. Ang. rolletildeling på servere. Neutron vil stå alene. Kontrolleren vil i tillegg til OpenStack-modulen Nova ha modulene Keystone og Glance. Kontrolleren vil også ha message-queue, RabbitMQ. Da kontrolleren inneholder flere moduler og spesielt mtp. Glance, som håndtere image, vil kontrolleren kreve mer lagringskapasitet enn Neutron.

5. Neste møte vi bli 08:30, fredag 14.02, samme møterom.

Møtelogg

14 Februar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Det er mulig å teste OpenStack uten å ha fungerende nettverk vha. Horizon. Dvs. opprette VM'er på compute nodene. Dette blir aktuelt å teste så fort som mulig for å avdekke om de problemene vi møter ligger i nettverksnoden (Neutron).

2. For at vi ikke skal bruke veldig mye tid på å feilsøke problemer med OpenStack har Erik sagt at hvis vi feilsøker på èn ting mer enn èn dag, kan vi out-source problemet til han.

3. Mens noen jobber med OpenStack installasjon, kan noen andre starte å lese om lab-miljøene svenske FOI tilbyr (cyber range). Dette vil bli relevant til scenario ut-rullinger i AutoStack

4. Hvordan skal vi løse nettverksoppsettet? Man vil alltid ha den manuelle jobben med å koble opp systemet hos en kunde før selve utrullingen kan skje. Det vil være unaturlig at en kunde får innsyn i hvordan vi sette opp løsningen før ut-rulling, så dette blir en oppgave AutoStack tar. Det er blitt diskutert flere tilnærminger til hvordan dette nettverket kan settes opp og hvordan rolletildeling skal foregå, det er derfor viktig at de forskjellige tilnærmingene dokumenteres og at det argumenteres for og imot.

5. Frem til nå har det ikke vært veldig kritisk med versjonskontroll, men det vil bli aktuelt når vi starter å skrive lange, store manifester. Vi bør med dette sette opp en Git server (evt. SVN) for manifestene og evt. andre skript vi lager. Erik vil kunne holde et lynkurs i Git etterhvert. Boken ProPuppet har en guide for hvordan Git kan effektivt brukes sammen med Puppet.

Møtelogg

19 Februar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Det finnet et formelt krav om at det skal skrives stautsrapporter underveis i oppgaven (4stk). Disse må skrives og ha med som vedlegg i sluttrapporten. Veileder fører egen statusrapport.
2. Det er viktig at vi dokumenterer den manuelle jobben vi har hatt med å sette opp OpenStack. Den manuelle jobben er viktig prosess for å på sikt få nok kunnskap om OpenStack til å kunne automatisere det.
3. OpenStack modulene til Puppet er moduler som inneholder mange variable for å ta høyde for alt av hvilken versjon som kjøres til hardware etc. Erik anbefaler med dette at vi først ser på å automatisere deler av løsningen uten å benytte OpenStack modulene, da disse er veldig store og komplekse. Når vi har god oversikt over egenkomponerte manifester som automatisk setter opp deler av OpenStack, kan vi begynne å utforske OpenStack modulene til Puppet.
4. Hvis vi får tid og automatiseringen blir "fullstendig" kan det være veldig aktuelt med en live-demo under presentasjonen av oppgaven.

Møtelogg

26 Februar - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Problemene vi har med nettverk på instanser som opprettes kan skyldes oppsett av dhcp-agent.ini på neutron.
2. Navn på OpenStack-modulene i Puppet skal ha en bestemt navnkonvensjon, openstack-%NAVN%
3. Noen av serverene våre har kun to nettverkskort. Dette begrenser dem til å ha enkelte roller i OpenStack. Skulle dette bli et problem, har Erik flere nettverkskort liggende.
4. Skulle det bli aktuelt å hente informasjon publisert i en eller annen bok, har vi muligheten til å få tilgang til denne informasjonen gjennom Safari-books-online.
5. Det er essensielt å få opp GIT sammen med Puppet slik at noder kan test-konfigureres iht. et utviklermiljø, for så å ha muligheten for å rulle tilbake eventuelle endringer.

Møtelogg

5 Mars - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Få kontroll på Git og lære seg bruken av det når vi skal starte med Scrum-fasen av oppgaven som innebærer å lage skrift og Puppet-manifester.

2. Prøv å holde store bolker med kode utenfor hoveddokumentet og heller referer til vedlegg.

3. Dokumentasjonen av den manuelle installasjonen av Openstack samt
   Puppet-utrullingen av denne, har vært vanskelig å plassere i rapporten. Det er verken testing eller implementasjon men heller en prosess for å nå det endelige målet. Etter råd fra veileder plasseres disse bolkene under implementasjon.

4. Det vil fremover bli aktuelt at alle i gruppen setter seg godt inn i Puppet og klargjør for å benytte de modulene vi skal bruke i sluttproduktet av løsningen. Vi vil med dette jobbe videre iht. Scrum-modellen med sprinter på ca èn uke. Et bestemt antall dager en sprint skal bestå av må nærmere avklares.

Møtelogg

12 Mars - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Hanno har mulighet for å se på rapporten og gi tilbakemeldinger på hvordan vi ligger an og hva vi bør gjøre anderledes.

2. Ang. valg av Openstack-moduler til Puppet har det vært aktuelt å velge den modulen som setter opp alle rollene i Openstack på èn node. Dette utgår mtp. dette ikke er relevant for multinode-oppsettet vi bygger som er mye mer skallerbart når flere computer-noder integreres i løsningen. Vi fikk dermed bekreftelse på at valg av enkelt-moduler som tar for seg hver Openstack-node var riktig vei å gå.

Møtelogg

26 Mars - 2014

Deltagere

Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
15 minutter.

Referat

1. Hanno fikk en oppsumering av hva som gruppen hadde jobbet med de to siste ukene.
2. Det ble anbefalt å ha èn person skrivende på rapporten til enhver tid og at største delen av rapporten burde være klar tidlig mai. Hanno har også da muligheten til å gå gjennom rapporten og gi tilbakemeldinger som vi kan jobbe med før innlevering.

Møtelogg

2 April - 2014

Deltagere

Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Det er ønskelig at vi krever så lite informasjon som mulig fra kunden for å sette opp Openstack (ip-adresser, mac-adresser, etc). Dette for å gjøre det enklest mulig for kunden. En slik forenkling er mulig, men vil kreve mye mer arbeid for oss i oppgaveperioden. Det vil alltid være aktuelt å gjøre det så enkelt som mulig for kunden, men systemets hovedfunksjonalitet vil bli prioritert før en slik forenklingsprosess startes. Det vil fortsatt være aktuelt å gå gjennom listen med informasjon vi tror vi trenger fra kunden i dag og se om denne informasjonen på andre enkle måter kan innhentes.

2. Som supplerende informasjon til punktet ovenfor, vil et forenklingstiltak være å ikke gjøre Openstack-løsningen tilgjengelig på nett umiddelbart etter ut-rullingen, men kunne tilby kunden verktøy for å gjøre løsningen tilgjengelig på internett i etterkant.

3. Det er bestemt at vi skal fremover holdet høyt fokus på å få til nettverk på virtuelle maskiner som opprettes i løsningen og nedprioritere andre deler av oppgaven, eksempelvis overvåkning. Det er viktigere at vi får til kjernen med oppgaven enn tilleggsfunksjonalitet.

Møtelogg

11 April - 2014

Deltagere

Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
15 minutter.

Referat

1. Hanno fikk en oppdatering på hva gruppen har drevet med siden sist møte, som har vært å forholde seg til Openstack sin egen installasjonsguide og implementere denne installasjonsprosessen med Puppet.
2. Planen fremover vil være å fullføre det som gjenstår på Openstack installasjonen og deretter få satt opp overvåkning. Openstack installasjonen har høy prioritet, overvåking ikke, fordi det må også fokuseres på rapportskriving fremover. Som nevnt i tidligere møter, jobber vi for å rekke å bli ferdig med største deler av rapporten før 1. mai.
3. Det kan være aktuelt å få andre til å lese gjennom rapporten vår, selv om ikke rapporten er ferdig er det fortsatt enkelte kapitler som er ferdige som kan leses.
4.

Møtelogg

23 April - 2014

Deltagere

Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Det vil være naturlig å lage en brukerguide til løsningen som utvikles. Denne guiden er ikke et krav iht. oppgaveteksten og vil være noe som nedprioriteres i første omgang. Arbeidet med å lage en brukerguide kan dog være en god forberedelse for presentasjonen av oppgaven hvor vi får trening i å forklare hvordan løsningen skal brukes.

2. Overvåkingsverktøyet Nagios sløyfes i oppgaven for å prioritere rapporten og det siste tekniske som gjenstår for å få en bedre fungerende skyløsning. Det er fortsatt viktig å ta med i rapporten at det idéelle ville vært å implementere Nagios i løsningen.

3. Hvis det skulle blitt aktuelt å hvis en demo av produktet under presentasjonen, rekker vi ikke vise hele installasjonen hvis vi inkluderer utrullingen av operativsystem. Nå er ikke installasjon av operativsystem like spennende, så demoen kunne startet fra det øyeblikk nodene kobler seg til Puppet-master. En evt. demo vil kreve å ha to skjermbilder i presentasjonsrommet samtidig hvor ett bilde viser installasjonen og det andre viser presentasjonen.

Møtelogg

30 April - 2014

Deltagere

Erik Hjelmås
Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. I håp om å komme i mål med en fullverdig Openstack installasjon hvor VM'er som opprettes har internett tilgang, er det planlagt i løpet av uken å gi Erik Hjelmås og Kyrre Begnum tilgang til systemet slik at de med sin ekspertise kan feilsøke på problemet. Gruppen vil ha fokus på rapportskrivingen og hvis Erik og Kyrre løser problemet, får det raskt høy prioritert for å bli implementert i automatiseringsprosessen.

2. I oppgaveteksten står det at løsningen skal utføre dynamisk ressurstildeling ut ifra den maskinvare installasjonen oppdager. Dette er det ikke blitt lagt veldig mye arbeid i da oppdragsgiver har vært fornøyd med slik ressurstildeling skjer i dag hvor det enten er sekvensen på når servere kontakter Newman som avgjør, eller at vi innehar informasjon om mac-addresser på servere i forkant av installasjonen. Dette er viktig at blir drøftet i rapporten.

3. Det endelige målet med oppgaven er å i tillegg til å rulle ut en Openstack-implementasjon, også kunne rulle ut noen virtuelle maskiner. Disse maskinene trenger ikke ha noe spesielt innhold men er kun for å bevise at det er mulig. Her kan det da refereres til andre bacheloroppgaver som omhandler oppsett av virtuelle miljøer som eksempelvis oppgaven Jungle. Poenget er å formidle at det vil være mulig å kunne rulle ut en Jungle i Openstack-implemenasjonen vi ruller ut.

4. Det er tid for å finne personer som kan lese rapporten for å gi tilbakemeldinger på både teknisk innhold og kvaliteten på språket. Hanno får rapporten 7 mai i den tilstanden den er i da for å kunne gi tilbakemeldinger.

Møtelogg

14 Mai - 2014

Deltagere

Hanno Langweg
Anders Godtland Noem
Geir André Tufte
Kjetil André Finsrud

Tid og Sted

Møterom MY07, Mustad
30 minutter.

Referat

1. Under presentasjonen av oppgaven, vil det i tillegg til det som presenteres også stilles spørsmål til rapporten og evt. valg vi har gjort der.
2. Hele gruppen burde være i stand til å svare på spørsmål under presentasjonen.
3. Det er vanlig at alle presentere en liten bit av oppgaven, men det finnes ingen regel.
4. Hvis tiden strekker til, er det aktuelt å legge til et scenario i rapporten basert på den tidligere bacheloroppgaven Jungle.

Tillegg G

Prosjektavtale

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

ERIK HJELMÅS

(oppdragsgiver), og

KJETIL ANDRÉ FINSRUD

ANDERS GODTLAND NOEM

GEIR ANDRÉ TUFTE (student(er))

Avtalen angir avtalepartenes plikter vedrørende gjennomføring av prosjektet og rettigheter til anvendelse av de resultater som prosjektet frembringer:

1. Studenten(e) skal gjennomføre prosjektet i perioden fra 8.1.2014 til 19.05.2014.

   Studentene skal i denne perioden følge en oppsatt fremdriftsplan der HiG yter veiledning. Oppdragsgiver yter avtalt prosjektbistand til fastsatte tider. Oppdragsgiver stiller til rådighet kunnskap og materiale som er nødvendig for å få gjennomført prosjektet. Det forutsettes at de gitte problemstillinger det arbeides med er aktuelle og på et nivå tilpasset studentenes faglige kunnskaper. Oppdragsgiver plikter på forespørsel fra HiG å gi en vurdering av prosjektet vederlagsfritt.

2. Kostnadene ved gjennomføringen av prosjektet dekkes på følgende måte:
   - Oppdragsgiver dekker selv gjennomføring av prosjektet når det gjelder f.eks. materiell, telefon/fax, reiser og nødvendig overnatting på steder langt fra HiG. Studentene dekker utgifter for trykking og ferdigstillelse av den skriftlige besvarelsen vedrørende prosjektet.
   - Eiendomsretten til eventuell prototyp tilfaller den som har betalt komponenter og materiell mv. som er brukt til prototypen. Dersom det er nødvendig med større og/eller spesielle investeringer for å få gjennomført prosjektet, må det gjøres en egen avtale mellom partene om eventuell kostnadsfordeling og eiendomsrett.

3. HiG står ikke som garantist for at det oppdragsgiver har bestilt fungerer etter hensikten, ei heller at prosjektet blir fullført. Prosjektet må anses som en eksamensrelatert oppgave som blir bedømt av faglærer/veileder og sensor. Likevel er det en forpliktelse for utøverne av prosjektet å fullføre dette til avtalte spesifikasjoner, funksjonsnivå og tider.

4. Den totale besvarelsen med tegninger, modeller og apparatur så vel som programlisting, kildekode, disketter, taper mv. som inngår som del av eller vedlegg til besvarelsen, gis det en kopi av til HiG, som vederlagsfritt kan benyttes til undervisnings- og forskningsformål. Besvarelsen, eller vedlegg til den, må ikke nyttes av HiG til andre formål, og ikke overlates til utenforstående uten etter avtale med de øvrige parter i denne avtalen. Dette gjelder også firmaer hvor ansatte ved HiG og/eller studenter har interesser.

   Besvarelser med karakter C eller bedre registreres og plasseres i skolens bibliotek. Det legges også ut en elektronisk prosjektbesvarelse uten vedlegg på bibliotekets del av skolens internett-sider. Dette avhenger av at studentene skriver under på en egen avtale hvor de gir biblioteket tillatelse til at deres hovedprosjekt blir gjort tilgjengelig i papir og nettutgave (jfr. Lov om opphavsrett). Oppdragsgiver og veileder godtar slik

offentliggjøring når de signerer denne prosjektavtalen, og må evt. gi skriftlig melding til studenter og dekan om de i løpet av prosjektet endrer syn på slik offentliggjøring.

5. Besvarelsens spesifikasjoner og resultat kan anvendes i oppdragsgivers egen virksomhet. Gjør studenten(e) i sin besvarelse, eller under arbeidet med den, en patentbar oppfinnelse, gjelder i forholdet mellom oppdragsgiver og student(er) bestemmelsene i Lov om retten til oppfinnelser av 17. april 1970, §§ 4-10.

6. Ut over den offentliggjøring som er nevnt i punkt 4 har studenten(e) ikke rett til å publisere sin besvarelse, det være seg helt eller delvis eller som del i annet arbeide, uten samtykke fra oppdragsgiver. Tilsvarende samtykke må foreligge i forholdet mellom student(er) og faglærer/veileder for det materialet som faglærer/veileder stiller til disposisjon.

7. Studenten(e) leverer oppgavebesvarelsen med vedlegg (pdf) i Fronter. I tillegg leveres et eksemplar til oppdragsgiver.

8. Denne avtalen utferdiges med et eksemplar til hver av partene. På vegne av HiG er det dekan/prodekan som godkjenner avtalen.

9. I det enkelte tilfelle kan det inngås egen avtale mellom oppdragsgiver, student(er) og HiG som nærmere regulerer forhold vedrørende bl.a. eiendomsrett, videre bruk, konfidensialitet, kostnadsdekning og økonomisk utnyttelse av resultatene.

   Dersom oppdragsgiver og student(er) ønsker en videre eller ny avtale, skjer dette uten HiG som partner.

10. Når HiG også opptrer som oppdragsgiver trer HiG inn i kontrakten både som utdanningsinstitusjon og som oppdragsgiver.

11. Eventuell uenighet vedrørende forståelse av denne avtale løses ved forhandlinger avtalepartene i mellom. Dersom det ikke oppnås enighet, er partene enige om at tvisten løses av voldgift, etter bestemmelsene i tvistemålsloven av 13.8.1915 nr. 6, kapittel 32.

12. Deltakende personer ved prosjektgjennomføringen:

HiGs veileder (navn): HANNO LANGWEG

Oppdragsgivers kontaktperson (navn): ERIK HJELMÅS

Student(er) (signatur): K.A. Eimund dato 12/1-14

Andreas Nøeen dato 12/1-14

Geir Andre Tuhl dato 12/1-14

dato

Oppdragsgiver (signatur): dato 13/1-14

IMT Dekan/prodekan (signatur): dato

Versjon Januar 2011hb