BACHELOROPPGAVE:

SkyHiGh ADM

FORFATTERE:

Lars Erik Pedersen

Jon Arne Westgaard

Hallvard Westman

DATO:

23.05.2012


SKYHIGH

Sammendrag av Bacheloroppgaven

Tittel:	SkyHiGh ADM	Nr: -
		Dato: 23.05.2012
Deltakere:	Lars Erik Pedersen	
	Jon Arne Westgaard	
	Hallvard Westman	
Veiledere:	Hanno Langweg	
Oppdragsgiver:	Erik Hjelmås, Høgskolen i Gjøvik	
Kontaktperson:	Erik Hjelmås, Høgskolen i Gjøvik	
Stikkord	OpenStack, nettsky, virtualisering, implementasjon	

Antall sider: 169	Antall vedlegg: 8	Tilgjengelighet: Åpen

Kort beskrivelse av bacheloroppgaven:

Målet for oppgaven er å implementere OpenStack som rammeverk for virtuelle datalaber ved Høgskolen i Gjøvik. Implementasjonen er ment å erstatte dagens løsning for dette. Rapporten presenterer prosessen med å realisere dette prosjektet, og omfatter installasjon, implementasjon, konfigurasjon og utvikling.

Summary of Graduate Project

Title:	SkyHiGh ADM	Nr: -
		Date: 23.05.2012
Participants:	Lars Erik Pedersen	
	Jon Arne Westgaard	
	Hallvard Westman	
Supervisor:	Hanno Langweg	
Employer:	Erik Hjelmås, Høgskolen i Gjøvik	
Contact person:	Erik Hjelmås, Høgskolen i Gjøvik	
Keywords	OpenStack, nettsky, virtualisering, implementasjon	
Pages: 169	Appendixes: 8	Availability: Open

Short description of the main project:
This thesis is based on an implementation of OpenStack as a framework for virtual computer labs at Gjøvik University College. This implementation is intended to replace the current solution for this purpose. The report presents the process of realizing this project, and includes the installation, implementation, configuration and development that has been done.

Forord

"I've come to realise that getting OpenStack up and running does earn you the title of Jedi Knight. This is Cloud Wars, right? [29]

Som aktive brukere av Amazons skytjenester og erfaring fra bruk av virtuelle maskiner i undervisning ved HiG, var denne oppgaven tiltalende for oss. Siden to av gruppens medlemmer allerede hadde utført forprosjektet som førte til dette oppgaveforslaget fikk vi også en uoffisiell førsterett på oppgaven. Prosjektet ble delt i to deler, fordi en annen gruppe også meldte interesse. Oppdragsgiver så det hensiktsmessig å dele prosjektet inn i en administrasjonsdel (dette prosjektet), og en del som tok for seg ytelsestesting av systemet (SkyHiGh I/O).
Rapporten presenterer våre erfaringer ved implementasjon av OpenStack, og hvilke forhåndsregler man må ta dersom man ønsker å benytte seg av dette rammeverket.

Vi retter en takk til uksysadmin, mjfork, kiall, ohnoimdead (aktive OpenStack brukere), samt resten av de hjelpsomme sjelene på #OpenStack@freenode for support over IRC. Deres kunnskap og hjelpsomhet har bidratt i stor grad til å drive prosjektet fremover.

Vi takker også veileder Hanno Langweg og oppdragsgiver Erik Hjelmås for god hjelp under prosjektet, samt Jon Langseth for god hjelp med nettverksoppsett.

Innholdsfortegnelse

Figurer

Tabeller

1 Innledning

1.1 Bakgrunn

Nettskyer har fått mye oppmerksomhet de siste årene, og er en teknologi som gjør det mulig for sluttbrukere å kjøre *virtuelle maskiner* på andres maskinvare og infrastruktur. Dette åpner for muligheter for dynamisk skalering av ressurser, fleksibilitet og økonomiske besparelser.
Slike skyer kan man benytte seg av i undervisning hvor man da kan bruke disse til forskning eller som lab til studenter. Skulle man komme i skade for å kjøre en kommando eller gjøre en endring som ødelegger konfigurasjonen, er det bare å utføre et par tastetrykk for å gjenopprette opprinnelig konfigurasjon.

I emnene Ethical Hacking And Penetration Testing [93], Systemadministrasjon [24] og Database- og applikasjonsdrift [9] brukes virtuelle maskiner. Her får elevene utdelt ett sett med virtuelle maskiner som man da kan kjøre tester og gjøre oppgaver på. På denne måten får man nytte av praktisk undervisning, uten å måtte sette opp fysiske maskiner.
Dagens løsning er basert på MLN [45], som er et administrasjonsverktøy for virtuelle nettverk og maskiner som støtter Xen, VMware Server og User-Mode Linux. Ved bruk av MLN er det en del utfordringer i følge oppdragsgiver:

- Kompleks på administrasjonssiden
- Har ytelsesproblemer
- Skalerer ikke i form av maskinkraft
- Administreringen av MLN gjøres i egne konfigurasjonsfiler
- Konfigurasjonen skrives i MLNs eget språk.
- Har ikke noe grafisk grensesnitt
- All administrasjon gjøres via kommandolinjen av en administrator
- Når en hel klasse bruker de virtuelle maskinene samtidig går mye av tiden til å vente på at kommandoer skal kjøres og at de virtuelle maskinene skal respondere.

Et forprosjekt i emnet systemadministrasjon konkluderte med at OpenStack er den beste skyløsningen basert på åpen kildekode for å løse disse problemstillingene [39].

1.2 Oppgavebeskrivelse

Formålet med bacheloroppgaven SkyHiGh ADM er å sette oppe en implementasjon av OpenStack. Med bruk av dette systemet skal SkyHiGh ADM se på muligheten for å erstatte og utvide dagens løsning for virtuelle datalaboratorier ved Høgskolen i Gjøvik.
Den nye løsningen skal kunne brukes selvstendig av både studenter og emneansvarlige,

slik at administrator får mindre arbeidslast enn han har i dag. I tillegg skal den nye løsningen være skalérbar, noe den nåværende ikke er.

Oppdragsgiver ønsker en løsning basert på OpenStack, som er et Open Source Cloud Computing-prosjekt. OpenStack er modulbasert og har gode muligheter for skalering. Dette blir nyttig i fremtiden ved tilbud om datakraft som tjeneste til studenter og ansatte i andre sammenhenger enn til undervisningsbruk.
Per dags dato er det ikke mulig for studenter/faglig ansatte, på en enkel måte, å opprette og administrere en eller flere virtuelle maskiner. OpenStack har et webgrensesnitt som forenkler administrasjon noe, men dette må utvikles for å tilfredsstille eksisterende funksjonalitet i MLN.

Gruppen har selv satt seg ett mål:

- Å plassere Gjøvik på OpenStack-kartet [60].

1.3 Organisering av rapporten

1. Innledning - En introduksjon av rapporten, som gir en kort innføring av oppgavens innhold og struktur.
2. Kravspesifikasjon - I kravspesifikasjonen vil gruppen spesifisere kravene som er stilt av oppdragsgiver.
3. Teoretisk grunnlag- Kapittelet gir en grundig teoretisk gjennomgang av alle modulene og deres struktur i OpenStack. Dette er nødvendig for å bygge opp en forståelse for systemet som er implemenert.
4. Design - Design vil utvide designdokumentet som er utredet på bakgrunn av kravspesifikasjonen. Her vil spesifikke løsninger på kravene bli fremstilt.
5. Gjennomføring- Gjennomføringen av prosjektet blir her beskrevet i de forskjellige fasene som prosjektet har gjennomgått og redegjøre for prosessen i sin helhet.
6. Driftsrutiner- Driftsrutine som beskriver aktiviteter under den daglige driften av systemet for å ivareta ytelse, pålitelighet, tilgjengelighet og sikkerhet.
7. Diskusjon - I dette kapittelet diskuteres det om hvorvidt OpenStack faktisk er rett rammeverk for å erstatte dagens virtualiseringsløsning.
8. Avslutning - Her kommer konklusjonen av oppgaven, samt en evalurering av prosjektarbeidet.
9. Litteraturliste og referanser - Oversikt over litteratur og referanser brukt i denne rapporten
10. Vedlegg - Dokumenter og figurer som er relevante for rapporten: Forprosjektrapport, kildekode, script, tester, arbeidslogg, figurer og møtereferater.

1.4 Konvensjoner

I denne rapporten vil ordet ”gruppen” referere til samtlige gruppemedlemmer: Hallvard Westman, Lars Erik Pedersen, Jon Arne Westgaard. De seksjoner som omhandler direkte

handlinger, eller evaluering av gruppen, kommer ordet ”vi” til å brukes med samme betydning som ordet ”gruppen”.

- Ord som befinner seg i ordlisten vil være formatert i *kursiv* ved første anvendelse, i etterkant formateres de normalt.
- Ord som blir forklart i teksten vil også være formater med *kursiv* ved første anvendelse, men vil ikke befinne seg i ordlisten.
- Kommandoer og utdrag fra script/kode er formatert med `denne fonten`.
- For kildehenvisning benyttes vancouver-stilen [88].

1.5 Formål

Formålet med prosjektet er først og fremst å bedre brukeropplevelsen for de som benytter seg av virtuelle maskiner og virtuelle datalaboratorier ved Høgskolen i Gjøvik. Både studenter, emneansvarlige og driftsansvarlige skal få en mer effektiv hverdag. Det skal også bli mindre ansvar på én enkelt administrator. Vi håper at resultatet av dette blir mer tid frigjort til forskning for de som har stått ansvarlige for den tidligere løsningen, og at vi bidrar til mer effektiv læring for studentene. Formålet skal oppnås ved å erstatte dagens løsning med nettskyrammeverket OpenStack.

1.6 Målgruppe

1.6.1 Løsningens målgruppe

SkyHiGh-prosjektet er startet av oppdragsgiver Erik Hjelmås fra IMT-avdelingen ved Høgskolen i Gjøvik. Oppdragsgiver har stått som eneste administrator av MLN, og har følgelig hatt alle arbeidsoppgaver i forbindelse med behovet for virtuelle maskiner ved HiG frem til nå. Dette prosjektet retter seg mot emneansvarlige og studenter, ved å tilby et verktøy med større grad av selvstendighet. I tillegg vil oppdragsgiver være en selvskreven målgruppe, da prosjektet skal føre til mindre arbeidsbelastning for administrator.

1.6.2 Rapportens målgruppe

Rapporten retter seg først og fremst mot oppdragsgiver og IMT-avdelingen ved HiG, samt andre teknologiinteresserte utenfor skolen. Med bakgrunn i denne målgruppen, forventes det en teknologisk forståelse på nivå med en høyere utdanning i informatikk.

1.7 Avgrensning

Prosjektet skal først og fremst implementere OpenStack-rammeverket for å virkeliggjøre målene nevnt i vedlegg A. Vi vil ikke utføre ytelsestesting og analyse da dette dekkes av bacheloroppgaven SkyHiGh IO. Mulighetene for *high availability* og redundans skal ikke dekkes. Primært skal systemet utvikles for å bli brukt på HiG, for de aktuelle emnene, ikke for eksterne brukere.

1.8 Studentenes faglige bakgrunn

Gruppemedlemmene kommer fra to forskjellige studieretninger. Jon Arne Westgaard og Lars Erik Pedersen studerer Drift av Nettverk og Datasystemer. Hallvard Westman studerer Programvareutvikling. Hallvard har gode kunnskaper innen forskjellige programmeringsspråk og utviklingsmetoder. Jon Arne og Lars Erik stiller med kompetanse innenfor nettverk og scripting i både Bash og Perl, fra tidligere emner ved Høgskolen i Gjøvik. Alle har gode kunnskaper innen Linux, SQL og systemutvikling, samt C++ og Java fra programmeringsfag ved HiG.

Under prosjektet må samtlige gruppemedlemmer tilegne seg gode kunnskaper og ferdigheter innen Python som programmeringsspråk. Det vil også være nødvendig for samtlige å tilegne seg en overordnet forståelse for Django som rammeverk. Hovedsaklig vil det være Hallvard som må tilegne seg dybdekunnskap her . I tillegg vil det være nødvendig for alle å tilegne seg god kunnskap om OpenStack-rammeverket og dets API'er. Dette er områder ingen av gruppmedlemmene har erfaring med, og må derfor læres fra bunnen av.

1.9 Roller

Vår oppdragsgiver er førsteamanuensis Erik Hjelmås ved Høgskolen i Gjøvik. Oppdragsgiver vil være en sterk faglig ressurs, og være til god hjelp med det tekniske innholdet av prosjektet. Førsteamanuensis Hanno Langweg er vår veileder. Han vil bistå med teoretisk hjelp rundt det å arbeide i et stort prosjekt, samt komme med innspill til arbeidet underveis.
Arbeidsoppgavene i gruppen ble fordelt slik:

- Lars Erik Pedersen
  Valgt som prosjektleder og kontaktperson, og har det helhetlige ansvaret for gruppa.

- Hallvard Westman
  Webansvarlig samt dedikert systemutvikler.

- Jon Arne Westgaard
  Har ansvaret for installasjoner.

1.10 Arbeidsrammer

Gruppen har arbeidsdag fra kl. 8 - 16 mandag til og med torsdag. Fredag er ikke arbeidsdag på grunn av andre forelesninger. Arbeidstiden vil hovedsaklig bli brukt på grupperommet vi har fått tildelt av skolen. Alt utstyr som er lånt er plassert på dette grupperommet. Hjemmekontor skal bare benyttes unntaksvis, da vi er avhengig av å være i umiddelbar nærhet av utstyret, samt at å jobbe mest mulig felles er noe som skal etterstrebes.

1.11 Terminologibruk

Liste over terminologier:

- VM/virtuell maskin: En simulert versjon av et operativsystem
- Hypervisor: En programvare som ligger i bunnen og behandler og administrerer virtuelle maskiner, kontrollerer ressursene til de virtuelle maskinene (prosessor, ram, disk o.l.)
- Image: En diskfil med en ferdig installert virtuell maskin
- Flavor: Et sett med virtuell maskinvarekonfigurasjon. Antall virtuelle prosessorer, mengde RAM, lagringsplass
- Instans: Se VM.
- Tenant: Et prosjekt (OpenStack-terminlogi). Instansene er bundet til et prosjekt
- IP-pool: En rekke IP-adresser
- MVC: Model view controller, et pattern for å skille logikk,data og presentasjon.
- View: Programkode som genrerer data for visning av en nettside.
- Nettverksbro: Et logisk nettverkskort, som binder sammen to L2-nettverk
- LDAP: Lightweight Directory Access Protocol
- KVS: Key Value Storage
- L2-nettverk: OSI Data link layer. Protokollen som frakter mellom grensesnitt. F.eks ethernet
- L3-nettverk: OSI Networking layer. Protokollen for adressering av nettverksressurer (IPv4)
- DHCP-relay: Tjeneste som lytter etter DHCPREQUEST, og videresender disse til en gitt IP-adresse
- High availability: Redundant nettverksoppsett
- IRC: Internet Relay Chat. Direktemeldingssystem over internet. Organisert i forskjellige kanaler
- Proof-of-concept: Demonstrasjonsoppsett
- SSH: Secure Shell. Protokoll for sikker dataoverføring mellom nettverksressurser. Oftest brukt for fjerntilgang på kommandolinje, og filoverføring
- API: Application Programming Interface. Grensesnitt for å benytte seg av eksisterende tjenester i egenutviklende applikasjoner.
- SSL: Secure Socket Layer. Krypteringsmekanisme for HTTP-trafikk.
- SAN: Storage Area Network. Sentralisert lagringsnettverk
- NAS: Network Attached Storage. Filsystem som monteres over nettverk
- IPAM: IP Address Management.
- DHCP: Dhynamic Host Control Protocol

- OVF: Open Virtualization Format. Format for lagring de virtuelle tjenesten som kjøres i en virtuell maskin.
- VMDK: Virtual Machine Disk. VmWare sitt filformat for disk-image
- VHD: Virtual Hard Disk. Microsoft sitt filformat for disk-image
- VDI: Virtual Disk image. VirtualBox sitt filformat for disk-image
- qcow2: QEMU copy-on-write. Strategi for dynamisk allokering av lagringsplass på et disk-image. Brukes av qemu.
- HTTP: HyperText Transfer Protocol. Protokoll for webtrafikk
- iSCSI: Internet Small Computer System Interface. IP-basert standard for lagringsnettverk
- ICMP: Internet Control Message Protocol. Protokoll for diverse statusmeldinger over nett. Brukes f.eks av ping.
- Trigger: Mekanisme som utløses i en database, ved definerte hendelser.
- NAT: Network Address Translation. Metode for å oversette en nettverksadresse til en annen. Brukt for å gi private nett internettilgang.
- LAN: Local Area Network. Fysisk lokalt nettverk.
- VLAN: Virtual Local Area Network. Metode for å skape flere virtuelle LAN i samme switch
- VTP: VLAN Trunking Protocol. En properitær cisco-protokoll for å distribuere oppsett av VLAN.
- VPN: Virtual Private Network. Protokoll for å koble seg til et LAN over internett.
- IEEE 802.1Q: Standarden for VLAN-tagging
- RC: Release Candidate. Siste testutgave av en programpakke, før ferdig stabil versjon.
- NTP: Network Time Protocol. Protokoll for tidssynkronisering
- git: Distribuert revisjonskontroll og kildekodehåndtering
- SVN: Subversion. Sentralisert revisjonskontroll og kildekodehåndtering
- bash: Bourne Again Shell. Standard shell i de fleste linuxdistribusjoner
- GUI: Graphical User Interface, det grafiske grensesnittet som blir presentert til brukeren
- Dual Stack: Kjøre IPv6 og IPv4 samtidig.
- DHCPDISCOVER: Pakker som blir sendt ut av DHCP-klienter for å finne en DHCP-server
- DHCPREQUEST: Pakker som blir sendt av DHCP-klienter, forespørsel om en gitt IP-adresse

2 Kravspesifikasjon

Et tidligere prosjekt [39] i emnet Systemadministrasjon konkluderte med at OpenStack er en anbefalt plattform for SkyHiGh-prosjektet.
Dette prosjektet omhandler i hovedsak å flytte et tjenestetilbud over på et nytt rammeverk. Kravene er stilt av eksisterende system og oppdragsgiver.

2.1 Funksjonelle krav

Dette er ikke et tradisjonelt utviklingsprosjekt, men et endringsprosjekt. Løsningen skal baseres på allerede eksiterende programvare. Vi må derfor se på eksisterende funksjonalitet i MLN for å se om våre mål tilfredsstilles i den nye løsningen. Tabell 1 reflekterer hvilken funksjonalitet som finnes i dagens løsning, og hvorvidt det må gjøres utvikling for at det skal fungere på samme måte i OpenStack.
En forutsetning for at flyttingen skal kunne sees på som gjennomførbar og suksessfull, vil være at all eksisterende funksjonalitet enten må være tilfredsstilt av OpenStacks webgrensesnitt, eller å utvide de modulene som allerede finnes. I tillegg skal alle administrasjonsoppgaver kunne utføres ved hjelp av kommandolinjen. Det vil si at eventuelle tillegg i webgrensesnittet også må realiserers i form av script. Det legges ikke opp til at studentene skal kunne gjøre operasjoner mot sine *instanser* via kommandolinjen. De skal primært benyttes seg av webgrensenittet.

2.1.1 Rammer og funksjonalitet som er tilfredsstilt av OpenStack

Oversikt

Eksisterende krav	**Nye Krav**	**OpenStack**	**Utvikles videre**
Tilbyr brukere full tilgang til et gitt antall instanser definert av administrator		Ja	Nei
IP-Adresse på HiGs nett		Ja	Nei
Rulle ut flere instanser med en gitt konfigurasjon		Nei	Ja
Administrere batcher som er rullet ut		Nei	Ja
	Mulighet for enkel administrering av instanser for sluttbrukere	Ja	Nei
	Lastbalansering	Ja	Nei
	Skalering	Ja	Nei
	Webgrensesnitt	Ja	Ja
	Opplasting av disk-*image* fra egen maskin for sluttbrukere	Nei	Ja

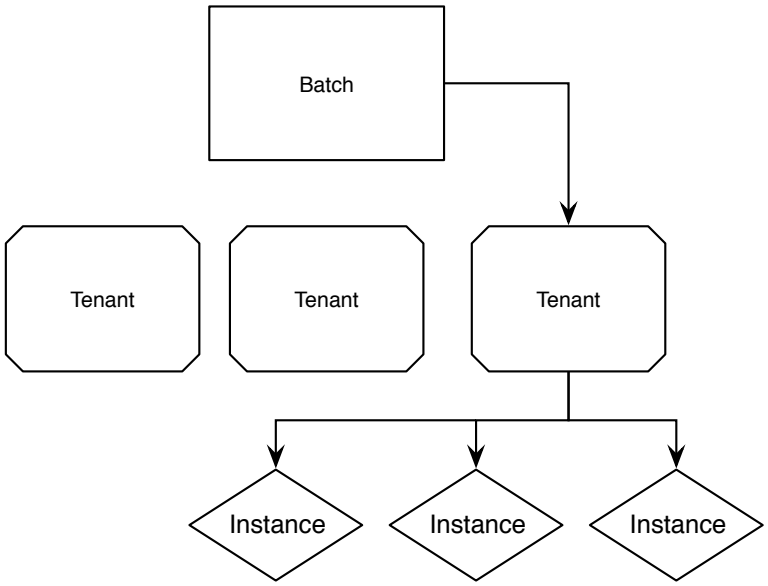
Tabell 1: Rammer og funksjonalitet som er tilfredsstilt av OpenStack

2.2 Spesifisering av krav som skal utvikles videre

2.2.1 Rulle ut flere instanser med en gitt konfigurasjon

I den eksisterende løsningen har administrator mulighet for å sette opp en konfigurasjon for et sett med virtuelle maskiner. Dette kalles prosjekter i MLN. Ett prosjekt kan også lagres, slik at det kan hentes opp igjen på et senere tidspunkt. Dette blir gjort ved hjelp av kommandolinje og konfigurasjonsfiler. I OpenStack kalles et slikt sett med instanser en *tenant*. Det finnes ingen implementasjon av en samling tenants, og følgelig må dette utvikles for å tilfredsstille eksistrende funksjonalitet. Ett sett med tenants vil fra nå av kalles for en batch. Da vil altså en tenant reflektere en gruppe i ett gitt emne slik som i figur 1, og en batch reflekterer alle gruppene i et enkelt emne.

Figur 1: Batch-abstraksjon



2.2.2 Administrere batcher som er rullet ut

Siden eksisterende løsning ikke har konseptet batch, det vil si en abstraksjon for en samling prosjekter, finnes det heller ingen mulighet for å gjøre operasjoner på en slik. F.eks å slette alle gruppene i et emne på en gang, eller å liste alle prosjekter som hører til et spesifikt emne. Denne funksjonaliteten må også implementeres i den nye løsningen, og vil være en naturlig utvidelse av kapittel 2.2.1.

2.2.3 Regnekraft som tjeneste

Det skal være mulig for studenter og ansatte å flytte en virtuell maskin fra sin egen maskin til OpenStack via webgrensesnittet, slik at man får mulighet til å teste programvare o.l på bedre maskinvare enn man har fysisk tilgjengelig på egen maskin. Denne funksjonaliteten finnes ikke i eksisterende løsning, og heller ikke i OpenStack via webgrensesnittet. Det er imidlertid støtte for å registrere disk-image i både VMWare og VirtualBox sitt diskformat, men dette skjer via kommandolinjen.

2.2.4 Webgrensesnitt

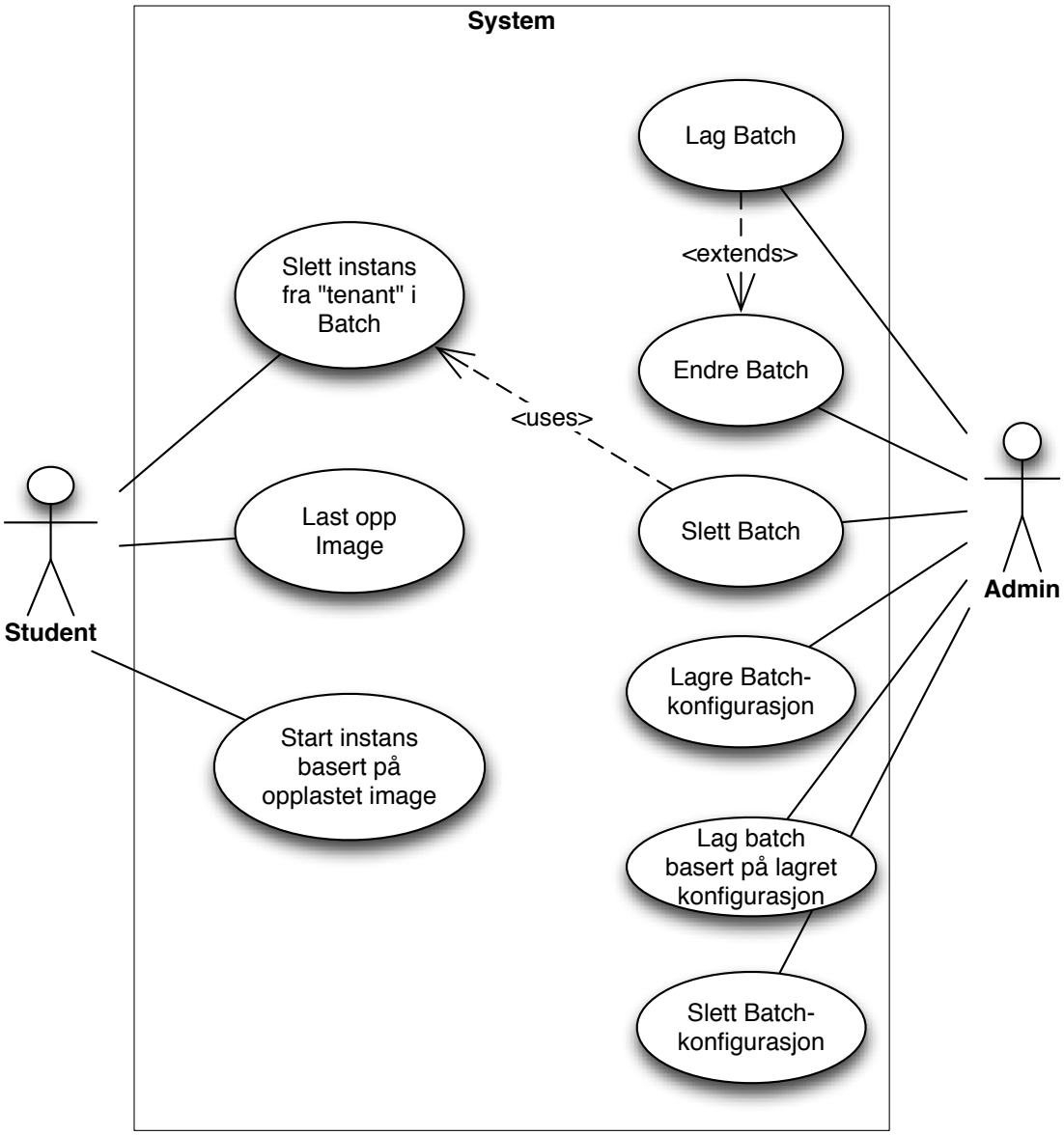
Løsningen skal kunne administreres ved hjelp av et webgrensesnitt. Både faglærer og elever skal ha tilgang til dette grensesnittet med respektive rettigheter. OpenStack oppfyller allerede kravet om et webgrensesnitt med modulen Horizon. Det som spesifiseres av krav vil derfor også måtte tilfredstille kravene Horizon stiller til videre utvikling. Dette innebærer også konvensjoner som er lagt til grunn. Horizon viser til Python PEP8-stil [13] når det gjelder python, samt Django's Style Guide når det gjelder Templates [19]. Foruten om dette følger de CSS, XHTML og JavaScript-standarder som er oppgitt på OpenStack's Contributing Guide [48].

2.3 Krav til utvikling

De funksjonelle kravene vi skal gå inn på er kun ment å dekke utviklingsdelen av dette prosjektet. Oppsett, installering og implementasjon dekkes i egne kapitler, da dette ikke er emner som skal kravspesifiseres på samme måte som en utviklingsprosess.

2.3.1 Use Case

Figur 2: Use Case-diagram



2.3.2 Use Case-detaljer

Lag Batch er knyttet til Endre Batch med <extends> slik at kravene for parameter til Lag Batch blir reflektert i Endre Batch.
Slett Batch er knyttet til Slett instans med <uses> fordi nåværende funksjonalitet krever at instanser må slettes før man kan slette en tenant. Gruppen ønsker å fremheve dette i Use Case for å poengtere at dette ikke er en løsning på et problem, men en avhengighet som må tas i betraktning når løsning skal designes.

2.4 Høynivå use-case

Use Case ID	1	
Use Case navn	Lag Batch	
Hensikt	Opprette et sett prosjekter med et gitt antall instanser	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Legg til instanser i batchens konfigurasjon • Oppgi antall prosjekter og navngi batchen • Start opprettingen	• Kjør Use Case 3 • Kjører use Case 2 • Starter instansene • Opprett batchen i databasen
Alternativ arbeidsflyt:	Dersom instansene ikke starter opp, må de bygges på nytt. Eventuelt gis beskjed om at maksgrense for antall instanser er nådd	

Tabell 2: Use Case 1

Use Case ID	2	
Use Case navn	Last opp image	
Hensikt	Lar brukeren laste opp et egendefinert image som kan benyttes i OpenStack	
Hovedrolle	Student	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velg lokalt image som skal lastes opp	• Status på opplastningen • Vis opplastet image i liste over tilgjengelige image.
Alternativ arbeidsflyt:	Dersom opplastningen ikke er suksessfull får brukeren tilbakemelding om feilen med mulighet for å prøve på nytt.	

Tabell 3: Use Case 2

Use Case ID	3	
Use Case navn	Start instans basert på opplastet image	
Hensikt	La brukeren starte en instans basert på egendefinert image	
Hovedrolle	Student	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Ved vanlig prosedyre for å starte en instans vil brukeren velge sitt egendefinerte image	• Instans starter opp normalt basert på egendefinert image.
Alternativ arbeidsflyt:	Baserer seg på start av instans	

Tabell 4: Use Case 3

Use Case ID	4	
Use Case navn	Endre batch	
Hensikt	Legge til/slette prosjekter fra en batch	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velger batch som skal endres • Velger legg til/slett prosjekt	• Viser informasjon om batch • Prosjekt legges til / slettes fra batch • Registrerer endringen i databasen • Opprett batchen i databasen
Alternativ arbeidsflyt:	Bruker kan avbryte sekvensen	

Tabell 5: Use Case 4

Use Case ID	5	
Use Case navn	Slett Batch	
Hensikt	Slette en hel batch	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velg “Delete batch” på batch som skal slettes • Velger “Delete batch”	• Dialogboks hvor sletting må bekreftes • Sletter alle instanser i hvert prosjekt • Sletter brukeren • Slett prosjektene • Slett batchen fra databasen
Alternativ arbeidsflyt:	Bruker kan avbryte sletting i dialogboks	

Tabell 6: Use Case 5

Use Case ID	6	
Use Case navn	Slett instans fra tenant i batch	
Hensikt	Slette enkeltinstanser fra en tenant i en batch	
Hovedrolle	Student	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velger sitt prosjekt • Velger "Delete/terminate instance" på instans som skal slettes • Velger "Delete instance"	• Viser liste med informasjon om prosjekt • Dialogboks med bekreftelse • Instans slettes
Alternativ arbeidsflyt:	Bruker kan avbryte sletting i dialogboks	

Tabell 7: Use Case 6

Use Case ID	7	
Use Case navn	Lagre Batch-konfigurasjon	
Hensikt	Lagre et instansoppsett for senere bruk	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Setter opp en liste med instanser • Velger lagre oppsett	• Lagrer oppsettet i databasen
Alternativ arbeidsflyt:	n/a	

Tabell 8: Use Case 7

Use Case ID	8	
Use Case navn	Slett Batch-konfigurasjon	
Hensikt	Slette en konfigurasjon som alleredet er opprettet	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velger konfigurasjon som skal slettes fra liste • Bekrefter sletting	• Sletter oppsettet i databasen
Alternativ arbeidsflyt:	n/a	

Tabell 9: Use Case 8

Use Case ID	9	
Use Case navn	Lag batch basert på lagret konfigurasjon	
Hensikt	Lage en batch ut ifra tidligere lagret konfigurasjon	
Hovedrolle	Admin	
Arbeidsflyt	**Brukerens handlingsmønster**	**Systemrespons**
	• Velger "Batch configuration" • Velger en lagret batch • Velger "Create batch" • Eventuelt endrer konfigurasjonen • Oppgi antall prosjekter og navngi batchen • Start opprettingen	• Viser liste med informasjon om lagrede batch-konfigurasjoner • Viser informasjon om lagret batch • Vis listen med instanser som skal opprettes • Se Use Case 1
Alternativ arbeidsflyt:	n/a	

Tabell 10: Use Case 9

2.5 Ikkefunksjonelle krav

De ikkefunksjonelle kravene dekker områder som ikke omhandler utvikling.

2.5.1 Ytelse

Brukeropplevelsen skal stå i fokus. Å jobbe på en virtuell maskin skal ikke på noen måte oppleves tregt, og ytelsen skal være så nært en fysisk maskin med samme maskinvare som mulig. Ansvaret for å oppfylle disse kravene er primært lagt til bachelorgruppen SkyHiGh I/O rent fysisk, men systemet skal selv sørge for at øvre grenser for ressursbruk ikke overskrides. Disse kravene avhenger av maskinvaren som er tilgjengelig.

2.5.2 Pålitelighet

Systemet skal ha en oppetid på 99.9%. Dette tilsvarer 8 timer, 45 minutter og 36 sekunder nedetid per år, 43 minutter og 48 sekunder per måned, eller 1 minutt og 26 sekunder per dag. I denne oppetiden skal systemet være 100% stabilt, slik at brukeropplevelsen blir opprettholdt. Kravet om oppetid avhenger fullt og helt av kravene om ytelse og ressursbruk.

2.5.3 Tilgjengelighet

Systemet skal kun være tilgjengelig på HiGs interne nettverk, enten via kablet eller trådløs forbindelse. Nodene skal ha internettilgang, for å kunne motta oppdateringer. Lagringsnettet skal ikke være tilgjengelig utenfor controller-noden/compute-nodene, og følgelig isolert fra internett.

2.5.4 Sikkerhet

Sikkerheten skal ivaretas av mekanismer som allerede finnes i OpenStack, i form av brukerautentiseringsmodulen (keystone) og brannmuroppsettet (security groups) som

følger med rammeverket. Nettverkssikkerhet skal settes opp i form av aksesslister i systemets routere, og i pakkefiltreringstjeneste på hver node

Pakkeoppdateringer skal testes før de blir installert på produksjonssystemet for å unngå kompatibilitetsproblemer.

2.5.5 Lisens

OpenStack følger Apache 2.0-lisensen [23], som hovedsaklig sier at alle står fritt til å endre og distribuere materialet. Hvis noe av koden endres må opprinnelig kopirett følge med, opprinnelige bidragsytere krediteres og alt materiale gjøres tilgjengelig uten at opprinnelig bidragsytere kan holdes ansvarlig. All kode som produseres i dette prosjektet vil bli underlagt Apache 2.0-lisensen.

3 Teoretisk grunnlag

3.1 Hva er nettsky?

En nettsky (cloud computing) er levering av databehandling som en tjeneste, snarere enn et produkt. Delte ressurser, programvare og informasjon er gitt til datamasiner og andre enheter som et verktøy over nettverket, enten internt eller over internett. Grunnen til at det kalles en nettsky, er at tjenestens fysiske lokasjon er usynlig for brukeren, og også ofte for de som leverer nettskyen. Det skal ikke være mulig for tjenesteleverandøren å peke ut at bruker X's data ligger på server Y. Selve skyen er rent teknisk et større antall fysiske servere, som tilbyr en eller annen form for skytjeneste.

Det finnes i all hovedsak tre forskjellige måter å levere skytjenester på:

Infrastructure as a service (IaaS)
IaaS er å tilby tilgang til virtuelle maskiner i skyen. Eksempler på en slik tjeneste er Amazon EC2 og Rackspace Cloud Hosting. En Iaas løsning kan settes opp internt, så vel som globalt, og det er den interne varianten som er vesentlig for dette prosjektet.

Platfom as a Service (PaaS)
PaaS er for eksempel å tilby en utviklingsplatform for web-applikasjoner i skyen. På denne måten for man tilgang på "ubegrensede" ressurser, og får en meget skalerbar løsning. Eksempler på PaaS er Google App Engine, Windows Azure og Amazon Elastic Beanstalk.

Software as a service (SaaS)
SaaS vil si å tilby programvare i skyen, uten at brukeren trenger å behandle infrastrukturen eller plattformen programvaren kjører på. Eksempler er Google Docs, GMail og Picasa. Poenget er at brukeren får tilgang til disse tjenestene på internett, i stedet for å installere lignende programvare på egen maskin. [91].

En karakteristikk ved tjenesteleveranse fra en sky, er at man kun betaler for faktisk bruk (båndbredde, I/O-operasjoner) og at man enkelt kan skalere opp ytelsen og tilgjengelighet i perioder der man forventer mer pågang. Dette skal i følge leverandørene [6] gi store økonomiske gevinster. Utdanningsinstitusjoner kan forenkle lab-arbeid i forbindelse med IT-studier ved hjelp av en sky. I stedet for å bruke tid og penger på å implementere, og vedlikeholde en fysisk datalab, bruker man heller virtuelle maskiner i en intern IaaS løsning.

Slike løsninger gjør hverdagen mer effektiv, både for bedriften som helhet, de ansatte og også for brukerene [36]. En annen gevinst man får ved bruk av en nettsky er energisparing [84]. Ved å samle tjenester i virtuelle maskiner minker man antallet fysiske servere, som igjen fører til et lavere strømforbruk, og lavere kostnader.

Bedrifter og organisasjoner får utfordringer med lagring av personopplysninger i skyen, og hvordan ivareta sikkerheten rundt dette. Når slike data lagres i en sky hos en ekstern leverandør vet man ikke alltid hvilken fysisk lokasjon dataene ligger på og kan således ikke ha full kontroll på dataene. Et eksempel på dette er Narvik kommune, som ikke fikk lov av datatilsynet til å bruke Google Apps for å levere e-post, kalender, samskriving og den andre tilhørende tjenestene, fordi de mener det strider mot personvernloven [33] [32].
I Danmark er bruk av skytjenester i kommunal sektor blitt forbudt, på mye av det samme grunnlaget som saken i Narvik [83]. Sikkerheten i skyen er også et utbredt tema, og en stor utfordring. Problemstillingene er hovedsaklig todelt, utfordringer for leverandør, og utfordringer for bruker. For leverandøren er det blant annet viktig å konfigurerere *hypervisoren* skikkelig, slik at man ikke kan få tilgang på vertsoperativsystemet. En annen stor utfordring, er å hindre brukerene å misbruke tjenesten. I 2011 ble Sonys PlayStationNetwork offer for et massivt datainnbrudd, der angrepet ble initiert fra Amazons skytjeneste EC2 [4].

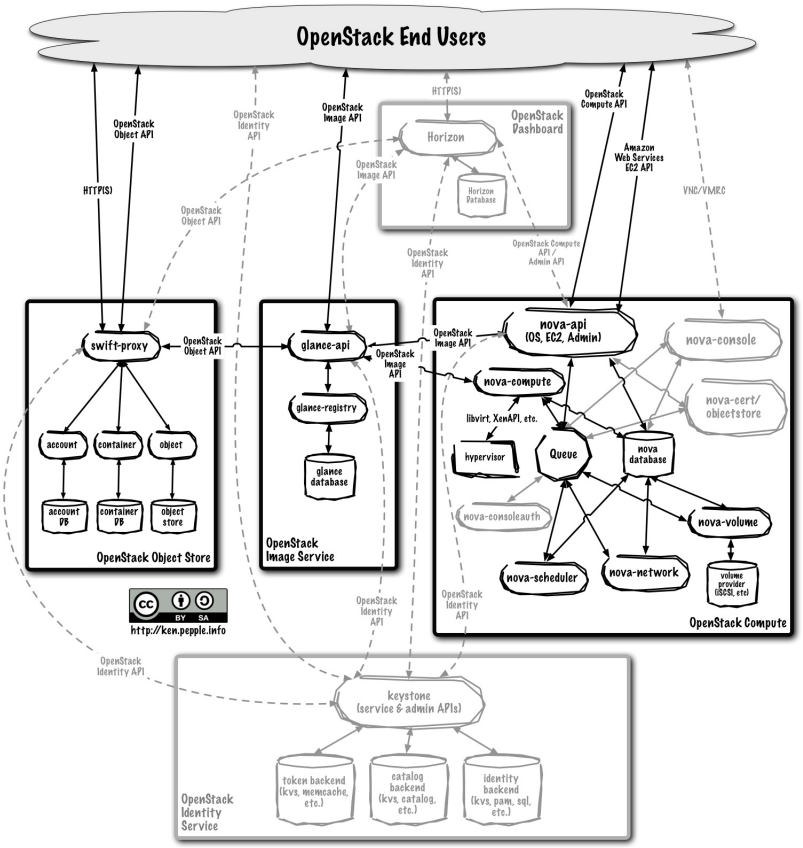
3.1.1 Virtualisering

Virtualisering er muligheten til å la en datamaskin kjøre flere virtuelle maskiner, som kan kjøre forskjellige operativsystemer. Teknologien er over 40 år gammel og ble først utviklet av IBM. De første utgavene besto stort sett av virtuelle maskiner som var eksakte kopier av maskinen de kjørte på. Teknologien ble i stor grad ignorert, helt til behovet på nytt meldte seg i moderne tid. Blant andre Sun og Hewlett Packard utviklet teknologien til å bli den virtualiseringen vi kjenner i dag [82].

3.2 OpenStack

OpenStack er et Open Source cloud-computing-prosjekt som tilbyr Infrastructure as a Service (IaaS), startet av Rackspace og Nasa i juli 2010. Per 17.april 2012 har over 150 bedrifter sluttet seg til prosjektet, som AMD, Intel, Canonical, SUSE Linux, Red Hat, Cisco, Dell, HP og IBM [49]. Første versjon, Austin, ble sluppet 21. oktober 2010, og har siden da vært i kontinuerlig utvikling. Nyeste versjon er Essex, sluppet 5.april 2012. Hovedmålet til OpenStack-prosjektet er å gi alle organisasjoner mulighet til å opprette og tilby nettskyer som kan kjøre på standard maskinvare.

Figur 3: OpenStack-moduler [65]



3.2.1 Moduler

Selve OpenStack er et modulbasert prosjekt, hvor hovedmodulene er Compute (Nova), Image Service (Glance) og Identity Service (Keystone). Disse modulene er det absolutte minimum av hva som må implementeres i et fungerende system. De øvrige modulene som er beskrevet under er valgfrie, og noen av dem er heller ikke en offisiell del av prosjektet enda. Siden det er modulbasert kan alle modulene samt tjenestene tilhørende de forskjellige modulene både dupliserers og distribueres til flere forskjellige fysiske servere. Dette gjør systemet meget skalerbart og enkelt å lastbalansere [61]. Hvordan modulene er knyttet sammen, ser vi i den logiske arkitekturen i figur 3.

RabbitMQ

RabbitMQ er ikke en del av OpenStack, men er en avhengighet i systemet. RabbitMQ er den tjenesten de forskjellige modulene bruker for å snakke med hverandre. Teknologien bak er AMQP [90]. Enkelt forklart består dette i at en tjeneste sender en “message” (forespørsel) som så blir tolket, og sendt til rett mottaker. OpenStack benytter kun asynkrone kall til RabbitMQ for å sikre at ingen tjenester blir hengde å vente på hverandre. Eksmempler på bruk her, vil f.eks være en forespørsel om å starte en instans.

Nova

Nova er kjernemodulen i OpenStack som tar seg av opprettelsen og administreringen av de virtuelle maskinene, samt håndtering av nettverk. Nova modulen er delt opp i forskjellig tjenester, som har sin spesifikke oppgave. Gjennom alle disse forskjellige tjenestene innfører nova en noen konsepter og begreper:

- *Instance:* En instance er ganske enkelt en virtuell maskin. Når en slik opprettes må man velge en *flavor*, et keypair og en eller flere security groups.
- *Flavor:* En flavor er et navngitt sett med maskinvarekonfigurasjon som kan velges for en instans, og inneholder antall virtuelle prosessorkjerner, mengde minne og størrelse på disk.
- *Security group:* Security group er et sett brannmurregler for innkommende trafikk som blir tilegnet den virtuelle maskinen. Standardgruppen benytter seg av "implicit deny all", det vil si all trafikk inn mot den virtuelle maskinen vil bli forkastet. Disse gruppene eksisterer per tenant, og er således ikke globale.
- *Keypair:* Keypairs er et *SSH* nøkkelpar bestående av en offentlig, og en privat nøkkel. Ved oppstart av en instans blir den valgte nøkkelen injisert, slik at brukeren får tilgang til den ved hjelp av sin nøkkel. Nøkkelparene fungerer kun for linuxbaserte instanser, da Windows ikke har pålogging via SSH.

Nova-api

Tilbyr et grensesnitt mot brukerene, som tar i mot kall til de forskjellige tjenestene. Det er støtte for OpenStacks eget *API*, Amazons EC2 API [5] samt et eget administrativt API forbeholdt systemadministratorer. Sistnevnte er realisert i form av klienten nova-manage. I tillegg til å ta i mot disse kallene, er det også denne tjenesten som initierer handlinger hos de andre tjenestene (f. eks å starte en instans). Nova-api står også for sjekk mot brukerens kvoter i prosjektet det jobbes på. Nova-api kjøres typisk på den maskinen man velger som controller.

Nova-network

Fungerer konseptuelt på akkurat samme måten som volume og compute, og håndterer følgelig alle forespørsler i køsystemet som har med nettverk å gjøre. Dette omfatter opprettelse av *nettverksbroer*, endring i brannmur, tildeling av nettverk, *DHCP*-oppsett og tildeling av floating-IP's. Nova-network er dypere forklart i seksjon 5.2.1.

Nova-scheduler

Denne tjenesten tolker alle meldinger i køsystemet, og sender dem til den noden forespørselen gjelder. F.eks en forespørsel om å terminere en virtuell maskin skal sendes til en nova-compute node. I nåværende utgave av OpenStack kjører nova-scheduler i noe man kaller "muliti"-mode, som gjør at man kan velge en separat algoritme for compute og en for volume. Det tilbys fire forskjellige algoritmer:

- Chance
- Cost & Weight
- Filter

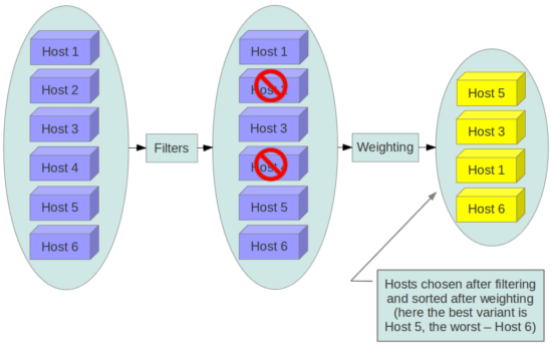
- Simple

Chance velger en tilfeldig node på tvers av alle tilgjengelighetssoner.

Simple velger den noden som på det tidspunktet meldingen blir tolket har minst last, det vil si den noden som har kjører minst antall instanser og/eller minst antall volumer.

Filter sorterer ut hvilke noder som er aktuelle å starte opp en instans på, ved å legge på ulike filtre. Standardfiltrene som følger med, gir blant annet mulighet til å vurdere mengde ledig minne, ledige prosessorkjerner, vurdere ressurser i forhold til ønsket flavor, eliminere noder direkte og å tvinge oppstart på en gitt node. I tillegg kan man kode sine egne filtre, dersom behovet skulle dukke opp. Etter at denne filtreringen er gjort, sitter man igjen med et sett noder som kvalifiserer til å kjøre instansen som er blitt forespurt. Av disse blir det gjort en vurdering med *Cost & Weight* algoritmen, for å sortere de aktuelle nodene ut i fra en vekting. Vektingen gir nodene en poengsum (cost), og den noden som endre opp med lavest cost blir valgt. Hvilken verdi hver enkelt vektingsregel har, er konfigurerbart, slik at man kan skreddersy hva som er viktigst når en node skal vurderes. Figur 4 viser arbeidsflyten.

Figur 4: Filtering workflow [58]



Denne algoritmen ble introdusert med Essex, og er derfor veldig tidlig i utviklingen. I skrivende stund (mai 2012) finnes det bare én réell vektingsregel, denne vekter mengden ledig minne.

Filter er standard for nova-compute, og filterne for ledig minne, ledige prosessorkjerner, og vurdering av ressurser opp mot flavor er aktivert som standard. For nova-volume er chance standardvalget.

Nova-compute

Denne tjenesten står for oppretting og terminering av virtuelle maskiner. For å realisere dette, snakker nova-compute med de forskjellige API'ene for de hypervisorene som er støttet. Eksempelvis libvirt for KVM/qemu [38] [10] og XenAPI [26] for XenServer/XCP [27]. Når nova-api har mottatt en forespørsel om en ny virtuell maskin, blir den plassert i køsystemet, hentet ned av nova-scheduler, og sendt til en nova-compute node. Deretter

blir det kjørt en rekke systemkommandoer for å gjennomføre forespørselen som kom gjennom API'et. En typisk serie av systemkommandoer ved oppstart av en ny virtuell maskin vil være:

- Lage konfigurasjonsfil for virtualiseringsplatformen man har valgt. F.eks lage en XML-fil for libvirt
- Sette opp brannmurregler i iptables
- Opprette en katalog der blant annen disk-filen til den nye virtuelle maskinen blir lagret. Disse ligger under /var/lib/nova/instances/instance-XXXXXXX
- Kopiere over image-fil med operativsystem fra glance til den compute-noden som skal kjøre den akutelle virtuelle makinen. Dette gjøres kun dersom det valget imaget aldri har blir kjørt fra compute-noden. Imagene blir lagret i katalogen `/var/lib/nova/_base`.
- Kopiere over image-filen til den nye instansens katalog. Dersom image-filen allerede finnes på compute-noden blir den ikke hentet fra glance først.
- Kjører kommandoer for å starte instansen. Hvilke kommandoer som blir kjørt avhenger her av hvilken hypervisor som er valgt.

Arbeidsflyten er basert på loggfilene.

Underveis i denne prosessen vil det hele tiden rapporteres til databasen som tilstanden på den nye instansen.

Nova-volume

Håndterer opprettelse, sletting og tilegning av fysisk lagringsplass til instanser. Man har mulighet til å legge til ekstra lagringsplass på instansene i etterkant av opprettelsen, slik at man enkelt kan løse utfordringer i forbindelse med dette. Volumene blir typisk hentet fra *iSCSI*. Nova Volume tilsvarer Amazons Elastic Block Storage [8]. Tjenesten fungerer på akkurat samme måte som compute. Det blir sendt forespørsler til køsystemet, som nova-volume henter ned, og prosesserer.

Glance

De virtuelle disk-imagene blir administrert via Glance, og støtter disk-formater som Raw, Machine (kernel/ramdisk som ikke ligger i selve disk-image, a.k.a. AMI), *VHD* (Hyper-V), *VDI* (VirtualBox), *qcow2* (Qemu/KVM), *VMDK* (VMWare) og *OVF* (VMWare, andre). Disse disk-imagene kan lagres på flere forskjellige måter. Man kan bruke lokalt filsystem (/var/lib/glance/images), Swift, direkte hos Amazon S3 [7] , eller *HTTP*. Opplastingen og registreringen foregår enten gjennom Glance API eller EC2-api'et. Glance består av to tjenester, i tillegg til en database:

- glance-api
- glance-registry

Glance-api tar i mot alle API-kall som omhandler registrering av nye image, overføring av image til compute-noder og lagring av image. Glance-registry lagrer og prosesserer metadata angående nye image.

Keystone

Keystone er autentiserings- og identitetstjenesten i OpenStack. Alle modulene i OpenStack benytter seg av denne som standard. Keystone er i all hovedsak en database som inneholder informasjon om brukere, prosjekter, roller, tokens, tjenester, endpoints for tjenestene. Hvilket system som brukes for den databasen kan man velge selv, men det vanligste er en form for SQL-database. Det er også støtte for *LDAP* og *KVS*. Ved hjelp av disse tilbys det autentisering på flere forskjellige måter. En bruker kan autentiseres ved:

- Brukernavn og passord
- Brukernavn og API-nøkkel

Autentiseringen foregår ved standard challenge-response uten *SSL*. Ved en gyldig autentisering blir det satt informasjon om prosjekt, bruker og roller i HTTP-headerene. Brukeren blir tildelt en token, som er bundet til et privilegier. Det foregår ingen identifisering ved innlogging, og det er heller ingen innebygde mekanismer for dette.

Keystone introduserer noen konsepter i forbindelse med sine tjenester:

User

En bruker er som regel en representasjon av en person, som skal ha tilgang til de forskjellige tjenestene. Men, en bruker kan også være en tjeneste i seg selv.

Token

Konseptet med tokens er ganske enkelt. En token er tilfeldig generert tekststreng som brukes for å gi brukere tilgang til tjenester. En token er gitt et sett privilegier (scope), der man f.eks kan få administrative rettigheter.

Tenant

Er i all hovedsak det keystone kaller et prosjekt. Alle instanser, nettverk, volumer, IP-adresser, kvoter og sikkerhetsregler er knyttet til en tenant.

Service

Alle tjenestene i OpenStack må registreres i Keystone. Hver tjeneste har en bruker knyttet til seg, og hver tjeneste tilbyr et eller flere endpoint som brukerne kan aksessere tjenester gjennom.

Endpoint

Et endpoint er informasjon om hvor en service kan aksesseres. I praksis er det en URL til tjenestens API.

Role

Alle brukere knyttes til et prosjekt med en bestemt rolle. Til hver rolle er det tilknyttet et sett med privilegier og rettigheter. I Keystone så er listen over roller en bruker har i de forskjellige prosjektene knyttet til den token man får ved en vellykket autentisering.

Horizon

Horizon er webgrensesnittet i OpenStack. Gjennom horizon får administratorer og brukere tilgang til den mest grunnleggende tjenestene. Brukere har adgang til å:

- Administrere instanser innenfor de prosjektene de er medlem i (opprette, slette, restarte)
- Ta snapshots av kjørende instanser, samt administrere disse (redigere, slette)
- Allokere såkalte floating IP adresser (se nettverkskapittelet)
- Administrere volumer innenfor de prosjektene de er medlem i (opprette, slette, knytte til instans, fjerne fra instans)
- Administrere sikkerhetsgrupper, det vil si grupper av brannmurregler for instansene i prosjektet
- Administrere nøkkelpar for tilgang til instanser.

En administrator vil i tillegg til nevnte rettigheter, ha mulighet til å:

- Monitorere samlet ressursbruk for alle prosjekter fra tidenes morgen
- Håndtere alle kjørende instanser i hele implementasjonen.
- Få en oversikt over registrerte tjenester
- Definere flavors, det vil si forhåndsdefinerte maskinvarekonfigurasjoner som brukerne kan velge ved opprettelse av instanser.
- Administrere alle disk-images som er registrert
- Administrere prosjekter (opprette, endre, slette, modifisere tilknyttede brukere)
- Administrere brukere (opprette, endre, slette)
- Sette kvoter per prosjekt. En kvote inneholder maksimum antall CPU kjerner, instanser, MB RAM, floating IP'er, volumer (og samlet størrelse på disse) og filer man kan injisere i instanser (og maks størrelse på disse).

Horizon kjører via webserveren Apache via mod_wsgi, og krever kun en database i tillegg. All informasjon blir hentet fra andre tjenester via deres API'er, slik at selve webapllikasjonen lagrer svært lite data i seg selv.

Swift

Swift er OpenStacks modul for det man kaller *object storage* i sky-verden. Object Storage er annen måte å tenke lagring på, i forhold til tradisjonelle filsystemer. Funksjonaliteten er mye den samme, men man vil ikke kunne montere Object Storage på samme måte som man kan fra f.eks et *SAN* eller et *NAS*. Swift kan sammenlignes direkte med Amazons S3. Object Storage baserer seg på følgende konsepter:

Accounts og Account servere

Brukere av lagringssystemet må identifiseres og autentiseres (f.eks via keystone) for å få tilgang til sine filer. Hver konto er tilknyttet en bruker, som igjen har sine *containers* tilknyttet seg.

Containers og objects

Containers er nærmest ekvivalente til mapper i tradisjonelle filsystemer. Forskjellen er at containers ikke kan nøstes på samme måte. Objects er på samme måte ekvivalent til filer. Et object kan ha en mengde metadata knyttet til seg i form *key-value* par. Dette er ment for å beskrive objekter best mulig.

Quantum

Quantum er tjeneste for virtuelle *L2* nettverk. Den tilbyr et API for nettverkstrafikk mellom enheter fra de forskjellige OpenStack tjenestene, hovedsaklig i form av virtuelle switcher. For å realisere dette, bruker man en plugin (f.eks Open vSwitch eller Cisco Quantum plugin), slik at Quantum API blir et abstraksjonslag på samme måte som Nova. De virtuelle switchene vil ha samme muligheter for konfigurasjon som fysiske switcher, f.eks QoS, port security, monitorering osv. Det mest naturlige er å knytte Quantum sammen med Nova, for å skape interne logiske nett for prosjekter. Quantum er også natulig å kombinere med Melange. I skrivende stund mangler Quantum muligheten for å lage identiske interne nett, slik at muligheten til å ha identiske prosjekter, med identiske interne nett (som ikke er det samme logisk nettet) ikke er til stede [68].

Melange

Melange er en tjeneste med fokus på *IPAM* og *DHCP*, for å gi en mer fleksibel løsning for *L3* addressering av virtuelle maskiner enn den som er innebygget i Nova. I fremtiden er det ventet at den også vil håndtere DNS, ruting og gateway mot internett, og i så måte ta over mye av de standard nettverksløsningene som finnes i nova i nåværende utgave.
Melange-modulen er meget ny, og ble først innlemmet som en offisiell modul i Essex-utgaven. Det foreligger mange dokumenter på Melange sin fremtid, både som uavhengig prosjekt, som OpenStack-modul, og hvordan den skal implementeres med nova. Blant annet er det planlagt at den skal slås sammen med Quantum [67] [31] [85].

3.3 Open Source

OpenStack bruker en åpen utviklingsmodell, og all OpenStack-kode er fritt tilgjengelig under Apache 2.0-lisensen [23]. Hver sjette måned holdes et åpent utviklingsmøte ("Design summit") for å samle krav og spesifikasjoner til neste versjon.

Selve utgivelsessyklusen består av fire faser:

- Planlegging: Utviklingsmøter og framlegging av Launchpad Blueprints (Løse kravspesifikasjoner)
- Implementasjon: Mange milepæler med koderevisjon
- QA: Testing, prioritering av bugs og dokumentasjon. Kun "branches" som fikser bugs og ikke kommer med nye funksjoner blir lagt til
- Release: Release Candidate Freeze to dager før Release Day

Under utviklingssyklusen brukes alfabetiske navn (Austin, Bexar...Folsom) etter byer eller fylker i nærheten av forrige utviklingsmøte. Endelig utgivelsene blir navngitt med "år.nr", og første release i 2012 er da 2012.1.

Retningslinjene for OpenStack-fellesskapet kan oppsummeres slik: *"Each project community should be self-managing by the contributors, and all disputes should be resolved through active debate and discussion by the community itself"* [47]. Ved uenighet kan Project Policy Board tre inn, som er et styre bestående av valgte og utnevnte medlemmer. Det kjøres halvårlige valg for en Project Technical Lead for hvert prosjekt hver sjette måned.

For å bidra med kode må man signere Individual Contributor License Agreement. Bidrar man på vegne av et firma burde en autorisert representant fra bedriften signere Corporate Contributor License Agreement [52].

Launchpad er kjernen hvor feilrapportering, blueprints, spørsmål og kode finner sted [57]. En Launchpad-konto er nødvendig for å delta.

Kanaler for å komme i kontakt med resten av bidragsyterene er hovedsaklig mailingliste, IRC og forum [63], samt dokumentasjon [66] og wiki [69].

3.3.1 MVC

Model view controller (MVC) er et design-pattern som er designet for å skille data, logikk og presentasjon i et grensesnitt. Ett typisk *MVC* vil ha en modul (*View*) som rendrer et *GUI* for brukeren, en modul (*Controller*) som bestemmer hva som skal rendres og med hvilke data, i tillegg til en modul (*Model*) som fungerer som et abstraksjonslag mot datalagringen.
Når en bruker trykker på en link i en nettleser, er det controlleren sitt ansvar å behandle forespørselen, og sende den videre til riktig view. Når viewet mottar forespørselen spør den modellen etter riktig data, og viser resultatet frem for brukeren.

3.3.2 Django

Django er ett web-rammeverk for Python, med sterk fokus på effektivitet. Det er opprinnelig designet for publisering i avisredaksjoner men har i senere tid gått over til å bli et rammeverk for generell webutvikling. Styrken i Django ligger i hvordan de har implementert MVC. Måten å skille logikk, data, og presentasjon sørger for et kjapt utviklingsmiljø som fører med seg store muligheter for utvidelser. Et viktig punkt i forbindelse med måten de har implementert MVC er at de bruker et template-system som gjør at et view ikke nødvendigvis er hvordan data skal vises, men hvilken data som blir vist. Template-systemet opptar altså rollen for hvordan data skal vises.

Strukturen for ett django-prosjekt ser slik ut:

```
Mitt prosjekt
    manage.py
    minside
        settings.py // Registrere din applikasjon
        urls.py     // Her vil du sette hvilke URL'er som skal
                    // peke til hvilke view i din applikasjon.
        wsgi.py     // wsgi som i mod_wsgi er en apache modul for å hoste
                    // django-apps med apache som webserver.
```

Django-prosjektet håndterer utvidelser (kalt *applikasjoner*) som individiuelle moduler uavhengig av hverandre. Eksempelvis hvis en bruker har laget en applikasjon i sitt prosjekt for å vise en kalender skal denne kunne benyttes i hvilket som helst annet prosjekt.

Med dette utgangspunktet registrerer man applikasjoner som med følgende struktur:

```
minapplikasjon
    __init__.py
    models.py
    views.py
    tests.py
    templates
        minapplikasjon
```

3.3.3 Horizon (oppbygging)

Horizon er skrevet i python [22] med Django som web-rammeverk. Dette innebærer i hovedsak at Djangos implementasjon av MVC dikterer hvordan Horizon opererer.

Horizon baserer seg i utgangspunktet på følgende tre applikasjoner: Syspanel, Dash og Settings. Disse er kalt dashboards og består igjen av panels som er spesifikt for Horizon. Panels har sine egne views og controller. Dette er ett eksempel på Django's styrke, altså at mvc er gjenbrukbart gjennom hele systemet ved bruk av applikasjoner. Eksempelvis er Django-prosjektet et mvc, dashboard er ett mvc og panel er ett mvc. En horizon-applikasjon er strukturert på følgende måte:

```
syspanel
    dashboad.py
    templates
    panel
        panel.py
        urls.py
        views.py
        forms.py
        templates
```

3.3.4 Model

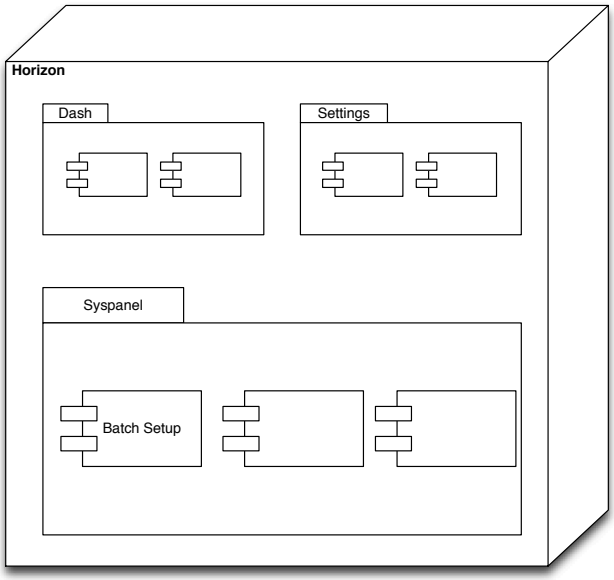
Horizon benytter seg ikke av *model* på den tradisjonelle Django-måten. Dette fordi Horizon kobler seg direkte til databasene fra de andre OpenStack-modulene. De bruker SQLAlchemy [76] som abstraksjon mot databaselaget, slik at Django behandler datasettene på samme måte, bare ikke ved hjelp django-modeller.

4 Design

Modulene som skal utvikles basert på dette dokumentet har som formål å utvide funksjonaliteten som allerede eksisterer i Horizon-prosjektet. Modulene skal være fullverdige komponenter i det eksisterende systemet, og svare til de krav som stilles av Horizon.
Kravene som er beskrevet i kravspesifikasjonen vil bli implementert som moduler i Horizon. Funksjonaliteten vil befinne seg i det eksisterende dashboardet *Syspanel*, og i menyen være navngitt *Batch Setup*. Herunder vil brukeren ha mulighet å se en oversikt over de eksisterende batcher i en tabell. Tabellen vil inneholde funksjonalitet for manipulering av de eksisterende batcher. Her vil brukeren kunne opprette en ny konfigurasjon av instanser som brukeren kan opprette en ny batch ut i fra. Konfigurasjonen vil også kunne lagres og da gjøre det mulig å opprette batcher ut fra en lagret konfigurasjon. Manipulering av konfigurasjon vil kunne utføres fra en lignende tabell som oversikt over batches.

4.1 Arkitektur

Figur 5: Domenemodell uml



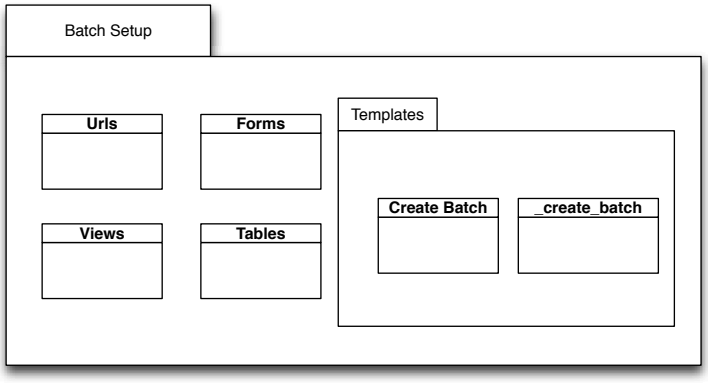
I figur 5 ser man Batch Setup i et overordnet perspektiv som en av mange moduler i Horizon. Dash, Settings og Syspanel er tre likestilte *Dashboards* som hver for seg inneholder *panels*.

Dash er tilgjengelig for alle brukere som kan autentisere seg og er grensesnittet som lar den generelle bruker behandle sine egne tenants og instanser.
Settings er også tilgjengelig for alle brukere, og lar brukeren velge visningsspråk samt

laste ned autentisering-filer for brukerens tenants.
Syspanel er et overordnet grensesnitt som kun er tilgjengelig for brukere med Admin-rettigheter. Syspanel lar administrator ta seg av vanlige administrative oppgaver som f.eks sette kvoter på instanser i en enkelt tenant.
Batch Setup er hovedmodulen som skal utvikles av SkyHiGh og faller naturlig under syspanel da dette er en oppgave for administrator alene, og gruppen så ingen grunn til å vurdere dette videre.

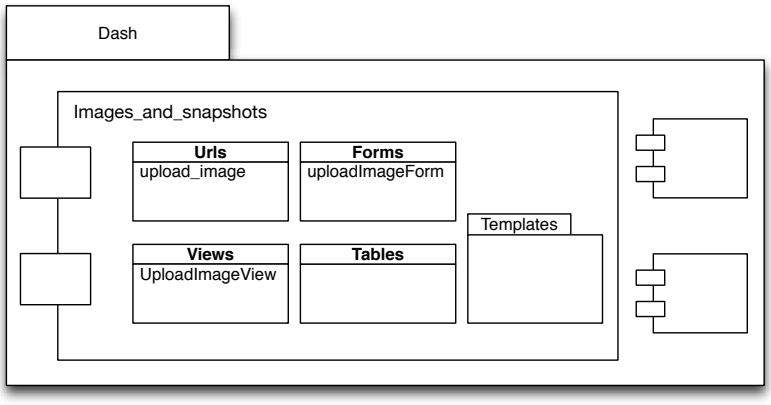
4.2 Dekomposisjon

Figur 6: Dekomposisjon batch setup



I figur 6 ser man dekomposisjonen som ligger til grunn for alle paneler. I følge kravspesifikasjonen skal konvensjoner innført av Horizon ivaretas, derfor ble samme strukturen også lagt til grunn for Batch Setup. Funksjonaliteten til hver enkelt pakke er beskrevet i klassediagrammet som blir beskrevet i 4.3.1.

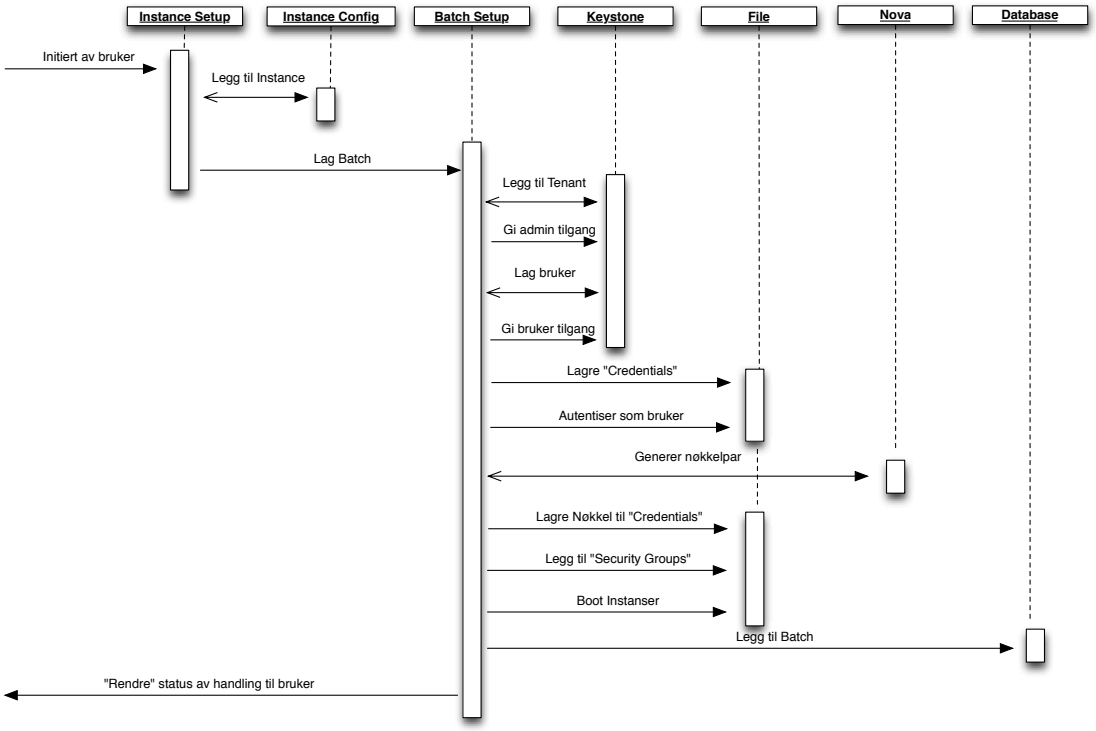
Figur 7: Dekomposisjon dash



I dekomposisjonen i figur 7 ser man hvordan kravet for Opplasting av image i use case to vil forholde seg til modulen Dash. Denne vil integreres i det eksisterende images and snapshots og vises som ett valg under dette panelet. Siden dette ikke er en egen modul

vil det ikke bli konstruert klassediagram eller sekvensdiagram for de to use casene som har ført til denne dekomposisjone. Gruppen konkluderte med at dette ikke er så komplekst at det krever det. Usecase 3 vil inngå i funksjonaliteten som eksisterer for å starte vanlige images, derfor anser vi dette som løst og vil ikke designe dette.

Figur 8: Sekvensdiagrammet for Use Case 1, Lag Batch



Figur 8 beskriver arbeidsflyten når man oppretter en batch med instanser i. Det første brukeren vil gjøre, er å legge til én og én instans i oppsettet. Når listen med instanser er ferdig, vil brukeren velge navn på sin batch, og hvor mange tenants han vil ha i batchen. Det første som da skjer er at tenant blir opprettet. Navnet på hver enkelt tenant blir automatisk generert ut i fra navnet brukeren gav til batchen. Dersom batchen heter “sysadmin” vil tenants i denne batchen hete “sysadmin1, sysdamin2 .. sysadminN”.
Deretter blir systemets administrator lagt til med rollen “member” i nyopprettet tenant. Dette må gjøres for at administratoren senere skal kunne gå inn i hver tenant å gjøre endringer. En bruker til hver tenant blir også opprettet. Brukernavnet blir automatisk generert ut i fra navn på tentant. Dersom tenant heter “sysadmin1” vil brukeren hete “admin_sysadmin1”. Passordet er en tilfeldig streng på 10 tegn. Brukeren vil også bli gitt rollen “member” til sin respektive tenant. Alle disse interaksjonene skjer med identitetssystemet keystone.

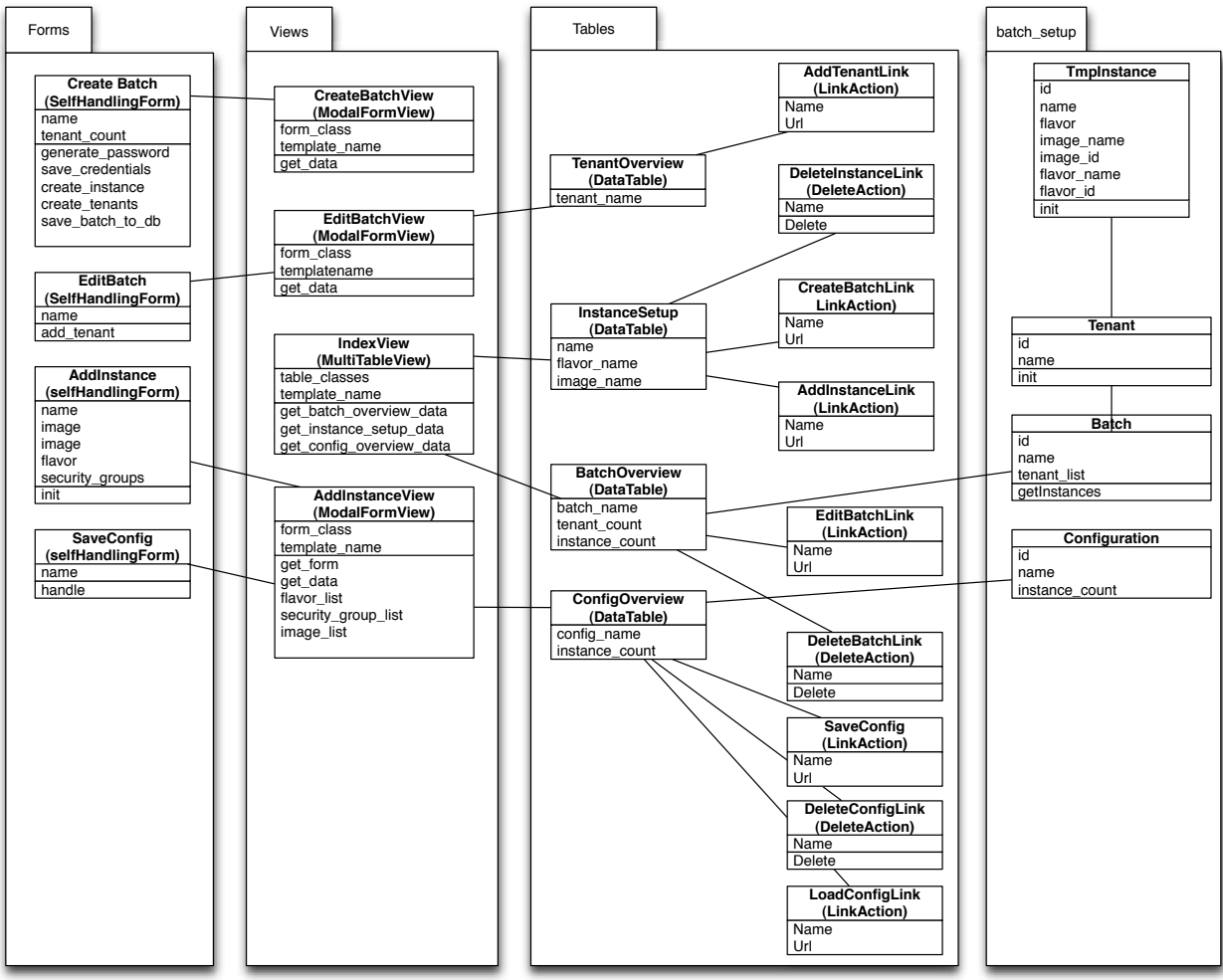
Siden OpenStack er modularisert i form av tenants og brukere med forskjellige roller, er brukeren hele tiden autentisert i en tenant i inneværende sesjon. Dette byr på utfordringer fordi alle interaksjonene mot nova er tenant-spesifikke, f.eks å opprette instanser.

Måten det er løst på, er å la python-koden kjøre et bash-script. Dette scriptet kan forandre hvilken tenant det er autentisert i ved hjelp av miljøvariabler. Derfor blir en fil, her kalt credentials, med de nødvendige miljøvariablene generert ut ifra opplysingene om tenant-navn, brukernavn og passord. Samtidig blir også brukernavnet og passordet lagret i en tekstfil, for at man skal kunne dele ut disse til grupper når operasjonen er gjennomført. Når man nå er autentisert i en ny tenant via bash-scriptet, blir det generert et SSH-nøkkelpar som både knyttes til brukeren i keystone, og også brukes for tilgang til instansene. Nøkkel blir i tillegg lagret i en tekstfil i samme katalog som brukernavn, passord og credentials. I security-groups blir det åpnet for SSH og *ICMP*-trafikk. Når alle de forestående oppgavene er fullført, blir instansene startet én og én. Deretter avslutter bash-scriptet, batch_setup registrerer batchen, og tenant som er knyttet til den i sin database. Til slutt blir det sendt en statusmelding til brukeren som viser om operasjonen var vellykket eller ikke.

4.3 Datadesign

4.3.1 Klassediagram

Figur 9: Klassediagrammet for modulen Batch Setup



Arkitekturen til ett panel

I urls vil man ved bruk av regular expressions [92] definere url-ene som skal benyttes for de forskjellige views. Slik at en url for Batch Setup vil allokeres ved hjelp av denne setningen:

```
('horizon.dashboards.syspanel.batch_setup.views',url(r'^$',
IndexView.as_view(), name='index')
```

Denne kodelinjen setter url-en som blir benyttet av brukeren og deretter kaller korrekt view. Viewet som blir kalt, *IndexView*, krever to lokale variable, som er *table_classes* og t*emplate_name*. Generelt for alle views forventer IndexView en template, *template_name*, som er definert under templates i det aktuelle dashboardet. Det er denne templaten som definerer hvordan data skal presenteres i HTML. IndexView arver etter et *MultitableView*

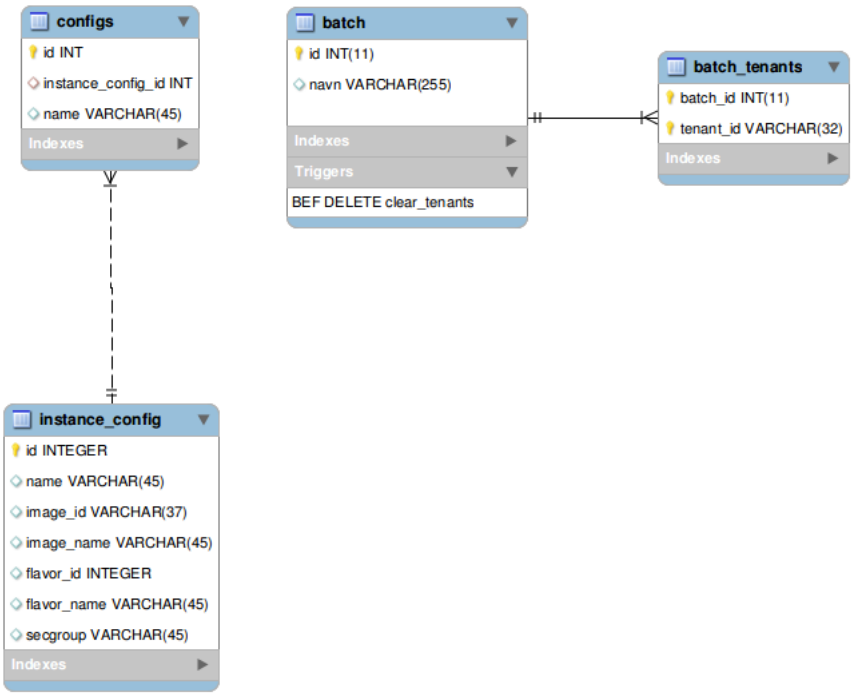
som forventer en liste av godkjente tabellobjekter. Det er da table_classes som inneholder denne listen av objekter. Nå vet altså templaten hvilke tabeller den skal bruke som mal for å vise data, men ikke hvordan den aktuelle data skal vises. Dette blir gjort ved hjelp av metoder i IndexView-klassen som må reflekteres i den aktuelle templaten, i dette tilfellet index-template. Disse metodene tilsvarer da tabellene som er lagt i table_classes, og returnerer data til tabellene som er deklarert i tables.py.
Objektene som blir returnert til table_classes krever i utgangspunktet bare en id, og lager da kolonner ettersom variabler blir deklarert av typen *tables.coloumn*. En indre klasse *Meta* tar seg av det obligatoriske for tabellen, slik som tabellnavn (syntaktisk) og visningsnavn. I meta registrerer man også objekter av typen *Actions*, disse vil da ta seg av handlinger som slette, linke og legge til elementer i tabellen.

Hvis man tar for seg ett view som *CreateBatchView* tar dette en *form_class*, på lik linje med at IndexView forventet en varibel table_classes. Denne inneholder lokale variable i forhold til input-felter som skal inkluderes samt en handle-metode sender brukeren til rett sted etter skjema er sendt. Mor-klassen *SelfHandlingForm* validerer input fra form og returnerer disse til handle-metoden som behandler disse.

4.3.2 Abstraksjon mot database

Horizon har opprinnelig lave koblinger mellom objekter. En tenant vet ikke hvilke instanser som er tilknyttet den. Det er kun instansen som vet hvilken tenant den hører til.

Figur 10: ER-modell batch db



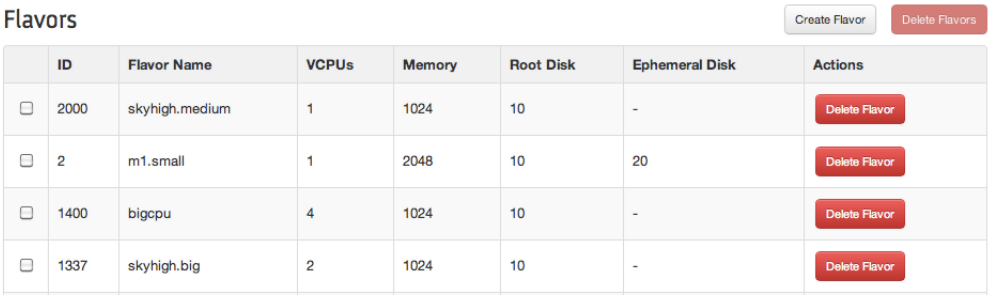
Vi har valgt å benytte nye tabeller i dashboard-databasen for å gjøre data tilgjengelig med samme konvensjon som de resterende tabellene. Vi skal skrive vårt eget abstraksjonslag mot databasen slik at vi ikke trenger å endre funksjonalitet i hvordan Horizon instansierer objekter. Grunnet manglende dokumentasjon, vet vi ikke hva endringer på det nivået kan affektere. Vi konkluderte med at vi ikke ville bruke tid på grundigere undersøkelser av API'et enn det som var nødvendig for prosjektet.

Triggere

Det skal benyttes *triggere* i SQL-databasen for å slette relasjoner mellom tenants og batcher, og også for å slette instanskonfigurasjoner fra en konfigurasjon. Når en rad slettes fra batch, vil alle relasjoner til denne raden slettes fra *batch_tenants*. Det samme vil også skje i *configs* og *instance_config*.

Skjermbilde

Figur 11: Horizon Table



Skjermbildet i figur 11 viser hvordan konvensjonen for å behandle dynamisk innhold i Horizon forholder seg. Slik vil oversikt over en batch, samt konfigurering av en tenant også se ut.

Løsning av use case

Use Case	Løsning
1	Batch Setup
2	Images And Snapshots
3	Images And Snapshots
4	Batch Setup
5	Batch Setup
6	Instances And Volumes
7	Batch Setup
8	Batch Setup
9	Batch Setup

Tabell 11: Tabell for krav og løsning

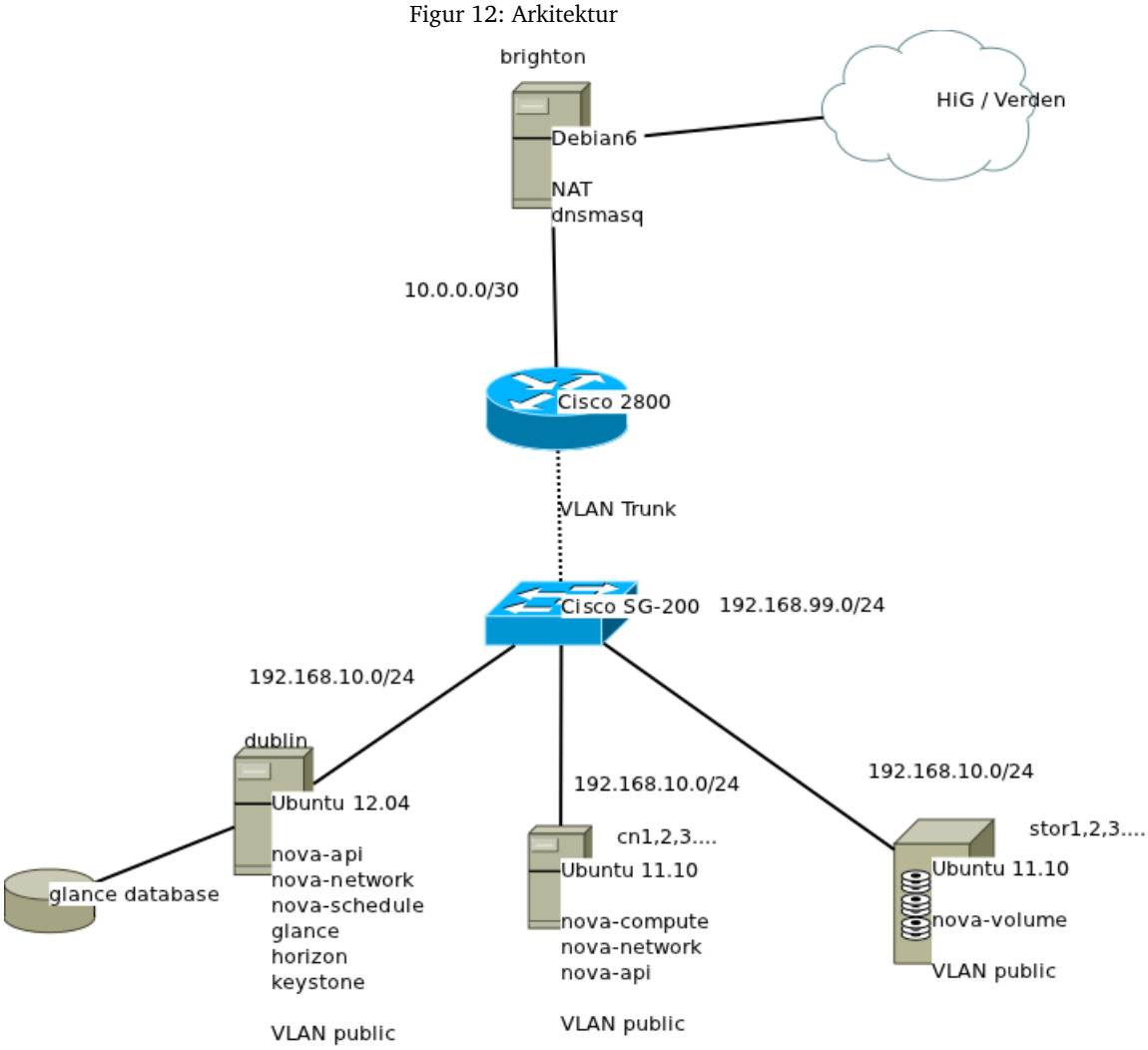
I tabell 11 ser man hvilke moduler i Horizon use casene vil bli løst i. Ut fra klassediagrammet (figur 9) ser man hvordan use casene blir løst i modulen Batch Setup. Dekomposisjonen for Use Case 2 og 3 (figur 6) beskriver implementasjonen av kravene til Dashboardet Images And Snapshots. Use Case 6 blir løst av funksjonaliteten i Dashboardet Instances And Volumes.

5 Gjennomføring

5.1 Implementasjon

5.1.1 Nettverksimplementasjon

Figur 12: Arkitektur



Rutere

Enhet, modell	Nettverkskort	MAC-adresse	IP/MASK	Gateway
brighton, HP ProLiant DL320	eth0	00:1f:29:13:02:8a	DHCP	
	eth1	00:1f:29:13:02:8b	10.0.0.1/30	10.0.0.2
SkyRoute, Cisco 2800	fa0/0	ec:44:76:68:16:94	10.0.0.2/30	
	fa0/1.10		192.168.10.1/24	
	fa0/1.30		192.168.30.1/24	

Tabell 12: Rutere

Switch

Enhet, modell, Management IP	Port	Hastighet	VLAN(s)	Trunk status
switchd13d24, Cisco SG-200 26, 192.168.99.254/24	GE1-12	1000Mbit	10 (public)	av
	GE13-18	1000Mbit	100-115 (VLAN for *VM*'er)	på
	GE19-24	1000Mbit	30 (storage)	av
	GE25	1000Mbit	99 (native/management)	av
	GE26	1000Mbit	10,30,99	på

Tabell 13: Switch

Servere

Navn	CPU	RAM	Disk	IP
dublin	Intel Xeon 3060 2.4 GHz	4GiB	2 x 250 GB	eth0: 192.168.10.2, eth1: oppe - ingen IP, eth3: 192.168.30.2
manchester	Intel Xeon 3060 2.4 GHz	4GiB	2 x 250 GB	eth0: 192.168.10.10, eth1:, oppe - ingen IP
newcastle	Intel Xeon 3060 2.4 GHz	2GiB	2 x 250 GB	eth0: 192.168.10.12, eth1: oppe - ingen IP
cardiff	Intel Xeon 3060 2.4 GHz	2GiB	2 x 250 GB	eth0: 192.168.10.11, eth1: oppe - ingen IP
kingston	Intel Xeon 3060 2.4 GHz	4GiB	12 x 250 GB	bond0: 192.168.30.10, eth0: oppe - link aggregation, eth1: oppe - link aggregation

Tabell 14: Servere

I prosjektperioden vil nettverkstopologien være som vist i figur 12 . Satt i produksjon vil man antageligvis fjerne ruteren og DHCP-serveren vi ser øverst i figuren, slik at man benytter seg av offentlige IP'er (globale IP'er delt ut fra HiG). For testformål har vi satt opp vårt eget lokale nettverk bak en dedikert linuxmaskin som kjører *NAT* og DHCP, slik at vi beskytter testoppsettet fra offentligheten, og unngår å forstyrre noe av HiGs infrastruktur.

5.1.2 VLAN

På switchen er det satt opp ulike *VLAN*:

- Standard management-VLAN er flyttet vekk fra VLAN 1, til VLAN 99, i følge beste praksis [3].
- VLAN 10 for internettilgang (public)
- Et dedikert VLAN til lagringsnodene (VLAN 30, storage)
- Åtte trunk-porter for trafikk mellom instansene, som aksepterer VLAN 100-115

De samme VLAN'ene er satt opp med virtuelle grensesnitt på ruteren for å kunne rute trafikk mellom disse [2]. Selv om det her er satt opp at lagringsnoden er knyttet til et eget beskyttet VLAN, var ikke det tilfelle under prosjektperioden. Dette på grunn av at vår ruter kun har 100 Mbps porter. Hvis lagringsnoden hadde stått på eget VLAN, ville all trafikk måtte gå via ruteren, og gevinsten ved en 1 Gbps switch ville forsvunnet. Derfor sto også lagringsnoden koblet i samme VLAN som resten av nodene. I produksjon anbefales det å bygge infrastruktren slik figuren viser, men da må man ha gigabitutstyr i alle ledd.
I tillegg er det satt opp aksesslister som begrenser konsolltilgang til kun de interne nettene. Konfigurasjonen finnes i vedlegg G.

5.1.3 Bruk av VLAN i OpenStack

Det blir opprettet ett nytt VLAN for hvert tenant i OpenStack. Dette vil kunne bli et problem dersom man ruller ut i stor skala, siden maks teoretisk antall VLAN er 4096, og av dem igjen er noen reserverte. For vår oppdragsgiver vil det trolig ikke bli noe problem i første omgang, men gitt et scenario der hver student ved skolen skal få tilgang, i tillegg til de emnene som skal bruke systemet vil dette bli et problem.
Løsningen på dette problemet kan imdlertid være på vei.
Sommeren 2011 lanserte Cisco nyheten VXLAN (Virtual eXtensible LAN) [79] [78]. I korte trekk er dette en utvidelse av dagens VLAN-teknologi, der VLAN-ID feltet i pakken er doblet fra dagens 12 bit til 24 bit. Det åpner for et teoretisk maksimum på over 16 millioner (2^{24}) unike VLAN-ID'er. Et annet skaleringsproblem er utrulling av VLAN til ulike switcher. I dag er *VTP* eneste mulighet, men det er ofte deaktivert av sikkerhetshensyn, og man er da tvunget til å konfigurere switchene mer eller mindre manuelt. VXLAN har muligheten for å distribusjon over lag 3 (f.eks IP). Muligheten for å bygge inn støtte for VXLAN i OpenStack er allerede i utviklerenes tanker [43].

5.1.4 Valg av DHCP-server

Valget av DHCP-server falt på dnsmasq [34]. Den har en meget enkel konfigurasjon, samtidig som den uten konfigurasjon over hodet fungerer som DNS-forwarder for alle DHCP-klienter. I tillegg er det dnsmasq som blir brukt av OpenStack, og det vil være hensiktsmessig å holde seg til den samme i hele infrastrukturen. Eksempel på konfigurasjon av dnsmasq ligger i vedlegg B. ISC DHCP Server ble også vurdert, da vi hadde erfaring med denne fra emnet Systemadministrasjon [24], av tidligere nevnte grunner ble denne valgt bort. Eksempel på konfigurasjon ligger i vedlegg G. For å få det interne nettverket bak ruteren til å få kontakt med dnsmasq måtte det settes opp et *DHCP-relay* i SkyRoute. I en cisco-ruter er slik funksjonalitet innebygget, og gjøres med kommandoen `ip helper-address <ip>` på det nettverksgrensesnittet pakkene vil komme inn på [1].

I vårt tilfelle innebærer det alle de virtuelle netverksgrensesnittene på fa0/1. I tillegg må det legges inn en statisk rute til vårt interne nett på brighton via kommandoen: `route -add net 192.168.0.0 netmask 255.255.0.0 gw 10.0.0.2`. Ruter som blir lagt inn med denne fremgangsmåten slettes ved restart, og kommandoen ble derfor lagt inn i oppstartsscriptet `/etc/rc.local`.

5.1.5 Iptables

Iptables [46] er en modul i linuxkjernen som utfører filtrering av IP-pakker. Ved hjelp av denne kan man filtrere ut hvilke pakker som skal ha aksess til hvilke nett og hvor de skal sendes. Man har også muligheten til å sette op NAT. Dette utnyttet vi for å dele ut internett til vårt interne nett via `brighton`. Disse filterne ble satt opp:

```
1. iptables -A FORWARD -s 10.0.0.0/30 -i eth1 -j ACCEPT
2. iptables -A FORWARD -s 192.168.10.0/24 -i eth1 -j ACCEPT
3. iptables -A FORWARD -d 10.0.0.0/30 -i eth0 -j ACCEPT
4. iptables -A FORWARD -d 192.168.10.0/24 -i eth0 -j ACCEPT
5. iptables -A FORWARD -s 192.168.30.0/24 -i eth1 -j DROP
6. iptables -A FORWARD -d 192.168.30.0/24 -i eth0 -j DROP
7. iptables -A FORWARD -s 192.168.99.0/24 -i eth1 -j DROP
8. iptables -A FORWARD -d 192.168.99.0/24 -i eth0 -j DROP
9. iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

Linje 1-3 aksepterer trafikk fra gitte nettverk ut mot internett. Fra linje 4 til 6 aksepteres trafikk fra internett til de samme nettverkene. Påfølgende linjene 5 til 8 sperrer trafikk både til og fra internett for VLAN30 og VLAN99. Linje 9 forteller at pakker på vei ut mot internett skal oversettes med NAT, at alle skal bruke den samme globale IP-adressen, samt at pakkene skal sendes til eth0. For at dette skal fungere må man sette et kernel-flagg som muliggjør videresending av IPv4-adresser, slik:

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

Da dette blir resatt når maskinen starter på nytt må flagget settes i filen /etc/sysctl.conf. Filterne i iptable blir også resatt ved en omstart. Derfor ble konfigurasjonen lagret ved hjelp av kommandoen:

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

Konfigurasjonen blir hentet tilbake i oppstarten ved å legge inn:

```
iptables-restore < /etc/iptables.conf
```

i oppstartsscriptet `/etc/rc.local`.

5.2 Nettverk i OpenStack

5.2.1 Nettverksmodeller

Tjenesten nova-network håndterer alle nettverksoppgaver i OpenStack. Det finnes tre forskjellige modeller å velge i mellom: [62]

- FlatNetworking

- FlatDHCP
- VLAN Network

Uavhengig av hvilken modell man velger er det mange forskjellige muligheter for infrastruktur i OpenStack. Vi vil ikke gå i dybden på alle disse, men fokusere på den som er gjeldende for vårt oppsett.

Nova-compute støtter både IPv4 og IPv6 ved å sette opp *dual-stack*. For at dette skal fungere må hver node som kjører en nova-service ha `python-netaddr` og `radvd` installert. Disse installeres via apt-get, for så å kjøre disse kommandoene:

```
sudo bash -c "echo 1 > /proc/sys/net/ipv6/conf/all/forwarding"
sudo bash -c "echo 0 > /proc/sys/net/ipv6/conf/all/accept_ra"
```

I tillegg må `-use_ipv6=True` legges til i nova.conf, og ytterligere konfigurasjon kan gjøres via `nova-manage`.

Flat Networking
Dette er den enkleste modellen. Man definerer et subnet instansene skal bruke, og ved opprettelse blir det injisert en IP fra dette. Måten dette gjøres på er at instansens diskfil monteres opp, og informasjon blir lagret i `/etc/network/interfaces`. Dette gjelder bare for linux-instanser, da denne måten å injisere filer på ikke fungerer på windows. Nettverksbroen for å binde sammen eksternt og internt nett må settes opp manuelt på alle noder i infrastrukturen. Alle instanser er koblet til den samme nettverksbroen, og man er begrenset til én slik. Dette gjør denne modellen mest aktuell for mindre implementasjoner.

Flat DHCP
Flat DHCP fungerer tilnærmet på samme måten som Flat Networking, men her starter man en DHCP-server for å dele ut IP-adresser til instansene fra det spesifiserte subnettet. Alle instansene er fremdeles koblet til den samme nettverksbroen. DHCP-serveren som benyttes er dnsmasq, og denne settes opp til å lytte på *DHCPDISCOVER* på nettverksbroen.

I begge disse flate modellene får instansene delt ut offentlige IP-adresser, enten globale rutbare IP-adresser eller fra et lokalt nett hvor NAT er satt opp i. Nova-network fungerer altså ikke som gateway.

VLAN Network
Denne modellen er standardmodellen og også den vi har valgt. Her opprettes det et eget VLAN, et eget subnet og en egen nettverksbro for hvert tenant. For hvert av disse VLANene blir det startet en DHCP-server, og instansene blir tildelt en privat IP-adresse derfra som bare er tilgjengelig innenfor sitt VLAN (i praksis innenfor sitt tenant). Instansene bruker da compute-nodens nettverksbro som gateway. For å få tilgang til instansene må man enten benytte seg av *VPN* eller konseptet floating IP's.

CloudPipe [53] er en spesiell instans som opptrer som VPN-server. For å få aksess til instansene må man sette opp en slik, og hente ut sertifikat og nøkkel fra denne. Alternativet til dette er å benytte seg av konseptet floating IP. Man definerer et sett med globale IP-adresser, gjerne et helt subnet, som man da kan legge til instansene. Måten det fungerer på er at IP-adressen blir lagt til på gitt tenant's nettverksbro, og det blir opprettet iptables-regler for å rute trafikken til rett instans. Vi har valgt å bruke floating IP i vår implementasjon med tanke på brukervennlighet. Her trengs det ingen komplisert oppkobling for sluttbrukeren.

Denne modellen passer godt i større implementasjoner siden den er mest dynamisk, og det trengs da ingen manuell konfigurasjon av nettverksbroer eller andre nettverkskort. Modellen avhenger av at man har nettverksutstyr som takler IEEE 802.1Q VLAN-tagging.

5.2.2 Floating og fixed IP adressering

OpenStack bruker to konsepter for IP-adressering:

- Fixed IP
- Floating IP

Alle instanser får en fixed IP fra sin tenant's respektive fixed nettverk. Hvordan denne deles ut kommer an på hvilken av nettverksmodellene man har valgt. Det subnettet fixed IP bruker er ikke tilgjengelig utenfor instansene i samme tenant, og er følgelig å anse som et internt nett for alle instanser innenfor denne tenant. Compute noden som instansene kjører på vil som tildligerere nevnt få satt opp regler i iptables for å rute trafikk inn og ut mot internett.

For å gi instansene en offentlig IP-adresse må man tilegne den en floating IP. Å gjøre dette er en to-stegs prosess. Først må man allokere en adresse fra et definert *IP-pool*, og deretter må man koble denne adressen til en instans. I konfigurasjonen av nova definerer man et subnet med offentlig adresser man kan allokere floating IP adresser i fra. Dette trenger ikke nødvendigvis å være globalt rutbare adresser, men det må naturligvis være adresser på et nett som er tilgjengelig for brukeren.

Floating IP blir satt opp av nova-network. Det blir satt opp regler i iptables både for filtrering av trafikk og for NAT. I tillegg blir adresser fysisk lagt til på det nettverkskortet som er definert som public interface i novas konfigurasjon.

Her ser vi utdrag fra loggen etter at en floating IP (192.168.10.129) er tilknyttet en instans:

```
2012-04-24 14:15:06 DEBUG nova.utils [req-0420a717-6545-4a2e-a02e-22837f484250
ef2fbadc9e1e4fa4b89f0299741a25d8 2a118220740f4df2b5a0cfc7398dffb7] Running cmd
(subprocess): sudo ip addr add 192.168.10.129/32 dev eth0 from (pid=13753)
execute /usr/lib/python2.7/dist-packages/nova/utils.py:219
```

```
2012-04-24 14:15:06 DEBUG nova.utils [req-0420a717-6545-4a2e-a02e-22837f484250
ef2fbadc9e1e4fa4b89f0299741a25d8 2a118220740f4df2b5a0cfc7398dffb7] Running cmd
(subprocess): sudo iptables-save -t filter from (pid=13753) execute
/usr/lib/python2.7/dist-packages/nova/utils.py:219

2012-04-24 14:15:06 DEBUG nova.utils [req-0420a717-6545-4a2e-a02e-22837f484250
ef2fbadc9e1e4fa4b89f0299741a25d8 2a118220740f4df2b5a0cfc7398dffb7] Running cmd
(subprocess): sudo iptables-restore from (pid=13753) execute
/usr/lib/python2.7/dist-packages/nova/utils.py:219

2012-04-24 14:15:06 DEBUG nova.utils [req-0420a717-6545-4a2e-a02e-22837f484250
ef2fbadc9e1e4fa4b89f0299741a25d8 2a118220740f4df2b5a0cfc7398dffb7] Running cmd
(subprocess): sudo iptables-save -t nat from (pid=13753) execute
/usr/lib/python2.7/dist-packages/nova/utils.py:219

2012-04-24 14:15:06 DEBUG nova.utils [req-0420a717-6545-4a2e-a02e-22837f484250
ef2fbadc9e1e4fa4b89f0299741a25d8 2a118220740f4df2b5a0cfc7398dffb7] Running cmd
(subprocess): sudo iptables-restore from (pid=13753) execute
/usr/lib/python2.7/dist-packages/nova/utils.py:219
```

Før kallene til iptables blir reglene generert av novas API. I dette tilfellet vil følgende regel bli lagt til:

```
Chain nova-network-float-snat (1 references)
target     prot opt source          destination
SNAT       all  --  172.16.0.3     anywhere       to:192.168.10.129
```

5.2.3 Valg av modell

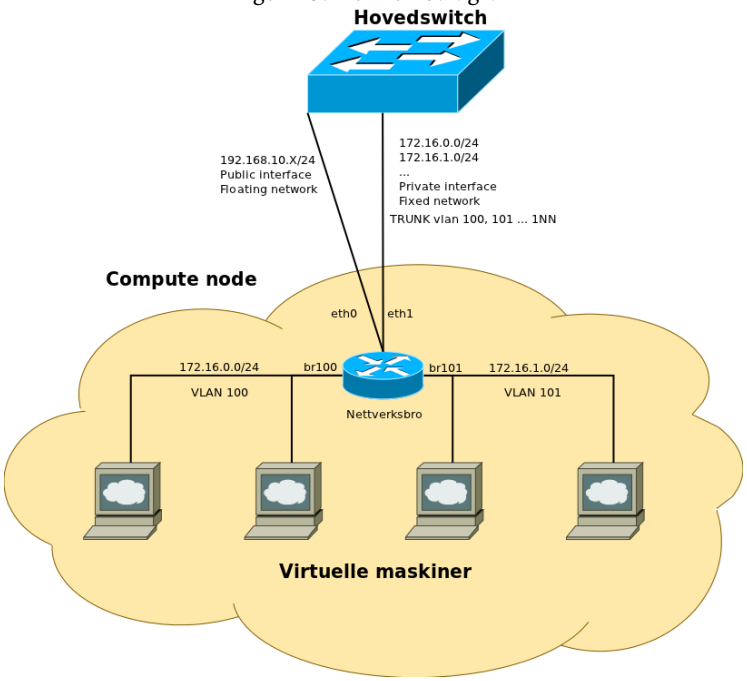
Som nevnt i 5.2.1 valgte vi å gå for VLAN Manager modellen. Grunnlaget for dette er at til oppdragsgivers bruk er man avhengig av å separere trafikk mellom instanser i de forskjellige tenant's og emnene. Man vil kunne gjennomføre dette med de to andre modellene, men det vil kreve manuelt oppsett av både nettverksbroer og VLAN på alle compute nodene. Med VLAN Manager blir dette gjort automatisk, og er følgelig den mest hensiktsmessige modellen.

Alle våre noder er satt opp med to nettverkskort. OpenStack opererer med public og private interface; Public interface skal adresseres med offentlig tilgjengelig adresse, men private interface er satt opp uten IP adresse. På volume noden er de to to nettverkskortene satt opp med aggregering for å doble den teoretiske båndbredden fra 1 Gbps til 2 Gbps. Dette er fyldig forklart i bacheloroppgaven SkyHiGh I/O [17].
I figur 13 ser vi hvordan nettverket fungerer i praksis. Compute noden oppretter et VLAN-interface per tenant. Dette knyttes til nettverkskortet som er definert som private interface, i vårt tilfelle eth1. Deretter settes det opp en nettverksbro knyttet til dette VLAN-interfacet. Broen er nummerert etter sin respektive VLAN ID, og fungerer som gateway for alle instanser i sin respektive tenant. Switchporten eth1 er koblet til er satt opp som trunk port for alle VLAN ID'er tennant kommer til å bruke. Derfor er det også satt opp uten IP adresse, slik at eth1 blir å regne som en switch for alle nettverksbroene compute-noden er satt opp med. Dette står det imdlertid ingenting om i dokumentasjonen, men det kommer ikke til å fungere på noen annen måte. Setter man en IP på dette nettverkskortet vil private interface ikke fungere som det skal. Siden trafikken på det interne nettet er VLAN-tagget trafikk, blir trafikk mellom instanser i samme tenant på forskjellige

compute noder videresendt rett.

Eth0 har egentlig ingenting med det interne nettverket mellom instansene å gjøre, men er med i figuren for å vise rollen til nettverkskortet som er public interface. Trafikk ut mot / inn fra internett blir rutet til dette nettverkskortet av iptables.

Figur 13: Nettverksdiagram



5.3 Infrastruktur

5.3.1 Maskinvare

Den infrastrukturen som var tilgjengelig under prosjektperioden var fire HP ProLiant DL320 G5 med dobbeltkjernet Intel Xeon 3060 prosessor. To av disse er utstyrt med 4 GiB minne, og to med 2 GiB minne. I tillegg har vi en lagringsboks av typen HP ProLiant DL 320s med samme prosessor som de andre maskinene og 4 GiB minne og 12x 250GB harddisker. Vi valgte å bruke den ene serveren med 4 GiB minne som controller-node, og de andre som henholdsvis compute og volume-noder. Anbefalt maskinvarekonfigurasjon fra OpenStack ligger i en litt annen skala enn det vi her hadde tilgjengelig. På controlleren anbefales en 64-bit x86 prosessor, 12 GiB RAM, 30 GB diskplass og 1 stk gigabit-nettverkskort. For compute-noder er anbefales det 64-bit x86-prosessor, 32 GiB RAM, 30 GB diskplass. Alle noder må ha gigabit nettverkskort.

Vårt utstyr når ikke helt opp til de anbefalte spesifikasjonene, men etter testinstalasjon konkluderte vi med at dette holdt til et *proof-of-concept*.

Hypervisor og operativsystem

Prosessen frem mot endelig infrastruktur og maskinoppsett endte til slutt med fem forskjellige forsøk før vi fant en som fungerte:

1. Første oppsett var et forsøk på å tilfredsstille alle krav vår oppdragsgiver stilte i forbindelse med operativsystem og hypervisor. Følgelig ble alle noder satt opp med Debian Wheezy og Xen som hypervisor ved hjelp av pakken xen-linux-system.
   Vi fant etter hvert ut at OpenStack krever at man kjører XenCloudPlatorm i stedet for XenServer, som følger med nevnte pakke. Grunnen til dette er at OpenStack kommuniserer med tjenesten xapi. XenServer blir levert med xend, som det ikke er støtte for. På dette tidspunktet kom det ikke klart frem av dokumentasjonen at man måtte kjøre Xen Cloud Platform eller hvilke pakker man trengte for bruke Xen. Eksempler på konfigurasjon var dårlig eller ikke-eksisterende. Dokumentasjonen er forbedret i ettertid, og har nå en seksjon som forklarer Xen's oppbygning og hvordan man kan benytte seg av Xen som hypervisor. Eksempler på konfigurasjon er fremdeles mangelfullt.

   Løsning: På dette tidspunktet ble det besluttet å bytte operativsystem fra Debian til Ubuntu 11.10 på controller-noden siden all dokumentasjon og stabile pakker ble distribuert til denne distroen. De pakkene som ligger i Debian Wheezys pakkebrønn er hele tiden bygget av nyeste kode, og egner seg dermed dårlig i et produksjonsmiljø. Vi fortsatte imdlertid med Debian på compute-nodene for å finne ut om bytte av operativsystem i det hele tatt ville gjøre noen forandring. Dersom det ikke hadde hatt noen hensikt ville det vært en del bortkastet tid å kjøre oppgradering på alle nodene.

2. Videre fant vi ut at OpenStack må kjøres i det Xen kaller DomU, et domene uten direkte tilgang på maskinvaren. I praksis betyr det at alle tjenestene måtte kjøres på en virtuell maskin. Umiddelbart så vi her at den maskinvaren vi hadde tilgjengelig kom til å bli en flaskehals, siden vi ble nødt til å kjøre virtuelle maskiner inni en virtuell

maskin. I tillegg til disse hindringene, er også dokumentasjonen for å benytte Xen meget mangelfull, noe som gjorde det for tidkrevende å implementere [51].

Løsning: For å komme rundt dette problemet besluttet vi å bytte hypervisor til KVM. Denne er brukt somt standard, og så godt som all dokumentasjon er skrevet rundt denne. Med KVM ferdig satt opp klarte vi å få hele rammeverket til å kjøre, alle tjeneste ble oppdaget og de snakket med hverandre.

3. Imidlertid støtte vi på et problem, der compute-nodene ødela databasestrukturen til nova da dette skulle synkroniseres. Ved å undersøke versjonsnumrene på pakkene i de forskjellige operativsystemene fant vi at Debian og Ubuntu kom med forskjellige utgaver av OpenStack-modulene i sine pakkebrønner, og det var dette som førte til versjonskonflikt i databasen. Versjonskonflikten ødela webgrensesnittet og all annen funksjonalitet. Det var ikke mulig å få lik versjon på disse operativsystemene fordi versjonen som lå i Debian var en tidlig *bleeding edge* utgave av OpenStack Essex, mens den mest oppdaterte pakkebrønnen som kunne legges til i Ubuntu var Diablo utgaven. Det var heller ikke mulig å legge inn Ubuntu sin pakkebrønn i Debian, på grunn av misforhold i digitale signaturer og avhengigheter på eldre pakker enn de som fantes i Debian.

   Løsning: Vi løste det ved å legge inn Ubuntu 11.10 som operativsystem på samtlige noder i infrastrukturen for å forsikre oss om at vi ikke ble hindret av kompatibilitetskonflikter.

4. Dette oppsettet fungerte fint, men var svært ustabilt. Det var tilfeldig hvor ofte en instans faktisk startet opp. Etter å ha inspisert loggfiler fant vi fort ut at loggingen var veldig god da instansene faktisk startet opp. Derimot var loggingen meget dårlig da operasjoner feilet. Det eneste som ble logget da instansene enten hang i "BUILD" tilstand eller endte opp i umiddelbar "ERROR" var:

   ```
   2012-03-05 17:37:58 WARNING nova.compute.manager [-] Found 3 in the
   database and 0 on the hypervisor.
   ```

   Bytte av OpenStack versjon

   I tillegg var det en mer konsistent feil ved tildeling av floating IP. Operasjonen feilet systematisk to av tre ganger. Ved inspeksjon av loggfilene fant vi ut at API-kallene ble konsekvent sendt til feil nettverks-node, med den følge at kommandoen aldri returnerte. Dette mistenkte vi at var en feil i koden og vi sendte inn en feilrapport [73], i tillegg til å snakke med forskjellige folk på OpenStacks offisielle *IRC*-kanal. Dessverre viste det seg at svært få andre hadde opplevd det samme og vi fant ingen løsning på denne feilen.

   Løsning: På dette tidspunktet var alle forsøkene på implementasjon gjort med Diablo-versjonen av OpenStack. Etter samtaler med OpenStack-utviklerere ble vi anbefalt å oppgradere til Essex-utgaven som nå var i *RC*-utgave. Det ble ikke lenger fikset feil i Diablo, og også i forhold til utviklingsdelen av prosjektet ble dette anbefalt fordi Essex var den første versjonen som faktisk hadde Horizon som en offisiell modul. Den nye utgaven hadde allerede implementert små deler av det vi mente den manglet tid-

ligere, og ikke minst var den nye utgaven laget for å være utvidbar, noe som ikke var tilfelle med Diablo-utgaven. Oppgraderingen til Essex medførte at operativsystemet igjen måtte oppgraderes, denne gangen til betautgaven av det kommende Ubuntu 12.04 LTS. I forkant av denne oppgraderingen vurderte vi hvilken risiko dette ville ha, og eventuelle fallgruver som kunne dukke opp.

Løsning: Ut i fra denne vurderingen kom vi opp med to akseptable utfall:

- Suksessfull oppgradering og tilsvarende eller bedre funksjonalitet enn det systemet hadde før oppgradering
- Feilet oppgradering, og en tilbakerulling til sist fungerende oppsett

Det ble besluttet å beholde Ubuntu 11.10 på compute nodene i tilfelle vi endte opp med sistnevnte utfall. For å realisere muligheten til å rulle systemet tilbake tok vi en klone av hele disken med dd [71], for backup i tilfelle feilet oppgradering.

5. Oppgraderingen var vellykket, Essex ble installert på controller-noden og oppsettet fungerte utmerket. Instansene startet stabilt, og feilen med floating IP var borte. Compute-nodene fortsatte å kjøre på Ubuntu 11.10, men selfølgelig med Essex i stedet for Diablo. Dette fungerte også godt siden det var mulighet for å legge til en pakkebrønn fra OpenStack med samme pakkeversjoner som Ubuntu 12.04 har i sine offisielle pakkebrønner.

Endringene i Essex var ganske omfattende og en del kommandoer var forandet. Blant annet var administrasjonen av keystone forandret, da denne nå hadde fått en klient på lik linje med nova-client. Sammen med Essex kom det også et meget godt dokument fra den offisielle dokumentasjonen. Dette dokumentet går nøyaktig gjennom hva som må gjøres for å få et fungerende system [64].

5.4 Installasjon og konfigurasjon

Det er en del forskjellige fremgangsmåter å velge mellom når man skal installere OpenStack, og man bør på forhånd av installasjonen ha helt klart for seg hvilken infrastruktur man skal kjøre på og hvilken nettverksmodell det legges opp til. Alt kan selvfølgelig forandres etter installasjon, men det vil være mest hensiktsmessig å ha en god plan før man setter i gang.

Når man skal installere har man i prinsippet to valg:

- Benytte seg av et av mange automatiserte script som har blitt laget for å assistere installasjonen.
- Manuelt oppsett

I løpet av prosjektperioden prøvde vi begge metodene for å finne ut hva som lønner seg for et produksjonssystem.

5.4.1 Automatiserte script

Ved hjelp av DevStack [12] kan man sette opp en fungerende OpenStack-installasjon med så lite som to kommandoer. DevStack er laget for å brukes i et utviklingsmiljø som et proof-of-concept hvor man kan teste de forskjellige komponentene, og er ikke anbefalt til bruk i produksjon. Installasjon av OpenStack via DevStack gjøres slik:

```
git clone git://github.com/OpenStack-dev/devstack.git
cd devstack; ./stack.sh.
```

Scriptet er interaktivt, og vil underveis be om å velge hvilke moduler som skal installeres, samt velge passord og tokens for de forskjellige tjenestene.

Selve installasjonen tar gjerne litt tid, men det er ingen konfigurasjon som må gjøres. Når installasjonen er ferdig har du en fungerende proof-of-concept OpenStack-installasjon. Vi installerte med DevStack-scriptet i virtuelle maskiner når vi først begynte å teste OpenStack. Siden scriptet laster ned kildekode fra git i stedet for å kjøre installasjon fra pakkebrønn, betyr det at systemet ikke kan oppdateres på en konvensjonell måte via pakkebehandler. Alle filer vil ligge i andre kataloger enn det som står i dokumentasjonen til OpenStack. DevStack setter opp en installasjon, der alle moduler er på samme node. Derfor vil det ikke være aktuelt å sette dette i produksjon. I tillegg har man lite innflytelse på nettverksoppsettet. Man kan heller ikke ha et system som ikke lar seg vedlikeholdes med en pakkebehandler i produksjon. Allikevel var denne metoden nyttig i testfasen, da vi fort fikk opp et fungerende system slik at vi kunne kartlegge funksjonalitet relativt tidlig.

Et annet script vi testet var scriptet til ManagedIT [42]. Dette scriptet kommer med en konfigurasjonsfil der man endrer informasjonen slik at den passer med ønsket oppsett, og deretter blir OpenStack installerert via pakkebrønnen. Scriptet fungerer bare på Ubuntu, og det legger til sin egen pakkebrønn for å sikre at bare den versjonen scriptet er laget for blir installert. Oppsettet bygges ut ifra konfigurasjonen på akkurat samme måte som en manuell installasjon, og man sitter således igjen med et stabilt system. Erfaringen med dette scriptet var meget god, og ved nøye studering av kildekoden her tilegnet vi oss god kunnskap om hvordan en manuell installasjon skal utføres.

5.4.2 Manuell installasjon

Ved manuell installasjon kan man enten laste ned kildekoden fra Launchpad [57], laste ned fra GitHub [59] eller installere fra pakkebrønnen. Per april 2012 er OpenStack pakket for Ubuntu, Debian, CentOS, Fedora og RHEL [50]. Etter test av de to nevnte scriptene konkluderte vi med at en manuell installasjon er veien å gå. På den måten lærer man mest om hvordan de forskjellige modulene henger sammen og hvordan de skal konfigureres. Man vil ha full kontroll på systemet, og det er mye enklere når man har satt det opp selv.

Ved siden av disse metodene finnes det en dedikert OpenStack linuxdistribusjon bygget på Ubuntu 10.04 LTS. Firmaet som har laget denne heter StackOps [77] og har spesialisert seg på levering av OpenStack med hardware. Per 24. april 2012 er denne i versjon 0.5, og distribusjonen er modifisert til å utelukkende inneholde de komponentene som kreves for at OpenStack Nova skal kjøre.

5.4.3 Installasjon

Fra og med Essex-versjonen inneholder dokumentasjonen en meget god steg-for-steg guide for et fungerende oppsett som vi valgte å benytte [64].

Siden guiden er meget detaljert og selvforklarende har vi valgt å dokumentere de overordnede valgene vi gjorde, og feil vi oppdaget i guiden.

Før installasjonen

Følgende forhåndskrav må være tilfredsstilt:

- Controlleren må settes opp som *NTP* server for alle de andre nodene i infratrukturen. Tidssynkronisering er kritisk for logg av ressursbruk, samt databasebehandling.
- En root-bruker eller en bruker med sudo-rettigheter under installasjonen.

Keystone

Keystone installeres med kommandoen `apt-get install keystone`. Databasen `keystone` må opprettes, samt MySQL-brukeren `keystone`. I konfigurasjonsfilen til Keystone må mysql velges, samt at brukernavn og passord til databasen og admin-token må legges inn. Så kan databasen initialiseres med kommandoen `keystone-manage db_sync`.
Tenants, users og roles må så defineres, minimum 1 stk må settes opp for at de andre servicene kan autensiere seg mot Keystone. Dette gjøres med `keystone`-kommandoen i kommandolinja, og utføres på følgende måte:

- Lag en standard tenant
- Lag en standard bruker
- Lag standard roller
- Gi rolle til bruker i en tenant
  - Her fant vi en feil i guiden, der den sier at standardrollen for vanlige brukere skal hete “memberRole”. Dette kommer ikke til å fungere, fordi alle de andre modulene er konfigurert med en standard brukerrolle som heter “member”. Dersom man kaller rollen for “memberRole” ender man opp med at brukeren ikke får rettigheter til noe som helst.
  - På det tidspunktet vi kjørte installasjonen sto det også at administrator rollen skulle hete “adminRole”, noe som ville hatt samme resultat som punktet over for administratoren. Rollen skal hete “admin”. Dette er imidlertid rettet i etterkant.
- Lag en service-tenant
- Lag en Glance/Nova/Swift Service User i Service Tenant (må gjøres for alle services som skal autensiere seg mot Keystone)
- Gi admin-role til Glance/Nova/Swift Service User i deres respektive tenant.

Når tenants og roles er satt opp, må services og endpoints opprettes i Keystone, eksempel på kommando for nova:

```
--token 012345SECRET99TOKEN012345 \
--endpoint http://192.168.206.130:35357/v2.0/ \
service-create \
 --name=nova \
 --type=compute \
 --description="Nova Compute Service"''

‘‘keystone --token 012345SECRET99TOKEN012345 \
--endpoint http://192.168.206.130:35357/v2.0/ \
endpoint-create \
 --region RegionOne \
 --service_id=abc0f03c02904c24abdcc3b7910e2eed \
 --publicurl='http://192.168.206.130:8774/v2/%(tenant_id)s' \
 --internalurl='http://192.168.206.130:8774/v2/%(tenant_id)s' \
 --adminurl='http://192.168.206.130:8774/v2/%(tenant_id)s'''
```

Etter oppsett av services og endpoints er det på tide å verifisere at installasjon og konfigurasjon virker, noe som kan gjøres ved en enkel liten curl-kommando:

```
-d '{"auth": {"tenantName": "adminTenant", \
"passwordCredentials":{"username": "adminUser", "password": "secretword"}}}' \
-H "Content-type: application/json" http://192.168.206.130:35357/v2.0/tokens | \
python -mjson.tool''
```

Har alt gått etter planen skal man i dette tilfellet få tilbake info om adminUser.

Glance

Videre følger installasjonen av glance: `apt-get install glance`, oppretting av glance-database og bruker, og tilpasning av konfigurasjonsfiler (admin-token o.l.).
Neste steg er å laste opp et image og eksportere environment-variabler som inneholder brukernavn, tenant, passord, autensierings-url og regionnavn. Imagene må lastes opp via glance sitt kommandolinjeverktøy som finnes i pakken python-glance. Opsjonene til verktøyet er mange, og er godt forklart i Glance egen dokumentasjon [70]. Eksempel for å laste opp et ferdig Windows image:

```
glance add name="Windows_Server_2008_R2_Enterprise_x64" \
is_public=true container_format=ovf disk_format=raw < windowsserver.img
```

Ubuntu har selv laget ferdige image som er klargjort for distribusjon i en sky [86] og disse kan legges direkte til glance. Man kan også lage image selv ved å kjøre en vanlig installasjon i f.eks VirtualBox eller direkte på KVM, for så å laste opp diskfilen. Dette er den mest hensiktsmessige måten å laste opp et Windows image på. Hvordan dette gjøres forklares i dokumentasjonen [54]. Her fant vi imdlertid en grov feil i guiden for å laste opp og lage et Windows image. Dokumentasjonen forteller hvordan man skal sette i gang installasjon med KVM, som forutsetter at man laster inn en driver for den virtuelle diskkontrolleren. Kommandoen inneholder en .vfd fil, som ikke lenger finnes på lenken i dokumentasjonen. Derimot ligger det en .iso fil der, som man kan benytte. Siden filformatet på driverene er endret, så er også eksempelkommandoen feil, og må endres til

```
sudo kvm -m 1024 -cdrom win2k8\_dvd.iso -drive file=windowsserver.img, \
if=virtio,boot=on -drive file=virtio-win-0.1-22.iso,media=cdrom \
-boot d -nographic -vnc :0
```

Feilen ble rapportert på launchpad, og den ble raskt rettet [72].

Nova

Som nevnt i 5.3.1 valgte vi å kjøre med KVM hypervisor. KVM er standard i OpenStack, og det er ingen konfigurasjon som må gjøres. Skal man bruke Xen, XCP eller Citrix XenServer står det godt forklart i install-guiden hva som kan og må konfigureres [55].

For nettverksoppsettet ble det valgt VLAN Manager, og fremgangsmåten er beskrevet i 5.2.3.

Før installeringen av Nova opprettes databasen "nova" med bruker "nova", etterfulgt av installasjon av rabbitmq-server som ligger i pakkebrønnen.

Selve installasjonen av Nova på controlleren gjøres med denne kommandoen:

```
sudo apt-get install nova-compute nova-volume nova-vncproxy nova-api nova
-ajax-console-proxy nova-cert nova-consoleauth nova-doc nova-scheduler
 nova-network
```

Etter installasjonen av Nova må `nova.conf` tilpasses til vårt oppsett. Først at filens eier er satt til root:nova, samt legge til eller endre valgene slik at rett hypervisor, nettverk, sql-passord og lignende er valgt. Vår nova.conf-fil ligger i vedlegg G. Når filen er ferdig satt opp må nova-api, compute, network, scheduler, vncproxy og volume restartes, etterfulgt av database-sync/opprette tabeller i databasen med `nova-manage db sync`.
Nova logger til /var/log/nova/nova-manage.log, og eventuelle feil vil vises i loggfilen. Nettverkene for fixed og floating netterk må opprettes, for fixed nettverk blir kommandoen:

```
 nova-manage network create <name> --fixed_range_v4=X.X.X.X/XX
--num_networks=X --network_size=X --vlan=XXX
```

`-fixed_range_v4` må være et subnet innenfor det man har satt i novas konfigurasjonsfil under paramtereren `-fixed_range`. For floating nettverk, kjører man `nova-manage floating create -ip_range=X.X.X.X/XX`. Også her må dette samsvare med novas konfigurasjon. Installasjon og konfigurasjon av compute-noder er en del enklere enn controlleren, da compute-nodene kun trenger pakkene nova-network, nova-compute og nova-api. Konfigurasjonsfilen nova.conf kan kopieres over fra controlleren uten modifikasjon.

Når alle compute-nodene og volume-nodene er installert og konfigurert skal det meste være ferdig satt opp, og kommandoen `sudo nova-manage service list` skal forhåpentligvis gi oss et par smile-fjes tilbake:

```
root@dublin:~# nova-manage service list
Binary           Host        Zone  Status     State Updated_At
```

```
nova-network      dublin      nova  enabled    :-)   2012-04-30 12:36:07
nova-scheduler    dublin      nova  enabled    :-)   2012-05-01 11:03:41
nova-compute      cardiff     nova  enabled    :-)   2012-05-01 11:03:34
nova-compute      newcastle   nova  enabled    :-)   2012-05-01 11:03:39
nova-network      cardiff     nova  enabled    :-)   2012-05-01 11:03:41
nova-compute      manchester  nova  enabled    :-)   2012-05-01 11:03:41
nova-compute      dublin      nova  disabled   XXX   2012-03-25 13:01:54
nova-network      newcastle   nova  enabled    :-)   2012-05-01 11:03:37
nova-volume       kingston    nova  enabled    :-)   2012-04-10 13:13:14
nova-cert         dublin      nova  enabled    :-)   2012-05-01 11:03:50
nova-network      manchester  nova  enabled    :-)   2012-05-01 11:03:39
nova-cert         manchester  nova  enabled    :-)   2012-05-01 11:03:48
nova-cert         newcastle   nova  enabled    :-)   2012-05-01 11:03:35
nova-cert         cardiff     nova  enabled    :-)   2012-05-01 11:03:38
nova-consoleauth  dublin      nova  enabled    :-)   2012-05-01 11:03:47
```

Alle OpenStack-kommandoene som kjøres via kommandolinjen bruker credentials som inneholder brukernavn, passord og URL'er til API'et. Disse kan brukes ved hjelp av `export` i kommandolinjen, eller for å få de lagret permanent legge de i .bashrc (eller tilsvarende). Eksempel på en slik fil ligger i vedlegg H.

Horizon

Når compute og controller er installert kan Horizon (webgrensesnittet) og nødvendige pakker installeres, og utføres i hovedtrekk på samme måte som resten av modulene via apt-get: `sudo apt-get install libapache2-mod-wsgi openstack-dashboard`. Databasen dash opprettes med tilsvarende bruker, og i `/etc/openstack-dashboard/local_settings.py` må database settes til MySQL-databasen dash. Etter at databasen er syncet med kommandoen `/usr/share/OpenStack-dashboard/manage.py syncdb` kan webgrensesnittet nås via en webleser på IP-adressen til maskina det ble installert på. For å logge inn brukes brukernavn og passord som ble satt opp tidligere i Keystone.

5.5 Utvikling

5.5.1 Forberedelse

I forberedelsen til utviklingsfasen jobbet gruppen med opplæring av både python og Django. Dette ble gjort hovedsaklig med Programming Python [41], samt nettressursene Beginner's guide to Python [74] og Python Programming Manual [15]. Til Django ble nettsiden til Django [21] brukt som ressurs. Devstack ble brukt som test-installasjon lokalt.

5.5.2 Metode

Gruppen fikk tidlig et inntrykk av at en agil utviklingsmodell måtte benyttes i dette prosjektet, siden ingen i gruppen hadde vært borti et slikt prosjekt før. OpenStack er nytt og ingen har utført et slikt prosjekt på HiG tidligere. Oppdragsgiver hadde heller ikke benyttet seg av OpenStack eller Cloud Computing på en slik måte. Noe som gjorde det

vanskelig å fastsette rekkefølge på inkrementer i utviklingsperioden.

Gruppen valgte å benytte seg av SCRUM som systemutviklingsmodell for videreutviklingen av Horizon. Gruppen konkluderte med at SCRUM passet best med tanke på å jobbe med Open Source-Miljøet, levering av inkrementer, tillegg av krav og tidsestimater. Gruppen så det også som en fordel i forhold til læringskurven ved OpenStack.

Vi valgte å benytte oss av noen konsepter fra inkrementell utvikling. Ved hver demo satt vi av en periode til å teste, samt rulle ut inkrementet på systemet. Dette håpet vi skulle sørge for at SkyHiGh-IO fikk den oppetiden de krevde. Vi ble inspirert av spiralmodellen ettersom hver sprint besto av en fase med innhenting av informasjon, en fase med utvikling, og en fase med testing/utrulling.
Med utgangspunkt i Henrik Knibergs bok “Scrum and XP from the trenches” [35]valgte vi ut en moderert versjon av Scrum som baserer seg på det vi anså som det mest aktuelle fra SCRUM.

- Product Backlog (Indexcards og taskboard)
- Sprint
- Demo
- Daily Scrum

Vi valgte å ikke ta med *Sprint Retrospectives*, da vi hadde en veldig kort utviklingsperiode i forhold til de resterende fasene i prosjektet, altså en måned mot de resterende fire. Det ble også lagt lite vekt på de ulike rollene. Vi valgte å ikke utnevne scrum master på grunnlag av at vi i hovedsak var en såpass liten gruppe og fattet beslutninger i fellesskap.

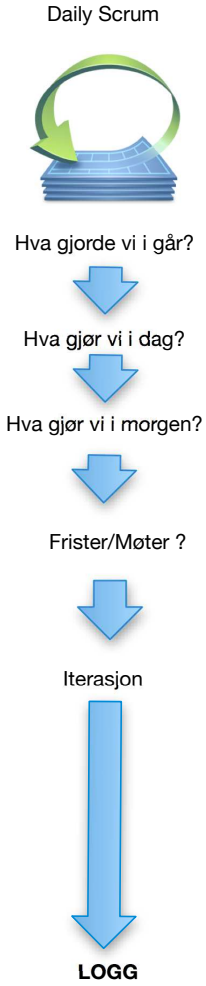
Når det kom til estimering vurderte vi *planning poker*, med tanke på at gruppen var uerfarene i bruk av OpenStack. Vi ønsket å gjøre dette enklest mulig, og valgte derfor bare å sette standard estimat i første sprint, for så å revurdere disse når vi hadde gjennomført en sprint.

Demo

Demo skulle gjennomføres etter hver sprint og presenteres hovedsaklig for veileder og oppdragsgiver. Målet med å benytte demo var å få gjort index-kort helt ferdig samt teste de når de var implementert. Å vise dette for et publikum skulle hjelpe på motivasjonen for å fullføre.

Daily scrum

Figur 14: Daily scrum



Vi bestemte å gjennomføre *daily scrum* gjennom hele prosjektet og ikke bare utviklingsfasen. Det ble ansett som en mulighet for å tilføre struktur til hele prosjektet. Ved å kartlegge fullførte oppgaver og tidsplan for resterende oppgaver i sprinten, ønsket vi å åpne for kommunikasjon slik at hele gruppen hadde kontroll på situasjonen.

Sprint

Lengde på sprint ble satt til en uke, start på mandag og demo på torsdag. Dette fordi vi ønsket å være så fleksible som mulig med tanke på hvor uerfarene vi var med både utviklingsmodell og rammeverkene vi skulle benytte. Utover i prosjektet ble utviklingsmetoden tilpasset en arbeidsflyt som i større og større grad konsentrerte seg om informasjonstilegning. I forprosjektet konkluderte vi med å benytte konseptet fra spiralmodellen om å jobbe i tre forskjellige faser:

- Innhenting av informasjon
- Utvikling

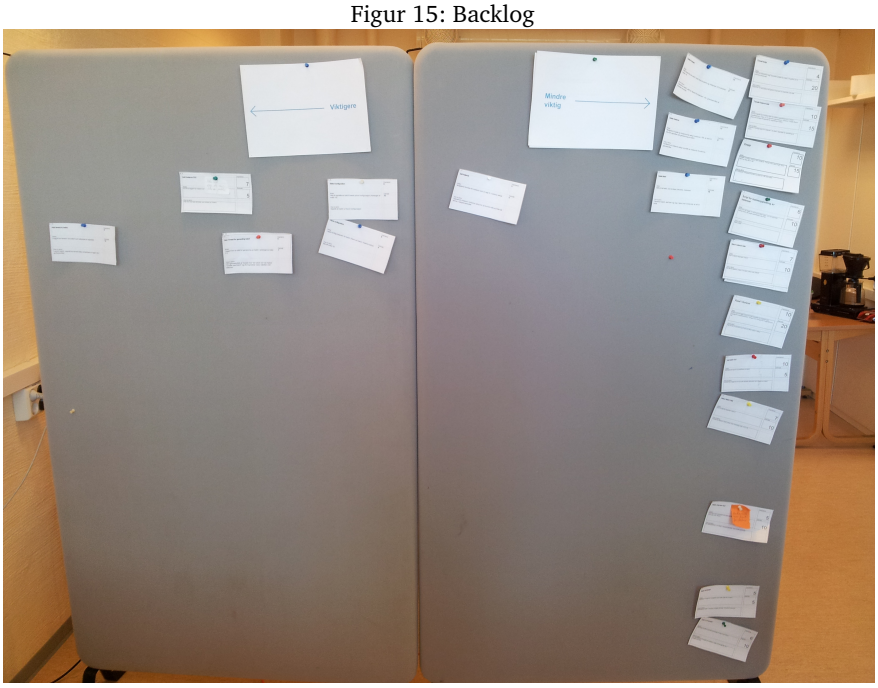
- Testing/implementering

Dette ble som følger av endringen i arbeidsflyten forskjøvet mer mot innhenting av informasjon.

Product backlog

Product Backlog ble utarbeidet første dag i sprint 1. Alle index cards ble satt på tavlen med en gang slik at gruppen fikk en oversikt. De kortene som var aktive med ble alltid satt til høyest prioriet på tavlen selv om de hadde lav overordnet prioritet. Når kort er ferdig utført blir de satt på en egen rad helt til høyre. På den måten har gruppen fremdeles oversikt og kan hente tilbake krav dersom det skulle være nødvendig. Dette var nødvendig når feil ble funnet. Feilene ble notert på det index card der den oppstod, og kortet plassert tilbake slik at de reflekterte mordi prioriten.

Figur 15: Backlog



Figur 15 viser tavlen med index cards slik den så ut i siste sprint. De to radene med kort til høyre er utførte kort.

Verktøy

Til å skrive kode benyttet vi Netbeans [11] som utviklingsmiljø. Det er en IDE som kommer både med python-plugin til siste utgivelse av netbeans, og en egen versjon designet for python. Ettersom vi alle var vant med denne IDE'en ble denne beslutningen fattet kjapt ettersom vi ikke kunne finne noen vital grunn til å bruke noe annet.

Bash-scriptene ble skrevet i vim [89]. Et script er såpass kort at man ikke trenger mer avansert programvare for å holde strukturen på dette.

Valget når det kom til versjonskontroll stod mellom *Git* og *SVN*. Hele gruppa hadde erfaring med begge parter fra før, men valget falt til slutt på GIT. Grunnen til dette var i hovedsak positive erfaringer med GIT. HiG har en egen SVN-server, men siden et privat "repository" ikke var essensielt for dette prosjektet falt valget naturlig på Git.
Rapporten ble skrevet ved hjelp av Google Documents, og formatert ved hjelp av Restructured Text og LaTeX. Google Documents ble valgt på grunnlag av muligheten til samskriving, samt gode erfaringer fra tidligere.

Arbeidsflyt

For å teste kildekode måtte kode dyttes over på serveren som kjørte Horizon. Versjonskontroll var alltid med fra starten, men arbeidsflyten var ineffektiv. Fremgangsmåten var å først legge til endringer lokalt på sin egen maskin, deretter måtte man godkjenne disse med ett endringsvarsel, etterfulgt av godkjenning av endring. Til slutt måtte man ved bruk av kommandolinjen laste ned endringene på serveren som kjørte Horizon, for så å teste koden.

For å effektivisere denne løsningen benyttet vi Dropbox [14] til å synkronisere filer mellom våre lokale maskiner og serveren. Dette effektiviserte arbeidsflyten drastisk, og siden Dropbox håndterer konflikter mellom filer som er endret vurderte gruppen dette som en valid løsning. Det viste seg desverre å være ustabilt til tider da konflikter ikke alltid ble løst, slik at vi gikk tilbake til den gamle arbeidsflyten for å være på den sikre siden.

Innformasjonstilegning

Prosessen vår for informasjonstilegning bestod først og fremst av å lese dokumentasjon. OpenStack tilbyr dette på deres hjemmesider [66] for alle modulene, basert på bidrag fra open-source miljøet. Dokumentasjon for Django [20] har også spilt en sentral rolle, på lik linje med dokumentasjon for Python [22]. I tillegg til benyttet vi kildekode for å se hvordan det fungerte i praksis, samt fikk tilgang til noe dokumentasjon ved hjelp av denne.

Informasjonstilegning ble en større del av iterasjonene enn hva gruppen hadde tenkt i utgangspunktet. Vi opplevde at dette kunne kreve opp til to dager i løpet av hver iterasjon, noe som er det dobbelte av estimatet. Mye av årsaken til dette var at vi måtte bruke mer tid på å tolke kildekode enn først antatt. Tolkning av kildekode ble benyttet der dokumentasjonen ikke var tilstrekkelig, og prosessen for dette er beskrevet under.
For å kartlegge arkitekturen til en horizon-modul undersøkte vi kildekode nøye for så å reprodusere funksjonalitet i andre scenarioer. En prosesss som denne ble utført på lokale installasjoner ved bruk av Devstack.
Et typisk eksempel er måten vi opprettet et panel på. Ved å kopiere kildekode for et eksisterende panel, endre navn, for så å strippe det ned funksjon for funksjon kunne vi kartlegge modulene ved hjelp av klassediagrammet i 4.3.1. Beskrivelsen av diagrammet er et direkte resultat av disse prosessene.

5.5.3 Problemer

Autentisering

En problematikk som vi støtte på midveis i utviklingen var et problem som omhandlet autentisering. OpenStack er på overordnet vis strukturert slik at en bruker ikke skal kunne utføre noen handlinger på vegne av en annen bruker. Dette var da en funksjonalitet vi var avhengig av i forbindelse med Batch Setup for å kunne opprette instanser, security groups og nøkkelpar på vegne av andre brukere. For å løse dette problemet hadde vi to umiddelbare alternativer:

- Å endre modulen for autentisering til å ta ett unntakstilfelle

eller

- Re-autentisere administrator for hver tenant ved hjelp av et tredjeparts-script.

Etter å ha undersøkt hvordan man kunne endre modulen for autentisering som vi anså som det beste alternativet, viste det seg at dette krevde for mye tid. Autentiseringsmodulen benyttet validering av session-tokens [56] på en måte som ikke lot oss manipulere disse uten å utføre store endringer på selve keystone, og vi fant det lite hensiktsmessig å endre koden til systemets grunnfunksjonalitet.
Derfor valgte vi til slutt å skrive et tredjeparts script i Bash, *spawn.sh*, som tok flavor, image, navnet på maskinen samt autentiseringsfilen for gitt tenant og opprettet de nødvendige security groups, nøkkelpar og instanser. Dette scriptet er en nedstrippet, automatisert versjon av det interaktive scriptet som ble utviklet for å få samme funksjonalitet på kommandolinjen som man får i vår nye modul i Horizon, og ligger i vedlegg H.

Resultat

Etter siste scrum var utført satt vi igjen med følgende fullførte krav:

Krav	Utførte krav
Opprett batch	Ja
Last opp image	Nei
Start instans basert på opplastet image	Nei
Endre batch	Nei
Slett batch	Ja
Opprette batch fra lagret konfigurasjon	Ja
Slett instans fra tenant i batch	Ja
Lagre batch-konfigurasjon	Ja

Tabell 15: Resultat etter utførte Scrum-sprinter

Laste Opp image/Start instans basert på opplastet image

Dette kravet ble i starten nedprioritert for senere iterasjoner tidlig i utviklingen, da funksjonalitet som eksisterte i MLN var mest kritisk og måtte tilfredsstilles. Denne funksjonaliteten er også tilgjengelig via kommandolinjeklienten til glance, som har støtte for image fra VmWare og VirtualBox. Disse to kravene var de som hovedsaklig måtte lide av problemene nevnt i 5.5.2. De ble ikke fullført, og er det som hovedsaklig gjenstår når oppgaven leveres. Dette innebærer altså at brukere ikke har mulighet å laste opp eller starte image ved hjelp av webgrensesnittet.

Opprett Batch

Dette var det mest omfattende og kritiske av alle kravene, og hadde dermed høyest prioritet og ble utført først. Sekvensdiagrammet for use-caset som tok for seg dette kravet var relativt komplekst, og krevde at gruppen hadde innsyn i Keystone, Nova og Horizon. Her opplevde gruppen også størst problemer, spesielt i forhold til autentisering som står beskrevet i 5.5.3. Kravet krevde til slutt tre sprinter fra start til slutt, noe som var langt over estimert tid.

Edit Batch

Dette use-caset ble ikke fullført før siste sprint hadde endt. Estimert tid var 10 timer og gruppen brukte 25 timer på forsøket. Fremgangsmåten vi benyttet for å utføre dette baserte seg på å implementere lignende funksjonalitet fra et annet panel. Problemene oppstod ved validering av objektene som skulle uthentes fra API'et, for å vise informasjonen om en enkelt batch. Django modulene tillot ikke å benytte ID'er på samme måte som i panelet vi tok utgangspunkt i. Prosessen for å debugge dette bestod av å lete etter dokumentasjon innad i kildekoden samt teste lignende funksjonalitet i de lokale testmiljøene vi hadde på egne maskiner. Denne prosessen dro ut den resterende utviklingstiden vi hadde igjen, og etter endt sprint konkluderte gruppen med å ikke fortsette på denne funksjonaliteten. Årsaken til dette var at funksjonaliteten for å kunne redigere en enkelt batch delvis eksisterte i andre paneler, og at dette ikke var essensielt for leveranse av systemet.
Gruppen så det som fordelaktig å gå over til neste steg i prosjektet i stedet for å bruke mer tid på dette Use Caset.batch

6 Driftsrutiner

I henhold til oppgavebeskrivelsen utarbeides denne driftsrutinen, som har sin funksjon i å beskrive aktiviteter under den daglige driften av systemet for å ivareta ytelse, pålitelighet, tilgjengelighet og sikkerhet. Dersom ikke annet er spesifisert er det administrator som er ansvarlig for å utføre de ulike aktivitetene.
Driftsrutinen tar for seg de aktiviteter som må/bør gjennomføres, samt hendelser som oppstår under den daglige driften. Dersom denne driftsrutinen etterleves vil ønsket utfall være at systemet fungerer tilnærmet optimalt.

6.1 Installasjon av batch-setup

Dette er en veiledning for implementasjon av Batch Setup i Horizon. Batch Setup er panelet som ble utviklet i forbindelse med SkyHiGh-prosjektet, og vil legge til funksjonalitet som utfører masseoppretting av prosjekter, brukere og instanser i OpenStack.

Følgende data kreves for å gjennomføre denne veiledningen:

- Batch Setup //Templates
- Batch Setup //Hovedfiler
- dash.sql

Dette finner du i ZIP-filen vedlagt Bacheloroppgaven SkyHiGh, og i kildekoden på `https://github.com/hwestman/Horizon-Skyhigh`. Navnet på filen er batch_setup.zip.

Panelet består av tre komponenter og en endring som må implementeres på følgende måte:

1. Mappen Batch Setup (Hovedfiler) må ligge under følgende sti:
   `/usr/share/openstack-dashboard/horizon/dashboards/syspanel`
2. Mappen Batch Setup (Templates) må ligge under følgende sti:
   `/usr/share/openstack-dashboard/horizon/dashboards/syspanel/templates/syspanel`
3. Den vedlagte sql-filen `dash.sql` i vedlegg H må importeres i mysql.
4. Følgende endringer må gjøres i eksisterende kode:
   I filen `/usr/share/openstack-dashboard/horizon/dashboards/syspanel/dashboard.py` må linjen som inneholder følgende:

   ```
   panels = {_("System Panel"): ('overview', 'instances', 'services','flavors',
   'images', 'projects','users','quotas',)}
   ```

   editeres til

   ```
   panels = {_("System Panel"): (‘batch_setup','overview', 'instances',
   'services','flavors', 'images', 'projects','users','quotas',)}
   ```

6.2 Oppdatering av programvare og operativsystem

Oppdatering av operativsystemet og installert programvare bør utføres i henhold til gjeldende retningslinjer som administrator følger i dag. Det anbefales imidlertidig at oppdateringer kjøres hvertfall en gang i måneden, og kritiske sikkerhetsoppdateringer installeres etter hvert som de blir sluppet. Oppdatering av OpenStack gjøres på lik linje som med vanlige oppdateringer, men hvilke endringer som oppdateringene til OpenStack medfører må sjekkes før oppdateringen blir utført. For å sikre at en OpenStack-oppdatering ikke bryter sammen, kan den kjøres i et test-system. Dette kan for eksempel være en egen virtuell maskin som er satt opp likt som produksjonssystemet.

6.3 Oppgradering av programvare og operativsystem

Oppgraderinger av OpenStack til ny versjon anbefales ikke uten god planlegging. Slike oppgraderinger bør utføres før/etter et semester er startet/avsluttet, da en ny versjon av OpenStack ofte fører med seg større endringer som kan påvirke gjeldende konfigurasjon og script. Eksempler her er endringer av kommandoer fra Diablo til Essex (fra `keystone` til `keystone-manage`) og andre restruktureringer i rammeverket. Slike endringer kan føre til at egenutviklede script ikke virker som ønsket, og at det egenproduserte panelet i web-grensesnittet ikke oppfører seg som planlagt.

6.4 Sikkerhetskopiering

Alle konfigurasjonsfiler bør innlemmes i daglig backup. Med konfigurasjonsfiler menes her alle `.conf` og `.ini-filer` i `/etc/nova/`, `/etc/glance`, `/etc/keystone`, og hele `/usr/share/openstack-dashboard/horizon/dashboards/syspanel/batch_setup/`. I tillegg bør også databasene tas backup av, samt de virtuelle maskinene. De virtuelle maskinene kan tas backup av ved hjelp av snapshot-funksjonen i OpenStack.

6.5 Hendelseshåndtering

Når uforutsette hendelser inntreffer er loggfilene i `/var/log/` et godt utgangspunkt for å starte feilsøking. I filen `/etc/nova/nova.conf` kan man sette `-verbose=true` slik at man får så mye info som mulig i loggen. Relevante loggfiler finnes i følgende mapper:

- /var/log/glance/
- /var/log/keystone/
- /var/log/libvirt/
- /var/log/mysq/
- /var/log/nova/
- /var/log/rabbitmq/

Ved feilsøking kan kommandoen

```
tail -f <loggfil>
```

være meget nyttig, da kan man følge med i loggfilen samtidig som den oppdateres.
Søk på internett samt dokumentasjonen på `http://docs.openstack.org` gir ikke bestandig resultater, men en god ressurs er IRC-kanalen #OpenStack på freenode-nettverket.

Her kommer man lett i kontakt med de som utvikler OpenStack, og de har som oftest svaret eller en god formening om hva som kan være feil.

6.6 Daglig drift og vedlikehold

Bruk av systemet via kommandolinjen kan være tungvint, særlig i forbindelse med brukerhåndtering og konfigurasjonsendring. Kommandoene er lange, og baserer seg ofte på forskjellige ID'er som typisk er en 32 tegn lang hexadesimal streng. Eksempelvis, dersom man skal knytte en eksisterende bruker til et eksisterende tenant blir man tvunget til å taste inn tre slike ID'er: Bruker-id, tenant-id og id til rollen som brukeren skal ha i gitt tenant. Med andre ord må du notere ned disse, etter å ha funnet dem i de respektive listene. For å gjøre dette litt enklere laget vi et script som leter frem ID'er. Dermed kan nevnte eksempel løses med én kommando i stedet for fire operasjoner. Scriptet er gjengitt her:

```
#keystone-get-id.sh
#!/bin/bash
ARGUMENTS=2 # First is type of id to search for. Second is the [user/tenant/role] name

if [ $# -ne $ARGUMENTS ]; then # Wrong amount of arguments. Print usage and quit
    echo "Usage: keystone-get-id <user|role|tenant> <name>"
    exit 1
fi

if [ $1 == "user" ] || [ $1 == "role" ] || [ $1 == "tenant" ]; then
    RES=$(keystone $1-list | grep -iw $2 | awk '{ print $2 }') # Get id
    if [ -z $RES ]; then # Not found, print error message and quit
            echo "No $1 with name $2"
                exit 1
        fi
    echo $RES
else
    echo "Unknown type $1"
    echo "Usage: keystone-get-id <user|role|tenant> <name>"
    exit 1
fi
```

For å løse nevnte eksempel blir kommandoen:

```
keystone user-role-add --user $(keystone-get-id user johndoe) \
--tenant_id $(keystone-get-id tenant foo) --role $(keystone-get-id role member)
```

Fremdeles en lang kommando, men mye enklere!

Når det gjelder konfigurasjonsendring blir dette fort en meget tung oppgave, siden det er såpass mange forskjellige moduler og tjenester på et potensielt stort antall forskjellige servere. Nova fungerer som kjent slik at alle tjenester benytter samme konfigurasjonsfil på alle servere. Derfor har vi også laget et script for å rulle ut konfigurasjonsfilen samt et

script for å restarte tjenester på alle noder. Se vedlegg H for disse scriptene.
Løsningen blir levert med et interaktivt skript som lar administrator opprette batcher via kommandolinjen. Scriptet er kalt `batch_setup.sh` og er også vedlagt i vedlegg H.

6.7 Opplasting av image

Opplasting av et image til OpenStack via Glance har vist seg å være meget tids- og ressurskrevende, og bruker gjerne opp mot 100% av cpu-kraften som er tilgjengelig. Et Linux-image er gjerne på ca 500MB, men et Windows-image typisk er 10-20 GB. Å laste opp et slik tar ca. 28 minutter. Opplasting av image bør gjøres på tidspunkter hvor bruken av systemet er lav, slik at andre brukere ikke opplever tregheter. Se vedlegg B for testresultater.

6.8 Logging/overvåkning

Å overvåke at alle tjenester som må kjøre for at OpenStack skal virke faktisk kjører er essensielt.
Tjenester som må kjøre for at OpenStack skal kjøre på controlleren er: nova-api, nova-cert, nova-consoleauth, nova-network, nova-scheduler, glance-api, glance-registry, keystone-all, rabbitmq-server, mysqld, libvirtd, kvm-irqfd-clean, (iscisid), nova-scheduler, apache2 (for Horizon), glance-api, glance-registry, keystone, qemu-kvm.
Tjenester som må kjøres på compute-node er: nova-cert, nova-compute, nova-network, nova-api, libvirtd.
Ved en eventuell feil, sjekk derfor om disse tjenestene kjører, for så å sjekke loggfiler i `/var/log/` etter eventuelle feilmeldinger.
Overvåkningsprogrammer som Nagios [16] kan være et nyttig verktøy, og det er allerede utviklet noen plugins for OpenStack [30].

6.9 Feilhåndtering

Felles for de fleste av disse feilene, er at feilmeldingen ikke alltid er veldig beskrivende. Dermed kan det være vanskelig å fange opp hva som faktisk skjedde.

Sletting av tenant

I forbindelse med sletting av en tenant, kan det oppstå en del interessante feil:

- Dersom man prøver å slette en tenant som har brukere tilknyttet seg, vil man få feilmeldinger både i kommandolinja og i webgrensesnittet. For å komme rundt dette må man først fjerne alle roller som brukere har knyttet til tenant som skal slettes, for deretter å slette tenant.
- Hvis tenant har kjørende instanser vil disse fortsette å kjøre når tenant slettes. Sideeffekten av dette er at listingen av kjørende instanser vil knekke både i webgrensesnittet og i kommandolinjen. I tillegg vil ingen brukere (ei heller administrator) ha tilgang til å slette disse maskinen ved hjelp av OpenStack. Løsningen på dette er å logge seg inn på den aktuelle compute-noden, starte *virsh* som er konsoll-grensesnittet til KVM, og slette maskinen derfra med kommandoen `virsh destroy <domain-name>`. Her må man merke seg at domain-name ikke er den samme ID'en som maskinen har i nova. Når instansen er slettet må man manuelt rydde vekk alle spor etter den i novas

database.

- Når et tenant slettes, frigis ikke nettverket den var knyttet til, og dette må da gjøres manuelt via kommandoen `nova-manage project scrub <tenant_id>`. Om dette ikke gjøres vil det til slutt gå tom for nettverk.

Sletting av bruker

En bruker som er tilknyttet en tenant med en eller flere roller kan ikke fjernes før disse rollene er fjernet.

Navngiving

Ingen av modulene takler norske tegn i navngiving. F.eks navnet på en tenant eller navnet på en instans. Dersom et norsk tegn blir registrert må man endre navnet manuelt i databasen. Alle opplistinger vil knekke før det er ryddet opp i.

Masseoppretting

Dersom man oppretter for mange instanser i en og samme operasjon så kan en del av disse instansene ende opp i tilstanden `error scheduling`. Dette er enten en følge av at nova-scheduler ikke taklet datamengden å sluttet midlertidig å fungere, eller at ingen noder passerte filterne slik at ingen noder lenger er aktuelle for å starte disse instansene. Teknisk sett er ikke dette en feil i systemet men en begrensing med direkte tilknytning til maskinvare.

Floating IP

Når en floating IP assosieres med en instans blir ikke IP-adressen listet opp sammen med instansen, men er allikevel funksjonell. Dersom man kjører kommandoen på nytt vil kommandolinjen returnere en feilmelding, men IP-adressen vil nå listes opp.

Oppretting av instanser

Når instanser får statusen `error scheduling`, betyr det som oftes at ingen av nodene gikk gjennom filterene til nova-scheduler slik at man endte opp uten aktuelle kandidater til å starte instansen.

7 Diskusjon

I dette avsnittet vil gruppen ta for seg en diskusjon av hvorvidt OpenStack er modent for å løse vår problemstilling, og om det vil være hensiktsmessig for andre utdanningsinstitusjoner eller bedrifter med lignende scenarioer.

7.1 OpenStack - modent nok?

As a service provider, today, I don't think you can just take nova of the shelf and use it without doing any serious development on it. - Ken Pepple, Internap (først i verden til å lansere en offentlig sky basert på OpenStack) [75].
Dette sitatet er hentet fra en pressekonferanse fra januar 2012 med Ken Pepple fra Internap, Ray O'Brien fra NASA og Rodrigo Bezanquen fra MercadoLibre der de snakker om sine implentasjoner av OpenStack i sine respektive firmaer. Utsagnet kommer fra Pepples forklaring om detaljer rundt Internaps løsning.

Påstanden er rimelig bastant, men man må merke seg at Pepple her snakker om å være en kommersiell tjenestetilbyder. Med dette tatt i betraktning så har han nok rett, selv etter at Essex ble sluppet i etterkant av denne pressekonferansen. For kommersielt bruk trenger man moduler for mer avansert autentisering og identifisering, samt faktureringsystemer for kunders ressursbruk.

For en intern skyløsning stiller situasjonen seg noe annerledes. Ingen av brukerene skal faktureres, og det vil være naturlig å koble systemet sammen med allerede eksisterende autentiserings- og identifiseringstjenester som f.eks Active Directory [44]. Man vil i større grad kunne ta rammeverket ned fra hylla, konfigurere det og sette det i produksjon. Måten OpenStack presenterer seg mot både sluttbrukere og administratorer fremstår pent, oversiktlig og selvforklarende. I tillegg er den største gevinsten skalerbarheten OpenStack gir. Kraften av at man med enkle operasjoner kan utvide tilgjengelig maskinkraft med nye noder vil være et sjumilssteg sammenlignet med løsningen på tidligere ytelsesproblemer. Selvfølgelig fordrer dette tilgang på, og økonomi til nye servere, men for vår oppdragsgiver er ikke dette et problem.

Selv om det fremstår enkelt har vi erfart at det ikke alltid er det. Ved prosjektstart var dokumentasjonen for administrasjon mangelfull og lite grundig. Problemstillinger som oppsto var vanskelige å finne ut av på grunn av lite informative feilmeldinger samt dårlig dokumentasjon for feilhåndtering. Kanalene for support er mange og vidt spredt, slik at det kan være vanskelig å vite hvor man skal stille spørsmål. Det finnes en mailingliste, et diskusjonsforum, en launchpad-side og hele tre forskjellige IRC-kanaler. I tillegg er dette et meget ferskt prosjekt som til og med for utviklere fremdeles er nytt og spennende. Det faktum gir en ekstra utfordring når det kommer til å sette det i produksjon. Et OpenSource-prosjekt er alltid i løpende utvikling, og vi har erfart at det ofte går ut over

dokumenteringen. Plutselig vil man komme opp i situasjoner der man ikke ser andre muligheter enn å måtte gjøre større oppgraderinger på systemet. Bare på de fem månedene dette prosjektet har vart har vi måttet oppgradere operativsystemet tre ganger, og hele programvaren én gang for å komme til en nogenlunde stabil løsning. Dette bør stå som et tankekors når man vurderer å sette OpenStack i produksjon.

Dersom man effektivt skal kunne benytte seg av en skyløsning som rammeverk for virtuelle maskiner i et utdanningsmiljø trengs det modifikasjoner. Når man tar OpenStack ned fra hylla finnes det ingen lettvin måte å rulle ut et oppsett på en virtuell datalab. Dette er dog ikke eksklusivt for OpenStack. Uansett hvilket skyrammeverk man velger vil man måtte lage egne script for å få denne funksjonaliteten.

7.2 Alternativer

Alternativene til OpenStack er mange, og det finnes både properitære utgaver og de som er basert på åpen kildekode. De største properitære er VMWare ESXi [28] og Citrix CloudApp [80], og av de som er basert på åpen kildekode er Eucalyptus [81] og OpenNebula [40] de største. En dyptgående analyse av alle disse vil være et prosjekt i seg selv, og en tidligere bacheloroppgave [37] har gjort en ytelsestesting av ESXi og Citrix XenServer. Den konkluderer med at XenServer yter gjennomsnittelig best av disse.

De andre OpenSource alternativene viser seg å være litt mer komplekse i oppsett enn hva man ønsker i vårt tilfelle. Tar man f.eks Eucalyptus har man der fem forskjellige kontroller-noder som det anbefales å kjøre på separate maskiner [18].

7.2.1 Valg mellom properitær og åpen programvare

Diskusjonen om valget mellom properitær og åpen programvare er evig. Fra en systemadministrators synsvinkel vil support og kostnader være vesentlige faktorer. Er det dyrest å bruke egne konsulenttimer på å drive implementasjon, utvikling og feilsøking på åpen programvare, eller å bruke penger på properiær programvare med support inkludert i prisen? I samtaler med IT-Leder Stian Husemoen [25] ved HiG kom vi frem til at det ikke er noen særlig tendens i hva man velger. Det må gjøres kalkuleringer i hvert tilfelle, men hos dem velges oftest åpne løsninger eller egenutvikling. Properitære løsninger blir bare valgt dersom de åpne alternativene ikke fungerer eller at egenutvikling blir for dyrt. Properitær programvare er ofte også lisensbelagt, noe som gir en årlig kontinuerlig kostnad. Velger man en løning basert på åpen kildekode er man stort sett prisgitt prosjektets dokumentasjon. Det er et absolutt krav at denne er tilfredsstillende, ellers vil det gå med mengder av arbeidstimer på å studere kode, som gjerne også er dårlig kommentert.

7.3 Drift

Vår oppdragsgiver står selv for driftsansvaret for maskinen som utgjør denne OpenStack implementasjonen. IT-tjenesten ved HiG har som eneste ansvar å tilby fysisk lokasjon, strøm og nettverk. Dermed vil ikke implementasjon av skyen i eksisterende driftsrutiner være en problemstilling for andre enn oppdragsgiver. Driftsrutinene er selvdefinerte.

8 Avslutning

OpenStack er et veldig ferskt prosjekt, men samtidig også ganske voksent. Tilhengerene av det blir stadig flere, og det er ikke uten grunn. Det største styrken er modulariseringen og den utrolig enkle skalerbarheten. Muligheten for å utvide maskinkraften med nye noder ved å installere et lite antall pakker samt kopiere en konfigurasjonsfil er tilnærmet enestående. Bare i løpet av vår prosjektperiode har stabiliteten og dokumentasjonen forbedret seg, og det er ingen grunn til å tro at denne utviklingen kommer til å endre kurs i fremtiden.

For utdanningsinstitusjoner som ønsker et enkelt, ferdig rammeverk for virtuelle datalab'er, egner OpenStack seg i utgangspunktet ikke. Ene og alene fordi funksjonaliteten for å automatisere en utrulling ikke eksisterer. OpenStacks formål er å tilby en enkel skalerbar privat IaaS løsning der enkeltbrukere kan få tilgang til virtuelle maskiner. Dette gjør at man er bundet til mye manuell og repetitiv konfigurasjon dersom man skal sette opp forskjellige virtuelle lab-scenarioer. Imdlertid har man alle muligheter til å utvide funksjonaliteten selv, siden OpenStack er basert på åpen kildekode. Denne muligheten har vi benyttet, og utviklet en modul der man kan sette i gang masseoppretting av prosjekter med få tastetrykk. Med denne funksjonaliteten på plass har man i realiteten en fungerende løsning.

Man må ta i betraktning de feil og mangler som finnes i dokumentasjonen, særlig angående nettverksoppsett, konfigurasjon av hypervisor og konfigurasjon av keystone. Det forventes dog, basert på våre erfaringer, at disse problemstillingene vil forsvinne i fremtiden.

8.1 Resultater

Hovedoppgaven for SkyHiGh ADM var å sørge for at administratorbrukerne kan tilby fleksible lab-løsninger med virtuelle maskiner i sine emner gjennom et brukervennlig grensesnitt, samt prototype implementasjon og driftsplan. Gjennom installasjon av OpenStack, utvikling av webgrensesnittet og driftsplan har vi fullført de aller fleste kravene som var spesifisert.

Kravspesifikasjonen

Når vi skal evaluere kravene som ble stilt mot løsningen som blir presentert ser vi at vi kunne ikke oppfylle alle kravene. Som vist i avsnittet 5.5 ang. utvikling mangler det tre krav som ikke ble tilfredsstilt. Gruppen anser ikke dette som en vital mangel. Dette fordi funksjonaliteten eksisterer for administrator via kommandolinjen og er mulig å gjennomføre. Foruten om dette føler gruppen at vi fikk utført det som var spesifisert og er av tilstrekkelig kvalitet for produksjon.

Alternativer, muligheter og valg underveis

Bytte av hypervisor

Underveis sto vi ved en del veiskiller. Det første av disse var å bytte hypervisor fra XEN til KVM. I etterkant ser vi at det ble brukt for lang tid på å komme til denne beslutningen. Vi var for opptatte av å få dette til fordi det var et krav fra oppdragsgiver. Så mangelfull som dokumentasjonen var på det tidspunktet burde vi tatt denne beslutningen tidligere.

Endring av operativsystem

I løpet av perioden byttet vi operativsystem tre ganger. Først fra Debian Wheezy til Ubuntu 11.10, og derfra til Ubuntu 12.04 LTS. Det første byttet tok for lang tid å beslutte, men dette var mye fordi vi ikke forsto hvorfor ting ikke fungerte på Debian når det fungerte på Ubuntu. Vi brukte en del tid på å finne ut av dette, vel vitende om at det hindret utviklingsdelen av prosjektet. Det andre byttet skjedde som en naturlig følge av at vi oppgraderte OpenStack. Alle byttene viste seg å være til prosjektets beste, og en viktig suksessfaktor, selv om vi i ettertid nok skulle ønske at det ble brukt mindre tid det.

Valget om oppgradering til Essex

Dette var et krevende valg som det oppstod mest usikkerhet rundt. Vi hadde på forhånd avgjort at det skulle ikke oppgraderes fra Diablo for å beholde sikkerheten ved en stabil versjon. Det å gå over til en Release Candidate med flere kjente feil og mangler førte med seg flere usikkerhetsmomenter og var således ikke en lett avgjørelse å ta.

Vi er svært fornøyd med forhåndsreglene som ble tatt i forkant av denne oppgraderingen og i ettertid ser vi at risikoen for feil ble betydelig redusert av grepene vi gjorde. Her lærte vi av forhåndsreglene vi hadde tatt og ikke en feil som ble gjort, noe som var en svært hyggelig erfaring.

Scrum

Når det gjelder hvordan vi har benyttet SCRUM ser vi at iterasjonene burde vært justert tidlig i utviklingen. Dette var en problemstilling vi var inne på tidlig, men vi valgte å ikke justere opp antall dager i iterasjonene. Grunnen til dette var at vi forventet at utviklingen ville skyte fart så snart vi hadde fullført de første iterasjonene og ble tilvent prosessen. Dette viste seg å ikke være tilfelle og vi burde gjort dette etter første iterasjon, så snart vi fikk intrykk av hvor mye jobb som lå i de forskjellige index-kortene. Når vi ser på dette i ettertid ville vi foreslått iterasjoner på to uker i stedet for en. Vi tror at dette ville vært fleksibelt nok for prosjektet og samtidig gitt oss den arbeidstiden som var krevet. Med denne muligheten ville det også vært bedre struktur i forhold til flytting av iterasjoner. Det ble rett og slett for enkelt å flytte en iterasjon frem i tid, når vi så at det ble problematisk å nå tidsfrister.

8.2 Forslag til videre arbeid

Siden OpenStack utvikles i et rasende tempo vil vår egenutviklede modul trenge oppdateringer og tilpasninger etterhvert som flere av OpenStacks nye moduler blir implementer som kjernefunksjonalitet. For eksempel vil Quantum bli en kjernemodul i neste utgave, og det vil være naturlig å legge inn konfigurasjon for denne i vår modul.

Å sette opp en FEIDE [87]-integrasjon vil også være naturlig dersom systemet skal tilbys som en offentlig tjeneste til alle studenter. Det er per nå ingen mulighet for selvhjulpen brukerregistrering, og en FEIDE-integrasjon vil være i tråd med de aller fleste andre offentlige tjenester ved HiG.

De kravene som ikke ble tilfredsstilt av dette prosjektet bør også fullføres. Dersom skyen skal kunne tilby maskinkraft som en tjeneste for studenter og ansatte som ønsker å teste en allerede ferdig oppsatt virtuell maskin fra egen PC i et miljø med bedre ressurser, må det være mulighet for å laste opp maskiner ved hjelp av webgrensesnittet.

8.3 Evaluering av gruppens arbeid

8.3.1 Organisering

Organiseringen av gruppen er beskrevet i vedlegg A. Denne organiseringen har fungert godt, og alle problemstillinger har blitt løst uten nevneverdige problemer. Vi valgte å ikke la gruppelederververvet rullere, og føler at det var en grei løsning siden vi uansett ikke har hatt noe særlig behov for å ta sjefsavgjørelser ved uenighet.

8.3.2 Arbeidsfordeling

Siden gruppen består av to driftsstudenter og en programvareutvikler fant vi det natulig at vår utviklingsstudent sto for det meste av utviklingen, og at driftsstudentene foretok installasjon, konfigurasjon og infrastruktur. Siden veien frem til et fungerende stabilt oppsett viste seg å bli mye lenger enn vårt estimat, og at dette var en forutsetning for å starte utviklingen i særlig grad, kom vi etterhvert frem til at alle måtte ta sin del av utviklingsjobben. Dette fungerte bra, selv om læringskurven var litt bratt siden Python og Django var upløyd mark for alle tre.

8.3.3 Prosjekt som arbeidsform

Etter fem semestere på HiG har vi god erfaring med å bruke prosjekt som arbeidsform. Allikevel var et såpass omfattende prosjekt noe nytt for alle, og i starten følte vi oss nok ikke helt klare. Nå som prosjektet er ferdig ser vi at arbeidsprosessen har vært veldig lærerik. Å ha et arbeidspress fra en oppdragsgiver som skal ha et produkt har gitt oss en god forsmak på hva som venter i arbeidslivet. Man ser i etterkant at arbeidet med initiell planlegging og tidsestimering nesten ikke kan gjøres grundig nok, men at man uansett vil støte på avvik fra den opprinnelige planen.

8.3.4 Subjektiv opplevelse av bacheloroppgven

Å jobbe med bacheloroppgaven har gitt oss både store gleder, og like store nedturer. Gjennom dette har vi hevet kompetansenivået vårt både innen serverdrift og utvikling, der vi har satt oss inn i nye programmeringsspråk og bygget infrastruktur fra bunnen. I denne prosessen har vi både opplevd gleden av å lykkes og skuffelsen ved å stadig tøte på uforutsette problemer. Det ble fort besluttet at vi måtte være løsningsorienterte for å sørge for progresjon. Alt i alt føler vi at arbeidet med oppgaven har vært mest gøy, men til tider også meget frustrerende. Det har vært en god investering i tid der vi har lært masse og en fin avslutning på vårt studie ved HiG.

8.4 Konklusjon

Gjennom prosjektet har vi fått dyp innsikt i hvordan OpenStack fungerer, hvilken funksjonalitet som eksisterer, og hvilke mangler vi ser som eventuelt kan hindre å ta det i bruk for å realisere prosjektets mål.
Etter endt prosjekt, og alle punktene i dette kapitlet tatt i betraktning, konkluderes det med at OpenStack kan tas i bruk for å realisere en skalérbar intern nettsky for virtuelle datalaber. Det må her påpekes at OpenStack er et komplisert system hvor god kjennskap til systemet og god planlegging er en nødvendighet for et godt fungerende oppsett. Vår utvidelse av OpenStack minsker administratorens arbeidsbelastning, og gjør dermed både emneansvarlige og studenter mer selvstendige i arbeidet med de virtuelle laboratoriene. I tillegg til de funskjonelle målene, oppfylte vi også vårt eget ønske om å komme på OpenStack-kartet! (vedlegg F)

Bibliografi

[1] Cisco Networking Academy. Dhcp. In *Cisco CCNA Exploration 4.0 Accessing the WAN*. Cisco Networking Academy, 2009.

[2] Cisco Networking Academy. Inter-vlan routing. In *Cisco CCNA Exploration 4.0 LAN Switching & Wireless*. Cisco Networking Academy, 2009.

[3] Cisco Networking Academy. Vlans. In *Cisco CCNA Exploration 4.0 LAN Switching & Wireless*. Cisco Networking Academy, 2009.

[4] Pavel Alpeyev. Amazon.com said to have been used in sony attack. http://www.bloomberg.com/news/2011-05-13/sony-network-said-to-have-been-invaded-by-hackers-using-amazon-com-server.html, 2011.

[5] Amazon. Amazon elastic compute cloud api reference. http://docs.amazonwebservices.com/AWSEC2/latest/APIReference/Welcome.html?r=2056, 01.04.2012.

[6] Amazon. Amazon ec2 running ibm. http://aws.amazon.com/ibm/, 14.05.2012.

[7] Amazon. Amazon simple storage service (amazon s3). http://aws.amazon.com/s3/, 2012.

[8] Amazon. Elastic block store. http://aws.amazon.com/ebs/, 2012.

[9] Kyrre M. Begnum. Database- og applikasjonsdrift, emnebeskrivelse. http://www.hig.no/content/view/full/25638/language/nor-NO, 18.05.2012.

[10] Fabrice Bellard. Qemu web page. http://wiki.qemu.org/Main_Page, 2012.

[11] Oracle Corporation. Netbeans web page. http://netbeans.org/, 2012.

[12] DevStack. Devstack web page. http://www.devstack.org, 2012.

[13] Python Documentation. Python pep 8 - style guide for python code. http://www.python.org/dev/peps/pep-0008/, 18.05.2012.

[14] DropBox. Dropbox web page. https://www.dropbox.com/, 2012.

[15] Bucky Roberts (TheNewBoston Education). Python programming tutorial. http://www.youtube.com/watch?v=4Mf0h3HphEA&feature=list_related&playnext=1&list=SPEA1FEF17E1E5C0DA, 2009.

[16] Nagios Enterprises. Nagios web page. http://www.nagios.org/, 2012.

[17] Eirik Bakken Erik R. Grina. Skyhigh i/o. Technical report, Høgskolen i Gjøvik, 2012.

[18] Inc Eucalyptus Systems. Eucalyptus 2.0 architecture. http://open.eucalyptus.com/themes/eucalyptus/images/architecture-1.6.png, 2011.

[19] Django Software Foundation. Django coding style. https://docs.djangoproject.com/en/dev/internals/contributing/writing-code/coding-style/, 18.05.2012.

[20] Django Software Foundation. Django documentation. https://docs.djangoproject.com/en/1.4/, 2012.

[21] Django Software Foundation. Writing your first django app. https://docs.djangoproject.com/en/1.4/intro/tutorial01/, 2012.

[22] Python Software Foundation. Python v2.7.3 documentation. http://docs.python.org/, 2012.

[23] The Apache Software Foundation. Apache license version 2.0. http://www.apache.org/licenses/LICENSE-2.0.html, 2004.

[24] Erik Hjelmås. Systemadministrasjon, emnebeskrivelse. http://www.hig.no/index.php/content/view/full/21602/language/nor-NO, 18.05.2012.

[25] Stian Husemoen. Samtaler med it-leder ved hig, 14.05.2012.

[26] Citrix Systems Inc. Xen wiki, xenapi project. http://wiki.xensource.com/xenwiki/XenApi, 2012.

[27] Citrix Systems Inc. Xenserver web page. http://www.citrix.com/English/ps2/products/product.asp?contentID=683148, 2012.

[28] VMWare Inc. Vmware vsphere hypervisor esxi. http://www.vmware.com/products/vsphere-hypervisor/overview.html, 2012.

[29] Kevin Jackson. Twitter. https://twitter.com/#!/itarchitectkev/status/181695912391884800, 19.03.2012.

[30] jd. Nagios check keystone plugin. http://exchange.nagios.org/directory/Plugins/Software/check_keystone/details, 2012.

[31] Adam Johnson. Launchpad blueprint - refactor networking. https://blueprints.launchpad.net/nova/+spec/network-refactoring, 2011.

[32] Marius Jørgenrud. Her er brevet som forbyr google apps. http://www.digi.no/887985/her-er-brevet-som-forbyr-google-apps, 2011.

[33] Marius Jørgenrud. Narvik må svare for google apps. http://www.digi.no/873387/narvik-maa-svare-for-google-apps, 2011.

[34] Simon Kelley. dnsmasq web page. http://www.thekelleys.org.uk/dnsmasq/doc.html, 2012.

[35] Henrik Kniberg. *MScrum and XP from the Trenches*. InfoQ, 2007.

[36] Ringsaker kommune. Økt effektivitet med servervirtualisering. http://referanser.microsoft.no/windows-server/effektivt-med-server-virtualisering/, 2012.

[37] Øyvind Haugedal Kristoffer M. Elde. Ytelsestesting av virtualiseringsteknologier. Technical report, Høgskolen i Gjøvik, 2010.

[38] kvm. Kvm web page. http://www.linux-kvm.org/page/Main_Page, 2012.

[39] Ole Magnus H. Waaler Lars Erik Pedersen, Jon Arne W. Westgaard. Prosjekt skyhig(h). *HiG*, 2011.

[40] OpenNebula Project Leads. Open nebula web page. http://opennebula.org/, 2012.

[41] Mark Lutz. *Programming Python*. O'Reilly Media, fourth edition, 2010.

[42] ManagedIT. Managedit openstack setup scripts. https://github.com/managedit/openstack-setup, 2012.

[43] Kyle Mestery. Integrating vxlan in openstack quantum. http://blogs.cisco.com/openatcisco/integrating-vxlan-in-openstack-quantum/, 2012.

[44] Microsoft. Windows server 2008 r2 active directory overview. http://www.microsoft.com/en-us/server-cloud/windows-server/active-directory-overview.aspx, 2012.

[45] MLN. The mln project. http://mln.sourceforge.net/, 18.05.2012.

[46] Netfilter.org. The netfilter.org iptables project. http://www.netfilter.org/projects/iptables/index.html, 2012.

[47] OpenStack. Openstack community welcome guide. http://openstack.org/assets/welcome-guide/openstack-welcome-guide.pdf, 03.05.2012.

[48] OpenStack. Openstack contribution guide. http://horizon.openstack.org/contributing.html, 18.05.2012.

[49] OpenStack. Participating companies. http://openstack.org/community/companies/, 18.05.2012.

[50] OpenStack. Openstack packaging. http://wiki.openstack.org/Packaging/, 2011.

[51] OpenStack. Selecting a hypervisor, openstack compute administration manual (diablo). http://docs.openstack.org/diablo/openstack-compute/admin/content/selecting-a-hypervisor.html, 2011.

[52] OpenStack. Cla. http://wiki.openstack.org/CLA, 2012.

[53] OpenStack. Cloudpipe - per project vpns. http://nova.openstack.org/devref/cloudpipe.html, 2012.

[54] OpenStack. Creating a windows image. http://docs.openstack.org/trunk/openstack-compute/admin/content/creating-a-windows-image.html, 2012.

[55] OpenStack. Introduction to using xen, xcp and xenserver with openstack. http://docs.openstack.org/trunk/openstack-compute/admin/content/introduction-to-xen.html, 2012.

[56] OpenStack. Keystone middleware architecture. http://keystone.openstack.org/middlewarearchitecture.html, 2012.

[57] OpenStack. Launchpad openstack. https://launchpad.net/openstack, 2012.

[58] OpenStack. nova-scheduler workflow. http://docs.openstack.org/trunk/openstack-compute/admin/content/figures/filteringWorkflow1.png, 2012.

[59] OpenStack. Openstack at github. https://github.com/openstack, 2012.

[60] OpenStack. Openstack community map. http://openstack.org/community/, 2012.

[61] OpenStack. Openstack compute administration manual. http://docs.openstack.org/trunk/openstack-compute/admin/content/, 2012.

[62] OpenStack. Openstack compute administration manual, networking. http://docs.openstack.org/trunk/openstack-compute/admin/content/ch_networking.html, 2012.

[63] OpenStack. Openstack forums. http://forums.openstack.org/, 2012.

[64] OpenStack. Openstack install and deploy manual. http://docs.openstack.org/trunk/openstack-compute/install/content/, 2012.

[65] OpenStack. Openstack logisk arkitektur. http://docs.openstack.org/trunk/openstack-compute/admin/content/figures/nova-logical-arch-essex.jpg, 2012.

[66] OpenStack. Openstack manuals. http://docs.openstack.org, 2012.

[67] OpenStack. Openstack melange wiki. http://wiki.OpenStack.org/Melange, 2012.

[68] OpenStack. Openstack quantum administration guide. http://docs.openstack.org/trunk/openstack-network/admin/content/Preface-d1e71.html, 2012.

[69] OpenStack. Openstack wiki. http://wiki.openstack.org/, 2012.

[70] OpenStack. Using the glance cli tool. http://glance.openstack.org/glance.html, 2012.

[71] Stuart Kemp Paul Rubin, David MacKenzie. dd man page. http://linux.die.net/man/1/dd, 2010.

[72] Lars Erik Pedersen. Error in how to create a windows image. https://bugs.launchpad.net/openstack-manuals/+bug/1000185, 2012.

[73] Lars Erik Pedersen. Nova client sends floating ip call to wrong node. https://bugs.launchpad.net/nova/diablo/+bug/953930, 2012.

[74] David Pritchard. Beginner's guide to python. http://wiki.python.org/moin/BeginnersGuide, 2012.

[75] Boris Renski. Openstack in production panel. http://vimeo.com/33982906, 2012.

[76] SQLAlchemy. Sqlalchemy - the database toolkit for python. http://www.sqlalchemy.org/, 2012.

[77] StackOps. Stackops web page. http://stackops.com, 2012.

[78] Omar Sultan. Digging deeper into vxlan. http://blogs.cisco.com/datacenter/digging-deeper-into-vxlan/, 2011.

[79] Omar Sultan. Introducing vxlan. http://blogs.cisco.com/datacenter/introducing-vxlan/, 2011.

[80] Citrix Systems. Citrix cloudapp. http://www.citrix.com/English/ps2/products/product.asp?contentID=2314749, 2012.

[81] Eucalyptus Systems. Eucalyptus open source cloud platform. http://open.eucalyptus.com/, 2012.

[82] Andrew S. Tanenbaum. *Modern Operating Systems*. Pearson Education, third edition, 2009.

[83] Morten K. Thomsen. Datatilsynet forbyder google apps i kommuner. http://www.version2.dk/artikel/datatilsynet-forbyder-google-apps-i-kommuner-15352, 2010.

[84] Andreas Thue. Sparer energi med virtualisering. http://www-05.ibm.com/no/newworld/2007/1/art8.html, 2007.

[85] Troy Torman. Openstack folsom summit: Melange overview. http://www.slideshare.net/troytoman/openstack-folsom-summit-melange-overview, 2012.

[86] Ubuntu. Ubuntu cloud images. http://cloud-images.ubuntu.com/, 2012.

[87] Uninett. Feide web page. http://www.feide.no/, 2012.

[88] Biblioteket ved Høgskolen i Gjøvik. Vancouver siteringsstil. http://www.hig.no/biblioteket/oppgaveskriving/vancouver, 2012.

[89] vimonline development. Vim web page. http://www.vim.org/, 2012.

[90] VMWare. Amqp model explained. http://www.rabbitmq.com/tutorials/amqp-concepts.html, 11.04.2012.

[91] Wikipedia. Cloud computing. http://en.wikipedia.org/wiki/Cloud_computing, 14.05.2012.

[92] Wikipedia. Regular expressions. http://en.wikipedia.org/wiki/Regular_expression, 14.05.2012.

[93] Lasse Øverlier. Ethical hacking and penetration testing, emnebeskrivelse. http://www.hig.no/content/view/full/21640/language/nor-NO, 18.05.2012.

A Prosjektplan

Dette vedlegget inneholder prosjektplanen for det innledende arbeidet.

Prosjektplan SkyHigH

Lars-Erik Pedersen, Jon Arne Westgaard & Hallvard Westman

Innholdsfortegnelse

1 Mål og rammer

1.1 Bakgrunn

En av teknologiene som har fått mye oppmerksomhet i det siste er nettskyer og virtuelle maskiner. En virtuell maskin er enkelt forklart en emulasjon av en komplett datamaskin, og en nettsky er en samling av virtuelle maskiner. Fordeler som kostnadsbesparelser, bedre utnyttelse av ressurser, enkelt oppsett av nye maskiner samt miljøhensyn (mindre støy, reduksjon i strømforbruk, plassbesparing) kan her trekkes frem.

En virtuell maskin er ikke bare nyttig til bruk for bedrifter og forskningssammenheng, men også for vanlige personer som ønsker å teste ut forskjellige operativsystemer og applikasjoner uten å måtte sette opp en dedikert fysisk maskin.

Høgskolen i Gjøvik har benyttet seg av virtuelle maskiner i blant annet emnene Systemadministrasjon (1), Databaser og applikasjonsdrift (2) og Ethical Hacking & Penetration Testing (3) . Bruk av virtuelle maskiner i disse emnene har gitt studentene en liten smak av den virkelige verden i arbeidslivet hvor studentene er delt inn i grupper og blitt tildelt et eget sett med virtuelle maskiner (en virtuell lab), og da kunne jobbe med “reélle” caser mot virtuelle laber som å drifte et lite nettverk, drifte serverapplikasjoner samt teste sårbarheter. Dette ville ikke ha vært like gjennomførbart ved bruk av fysiske maskiner (plassproblemer, ikke nok maskiner til alle).

Fordelen med denne løsningen er at det å opprette en virtuell maskin (eller flere) er veldig enkel, skulle noe gå galt på en virtuell maskin kan man rulle tilbake til et “snapshot” (en tidligere versjon av den virtuelle maskinen), man har høy fleksibilitet ved at man kan endre på spesifikasjoner etter behov (som legge til mer minne eller diskplass), samt gjenbrukbarhet (en VM tar kun ressurser på host/server og diskplass, når den ikke er i bruk lengre sletter man selve VM-en for og frigjøre ressurser og diskplass).

En utfordring ved den eksisterende løsningen har vært ytelse. I lab-timene har det vært nødvendig å planlegge kjøring av enkelte ressurskrevende kommandoer til spesifikke tidspunkt da dette ellers har ført til at maskinene har blitt uholdbart trege. Det ønskes derfor en løsning som skalerer godt, og som i tillegg er tilpasset med tanke på brukervennlighet og administrasjon. Et forprosjekt i emnet Systemadministrasjon tok for seg OpenNebula, Eucalyptus og OpenStack, og konkluderte med at OpenStack var det mest passende. Løsningen skal brukes av de faglig ansatte, som skal ha mulighet til å enkelt opprette en eller flere virtuelle maskiner og gi studentene tilgang til disse i de forskjellige emnene.

1.2 Prosjektmål (Effektmål og resultatmål)

Resultatmål

SkyHighs mål er å implementere OpenStack for å realisere en løsning for en privat nettsky ved HiG. Løsningen skal ha følgende funksjonalitet:

- IaaS, for realisering av virtuelle lab'er i forskjellige emner
- Mulighet for å migrere virtuelle maskiner fra egen PC og inn i skyen.
- Et web-basert brukergrensesnitt
    - Administrasjonspanel, der man håndterer prosjekter
    - Brukerpanel der man kan starte, stoppe og restarte egne instanser
- Mulighet for å utvide brukergruppen, også til eksterne ressurser, f.eks VGS-elever.
- Implementert autentisering med HiGs systemer via LDAP/FEIDE.

Effektmål

SkyHigh skal på sikt overta for eksisterende løsning ved HiG, men skal i første omgang leve side om side med den. Det forventes at løsningen ha følgende effekt:

- Rette opp/fjerne ytelse- og stabilitetsproblemer ved nåværende løsning
- Forbedre brukervennlighet og administrasjon
- Enklere oppsett av prosjekter
- Mindre arbeidslast på administrator
- Ha bidratt til OpenStack-prosjektet (OpenSource).
- Sette Norge og Gjøvik på OpenStack-kartet (4)
- Være fremtidsrettet og skalerbar

2 Omfang

2.1 Oppgavebeskrivelse

Oppgaven vår går hovedsaklig ut på å sette opp en nettsky på HiG for de faglig ansatte (primært lærerne) som skal sørge for at foreleserene skal kunne tilby studentene fleksible lab-løsninger i sine emner gjennom et brukervennlig grensesnitt. Bakgrunnen for oppgaven er at nåværende løsning opplever stabilitets- og ytelsesproblemer, og ett nytt system med OpenStack og lastbalansering forventes å kunne løse dette. For at dette skal realiseres må det være mulighet for å fleksibelt kunne øke ressurser som CPU, minne og lagring på de virtuelle maskinene, samt en lettvint måte å legge til flere noder som kjører de virtuelle maskinene.

I tillegg ønskes det en mulighet for å migrere virtuelle maskiner man har satt opp på egen PC inn i nettskyen, for å teste oppsett på kraftigere maskinvare. Det vil si, å tilby ren regnekraft som tjeneste.

Det skal utvikles en driftsplan og en SLA for den ferdige løsningen, slik at den enkelt kan vedlikeholdes i etterkant av prosjektet.

2.2 Avgrensning

Prosjektet skal først og fremst implementere OpenStack-rammeverket for å virkeliggjøre målene nevnt i 1.2. Å bygge en privat skyløsning er et prosjekt som vanligvis spenner over en mye lengre tidsperiode enn vi har til rådighet. Derfor vil vi ikke drive ytelsestesting og analyse (da dette dekkes av en annen bacheloroppgave). Mulighetene for “high availability” og redundans skal ikke dekkes. Primært skal systemet utvikles for å bli brukt på HiG, for de aktuelle emnene, ikke for eksterne brukere. Allikevel bør det være enkelt å utvide det i den retningen.

3 Prosjektorganisering

3.1 Ansvarsforhold og roller

Oppdragsgiver og veileder

Vår oppdragsgiver er førsteamanuensis Erik Hjelmås ved Høgskolen i Gjøvik. Erik vil være en sterk faglig ressurs, og være til god hjelp med den tekniske biten av prosjektet. Førsteamanuensis Hanno Langweg er vår veileder. Han vil bistå med teoretisk hjelp rundt det å jobbe i et stort prosjekt, samt komme med innspill til arbeidet underveis.

Prosjektleder

Lars Erik Pedersen er, i følge gruppereglementet, valgt som prosjektleder for gruppa. Det er prosjektleders ansvar å løse eventuelle konflikter som måtte oppstå underveis, selv om hovedmålet er å bli enige i fellesskap. At frister blir overholdt er også prosjektleders overordnede ansvar, samt at fremgangen i prosjektet følger planen så godt som mulig.

Webansvarlig

Hallvard Alte Westman er ansvarlig for gruppens webside. Ansvaret inkluderer oppsett, design og vedlikehold. Resten av gruppa skal ha tilgang til å redigere innhold, slik at alle kan bidra med oppdateringer til websiden.

Kontaktperson

Prosjektleder er kontaktperson for prosjektet. Kontakpersonen skal holde kontakt med oppdragsgiver og veileder og avtale møter med disse. Annen mail- og telefonkorrespondanse kan bli delegert til de andre medlemmene.

3.2 Rutiner og regler i gruppa

Gruppa har opprettet en felles kalender, der det er planlagt 31 timer jobbing per uke med prosjektet. Fredager er fridag, siden Jon Arne og Hallvard har andre fag hele dagen. Helgene er også i utgangspunktet fridager, men dersom det er nødvendig vil også bruke denne tiden til prosjektet. Arbeidet vil i all hovedsak bli utført på vårt tildelte grupperom. Gruppens fulle reglement er lagt ved i vedlegg X.

3.3 Verktøy

Den ferdige prosjektrapporten skal skrives i ReStructuredText som etterpå genereres til et PDF-dokument, men skal først samskrives i Google Docs. Dette gjør at behovet for versjonshåndtering blir minimalt, siden alle hele tiden vil skrive i samme dokument og vi vil spare tid siden vi slipper å håndtere versjonskonflikter. Alle andre dokumenter er lagret i en felles DropBox for å holde alt synkront for alle gruppemedlemmer. I tillegg tilbyr DropBox automatisk versjonskontroll, som sikrer at vi alltid har en stabil backup av alle viktige filer. Alle egenutviklede script, vil bli versjonskontrollert med github, og vil også bli lagret i DropBox. Websiden vil kjøre Wordpress 3.1.4, da IT-tjenesten ved HIG ikke støtter PHP 5.2.4 Det kan også nevnes at hvert gruppemedlem daglig kjører full inkrementell backup av sin egen laptop mot hver sin private server for å sikre seg mot tap av data.

4 Planlegging, oppfølging og rapportering

4.1 Hovedinndeling av prosjektet (Oppdeling i prosjektfaser)

Karakteristikker for vårt prosjekt:

Prosjektet vil i første omgang basere seg mye på å utforske og studere OpenStack. Hverken oppdragsgiver eller gruppen har benyttet seg av systemet som skal implementeres tidligere. I første omgang er det da aktuelt med en enkeltinstallasjon av OpenStack, som gruppen kan benytte i den første kartleggingsfasen.

Basert på kunnskapen vi har tilegnet oss etter installasjon/anvendelse av OpenStack på en node(server) vil vi rulle systemet ut på samtlige noder. Dette vil da være en fase med fullt fokus på å få systemet til å fungere i sin helhet.

Når det tekniske fungerer, vil det være aktuelt å gå inn i en ny kravspesifisering i forbindelse med Horizon. Horizon er webgrensesnittet til OpenStack som skal benyttes av brukerne, og det er her systemutviklingsjobben skal gjøres. Her vil det være nødvendig å gjøre omfattende “research” på hvilken funksjonalitet som alleredet eksisterer i OpenStack og hvilken funksjonalitet oppdragsgiver krever i tillegg. Det kreves tillegning av informasjon i forhold til hver modul som skal utvikles til Horizon. Grunnen til dette er at vi skal jobbe mot Open Source Community som har sine egne standarer som skal overholdes med tanke på kodestruktur, dokumentasjon og lignende. Samtidig må gruppen ha mulighet å sette seg inn i nye språk (Python spesielt). Testing og ferdigstilling vil også være viktig etter hver modul, slik at SkyHigh-IO har mulighet til å gjennomføre stresstesting ved behov. Derfor kreves det at systemet har en viss oppetid under utvikling. Gruppen har behov for at hver modul kan utvikles fleksibelt, da gruppen hele tiden er i en læringsprosess, både i forhold til OpenStack og prosjekter av denne størrelsen. Alle kravene kan heller ikke låses i forkant av hele utviklingsprosessen, da gruppen må forholde seg til OpenStack API-et, og det kan tenkes at flere problemstillinger vil oppstå underveis med tanke på at arbeidsgiver er ukjent med OpenStack.

Integrering med oppdragsgivers eksisterende systemer vil være en egen fase i prosjektet, da dette er en omfattende jobb som er svært ulik de foregående fasene. Dette vil også kreve en egen kravspesifisering.

Det vil underveis være aktuelt og hele tiden sette seg inn i OpenStack Open Source Project, i forhold til konvensjoner etc.

Gruppen fikk tidlig et inntrykk av at en agil utviklingsmodell måtte benyttes i dette prosjektet siden ingen i gruppen hadde vært borti et slikt prosjekt før. OpenStack er nytt og ingen har utført et slikt prosjekt på HiG tidligere. Arbeidsgiver hadde heller ikke benyttet seg av OpenStack eller Cloud Computing på en slik måte, som gjorde det vanskelig å fastsette rekkefølge på inkrementer i utviklingsperioden.

Gruppen har valgt å benytte seg av SCRUM som systemutviklingsmodell for videreutviklingen av Horizon. Gruppen konkluderte med at SCRUM passer best med tanke på å jobbe med Open Source-Miljøet, levering av inkrementer, tillegg av krav og tidsestimater. Gruppen ser det også som en fordel i forhold til læringskurven ved OpenStack.

Vi valgte allikevel å benytte oss av noen konsepter fra inkrementell utvikling. Ved hver demo setter vi av en periode til å teste, samt rulle ut inkrementet på systemet, dette håper vi skal sørge for at SkyHigh - IO får den oppetiden de krever. Vi har også at vi er inspirert av spiral-modellen ettersom hver sprint vil bestå av en fase der vi innhenter informasjon, en fase der vi utvikler, og en fase der vi tester/ruller ut.

4.2 Plan/krav for statusmøter og beslutningspunkter

Sprint planning innebærer å velge ut krav som skal gjennomføres i en sprint, disse blir hengt opp og låst for denne perioden. Vi vil her gå gjennom hva demoen på slutten av sprinten burde inneholde. Daily scrum innebærer at gruppen har et 15 minutter møte på starten av dagen der vi går gjennom hva som har blitt gjort, hva man skal gjøre og hva som gjenstår i forhold til en iterasjon. Sprint Review : Det vil være statusrapporter i etterkant av hver iterasjon som et resultat av Sprint Review. I etterkant av hver sprint evaluerer vi iterasjonen og tar med oss videre ting vi kan gjøre bedre og etterlater ting som ikke fungerte så bra. Dette er i hovedsak hva statusrapporten vil inneholde. Møte med arbeidsgiver og veileder vil foregå før hver sprint-planning. Møte med veileder vil i tillegg foregå ukentlig i startfasen.

4.3 Ressursbehov

Minimum to servere for regnekraft Minimum en server for lagring

5 Organisering av kvalitetssikring

5.1 Dokumentasjon, standardbruk og kildekode

All koding og scripting som blir gjort skal kommenteres underveis, slik at vi unngår å bruke tid på å forstå hva vi egentlig skrev for fire dager siden. OpenStack bruker Apache 2.0-lisensen, som da betyr at eventuelle moduler som blir kodet til OpenStack-prosjektet må følge denne.

5.2 Risikoanalyse

BESKRIVELSE	SANNSYNLIGHET	KONSEKVENS
Langtids sykdom	Lav	Ressursmangel, innsnevring av oppgave, avskjedigelse
Datatap	Lav	Dobbeltarbeid
Mangel på kompetanse	Middels	Løsninger som ikke følger best practice
Dårlige tidsestimater	Høy	Ikke overholde interne frister.
Ødeleggende endringer	Middels	Ødelegge allerede fungerende moduler

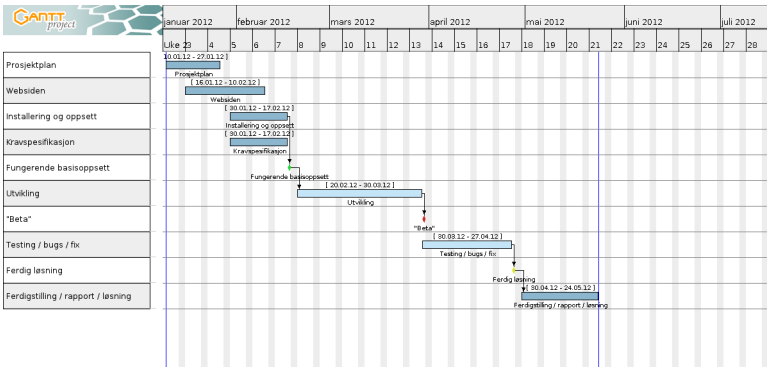
Som vi ser av tabellen er risikoen høyest for at vi har gjort dårlige eller feilaktige tidsestimater. For å unngå dette må vi planlegge nøye, og heller gjøre for høye estimater enn for lave. Dette vil gi oss litt buffertid på hver arbeidsmodul, som vil være god å ha når det dukker opp noe uforutsett. Vi må også hele tiden sørge for å følge fremgangsplanen og ikke havne på etterskudd.

Ellers ser vi en mulighet for at vårt nåværende kompetansenivå kan bli en utfordring. Vi har ingen erfaring med utrulling og drifting av slike virtualiseringsløsninger fra tidligere, annet en de virtuelle lab'ene vi har benyttet i emner her på HiG. En annen utfordring er at OpenStack er skrevet i Python, et programmeringsspråk vi heller ikke har kompetanse innenfor. Her må vi sørge for å utnytte fagmiljøet best mulig, samt å lese oss godt opp gjennom artikler og bøker funnet både på internett og biblioteket. Suksessfaktorer

For at vår løsning skal blir suksessfull, må den kunne erstatte løsningen HiG allerede har i dag 100%. Brukeropplevelsen og ytelsen må bli økt dramatisk både for administratorene (ansatte) og for brukerene (studentene). Et viktig ledd i dette er at driftsplanen blir så god at den kan brukes direkte av IMT, uten noe særlig behov for modifikasjon. Det er viktig at administratorene blir mest mulig selvhjulplne, slik at ikke alt ansvar blir liggende på en person, slik det er i dag. For at løsningen skal skalerere, samt løse ytelesesproblemene ligger mye av "ansvaret" på maskinvaren. Mengden fysisk minne er kritisk for at løsningen skal kunne skalere, samt at løsningen tilrettelegger for å enkelt kunne legge til flere noder.

6 Plan for gjennomføring

6.1 Fremdriftsplan



7 Prosjektavtale

Levert til Studenttorget for signatur av dekan.

8 Referanser

1: *http://www.hig.no/imt/emnesider/imt3292*

2: *http://www.hig.no/studiehaandbok/studiehaandboeker/2009_2010/emner/avdeling_for_informatikk_og_medieteknikk/imt3441_database_og_applikasjonsdrift*

3: *http://www.hig.no/studiehaandbok/studiehaandboeker/2010_2011/emner/avdeling_for_informatikk_og_medieteknikk/imt3491_ethical_hacking_and_penetration_testing*

4: *http://maps.google.com/maps/ms?msid=207730393988481837795.0004af95dec257674a36e&msa=0*

9 Grupperegler

Grupperegler for SkyHigh ADM

1. Dersom uenighet oppstår, skal konflikten løses ved en demokratisk avstemning
2. Prosjektets leder skal være Lars Erik Pedersen. Vervet skal ikke gå på rundgang.
3. Dersom en av gruppemedlemmene ikke utfører avtalt arbeid, uten en god begrunnelse, vil vedkommende få en passende sanksjon fritt bestemt av de to resterende gruppemedlemmene.
4. For at et medlem skal kunne avskjediges fra gruppen, må vedkommende få tre sanksjoner i løpet av samme uke.
    1. Dersom et medlem blir avskjediget, skal dette gjøres skriftlig, og det skal legges frem saklige argumenter på mislighold av avtale.
5. Eventuelle kostnader skal deles på 3.
6. Alle gruppens medlemmer har myndighet til å signere på vegne av gruppen, men minst to av medlemmene må være informert, og begge disse må samtykke.
7. Alle plikter å møte til avtalte tidspunkter. Dersom man skulle bli forhindret i å møte til avtalt tid, må dette avklares så fort som overhodet mulig.
8. Ved sykdom skal det meldes fra omgående, helst med en viss formening om når man blir frisk.
9. Sabotasje fører til direkte avskjedigelse.

Lars Erik Pedersen

Jon Arne Westad Westgaard

Hallvard Alte Westman

B Opplasting av Image, tester

Opplasting av et disk-image til Glance viste seg å være tid- og ressurskrevende. For å måle dette, brukte vi to enkle tester

Tidsbruk

Tidsbruken ble målt ved å laste opp det samme image fem ganger etter hverandre, og måle tiden med kommandoen time. Image vi testet med er et Windows Server image på 20 GiB. Denne kommandoen ble kjørt fem ganger:

```
time glance add name="Windows_Server_2008_R2_Enterprise_x64" \
is_public=true container_format=ovf disk_format=raw < windowsserver.img
```

Resultater

1. real 25m32.127s
   user 0m15.005s
   sys 0m32.706s
2. real 27m13.829s
   user 0m16.241s
   sys 0m32.018s
3. real 29m3.147s
   user 0m16.541s
   sys 0m32.018s
4. real 28m59.018s
   user 0m16.241s
   sys 0m32.250s
5. real 31m24.320s
   user 0m17.493s
   sys 0m31.654s

Dette gir en gjennomsnittlig opplastningtid på 28 minutter og 30.488 sekunder.

Ressursbruk

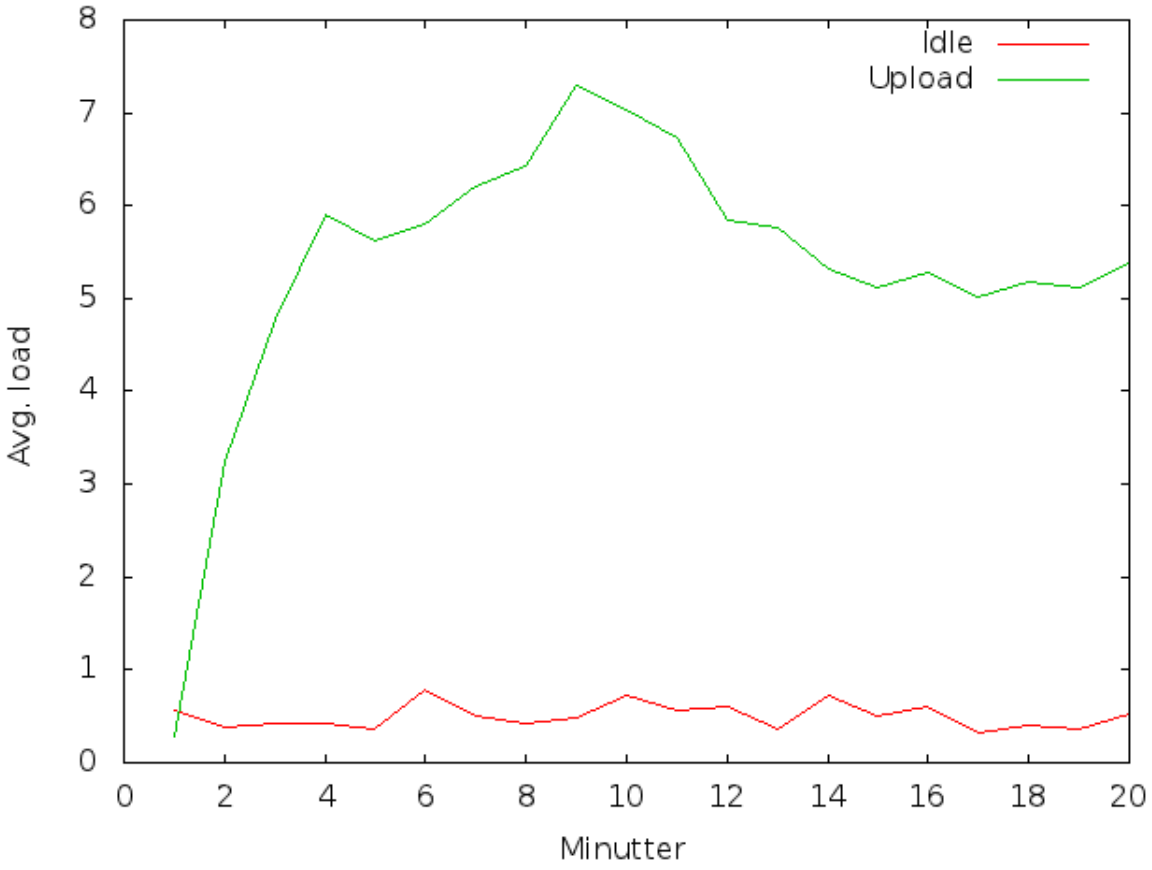
Med ressursbruk mener vi her bruk av CPU. Dette ble målt ved først å måle gjennomsnittlig last én gang i minuttet i 20 minutter ved idle, og deretter samme test startet samtidig som opplastingen. Målingen ble gjort med et enkelt skript:

```
#!/bin/bash
```

```
COUNT=1
while [ $COUNT -le 20 ]; do
‘uptime >> load.txt‘
sleep 60
((COUNT++))
done
```

Resultater

Figur 16: Gjennomsnittlig last ved image opplasting



Vi ser at lasten ligger stabilt mellom 6 og 7 ved opplasting av et image.

C Prosjektavtale

Vedlegget inneholder vår kontrakt med oppdragsgiver og skolen.

HØGSKOLEN I GJØVIK

PROSJEKTAVTALE

mellom Høgskolen i Gjøvik (HiG) (utdanningsinstitusjon),

Erik Hjelmås

(oppdragsgiver), og

Jon Arne Westad Westgaard,
Hallvard Alte Westman,
Lars Erik Pedersen

(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 9.1.12 til 23.5.12.

   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): Jon Arne A. Westgaard dato 25/1-12

Lars Erik Pedersen dato 25/1-12

Hallvard Vedvik dato 25/1-12

dato

Oppdragsgiver (signatur): dato 26/1-2012

IMT Dekan/prodekan (signatur): dato 15/2-2012

D Arbeidslogg

Fullstendig arbeidslogg slik den ble gjennomført kan leses på `http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/?page_id=5`

Uke2

Lest alle formelle dokumenter. Herunder emnebeskrivelse, utfyllende informasjon og retningslinjer. Har også skummet gjennom 2-3 tidligere oppgaver. Søkt litt rundt på ACM og IEEE etter relevante artikler. Jobbet med struktur for prosjektavtale og diskuterte systemutviklingsmodell. Skrev ut og signerte grupperegler og prosjektavtale. Satt opp verktøy for oppgavehåndtering (Trello.com), startet på Gantt-skjema (Gantt Project), valgte prosjektleder. Begynt å skrive på prosjektplan. Bakgrunn, prosjektorganisering, risikovurdering, planlegging. Funnet ut at StackOps finnes. Planlagt noen punkter til møte med ErikH

Uke 3

Ferdigstilte installasjonen av Ubuntu 11.10 og OpenStack på en server, testet ut webguiet. Skrev videre på prosjektplanen. Installert OS på en server. Forsøkt å sette opp proof-of-concept. Lest om OpenStack OpenSource Community. Skrevet videre på prosjektplanen, sett på OpenStack-community-et og Launchpad-siden.

Uke 4

Lest opp på systemutviklingsmetoder, samt drøftet karakteristikker. Tegnet use-case. Registrert og satt opp OpenStack dashboard utviklings-miljø. Satt opp wordpress. Møte med Tom Røise, jobbet litt med forprosjekt, satt opp autoinstall-image for Ubuntu Server 11.10 til bruk for re-innstallering av serverene, wbs. Sett litt på det spesifikke som må utvikles. Lest gjennom dokumentasjon for glance, keystone og swift.

Uke5

Designet nettverk, hatt uformelt møte med Jon Langseth. Designet arkitektur. Lest dokumentasjon. Redesignet nettverksoppsett, renskrevet dokumentasjonen av den. Begynt på kravspesifikasjon.

Uke 6

Kravspek, forberedelse til installasjon av OpenStack på Debian Wheezy, use-case & møte med Erik Hjelmås. Har jobbet med kravspec, usecase-tegning, testet openstack ved hjelp av devstack. Bygget rackskap.

Uke 7

Skrevet ferdig kravspesifikasjon. Jobbet med å sette opp utviklingsmiljø for horizon, innså at jeg måtte jobbe mer med python webutvikling så jeg tok ett steg tilbake og jobbet med å sette opp en simpel djangoinstallasjon. Installert Debian Wheezy på server som skal være controller. Startet installasjonen av Openstack - litt problemer med å få webgrensesnittet til å autensiere brukere.

Uke 8

Opplæring Django, konfigurasjon av enviroment. Lagt til rette for utviklingsperioden og sprint planning meet. Hjalp Jon Arne med å få opp controlleren... Fikk installert os på lagringsnoden. Jobbet med å få OpenStack til å kjøre på Debian Wheezy.

Uke 9

Installerte kvm på alle noder. Satte opp nova-volume. Registrerte alle hos controllernoden. Nå snakker de sammen.. Webgrensesnittet har knekt litt, så vi får ikke testet så mye enda. Snart kjørt gjennom 30 tutorials i Python. Django er litt mer tilnærmelig, og da også horizon.

Uke 10

Installerte Ubuntu 11.10 på alle servere. Satte opp igjen nova-compute. Forberedt utviklingsmiljø for horizon, og fått oversikt over struktur. Forandret nettverksoppsettet, fordi VLANmanager i OpenStack tydeligvis ikke fungerte. Satt opp templates for suspendmuligheter i horizon.

Uke 11

Jobbet med Horizon-api for oppretting av VM-er fant frem og fikk manipulert kildekode for oppretting. Forsøkte å finne ut hva som feiler, og hvorfor. Fant ut HVA som feiler under tildelingen av floating IP, men ikke hvorfor... Jobbet med templates for batchoppretting, mange problemer. Spesielt view-referanser. Lagde work-around for en bug, der IP-adresser fra slettede instanser ikke blir frigitt. Skrev på rapporten om nettverk i OpenStack og om vår arkitektur. Møte med veileder, planlagt oppgradering til nyere versjon av OpenStack. Installert testoppsett på en virtuell maskin som gikk meget bra. Evaluert dashboard, konkluderte med at man burde oppgradere, konsultert veileder. Forberedt oppgradering.

Uke 12

Satte i gang diskkloning og tok backup. Testet installasjonen på en virtuell maskin, ventet på at Lars Erik skulle trenge hjelp med installeringen. Mye banning og friske gloser fra Lars Erik under installering.

Uke 13

Fikset problematikken i horizon, slik at vi nå kan legge til paneler. Ordnet konfigurasjonsfeil i Horizon. Startet på et bash-script som skal gjøre samme jobben som vårt nye

panel i horizon. Skal hjelpe oss å finne strukturen, og arbeidsflyten. Fullførte gårsdagens script, selv om det enda trenger litt mer funksjonalitet. Bl.a å tilegne IP-adresser.

Uke 14

Uhell med dropbox, slettet prosjektet, skaper problemer med git, dette må reinitialiseres etter sprint. Påskeferie for Gruppa.

Uke 15

Skrev script for sletting av batch. Riktignok i bash, men planlegger å porte det til python etterhvert. Begynte å se på hvordan man kan få slettet en batch fullstendig. Fortsatt problemer med å få callbackfunksjonen fra en tabell til å motta rett ID. (her snakker vi om å legge til en tenant i en batch).

Uke 16

Skrevet teori om MVC, oppbygging av Horizon og Django. Jobbet med å omgjøre Architecture Document til Design-document. Skrev på rapporten. Justerte nettverksdelen, skrev ferdig om modulene. Leste meg opp på et par av de vi ikke har implementert.

Uke 17

Herfra og ut vil Gruppen fokusere på å skrive sluttrapport, vi ser ingen hensikt ved å fullføre logg for dette.

E Møtelogg

Vedlegget inneholder alle notater fra møtene vi har hatt med veileder og oppdragsgiver gjennom prosjektperioden.

Møtereferat

Møte nr: 1
Tidspunkt: 09.01.12 kl 08:30
Sted: Kråkereiret
Varighet: 30 minutter
Deltagere: Jon Arne, Lars Erik, Hallvard, Hanno

- Viktigste er å bli ferdig!
- Demo ved presentasjon lønner seg
- Hvorfor skal vi gjøre dette når amazon har en løsning?
- Kan vår løsning integreres med Rackspace? Kan skille en B fra en A.
- 3 sider tekst pr. uke, ca 50-60 sider
- Dette er et prosjekt som kan integreres med et regionalt prosjekt for bedrifter. (SkyLine)
- Kan vi få kontakt med bedrifter som bruker vm-løsninger? Erfaringer/mangler ved eksisterende løsning
- Kyrres løsning ved HiO
- Målgruppe? Alle studentene?
- Skal brukere få opprette maskiner selv? Selvbetjening, brukere restarte VM selv
- Løse problemer automatisk, overvåkning (nagios)
- Skalerbarhet, erik skal ikke må gjøre alt selv
- Anbefaler at vi deler opp prosjektet i fornuftige estimater, sette opp ant. timer per del, arbeidspakker
- Se på Mantis, be arbeidsgiveren opprette (IT-tjenesten)
- Snakk med hverandre, skrive referat
- Sett opp latex-dokument/rst-dokument, git/svn
- få opp en løsning om 3 uker som får opp en vm, letter å utvide derfra.
- IPV6 er lurt å tenke på
- Scrum
- Flytte VM fra skyen til lokal pc og andre veien
- Leie ekstern kapasitet?
- Sammenligne med Amazon, Windows Azuure ++
- Billing/kostnader, administrasjon av Vm-er, “utløpstid”, stoppe “døde” VM-er
- Kvote?
- Utvikle driftsplan for løsning
- Noe å presentere for Hanno hvert ukemøte
- Begrense innhold på web-side

- Møte hver torsdag 12.30-13.00 (Evt 12.00-13.00 med den andre gruppen)
- Referat fra møte med den andre gruppen
- Plan over HVA som skal gjøres (Ganske spesifikt)
- Grensesnitt for internt og ekstern arkitektur?
- -Hypervisor, hvilken?
- Hanno leste en artikkel i ACM / ie computer. Skole som har gjort dette i USA, finn denne.

Møtereferat

Møte nr: 2
Tidspunkt: 16.01.12 kl 11:00
Sted: 2. etasje K-bygg, utenfor Eriks kontor
Varighet: 60 minutter
Deltagere: Lars Erik, Jon Arne, Hallvard, Erik

- Kravspesifisering (I forhold til hva som allerede finnes i OpenStack)
    - Feide (Etterhvert)
    - Flytte VM fra lokal maskin til skyen (Glance)
    - SkyLine, prøve å få dette til
    - 3 punkter:
        - Regnekraft
        - Undervisningslab
        - Testing
    - Kjøra på med Ubuntu (som i admin-guide), men helst ikke bruk distro-spesifikke pakker
- Status på servere (hvor mange av de skal vi bruke?)
    - 1 controller, 6 compute, 1 nettverk, 1 lagring
- SkyLine? Samarbeid med Hannos prosjekt?
    - Software-testing i skyen
    - Næringslivet ønsker et enkelt miljø å teste en plattform
- Hvor teknisk skal rapporten være?
- Windows-lisenser?
- Målgruppe? Alle studentene?
    - Tja... (Kanskje etterhvert)
- Skal brukere få opprette maskiner selv? Selvbetjening, brukere restarte VM selv
- Hva ønsker Erik? Hvordan ser Erik for seg den ferdige løsningen? (Minstekrav, “kjekt å ha”)
- Hypervisor? (Fortsette med Xen?)
    - ja
- Utvikling? Hva må lages fra scratch? Hva finnes?
- Kjapp plan:
    - Prosjektplan
    - Få på plass hardware
    - Lage en skisse av hele oppsettet
- Bidra til OpenStack-prosjektet
- Redundans (Bachelor-prosjekt til neste år) - men se litt på det, legge til rette?

- Def: En samling av virtuelle maskiner er et prosjekt
- Schedulern
- Ikke kjøre noen VM-er på lagringsnoder
- Bruke 2 nettverkskort på serverene, 1 internt nett til lagring

Møtereferat

Møte nr: 3
Tidspunkt: 19.01.12 kl 12.30
Sted: Kråkereiret
Varighet: 30 min
Deltagere: Jon Arne, Lars Erik, Hanno

- Usecases
- Selvbetjening - ta sikte på at brukerene skal gjøre alt selv
- Tenk større - kan alle studenter ved HiG få tilgang til 2 VM-er hver?
- Tegne arkitekturplan
- Ressurshåndtering, eks. 20 timer med full cpu-kraft/20 gb trafikk i mnd, deretter strupe
- finn ut hvordan man kan laste opp image. Kun gjennom API eller gjennom SOAP/ReST?

Møtereferat

Møte nr: 4
Tidspunkt: 26.01.12 kl 12.30
Sted: Kråkereiret
Varighet: 30 minutter
Deltagere: Jon Arne og Lars Erik, Hanno og Erik

- Hva har vi gjort de siste 7 dagene:
    - Billing-system - overvåkning av ressursbruk per bruker, støtte for varsler (Atom-feed). Skal være innebygd, må se på hvordan bruke eksternt program til å behandle info. Gjøre automatisk - begrensninger på bruk?
    - Webside er satt opp
    - Sett på de forskjellige hypervisors som kan brukes
    - Sett på nettverksoppsettet på server
    - Devstack-script er for “dårlig”, knekker etter restart. Må sette opp “manuell” installasjon (fredag)
    - Laget use-case
    - Arkitekturoversikt
    - Nettverksdesign
    - Prosjektplan
- Kjøre på Debian, Xen
- Kravspesifikasjon, starte på mandag
- Endre litt på use-case

Møtereferat

Møte nr: 5
Tidspunkt: 01.03.12 kl. 12.30
Sted: Kråkereiret
Varighet: 30 min
Deltagere: Hanno, Erik, Jon Arne, Hallvard & Lars Erik

- Tekniske problemer i sluttrapport: Dokumenter de systematiske metodene brukt, begrunn valg av OS o.l., hva virket ikke/hvorfor

Møtereferat

Møte nr: 6
Tidspunkt: 15.03.12
Sted: Kråkereiret
Varighet: 30 minutter
Deltagere: Hallvard, Jon Arne, Lars Erik, Hanno

- Vurdering av oppgradering til Essex
- Burde sette en tidsfrist, på å prøve å oppgradere,
- Finn ut en mulighet å falle tilbake på.
- Utfall A fungerende essex : fungerende knapp
- Utfall B Gammel versjon av openstack, samme oppsett og funksjonalitet som nå, fungerende knapp
- Utfall C Ingen knapp, dåligere oppsett enn nå (UAKTUELT)

Møtereferat

Møte nr: 7
Tidspunkt: 22.03.12
Sted: Kråkereiret
Varighet: 30 minutter
Deltagere: Lars Erik, Jon Arne, Hallvard, Hanno, Erik

---

synliggjøre i rapporten:
avhengigheter, hvilke api er ikke stabilt, få fram mest mulig
vedtak, beslutninger om oppgraderinger,
Hvor er det det klapper sammen,

OpenStack
libvirt/python-libs
kvm

erikh: mest mulig modularinsering, vite avhengigheter,

hanno: Oppsummering av sprint, skjermbilde,

Møtereferat

Møte nr: 8
Tidspunkt: 03.05.12
Sted: Kråkereiret
Varighet: 30 min
Deltagere: Jon Arne, Lars Erik, Hallvard, Hanno, Erik

- -Hanno ville kalt “analyse” for diskusjon
- http://geekandpoke.typepad.com/geekandpoke/cloud/
- Hvorfor feiler oppretting etter, så så mange?

F Mail fra OpenStack angående kart

Vedlegget inneholder mailkorrespendanse med Steffano Maffulli i OpenStack, angående å bli plassert på det offisielle OpenStack-kartet.

From: Stefano Maffulli <stefano@openstack.org>
Subject: **Re: SkyHigh**
Date: May 15, 2012 2:36:00 AM GMT+02:00
To: Hallvard Westman <hallvard.westman@hig.no>

Hi guys,

first of all congratulations for your thesis. My colleague will put you on the map soon. In any case, I'd like to do even more for you: why don't you send me an abstract of your thesis and I'll see if it makes sense to publish it on openstack.org/blog as the first bachelor thesis on OpenStack. Would that give you bragging rights, too?

Let me know,
stef

On Mon 07 May 2012 05:25:37 AM PDT, Hallvard Westman wrote:

> Hello!
> We are three students from HIG University College in Gjøvik - Norway (http://www.hig.no/).
> We have written our bachelor-thesis based on our implementation and development of OpenStack.
> A gimmick of ours has been a goal to get on the Openstack map before we were finished with the thesis.
> We are now finished with the implementation of Openstack as well as some modules for Horizon, and we were now wondering if you had a place for us on the map.
>
> This is a link to our official website where you can get access to the entire bachelor-thesis. Sadly there is no way for you to see the actually implementation on any official channel just yet, but we would be happy to arrange a session to show it off.
> http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
>
> The thesis is written in Norwegian, and we are hoping to translate it, but it will depend on our grade. Hopefully you will put us on the map so we have something to brag about, which again can secure a strong enough grade for translation.
>
> Best regards
> Hallvard Westman, Lars Erik Pedersen, Jon Arne Westgaard.

G Konfigurasjon

Dette vedlegget inneholder konfigurasjon av OpenStack-modulene, samt konfigurasjon av nettverksutstyr.

nova.conf

```
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dhcpbridge=/usr/bin/nova-dhcpbridge
--debug=true
--logdir=/var/log/nova
--state_path=/var/lib/nova
--lock_path=/var/lock/nova
--verbose=False
--auth_strategy=keystone
--allow_resize_to_same_host=True
--compute_scheduler_driver=nova.scheduler.filter_scheduler.FilterScheduler
--dhcpbridge_flagfile=/etc/nova/nova.conf
--dmz_net=172.16.0.0
--dmz_mask=255.240.0.0
--fixed_range=172.16.0.0/12
--floating_range=192.168.10.128/25
--s3_host=192.168.10.2
--osapi_host=192.168.10.2
--rabbit_host=192.168.10.2
--volume_group=nova-volumes
--iscsi_helper=tgtadm
--osapi_compute_extension=nova.api.openstack.compute.contrib.standard_extensions
--my_ip=192.168.10.2
--public_interface=eth0
--vlan_interface=eth1
--sql_connection=mysql://nova:melkikakao2012@192.168.10.2/nova
--libvirt_type=kvm
--novnc_enable=true
--novncproxy_base_url=http://192.168.10.2:6080/vnc_auto.html
--vncserver_listen=0.0.0.0
--vncserver_proxyclient_address=192.168.10.2
--api_paste_config=/etc/nova/api-paste.ini
--image_service=nova.image.glance.GlanceImageService
--ec2_dmz_host=192.168.10.2
--glance_api_servers=192.168.10.2:9292
--force_dhcp_release=True
--connection_type=libvirt
```

```
--firewall_driver=nova.virt.libvirt.firewall.IptablesFirewallDriver
--multi_host=false
--keystone_ec2_url=http://192.168.10.2:5000/v2.0/ec2tokens
--ajax_console_proxy_url=http://192.168.10.2:8000
```

api-paste.ini

```
#############
# Metadata #
#############
[composite:metadata]
use = egg:Paste#urlmap
/: metaversions
/latest: meta
/1.0: meta
/2007-01-19: meta
/2007-03-01: meta
/2007-08-29: meta
/2007-10-10: meta
/2007-12-15: meta
/2008-02-01: meta
/2008-09-01: meta
/2009-04-04: meta

[pipeline:metaversions]
pipeline = ec2faultwrap logrequest metaverapp

[pipeline:meta]
pipeline = ec2faultwrap logrequest metaapp

[app:metaverapp]
paste.app_factory = nova.api.metadata.handler:Versions.factory

[app:metaapp]
paste.app_factory = nova.api.metadata.handler:MetadataRequestHandler.factory

#######
# EC2 #
#######

[composite:ec2]
use = egg:Paste#urlmap
/services/Cloud: ec2cloud

[composite:ec2cloud]
use = call:nova.api.auth:pipeline_factory
noauth = ec2faultwrap logrequest ec2noauth cloudrequest validator ec2executor
```

```
deprecated = ec2faultwrap logrequest authenticate cloudrequest validator ec2executor
keystone = ec2faultwrap logrequest ec2keystoneauth cloudrequest validator ec2executor

[filter:ec2faultwrap]
paste.filter_factory = nova.api.ec2:FaultWrapper.factory

[filter:logrequest]
paste.filter_factory = nova.api.ec2:RequestLogging.factory

[filter:ec2lockout]
paste.filter_factory = nova.api.ec2:Lockout.factory

[filter:totoken]
paste.filter_factory = nova.api.ec2:EC2Token.factory

[filter:ec2keystoneauth]
paste.filter_factory = nova.api.ec2:EC2KeystoneAuth.factory

[filter:ec2noauth]
paste.filter_factory = nova.api.ec2:NoAuth.factory

[filter:authenticate]
paste.filter_factory = nova.api.ec2:Authenticate.factory

[filter:cloudrequest]
controller = nova.api.ec2.cloud.CloudController
paste.filter_factory = nova.api.ec2:Requestify.factory

[filter:authorizer]
paste.filter_factory = nova.api.ec2:Authorizer.factory

[filter:validator]
paste.filter_factory = nova.api.ec2:Validator.factory

[app:ec2executor]
paste.app_factory = nova.api.ec2:Executor.factory

##############
# Openstack #
##############

[composite:osapi_compute]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: oscomputeversions
/v1.1: openstack_compute_api_v2
/v2: openstack_compute_api_v2
```

```
[composite:osapi_volume]
use = call:nova.api.openstack.urlmap:urlmap_factory
/: osvolumeversions
/v1: openstack_volume_api_v1

[composite:openstack_compute_api_v2]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap noauth ratelimit osapi_compute_app_v2
deprecated = faultwrap auth ratelimit osapi_compute_app_v2
keystone = faultwrap authtoken keystonecontext ratelimit osapi_compute_app_v2

[composite:openstack_volume_api_v1]
use = call:nova.api.auth:pipeline_factory
noauth = faultwrap noauth ratelimit osapi_volume_app_v1
deprecated = faultwrap auth ratelimit osapi_volume_app_v1
keystone = faultwrap authtoken keystonecontext ratelimit osapi_volume_app_v1

[filter:faultwrap]
paste.filter_factory = nova.api.openstack:FaultWrapper.factory

[filter:auth]
paste.filter_factory = nova.api.openstack.auth:AuthMiddleware.factory

[filter:noauth]
paste.filter_factory = nova.api.openstack.auth:NoAuthMiddleware.factory

[filter:ratelimit]
paste.filter_factory = nova.api.openstack.compute.limits:RateLimitingMiddleware.factory

[app:osapi_compute_app_v2]
paste.app_factory = nova.api.openstack.compute:APIRouter.factory

[pipeline:oscomputeversions]
pipeline = faultwrap oscomputeversionapp

[app:osapi_volume_app_v1]
paste.app_factory = nova.api.openstack.volume:APIRouter.factory

[app:oscomputeversionapp]
paste.app_factory = nova.api.openstack.compute.versions:Versions.factory

[pipeline:osvolumeversions]
pipeline = faultwrap osvolumeversionapp

[app:osvolumeversionapp]
```

```
paste.app_factory = nova.api.openstack.volume.versions:Versions.factory

##########
# Shared #
##########

[filter:keystonecontext]
paste.filter_factory = nova.api.auth:NovaKeystoneContext.factory

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.10.2
service_port = 5000
auth_host = 192.168.10.2
auth_port = 35357
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
admin_tenant_name = service
admin_user = nova
admin_password = nova
admin_token = ADMIN
```

keystone.conf

```
[DEFAULT]
bind_host = 0.0.0.0
public_port = 5000
admin_port = 35357
admin_token = ADMIN
compute_port = 8774
verbose = True
debug = True
log_config = /etc/keystone/logging.conf

# ================ Syslog Options ============================
# Send logs to syslog (/dev/log) instead of to file specified
# by `log-file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
# syslog_log_facility = LOG_LOCAL0

[sql]
connection = mysql://keystone:melkikakao2012@192.168.10.2/keystone
idle_timeout = 200
min_pool_size = 5
```

```
max_pool_size = 10
pool_timeout = 200

[ldap]
#url = ldap://localhost
#tree_dn = dc=example,dc=com
#user_tree_dn = ou=Users,dc=example,dc=com
#role_tree_dn = ou=Roles,dc=example,dc=com
#tenant_tree_dn = ou=Groups,dc=example,dc=com
#user = dc=Manager,dc=example,dc=com
#password = freeipa4all
#suffix = cn=example,cn=com

[identity]
driver = keystone.identity.backends.sql.Identity

[catalog]
driver = keystone.catalog.backends.sql.Catalog

[token]
driver = keystone.token.backends.sql.Token

# Amount of time a token should remain valid (in seconds)
expiration = 86400

[policy]
driver = keystone.policy.backends.rules.Policy

[ec2]
driver = keystone.contrib.ec2.backends.sql.Ec2

[filter:debug]
paste.filter_factory = keystone.common.wsgi:Debug.factory

[filter:token_auth]
paste.filter_factory = keystone.middleware:TokenAuthMiddleware.factory

[filter:admin_token_auth]
paste.filter_factory = keystone.middleware:AdminTokenAuthMiddleware.factory

[filter:xml_body]
paste.filter_factory = keystone.middleware:XmlBodyMiddleware.factory

[filter:json_body]
paste.filter_factory = keystone.middleware:JsonBodyMiddleware.factory
```

```
[filter:crud_extension]
paste.filter_factory = keystone.contrib.admin_crud:CrudExtension.factory

[filter:ec2_extension]
paste.filter_factory = keystone.contrib.ec2:Ec2Extension.factory

[app:public_service]
paste.app_factory = keystone.service:public_app_factory

[app:admin_service]
paste.app_factory = keystone.service:admin_app_factory

[pipeline:public_api]
pipeline = token_auth admin_token_auth xml_body json_body debug ec2_extension public_servi

[pipeline:admin_api]
pipeline = token_auth admin_token_auth xml_body json_body debug \
ec2_extension crud_extension admin_service

[app:public_version_service]
paste.app_factory = keystone.service:public_version_app_factory

[app:admin_version_service]
paste.app_factory = keystone.service:admin_version_app_factory

[pipeline:public_version_api]
pipeline = xml_body public_version_service

[pipeline:admin_version_api]
pipeline = xml_body admin_version_service

[composite:main]
use = egg:Paste#urlmap
/v2.0 = public_api
/ = public_version_api

[composite:admin]
use = egg:Paste#urlmap
/v2.0 = admin_api
/ = admin_version_api
```

glance-api.conf

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True
```

```
# Show debugging output in logs (sets DEBUG log level output)
debug = True

# Which backend store should Glance use by default is not specified
# in a request to add a new image to Glance? Default: 'file'
# Available choices are 'file', 'swift', and 's3'
default_store = file

# Address to bind the API server
bind_host = 0.0.0.0

# Port the bind the API server to
bind_port = 9292

# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/api.log

# Backlog requests when creating socket
backlog = 4096

# Number of Glance API worker processes to start.
# On machines with more than one CPU increasing this value
# may improve performance (especially if using SSL with
# compression turned on). It is typically recommended to set
# this value to the number of CPUs present on your machine.
workers = 0

# ================= Syslog Options ============================

# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
# syslog_log_facility = LOG_LOCAL0

# ================= SSL Options ===============================

# Certificate file to use when starting API server securely
# cert_file = /path/to/certfile

# Private key file to use when starting API server securely
# key_file = /path/to/keyfile

# ================ Security Options ==========================
```

```
# AES key for encrypting store 'location' metadata, including
# -- if used -- Swift or S3 credentials
# Should be set to a random string of length 16, 24 or 32 bytes
# metadata_encryption_key = <16, 24 or 32 char registry metadata key>

# ============ Registry Options ===============================

# Address to find the registry server
registry_host = 0.0.0.0

# Port the registry server is listening on
registry_port = 9191

# What protocol to use when connecting to the registry server?
# Set to https for secure HTTP communication
registry_client_protocol = http

# The path to the key file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_KEY_FILE environ variable to a filepath of the key file
# registry_client_key_file = /path/to/key/file

# The path to the cert file to use in SSL connections to the
# registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CERT_FILE environ variable to a filepath of the cert file
# registry_client_cert_file = /path/to/cert/file

# The path to the certifying authority cert file to use in SSL connections
# to the registry server, if any. Alternately, you may set the
# GLANCE_CLIENT_CA_FILE environ variable to a filepath of the CA cert file
# registry_client_ca_file = /path/to/ca/file

# ============ Notification System Options =====================

# Notifications can be sent when images are create, updated or deleted.
# There are three methods of sending notifications, logging (via the
# log_file directive), rabbit (via a rabbitmq queue), qpid (via a Qpid
# message queue), or noop (no notifications sent, the default)
notifier_strategy = noop

# Configuration options if sending notifications via rabbitmq (these are
# the defaults)
rabbit_host = localhost
rabbit_port = 5672
rabbit_use_ssl = false
```

```
rabbit_userid = guest
rabbit_password = guest
rabbit_virtual_host = /
rabbit_notification_exchange = glance
rabbit_notification_topic = glance_notifications

# Configuration options if sending notifications via Qpid (these are
# the defaults)
qpid_notification_exchange = glance
qpid_notification_topic = glance_notifications
qpid_host = localhost
qpid_port = 5672
qpid_username =
qpid_password =
qpid_sasl_mechanisms =
qpid_reconnect_timeout = 0
qpid_reconnect_limit = 0
qpid_reconnect_interval_min = 0
qpid_reconnect_interval_max = 0
qpid_reconnect_interval = 0
qpid_heartbeat = 5
# Set to 'ssl' to enable SSL
qpid_protocol = tcp
qpid_tcp_nodelay = True

# ============ Filesystem Store Options ========================

# Directory that the Filesystem backend store
# writes image data to
filesystem_store_datadir = /var/lib/glance/images/

# ============ Swift Store Options =============================

# Address where the Swift authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified,  default to 'https://'
swift_store_auth_address = 127.0.0.1:8080/v1.0/

# User to authenticate against the Swift authentication service
# If you use Swift authentication service, set it to 'account':'user'
# where 'account' is a Swift storage account and 'user'
# is a user in that account
swift_store_user = jdoe:jdoe

# Auth key for the user authenticating against the
# Swift authentication service
```

```
swift_store_key = a86850deb2742ec3cb41518e26aa2d89

# Container within the account that the account should use
# for storing images in Swift
swift_store_container = glance

# Do we create the container if it does not exist?
swift_store_create_container_on_put = False

# What size, in MB, should Glance start chunking image files
# and do a large object manifest in Swift? By default, this is
# the maximum object size in Swift, which is 5GB
swift_store_large_object_size = 5120

# When doing a large object manifest, what size, in MB, should
# Glance write chunks to Swift? This amount of data is written
# to a temporary disk buffer during the process of chunking
# the image file, and the default is 200MB
swift_store_large_object_chunk_size = 200

# Whether to use ServiceNET to communicate with the Swift storage servers.
# (If you aren't RACKSPACE, leave this False!)
#
# To use ServiceNET for authentication, prefix hostname of
# `swift_store_auth_address` with 'snet-'.
# Ex. https://example.com/v1.0/ -> https://snet-example.com/v1.0/
swift_enable_snet = False

# ============ S3 Store Options =============================

# Address where the S3 authentication service lives
# Valid schemes are 'http://' and 'https://'
# If no scheme specified,  default to 'http://'
s3_store_host = 127.0.0.1:8080/v1.0/

# User to authenticate against the S3 authentication service
s3_store_access_key = <20-char AWS access key>

# Auth key for the user authenticating against the
# S3 authentication service
s3_store_secret_key = <40-char AWS secret key>

# Container within the account that the account should use
# for storing images in S3. Note that S3 has a flat namespace,
# so you need a unique bucket name for your glance images. An
# easy way to do this is append your AWS access key to "glance".
```

```
# S3 buckets in AWS *must* be lowercased, so remember to lowercase
# your AWS access key if you use it in your bucket name below!
s3_store_bucket = <lowercased 20-char aws access key>glance

# Do we create the bucket if it does not exist?
s3_store_create_bucket_on_put = False

# When sending images to S3, the data will first be written to a
# temporary buffer on disk. By default the platform's temporary directory
# will be used. If required, an alternative directory can be specified here.
# s3_store_object_buffer_dir = /path/to/dir

# ============ RBD Store Options =============================

# Ceph configuration file path
# If using cephx authentication, this file should
# include a reference to the right keyring
# in a client.<USER> section
rbd_store_ceph_conf = /etc/ceph/ceph.conf

# RADOS user to authenticate as (only applicable if using cephx)
rbd_store_user = glance

# RADOS pool in which images are stored
rbd_store_pool = images

# Images will be chunked into objects of this size (in megabytes).
# For best performance, this should be a power of two
rbd_store_chunk_size = 8

# ============ Delayed Delete Options =============================

# Turn on/off delayed delete
delayed_delete = False

# Delayed delete time in seconds
scrub_time = 43200

# Directory that the scrubber will use to remind itself of what to delete
# Make sure this is also set in glance-scrubber.conf
scrubber_datadir = /var/lib/glance/scrubber

# =============== Image Cache Options =============================

# Base directory that the Image Cache uses
image_cache_dir = /var/lib/glance/image-cache/
```

```
[paste_deploy]
flavor = keystone
```

glance-api-paste.ini

```
# Default minimal pipeline
[pipeline:glance-api]
pipeline = versionnegotiation context apiv1app

# Use the following pipeline for keystone auth
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone
#
[pipeline:glance-api-keystone]
pipeline = versionnegotiation authtoken auth-context apiv1app

# Use the following pipeline to enable transparent caching of image files
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = caching
#
[pipeline:glance-api-caching]
pipeline = versionnegotiation context cache apiv1app

# Use the following pipeline for keystone auth with caching
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone+caching
#
[pipeline:glance-api-keystone+caching]
pipeline = versionnegotiation authtoken auth-context cache apiv1app

# Use the following pipeline to enable the Image Cache Management API
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = cachemanagement
#
[pipeline:glance-api-cachemanagement]
pipeline = versionnegotiation context cache cachemanage apiv1app

# Use the following pipeline for keystone auth with cache management
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone+cachemanagement
#
```

```
[pipeline:glance-api-keystone+cachemanagement]
pipeline = versionnegotiation authtoken auth-context cache cachemanage apiv1app

[app:apiv1app]
paste.app_factory = glance.common.wsgi:app_factory
glance.app_factory = glance.api.v1.router:API

[filter:versionnegotiation]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.version_negotiation:VersionNegotiationFilter

[filter:cache]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache:CacheFilter

[filter:cachemanage]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache_manage:CacheManageFilter

[filter:context]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.common.context:ContextMiddleware

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.10.2
service_port = 5000
auth_host = 192.168.10.2
auth_port = 35357
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
admin_token = ADMIN
admin_tenant_name = service
admin_user = glance
admin_password = glance

[filter:auth-context]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = keystone.middleware.glance_auth_token:KeystoneContextMiddleware
root@dublin:/etc/glance#
root@dublin:/etc/glance# cat glance-api-paste.ini
# Default minimal pipeline
[pipeline:glance-api]
pipeline = versionnegotiation context apiv1app
```

```
# Use the following pipeline for keystone auth
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone
#
[pipeline:glance-api-keystone]
pipeline = versionnegotiation authtoken auth-context apiv1app

# Use the following pipeline to enable transparent caching of image files
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = caching
#
[pipeline:glance-api-caching]
pipeline = versionnegotiation context cache apiv1app

# Use the following pipeline for keystone auth with caching
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone+caching
#
[pipeline:glance-api-keystone+caching]
pipeline = versionnegotiation authtoken auth-context cache apiv1app

# Use the following pipeline to enable the Image Cache Management API
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = cachemanagement
#
[pipeline:glance-api-cachemanagement]
pipeline = versionnegotiation context cache cachemanage apiv1app

# Use the following pipeline for keystone auth with cache management
# i.e. in glance-api.conf:
#   [paste_deploy]
#   flavor = keystone+cachemanagement
#
[pipeline:glance-api-keystone+cachemanagement]
pipeline = versionnegotiation authtoken auth-context cache cachemanage apiv1app

[app:apiv1app]
paste.app_factory = glance.common.wsgi:app_factory
glance.app_factory = glance.api.v1.router:API

[filter:versionnegotiation]
paste.filter_factory = glance.common.wsgi:filter_factory
```

```
glance.filter_factory = glance.api.middleware.version_negotiation:VersionNegotiationFilter

[filter:cache]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache:CacheFilter

[filter:cachemanage]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.api.middleware.cache_manage:CacheManageFilter

[filter:context]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.common.context:ContextMiddleware

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.10.2
service_port = 5000
auth_host = 192.168.10.2
auth_port = 35357
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
admin_token = ADMIN
admin_tenant_name = service
admin_user = glance
admin_password = glance

[filter:auth-context]
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = keystone.middleware.glance_auth_token:KeystoneContextMiddleware
```

glance-registry-paste.ini

```
[DEFAULT]
# Show more verbose log output (sets INFO log level output)
verbose = True

# Show debugging output in logs (sets DEBUG log level output)
debug = True

# Address to bind the registry server
bind_host = 0.0.0.0

# Port the bind the registry server to
bind_port = 9191
```

```
# Log to this file. Make sure you do not set the same log
# file for both the API and registry servers!
log_file = /var/log/glance/registry.log

# Backlog requests when creating socket
backlog = 4096

# SQLAlchemy connection string for the reference implementation
# registry server. Any valid SQLAlchemy connection string is fine.
# See:
# http://www.sqlalchemy.org/docs/05/reference/sqlalchemy/
#   connections.html#sqlalchemy.create_engine
sql_connection = mysql://glance:melkikakao2012@192.168.10.2/glance

# Period in seconds after which SQLAlchemy should reestablish its connection
# to the database.
#
# MySQL uses a default `wait_timeout` of 8 hours, after which it will drop
# idle connections. This can result in 'MySQL Gone Away' exceptions. If you
# notice this, you can lower this value to ensure that SQLAlchemy reconnects
# before MySQL can drop the connection.
sql_idle_timeout = 3600

# Limit the api to return `param_limit_max` items in a call to a container. If
# a larger `limit` query param is provided, it will be reduced to this value.
api_limit_max = 1000

# If a `limit` query param is not provided in an api request, it will
# default to `limit_param_default`
limit_param_default = 25

# ================= Syslog Options ============================

# Send logs to syslog (/dev/log) instead of to file specified
# by `log_file`
use_syslog = False

# Facility to use. If unset defaults to LOG_USER.
# syslog_log_facility = LOG_LOCAL1

# ================= SSL Options ===============================

# Certificate file to use when starting registry server securely
# cert_file = /path/to/certfile

# Private key file to use when starting registry server securely
```

```
# key_file = /path/to/keyfile

[paste_deploy]
flavor = keystone
root@dublin:/etc/glance# cat glance-registry-paste.ini
# Default minimal pipeline
[pipeline:glance-registry]
pipeline = context registryapp

# Use the following pipeline for keystone auth
# i.e. in glance-registry.conf:
#   [paste_deploy]
#   flavor = keystone
#
[pipeline:glance-registry-keystone]
pipeline = authtoken auth-context registryapp

[app:registryapp]
paste.app_factory = glance.common.wsgi:app_factory
glance.app_factory = glance.registry.api.v1:API

[filter:context]
context_class = glance.registry.context.RequestContext
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = glance.common.context:ContextMiddleware

[filter:authtoken]
paste.filter_factory = keystone.middleware.auth_token:filter_factory
service_protocol = http
service_host = 192.168.10.2
service_port = 5000
auth_host = 192.168.10.2
auth_port = 35357
auth_protocol = http
auth_uri = http://127.0.0.1:5000/
admin_token = ADMIN
admin_tenant_name = service
admin_user = glance
admin_password = glance

[filter:auth-context]
context_class = glance.registry.context.RequestContext
paste.filter_factory = glance.common.wsgi:filter_factory
glance.filter_factory = keystone.middleware.glance_auth_token:KeystoneContextMiddleware
```

Ruterkonfigurasjon

```
!
version 12.4
service timestamps debug datetime msec
service timestamps log datetime msec
service password-encryption
!
hostname SkyRoute
!
boot-start-marker
boot-end-marker
!
enable secret 5 $1$htxS$y.Zw1Uvq/Scqs81z8j.Iy1
!
no aaa new-model
ip cef
!
!
!
!
no ip domain lookup
ip domain name hig.no
!
!
!
!
!
interface FastEthernet0/0
 description link to brighton
 ip address 10.0.0.2 255.255.255.252
 duplex auto
 speed auto
!
interface FastEthernet0/1
 description link to switch
 no ip address
 duplex auto
 speed auto
!
interface FastEthernet0/1.10
 encapsulation dot1Q 10
 ip address 192.168.10.1 255.255.255.0
 ip helper-address 10.0.0.1
!
interface FastEthernet0/1.30
 encapsulation dot1Q 30
```

```
 ip address 192.168.30.1 255.255.255.0
 ip helper-address 10.0.0.1
!
interface FastEthernet0/1.99
 encapsulation dot1Q 99
 ip address 192.168.99.1 255.255.255.0
 ip helper-address 10.0.0.1
!
interface FastEthernet0/1.100
!
interface Serial0/1/0
 no ip address
 shutdown
 clock rate 125000
!
interface Serial0/1/1
 no ip address
 shutdown
 clock rate 125000
!
ip forward-protocol nd
ip route 0.0.0.0 0.0.0.0 10.0.0.1
!
ip http server
!
!
control-plane
!
banner motd ^C Break in, and a kitten will die! ^C
!
line con 0
 exec-timeout 60 0
 password 7 011E030850020D0E2A4D415B495445
 logging synchronous
 login
line aux 0
line vty 0 4
 access-class 10 in
 exec-timeout 15 0
 password 7 10430C150E1E190A07052579747961
 logging synchronous
 login
!
scheduler allocate 20000 1000
end
```

/etc/network/interfaces på kontrollernoden

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
#iface eth0 inet dhcp
iface eth0 inet static
address 192.168.10.2
netmask 255.255.255.0
broadcast 192.168.10.255
network 192.168.10.0
gateway 192.168.10.1
dns-nameservers 10.0.0.1

auto eth1
iface eth1 inet manual
pre-up ifconfig eth1 up
post-down ifconfig eth1 down

auto eth3
iface eth3 inet static
address 192.168.30.2
netmask 255.255.255.0
broadcast 192.168.30.255
network 192.168.30.0
gateway 192.168.30.1
```

/etc/network/interfaces på compute-nodene

```
# The loopback network interface
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
address 192.168.10.10
netmask 255.255.255.0
broadcast 192.168.10.255
network 192.168.10.0
gateway 192.168.10.1
dns-nameservers 10.0.0.1

auto eth1
```

```
iface eth1 inet manual
pre-up ifconfig eth1 up
post-down ifconfig eth1 down
```

/etc/network/interfaces på lagringsnode

```
# The loopback network interface
auto lo
iface lo inet loopback

auto bond0
iface bond0 inet static
address 192.168.10.185
netmask 255.255.255.0
gateway 192.168.10.1
dns-nameservers 8.8.8.8
post-up ifenslave bond0 eth0 eth1
post-down ifenslave -d bond0 eth0 eth1
bond-mode 6
bond-miimon 100
```

H Script og kildekode

Script

batch_setup.sh

```
#!/bin/bash

API="192.168.10.2"      # CHANGE THIS TO THE RIGHT ADDRESS
SQLPW="melkikakao2012" # CHANGE THIS TO THE RIGHT PASSWORD
error=0
login_details_dir="/root/login_details"
if [ ! -d $login_details_dir ]; then
        mkdir $login_details_dir
fi

function prompt()   {
        echo -n "batch> "; read $1
}

# Creates a random password. Provide amount of characters as only argument
function createPassword()   {
        pass=$(cat /dev/urandom | tr -cd "[:alnum:]" | head -c $1)
        echo $pass
}

# Fetches the id of a user, tenant or role. Specify one of those as first
    argument, name as the second
function getKeystoneId()   {
        echo $(keystone $1-list | grep -w $2 | awk '{ print $2 }')
}

# Wait until all instances has ACTIVE state. Param 1 = amount of instances to
    wait for
function waitForBoot()   {
        while [ $(nova list | grep ACTIVE | wc -l) != $1 ]; do
                sleep 3
        done
}

# Create crendentials, so you are able to launch instances in different tenants
```

```
# Param 1 = username; Param 2 = tenant name without number; Param 3 = project
    number; Param 4 = password
function create_rc()   {
        TID=‘getKeystoneId tenant $2$3‘
        ACCESS=$(keystone ec2-credentials-create --user $1 --tenant_id $TID |
            grep access | awk ’{ print $4 }’)
        SECRET=$(keystone ec2-credentials-get --access $ACCESS | grep secret |
            awk ’{ print $4 }’)
        FILE="$login_details_dir/$2/$3/$2$3rc"
        cat > $FILE << EOF
NOVA_API_HOST=$API
GLANCE_API=$API
KEYSTONE_API=$API
KEYSTONE_TENANT=$2$3
KEYSTONE_PASSWORD=$4
KEYSTONE_USERNAME=$1

NOVA_REGION="nova"

export OS_AUTH_USER=\$KEYSTONE_USERNAME
export OS_AUTH_KEY=\$KEYSTONE_PASSWORD
export OS_AUTH_TENANT=\$KEYSTONE_TENANT
export OS_AUTH_URL="http://\${KEYSTONE_API_HOST}:5000/v2.0/"
export OS_AUTH_STRATEGY="keystone"

export OS_USERNAME=\$KEYSTONE_USERNAME
export OS_TENANT_NAME=\$KEYSTONE_TENANT
export OS_PASSWORD=\$KEYSTONE_PASSWORD
export OS_REGION_NAME="RegionOne"

export NOVA_USERNAME=\$KEYSTONE_USERNAME
export NOVA_PROJECT_ID=\$KEYSTONE_TENANT
export NOVA_PASSWORD=\$KEYSTONE_PASSWORD
export NOVA_API_KEY=\$KEYSTONE_PASSWORD
export NOVA_REGION_NAME=\$NOVA_REGION
export NOVA_URL="http://\${NOVA_API_HOST}:5000/v2.0/"
export NOVA_VERSION="1.1"
export EC2_URL="http://\${NOVA_API_HOST}:8773/services/Cloud"
export EC2_ACCESS_KEY=$ACCESS
export EC2_SECRET_KEY=$SECRET
EOF
}

echo "##################################################"
echo "#### Welcome to interactive batch_setup script! ####"
```

```
echo "#################################################"

echo "--- How many projects do you want to make? ---"
prompt tenants

echo "--- Give name prefix for your $tenants new projects ---"
prompt tenant_name_prefix
mkdir $login_details_dir/$tenant_name_prefix
mysql -u root --password=$SQLPW -D dash -e "INSERT INTO batch(navn) VALUES(\"
    $tenant_name_prefix\");"
batch_id=$(mysql -u root --password=$SQLPW -D dash -e "SELECT id FROM batch
    WHERE navn=\"$tenant_name_prefix\"" | egrep '[0-9]+')

echo "--- Do you want to add the admin user to your projects? [y/n] ---"
prompt addadmin

echo "--- Creating $tenants new projects with prefix $tenant_name_prefix ---"
for i in $(seq 1 $tenants); do
        tenant_ids[$i]=$(keystone tenant-create --name $tenant_name_prefix$i --
            enabled true | grep id | awk '{ print $4 }')
        if [ $? -ne 0 ]; then
                error=1
        fi
        if [ $error -eq 1 ]; then
                echo "Error when creating projects, aborting"
                exit 1
        fi
        mysql -u root --password=$SQLPW -D dash -e "INSERT INTO batch_tenants
            VALUES(\"$batch_id\", \"${tenant_ids[$i]}\");"
        if [ $addadmin == "y" ]; then
                keystone user-role-add --user `getKeystoneId user admin` --role
                    `getKeystoneId role member` --tenant_id ${tenant_ids[$i]}
        fi
done
echo "--- Done creating projects! ---"

echo "--- I'll add some generic users with a random password for you. One per
    project ---"
for j in $(seq 1 $tenants); do
        source /root/openstack
        mkdir $login_details_dir/$tenant_name_prefix/$j
        pw=`createPassword 10`
        uname="admin_$tenant_name_prefix$j"
        keyname=$uname"key"
        thistenant=$tenant_name_prefix$j
```

```
        creds=$thistenant"rc"

        echo "--- Creating user $uname ---"
        user_ids[$j]=$(keystone user-create --name $uname --tenant_id ${
            tenant_ids[$j]} --pass $pw --enabled true | grep id | awk '{ print
            $4 }')
        echo -e "Username: $uname \nPassword: $pw" > "$login_details_dir/
            $tenant_name_prefix/$j/$thistenant.txt"

        echo "--- Adding member role ---"
        keystone user-role-add --user ${user_ids[$j]} --role `getKeystoneId role
             member` --tenant_id ${tenant_ids[$j]}

        echo "--- Creating rc-file ---"
        create_rc $uname $tenant_name_prefix $j $pw
        source $login_details_dir/$tenant_name_prefix/$j/$creds

        echo "--- Creating keypair ---"
        nova keypair-add $keyname > $login_details_dir/$tenant_name_prefix/$j/
            $uname".pem"
        chmod 600 $login_details_dir/$tenant_name_prefix/$j/$uname".pem"

        echo "--- Adding ping and ssh access ---"
        nova secgroup-add-rule default tcp 22 22 0.0.0.0/0 > /dev/null
        nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0 > /dev/null
done
echo "--- Done adding users, the login details is stored in $login_details_dir
    ---"

echo "--- You are now ready to login to horizon with your new users. Do you want
     to proceed to spawn instances? [y/n]"
prompt addinst
if [ $addinst == "n" ]; then exit 0;
elif [ $addinst == "y" ]; then
        ok="n"
        while [ $ok != "y" ]; do
                echo "--- This is the list of images you can spawn instances of
                    ---"
                nova image-list

                echo "--- This is the flavors you can choose ---"
                nova flavor-list

                echo "--- Now enter name, image and flavor for each machine.
                    Separated by spaces and in that particular order. Please
```

```
                don't fuck up ——"
            more="y"
            idx=0
            while [ $more == "y" ]; do
                prompt info
                ((idx++))                       # start indexing at 1.
                    When finished idx will contain the amount of
                    instances you will spawn
                allinfo[$idx]=$info
                echo "—— More machines? [y/n] ——"
                prompt more
            done
            echo "—— Will now spawn theese instances in all of $tenants
                projects you've made. Okey? [y/n] ——"
            a=1
            while [ $a -le $idx ]; do
                attr=${allinfo[$a]}
                name[$a]=$(echo $attr | cut -d' ' -f1)
                image[$a]=$(echo $attr | cut -d' ' -f2)
                flavor[$a]=$(echo $attr | cut -d' ' -f3)
                echo "$a, ${name[$a]}, ${image[$a]}, ${flavor[$a]}"
                ((a++))
            done
        read ok  # ADD PROMPT
        done

        echo "—— Booting shit up. Sit down and pray ——"
        for tenant in $(seq 1 $tenants); do
            thistenant=$tenant_name_prefix$tenant
            creds=$thistenant"rc"
            uname="admin_"$thistenant
            keyname=$uname"key"
            source $login_details_dir/$tenant_name_prefix/$tenant/$creds
            for instance in $(seq 1 $idx); do
                echo "—— Booting up ${name[$instance]} in project
                    $thistenant ——"
                nova boot --flavor ${flavor[$instance]} --image ${image[
                    $instance]} --key_name $keyname ${name[$instance]}
            done
            #waitForBoot $idx
        done
else
        echo "Error in input. Exiting..."
        exit 1
fi
```

spawn.sh

```
#!/bin/bash
RCFILE=$1
NAME=$2
FLAVOR=$3
IMAGE="$4"
TENANT=$5

CREDSDIR=$(dirname $RCFILE)
KEYFILE=$TENANT"_"$NAME".pem"
KEYNAME=$NAME"key"

if [ -e $RCFILE ]; then
        source $RCFILE
        nova keypair-add $KEYNAME > $CREDSDIR/$KEYFILE
        chmod 600 $CREDSDIR/$KEYFILE
        nova secgroup-add-rule default tcp 22 22 0.0.0.0/0 > /dev/null
   nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0 > /dev/null
        nova boot --flavor $FLAVOR --image $IMAGE --key_name $KEYNAME $NAME
        IP=$(nova floating-ip-create | egrep '[0-9]+' | cut -d '|' -f2)
        nova add-floating-ip $NAME $IP
else
        exit 1
fi
```

control.sh

```
#!/bin/bash
arguments=2

if [[ $# -lt $arguments ]]; then echo -e "usage: [stop/start/restart] [machines
    ]\n"; exit 0; fi

serv="api network compute cert"
action=$1
shift;
for i in "$@"
 do
   if [[ $action == "stop" ]]; then ssh root@$i "/etc/init.d/libvirt-bin stop;
       killall -9 libvirt"
   else ssh root@$i "killall -9 libvirt; /etc/init.d/libvirt-bin start"; ssh
       root@$i "/etc/init.d/libvirt-bin restart"; fi
   for x in $serv
   do
    ssh root@$i "$action nova-$x"
   done
```

```
done
```

distribute_config.sh

```
#!/bin/bash
servers="manchester newcastle cardiff" # fill this list

for x in $servers; do
        echo "copying to $x"
        ssh root@$x 'cp /etc/nova/nova.conf /etc/nova.conf.bak'
        scp /etc/nova/nova.conf root@$x:/etc/nova/nova.conf
done
```

openstack

Denne filen inneholder miljøvariable som trengs for å kjøre nova, glance og keystone kommandoene.

```
NOVA_API_HOST=192.168.10.2
GLANCE_API_HOST=192.168.10.2
KEYSTONE_API_HOST=192.168.10.2

KEYSTONE_TENANT="openstackDemo"
KEYSTONE_USERNAME="admin"
KEYSTONE_PASSWORD="melkikakao2012"

NOVA_REGION="nova"

# Common Stuff Dont change
export NOVA_USERNAME=$KEYSTONE_USERNAME
export NOVA_PROJECT_ID=$KEYSTONE_TENANT
export NOVA_PASSWORD=$KEYSTONE_PASSWORD
export NOVA_API_KEY=$KEYSTONE_PASSWORD
export NOVA_REGION_NAME=$NOVA_REGION
export NOVA_URL="http://${NOVA_API_HOST}:5000/v2.0/"
export NOVA_VERSION="1.1"

export OS_AUTH_USER=$KEYSTONE_USERNAME
export OS_AUTH_KEY=$KEYSTONE_PASSWORD
export OS_AUTH_TENANT=$KEYSTONE_TENANT
export OS_AUTH_URL="http://${KEYSTONE_API_HOST}:5000/v2.0/"
export OS_AUTH_STRATEGY="keystone"

export EC2_URL="http://${NOVA_API_HOST}:8773/services/Cloud"
export EC2_ACCESS_KEY="371bd4f44b314c2aa221a0a67d70a51c"
export EC2_SECRET_KEY="55c6e508283248b1a96492411606fd60"
export S3_URL="http://${GLANCE_API_HOST}:3333"
```

```
export OS_USERNAME="admin"
export OS_TENANT_NAME="openstackDemo"
export OS_PASSWORD="melkikakao2012"
export OS_REGION_NAME="RegionOne"
export OS_AUTH_STRATEGY="keystone"

#export NOVARC=$(readlink -f "${BASH_SOURCE:-${0}}" 2>/dev/null) ||
    NOVARC=$(python -c 'import os,sys; print os.path.abspath(os.path.realpath(
        sys.argv[1]))' "${BASH_SOURCE:-${0}}")
export NOVA_KEY_DIR="/root/creds"
#export S3_URL="http://192.168.10.2:3333"
export EC2_USER_ID=42 # nova does not use user id, but bundling requires it
export EC2_PRIVATE_KEY=${NOVA_KEY_DIR}/pk.pem
export EC2_CERT=${NOVA_KEY_DIR}/cert.pem
export NOVA_CERT=${NOVA_KEY_DIR}/cacert.pem
#export EUCALYPTUS_CERT=${NOVA_CERT} # euca-bundle-image seems to require this
    set
#alias ec2-bundle-image="ec2-bundle-image --cert ${EC2_CERT} --privatekey ${
    EC2_PRIVATE_KEY} --user 42 --ec2cert ${NOVA_CERT}"
#alias ec2-upload-bundle="ec2-upload-bundle -a ${EC2_ACCESS_KEY} -s ${
    EC2_SECRET_KEY} --url ${S3_URL} --ec2cert ${NOVA_CERT}"
```

Kildekode

All kildekoden er vedlagt i zip-filen som medfølger denne rapporten. Tips, zoom inn for å se detaljene.

Batch.py

```
# vim: tabstop=4 shiftwidth=4 softtabstop=4

#   Code by: SkyHigh
#   Bachelor Thesis written at Gjovik University College
#   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
#
#   This sourcecode has been written as an extension of the Horizon module
#   in the OpenStack project and is greatly inspired by this.
#   http://horizon.openstack.org/
#
#   Licensed under the Apache License, Version 2.0 (the "License"); you may
#   not use this file except in compliance with the License. You may obtain
#   a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0


"""
Batch
"""
import MySQLdb

class Batch():
    id=""
    name =""
    tenant_list = []
    tenant_count = 0
    instance_count = 0

    def __init__(self,request,id,batch_name, tenant_list):
        self.id = id
        self.name = batch_name
        self.tenant_list = tenant_list
        self.tenant_count = len(tenant_list)
        self.instance_count = self.GetInstances(request)

    def GetInstances(self,request):
        db = MySQLdb.connect(host="localhost", port=3306, user="root",\
        passwd="melkikakao2012", db="nova")
        cursor = db.cursor()
        count = 0
        for t in self.tenant_list :
                        cursor.execute("SELECT COUNT(*) FROM instances WHERE \
                        project_id = '%s' AND terminated_at IS NULL AND hostname IS NOT NULL"%t)
            data = cursor.fetchone()
            count = data[0]

        return count
```

Configuration.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17 """
18 Configurations of instances
19 """
20
21 class Config():
22     id=""
23     name=""
24     instance_count=0
25     def __init__(self,id,name,instance_count):
26         self.id = id
27         self.name = name
28         self.instance_count = instance_count
```

db.py

```
# vim: tabstop=4 shiftwidth=4 softtabstop=4

#   Code by: SkyHigh
#   Bachelor Thesis written at Gjovik University College
#   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
#
#   This sourcecode has been written as an extension of the Horizon module
#   in the OpenStack project and is greatly inspired by this.
#   http://horizon.openstack.org/
#
#   Licensed under the Apache License, Version 2.0 (the "License"); you may
#   not use this file except in compliance with the License. You may obtain
#   a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0

#!/usr/bin/python

"""
Database Connection
"""

import MySQLdb

class Mydb():
    db = MySQLdb.connect(host="localhost", port=3306, user="root", passwd="melkikakao2012", db="dash")
```

forms.py

```
# vim: tabstop=4 shiftwidth=4 softtabstop=4
#   Code by: SkyHigh
#   Bachelor Thesis written at Gjovik University College
#   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
#
#   This sourcecode has been written as an extension of the Horizon module
#   in the OpenStack project and is greatly inspired by this.
#   http://horizon.openstack.org/
#
#   Licensed under the Apache License, Version 2.0 (the "License"); you may
#   not use this file except in compliance with the License. You may obtain
#   a copy of the License at
#
#   http://www.apache.org/licenses/LICENSE-2.0

import logging

from django import shortcuts
from django.contrib import messages
from django.utils.translation import ugettext as _

from horizon import forms
from horizon import api
from horizon.views.auth_forms import Login, LoginWithTenant, _set_session_data
#from .views import Batch
import random
from .db import Mydb
from .TmpInstance import Tmp_Instance
import thread

LOG = logging.getLogger(__name__)

"""
Following classes defines and configures the forms used for data input for
their respective classes.

- Variables represent fields in forms as long as their specified by parent.
- Handle is selfexplanatory
- Other methods have been specified to keep with conventions.
"""
class SaveConfig(forms.SelfHandlingForm):
    name = forms.CharField(max_length=80, label=_("Config Name"))


    def handle(self, request, data):
                self.save(request, data)
        msg = _('%s was successfully saved to configs' % data['name'])
        LOG.info(msg)
        messages.success(request, msg)
        return shortcuts.redirect("horizon:syspanel:batch_setup:index")

        def save(self, request, data):
            cursor = Mydb.db.cursor()
            cursor.execute("INSERT INTO configs(name) VALUES('%s')" % data['name']) # Store config
            Mydb.db.commit()
            cursor.execute("SELECT LAST_INSERT_ID() AS id FROM configs") # Get id of new config
            res = cursor.fetchone()
            config_id = int(res[0])
            instances = request.session['cur_instances'] # Get instances

            for instance in instances:              # Iterate trough every instance
                cursor.execute("INSERT INTO instance_config(config_id, name, image_id, \
                                "image_name, flavor_id, flavor_name)
                                VALUES(%d, '%s', '%s', '%s', %d, '%s')" % (config_id,
                                                                           instance.name,
                                                                           instance.image_id,
                                                                           instance.image_name,
                                                                           int(instance.flavor_id),
                                                                           instance.flavor_name))
                Mydb.db.commit()
            cursor.close()


"""
Complex class, see sequence diagram in thesis:https://github.com/hwestman/Horizon-Skyhigh

"""
class CreateBatch(forms.SelfHandlingForm):
```

```
    name = forms.CharField(max_length=80, label=_("Batch Name"))
    tenant_count = forms.IntegerField(label=_("Tenant Count"),
                            required=True,
                            min_value=1,
                            initial=1,
                            help_text=_("Number of tenants to launch."))
    """
    Generates a pseudorandom password alphanumeric string
    :param 'length' length of pseudorandom password
    """
    def gen_pass(self, length):
        from random import choice
        import string
        chars = string.letters + string.digits
        pw = ''
        for i in range(length):
            pw = pw + choice(chars)
        return pw

    """
    Saves the generated username and password to a separate file
    for each created tenant
    """
    def save_creds_to_file(self, request, batch_name, tenant_name, username, password):
        import os
        d = "/root/login_details/%s" % batch_name
        if not os.path.exists(d):
            try:
                os.mkdir(d, 0700)
            except OSError:
                msg =_('Error creating creds directory in %s') % d
                LOG.info(msg)
                messages.warning(request, msg)

        path = "/root/login_details/%s/%s" % (batch_name, tenant_name)
        try:
            f = open(path, 'w')
            creds = ("Username: %s \nPassword: %s") % (username, password)
            f.write(creds)
            f.close()
        except IOError:
            msg = _("Unable to write creds for %s to file") % tenant_name
            LOG.info(msg)
            messages.warning(request, msg)

        return self.save_rc_file(username, batch_name, tenant_name, password)


    def save_rc_file(self, user, batch, tenant, pw):
        try:
            path = "/root/login_details/%s/%src" % (batch, tenant)
            f = open(path, 'w')
            rc = "NOVA_API_HOST=192.168.10.2\n \
                  GLANCE_API=192.168.10.2\n \
                  KEYSTONE_API_HOST=192.168.10.2\n \
                  KEYSTONE_TENANT=%s\n \
                  KEYSTONE_PASSWORD=%s\n \
                  KEYSTONE_USERNAME=%s\n \
                  NOVA_REGION='nova'\n \
                  export OS_AUTH_USER=$KEYSTONE_USERNAME\n \
                  export OS_AUTH_KEY=$KEYSTONE_PASSWORD\n \
                  export OS_AUTH_TENANT=$KEYSTONE_TENANT\n \
                  export OS_AUTH_URL=\"http://${KEYSTONE_API_HOST}:5000/v2.0/\"\n \
                  export OS_AUTH_STRATEGY='keystone'\n \
                  export OS_USERNAME=$KEYSTONE_USERNAME\n \
                  export OS_TENANT_NAME=$KEYSTONE_TENANT\n \
                  export OS_PASSWORD=$KEYSTONE_PASSWORD\n \
                  export OS_REGION_NAME='RegionOne'\n \
                  export NOVA_USERNAME=$KEYSTONE_USERNAME\n \
                  export NOVA_PROJECT_ID=$KEYSTONE_TENANT\n \
                  export NOVA_PASSWORD=$KEYSTONE_PASSWORD\n \
                  export NOVA_API_KEY=$KEYSTONE_PASSWORD\n \
                  export NOVA_REGION_NAME=$NOVA_REGION\n \
                  export NOVA_URL=\"http://\${NOVA_API_HOST}:5000/v2.0/\"\n \
                  export NOVA_VERSION='1.1'\n \
                  export EC2_URL=\"http://\${NOVA_API_HOST}:8773/services/Cloud\"\n"
                  % (tenant, pw, user)
            f.write(rc)
```

```
157             f.close()
158             return path
159         except IOError:
160             msg = _("Unable to write creds for %s to file") % tenant
161         LOG.info(msg)
162
163     def create_instances(self, rcfile, instances, tenant):
164         import os
165         for instance in instances:
166             os.system("/bin/bash /root/scripts/batch/spawn.sh %s %s %s %s %s"
167             % (rcfile, instance.name, instance.flavor_name, instance.image_name, tenant))
168
169
170
171     def lots_of_tenants(self, request, name,batchid, amount):
172
173         tenant_list = []
174
175         for i in range (amount):
176
177             t_name = name+str(i+1)
178             uname = "admin_"+t_name
179             l = api.keystone.role_list(request)
180             for role in l:
181                 if role.name == "member":
182                     role_id=role.id
183                     break
184             tmpTenant = api.keystone.tenant_create(request, t_name, "", True)
185             tenant_list.append(tmpTenant.id)
186             pw = self.gen_pass(10)
187             tmpUsr = api.keystone.user_create(request,
188                                 uname,
189                                 "",
190                                 pw,
191                                 tmpTenant.id,
192                                 True)
193             #               Add the new user to the new tenant
194             api.keystone.add_tenant_user_role(request,
195                                 tmpTenant.id,
196                                 tmpUsr.id,
197                                 role_id)
198             #               Add admin to the new tenant
199             api.keystone.add_tenant_user_role(request,
200                                 tmpTenant.id,
201                                 request.user.id,
202                                 role_id)
203
204
205             rcpath = self.save_creds_to_file(request, name, t_name, uname, pw)
206             instances = request.session['cur_instances']
207             self.create_instances(rcpath, instances, t_name)
208
209         self.batch_to_db(request,name,batchid,tenant_list)
210
211
212     def batch_to_db(self,request,name,batchid,tenant_list):
213
214         cursor = Mydb.db.cursor()
215
216         for tenant in tenant_list:
217             cursor.execute("INSERT INTO batch_tenants (batch_id,tenant_id) \
218             VALUES ('%s','%s')"%(batchid, tenant))
219             Mydb.db.commit()
220             LOG.info("tenant: %s"% tenant)
221
222     def handle(self, request, data):
223
224         cursor = Mydb.db.cursor()
225
226         cursor.execute("SELECT COUNT(*) FROM batch")
227         result = cursor.fetchone()
228         batchid = result[0]+1
229
230         LOG.info("batchid %s"% batchid)
231         cursor.execute("INSERT INTO batch (id,navn) VALUES ('%s','%s')"%(batchid, data['name']))
232         Mydb.db.commit()
233
234
```

```
235         def runCreate(threadName):
236             self.lots_of_tenants(request, data['name'],batchid, data['tenant_count'])
237 
238         try:
239             thread.start_new_thread(runCreate,("TheThread",))
240         except:
241             LOG.info("thread couldnt fork")
242 
243         msg = _('%s was successfully added to batches.') % data['name']
244         LOG.info(msg)
245         messages.success(request, msg)
246         return shortcuts.redirect('horizon:syspanel:batch_setup:index')
247 
248 class AddInstance(forms.SelfHandlingForm):
249     name = forms.CharField(max_length=80, label=_("Server Name"))
250     image = forms.ChoiceField(label=_("Image"),
251                               help_text=_("Which image to launch from"))
252 
253     image = forms.ChoiceField(label=_("Image"),
254                               help_text=_("Type of image."))
255     flavor = forms.ChoiceField(label=_("Flavor"),
256                               help_text=_("Size of image to launch."))
257 
258     security_groups = forms.MultipleChoiceField(
259                               label=_("Security Groups"),
260                               required=True,
261                               initial=["default"],
262                               widget=forms.CheckboxSelectMultiple(),
263                               help_text=_("Launch instance in these "
264                                           "security groups."))
265 
266     def __init__(self, *args, **kwargs):
267         flavor_list = kwargs.pop('flavor_list')
268         image_list = kwargs.pop('image_list')
269         security_group_list = kwargs.pop('security_group_list')
270 
271         super(AddInstance, self).__init__(*args, **kwargs)
272 
273         self.fields['flavor'].choices = flavor_list
274         self.fields['image'].choices = image_list
275         self.fields['security_groups'].choices = security_group_list
276 
277     def handle(self, request, data):
278 
279         image_name = api.glance.image_get_meta(request, self.data['image']).name
280         flavor_name = api.nova.flavor_get(request, self.data['flavor']).name
281         LOG.info("Here comes the flavor_name %s"%flavor_name)
282         list = request.session['cur_instances']
283         list.append(Tmp_Instance(str(len(list)+1),data['name'], data['image'],
284         image_name,data['flavor'],flavor_name))
285         request.session['cur_instances'] = list
286 
287 
288         msg = _('%s was successfully added to instances.'% data['name'])
289         LOG.info(msg)
290         messages.success(request, msg)
291         return shortcuts.redirect("horizon:syspanel:batch_setup:index")
292 
293 class EditBatch(forms.SelfHandlingForm):
294     name = forms.CharField(max_length=80, label=_("blabla"))
295 
296 
297     def handle(self, request, data):
298 
299         msg = _('%s was successfully added to instances.'% data['name'])
300         LOG.info(msg)
301         messages.success(request, msg)
302         return shortcuts.redirect("horizon:syspanel:batch_setup:index")
```

panel.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17 import horizon
18 from horizon.dashboards.syspanel import dashboard
19
20
21 class Batch_setup(horizon.Panel):
22     name = "Batch Setup"
23     slug = 'batch_setup'
24     roles = ('admin',)
25
26 dashboard.Syspanel.register(Batch_setup)
```

tables.py

```
183     class Meta:
184         name = "tenant_overview"
185         verbose_name = _("Tenants")
```

Tenant.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17 """
18 Tenats
19 """
20 class Tenant():
21     id = ""
22     name = ""
23     def __init__(self,id):
24         self.id = str(id)
```

TmpInstance.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17 """
18 Class for instances that is used temporary while defining configurations
19 """
20
21 class Tmp_Instance():
22     id = ""
23     name=""
24     flavor = ""
25     image_name = ""
26     image_id = ""
27     flavor_id = ""
28     flavor_name=""
29     def __init__(self,id,instance_name,image_id,image_name,flavor_id,flavor_name):
30
31         self.id = id
32         self.name = instance_name
33         self.image_id = image_id
34         self.image_name = image_name
35         self.flavor_id = flavor_id
36         self.flavor_name = flavor_name
```

urls.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17 from django.conf.urls.defaults import patterns, url
18 from .views import IndexView, CreateBatchView, AddInstanceView, EditBatchView, SaveConfigView
19
20 urlpatterns = patterns('horizon.dashboards.syspanel.batch_setup.views',
21     url(r'^$', IndexView.as_view(), name='index'),
22     url(r'^create_batch/$', CreateBatchView.as_view(), name='create_batch'),
23     url(r'^save_config/$', SaveConfigView.as_view(), name='save_config'),
24     url(r'^add_instance/$', AddInstanceView.as_view(), name='add_instance'),
25     url(r'^(?P<batch_id>[^/]+)/edit_batch/$', EditBatchView.as_view(), name='edit_batch'),)
```

views.py

```
1  # vim: tabstop=4 shiftwidth=4 softtabstop=4
2
3  #   Code by: SkyHigh
4  #   Bachelor Thesis written at Gjovik University College
5  #   http://hovedprosjekter.hig.no/v2012/imt/in/skyhighadm/
6  #
7  #   This sourcecode has been written as an extension of the Horizon module
8  #   in the OpenStack project and is greatly inspired by this.
9  #   http://horizon.openstack.org/
10 #
11 #   Licensed under the Apache License, Version 2.0 (the "License"); you may
12 #   not use this file except in compliance with the License. You may obtain
13 #   a copy of the License at
14 #
15 #   http://www.apache.org/licenses/LICENSE-2.0
16
17
18 from horizon import tables
19 from horizon import api
20 from .tables import BatchOverview, InstanceSetup, ConfigOverview
21 from .forms import CreateBatch, AddInstance, EditBatch, SaveConfig
22 from horizon import forms
23 import gc
24 import logging
25 from horizon import api, exceptions
26 from datetime import datetime
27 from .Batch import Batch
28 from .Configuration import Config
29 from .db import Mydb
30 import MySQLdb
31
32 LOG = logging.getLogger(__name__)
33
34 """
35 View that displays three tables:
36 - batches
37 - current batch configuration
38 - all batch configuration
39 presentet as tabledata
40 """
41 class IndexView(tables.MultiTableView):
42     table_classes = (BatchOverview, InstanceSetup, ConfigOverview)
43     template_name = 'syspanel/batch_setup/index.html'
44
45
46     """
47     Collects data for all batches from database
48     returns for rendering as a list
49     """
50     def get_batch_overview_data(self):
51         list = []
52         cursor = Mydb.db.cursor()
53
54         cursor.execute("SELECT id, navn FROM batch")
55         data = cursor.fetchall()
56         for row in data :
57                     tenant_list = []
58                     cursor.execute("SELECT tenant_id FROM batch_tenants \
59                     WHERE batch_id=%s"%row[0])
60                     tid = cursor.fetchall()
61                     for line in tid :
62                         tenant_list.append(line[0])
63                     list.append(Batch(self.request,str(row[0]),row[1],tenant_list))
64                 gc.collect()
65         return list
66
67     """
68     Collects data for all instances that has been added from session
69     Returns list for rendering
70     """
```

```
    def get_instance_setup_data(self):

        list = []
        if('cur_instances' in self.request.session):
            tmp_instance_list = self.request.session.get('cur_instances')
            list=tmp_instance_list

        else:
            self.request.session['cur_instances'] = []

        return list

    """
    Collects data for all configurations from database
    returns for rendering as a list
    """
    def get_config_overview_data(self):
        list = []
        cursor = Mydb.db.cursor(MySQLdb.cursors.DictCursor)

        cursor.execute("SELECT c.id, c.name, (SELECT count(*) \
        FROM instance_config i WHERE i.config_id = c.id) AS instances FROM configs c;")
        data = cursor.fetchall()
        for row in data:
            list.append(Config(str(row["id"]),row["name"],row["instances"]))
            LOG.info("rows in config = : %s"%row["name"])

        return list


"""
Form for creating batches
warns user for overloading the system based on usage
"""
class CreateBatchView(forms.ModalFormView):
    form_class = CreateBatch
    template_name = 'syspanel/batch_setup/create_batch.html'

        def get_context_data(self, **kwargs):
        context = super(CreateBatchView, self).get_context_data(**kwargs)
        try:
                    usage_list = api.nova.usage_list(self.request,
                    datetime(1970,1,1), datetime.today())
                    totals = {
                        'vcpus' : 0,
                        'ram' : 0
                    }
                    for usage in usage_list:
                        totals['vcpus'] += usage.vcpus
                        totals['ram'] += usage.memory_mb
                    context['totals'] = totals

        except:
                        exceptions.handle(self.request)
        return context

"""
Form for saving a configuration of instances
"""
class SaveConfigView(forms.ModalFormView):
    form_class = SaveConfig
    template_name = 'syspanel/batch_setup/save_config.html'

    def get_context_data(self, **kwargs):
        context = super(SaveConfigView, self).get_context_data(**kwargs)

        return context

"""
```

```
The view for editing a specific batch
Collects data from database
"""
class EditBatchView(forms.ModalFormView):
    form_class = EditBatch
    template_name = 'syspanel/batch_setup/edit_batch.html'
    context_object_name = 'batch_setup'
    def get_initial(self):

        return {'batch_id': self.kwargs['batch_id']}

    def get_object(self, *args, **kwargs):

        cursor = Mydb.db.cursor()

        cursor.execute("SELECT * FROM batch where id = %s"%kwargs["batch_id"])
        data = cursor.fetchone()
        #LOG.info("i logged: %s"%str(data[1]))
        cursor.execute("SELECT tenant_id FROM batch_tenants WHERE batch_id=%s"%data[0])

        tenants = cursor.fetchall()
        list = []
        for line in tenants :
            LOG.info("tenants is: %s"%str(line[0]))
            list.append(line[0])

        return list

"""
View for adding instances to a configuration
"""
class AddInstanceView(forms.ModalFormView):
    form_class = AddInstance
    template_name = 'syspanel/batch_setup/add_instance.html'

    """
    Specifies arguments for specific instances
    """
    def get_form_kwargs(self):
        kwargs = super(AddInstanceView, self).get_form_kwargs()
        kwargs['flavor_list'] = self.flavor_list()
        kwargs['image_list'] = self.image_list()
        kwargs['security_group_list'] = self.security_group_list()
        return kwargs

    def get_context_data(self, **kwargs):
        context = super(AddInstanceView, self).get_context_data(**kwargs)
        try:
            context['usages'] = api.tenant_quota_usages(self.request)
        except:
            exceptions.handle(self.request)
        return context
    """
    Following methods specifies options for all arguments in an instance
    """
    def flavor_list(self):
        display = '%(name)s (%(vcpus)sVCPU / %(disk)sGB Disk / %(ram)sMB Ram )'
        try:
            flavors = api.flavor_list(self.request)
            flavor_list = [(flavor.id, display % {"name": flavor.name,
                                                  "vcpus": flavor.vcpus,
                                                  "disk": flavor.disk,
                                                  "ram": flavor.ram})
                                                 for flavor in flavors]
        except:
            flavor_list = []
            exceptions.handle(self.request,
                              _('Unable to retrieve instance flavors.'))
        return sorted(flavor_list)
```

```
209     def security_group_list(self):
210         try:
211             groups = api.security_group_list(self.request)
212             security_group_list = [(sg.name, sg.name) for sg in groups]
213         except:
214             exceptions.handle(self.request,
215                               _('Unable to retrieve list of security groups'))
216             security_group_list = []
217         return security_group_list
218 
219     def image_list(self):
220         try:
221             all_images = api.image_list_detailed(self.request)
222             images = [(image.id,image.name)for image in all_images]
223 
224         except:
225             images = []
226             exceptions.handle(self.request, _("Unable to retrieve images."))
227 
228 
229         return images
```

templates/_add_instance.html

```
{% extends "horizon/common/_modal_form.html" %}
{% load i18n %}

{% block form_id %}add_instance_form{% endblock %}
{% block form_action %}{% url horizon:syspanel:batch_setup:add_instance %}{% endblock %}

{% block modal_id %}add_instance_modal{% endblock %}
{% block modal-header %}{% trans "Add Instance" %}{% endblock %}

{% block modal-body %}

    <h3>{% trans "Add Instances" %}</h3>
    <fieldset>
    {% include "horizon/common/_form_fields.html" %}
    </fieldset>
{% endblock %}

{% block modal-footer %}
  <input class="btn btn-primary pull-right" type="submit" value="{% trans "Add Instance" %}" />
  <a href="{% url horizon:syspanel:batch_setup:index %}" class="btn secondary cancel close">{% trans "Cancel" %}</a>
{% endblock %}
```

templates/add_instance.html

```
 1 {% extends 'syspanel/base.html' %}
 2 {% load i18n %}
 3 {% block title %}Add Instance{% endblock %}
 4
 5 {% block page_header %}
 6   {% include "horizon/common/_page_header.html" with title=_("Add Instance") %}
 7 {% endblock page_header %}
 8
 9 {% block syspanel_main %}
10     {% include "syspanel/batch_setup/_add_instance.html" %}
11 {% endblock %}
```

templates/_create_batch.html

```
{% extends "horizon/common/_modal_form.html" %}
{% load i18n %}

{% block form_id %}create_batch_form{% endblock %}
{% block form_action %}{% url horizon:syspanel:batch_setup:create_batch %}{% endblock %}

{% block modal_id %}create_batch_modal{% endblock %}
{% block modal-header %}{% trans "Create Batch" %}{% endblock %}

{% block modal-body %}
<div class="left">
    <h3>{% trans "Add Instances" %}</h3>
    <fieldset>
        {% include "horizon/common/_form_fields.html" %}
    </fieldset>
</div>
<div class="right">
    <h3>{% trans "Current usage" %}</h3>
    <p>
        {% trans "You currently have " %}
        <strong><span> {{ totals.vcpus }} </span></strong>
        {% trans "active VCPUs and " %}
        <strong><span>{{ totals.ram }}</span></strong>
        {% trans "MB of used RAM in your entire cloud. Please don't exceed a total of 8192MB RAM" %}


</div>
{% endblock %}

{% block modal-footer %}
  <input class="btn btn-primary pull-right" type="submit" value="{% trans "Create Batch" %}" />
  <a href="{% url horizon:syspanel:batch_setup:index %}" class="btn secondary cancel close">{% trans "Cancel" %}</a>
{% endblock %}
```

templates/create_batch.html

```
 1 {% extends 'syspanel/base.html' %}
 2 {% load i18n %}
 3 {% block title %}Create Batch{% endblock %}
 4
 5 {% block page_header %}
 6   {% include "horizon/common/_page_header.html" with title=_("Create Batch") %}
 7 {% endblock page_header %}
 8
 9 {% block syspanel_main %}
10     {% include "syspanel/batch_setup/_create_batch.html" %}
11 {% endblock %}
```

templates/_edit_batch.html

```
{% extends "horizon/common/_modal_form.html" %}
{% load i18n %}

{% block form_id %}edit_batch_form{% endblock %}
{% block form_action %}{% url horizon:syspanel:batch_setup:edit_batch batch_id%}{% endblock %}

{% block modal-header %}{% trans "Edit Batch" %}{% endblock %}

{% block modal-body %}
<div class="left">
    <fieldset>
    {% include "horizon/common/_form_fields.html" %}
    </fieldset>
</div>
<div class="right">
    <h3>{% trans "Select from dropdown and remove" %}:</h3>
</div>
{% endblock %}

{% block modal-footer %}
    <input class="btn btn-primary pull-right" type="submit" value="{% trans "Edit batch" %}" />
  <a href="{% url horizon:syspanel:batch_setup:index %}" class="btn secondary cancel close">{% trans "Cancel" %}</a>
{% endblock %}
```

templates/edit_batch.html

```
 1 {% extends 'syspanel/base.html' %}
 2 {% load i18n %}
 3 {% block title %}Edit Batch{% endblock %}
 4
 5 {% block page_header %}
 6   {% include "horizon/common/_page_header.html" with title=_("Edit Batch") %}
 7 {% endblock page_header %}
 8
 9 {% block syspanel_main %}
10     {% include 'syspanel/batch_setup/_edit_batch.html' with form=form %}
11 {% endblock %}
```

templates/index.html

```
{% extends 'syspanel/base.html' %}
{% load i18n %}
{% block title %}Batch Setup{% endblock %}

{% block page_header %}
  {% include "horizon/common/_page_header.html" with title=_("Batch Setup") %}
{% endblock page_header %}
{% block syspanel_main %}
  <div id="batch_overview">
      {{ batch_overview_table.render }}
  </div>

  <div id="instance_setup">
      {{ instance_setup_table.render }}
  </div>

  <div id="config_overview">
      {{ config_overview_table.render }}
  </div>

{% endblock %}
```

templates/_save_config.html

```
{% extends "horizon/common/_modal_form.html" %}
{% load i18n %}

{% block form_id %}save_config_form{% endblock %}
{% block form_action %}{% url horizon:syspanel:batch_setup:save_config %}{% endblock %}

{% block modal_id %}create_batch_modal{% endblock %}
{% block modal-header %}{% trans "Save Config" %}{% endblock %}

{% block modal-body %}
<div class="left">
    <h3>{% trans "Configuration" %}</h3>
    <fieldset>
        {% include "horizon/common/_form_fields.html" %}
    </fieldset>
</div>
<div class="right">
    <h3>{% trans "Arbitrary" %}</h3>
    <p>Arbitrary</p>

</div>
{% endblock %}

{% block modal-footer %}
  <input class="btn btn-primary pull-right" type="submit" value="{% trans "Save Config" %}" />
  <a href="{% url horizon:syspanel:batch_setup:index %}" class="btn secondary cancel close">{% trans "Cancel" %}</a>
{% endblock %}
```

templates/save_config.html

```
 1 {% extends 'syspanel/base.html' %}
 2 {% load i18n %}
 3 {% block title %}Save Config{% endblock %}
 4
 5 {% block page_header %}
 6   {% include "horizon/common/_page_header.html" with title=_("Save Config") %}
 7 {% endblock page_header %}
 8
 9 {% block syspanel_main %}
10     {% include "syspanel/batch_setup/_save_config.html" %}
11 {% endblock %}
```