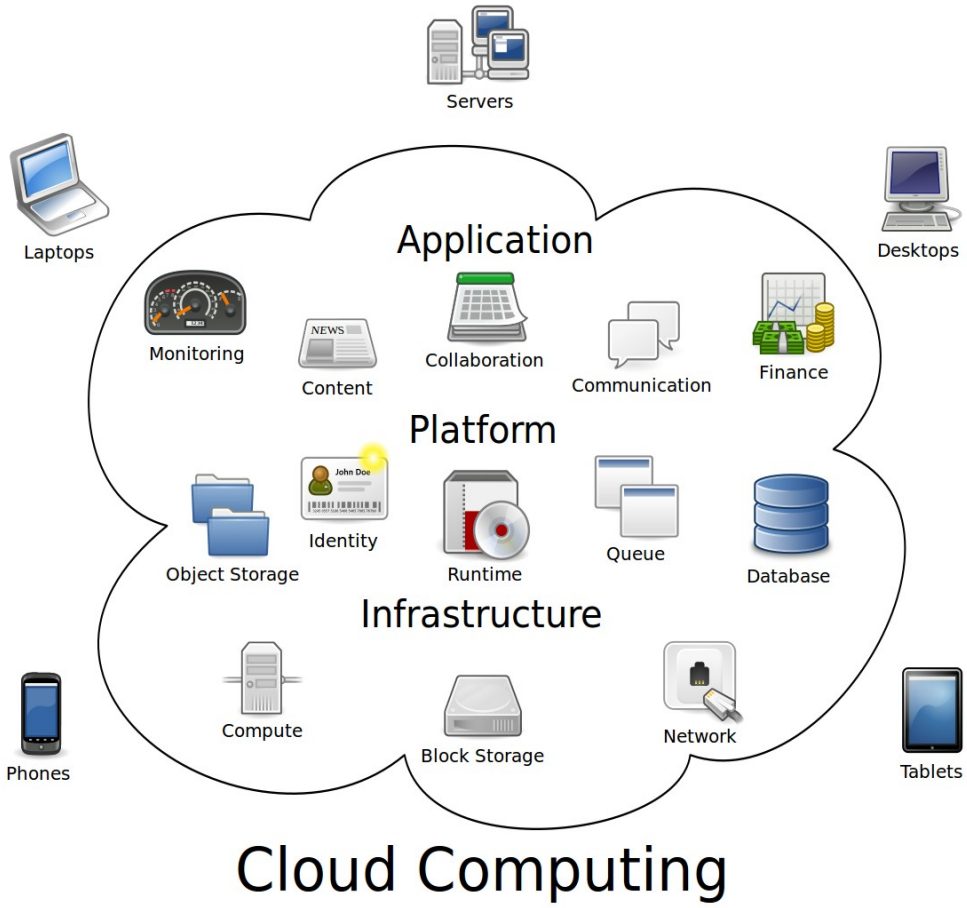
Monitorering av OpenStack ved UiB

1. Presentasjon av tema

I min oppgave ønsker jeg å se nærmere på OpenStack-rammeverket og jeg vil belyse dette knyttet til erfaringer fra Universitet- og Høgskolesektorens sky-prosjekt, hvor jeg har hatt utplassering høsten 2014. Jeg vil også følge dette prosjektet videre i forbindelse med mitt arbeid med oppgaven. Under følger en utgreiing av ulike deler som vil være sentrale i min oppgave.

Sky-teknologi

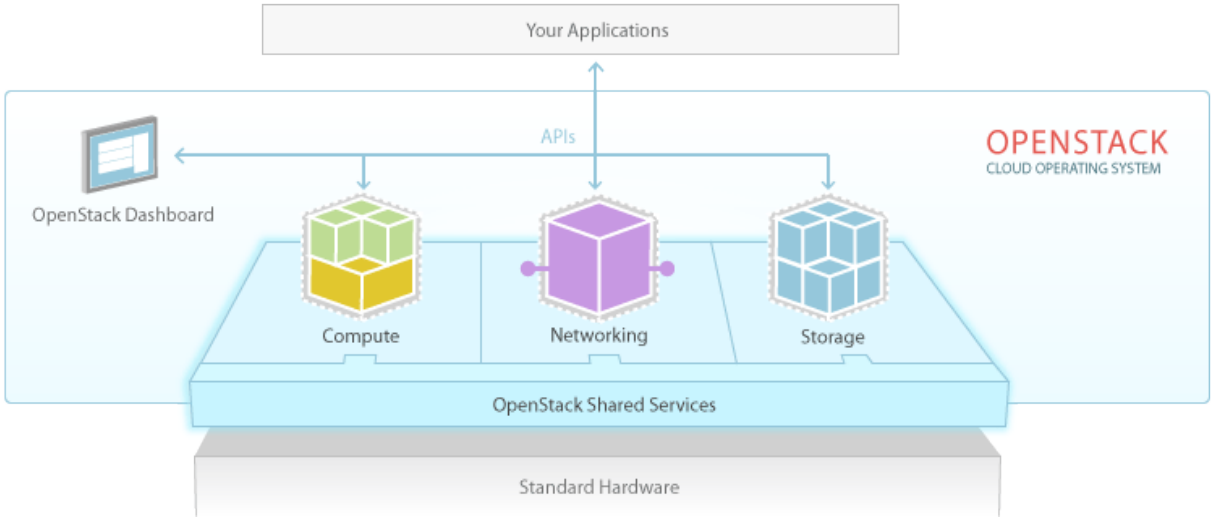
Sky-teknologi er en betegnelse for store grupper med servere som står i eksterne sentraliserte dataparker. I en slik datapark vil både bedrifter og privatpersoner ha mulighet til å leie prosessorkraft, lagring, nettverk og eventuelt andre maskinressurser etter behov. Nettskyer er ofte selvskalerende og det er mange selskaper som tilbyr sky-tjenester for sine kunder. Det finnes i dag mange store aktører innenfor sky-tjenester, blant annet Amazon Elastic Compute Cloud, Microsoft Azure og OpenStack, UH sky-prosjektet på Universitetet i Bergen tar for seg å implementere en Infrastruktur som tjeneste plattform basert på OpenStack. I utplasseringsfaget DAT156 har jeg fått være med på dette prosjektet og vil i de påfølgende avsnitt greie ut om OpenStack (Woodford, 2009).



Cloud Computing (Johnston, 2009).

OpenStack

OpenStack er et åpent kildekoderammeverk som styrer store mengder med compute, lagring og nettverksressurser gjennom et datasenter. Alt av OpenStack-komponenter styres ved hjelp av et webbasert kontrollpanel som gir brukere tilgang til og provisjonere de ressursene de trenger i form av virtuelle maskiner. I tillegg til å bli kunne styrt gjennom kontrollpanelet kan man også bruke OpenStack sitt innebyggede API. OpenStack består av tre hovedkomponenter. Compute, nettverk og lagring. Computenoden og dens tjenester er den komponenten som er ansvarlig for å styre de virtuelle maskinene og tilbyr funksjonalitet i forhold til dette. Nettverksnoden holder kontroll på de forskjellige nettverkene som støttes og viser en hierarkisk oversikt over hvordan nettverkene er bygget opp innad i skyen. Lagringsnoden tilbyr lagring som de andre nodene i OpenStack skal kunne benytte seg av. Lagringen er delt opp i to større deler, objektbasert lagring og den mer tradisjonelle lagringen som er blokkbasert. OpenStack har jeg vært med på å spesialtilpasse i UH sky-prosjektet (openstack.org, 2014).



(openstack-software-diagram.png, u.d)

Universitet- og Høgskolesektorens sky-prosjekt

Prosjektet går ut på å implementere en infrastruktur som tjenesteplattform (IaaS) som baserer seg på OpenStack. Plattformen blir en byggeblokk for framtidige leveranser av IT-infrastruktur i sektoren og vil i utgangspunktet ha en total kapasitet på 750 små instanser eller 275 store instanser samt 100TB med lagring fordelt på 3 geografiske spredte lokasjoner. En liten instans er definert som en virtuell maskin med 1vCPU, 4GB RAM og 10GB harddisk. En stor instans har derimot 4vCPU, 16GB RAM og 100GB harddisk. De ulike lokasjonene har allerede blitt bestemt. Deriblant Bergen (UIB), Trondheim (NTNU) og Oslo (UIO). På den endelige plattformen er det også blitt beskrevet forskjellige brukstilfeller en bruker av systemet har lov til å utføre selv. Dette går ut på at en bruker for eksempel skal kunne opprette egne instanser, legge til ulike ressurser i form av minne, CPU og harddiskstørrelse i tillegg til å legge instanser på tre forskjellige lokasjoner. Samtidig skal brukeren kunne overvåke sitt eget ressursbruk ut ifra tildelte ressurser. For å se en fullstendig liste over hva brukstilfeller en bruker skal kunne utføre selv, se link til UH prosjektbeskrivelse i litteraturavsnittet.

Status på prosjektet per 5.11.14 er at det har blitt laget et automatisert virtualisert testmiljø som oppretter de ulike komponentene som utgjør en OpenStack-plattform. Testmiljøet er ment som en *«proof of concept»* der vi installerer og tester de ulike teknologiene som ligger i plattformen. Dette for å se at testmiljøet gir den funksjonaliteten som er ønskelig og at målene for prosjektet blir nådd. Samtidig ble også den fysiske infrastrukturen for prosjektet nylig bestilt. At prosjektet skal være avsluttet til den 15.06.15 er avhengig at innkjøpet av

utstyr var gjort på denne siden av året. Hvis ikke dette hadde skjedd før nyttår ville prosjektet har pågått til over sommeren neste år. Leveringsdato for utstyret nærmer seg med stormskritt. I løpet av desember/januar vil prosjektet gå inn i en ny fase der en vil begynne å teste på den faktiske infrastrukturen som den endelige plattformen skal kjøre på (uninett.no, 2014).

Min rolle i prosjektet har i hovedsak gått ut på å bygge opp det automatiserte testmiljøet der vi kan kjøre OpenStack. Dette innebærer at jeg har fått lære hvordan de forskjellige komponentene er bygget opp og hvordan det vil fungere sammen i praksis. I tillegg har det blitt gjort mye feilsøking og feilretting underveis som har gitt meg større forståelse og teknisk innsikt innenfor OpenStack-rammeverket. Oppbyggingen av testmiljøet har vært i samarbeid med min kontaktperson på UIB som er en av ressursene som er med på fulltid i sky-prosjektet. Bachelorprosjektet som jeg skal jobbe med frem mot juni 2015 vil være en leveranse til sky-prosjektet. Problemstillingen som vi har utarbeidet i forbindelse med dette beskriver hva som skal leveres og hvordan dette skal foregå. Relevante kontaktpersoner og prosjekteier for UH sky-prosjektet er som følger:

- Tor Lædre – Prosjektdeltaker (kontaktperson) fra IT-avdelingen ved Universitetet i Bergen.
    - Epost: Tor.ladre@it.uib.no
    - Tlf: 95801892
- Jan Ivar Beddari – Prosjektleder for UH-sky
    - Epost: janivar@beddari.net
    - Tlf: 45066782

2. Problemstillinger

Ettersom dette er et prosjekt som ikke er fullført enda, vil jeg ikke ha mulighet til å se hvilken betydning prosjektet vil ha for sluttbrukere i praksis. Jeg vil derimot se på forskjellige monitoreringsløsninger for å finne ut hva som kan være det beste verktøyet for UH sky-prosjektet. Monitorering er et sentralt og viktig område innenfor skytjenester. Hvordan kan vi lage trender for å se endringer i et system over tid? Hvordan får vi ut korrekt informasjon når vi overvåker et såpass stort og komplekst rammeverk? På hva måte er det mest hensiktsmessig å overvåke de forskjellige komponentene og tjenestene for å sørge for at de fungerer som de skal til enhver tid? Jeg vil i løpet av bachelorprosjektet svare på følgende problemstilling med under-problemstillinger:

- Se på ulike monitoreringsløsninger som kan brukes til å overvåke OpenStack-rammeverket i Universitets- og Høgskolesektorens sky-prosjekt. Dette innebærer:
    - Kartlegge forskjellige monitoreringsverktøy og teste bruken av disse
    - Belyse fordeler og ulemper med forskjellige overvåkningsverktøy. Hva passer best til vårt bruk? Er noen verktøy bedre for sky enn for tradisjonell bruk?
    - Se på oppbyggingen av OpenStack-rammeverket for å kartlegge hvilke komponenter som er kritiske å overvåke i forhold til andre tjenester som ikke er kritiske.
    - Hvordan kan unyttige data som trace og debug filtreres for å få fram informasjonen som er nødvendig å vite?
    - Kartlegge eventuelle etiske og personvernsspørsmål knyttet til monitorering av OpenStack.
    - Se på skillelinjer mellom hva en bruker av systemet har lov til å se og gjøre i forhold til hva en operatør / administrator av systemet har lov til å se og gjøre.

3. Teori

Infrastruktur som tjeneste (Iaas)

Muligheten til å leie prosessorkraft, lagring, nettverk og andre fundamentale maskinressurser hvor kunden er i stand til å opprette, lagre data og/eller kjøre vilkårlig programvare. Dette inkluderer operativsystem og applikasjoner (uninett.no, 2014).

Winch

«Winch» er et helautomatisert testmiljø for å installere OpenStack på en virtuell plattform. Som nevnt tidligere er dette testmiljøet et «proof of concept» der vi har automatisert installasjonen av OpenStack og sett hvordan de forskjellige komponentene i rammeverket fungerer sammen i praksis. Et slikt testmiljø er viktig for å kunne teste ut forskjellige komponenter av rammeverket. Om det for eksempel kom en ny versjon til en av komponentene kan denne raskt implementeres i testmiljøet for å verifisere at den fungerer som den skal. Når testingen er fullført og eventuelle feil rettet kan komponenten produksjonssettes og en vet på forhånd at den vil fungere.

Monitorering

Monitorering kan kort oppsummeres som et system som overvåker andre systemer. Om en feil skulle skje på nettverket eller at en server skulle gått ned ville systemadministratorene få beskjed om dette. Det finnes mange monitoreringsløsninger som benyttes i dag. De fleste bedrifter har gjerne bare et system som overvåker infrastrukturen, mens andre bedrifter bruker ofte flere verktøy og skreddersyr i tillegg verktøyene for å passe til de tjenestene de drifter (compnetworking.about.com, 2014).

4. Metode og materiale

Utplassering ved UIB

I løpet av dette semesteret har jeg opparbeidet meg 4,5 måneders erfaring innenfor OpenStack-rammeverket på UIB. Jeg har fått en god teknisk innsikt i hvordan de ulike komponentene fungerer sammen og samtidig lært meg bruken av en rekke nye verktøy som jeg vil benytte i bachelorprosjektet. For at jeg skal tilegne med den kunnskapen som er nødvendig for å svare på problemstillingen vil feltarbeid være en sentral del av mitt videre arbeid. Jeg har bare positive erfaringer fra UIB og håper å kunne få brukt disse erfaringene når jeg skal skrive bacheloroppgaven.

Trello

Trello er et prosjektstyringsverktøy som er basert på prinsippet med gule lapper. Verktøyet gir en enkel og overordnet oversikt over de ulike aktivitetene som til enhver tid foregår i et prosjekt. I hovedsak ligger alle aktivitetene under tre hovedkategorier; planlagt, pågående eller ferdig. Deltakerne i prosjektet blir enten tildelt en aktivitet fra prosjektleder eller tildeler aktiviteten til seg selv. Når en deltaker begynner på aktiviteten flyttes den til pågående. I hver aktivitet kan alle prosjektdeltakerne delta med kommentarer, filer og oppdateringer. Verktøyet kan enkelt konfigureres til at alle prosjektdeltakerne får beskjed når det har skjedd en oppdatering. På denne måten vet alle deltakerne hva som er status på prosjektet til enhver tid. Under følger et lite utsnitt av hvordan verktøyet ser ut i praksis (trello.com, 2014).

(home-hero.png, u.d)

Git

Git er et versjonkontrollsystem som blir benyttet i sky-prosjektet for å holde kontroll på de ulike filene og endringer i et repository. Git muliggjør at alle prosjektdeltakerne kan jobbe på samme kode samtidig. Jeg kommer til å benytte Git når jeg skal jobbe med bachelorprosjektet. Slik som problemstillingen er lagt opp er det mest sannsynlig at jeg kommer til å få min egen branch under sky-prosjektet. Da har jeg mulighet til å teste og prøve ut forskjellige løsninger innenfor før dette blir levert til selve sky-prosjektet (github.com, 2014).

Vagrant / Virtualbox

Vagrant er et åpent kildekodeverktøy for å installere og provisjonere virtuelle testmaskiner. Blir ofte sett på som en wrapper rundt virtualiseringsplattformer, i vårt tilfelle VirtualBox. Det er ved hjelp av disse verktøyene at «winch» har blitt laget. I et slikt virtuelt miljø er det veldig stor frihet på hva man gjøre på operativsystemet i forhold til en fysisk maskin. Man kan slette filer man vanligvis ikke ville slettet, og teste ut sære løsninger. Om ting skulle gå galt kan man enkelt slette maskinen og starte på nytt igjen. Ved hjelp av vagrant kan man få en ferdig maskin opp å gå i løpet av få minutter. Eventuelle tilpasninger og programvare kan spesifiseres på forhånd slik at maskinen starter opp med installert. Da slipper man å bruke tid på å gjøre slike tilpasninger i etterkant og det er svært gunstig å kjøre testmiljøer på et slikt verktøy (vagrantup.com, 2013).

Puppet

Puppet er et system for å sentralisere, standardisere og administrere IT-infrastruktur. Puppet tillater en systemadministrator å definere en ønsket tilstand på infrastrukturen i organisasjonen. Deretter blir denne tilstanden automatisk håndhevet med gitte mellomrom. Tilstandene blir definert med en rekke moduler som kommer fra en sentralisert server. I sky-prosjektet bruker vi vagrant for å opprette maskiner mens vi bruker puppetmoduler for å modifisere maskinene slik vi ønsker (puppetlabs.com, 2014, linux.com, 2010).

The Foreman

The Foreman er et åpent kildekodeprosjekt som lar systemadministratorer provisjonere, konfigurere og monitorere fysiske og virtuelle maskiner. Foreman tilbyr provisjonering til maskiner uten operativsystem (såkalt bare-metal), virtualiseringsplattformer og skyløsninger. Foreman brukes ofte i kombinasjon med Puppet og i sky-prosjektet vil man benytte Foreman til å installere de ulike maskinene som OpenStack-rammeverket skal kjøre på. Puppet modulene vil på forhånd bli importert i Foreman og vil bli kjørt på de aktuelle maskinene etter de er installert (theforeman.org, 2014).

ReText

Et enkelt men kraftig tekstredigeringsverktøy som brukes for å skrive dokumentasjon som ligger på Git. Inneholder blant annet en nyttig preview funksjon som lar brukeren se hvordan dokumentasjonen vil se ut på Git uten at selve dokumentet må åpnes i en nettleser (http://sourceforge.net/projects/retext/, 2014).

5. Tidligere forskning, historie og veien videre

Ideen for et globalt datanettverk var introdusert på 1960-tallet av J.C.R Licklider som var en av de som var ansvarlig bak utviklingen av ARPANET. Visjonen til Licklider var at alle på kloden kunne være koblet sammen og ha tilgang til programmer og data der de befant seg. Det er dette begrepet som knytter seg opp til skyteknologi som vi kjenner i dag. Siden 1960-tallet har skyteknologien utviklet seg videre, og siden internett ble allemannseie på midten av 1990-tallet har det skjedd store utviklinger.

En av de første store aktørene innenfor skyteknologi kom i 1999, da Salesforce.com pionerte konseptet ved å kunne tilby sine kunder applikasjoner gjennom en enkel nettside. Dette skapte vei for flere aktører som ville gjøre det samme. I 2002 kom også Amazon på banen da de startet med sine skyløsninger. I 2006 kom produktet som de aller fleste innenfor IT-bransjen kjenner til i dag, Amazon Elastic Compute Cloud. Mohamed peker på i sin artikkel at noen av faktorene som har drevet utviklingen av sky fremover har vært evnen til å utvikle applikasjoner som folk benytter seg av («killer apps»). Dette kan vi se eksempler på i dag blant annet med Dropbox og Google Drive. Det å kunne minimere egne behov for backup og ha tilgang til de samme filene fra ett og samme sted har vært hovedargumentet for mange når de har valgt å bruke disse to tjenestene. Videre påpeker også Mohamet at virtualiseringsteknologi har modnet og utviklet seg fra tidligere. Det som var et lite tema for ti år siden har nå blitt noe som de aller fleste bedrifter benytter seg av i dag. Virtualisering er en av byggesteinene i dagens skyløsninger og det vil i fremtiden bli mer og mer vanlig at bedrifter minimerer egen infrastruktur for å heller plassere denne i en skyløsning. På denne måten vil man minimere kostnader, få større fleksibilitet og on-demand ressursskalering. Den aller største skepsisen til skyteknologi er sikkerhet og personvernsspørsmål. Hvordan kan en sikre data på en god og hensiktsmessig måte og sørge for at informasjon ikke kommer på avveie. Det er mange spørsmål omkring dette og mange gode svar på, både for og imot.

Uansett hva som blir svaret på mange av disse spørsmålene ser en uansett hvilken vei utviklingen innenfor IT går mot. Nesten alle av dagens store aktører innenfor bransjen har valgt å posisjonere seg med en eller annen form for skyløsning og det vil bli spennende å se hvilken av løsningene som kommer til å bli mest brukt (Mohamed, 2009).

6. Fremdriftsplan

- Trinn 1: Gjøre meg kjent med prosjektet og sette meg inn i OpenStack. Utarbeide prosjektbeskrivelse (Høst 2014).
- Trinn 2: Delta i prosjektet og jobbe i felt for å kunne svare på problemstillingen. Skrive innledningskapittel, metodekapittel og teorikapittel (Fra januar 2015 og frem til påske 2015).
- Trinn 3: Fortsette å jobbe i felt i de timene som inngår i bachelorprosjektet. Analysere/reflektere rundt arbeidet som er utført under feltarbeidet og ferdigstille oppgaven. (April-Juni 2015).

7. Litteratur

Brockmeier, Joe. (2010). *Introduction to puppet: Streamlined System Configuration.* Tilgjengelig fra: http://www.linux.com/learn/tutorials/325201-introduction-to-puppet-streamlined-system-configuration (Hentet 10.11.2014).

compnetworking.about.com (2014. *Net monitoring.* Tilgjengelig fra: http://compnetworking.about.com/od/itinformationtechnology/f/net_monitoring.htm (Hentet 24.11.2014).

github.com (2014). *About GitHub.* Tilgjengelig fra: https://github.com/about (Hentet: 11.11.14)

home-hero (u.d). Internett. Tilgjengelig fra: https://d2k1ftgv7pobq7.cloudfront.net/meta/p/res/images/fb4de993e22034b76539da073ea8d35c/home-hero.png (Hentet 24.11.2014).

Mohamed, Arif (2009). *A history of cloud computing.* Tilgjengelig fra: http://www.computerweekly.com/feature/A-history-of-cloud-computing (Hentet 24.11.2014).

openstack.org (2014). Software. Tilgjengelig fra: http://www.openstack.org (Hentet 25.07.14).

openstack-software-diagram (u.d). Internett. Tilgjengelig fra: http://www.openstack.org/themes/openstack/images/openstack-software-diagram.png (Hentet 24.11.2014).

puppetlabs.com (2014). *What is puppet*. Tilgjengelig fra: http://puppetlabs.com/puppet/what-is-puppet (Hentet: 10.11.2014)

ReText (2014). *Projects retext.* Tilgjengelig fra: http://sourceforge.net/projects/retext/ (Hentet 10.11.2014).

theforeman (2014). *Learn More*. Tilgjengelig fra: http://theforeman.org/learn_more.html (Hentet. 12.11.2014)

uninett.no (2014). *Plan UH-Iaas teknisk implementering*. Tilgjengelig fra: https://www.uninett.no/sites/drupal.uninett.no.uninett/files/Plan%20UH-IaaS%20teknisk%20implementering.pdf (Hentet 01.08.2014).

vagrantup.com (2013). *About.* Tilgjengelig fra: https://www.vagrantup.com/about.html (Hentet 24.11.2014).

Woodford, Chris. (2009). *Cloud computing introduction.* Tilgjengelig fra: http://www.explainthatstuff.com/cloud-computing-introduction.html (Hentet 11.11.2014)