Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

Ådne Helmersen

En sammenligning av Amazon Web Services og Microsoft Azure

Bacheloroppgave i Informasjonsbehandling
Veileder: Helge Hafting
Mai 2021

NTNU
Kunnskap for en bedre verden

Ådne Helmersen

En sammenligning av Amazon Web Services og Microsoft Azure

Bacheloroppgave i Informasjonsbehandling
Veileder: Helge Hafting
Mai 2021

Norges teknisk-naturvitenskapelige universitet
Fakultet for informasjonsteknologi og elektroteknikk
Institutt for datateknologi og informatikk

NTNU

Kunnskap for en bedre verden

Sammendrag

Oppgaven er en selvvalgt bacheloroppgave på studiet informasjonsbehandling ved NTNU. Jeg ønsket å bli bedre kjent med skyteknologi, og se om det er mulig å avdekke større forskjeller mellom de 2 største leverandørene, Amazon Web Services og Microsoft Azure. Denne sammenligningen skal kunne brukes av bedrifter til å se hvilken plattform som passer best for sin IT-infrastruktur. Det kan enten gjøres ved å se på de sammenligninger som er gjort i dette prosjektet, eller ved å få et innblikk i hvilke faktorer som spiller inn når et slikt valg skal tas.

Abstract

This assignment is a self-chosen bachelor thesis in the study of Information Technology at NTNU. I wanted to get better acquainted with cloud computing and see if it is possible to uncover bigger differences between the 2 largest providers, Amazon Web Services and Microsoft Azure. This comparison may be used by companies to see which platform is best suited for their IT infrastructure. This can either be done by looking at the comparisons made in this project, or by gaining an insight into which factors that come into play when such a choice is to be made.

Forord

Da er bachelorprosjektet våren 2021 over for min del. Det har vært en veldig givende prosess som har gitt meg økt forståelse for et fagfelt jeg ikke har lært mye om på studiet.

Jeg valgte å skrive om skyteknologi da det er et fagfelt jeg har fått interesse for i løpet av de siste årene, i tillegg til at det er svært relevant for de store bedriftene på arbeidsmarkedet i dag. Jeg har også fått meg jobb hvor jeg skal jobbe med IT-infrastruktur i sky, så jeg ønsket derfor å benytte bacheloroppgaven til å forberede meg best mulig til oppstart der. Jeg har tilegnet meg mye kunnskap på et spennende område som jeg gleder meg til å få mer praktisk erfaring med etter studiene.

Jeg vil takke veileder på NTNU, Helge Hafting for gode tilbakemeldinger og tips underveis i prosessen. Dette har gjort det enklere å holde jevn progresjon gjennom hele perioden fra oppstart til sluttprodukt.

INNHOLD

Innhold

Introduksjon

1.1 Om bacheloroppgaven

Skyteknologi har gjort det mulig for større aktører å sentralisere datakraft, for så å distribuere denne ved hjelp av internett. Maskinvaren befinner seg nå i store datasenter og blir driftet internt av aktørene som leverer tjenester til sluttbruker. Dette kan potensielt gi store fordeler for bedrifter som ønsker å oppdatere infrastrukturen sin fra den tradisjonelle «on premises» med intern drift og vedlikehold til bruk av skyteknologi.

I denne oppgaven skal jeg sammenligne plattformene til de to største aktørene på markedet i dag, Amazon Web Services og Microsoft Azure. Jeg skal se nærmere på hva de koster, hvordan skalering fungerer for de to og hvor tilgjengelige de er. Mot slutten av oppgaven skal et PHP-basert system implementeres på begge plattformer hvor framgangsmåte blir presentert. Oppgaven er selvvalgt og utarbeides av forfatter alene. Denne rapporten er sluttproduktet i faget IINI3011 Bacheloroppgave i Informasjonsbehandling våren 2021. Arbeidet med rapporten tar utgangspunkt i forstudierapporten som ligger vedlagt, og består i korte trekk av en teoridel, en presentasjon av resultater, en drøftingsdel og en praktisk gjennomføring som beskrives med dokumentasjon.

1.2 Problemstilling

Hensikten med oppgaven er å gjøre det enklere for bedrifter å velge leverandør når IT-infrastrukturen skal oppdateres. Rapporten skal i hovedsak bidra til økt kompetanse på området, men også gi et klarere bilde av om en slik omveltning kommer til å lønne seg, hvilke forretningsmessige fordeler den gir, hva den faktisk koster og hvilke utfordringer som kan være knyttet til overgangen.

1.3 Prosjektmål

Denne delen av oppgaven tar utgangspunkt i de målene som ble satt opp i forstudierapporten. Den sier noe om hva som skal oppnås med prosjektet og hvordan det skal gjennomføres. Målene er utarbeidet med utgangspunkt i en tradisjonell IT-infrastruktur og forklarer en bedrifts motivasjon til å gjennomføre en slik migrering. Merk at denne delen ikke tar for seg forskjellene mellom AWS og Azure, men heller årsaken til at man ønsker en endring.

1.3.1 Effektmål

Effektmålene beskriver hvilken motivasjon en bedrift har for å migrere dagens systemer ut i skyen:

- Øke sikkerhet mot tap av data
- Oppnå lavere kostnader knyttet til drift av IT-systemer
- Få tilgang til tjenester som gjør det mulig å få innblikk i data som tidligere har vært for dyrt eller komplisert

1.3.2 Resultatmål

Resultatmålene skal gjøre det klart hva man evt. sitter igjen med om man velger å migrere systemene sine. Disse legger også grunnlaget for å få svar på problemstillingen i forrige punkt.

- Lagre data på flere servere i datasenter hvor leverandør er ansvarlig for oppetid
- Benytte seg av auto-skalering slik at kapasiteten hele tiden møter bedriftens behov.
- Ta i bruk tjenester for data mining og BI som en del av det virtuelle nettverket for enklere å få innblikk i kompliserte og store datamengder.

1.3.3 Prosessmål

Som nevnt i forstudierapporten er det naturlig at jeg nevner prosessmålene for oppgaven da dette er et studieprosjekt hvor sluttproduktet er bacheloroppgaven på studiet Informasjonsbehandling på NTNU. Målene presenterer hva jeg skal oppnå ved å fullføre et slikt prosjekt og de gjelder kun for forfatter da jeg skriver oppgaven alene.

- Oppnå en karakter i faget som gjenspeiler det nivået jeg har lugget på ellers i løpet av studietiden min på NTNU.
- Oppnå økt kompetanse på feltet «Cloud Computing» da dette er noe som interesserer meg og det hører inn under det fagfeltet jeg skal jobbe med etter endt studium.
- Få god oversikt over forskjellene i produktene som Amazon og Microsoft tilbyr.
- Oppnå egenutvikling i form av økt kompetanse på å gjennomføre et omfattende prosjekt med store krav til dokumentasjon og kommunikasjon med en veileder.

1.4 Avgrensning av oppgaven

Som det framgår i problemstillingen så er hensikten med oppgaven å gjøre det enklere for bedrifter å se hvilke fordeler de kan oppnå ved å migrere til skyen, samtidig som de får et bilde av hvilken leverandør som passer best for sin forretningsmodell. Jeg skal oppnå dette ved å sammenligne de mest elementære skytjenestene AWS og Azure tilbyr, i tillegg til andre faktorer som er avgjørende når et slikt valg skal tas. Dette vil legge grunnlag for å avdekke eventuelle forskjeller mellom plattformene som kan være utslagsgivende når man skal ta et slikt valg. Pris, autoskalering, prøveperioder og tilgjengelighet er faktorene som skal sammenlignes. I tillegg skal en webapplikasjon implementeres på begge plattformer, og jeg skal presentere framgangsmåten ved hjelp av tekst og figurer. Mine erfaringer med dette vil inngå som en del av resultatene i denne rapporten.

1.5 Metode og struktur

I dette kapittelet skal jeg beskrive prosessen mot sluttproduktet, altså hvilke faser jeg går gjennom fra start til sluttlevering. Her vil det også forklares hvor informasjonsgrunnlaget blir hentet fra og hvordan informasjonshentingen legger grunnlag for diskusjon.

Både Amazon og Microsoft har gode dokumentasjonssider som jeg kommer til å benytte meg av. Informasjonen her blir kontinuerlig oppdatert og vil være den sikreste kilden når plattformene skal sammenlignes. Informasjonen som blir hentet herifra legger grunnlaget for kapittel 2 i denne rapporten - teori og trender. Her begynner jeg med å se på forskjeller mellom tradisjonell infrastruktur og skyteknologi, før fokuset rettes mot tjenestene som finnes på plattformene. Dette er første fase av prosjektet og all informasjon skal være hentet inn før jeg går videre.

I kapittel 4 skal resultatene fra informasjonshentingen presenteres. Resultatene legger grunnlaget for videre diskusjon og drøfting i kapittel 5, som skal ende med konklusjon for oppgaven. Kapittel 1.2 beskriver hvilke spørsmål som skal være besvart. I kapittel 3 skal et PHP-system som lar bruker konfigurere en bil implementeres på begge plattformer. Dette blir en praktisk gjennomføring hvor jeg får testet noen av tjenestene og gjort en sammenligning av disse. Denne gjennomføringen blir dokumentert med tekst og skjermbilder før jeg deler mine erfaringer som en del av resultatene.

Hele prosjektet skrives i en sekvens av aktiviteter eller faser. Dette blir gjerne omtalt som fossefallsmetoden og er en tradisjonell måte å styre prosjekter på. Jeg gjør meg ferdig med informasjonshentingen og presenterer denne før jeg går videre på resultatet. Resultatet skal være ferdig presentert før jeg går i gang på diskusjonen. Prosjektet rundes av med konklusjon og en sluttrapport hvor jeg beskriver prosessen fra start til slutt samt deler mine erfaringer rundt prosjektet. I etterkant skal også bacheloroppgaven presenteres over video for medstudenter og undervisere på campus.

2. Skyteknologi – teori og trender

2.1 Bakgrunn

De siste årene har mer og mer av datakraften vi benytter oss av kommet fra skyen. Vi kaller det skyen fordi vi som sluttbruker ikke har fysisk tilgang til eller vet noe om den fysiske plasseringen til maskinvaren vi benytter oss av. Servere og lagringsenheter er sentralisert i store datasentre rundt om på kontinentene og distribuerer datakraft ut til kundene. Prinsippet er det samme om man lagrer mobilbildene sine i skyen, som når en bedrifter flytter hele infrastrukturen sin med webservere, lagringsenheter mailservere osv. ut i skyen. Sistnevnte er derimot mye mer omfattende enn å opprette et abonnement i for eksempel ICloud. Vi skal i dette kapittelet se nærmere på hvordan den tradisjonelle infrastrukturen ser ut og hvilke fordeler man kan oppnå ved å flytte denne ut i skyen. Videre ser vi på klassifisering av skytjenester, som sier noe om hvor stor grad man baserer seg på skyen. Kapittel 2-3-2.5 er hoveddelen av teori-biten i denne rapporten og det er her teorigrunnlaget for videre diskusjon blir presentert.

2.1.1 Tradisjonell infrastruktur

Tradisjonell infrastruktur består av den maskinvaren og programvaren som en bedrift benytter seg av i det daglige. All maskinvaren er koblet til et nettverk og snakker sammen ved hjelp av en server. Denne serveren gjør det mulig for de ansatte å få tilgang til bedriftens data, applikasjoner og andre systemer de har behov for. Når bedriften får behov for økt datalagring eller kraft til prosessering av data må de selv gå til innkjøp av maskinvare slik at de kan skalere opp til nødvendig kapasitet. Et slikt oppsett krever også at systemene driftes internt og hele tiden er oppdatert for å være sikret mot innbrudd, tap av data og andre katastrofer. Hvis en bedrift har behov for økt kapasitet i en periode må de med denne infrastrukturen være skalert for dette året rundt, da man ikke har råd til å la brukerne vente på svar fra nettsider og databaser. Dette medfører unødvendige kostnader da infrastrukturen ikke blir brukt til sitt fulle potensial hele tiden. Fordelen med denne tilnærmingen er sikkerheten og kontrollen man har over egne data, da disse er lagret on-premises og fysisk sikret fra omverdenen, i motsetning til i et datasenter [1].

2.1.3 Fordeler med skybasert databehandling

I distribuerte datasenter vil flere servere jobbe sammen, men er koblet sammen slik at de fremstår som en for sluttbruker. På denne måten vil man ikke tape data eller stå i fare for nedetid hvis en server krasjer. En fysisk nærliggende server vil da ta over arbeidet og fortsette der den ødelagte slapp. I stedet for å kjøpe den maskinvaren man behøver, leier man heller den kapasiteten man behøver til enhver tid i datasenteret. All infrastrukturen man har i skyen henger sammen i et sett av tjenester og blir referert til som bedriftens VPC («Virtual Private Cloud») [2]. Oppstår det et økt behov for lagringsplass eller datakraft vil skyen automatisk skalere dette opp og legge til den tjenesten man behøver i VPC. Det samme gjelder også for programvare; hvis en bedrift skal ansette flere behøver de også programvare for de nye ansatte. Da vil skyen automatisk skalere opp lisensen for disse, som er kostnadsbesparende sammenlignet med å kjøpe inn egne lisenser og installere dette selv for hver ansatt. For å gjøre det enklere å forstå at en bedrift kan ha periodevis økt behov for kapasitet kan vi se på et eksempel: Når Norges befolkning skal sjekke skattemeldingen sin er det et stort antall personer som logger seg inn på Skatteetaten i samme periode. Dette medfører mange flere forespørsler mot web-serverne enn det de har resten av året. Oppdages en slik økt etterspørsel vil infrastrukturen her automatisk legge til flere web-servere og ta hånd om den økte etterspørselen, slik at folk slipper å vente på svar og unngår få tidsutløp i nettleseren.

En annen fordel med skyteknologi er graden av automatisering man kan oppnå. I skyen blir all drift tatt hånd om av leverandør og de tar seg av alt fra maskinvare til å sørge for at sikkerheten står til kravene. Dette i stor kontrast til den tradisjonelle modellen hvor bedrifter må ha flere ansatte til drift av IT-systemer, som kan være veldig kostbart.

Som nevnt er automatisk skalering en fordel man oppnår med skyteknologi. Dette vil naturligvis være kostnadsbesparende da man kun betaler for den kapasiteten man bruker. Dette er revolusjonerende og gjør at IT-ressurser nå kan sammenlignes med andre kostnader som strøm og vann. Mindre sjanse for nedetid er en annen faktor som vil gjøre modellen billigere på sikt, da leverandør er ansvarlig for maskinvaren som befinner seg i sentrene og bedriften slipper å ta ansvar for dette. I skyen er man også sikret å ha tilgang til det nyeste som finnes av teknologi til enhver tid. Tjenestene på plattformen og maskinvaren i datasenteret blir kontinuerlig oppdatert. I den tradisjonelle modellen må bedriften selv stå for kostnader knyttet til oppgradering av

maskinpark, da denne har en viss levetid. Dette gjelder for all maskinvare, og utviklingen går fort. Disse kostnadene slipper altså bedriften unna, da det blir tatt hånd om av leverandøren.

Med sentralisert maskinvare i datasenter betyr dette at lagring av data vil skje utenfor bedriftens kontroll. Hvem som helst med tilgang til bedriftens VPC over internett kan laste ned kritiske data, noe som kan få store negative konsekvenser for bedriften. Å velge en leverandør som forsikrer kunden om at nødvendige sikkerhetstiltak er overholdt vil da være helt kritisk [1]. Noen typer data vil likevel ikke være egnet for oppbevaring utenfor bedriftens kontroll, og det finnes derfor flere tilnærminger når man skal migrere til skyen. Disse skal vi se nærmere på i kapittel 2.2.

2.2 Klassifisering av skytjenester

Etter hvert som skyteknologi har blitt mer utbredt og populært har det kommet flere modeller og tilnærminger på markedet. Det som skiller modellene fra hverandre er i hvor stor grad man belager seg på skyen. Som nevnt i forrige kapittel vil ikke all data være egnet for oppbevaring i et datasenter man ikke har fysisk kontroll over. Da vil man være avhengig av en alternativ løsning hvor man likevel oppnår fordelene med bruk av sky, samtidig som man kan håndtere kritiske data internt. I de neste delkapitlene skal vi kort se på hvilke modeller som finnes og hva som skiller de fra hverandre. De 3 første modellene er beskrivelser av produktet kunden kjøper og sier noe om hvor stor del av infrastrukturen man ønsker at leverandøren skal håndtere. De siste 3 modellene beskriver tilnærminger for deployering i skyen.

2.2.1 Modeller for skyteknologi

2.2.1.1 Infrastructure as a Service (IaaS)

IaaS består av de tjenester i skyen som gir tilgang til nettverksfunksjoner, datalagring, servere og virtuelle maskiner. Denne modellen gir den høyeste fleksibiliteten for en bedrift, da man selv har mulighet til å styre IT-ressursene på lik linje som i en tradisjonell modell [3]. Dette gjør at man kan sy sammen og gjenskape den nåværende infrastrukturen i skyen. Dette kalles en migrering og er ofte en omfattende og krevende prosess hvor man gjerne benytter seg av ekstern kompetanse.

2.2.1.2 Platform as a Service (PaaS)

Med PaaS styrer leverandøren alt av underliggende infrastruktur som maskinvare og operativsystemer og planlegger skalering etter behov [3]. Man leier altså en ferdig driftet og konfigurert plattform og kan passe godt for bedrifter som driver med systemutvikling. PaaS gir mindre fleksibilitet enn IaaS, men kan være enklere og mer effektivt da man kun trenger å fokusere på deployering av applikasjoner.

2.2.1.3 Software as a Service (IaaS)

SaaS tilbyr et komplett produkt som er klart til å tas i bruk og blir distribuert direkte ut til kunde. Denne formen for skyteknologi er nok den de fleste har kjennskap til fra jobb og hverdag, da vi benytter oss av flere slike tjenester hver eneste dag. Web-basert e-post er et eksempel på SaaS, det samme er Office 365. Likt som i PaaS så er all underliggende infrastruktur styrt av leverandør og alt man behøver å tenke på med IaaS er hvordan man skal integrere produktet i sin bedrift [3].

2.2.2 Modeller for deployering I skyen

2.2.2.1 Cloud

En applikasjon deployert etter denne modellen benytter seg fullt ut av den sentraliserte datakraften i datasenteret og bedriften behøver ikke stille med egen infrastruktur her. Applikasjonen kan enten utvikles i skyen eller migreres fra eksisterende systemer for å oppnå de fordeler det gir [3]. Ønsker man å oppnå alle fordelene nevnt i kapittel 2.1.3 så går man for denne modellen.

2.2.2.2 Hybrid Cloud

Hybrid deployering er en måte å kombinere eksisterende IT-ressurser med skybasert infrastruktur. Det mest vanlige er å utvide den tradisjonelle infrastrukturen man besitter i dag ved å skalere denne ut i skyen [3]. Hybrid sky gir også noen fordeler i forhold til sikkerhet, da man

med denne modellen kan velge å oppbevare kritiske data i den maskinvaren man har fysisk kontroll over. Dette vil for eksempel være nødvendig i helsesektoren da man ikke har mulighet til å oppbevare pasientjournaler utenfor landegrensene. Forsvaret vil også ha systemer med slik informasjon og vil kunne ta nytte av hybride skyløsninger.

2.2.2.3 On-premises

On-premises kalles også privat sky og går ut på at man tar i bruk teknikker fra skyteknologien uten å deployere systemer i skyen. Man oppnår ikke de samme fordelene her, men man kan ta i bruk applikasjonsstyring og teknikker for virtualisering for å forbedre utnyttelse av ressursene man har internt [3].

2.3 Om leverandørene

Før vi vi går over til å se på forskjellene mellom AWS og Azure skal jeg presentere de to leverandørene. Jeg har valgt å sammenligne disse to da de er klart størst på markedet i dag, og stadig voksende. De siste årene har begge aktørene bygd nye datasentre og vi kan komme til å få sentre fra begge aktørene på norsk jord i framtiden. Informasjon om leverandørene er hentet fra deres egne hjemmesider.

2.3.1 Amazon Web Services (AWS)

AWS er et datterselskap av Amazon som ble lansert i juli 2002. AWS er til nå verdens største skyplattform og er satt sammen av over 200 tjenester som distribueres fra datasentre plassert over hele kloden. AWS er den plattformen med flest funksjoner og tilbyr infrastrukturteknologi som databaser, lagring, databehandling, kunstig intelligens og datanalyse. AWS er også den plattformen med størst kundebase i dag, med flere millioner aktive kunder og titusenvis av partnere globalt. AWS har 80 tilgjengelighetssoner og 25 geografiske regioner rundt om i verden [4] og har store planer om videre utbygging. Under kan vi se et kart som viser AWS sine datasentre i dag, og hvor de planlegger å bygge nye.

Figur 1 AWS-regioner i dag, https://aws.amazon.com/what-is-aws/?nc1=f_cc

2.3.2 Microsoft Azure

Også Azure er en plattform som tilbyr over 200 produkter og tjenester som skal gjøre det enklere for bedrifter å sette nye ideer til live gjennom å løse dagens utfordringer og å bygge den teknologiske framtiden. De tilbyr alle modellene for administrering av skyen og har et stort fokus på sikkerhet. Azure invester over en milliard amerikanske dollar hvert år for å sikre kundenes data fra cybertrussler [5]. Azure er eid og skapt av Microsoft og ble lansert 1. Februar 2010. Bildet på neste side viser en oversikt over Azure sine datasenter i dag. Som vi kan se så har de allerede 2 senter på plass i Norge; et i Oslo og et i Stavanger.

Figur 2. Azure-regioner i dag, https://azure.microsoft.com/nb-no/global-infrastructure/geographies/

2.4 Prising og skalering

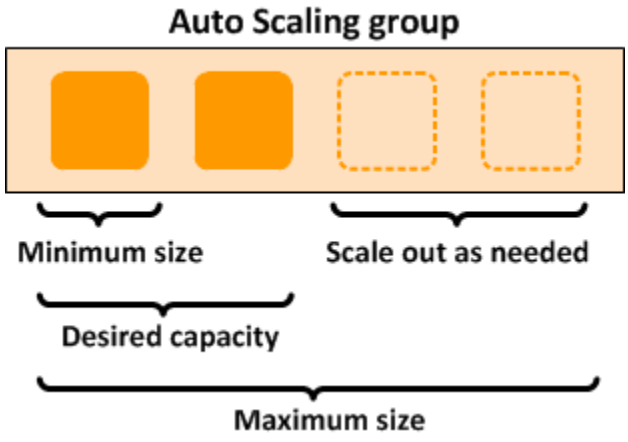
Vi skal her se nærmere på prising og skalering av de to plattformene. Prising er en viktig faktor når man skal velge skyleverandør, og som nevnt tidligere har skyteknologi gjort at man kan se på IT-ressurser på samme måte som man ser på vann og elektrisitet siden man nå har mulighet til å betale for kun de tjenester og produkter man bruker. Det er likevel flere faktorer som spiller inn når man skal undersøke hva disse tjenestene kommer til å koste. Det er heller ikke enkelt å analysere en slik forskjell mellom plattformene, da prisene er i konstant endring. Trenden til nå har vært at jo større kundemassene blir og jo større konkurransen mellom plattformene blir, jo lavere blir prisene. En prisanalyse vil ikke være holdbar flere år fram i tid, men vil likevel kunne være til stor hjelp da den avdekker prisforskjeller man kanskje ikke har vært klar over fra tidligere. Tallene vil også være til hjelp når budsjett for en evt. migrering til skyen skal utarbeides. Det som derimot kan gjøre det enklere å finne ut hva et oppsett i skyen vil komme til å koste etter dagens satser er priskalkulatorer. Både AWS og Azure tilbyr en priskalkulator som gjør det mulig å plotte inn de tjenestene man ser for seg å bruke, for så å få oppgitt pris per måned. Begge aktører lar også kunden reservere ressurser på forhånd for å spare penger.

Hva man ender opp med å betale er som vi har sett avhengig av forbruk. Prising henger derfor tett sammen med auto-skalering av infrastrukturen. For å forstå fordelene man kan oppnå med

skybasert infrastruktur er det greit å forstå hvordan disse tjenestene fungerer. Disse skal vi se nærmere på nå, for å avdekke om det finnes noen forskjeller mellom plattformene på dette området. Hva de ulike tjenestene koster på hver plattform blir presentert under resultater ved hjelp av priskalkulatoren.

2.4.1 Amazon EC2 Auto Scaling

Informasjonen i dette kapittelet er hentet fra den delen av informasjonsbasen til AWS som tar for seg automatisk skalering av EC2-instanser [6]. EC2 er betegnelsen på AWS sine virtuelle maskiner i skyen. Amazon EC2 Auto Scaling skal sørge for at man til enhver tid har riktig antall maskiner i det virtuelle private nettverket. Man konfigurer selv gruppen med maskiner ved å sette et minimum og maksimum antall maskiner som kan kjøre i nettverket. AWS vil deretter sørge for at man hele tiden møter kapasiteten man behøver innenfor disse grensene. Under ser vi et eksempel på en gruppe med minimum størrelse på 1 maskin, ønsket kapasitet på 2 maskiner og maksimal størrelse på 4 maskiner.

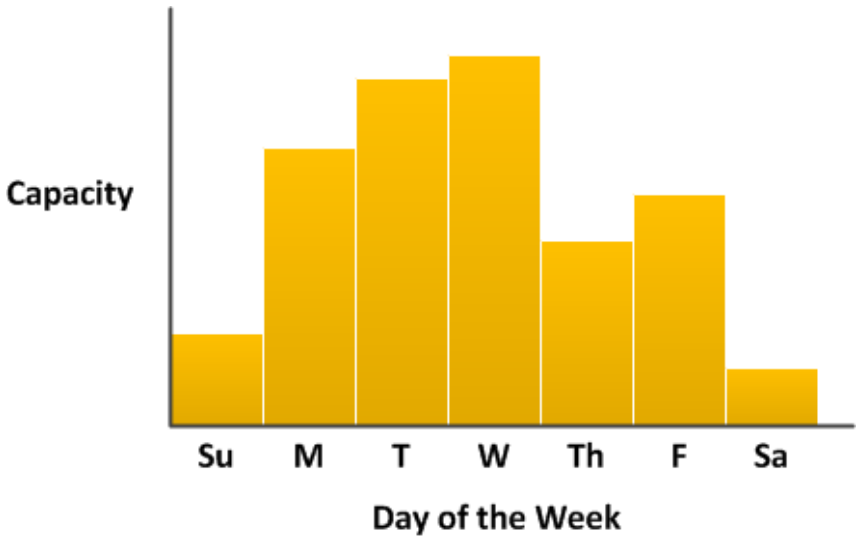


Figur 3. AWS Auto Scaling group, https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Informasjonen er hentet fra AWS sine hjemmesider [6] og de trekker fram disse fordelene ved å ta i bruk Auto Scaling:

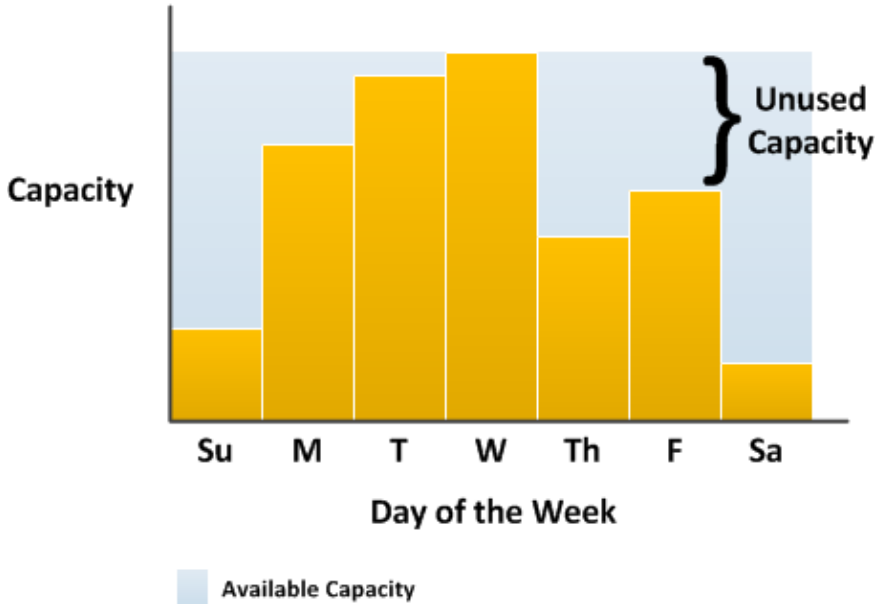
- «Amazon EC2 Auto Scaling vet når en maskin er ikke fungerer optimalt slik at den kan drepes og erstattes av en ny. Man kan også konfigurere instans-gruppen til å benytte seg av flere tilgjengelighetssoner. Hvis en sone blir utilgjengelig, vil maskiner i en annen sone ta over og kompensere for den utilgjengelige sonen.»
- «Bedre tilgjengelighet: Amazon EC2 Auto Scaling sørger for at applikasjoner alltid har kapasiteten som behøves for den gjeldende trafikken.»
- «Bedre styring av kostnader: Dynamisk øking og synking av kapasitet etter hvert som behovet endrer seg gjør at man sparer penger. Instanser startes opp når det behøves og drepes når det ikke er behov for dem lengre.»

For å illustrere effekten av Auto Scaling har AWS utarbeidet et eksempel som viser hvordan man kan spare penger på dynamisk justering av kapasitet i infrastrukturen. Første figur nedenfor viser kapasiteten som kreves av en webapplikasjon gjennom en uke. Som vi kan se så er forbruket størst i midten av uken, og betydelig lavere lørdag og søndag.



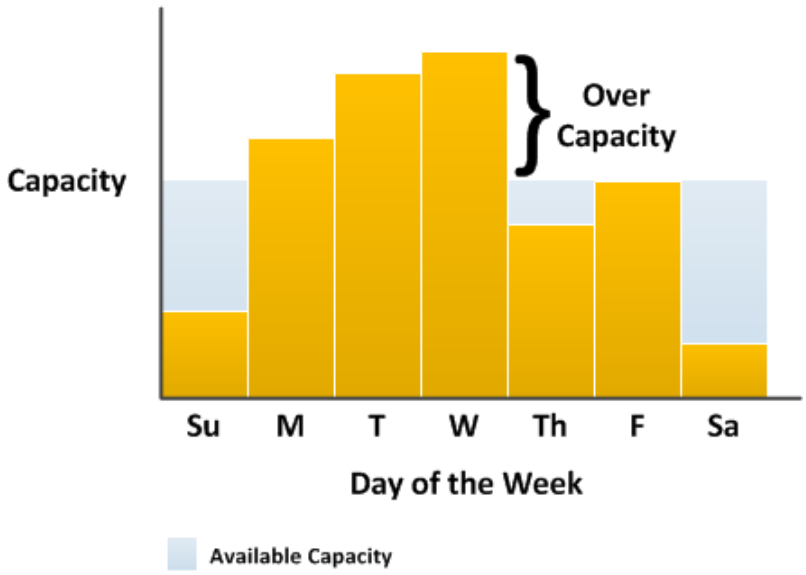
Figur 4. Ressurs-behov gjennom en gjennomsnittlig uke. https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Tradisjonelt har man da to muligheter når et slikt problem skal håndteres. Den første er å legge til så mange servere at man til enhver tid er rustet for den høyeste pågangen gjennom uken. Ulempen med dette er at man kjører med ubrukt kapasitet mesteparten av tiden.



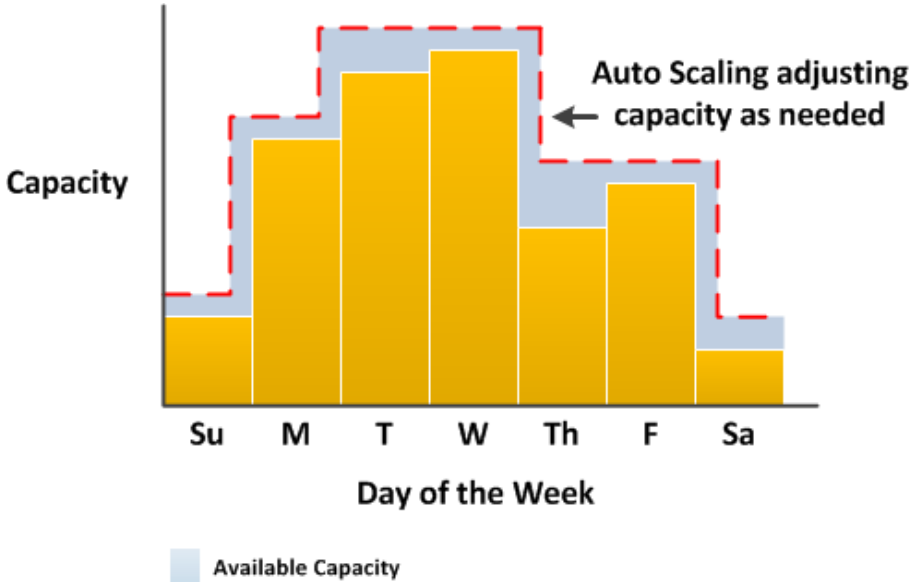
Figur 5. Ubrukt kapasitet ved tradisjonell infrastruktur. https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Den andre muligheten man har er å legge seg på en kapasitet som dekker det gjennomsnittlige behovet. Fordelen med dette er at man sparer penger ved å ikke kjøpe unødvendig mye maskinvare. Ulempen er selvfølgelig at sluttbruker av applikasjonen kan få en dårlig opplevelse da applikasjonen har lengre responstid på mandag, tirsdag og onsdag. Se figur under.



Figur 6. Underdimensjonert infrastruktur. https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

Til slutt har vi en figur som illustrerer hvordan man har et tredje valg med EC2 Auto Scaling. Her ser vi hvordan man får fordeler av å kunne legge til og drepe instanser kontinuerlig for å hele tiden følge etterspørselen av ressurser for applikasjonen. «Du har nå en kostnadseffektiv infrastruktur som tilbyr den beste kundeopplevelsen samtidig som kostnader minimeres».



Figur 7. Illustrasjon av fordelen med Auto Scaling. https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

2.4.2 Azure Autoscale

Azure Autoscale er navnet på tjenesten levert av Microsoft Azure for skalering av infrastrukturen. Azure tilbyr skalering for flere tjenester hvor etterspørselen av kapasitet hele tiden er i endring. Informasjonen er hentet fra Azure sine hjemmesider og de trekker fram disse fordelene med Autoscale [7]:

- «Optimalisere responstid fra applikasjoner»
- «Lar kunden skalere etter egendefinerte verdier» (f.eks. planlegge kapasitet etter tider på døgnet hvor pågangen er større)
- «Forutse etterspørsel med forskjellige tidsplaner. Denne tjenesten kalles «Scheduled autoscale» og lar kunden varsle om oppkommende hendelser som kommer til å kreve økt kapasitet i infrastrukturen. Da vil Autoscale legge til flere maskiner slik at man er beredt før pågangen inntreffer.»
- «Spare penger ved å kun betale for den kapasiteten man bruker.» Her trekker Azure fram et eksempel hvor en bedrift benytter seg av PaaS og jobber med å utvikle applikasjoner. Det typiske for slike bedrifter er at det meste av arbeidet utføres i normal arbeidstid, mellom 8 om morgenen og 4 på ettermiddagen. Da kan infrastrukturen tilpasse seg dette og skalere ned når systemene ikke er under like stor belastning slik at timesprisen blir lavere i disse timene.
- «Få rask beskjed når det oppstår feil i infrastrukturen. Azure overvåker ytelsen til de virtuelle maskinene i nettverket, slik at kunden kan tilpasse og legge til varslinger for ulike beregninger av ytelse, for eksempel responstid fra en webapplikasjon.»

2.4.3 Prøveperioder

En «Free Tier Account» er et tilbud som lar kundene prøve ut tjenestene gratis opp til en viss grense for hver tjeneste. Tilbudet er tilgjengelig for alle brukere, både studenter, bedrifter og de som kun ønsker å eksperimentere med plattformene. Det vil si at prøveperioden også er en faktor som spiller inn når man skal bestemme seg for hvilken plattform man skal gå for, da man får innblikk i hvilken leverandør som tilbyr de best egnede tjenestene for sitt bruk. Både Azure og AWS tilbyr prøveperiode for sine kunder [8] [9] og jeg skal se nærmere på hvor lang denne

perioden er og trekke fram noen av tjenestene de tilbyr i gratisperioden. Dette vil legge grunnlag for å kunne anbefale en av plattformene basert på hvem som tilbyr mest eksperimentering i prøveperioden.

2.5 Tilgjengelighet og Servicenivåavtaler (SLA)

I kapittel 2.3 fikk vi et innblikk i hvor tjenestene til aktørene er tilgjengelig for sluttbruker gjennom geografisk og fysisk plassering av maskinvaren deres. Det skal vi se mer på i dette kapittelet, for senere å avdekke om det finnes noen forskjeller mellom AWS og Azure på dette området. Tilgjengelighet handler også om hvor stor del av tiden tjenestene er tilgjengelig for kunden. Dette kommer fram i såkalte servicenivåavtaler som vi også skal se nærmere på. En servicenivåavtale er en avtale mellom leverandør og kunde som skal sikre at minstekravet hele tiden nås hva angår tjenestenivå. Avtalen gir en sikkerhet for pålitelighet, tilgjengelighet og responstid fra systemer og applikasjoner. Den skal også sikre en gjensidig forståelse av tjenestenes funksjon, prioriterte områder, ansvarsfordeling og garantier fra tjenesteleverandør [10]. Høy tilgjengelighet har man når en tjeneste er tilgjengelig for kunden 99,999% av tiden, som gir en nedetid på kun 5 minutter per år [11].

Før vi går i gang med teorien for hver plattform skal jeg forklare noen begreper som kommer til å gå igjen i løpet av dette kapitelet og kapittelet som presentere resultatene. Definisjonene er hentet fra dokumentasjonsbasen til Azure [12] og SLA-siden til Azure [13], men de samme definisjonene gjelder også for AWS.

Region: «En gruppe datasenter som er koblet sammen gjennom et nettverk med en definert og lav responstid»

Tilgjengelighetssone: «Unike fysiske lokasjoner i en region. Hver sone består av et eller flere datasenter utstyrt med egen strømforsyning, kjøling og nettverk. For å sikre god stresshåndtering består hver region av minst 3 slike soner. Separering av sonene sikrer applikasjoner og data fra å bli utilgjengelig eller gå tapt ved feil i datasenter. Denne teknikken gjør at Azure kan sikre sine kunder 99,9% oppetid.»

Tilgjengelighetssett: «To eller flere virtuelle maskiner som er fordelt på forskjellige feildomener.»

Blob Storage Account: «En konto som er spesialisert for lagring av data i klynger. Denne kontotypen gjør det mulig for kunden å spesifisere tilgangsnivå som indikerer hvor ofte dataene i den kontoen er tilgjengelig.»

Konto for geografisk redundant lagring (GRS): "En lagringskonto hvor data replikeres synkront i en primær region og deretter replikeres asynkront til en sekundær region. Du kan ikke lese data direkte fra eller skrive data til den sekundære regionen tilknyttet GRS-kontoer»

Lokal redundant lagrinskonto (LRS): «En lagringskonto der data kun replikeres synkront innenfor en primærregion.»

Read-Access Geographically Redundant Storage (RA-GRS) Account: «En lagringskonto hvor data replikeres synkront i en primær region og deretter replikeres asynkront til en sekundær region. Du kan lese data direkte fra, men kan ikke skrive data til den sekundære regionen tilknyttet RA-GRS-kontoer.»

Tilgjengelighet i Azure

Azure regner ut oppetid for sine tjenester ved å trekke nedetid fra maksimalt antall minutter med oppetid. For eksempel så kan Azure garantere 99,99% oppetid for sine virtuelle maskiner når 2 eller flere instanser kjører i 2 eller flere tilgjengelighetssoner innenfor den samme regionen [13]. For å regne ut oppetid på tjenester for datalagring, også kalt feilraten, tar de antall mislykkede overføringer delt på totalt antall overføringer per time [14].

Tjenestenivåavtalen inneholder retningslinjer for hva som gjelder når kunden benytter seg av virtuelle maskiner levert av Azure. De opererer med tjenestekreditt og har definerte satser på hvor mye kunden skal kompenseres når tjenestenivået synker under gitte prosentverdier. Når oppetiden per måned går under 99,99% kompenseres kunden med 10% for det totale beløpet de blir belastet den måneden. Synker oppetiden til under 99% kompenseres kunden med 25% og hvis oppetiden komme under 95% skal kunden bli kompensert 100% av de månedlige kostnadene knyttet til virtuelle maskiner.

Også for datalagring finnes slike retningslinjer i tjenestenivåavtalen. Her sier avtalen at Azure sikrer kunden minst 99,99% suksess både ved lesing av data og skriving av data. Denne suksessraten gjelder uavhengig av hvilken type lagringskonto man benytter seg av.

Tilgjengelighet i AWS

Også AWS regner ut månedlig oppetid for sine instanser ved å trekke nedetid fra maksimalt antall minutter med oppetid. Samme prinsipp brukes til å regne ut både oppetid per time og oppetid per måned. I de tilfeller de inkluderte tjenestene ikke oppfyller serviceforpliktelsen vil kunden ha krav på kompensasjon i form av tjenestekreditt. Kompensasjonen regnes ut som en prosentandel av kostnadene kundene har hatt til tjenestene, men gjelder ikke for engangsbeløp som er betalt inn på forhånd, som for eksempel ved reservasjon av kapasitet [15].

Tilgjengeligheten mot lagringstjenesten til AWS som kalles Amazon S3 oppgis i det som heter feilrate, også likt som i Azure. Feilrate defineres i AWS som «det totale antallet interne serverfeil som returneres av Amazon S3-tjenesten med feilstatus InterntalError eller ServiceUnavailable, delt på det totale antallet forespørsler av gjeldende type forespørsel i løpet av et 5-minutters intervall». AWS regner ut oppetiden ved å trekke gjennomsnittet av feilfrekvensen fra 100% oppetid for hvert av 5-minutters intervall som inngår i den månedlige faktureringssyklusen. Også for lagringstjenestene har kunden krav på kompensasjon i de tilfeller tjenesten ikke oppfyller serviceforpliktelsen [16].

3. Implementering

Hosting av nettsider er en av flere hundre tjenester som tilbys på plattformene, og noe mange bedrifter vil ha nytte av å kunne. I dette kapittelet skal jeg derfor deployere en PHP-basert nettside på begge plattformer og forklare steg for steg hvordan dette gjøres. Jeg begynner med å kort forklare hva nettsiden inneholder før jeg går over til å beskrive framgangsmåten for implementering. Implementeringen er den viktigste delen her, så innholdet på nettsiden er kun ment som et eksempel. En oppsummering av mine erfaringer med implementeringen vil inngå i resultatdelen i denne rapporten.

Nettsiden er utviklet av forfatter i et tidligere emne kalt «IINI3003 Webprogrammering i PHP». Innholdet på siden er en bilkonfigurator som lar sluttbruker velge attributter og legge inn bestilling på en bil. Siden inneholder admin-funksjonalitet og er koblet til en database som holder styr på prislister og bestillinger. Prosjektet ligger lagret på forfatters område på GitHub og vil bli hentet ut derifra.

3.1 Implementering på Azure

1. Første steg for deployering på Azure var å få på plass nødvendig programvare. Jeg lastet ned PHP-utvidelsen til Xampp-serveren jeg allerede hadde på maskinen. Deretter lastet jeg ned Git som gjør det mulig å deployere PHP-kode til webapplikasjonen.
2. Inne i Git begynte jeg med å teste at applikasjonen fungerte ved å laste den ned lokalt og deretter kjøre den på serveren:

   git clone https://github.com/aadnehelmersen/php_prosjekt2

   cd php_prosjekt2

   php -S localhost:8080

   Koden over lager først en kopi av koden som befinner seg på GitHub i området «php_prosjekt2» for bruker «aadnehelmersen». Deretter navigerte jeg til mappen som koden ble kopiert til med cd-kommandoen, før jeg testet siden lokalt med den innebygde

PHP-serveren på port 8080. Dette fungerte og ga meg en bekreftelse på at filene i GitHub er strukturert riktig og er klare for deployering.

3. Azure sitt skall «Azure Power Shell» brukes til å kjøre kommandoer direkte i nettleseren. I Power Shell brukte jeg denne kommandoen til å opprette en distribusjonsbruker:

   *Az webapp deployment user set –user-name aadnehelmersen –password ******

4. Når distribusjonsbrukeren er opprettet er det denne som benyttes til å distribuere appen. Dette er altså ikke samme bruker som benyttes ellers på Microsoft sine tjenester.
5. Når distribusjonsbrukeren er opprettet må det opprettes en ressursgruppe. Azure definerer dette som en logisk container hvor tjenestene man benytter seg av kan administreres. Koden for å opprette ressursgruppen:

   Az group create –name ressurser –location "Norway East"

   Her har jeg opprettet en ressursgruppe med navn ressurser og valgt region «Norway East» som er min nærmeste region.

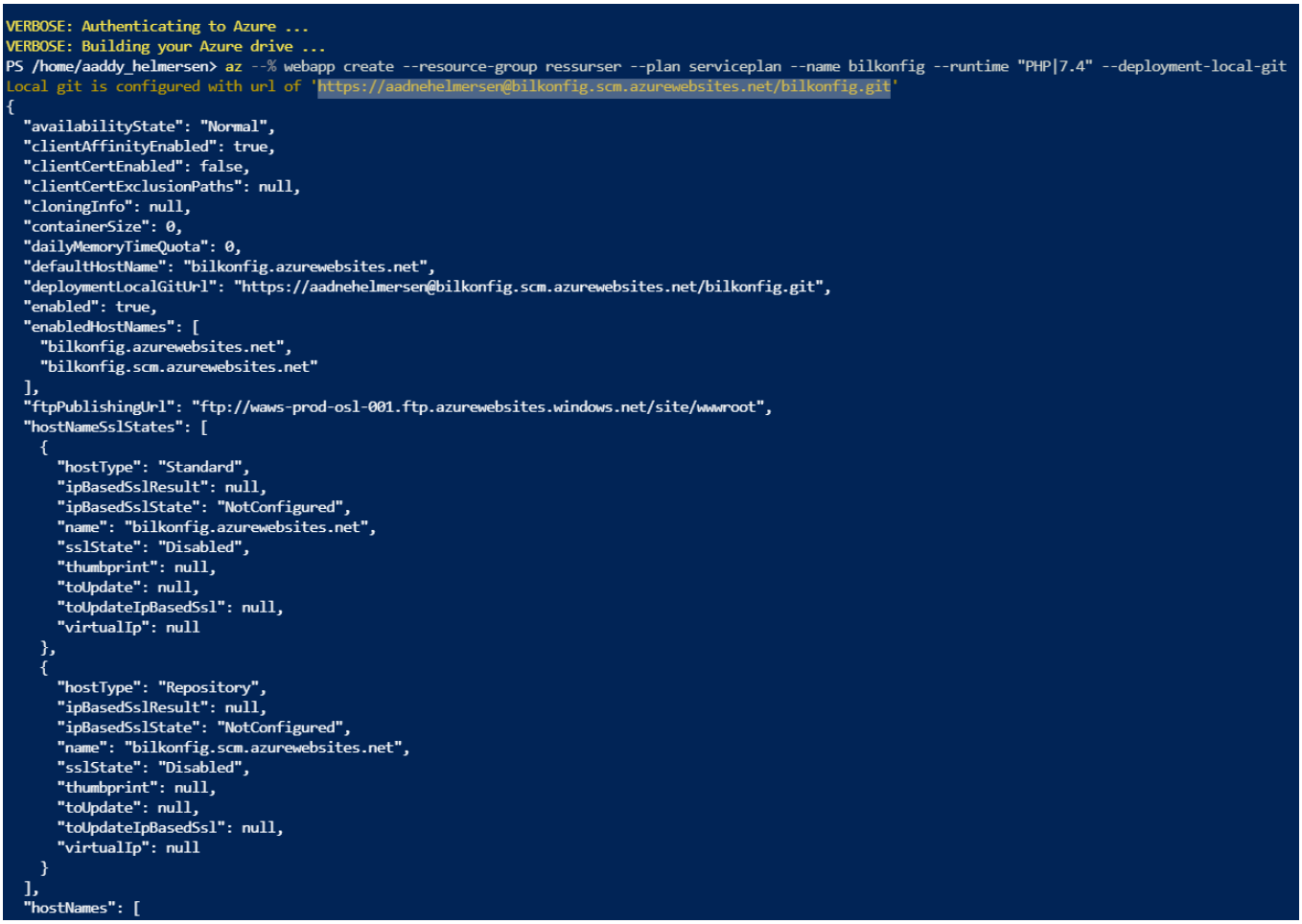
6. Neste steg da er å opprette en serviceplan. Dette er et oppsett hvor man bestemmer hvilke ressurser applikasjonen skal ha til rådighet. Denne kan endres senere og administreres gjennom distribusjonssenteret i Azure. Jeg har valgt å la Azure sette opp planen for meg ved å definere at jeg skal holde meg innenfor en grense som gjør at applikasjonen kan kjøres gratis, slik:

   az appservice plan create –name serviceplan –resource-group ressurser –sku FREE

7. Til nå har jeg opprettet bruker, ressursgruppe og serviceplan. Det er nå klart for å opprette applikasjonen i serviceplanen:

   az --% webapp create –resource-group ressurser –plan serviceplan –name bilkonfig –runtime "PHP|7.4" –deployment-local-git

I figur 8 har jeg opprettet en applikasjon med navn «bilkonfig» og under ser vi en skjermdump av utdata fra Power Shell som viser at det var vellykket.



Figur 8 PowerShell i Azure

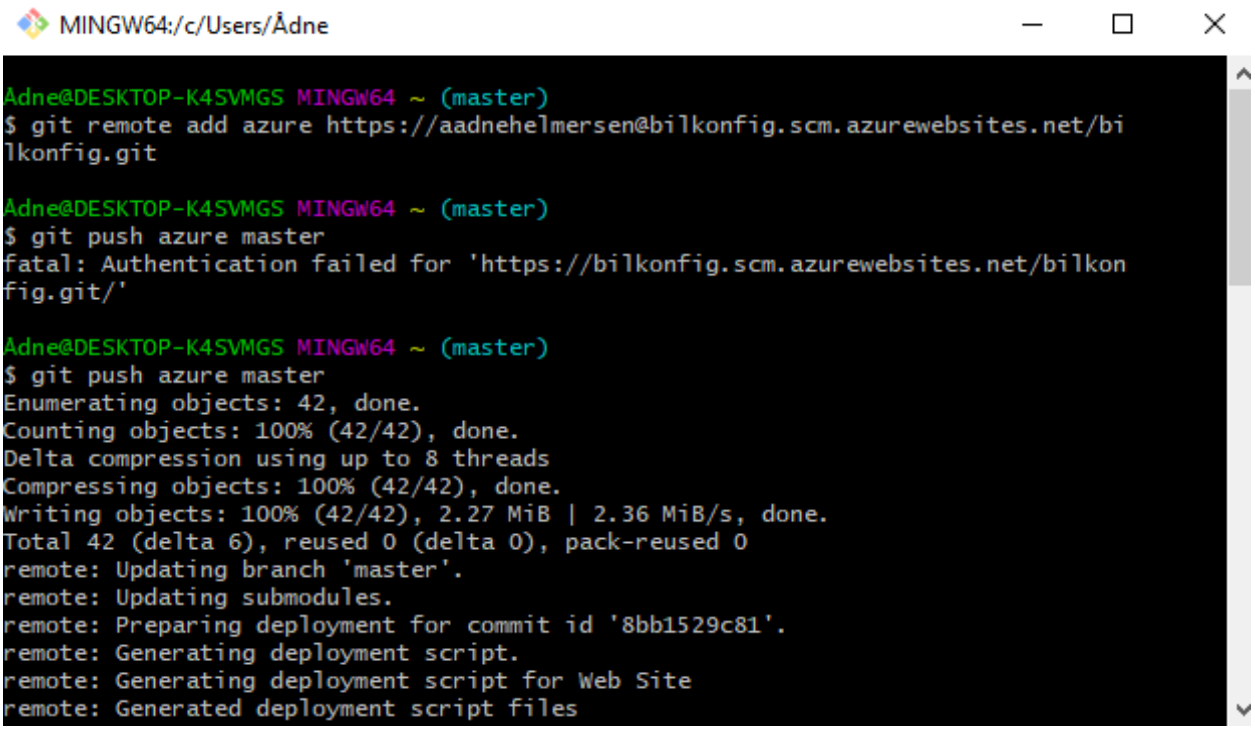
8. Etter dette steget har man opprettet en tom webapplikasjon i Azure som er klar til å ta imot kode. Dette gjøres lokalt i Git slik:

   git remote add azure
   https://aadnehelmersen@bilkonfig.scm.azurewebsites.net/bilkonfig.git

9. URLen ble opprettet i Power Shell i forrige steg og kommandoen forteller Git at jeg ønsker å opprette en forbindelse med dette Azure-området.

git push azure master

10. Siste som gjenstår da er å skyve master-grenen fra GitHub som vi kopierte i starten av prosessen. Git overfører forespørselen til Azure som ber om at det logges inn (med brukernavn og passord som ble opprettet for distribusjonsbruker). Skjermdump av terminalen vises i figur 2. «Authentication failed» skyldes at jeg skrev inn feil passord en gang.

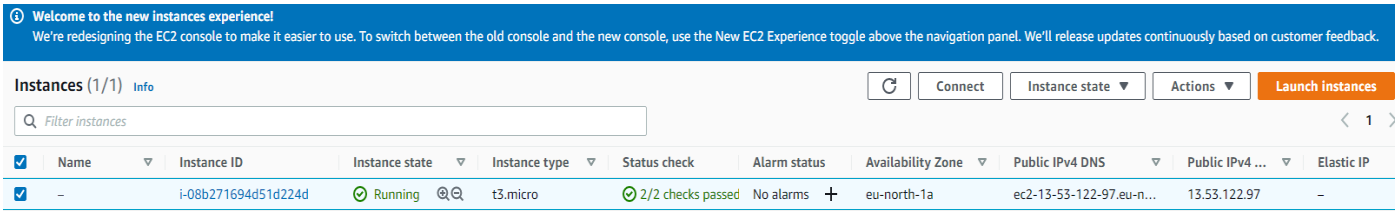


Figur 9 Skyve master-gren fra GitHub-området til Azure

URL til nettsiden: https://bilkonfig.azurewebsites.net/bilkonfig.php

3.2 Implementering på AWS

Før man kan gå i gang med å skyve koden ut i skyen må man starte opp en instans i AWS. Dette ble gjort gjennom brukergrensesnittet til AWS som kalles «AWS Management Console». Jeg valgte å starte opp en EC2-instans som kjører på Ubuntu. Jeg gikk også for en maskin som inngår i «free tier» som vil si at tjenesten kan kjøres gratis. Maskinen ble satt opp med gratis SSD-lagring og jeg la også til HTTP som en sikkerhetsgruppe, så det er mulig å nå applikasjonen også utenfor en SSH-klient. Til slutt lastet jeg ned et nøkkelpar (en fil man lagrer lokalt for å få tilgang til å konfigurere instansen) og kjørte i gang EC2-instansen. Under vises en skjermdump av konsollen som holder en oversikt over instansene som kjører på brukeren.



Figur 10. Skjermdump av AWS-konsollen som holder oversikt over instansene man har startet opp

1. Med koden i GitHub og EC2 oppe og går er det klart for å sette i gang med deployeringen. Første steg er å åpne en SSH-klient, og også her valgte jeg å bruke Git. Deretter kobler man seg til EC2-instansen gjennom Git med kommandoen

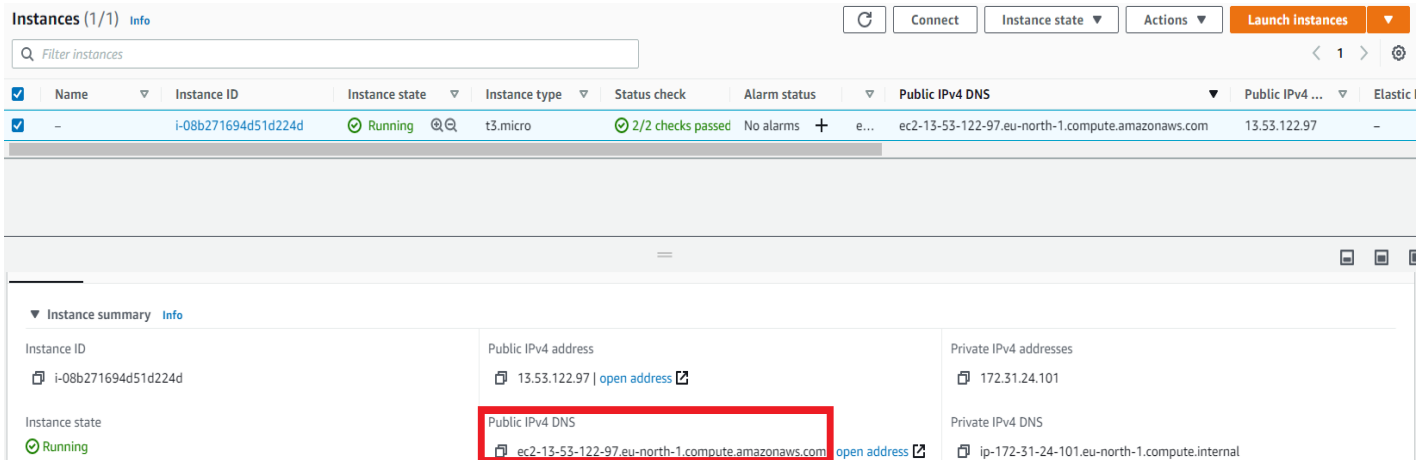
   ssh -i "php-key.pem" ubuntu@ec2-13-53-122-97.eu-north-1.compute.amazonaws.com

   php-key.pem er nøkkelparet jeg nevnte tidligere, og resten av kommandoen er adressen til EC2-instansen jeg har startet opp i AWS.

2. Neste steg da er å sørge for at instansen har alle nødvendige pakker for å kunne kjøre en PHP-applikasjon. Jeg kjørte først kommandoen *Sudo apt-get update* som sørger for å installere nyeste versjon av alle pakker som finnes på instansen. Deretter kjørte jeg kommandoen

   sudo apt-get install apache2

Denne kommandoen installerer Apache på instansen og lar oss styre innholdet på siden gjennom SSH-klienten. Det samme måtte jeg gjøre for å installere PHP og mcrypt på EC2-instansen. Med apache2, PHP og mcrypt installert er nettstedet klart til bruk. Figur 4 viser hvordan man finner offentlig DNS (IPv4) for sitt område.



Figur 11. Skjermdump av konsollen som viser hvor man finner IPv4-adressen

3. Når man åpner adressen, finner man mappa hvor filene skal plasseres. Standard sti til mappa er /var/www/html. Man navigerer seg til denne mappa i Git med cd-kommandoen slik: *cd /var/www/html* og lister ut innholdet i mappa med ls-kommandoen. Den eneste fila som befinner seg i mappa da er index.html. Denne kan man slette, for deretter å fylle mappa med filene som befinner seg på GitHub.
4. Man kloner innholdet på GitHub ved å kopiere HTTPS-lenken på området og kjører denne inn i git clone-kommandoen slik:

```
ubuntu@ip-172-31-24-101:/var/www/html$ sudo git clone https://github.com/aadnehelmersen/php_prosjekt3
Cloning into 'php_prosjekt3'...
remote: Enumerating objects: 36, done.
remote: Counting objects: 100% (36/36), done.
remote: Compressing objects: 100% (36/36), done.
remote: Total 36 (delta 3), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (36/36), done.
```

Figur 12. Skjermdump av Git hvor innholdet på GitHub blir klonet

Siden hjemmesiden på webapplikasjonen min heter index.php er det nå nok å bruke URL til mappa php_prosjekt3 og den endelige URLen ble da: http://ec2-13-49-57-185.eu-north-1.compute.amazonaws.com/php_prosjekt3/

4. Resultater

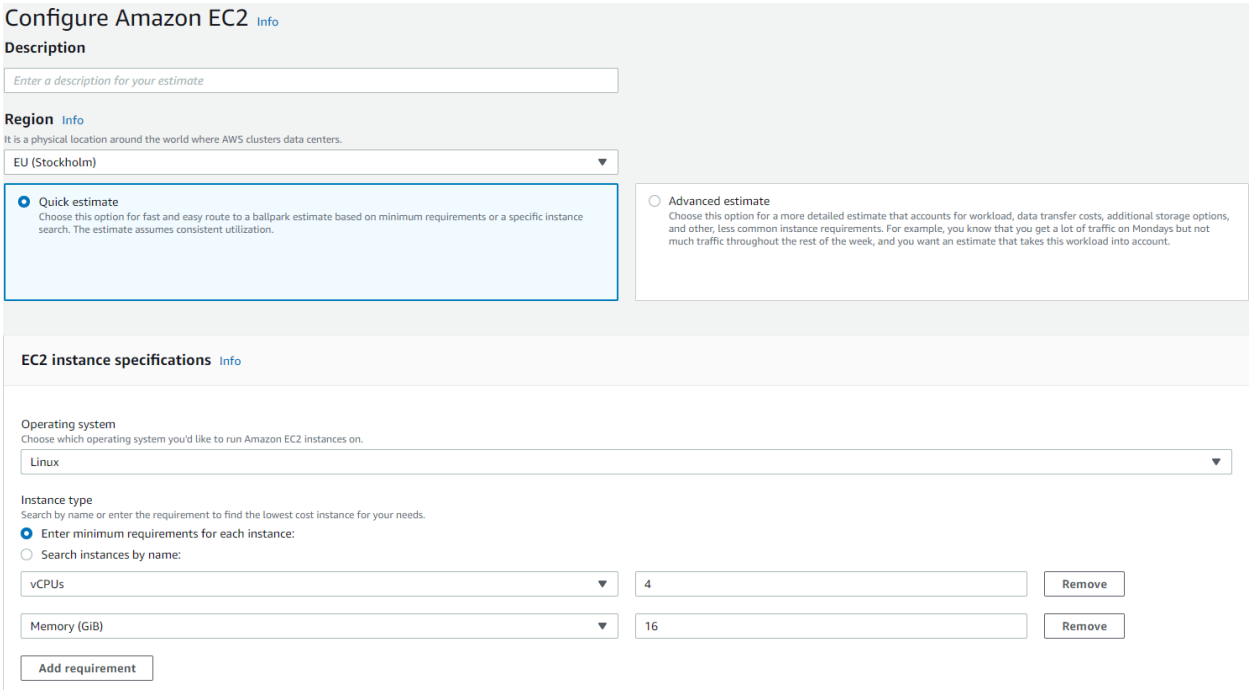
I kapittel 2 så jeg på teorien som beskriver tjenestene til plattformene. I dette kapittelet skal resultatene presenteres slik at det er mulig å se hvor plattformene skiller seg fra hverandre. Jeg begynner med å se på prising og skalering for tjenestene. Her skal jeg presentere utfordringene knyttet til dette, for så å sette opp et eksempel på hvordan man kan sammenligne pris for Azure og AWS. Deretter går jeg over på prøveperioder og presenter hva jeg har funnet ut der. Videre skal tjenestenivåavtalene og tilgjengeligheten sammenlignes. Hensikten her er å så på om det finnes noen forskjeller hva angår tjenestekreditt tilbake til kunde når tilbudet fra leverandør ikke yter som lovet. Til slutt i dette kapittelet vil jeg oppsummere resultatene fra implementeringen av webapplikasjon på plattformene. Her vil jeg gå nærmere inn på eventuelle hindre og utfordringer, samt konkludere med hvilken plattform jeg foretrekker.

4.1 Pris og skalering

Det store antallet tjenester som er tilgjengelig for sluttbruker innen skyteknologi gjør det vanskelig å sammenligne pris for plattformene. Det er veldig mange faktorer som spiller inn på hva man ender opp med å betale for infrastrukturen sin, da alle tjenester er priset forskjellig og vil variere i pris etter hvor langt fram i tid man reserverer de. Det finnes mange prisanalyser på nett, men det er lite poeng i å presentere en av disse da de allerede er utdatert siden publiseringsdato på grunn av at prisene er i konstant endring. Jeg vil derfor gjennomføre en prisanalyse for begge plattformer ved hjelp av priskalkulatoren de tilbyr på sine hjemmesider. Kalkulatorene gjør det mulig å estimere pris for de fleste tjenester på plattformene, men i dette tilfellet skal jeg fokusere på virtuelle maskiner, som regnes som bærebjelken i infrastrukturen.

Priskalkulator AWS EC2

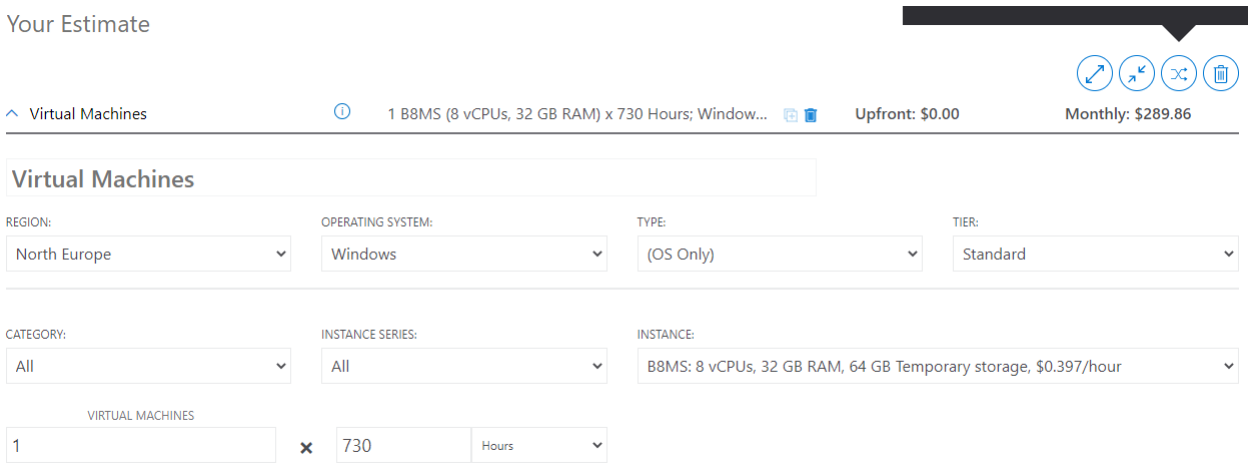
Nedenfor vises en skjermdump av priskalkulatoren til AWS. Man begynner med å velge region for deretter å sette opp den instansen man ønsker et prisestimat for. For å kunne velge type maskin må man først bestemme seg for hvilket operativsystem man skal kjøre. Dette vil avgjøre hvilke maskiner som blir tilgjengelig i kalkulatoren. Deretter kan man enten velge å søke opp en maskin etter navn, eller man kan velge å plotte inn minstekravene og la kalkulatoren finne den best egnede maskinen.



Figur 13. Priskalkulator AWS https://calculator.aws/#/createCalculator/EC2

Priskalkulator Azure VM

Nedenfor vises en skjermdump av priskalkulatoren til Azure. Fremgangsmåten er veldig lik som i AWS, men man har her mulighet til å filtrere ut maskiner etter serie og kategori før man åpner nedtrekkslisten, som gjør ting mer oversiktlige.



Figur 14. Priskalkulator Azure https://azure.microsoft.com/en-us/pricing/calculator/?&ef_id=Cj0KCQjw4ImEBhDFARIsAGOTMj9-ZCCZ97Wo9UWhX8mCuJag67ICZvM_M3NKdku6L9kgJV9VjwpbjqQaAvSOEALw_wcB:G:s&OCID=AID2100088_SEM_Cj0KCQjw4ImEBhDFARIsAGOTMj9-ZCCZ97Wo9UWhX8mCuJag67ICZ

Jeg skal nå konfigurere 4 like par med maskiner på begge plattformer for å sammenligne månedsprisen mellom disse. Forskjellen på maskinene vil være hvilke oppgaver de er satt til å gjennomføre. Den første maskinen vil være en hybridmaskin som skal fungere greit til de fleste oppgaver (omtales i skyen som «General-purpose»), den andre maskinen vil ha kraftigere prosessor og er ment for kjøring av større applikasjoner med store krav til prosessorkraft («Compute-Optimized»), den tredje maskintypen vil være best egnet for arbeid hvor det skal prosesseres store datasett («Memory-Optimized») og den siste maskintypen vil være velegnet for tungt arbeid mot data som er lagret lokalt («Storage-Optimized») [17]. Alle maskinene vil kjøre på Linux Ubuntu med Stockholm som region for AWS og Nord-Europa som region for Azure. Alle maskiner settes også opp med 8 prosessorkjerner.

Maskintype	AWS	AWS RAM (GB)	Azure	Azure RAM (GB)
General-Purpose	t4g.2xlarge	32	B8MS	32
Compute-Optimized	c6g.2xlarge	16	F8	16
Memory-Optimized	r6g.2xlarge	64	E8a v4	64
Storage-optimized	i3en.2xlarge	64	L8s v2	64

Tabell 1. Spesifikasjoner for testmaskiner

«Pay as you go», også kalt "On demand" er modellen som gjør det enklest å sammenligne maskinene. Tabell 2 viser prisen for maskiner med lik ytelse innenfor samme kategori for begge plattformer.

Maskintype	AWS	AWS Pris (pr måned)	Azure	Azure Pris (pr måned)
General-Purpose	T4g.2xlarge	200,90 USD	B8MS	265,77 USD
Compute-Optimized	c6g.2xlarge	213,16 USD	F8	329,96 USD
Memory-Optimized	r6g.2xlarge	315,57 USD	E8a v4	411,72 USD
Storage-optimized	i3en.2xlarge	692,04 USD	L8s v2	502,24 USD

Tabell 2. Prisoversikt "Pay as you go"

Reservering av ressurser

Både AWS og Azure tilbyr rabattløsninger for kunder som knytter seg til et gitt forbruk over en viss periode (1 og 3 år per nå). Dette kalles reservering av ressurser og er en måte for aktørene å sikre seg kundene over lengere perioder. Etter litt eksperimentering i kalkulatorene har jeg avdekket et skille mellom Azure og AWS, der Azure ikke har like mange alternativer for reservering av ressurser som det AWS har. Se oversikt i tabell 3.

	AWS	Azure
Binde seg til abonnement på 1 eller 3 år	X	X
Betale hele beløpet på forhånd for å oppnå høyere rabatt	X	
Betale en del av kostnadene på forhånd og balansen per måned	X	
Betale månedlig selv om instansene er reservert for en tidsperiode (fortsatt med rabatt)	X	X

Tabell 3. Reservering av ressurser, AWS vs Azure

De virtuelle maskinene eller instansene vi så på i forrige analyse kunne blitt billigere hvis man hadde reservert de for en bestemt periode. Jeg skal nå gjennomføre en analyse med samme maskiner, men med reservering av ressurser for 1 og 3 år. Det er viktig å merke seg at den prisen som gjelder når reservasjon blir gjort vil gjelde for hele perioden. Det vil si at en bedrift ikke vil få nytte av at prisene går ned i løpet av perioden.

Reservasjon av ressurser i 1 år:

Maskintype	AWS	AWS Pris (pr måned)	Azure	Azure Pris (pr måned)
General-Purpose	T4g.2xlarge	127,02 USD	B8MS	193,25 USD
Compute-Optimized	c6g.2xlarge	134,90 USD	F8	224,50 USD
Memory-Optimized	r6g.2xlarge	196,22 USD	E8a v4	242,05 USD
Storage-optimized	i3en.2xlarge	473,04 USD	L8s v2	319,97 USD

Tabell 4. Reservering av ressurser for 1 år, AWS vs Azure

Reservasjon av ressurser i 3 år:

Maskintype	AWS	AWS Pris (pr måned)	Azure	Azure Pris (pr måned)
General-Purpose	T4g.2xlarge	86,58 USD	B8MS	128,75 USD
Compute-Optimized	c6g.2xlarge	91,47 USD	F8	150,45 USD
Memory-Optimized	r6g.2xlarge	153,63 USD	E8a v4	162,56 USD
Storage-optimized	i3en.2xlarge	304,41 USD	L8s v2	212,44 USD

Tabell 5. Reservering av ressurser for 3 år, AWS vs Azure

4.2 Prøveperiode

4.2.1 Azure Free Tier

Når man oppretter gratiskonto i Azure og starter på prøveperioden har man 200 Amerikanske dollar i kreditt man kan bruke på de tjenestene man ønsker. Denne kreditten varer i 30 dager og må brukes opp innen denne perioden. 30 dager og 200 dollar gjelder også hvis man oppretter «Pay as you go» fra dag 1 [18].

Kreditten man får ved opprettelse av konto gjelder altså for alle tjenester, også de som ikke er gratis i utgangspunktet. Et begrenset utvalg av tjenestene har man mulighet til å benytte seg av i opptil 12 måneder, og et eksempel er en virtuell maskin man kan kjøre i opptil 750 timer per måned. I tillegg finnes det tjenester på Azure som alltid er gratis, som kan benyttes så lenge man ønsker.

I Azure portal har man mulighet til å filtrere ut de tjenestene som er gratis til enhver tid. Dette gjør at man unngår kostnader man ikke regner med når man prøver seg fram med tjenestene på plattformen.

Betalingsmåte må legges inn i det kontoen opprettes slik at Azure har mulighet til å belaste kunden hvis forbruket går utover det som er inkludert i gratisversjonen. Har man benyttet seg av tjenester som normalt koster penger i løpet av 12 måneders perioden må man oppgradere til «Pay as you go» for å kunne fortsette på arbeidet etter perioden er over [18].

4.2.2 AWS Free Tier

AWS beskriver et tredelt tilbud av prøveperioder for sine kunder. Ulike tjenester hører under ulike prøveperioder, og AWS skiller mellom 12 måneders prøveperiode hvor man får tilgang til tjenestene gratis i 12 måneder, for deretter å gå over til «Pay as you go», tjenester som alltid er gratis og tjenester man får teste ut gratis i en kort periode (Free trials).

Tjenestene som inngår i 12 måneder prøveperiode er gratis opptil et visst forbruk, og må betales for når man bruker mer enn grensene for hver enkelt tjeneste. Etter 12 måneder blir man belastet

på vanlig måte for alle disse tjenestene. Tjenester som alltid er gratis kan brukes så lenge man vil innenfor visse grenser. Her er det egne grenser og satser som gjelder for hver enkelt tjeneste, slik at betaling automatisk starter opp i det man går over disse. For å kunne ta i bruk tjenestene kreves det at man legger inn betalingsinformasjon slik at AWS kan belaste en konto for forbruk utover satte grenser.

Alle som ønsker, kan opprette prøvekonto i AWS og begynne å teste ut tjenestene de tilbyr. Det eneste unntaket som gjelder, er at organisasjoner som allerede benytter seg av AWS kun får ha en Free tier-konto [19].

4.3 Tilgjengelighet

4.3.1 Tilgjengelighet i Azure

Azure lover kundene sine 99,99% oppetid hvis de virtuelle maskinene er deployert over 2 eller flere tilgjengelighetssoner innenfor samme region. Hvis kunden har deployert maskiner i et tilgjengelighetssett lover de 99,95% oppetid. Har man deployert single maskiner (ingen redundans) vil oppetid variere etter hvilken type harddisk som finnes i maskinene (SSD er dyrere, men gir en høyere oppetidsprosent) [13]. Figur 15 nedenfor viser tjenestenivået og tjenestekreditten Azure lover kunden når de virtuelle maskinene er deployert over 2 eller flere tilgjengelighetssoner.

MÅNEDLIG OPPETIDSPROSENT	SERVICEKREDITT
<99,99%	10%
<99%	25%
<95%	100%

Figur 15. Oppetidsprosent med virtuelle maskiner deployert over 2 eller flere tilgjengelighetssoner https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

Figur 16 viser lovet tjenestenivå og tjenestekreditt i Azure, hvor kunden har deployert maskiner i et tilgjengelighetssett.

MONTHLY UPTIME PERCENTAGE	SERVICE CREDIT
< 99.95%	10%
< 99%	25%
< 95%	100%

Figur 16. Oppetidsprosent med virtuelle maskiner deployert maskiner i et tilgjengelighetssett https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

Figur 17 viser lovet tjenestenivå og tjenestekreditt i Azure hvor kunden har deployert enkle instanser. Her er det ikke tatt hensyn til redundans med bruk av flere tilgjengelighetssoner og man er da mer utsatt for nedetid som følge av katastrofer lokalt i datasentre. Som vi ser så er tjenestenivået avhengig av disktype og tjenestenivået vil alltid ligge på det som gjelder for billigste disktype i en maskin.

UPTIME PERCENTAGE (PREMIUM AND ULTRA SSD)	UPTIME PERCENTAGE (STANDARD SSD MANAGED DISK)	UPTIME PERCENTAGE (STANDARD HDD MANAGED DISK)	SERVICE CREDIT
< 99.9%	< 99.5%	< 95%	10%
< 99%	< 95%	< 92%	25%
< 95%	< 90%	< 90%	100%

Figur 17. Oppetidsprosent hvor kunden ikke har sørget for redundans https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

I figur 18 ser vi en oversikt over hvilke oppetidsprosenter og tjenestekreditt som gjelder for lagringstjenestene i Azure. Som vi kan se er det mange faktorer som avgjør hvilket nivå man kan forvente. Se kapittel 2.6 for definisjoner og hva som skiller de ulike lagrinskontoene fra hverandre.

Servicekreditt - Hot Blobs i LRS-, ZRS-, GRS- og RA-GRS-kontoer (skriveforespørsler), blobs i Block Blob Storage-kontoer og filer i lagringskontoer:

MÅNEDLIG OPPETIDSPROSENT	SERVICEKREDITT
<99,9%	10%
<99%	25%

Servicekreditt - Hot Blobs i RA-GRS (les forespørsler) Kontoer:

MÅNEDLIG OPPETIDSPROSENT	SERVICEKREDITT
<99,99%	10%
<99%	25%

Servicekreditt - Cool Blobs i LRS, GRS, RA-GRS (skriv forespørsler) kontoer:

MÅNEDLIG OPPETIDSPROSENT	SERVICEKREDITT
<99%	10%
<98%	25%

Servicekreditt - Cool Blobs i RA-GRS (les forespørsler) Kontoer:

MÅNEDLIG OPPETIDSPROSENT	SERVICEKREDITT
<99,9%	10%
<98%	25%

Figur 18. Oppetidsprosenter og tjenestekreditt for lagringstjenester i Azure https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

4.3.2 Tilgjengelighet i AWS

AWS lover kundene sine en månedlig oppetidsprosent på 99,99% for sine virtuelle maskiner [15]. I tilfeller hvor oppetidsprosenten blir lavere enn dette har man som kunde rett på tjenestekreditt etter satsene presentert nedenfor. Dette gjelder på samme måte som i Azure, når man har opprettet redundans ved å deployere instanser i flere enn 1 tilgjengelighetssone. AWS lover en oppetid på 90% per time for enkle EC2-instanser og kunden slipper da å betale for timer hvor oppetiden havner under 90%.

Månedlig oppetidsprosent	Prosent av tjenestekreditt
Mindre enn 99,99%, men lik eller større enn 99,0%	10%
Mindre enn 99,0%, men lik eller større enn 95,0%	30%
Mindre enn 95,0%	100%

Figur 19. Oppetidsprosenter og tjenestekreditt for EC2-instanser i AWS. https://aws.amazon.com/compute/sla/

Nedenfor kan vi se hvilken oppetidsprosent kundene kan forvente av lagringstjenestene til AWS.

Månedlig oppetidsprosent	Prosent av tjenestekreditt
Mindre enn 99,9%, men større enn eller lik 99,0%	10%
Mindre enn 99,0%, men større enn eller lik 95,0%	25%
Mindre enn 95,0%	100%

Figur 20. Oppetidsprosenter og tjenestekreditt for lagringstjenester i AWS https://aws.amazon.com/s3/sla/

4.4 Erfaringer implementering

I kapittel 3 deployerte jeg en PHP-applikasjon på AWS og Azure etter framgangsmåter basert på best practice. Jeg forsøkte å få prosessen så lik som mulig for begge leverandørene for å gjøre det enklere å plukke opp forskjeller som måtte finnes mellom plattformene når det kommer til hosting av webapplikasjoner. Det viste seg likevel å være noen større forskjeller mellom de på veien fra filer på GitHub til deployert webside.

Begge plattformer tilbyr mulighet for gratis hosting av applikasjoner. De har begge «free tier» tjenester med begrenset kapasitet som gjør det mulig å få praktisk erfaring med tjenestene uten å måtte betale for dem. Begge plattformer krever da at man oppretter et abonnement og legger inn betalingsmåte slik at man kan belastes hvis man overskrider maksgrenser for lagring og annen kapasitet i infrastrukturen. I mitt tilfelle er ikke dette noe jeg behøver å tenke over da jeg kun har deployert siden som en demonstrasjon.

Som sagt så var utgangspunktet likt for begge framgangsmåtene, men allerede i første steg av deployeringen så var det forskjeller mellom dem. Framgangsmåten i Azure baserer seg mere på koding i Cloud Shell, mens det i AWS er mere som kan gjøres gjennom brukergrensesnittet i konsollen. Det kan tenkes at en med erfaring som er ute etter effektivitet og standardisering setter pris på å kunne kode hele oppsettet. Dette gjør det enklere å holde styr på versjonslogger og det vil også være enklere å gjøre endringer når man oppdager feil. For nye brukere vil dette derimot være mere tungvint da man må kunne syntaks i tillegg til å ha en dypere kunnskap om hvilke tjenester som kreves for sitt oppsett i skyen. Det var enklere for meg å forstå hva som skjedde i hvert steg når igangsetting av instans skjedde gjennom brukergrensesnitt enn i Cloud Shell.

Det skal derimot sies at dokumentasjonen som finnes på Azure docs er veldig god og jeg støtte ikke på noen problemer underveis der. Det finnes veiledninger for alle tjenester på Azure med framgangsmåter og tips til hvordan man skal gå fram. Det samme gjelder for AWS og jeg opplever også AWS sitt brukergrensesnitt som ryddigere og enklere å forstå.

5. Diskusjon

5.1 Pris og prøveperioder

Å finne ut hvilken leverandør som er billigst er som nevnt tidligere ingen enkel oppgave på grunn av alle faktorer som spiller inn i en slik analyse. Jeg har sett på virtuelle maskiner av typen general, compute, memory og storage-optimized når jeg gjorde en sammenligning av priser mellom plattformene. Det viser seg at de 3 førstnevnte typene er billigst hos AWS med god margin, så basert på denne analysen kan jeg derfor konkludere med at AWS er billigere når det kommer til prosessering. Dette gjelder altså både for «Pay as you og» og ved reservering av ressurser 1 eller 3 år i forveien.

Samtidig så er de virtuelle maskinene bare en liten del av infrastrukturen, og jeg har nå kun sammenlignet priser med Linux som operativsystem. Søker man opp prissammenligninger på nett vil man finne resultater hvor både AWS og Azure går ut som vinner, nettopp fordi det er forskjellige tjenester og faktorer som sammenlignes fra gang til gang. For å illustrere hvor store forskjellene kan være skal jeg her vise hvordan resultatet endres for tilsvarende «General-Purpose» maskiner som tidligere når man endrer operativsystem fra Ubuntu til Windows Server:

Maskintype	**AWS**	**AWS Pris (pr måned on demand)**	**Azure**	**Azure Pris (pr måned on demand)**
General-Purpose	T3.2xlarge	359,74 USD	B8MS	289,81 USD

Tabell 6. Ny sammenligning av "General-purpose" maskiner, nå med Windows Server

Hva som avgjør om AWS eller Azure er billigst for deg eller din organisasjon kommer an på flere faktorer, som hvor du befinner deg i verden, hvilke tjenester og operativsystem som er nødvendig i din infrastruktur, om du har mulighet til å betale for reserverte ressurser og mange flere. Priskalkulatoren som begge plattformer tilbyr er et godt hjelpemiddel som gjør det mulig å se hvor man kan kutte kostnader og gjøre det enklere å sette opp budsjett for deployering av infrastruktur i sky.

Når det kommer til prøveperioder er det Azure som går av med seieren da de har det beste tilbudet etter min mening. Med 200 dollar i kreditt er det ingen begrensning for hvilke tjenester man kan teste ut den første måneden. Det vil si at man har en hel måned til å sette opp et

testmiljø for å se hvordan dette fungerer, for deretter å ha 11 måneder med begrenset kapasitet som kan suppleres med «On demand»-ressurser.

5.2 Tilgjengelighet og servicenivåavtaler

Jeg har i resultatene sett på oppetidsprosenter for tjenester innen lagring og instanser eller virtuelle maskiner. Jeg kan ut fra tallene presentert i tabellene konkludere med at både AWS og Azure tilbyr en oppetidsprosent på 99,99% for virtuelle maskiner i infrastrukturen. Dette tallet er basert på et optimalt scenario hvor kunden selv sørger for redundans ved å deployere instansene over 2 eller flere tilgjengelighetssoner innenfor samme region. Dette gjelder for begge leverandører. Ved deployering av enkle instanser vil valg av disktype naturligvis spille en stor rolle for oppetidsprosenten, og er noe man bør tenke på hvis man for eksempel skal lage et oppsett for hosting an en enkel nettside slik jeg har gjort i denne oppgaven.

For lagringstjenester er det flere faktorer som spiller inn og Azure går mere i dybden på hva kunden kan forvente refundert av tjenestekreditt når tjenestene ikke lever opp til forventet oppetid. Det som skiller plattformene her, er måten de regner ut feilraten for skrive- og leseforespørsler på. AWS tar utgangspunkt i 5minutts intervaller, mens Azure benytter seg av hele timer. Sistnevnte vil nok gi mer nøyaktige tall på antall forespørsler som feiler mot lagringstjenestene så jeg velger å sette Azure som vinner akkurat her.

Som vi har sett så er det veldig lite som skiller aktørene på dette området og jeg vil si at denne sammenligningen er en svakhet for oppgaven da den ikke belyser noen forskjell av betydning mellom plattformene. Jeg tror likevel teoridelen i kombinasjon med resultatene er verdifull informasjon som supplerer resten av oppgaven.

5.3 Vil en overgang til infrastruktur basert på skyteknologi lønne seg?

Jeg har til nå sett på noen av de faktorene som kan være avgjørende når en bedrift skal bestemme seg for hvilken leverandør som best dekker sine behov. Det kan likevel være vanskelig å se hvilke fordeler man kan oppnå med en slik overgang, og om den faktisk kommer til å lønne seg. I 2020 publiserte konsulentselskapet Capgemini en artikkel [20] som tar for seg årsaker til at en bedrift ønsker å flytte infrastrukturen sin ut i skyen:

- «De har et ønske om å legge ned og avslutte bruken av dagens datasentre»
- «De har et behov for tredjeparts styring av infrastrukturen»
- «Et ønske om å redusere kostnader»
- «Time to market»
- «Et ønske om å forbedre virksomhetens smidighet og redusere teknisk gjeld»

Det er altså viktig å påpeke at kostnader ikke er den eneste faktoren som spiller inn på om en slik overgang kan lønne seg for en bedrift, da det ofte er flere fordeler de ønsker å oppnå med endringen. Samtidig er alle kunder ute etter å få konkrete tall på hva endringen faktisk koster, og det kan man finne ut ved å sette opp en investeringsanalyse;

Current State		Year 1	Year 2	Year 3	Year 4	Year 5
Hardware	Expense	$7,400,000	$7,400,000	$7,400,000	$7,400,000	$7,400,000
Hardware	Capital	$600,000	$600,000	$600,000	$600,000	$600,000
OS/VM	Capital	$900,000	$900,000	$900,000	$900,000	$900,000
Total Capital		$1,500,000	$1,500,000	$1,500,000	$1,500,000	$1,500,000
Total Expense		$8,100,000	$8,100,000	$8,100,000	$8,100,000	$8,100,000
Total Spend		$9,600,000	$9,600,000	$9,600,000	$9,600,000	$9,600,000

Future State		Year 1	Year 2	Year 3	Year 4	Year 5
Legacy	Expense	$7,600,000	$3,800,000	$0	$0	$0
Setup	Expense	$300,000	$100,000	$0	$0	$0
Migration	Expense	$2,200,000	$3,400,000	$0	$0	$0
Managed Services	Expense	$100,000	$1,200,000	$1,600,000	$1,600,000	$1,600,000
Azure IaaS	Expense	$300,000	$800,000	$1,100,000	$1,100,000	$1,100,000
Depreciation Note	Expense	WIP	WIP	N/A	N/A	N/A
Total Capital		$0	$0	$0	$0	$0
Total Expense		$10,500,000	$9,300,000	$2,700,000	$2,700,000	$2,700,000
Total Spend		$10,500,000	$9,300,000	$2,700,000	$2,700,000	$2,700,000

Savings	Year 1	Year 2	Year 3	Year 4	Year 5
Total Capital	$1,500,000	$1,500,000	$1,500,000	$1,500,000	$1,500,000
Total Expense	($3,100,000)	($1,900,000)	$4,700,000	$4,700,000	$4,700,000
Total Savings	($1,600,000)	($400,000)	$6,200,000	$6,200,000	$6,200,000

Figur 21. Eksempel investeringsanalyse migrering til sky. https://www.capgemini.com/2020/03/the-roi-of-moving-to-the-cloud/

Figur 21 viser en investeringsanalyse som er hentet fra samme artikkel som tidligere. Den er kun ment som et eksempel, men illustrerer på en god måte hvordan kostnadene vil løpe de første årene etter overgangen til sky.

Den første tabellen tar for seg et tradisjonelt «on premises» oppsett og viser utgiftene knyttet til dette gjennom 5 år. Tabell 2 viser hvilke utgifter den samme bedriften ville hatt gjennom de samme 5 årene med migrering til skyen. Ikke overraskende så vil man ha større utgifter de første par årene knyttet til migreringen som må være en sømløs overgang hvor bedriften fortsatt skal være operativ. Migreringen er en omfattende prosess som gjerne krever at man leier inn ekstern kompetanse for å sikre at bedriften har kapasitet til å drive som vanlig under perioden. Dette medfører store kostnader og det er nok hvor overkommelig disse er som avgjør om en bedrift har mulighet til å gjennomføre en så omfattende endring det er å migrere til skyen.

I tabell 3 ser vi hvordan bedriften sparer penger hvert år fra og med år 3, når utgifter knyttet til migrering, oppsett og drift av eldre systemer kuttes bort. Det er altså kostnadsbesparende å gjennomføre en migrering til skyen, men det krever store midler den første perioden. Det kan også tenkes at hvilke tjenester som inngår i infrastrukturen fra før, og hvilke tjenester man begynner å ta i bruk etter migrering vil være med å avgjøre hvor lønnsomt det blir. Kanskje har man kjørt den gamle serverparken med windows server, som vi har sett er mye dyrere enn Linux i skyen. Kanskje har ikke de IT-ansatte den kompetansen som kreves for å styre infrastrukturen gjennom sky og må kurses, eller til og med suppleres med flere ansatte.

Hvor lønnsomt det blir for en bedrift å flytte infrastrukturen ut i sky avhenger av flere faktorer, men det er liten tvil om at man vil oppnå fordeler samt at det på sikt kommer til å være kostnadsbesparende, selv om det krever mye å få gjennomført endringen.

6. Konklusjon og anbefalinger

Det store utvalget av tjenester som finnes på plattformene AWS og Azure gjør valget av skyleverandør mer komplisert enn ved første øyekast. På noen områder har jeg lyktes med å belyse forskjeller mellom leverandørene, mens det på andre områder har vært vanskelig å avdekke forskjeller som er av betydning for nye kunder. Årsaken til dette er en kombinasjon av at de benytter seg av samme teknologi og at konkurransen mellom de er så stor at de hele tiden justerer tilbudet etter konkurrentene.

AWS er fortsatt størst og har best geografisk tilgjengelighet sett over hele kloden. AWS har en større markedsandel enn Azure og tilbyr også flere tjenester. Azure har derimot størst vekst og har etablert datasentre i Norge. Begge lover en oppetidsprosent på over 99% for sine virtuelle maskiner, som er bærebjelken i infrastrukturen. Begge tilbyr også tjenestekreditt når tjenestene ikke yter som lovet.

Pris og skalering har fått mye plass i denne rapporten, da jeg anser reduserte kostnader som en avgjørende faktor. Som vi har sett er dette vanskelig å sammenligne da prisene er i konstant endring. Det er likevel noen større forskjeller mellom de i det jeg skriver denne rapporten. AWS tilbyr flere valgmuligheter for de som ønsker å reservere ressurser for å oppnå høyere rabatt. AWS har også billigere virtuelle maskiner som kjører på Linux. Azure sine virtuelle maskiner er billigst hvis de skal kjøre på Windows Server. Azure tilbyr en bedre prøveperiode med 200 dollar i tjenestekreditt den første måneden, som betyr at man får testet ut alle tjenester man ønsker før man begynner å betale for forbruket. Sistnevnte gjør at jeg velger å sette Azure som vinner på pris per nå. Prøveperioden er viktig da den gjør det mulig å se hvordan plattformen dekker sitt behov og hva det kommer til å koste.

Det å skulle sammenligne alle tjenestene som finnes på plattformene er en omfattende oppgave og man vil nok aldri komme fram til en klar vinner som passer best for alle bedrifter. AWS kan se ut som en total vinner basert på noen av sammenligningene gjort i dette prosjektet, men konklusjonen blir at en slik sammenligning må gjennomføres sett opp mot en spesifikk bedrift. Både Azure og AWS tilbyr det beste som finnes av skyteknologi, så uansett hvilken man velger vil man oppnå store fordeler sammenlignet med on-premises infrastruktur.

7. Litteraturliste

[1] Leading Edge, *How is Cloud Computing different from Traditional IT Infrastructure?* Hentet fra: https://www.leadingedgetech.co.uk/it-services/it-consultancy-services/cloud-computing/how-is-cloud-computing-different-from-traditional-it-infrastructure/

[2] AWS, *Amazon Virtual Private Cloud*, 2021. Hentet fra: https://aws.amazon.com/vpc/?vpc-blogs.sort-by=item.additionalFields.createdDate&vpc-blogs.sort-order=desc

[3] AWS, *Types of Cloud Computing*, 2021. Hentet fra*:* https://aws.amazon.com/types-of-cloud-computing/

[4] AWS, *What is AWS*, 2021. Hentet fra: https://aws.amazon.com/what-is-aws/?nc1=f_cc

[5] Azure, *What is Azure?* 2021. Hentet fra: https://azure.microsoft.com/en-au/overview/what-is-azure/

[6] AWS, *Amazon EC2 Auto Scaling benefits.* 2021. Hentet fra: https://docs.aws.amazon.com/autoscaling/ec2/userguide/auto-scaling-benefits.html

[7] Azure, *Azure Autoscale.* 2021. Hentet fra: https://azure.microsoft.com/en-us/features/autoscale/

[8] AWS, *AWS Free Tier.* 2021. Hentet fra: https://aws.amazon.com/free/?all-free-tier.sort-by=item.additionalFields.SortRank&all-free-tier.sort-order=asc&awsf.Free%20Tier%20Types=*all&awsf.Free%20Tier%20Categories=*all

[9] Azure, *Create your Azure free account today*, 2021 Hentet fra: https://azure.microsoft.com/en-us/free/search/?&ef_id=CjwKCAjw1uiEBhBzEiwAO9B_HYl4USe7H-OzkQc0VLfjnoBfG14BJILi1IKPp2_cR6saEBz0ZZLhtRoCuvYQAvD_BwE:G:s&OCID=AID2100088_SEM_CjwKCAjw1uiEBhBzEiwAO9B_HYl4USe7H-OzkQc0VLfjnoBfG14BJILi1IKPp2_cR6saEBz0ZZLhtRoCuvYQAvD_BwE:G:s&gclid=CjwKCAjw1uiEBhBzEiwAO9B_HYl4USe7H-OzkQc0VLfjnoBfG14BJILi1IKPp2_cR6saEBz0ZZLhtRoCuvYQAvD_BwE

[10] James Montgomery, Sonia Leliii, *cloud SLA (cloud service-level agreement), TechTarget*, 01.2021. Hentet fra: https://searchstorage.techtarget.com/definition/cloud-storage-SLA

[11] Mina Nabi, Maria Toeroe, Ferhat Khandek, *Availability in the cloud: State of the art, ScienceDirect*, 2015 Hentet fra: https://www.sciencedirect.com/science/article/abs/pii/S1084804515002878?via%3Dihub

[12] Microsoft, *Regions and Availability Zones in Azure*, 09.04.2021, Hentet fra: https://docs.microsoft.com/nb-no/azure/availability-zones/az-overview

[13] Azure, *SLA for Virtual Machines*, 07.2020. Hentet fra: https://azure.microsoft.com/en-us/support/legal/sla/virtual-machines/v1_9/

[14] Azure, *SLA for Storage Accounts*, 06.2019. Hentet fra: https://azure.microsoft.com/en-us/support/legal/sla/storage/v1_5/

[15] AWS, *Amazon Compute Service Level Agreement*, 22.07.2020. Hentet fra: https://aws.amazon.com/compute/sla/

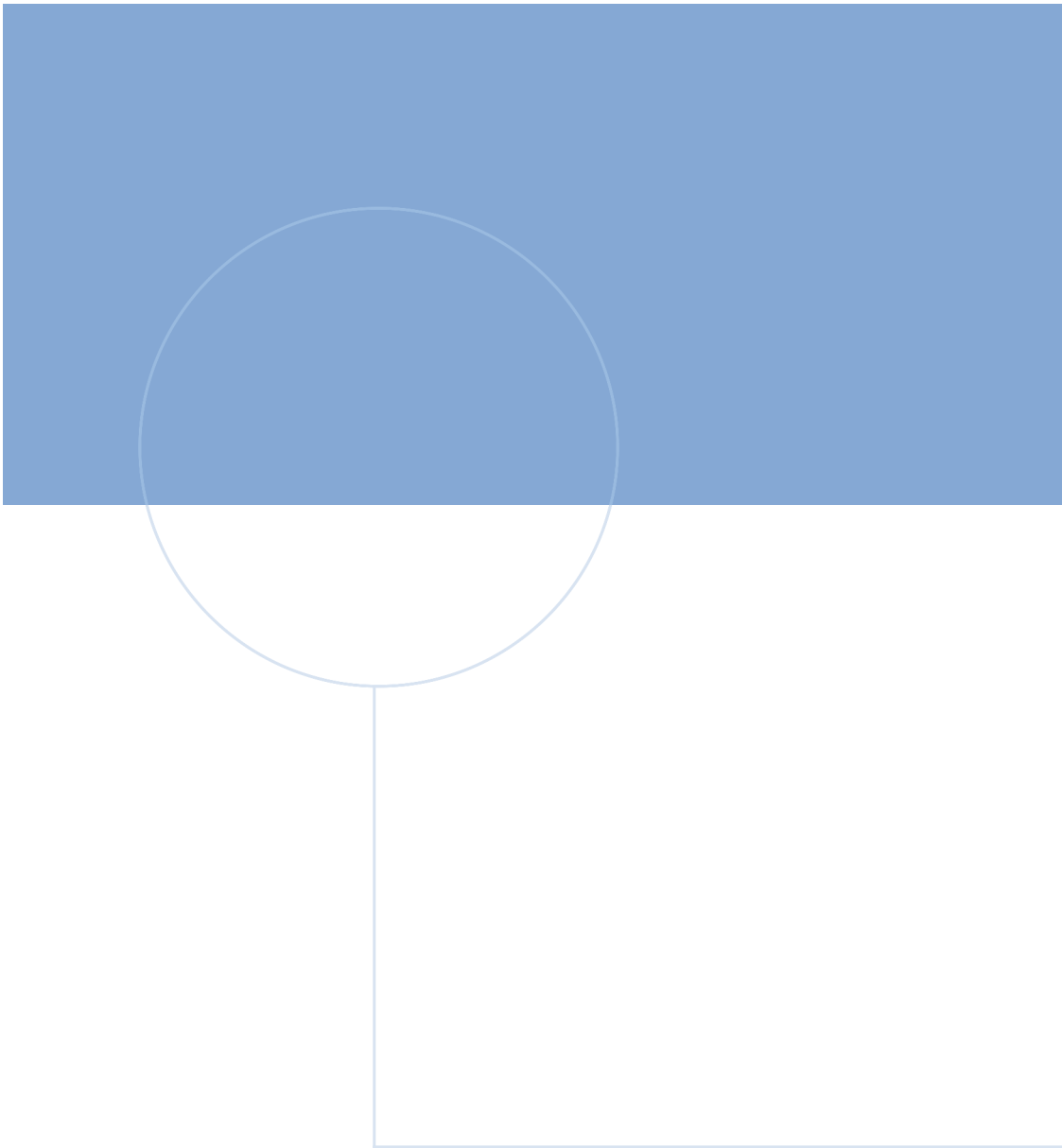
[16] AWS, *Amazon S3 Service Level Agreement*, 31.07.2019. Hentet fra: https://aws.amazon.com/s3/sla/

[17] AWS, *Amazon EC2 Instance Types*. Hentet fra: https://aws.amazon.com/ec2/instance-types/

[18] Microsoft, *Create free services*, 30.03.2021. Hentet fra: https://docs.microsoft.com/en-us/azure/cost-management-billing/manage/create-free-services

[19] AWS, *AWS Free Tier FAQ*. Hentet fra: https://aws.amazon.com/free/free-tier-faqs/

[20] Tejas Vadalia, *The ROI of moving to the cloud*, *Capgemini*, 02.03.2020. *Hentet fra: https://www.capgemini.com/2020/03/the-roi-of-moving-to-the-cloud/*



NTNU

Kunnskap for en bedre verden