Molnbaserad Arkitektur

Molnbaserad arkitektur handlar om att använda placera ett datacenter på en separat klient. Alltså istället för att köpa en egen server som hanterar alla användare som använder sig utav den så betalar man för en virtuell server. Man kallar ofta detta för **klient, server model.**

De tre vanligaste företagen som förser sådana tjänster är Google, Amazon och Microsoft.

När man skaffar en molnbaserad tjänst så tar företaget som du köper av hand om allt som underhåll brandväggar osv.

Om man lägger hela datacentret i molnet (separata servrar) kallas det ”**cloud base deployment**”.

Om man istället använder en egen server så kallar man det ”**On-premise deployment**” eller ”**private cloud”.**

Man kan också göra en delad lösning där man har en egen server som kopplas ihop med publikt moln (ett separat datacenter).

Ett exempel på en molnlagrad service vore office 365. Du har word, powerpoint, teams etc på en separat server som du gör anrop till. Om du skriver i word med en webbläsare så skriver du egentligen på en annan dator någonstans som sparar filen i ditt namn.

När man skaffar en molnbaserad lösning att lagra sin tjänst så får man ofta välja själv vilken prestanda det ska ha. Man kan välja om man vill lägga mer krut i komputation, alltså processorkraft om du till exempel bygger ett system som ska beräkna enorma summor.

Eller så kan du välja om du vill ha mer fokus på minne. Du kanske har mycket data som ska laddas upp och ned konstant, tex youtube eller Netflix.

Ofta så grupperar man olika lösningar i olika grupper baserat på deras prestandainriktning. Så kallade **”Instance families”**

Dessa är

* **Generall purpose** (Allmänt OK på allt)
* **Computic Optimized** (Beräkningar, stora CPU:er)
* **Memory Optimized** (Mycket minne för att hålla mycket data samtidigt, minnet i servern)
* **Accelerated Computing** (sökningar i stora listor och register)
* **Storage Optimized** (Lagrar data på disk)

Maskinerna som skapar molnet som du placerar din tjänst i kan skalas upp och ned i prestanda, efter hur stort tryck det är på din tjänst. Alltså hur många användare som kör din tjänst. Detta kallas ”**vertical scaling”.**

**CaaS** = Compute as a service.

Man kan betala för dessa tjänster på lite olika vis. Det enklaste är att köpa ”**on demand**”. Då betalar du enbart för det som används.

Man kan också betala för en längre tid om man vet att man kommer att behöva tjänsten en lång tid framåt. Om man vill vara säker på att man har en vis kapacitet så kan man reservera en viss mängd prestanda som man då är låst vid för en längre tid.

Det finns olika standardiserade prismodeller för betalning:

* **On demand**. Betala efter användning. Du kan se till att din tjänst använder sig utav skalar upp och ner beroende på hur stor belastning den utsätts för. Dvs om du enbart har två användare online på måndag så kanske det bara krävs en serverdator för att tjänsten ska köras utan problem. Du används och betalas/fakturerats enbart den servern. PÅ tisdag så har vi kanske rea och då har vi 200 kunder. Vi behöver då 10 servrar som automatiskt hoppar igång och kör vår tjänst och sedan faktureras.
* **Savings plan**. Man betalar för en längre tid som man binder sig till. Kan bli billigare i längden.
* **Reserved Instance**. Man köper en specifik kapacitet som inte ändras. Om man finner att man behöver mer prestanda så kan det bli jobbigt. Man skriver ofta upp sig på 1 till 3 år.
* **Spot Instance**. Man får ett väldigt lågt pris men vår tjänst placeras i ganska så låg prioritet. Om de behöver göra underhåll till exempel så är det inte garanterat att vår tjänst flytas utan vi kan få nedtid. Om någon annan tjänst behöver mer prestanda kan vår tjänst stängas ned så att de får tillgång till vår server. Kort sagt så placeras vår instans där det finns plats i servrar som är ”över” och står tomma.

Två viktiga termer gällande betalning vid användande utav molnbaserade tjänster.

**”Capex”** = capital expense. Att lägga ut en stor summa i förväg. Bygga egna servrar i förväg till exempel.

”**Opex**” = operational expense. Att betala allt eftersom. Betala för användning. När man använder de publika molnen och enbart betalar för den hårdvara som används.

## Regioner

När man skaffar en molntjänst så behöver/kan man välja ett flertal olika regioner. Dessa regioner är den fysiska platsen där servern står. Data man sparar i molntjänsten lämnar då aldrig regionen man väljer men den kan röra sig mellan olika datacenter i regionen.

Olika regioner skiljer sig åt på några vis

Det första och kanske mest uppenbara är hur långt ifrån dig och kunden du utvecklar för din tjänst hamnar. Om du har en kund i Danmark så kan det vara bra att försöka placera din tjänst i en region nära Danmark. Det korta avståndet kan göra så att responstider blir lite kortare.

En viktig sak att tänka på gällande regioner är att det kan finnas olika regulationer och lagar i olika regioner. Om ett datacenter befinner sig i ett EU-land så måste GDPR följas. Dvs att personlig data måste hanteras på ett vist sätt.

Olika regioner kan också ha olika tjänster. Om regionen är lite mindre och inte like använd så lär infrastrukturen inte vara lika stark. Ned och uppladdnings hastigheter kan vara långsammare.

Pris kan också variera beroende på region.

Om man använder sig utav flera olika datacenter inom samma region för att ha en redundans ifall ett center skulle gå ned så kallar man det **Availability zones.**

Om man väljer att ha en kopia utav sin tjänst i en annan region utöver den som kunden använder så kallas det en **edge location.** Om man ska lansera sin tjänst i en ny region så kan man använda sig utav en edge location som gjorts i förväg för att snabba på processen.

Ett vanligt namn för edge locations är **CDN, Content Delivery Network**

## EC2

EC2 står för **elastic compute cloud.**

Det är en servertyp som erbjuds utav **AWS** som man kan placera sin tjänst på.

En **s3 (simple storage device)** är en disk yta i AWS. Används för att lagra filer.

1 h 08 min

För att skapa en EC2 server följ stegen nedan. Taget från Thomas dokument, finns på teams.

1. Logga in i AWS educate portalen och klicka AWS Account.
2. Starta ditt Starter Account.
3. Klicka refresh på knappen om det behövs. Här kommer vi vidare till en tredjeparts leverantör till AWS som erbjuder utbildningar. Detta ger oss 30$ att spendera i AWS per år.
4. Klicka AWS management console.
5. Bekanta er med konsolen (Web GUI) och välj EU-NORTH-1 som region.
6. Skapa en liten free tier linux dist. Kom ihåg att vi bara har 30$ att leka med. Jag skapar själv Amazon linux 2 ami.
7. Som ni ser behövs det lite rättigheter för att kunna göra detta.
8. Tricket är att använda AWS Resource Group. Men då vi inte är huvud admins för denna student VPC så kan vi inte ändra så att vi blir en användare i EU north.
9. Istället väljer vi region US East North Virginia igen. Där kan vi skapa EC2 maskiner.
10. Skapa EC2an och välj Configure Instance Details.
11. Välj request spot instance.
12. Välj next tills du kommer till Configure Security Group. Gruppen innan är Tags. Det används frekvent hos kunder, t.ex. för att specificera kostnadsställe. Så applikationsutvecklarna får ta sin kost för en staging miljö medans prod får ta sin osv.
13. Skapa en ny Security Group. Döp den till DNS.
14. I Security group regeln, tillåt SSH från din hemaddress IP och eller där du sitter nu och pluggar.
15. Välj Launch och skapa sedan nya SSH nycklar. Jag döper de till DNS environment.
16. Som ni ser står det att det inte går att ladda ned dessa nycklar igen. Håll de säkra och säkerhetskopierade. Ett vanligt problem hos vissa kunder är att dessa är borta. Speciellt om det är konsulter eller extern part som gjort lösningen eller installerat servrar.

## ELB, Elastic load balancer

För att kunna skapa möjligheten att skala server/dator kapaciteten upp och ned dynamiskt beroende på belastning, så behöver man ett system som övervakar trafiken och gör ändringar utifrån den. Detta system kallas **elastic load balancer** och det är detta system som sorterar vart olika instanser ska placeras och hur mycket hårdvara som behöver vara igång för att allt ska funka smidigt.

Beroende på vilken typ utav applikation man använder sig utav så finns det två olika soreters ELB i AWS.

### **Application Load Balancer**

**http och HTTPS**

### **Network Load Balancer**

**Optimerad för att skicka data över ett nätverk. Används ofta när man hanterar HTML, alltså när man skickar en webbsida till användaren**

## VPC, Virtual Private Cloud

En **VPC** är en privat instans/server som du kan skapa i det publika molnet. Säg att du har en publik tjänst och du får data från det som du vill spara någonstans där bara du och dit företag kan få tag på det. Då kan ett VPC vara användbart. Man kan säga att man vanligtvis lägger backend i VPC. Dock inte alltid.

VPC:er kan också vara bra att använda för att spara pengar och för att organisera företaget. Om man behöver flera olika instanser/servrar för olika delar utav företaget. Då går fortfarande all trafik till samma fysiska server och det blir då mycket enklare att hålla koll på.

## Skapa en ”kontaktväg” till VPC.

För att kunna komma åt din VPC som är placerad i molnet så måste man skapa en skyddad väg som inte är öppen för någon annan.

Detta kallas en ”**virtual private gateway, VPG”.**

Man kan också göra en så kallad **Direct connet/Express route** vilket är att låta de som skapade molnet också hantera kontakten du får med molnet. Detta är vanligt.

Ett vanligt namn på vägen in i din VPC är **internet gateway**.

## Meddelande köer, PUBSUB (publish and subscribe)

När man har en applikation som behöver kommunicera med olika databaser och liknande så är det inte särskilt bra att användaren ska behöva vänta på att alla dessa meddelanden. Därför så har man ett system som låter dessa meddelanden läggas i en sorts kö där de hanteras medan användaren kan fortsätta knappa på.

Man kan efterlikna det med Promises och Async await i Java script.

Vi skickar en fråga och fortsätter med det vi gör samtidigt som vi håller koll på vilket svar vi får tillbaka och när vi får det svaret så gör vi något.

## Arbeta i Ditt Moln

Det finns 3 olika sätt att arbeta i molnet.

**Webbaserat, T.ex. AWS Management Console. Vissa har även mobil appar som kanske främst lämpar sig för övervakning.**

**CLI, Command Line Interface. Här kan det vara värt att känna till olika kostnader det kan medföra, t.ex. kan det i MS Azure krävas en diskyta för detta.**

**SDK, Software Development Kits. Det finns stöd för många olika språk så som C++, Java, .NET, Python med flera.**

### Boto 3

Boto 3 är en **SDK (software development kit)** och man kan se det som en verktygslåda som hjälper oss att göra massor olika saker med AWS servrar. För att kunna använda det så måste vi först installera det.

## DNS

En IP-adress är ett ”telefonnummer” som är unikt för varje dator och server. När man skriver en hemsida i en webbläsare så är det första den gör att kolla vilken IP-adress som motsvarar den hemsidans namn (dess domännamn).

**DNS står för: Domain Name System.**

När man försöker nå en hemsida så ställer man en fråga till sin interna DNS server, din router, som då skickar en fråga till en så kallad **rootserver**. Dessa är de servrarna som håller uppslag på vart de ska referera till beroende på vilken ändelse du skrev in (.se.com etc).

Den skickar ett svar: ”nej jag vet inte vart den hemsida du vill åt är men kolla denna server”. Om du skrev ”svt.se” så skickar den din förfrågan till servern som håller koll på alla IP-adresser med ändelsen ”.se”. Dessa servrar kallas **top-level domain server**.

Denna server skickar svarar med en referens till den hemsida du vill åt (svt.se) och dit modem gör en sista förfrågan till den som sedan svarar med den IP-adressen du vill åt.

Två viktiga huvudfunktioner som DNS är uppbyggt efter:

**Authorative DNS**

Den del som skickar svar på förfrågningar

**Recursive DNS**

Skickar frågor/förfrågningar för DNS klienten (servern/routern).

## Bind

Bind är ett verktyg som används för att publicera din DNS på internet. Man kan se det som ett operativsystem som sköter dina DNS förfrågningar och svar. Alltså den mjukvaran som används för att kunna prata med andra servrar.

När du skapar en EC2 server så är den helt tom. Man kan se det som ett hus med enbart vägar. Vi har en plats att nyttja men inga goda sätt att göra det på. Om vi installerar BIND så får vi dörrar och fönster så att vi kan släpa in andra i vårt hus samt signalera till andra hus att de ska skicka något till oss.

BIND står för ”Berkeley Internet Name Domain”.

## DIG

DIG är ett verktyg som kommer med BIND om du laddar ned det och det används för att ställa frågor (göra en förfrågan/request) till olika DNS servrar och visa samt analysera vilket svar du får tillbaka.

Om du skickar en förfrågan efter svt.se med DIG så kommer den att visa:

hur din förfrågan såg ut (vilket DNS record du skickade och den data som du skickade med)

vilket svar du får tillbaka (formaterat efter det DNS record du använde)

plus lite övrigdata som namn på servern, hur lång tid förfrågan och svaret tog och när den gjordes etc.

## DNS records

DNS records är de olika förfrågningar/requests som man kan skicka till en DNS server. Alltså de olika sorters frågor man kan ställa. Man kan se dem som standardiserade modeller att skicka förfrågningar till servrar för att få ett vist svar.

Ett exempel vore den vanligaste sortens DNS record förfrågning (den vanligaste sortens fråga du skickar till en DNS server) som man gör. Detta är en **A** DNS record.

Detta är när du använder ett domännamn, tex ”svt.se”, för att hitta en IP-adress. Du skriver svt.se och din router formaterar detta enligt **A.**

När servern du försöker kommunicera med då får din sökning/förfrågan så ser den att det är ett A DNS record och vet då att den ska först leta upp mappen med svar på A record frågor. Sedan svarar den med den IP-adress som motsvarar det domännamnet som du angav.

Några vanliga DNS records

### A (host, Adress mapping record)

Ger IP-adress för det domännamn du sökte efter. IP-adressen du får är i IPV4

### AAAA (host, quad A record)

Samma som A. Ger IP-adress för det domännamn du sökte. IP-adressen du får är i IPV 6.

### CNAME (alias)

Ger alias på en existerande server. Dvs att det är ett sätt att ge andra namn på en server. Kan också användas för att ge en server flera namn. Säg att du söker efter svt.se men du skriver istället Sverigestelevision.se. Om de som satte upp DNS servern använde sig utav CNAME så pekar båda domännamnen (svt.se och Sverigestelevision.se) på samma server.

### MX (mail exchanger)

Anger mailservrar för ett domännamn om de finns. Om du söker svt.se med MX DNS record så kommer du få tillbaka ett svar med alla servrar som kan hantera mail på den domänen.

Mailservrar ges ett prioritetsnummer vilket anger i vilken ordning mail ska skickas, där den servern med lägst nummer är först. När man anger prioritetsnummer så ska man skriva i tio-tal. Alltså 10,20 och 3 etc. Inte 1,2 och 3.

### PTR (pointer)

Anger domännamn för en IP-adress. Om du gör en sökning med en IP-adress så får du alltså tillbaka det domännamn som står för IP-adressen.

## Linux

Linux är ett alternativt operativsystem man kan använda sig utav. Det är väldigt vanligt att servers använder Linux.

Man behöver inte installera över Windows utan man kan använda sig utav ett program som kallas Windows terminal.

## Säkerhet

Sätt säkerhet på dina filer i lagringsmedia (s3)