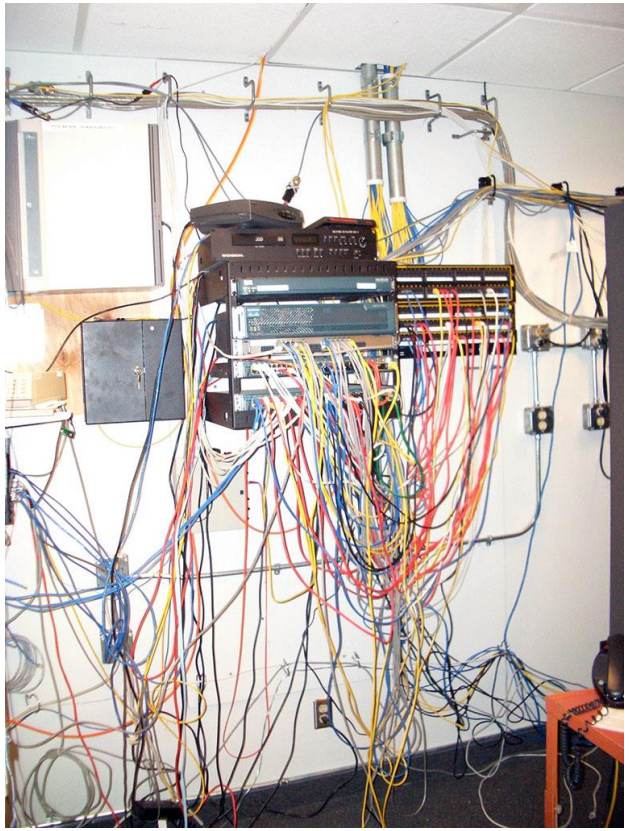
Intro Systemarkitektur

Av: Niklas Hjelm

Arkitektur – Generellt

- En beskrivning eller ritning över en lösning på ett problem eller behov.
- En arkitekt är någon som designar en tänkt lösning för andra så de kan implementera.
- En arkitekt behöver inte ha detaljkunskaper

Systemarkitektur



- Begreppet systemarkitektur innebär en ritning eller karta över flera samverkande lösningar på ett stort komplext problem.
- Innefattar både mjukvara och hårdvara.
- Kräver en hög nivå av kunskap om krav, kunder.
- Många roller inblandade.
  - Utvecklare, administratörer, testare mfl.

Mjukvaruarkitektur

- Mjukvaruarkitektur är en definition av strukturen i en mjukvarulösning eller en applikation.
- Omfattar allt från val av språk till ramverk och interna lösningar.
- Designmönster.
- Kommunikation inom applikationen.
- GUI – Graphical User Interface.

Monolitisk arkitektur

- Det traditionella sättet att designa en mjukvarulösning
- Monolitisk i detta sammanhang innebär ”allt i ett”
  - https://microservices.io/patterns/monolithic.html

- Ett monolitiskt system är ofta svårare att hantera med många utvecklare och många team.
- Under senaste året har denna arkitektur fått nytt liv i form av ”Modulär Monolitisk Arkitektur”

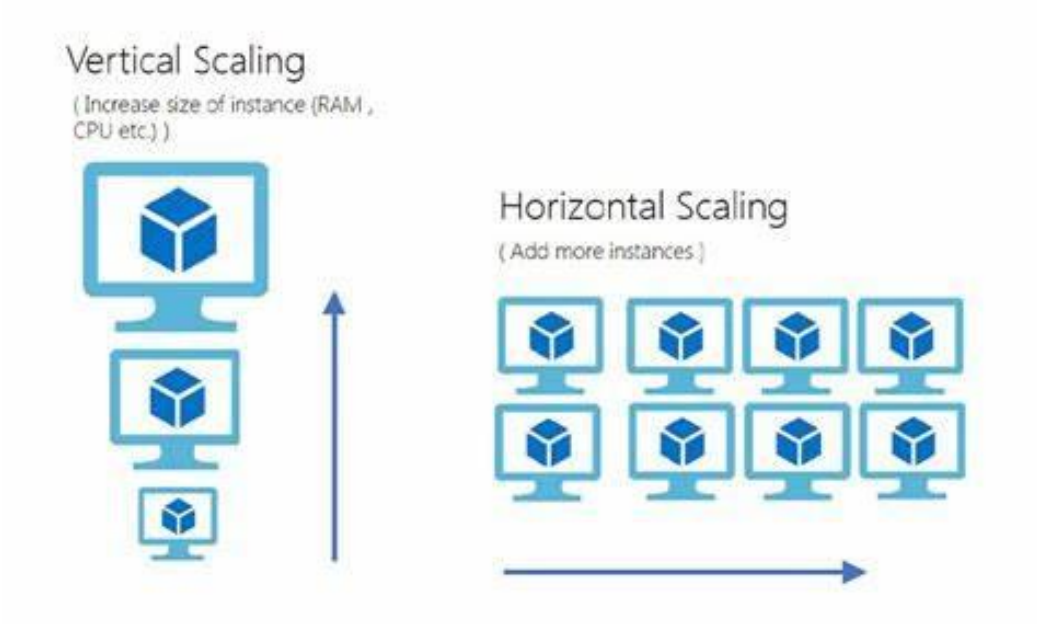
Modulär Monolitisk arkitektur

- Det innebär kod med hög separation och lösa kopplingar.
- SOLID och Designmönster är centralt för att åstadkomma detta.

- Olika former av arkitektur för modulära monoliter:
  - Onion architecture
  - Clean Architecture
  - N-Layer Architechture
  - Med flera.

Applikationsskalning

- Att skala (eng. Scaling/To scale) innebär att öka kapaciteten för systemet
- Vertikal skalning innebär att man ökar/uppgraderar hårdvaran på servern.
- Horisontell skalning innebär att man utökar antalet servrar och kör systemet parallellt



Databasskalning

- Vertikal skalning av databaser innebär som för applikationer att vi utökar de tillgängliga resurserna på hårdvaran.
- Horisontell skalning av databaser är lurigare.
- Ofta används s.k. ”Read Replicas” eller Läsreplikor som speglar informationen i huvuddatabasen men som bara hanterar läs-operationer.
  - Detta ökar prestandan och kan användas för system med många täta läs-operationer.
- Ett annat sätt att åstadkomma horisontell skalning är med s.k. databaskluster där databasen kortfattat delas upp på flera servrar.

Databasreplikor

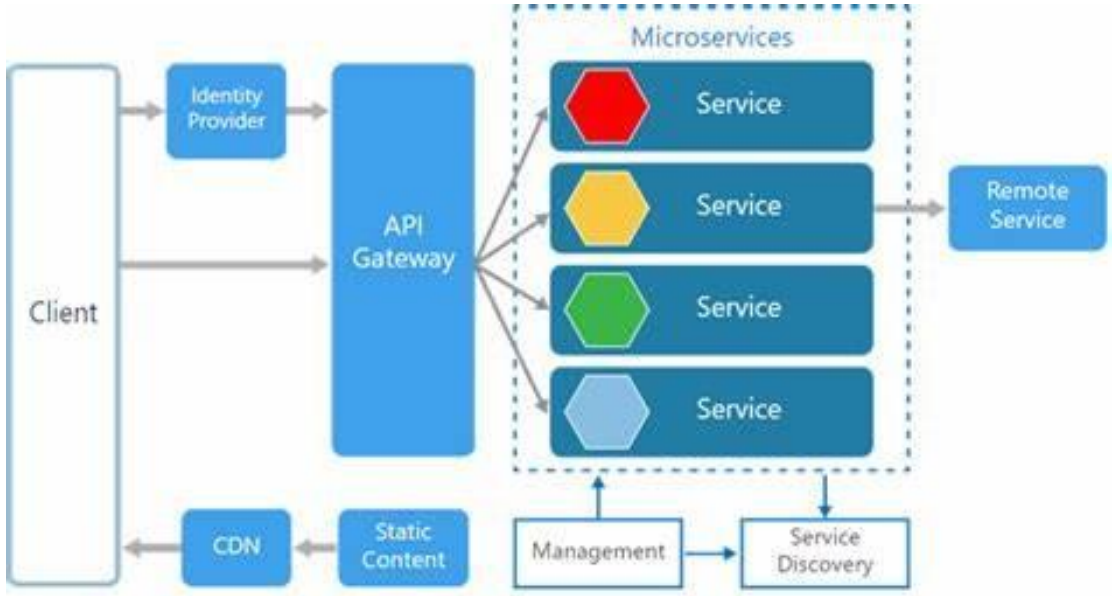
- Databasreplikor kan distribueras på olika fysiska platser. Detta används bland annat för att placera servrar närmare användaren i stora globala system.
- Läsoperationer kan ske snabbare och dessa utgör oftast den absoluta majoriteten av databasoperationer i ett system.
- Ett distribuerat system är också mindre känsligt för fel. Om en del inte svarar kan en annan del ta över.

Distribuerade Applikationer

- Detta är en typ av arkitektur som har använts sedan 60-talet. Det innebär att olika delar av en applikation körs på olika maskiner.
- Exempel på sådana system är äldre universitetsdatorer där alla krävande operationer skedde på en centraldator och hårdvaran hos användaren kunde vara mycket mer lättviktig.
- Idag används detta hela tiden med hjälp av internet och molnet.

Mikrotjänster

- Detta är en förlängning av distribuerade system som låter oss skala varje del av ett system individuellt och oberoende av de andra delarna.
  - https://microservices.io/patterns/microservices.html
- Oftast används WebAPIer med kommunikation över http



Fördelar

- Låter oss arbeta med och leverera olika delar av ett system oberoende av de andra.
- Möjliggör Continous Delivery av varje enskild del.
- Lägre komplexitet i varje enskild del
- Fel är mer isolerade och påverkar en mindre del av systemet
- Låter oss uppdatera teknikstacken enklare vid behov

Nackdelar

- Systemet som helhet blir mycket mer komplext
  - Kommunikation och kontrakt mellan tjänsterna måste hanteras
  - Hantera anrop som kräver interaktion mellan flera tjänster blir komplext
- Deployment blir mer komplext då tjänsterna ska ha åtkomst till varandra och sina beroenden
- Använder mer minne och resurser då varje tjänst behöver .NET-runtime (eller JVM, motsvarande för Java osv.)

När ska vi använda mikrotjänster?

- När ett system blir stort och komplext med flera tydligt separata ansvarsområden
- När flera olika klientapplikationer behöver olika delar av systemet
- När det av säkerhetsskäl är lämpligt att separera delar av en applikation