Operativ Systemer 10

Lavet af: Vivek Misra

Processer

For, at kunne holde styr på de forskellige former af Operativ Systemer.

Så henviser vi venligst tilbage til Powerpoint 1.

System Arkitektur

- System arkitektur fortæller, hvordan systemet er egentligt opbygget ift. Samling.
- Når vi snakker om System Arkitektur, så plejer at vi benævne 3 begreber som er følgende:
- **Centraliseret:** Det er der, hvor systemet er sammenlagt hos en Inode. HUSK, at node er en maskine og ikke noget som illustrer informationer som INODE.
- **Distribueret:** Distribueret system er der, hvor den centraliseret enhed er lidt mere opdelt. Det betyder, at vi har Inodes som uddelegere arbejde til sub-inodes.
- **Decentraliseret:** Det er her, hvor alt er opløst og uddelt, sådan at alle noder er forbundet med deres par. Så vi snakker om Node til Node forhold her!

Forskellige System Arkitektur

- **Monolithic Architecture:** Dette form for System Arkitektur er der, hvor alle system komponenter er forbundet med hinanden. Selve denne form for arkitektur indebærer forskellige processer som fungerer uafhængigt af andre, og er inter-forbundet med hinanden, der gør det besværligt for dem at bryde fra.
- **Publish Subscribe Architecture:**
  - **Publishers:** Denne arkitektur er således, at vi har to parter. Den ene er Publishers, som er dem der deler informationen. Her afsendes videoer og informationer, uden at vide hvem der vil se til det.
  - **Subscriber:** Denne del i arkitektur er dem som modtager informationen. De udtrykker deres interesse i særlige form af information, som er tilgængeligt for dem.
- **Microservice Architecture:** Det er der, hvor en hel system er delt oppe i små bidder som er kaldt Microservices. Disse microservices har deres eget arbejdsområde, og som sammen med andre Microservices udgør et helt system.
- **Client-Server Architecture:** Denne arkitektur kender vi fra hverdagen, hvor det kan ses at vi har en afsender (Client) som laver en request og så har vi en modtager (Server) som udfører requestet.

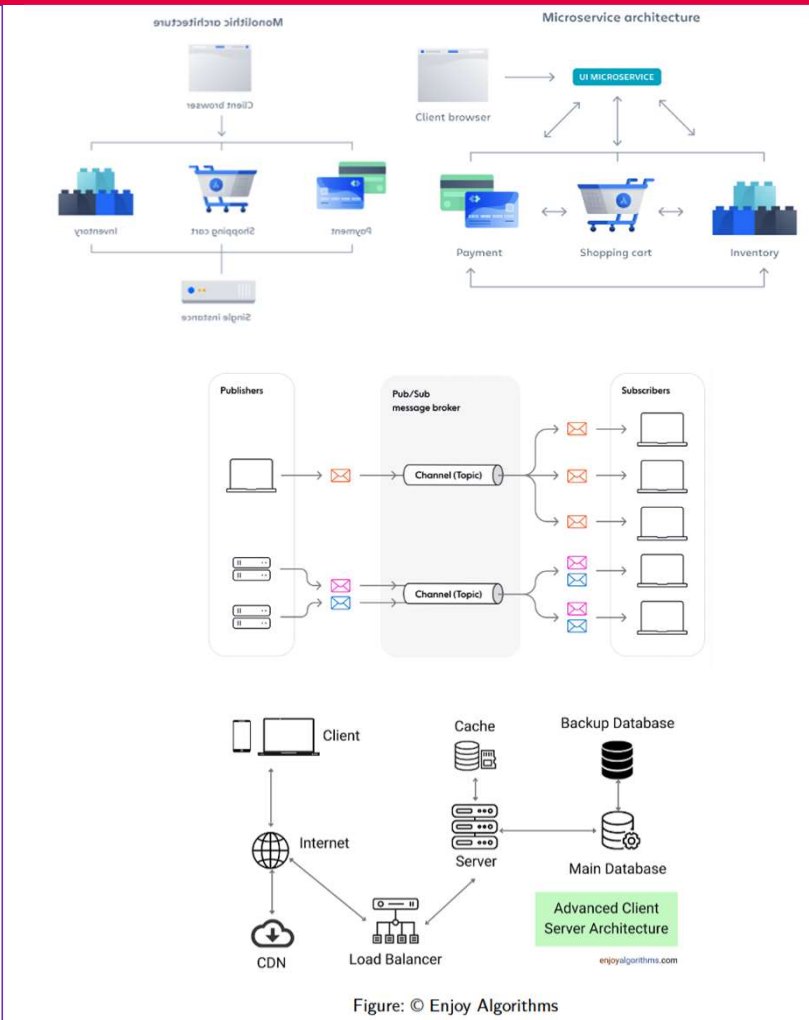


Figure: © Enjoy Algorithms

12-Factor Applikatonen

- 12-Faktor Applikationen er en sæt af principper eller guide som er designet til at kunne bygge software applikationer der er skalerbar, håndterbar og nemt at deployere i moderne cloud-miljøer. Disse principper var egentligt udviklet af Ingeniører ved Heroku, en cloud platform som en Service (PaaS) provider, hvor de sigtede i at lave en robust og justerbare applikationer.

1. Codebase
   - Each application in its own VC repo
2. Dependencies
   - Explicitly declare dependencies
3. Config
   - Store configs in the environment
4. Backing services
   - Treat as attached resources
5. Build, release, run
   - Separate stages
6. Processes
   - One or more. But, stateless
7. Port binding
   - Export through ports
8. Concurrency
   - Utilise concurrency to scale
9. Disposability
   - Fast start-up, graceful shutdown
10. Dev/prod parity
    - Keep environments similar
11. Logs
    - Directly to `stout`
12. Admin processes
    - Only as one-offs

XaaS

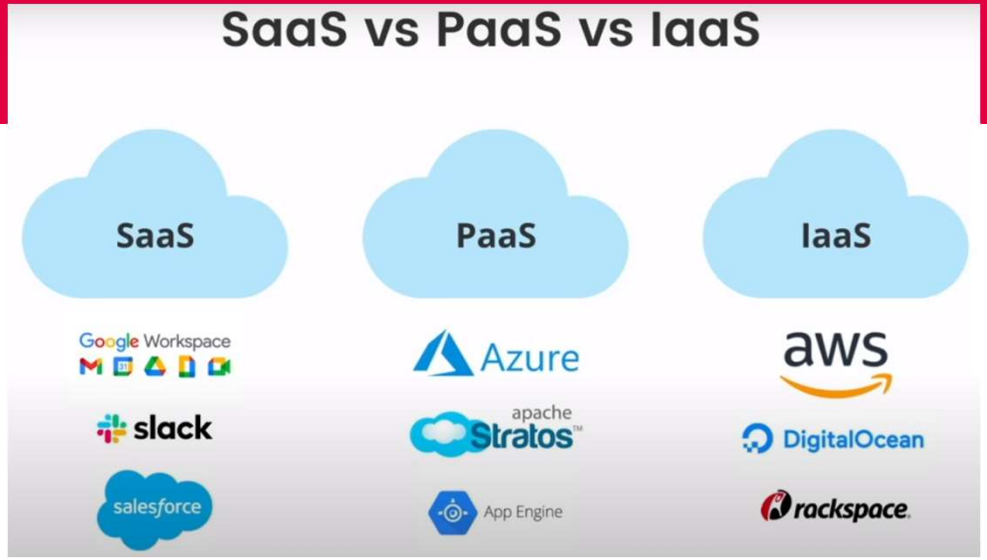
Nu skal vi til at se på, hvad XaaS er.

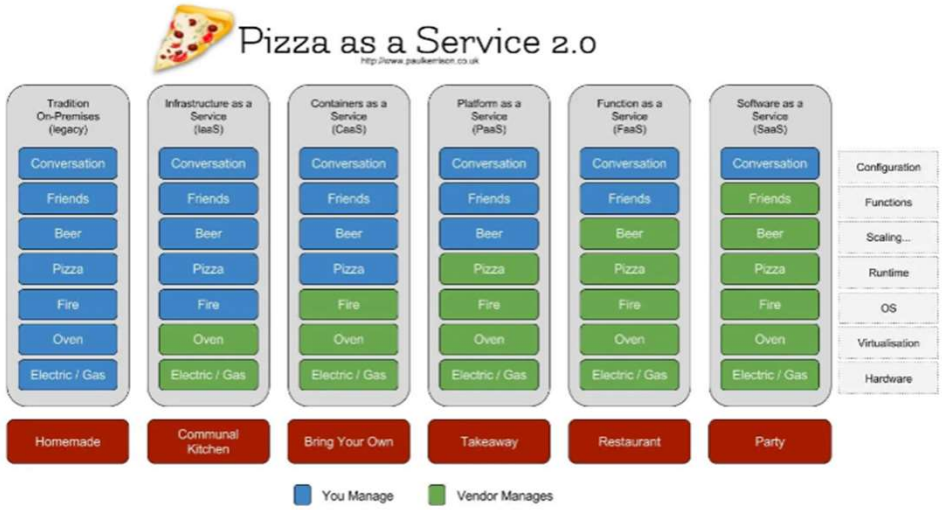
XaaS

- XaaS involverer forskellige brugere blandt en stor bredde af service.
- Disse service kan eksempelvis være:
  - Software
  - Infrastruktur
  - Storage
- Når man kigger fra en ”service-synsvinkel”. Så kan det ses, at Applikationer og Service er leveret gennem skyen.
- Vi kan se, at Subscription er baseret efter kundens abonnement eller produkter som er på gående basis.
- Man er i stand til at skalere applikationer, men samtidig er man også i stand til at ofre services på Software, Platform og Infrastruktur.
  - Det viser vi på næste side.

XaaS

- SaaS: Det er Software som Service.
  - Her kan det ses, at alt fra applikationer, netværker og virtualisering er styret af provideren.
- PaaS: Det er Platform som Service.
  - Her kan det ses, at alt fra runtime til netværker er styret af Provideren. Men dataet er styret af klienten.
- IaaS: Det er Infrastruktur som Service.
  - Her kan det ses, at alt fra Virtualiseringen og til serveren er styret af Provideren





RAFT

Vi kommer her til at introducere til Concensus Algoritmer, hvor RAFT kommer til at forklare om hvordan det stemmer med vores hverdag i en demokratisk samfund!

Concensus Algoritme

- Når vi snakker om en Consensus Algoritme, så består det af følgende egenskaber:
  - Agreement: Det er der, hvor alle korrekte processer skal være enig i en fælles værdi.
  - Validity: Værdien som alle er enige i, skal være angivet af en anden process.
  - Termination: En værdi vil være afgjort af enhver korrekt process.
- Det er også vigtigt, at understrege at en af egenskaberne ved Concensus er CAP-Teorien.
  - Som vi allerede kender på forhånd, så er det bestående Consistency, Availability og Partition Tolerance. Vi kender at det er kun (to) ud af de 3 værdier som kan drive en system.
- Der er eksempler på Concensus Algoritmer, som er vist forneden.
  - Paxos
  - Chandra-Toueg
  - Raft
- Vi fokusere kun på Raft!

Introduktion til RAFT

- RAFT som også er kendt som Replicated and Fault Tolerant.
- Når vi snakker om valgperiode, så plejer vi at referer til en tidsperiode.
  - Men når vi snakker om RAFT, så plejer vi at henvende os til ”terms” og ikke tid.
  - Det plejer, at vores nodes (maskiner) som har følgende roller, når det kommer til RAFTS.
    - Follower
    - Candidate
    - Leader
  - Alle nodes i klusteren har potentialet til at være leder.
  - Udvalget af lederskab foregår ved, at en node ”times out”, hvor den så laver en request til de forskellige nodes, der skal vote kandidat-noden baseret på samme interesse.
  - Når flertallet er dannet, er den enkelte node blevet til leder i nogle sekunders kort tid.
    - Sammen med det her, så sender den enkelte Node en ”heartbeat” ud til de forskellige noder, for at indikere levetiden på den enkelte node.

Introduktion til RAFT

- Når den enkelte node dør, så ender det med at en node udvælger sig selv som kandidat.
  - Den gamle leder vil ikke modtage heart.
  - Men det gør, at de forskellige noder vil stemme på kandidaten som deres nye leder gennem kandidatens request.
- Det skal altid bemærkes, at lederen er den som kan modtage data fra Klienten gennem ”append” af beskeder.
  - Når beskederne er modtaget hos lederne, bliver de ”committed” til de andre noder.
  - Når de forskellige Noder sender service tilbage, så betragtes det hos lederen at ”commit” er taget sted.
  - Man kan også sige, at man laver replicate log fra lederen og til noderne. Og bagefter så har man en node som sender en written log tilbage som en bekræftelse.

Certificate

Her kommer vi til at snakke om Godkendelsesprincipperne.

Introduktion til Certificate

- Som vi kommer til at benævne i Powerpoint 11, så går Certificate ud på at man under en SSL-Forbindelse kan se ID på websiden som man besøger hos således, at man ikke bliver snydt.
- Dette er med til at bekæmpe fremmede invasioner og hack på ens computer, og sikre at der er sikker dataoverførsel i form af enkryption.
- Certificate i SSL følger af X.509 standard, hvor mange protokoller bruges til autentikation.
- En Certificate indeholder følgende:
  - Bekræfter, at CA underskrev det.
  - Sørger for integritet af indhold.
  - Certificate for websiden.
  - Valideringsperiode.
  - Subject Navn
  - Certificate Signatur.
- Billedet til højre viser et eksempel på Certificate.

```
1    Certificate:
2 1    Certificate:
3 2    Data:
4 3        Version: 3 (0x2)
5 4        Signature Algorithm: sha256WithRSAEncryption
6 5        Issuer: C = US, O = Let's Encrypt, CN = R3
7 6        Validity
8 7            Not Before: Nov 10 06:04:40 2023 GMT
9 8            Not After : Feb  8 06:04:39 2024 GMT
10 9       Subject: CN = ahmadhamid.dk
11 10      X509v3 extensions:
12 11          X509v3 Subject Alternative Name:
13 12              DNS:*.ahmadhamid.dk, DNS:ahmadhamid.dk
   13  ...
```

Dannelse af Certificate

- Nu kommer spørgsmålet om, hvordan en certificate egentligt dannes?
- STEP 1: Det første som sker er, at afsenderen som danner websiden laver en enkrypteret nøgle.
- STEP 2: Den anden er, at afsenderen fylder en CSR aka. Certificate Signing Request, som teknisk set er en dokumentation eller formular af en Certificate Ansøgning.
- STEP 3: Når CSR'en er opfyldt, så er formularen sendt til CA aka. Certificate Authority.
- STEP 4: Certificate Authority bekræfter integritet af klienten.
- STEP 5: Når dette integritet er bekræftet, så er en certificate genereret med alt signatur og dermed givet til afsenderen af websiden.

Dannelse af Certificate Signing Request

- Nu spørger vi, hvordan en CSR egentligt er dannet?
- Som vi allerede kender på forhånd, så er CSR også kendt som Certificate Signing Request som teknisk set er en formular, der beskriver hvem der er afsenderen af websiden og at de ansøger for en certificate.
- Når denne certifikat ansøgning kommer hos CA, som også er kendt som Certificate Authority, så tjekker de oplysningerne igennem og derved uddelegerer en certifikat bevis der bekræfter at websiden er lovlig og troværdig.

Dannelse af Certificate Authority

- Nu spørger vi, hvad er Certificate Authority.
- Som vi allerede kender på forhånd, så er Certificate Authority en tredjepart eller ”autoritet” som underskriver eller godkender afsenderens ansøgning.
- CA har ansvaret for at godkende afsenderens oplysninger og generere en certifikat på baggrund af det, for afsenderens webside.
- Selve certifikatet udstedet af CA indeholder følgende:
  - Sørger for, at du ejer web-adressen.
  - Sørger for, at du ejer din ”handel” på siden.
  - Sørger for, at identificere dig selv ift. Identifikationstyveri.

Self-Signed Certificate

- Self-Signed Certificate er der, hvor man selv genskaber sin eget diplom for sit eget websiden.
- Her er der ingen autoritet som udsteder et bevis for websiden, men det er omvendt afsenderen som selv genskaber deres eget certifikat.
- Ulempen her er, at certifikatet udstedet af afsenderen er utroværdig grundet ”bias”, da det er ikke en overbevisende tredjepart som godkender det.
- Dog skal det gøres obs på, at en self-signed certifikat etablerer en enkrypteret kommunikation uden at etablere troværdigheden til serveren.

SLUT 10

Lavet af: Vivek Misra