Introduksjon til serverless

Henrik Wingerei, Fredrik V. Mørken

Agenda

Del 1

- Introduksjon til serverless
- Hands-on
- Oppsummering

Del 2

- Introduksjon til Serverless Framework
- Hands-on
- Oppsummering

Del 3

- Bonusoppgaver
- Oppsummering og avslutning

Del 1: Introduksjon og case

Why The Future Of Software And Apps Is Serverless

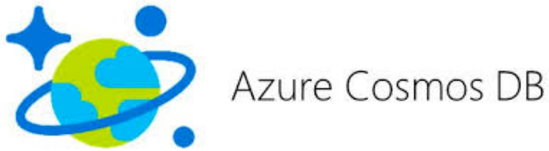
The phrase “serverless” doesn’t mean servers are no longer involved. It simply means that developers no longer have to think that much about them.

Ken Fromm, 2012

Serverless

- Tredjepart (skyleverandør) håndterer oppsett, provisjonering og forvaltning av servere
- All funksjonalitet kjøres på “managed services”, høynivåtjenester der underliggende infrastruktur er abstrahert bort

Managed services



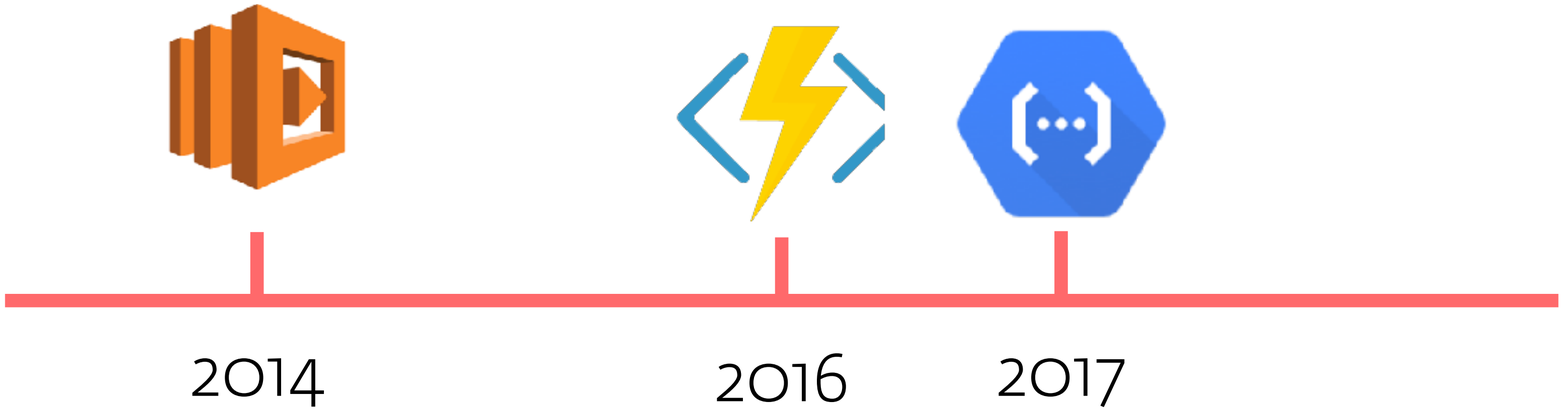
Amazon Kinesis

Serverless

- Kan bruke enkelttjenester, eller sy de sammen til en fullstendig applikasjon
- Har i lang tid hatt managed services for database, serving av statiske filer, CDN, etc, men backend har typisk fortsatt kjørt på VM-er eller PaaS-løsninger

Backend as a Service

Function as a Service



Function as a service

- AWS Lambda og andre FaaS-varianter muliggjør nå full serverless computing
- Backendkode skrives som rene funksjoner
  - Input: Et event i skyplattformen
    - Eks.: HTTP-request, ny fil lastet opp i S3, endring i databasetabell
  - Output: Eks.: HTTP-response

Function as a service

- Funksjonens runtime starter og stopper automatisk for hver request
- Ingen egen provisjonering og drift av servere
- Faktureres per kjøring av funksjonen

Hvorfor serverless?

- Ingen håndtering av infrastruktur
- Automatisk skalering
- Individuell skalering av funksjoner
  - Billing by function
- Reduserte kostnader
  - Betal kun for bruk
  - Reduserte drifts- og utviklingskostnader

Case

Demo

Lambda

Lambda

DynamoDB

Lambda

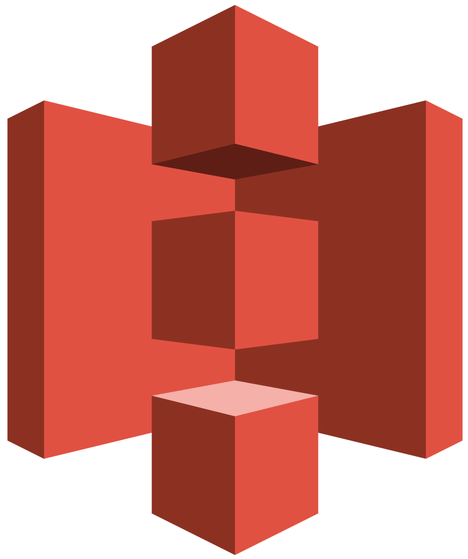
DynamoDB

API Gateway

API Gateway

Lambda

DynamoDB

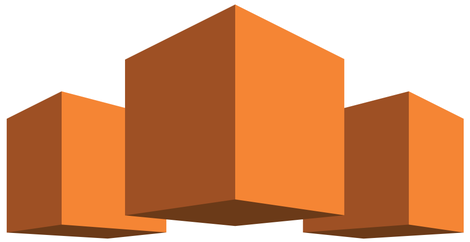


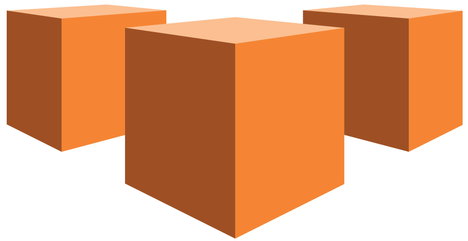
S3

API Gateway

Lambda

DynamoDB





Cloudfront

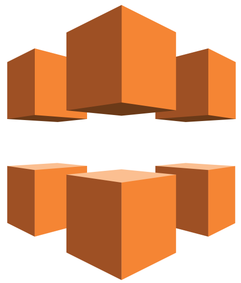
S3

API Gateway

Lambda

DynamoDB

Klient



Cloudfront

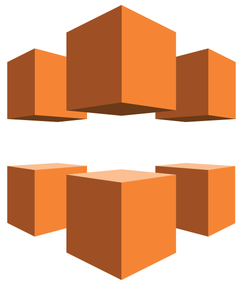
S3

API Gateway

Lambda

DynamoDB

Klient



Cloudfront

S3

Innlogging AWS

https://henriwi.gitbooks.io/serverless-workshop/

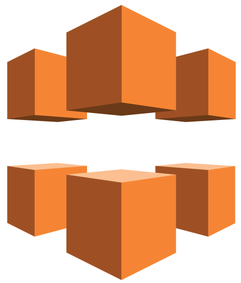
https://github.com/henriwi/serverless-workshop

API Gateway

Lambda

DynamoDB

Klient



Cloudfront

S3

Utfordringer fra case

- Manuelt oppsett
- Redigert kode i nettleseren
- AWS webkonsollet er lite brukervennlig
- Kan ikke teste lokalt

Del 2: Automatisert oppsett

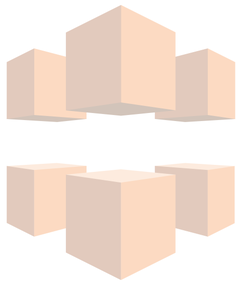
Serverless framework

- Node.js CLI verktøy
- Et verktøy for å utvikle, teste og deploye serverless-applikasjoner
- Støtter flere skyleverandører

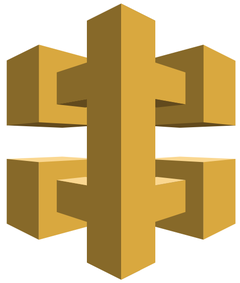
Serverless framework

- Konfigurerer applikasjonen i yaml
- Serverless Framework (SF) bruker CloudFormation til å provisjonere opp alle tjenestene som applikasjonen din bruker

Klient



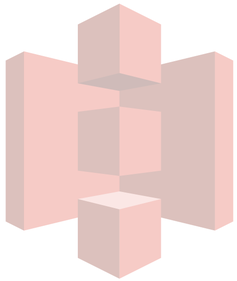
Cloudfront



API Gateway

Lambda

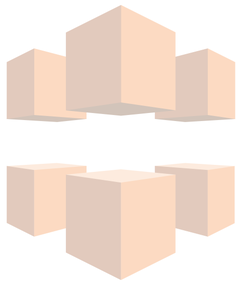
DynamoDB



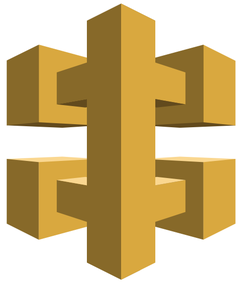
S3

Service

Klient



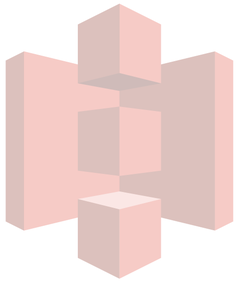
Cloudfront



API Gateway

Lambda

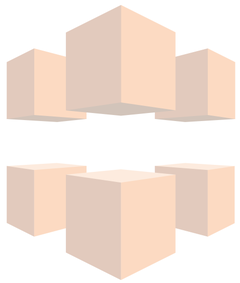
DynamoDB



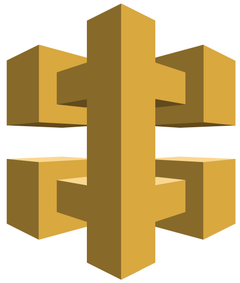
S3

Function

Klient



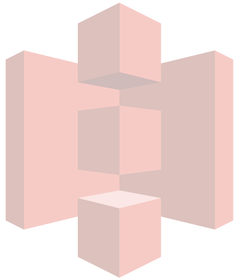
Cloudfront



API Gateway

Lambda

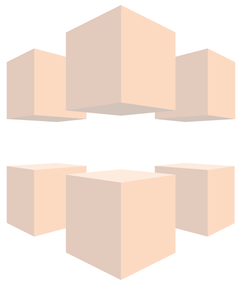
DynamoDB



S3

Events

Klient

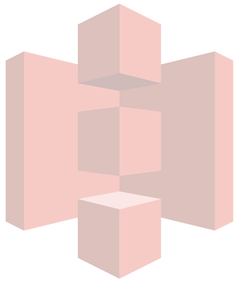


Cloudfront

API Gateway

Lambda

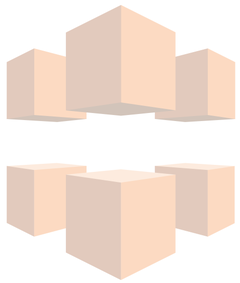
DynamoDB



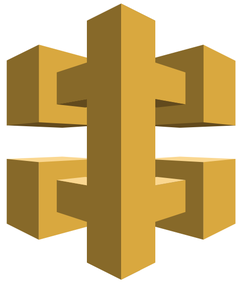
S3

Resources

Klient

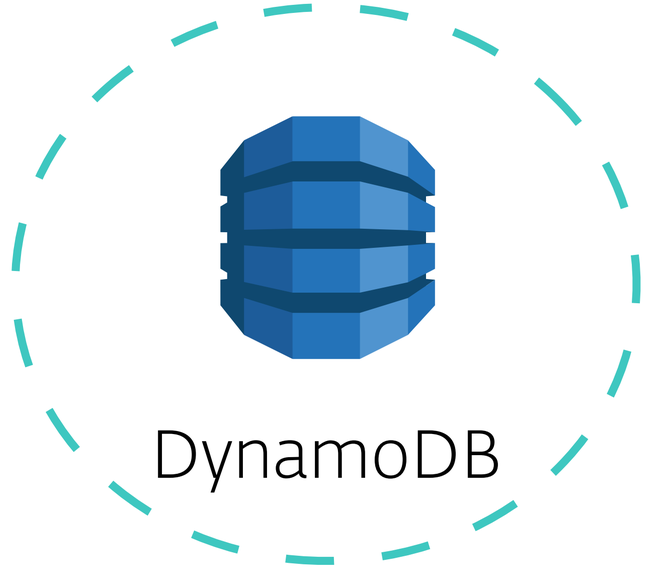


Cloudfront

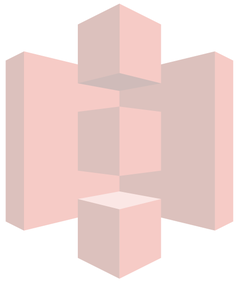


API Gateway

Lambda



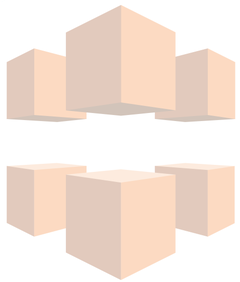
DynamoDB



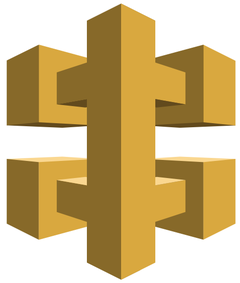
S3

Resources

Klient



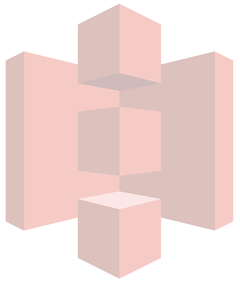
Cloudfront



API Gateway

Lambda

DynamoDB



S3

Serverless framework

- Deployer hele servicen inkl. funksjoner og alle ressurser
- Kan deploye kun lambda-funksjonene og invokere disse
- Mulighet for å invokere lambda-funksjonene lokalt
- Støtte for å lese logger, hente ut metrikker ++

Case – Serverless framework

Oppsummering
Serverless framework

Del 3: Bonusoppgaver

Avslutning

Alternativer for automatisert oppsett

- AWS CLI
- SAM (AWS Serverless Application Model)
- Terraform
- Apex (http://apex.run/)
- Zappa, for Python på AWS (https://www.zappa.io/)
- Claudia.js (https://claudiajs.com/)

Fordeler med serverless og FaaS

- Ingen håndtering av infrastruktur
- Automatisk skalering
- Individuell skalering av funksjoner
  - Billing by function
- Reduserte kostnader
  - Betal kun for bruk
  - Reduserte drifts- og utviklingskostnader

Utfordringer

Nytt computing-paradigme

- Kompetanse, verktøy, dokumentasjon
- Test, deploy, konfigurasjon, miljøer, versjonering

Utfordringer

Vendor-spesifikk kunnskap
(gjelder alt av cloud, men spesielt her)

Utfordringer

Vanskelig å sammenligne pris vs. tradisjonell “serverfull” computing

Utfordringer

Black box – begrenset innsikt i hvordan plattformen fungerer

- Liten mulighet for konfigurasjon og tuning av runtime

Utfordringer

Utforende å feilsøke

Utfordringer

Lang oppstartstid på “kalde” lambdaer

Spådom

Serverless blir den dominerende modellen i fremtiden

Spådom

Bør vurdere å ta i bruk på områder der det passer spesielt godt

- Batcher
- Beregninger
- Funksjonalitet som brukes periodisk eller sjeldent
- Støttefunksjonalitet (e-post-utsending, logging)
- Hendelsesdrevne systemer (f.eks. med Kinesis)

Spådom

På tradisjonelle webapper (CRUD) går kanskje ikke kost-nytte-regnskapet opp helt ennå

- Større kostnader pga kompetanse, plattformenes modenhet, etc
- Mindre nytteverdi for “gjennomsnittlige” apper med jevn trafikk, mindre behov for individuell skalering, etc.

Takk for deltakelsen, og takk for oss!

Henrik Wingerei

Fredrik V. Mørken