Indhold

[Modul 1 2](#_Toc168073971)

[Modul 2 2](#_Toc168073972)

[Modul 3 2](#_Toc168073973)

[Modul 4 3](#_Toc168073974)

[Modul 5 3](#_Toc168073975)

[Modul 6 3](#_Toc168073976)

[Modul 7 4](#_Toc168073977)

[Modul 8 4](#_Toc168073978)

[Modul 9 4](#_Toc168073979)

[Modul 1: Hvad er cloud computing, fordele og ulemper 5](#_Toc168073980)

[Modul 2: Virtuelle maskiner og storage 7](#_Toc168073981)

[Modul 3: Virtuelle netværk og netværkssikkerhed 8](#_Toc168073982)

[Modul 4: Azure Web Apps og Load Balancing 10](#_Toc168073983)

[Modul 5: Cloud Databaser 12](#_Toc168073984)

[Modul 6: Containers 14](#_Toc168073985)

[Modul 7: Serverless Computing 17](#_Toc168073986)

[Modul 8: Messaging og Sikring af Information 19](#_Toc168073987)

[Modul 9: Tilgængelighed og Skalering af Applikationer 21](#_Toc168073988)

## Modul 1

**Beskrivelse:** Cloud computing indebærer levering af computing-tjenester over internettet. De primære fordele inkluderer skalerbarhed, fleksibilitet, og omkostningsbesparelser. Besparelser kommer fra at der ikke længere skal holdes hardware on-premise

Mens ulemperne kan være sikkerhedsproblemer og afhængighed af internetforbindelse. Cloud services kategoriseres typisk i IaaS, PaaS, og SaaS. NIST's essentielle karakteristika for cloud computing inkluderer on-demand self-service, bred netværksadgang, ressourcepuljering, hurtig elasticitet, og målt service.

**Relaterede emner:**

1. On-premise vs. cloud computing
2. IaaS, PaaS, SaaS modeller
3. NIST definitioner og karakteristika
4. Omkostningsbesparelser ved cloud computing
5. Sikkerhedsudfordringer i cloud computing

## Modul 2

**Beskrivelse:** Virtuelle maskiner (VM'er) og storage er grundlæggende komponenter i cloud computing. Oprettelse af en VM med en webapp på Azure kræver overvejelser om ressourcetildeling, ydeevne, og skalerbarhed. Azure storage tilbyder forskellige typer lagringsløsninger såsom blob, file, queue og table storage.

**Relaterede emner:**

1. Azure VM oprettelse og konfiguration
2. Typer af Azure storage (blob, file, queue, table)
3. Webapp deployment på en VM
4. Designovervejelser for VM'er
5. Omkostningsoptimering af storage

## Modul 3

**Beskrivelse:** Virtuelle netværk i Azure inkluderer tildeling af IP-adresser, private netværk, og netværkssikkerhedsgrupper. Azure Resource Manager (ARM) muliggør administration af ressourcer via templates. Budgettering og omkostningsberegning er også en del af netværksdesign.

**Relaterede emner:**

1. IP-adressering og subnets
2. Azure networking basics
3. Netværkssikkerhedsgrupper (NSG)
4. SSH med nøglepar
5. ARM-templates og budgetstyring

## Modul 4

**Beskrivelse:** Azure Web Apps tilbyder en platform til at hoste webapplikationer med load balancing for at forbedre tilgængeligheden. Konfiguration og integration med storage er vigtige aspekter. Sammenligning af Azure App Service med VM'er for webapplikationer indebærer vurdering af fordele og ulemper.

**Relaterede emner:**

1. Azure Web Apps konfiguration
2. Load balancing teknikker
3. Webapp integration med Azure storage
4. Fordele/ulemper ved App Service vs. VM
5. Skalerbarhed og ydeevne

## Modul 5

**Beskrivelse:** Cloud-databaser som Azure Cosmos DB tilbyder høj tilgængelighed og skalerbarhed. Designovervejelser for Cosmos DB inkluderer datafordeling, partitionering, og omkostningsstyring. Service Level Agreement (SLA) og Request Units (RUs) er essentielle koncepter.

**Relaterede emner:**

1. Cosmos DB arkitektur og funktioner
2. .NET integration med Cosmos DB
3. SLA og RUs i Cosmos DB
4. Designovervejelser for databaseoprettelse
5. Opsætning og administration af Cosmos DB

## Modul 6

**Beskrivelse:** Containers tilbyder letvægts virtualisering sammenlignet med VM'er. Azure Containers muliggør hurtig deployment og skalerbarhed. Workflowet for containerisering omfatter build, ship og run processer.

**Relaterede emner:**

1. Containers vs. VM'er
2. Fordele/ulemper ved containers
3. Azure Container Instances (ACI)
4. Deployment workflow for containers
5. Container orchestration med Kubernetes

## Modul 7

**Beskrivelse:** Serverless computing eliminerer behovet for infrastrukturadministration. Azure Functions muliggør kørslen af kode på begivenhedsbasis. Fordele inkluderer omkostningsbesparelser og skalerbarhed, mens ulemper kan være kompleksitet i fejlfinding og begrænsninger i runtime.

**Relaterede emner:**

1. Serverless vs. container-baserede løsninger
2. Opbygning af Azure Functions
3. Integration af Azure Functions med Cosmos DB
4. Omkostningsstyring i serverless computing
5. Event-driven architecture

## Modul 8

**Beskrivelse:** Messaging og sikkerhed er centrale aspekter i cloud-løsninger. Azure messaging platforms inkluderer Service Bus og Key Vault for sikker meddelelseshåndtering og nøgleopbevaring. Messaging sikrer asynkron kommunikation mellem tjenester.

**Relaterede emner:**

1. Azure Service Bus
2. Azure Key Vault
3. Messaging-konceptet
4. Sikkerhed i messaging
5. Sammenligning af Azure messaging platforms

## Modul 9

**Beskrivelse:** Tilgængelighed og skalering af applikationer er afgørende for at imødekomme brugerkrav. Load balancing, DTO messaging, og forskellige skaleringsteknikker sikrer optimal performance og pålidelighed.

**Relaterede emner:**

1. Skalering af applikationer
2. Load balancing metoder
3. DTO (Data Transfer Object) koncept
4. Automatisk skalering
5. Performance monitoring og optimering

Disse beskrivelser og punkter giver et overblik over de væsentligste emner i cloud computing kurset, med fokus på Microsoft Azure.

## Modul 1: Hvad er cloud computing, fordele og ulemper

**Beskrivelse:** Cloud computing indebærer levering af computing-tjenester som servere, storage, databaser, netværk, software, analyse og ai over internettet (cloud) for at tilbyde hurtigere innovation, fleksible ressourcer og skalaøkonomier. Man betaler som regel kun for cloud-tjenester man bruger, hvilket hjælper med at reducere driftsomkostninger, drive infrastrukturen mere effektivt og skalere efter behov.

**Fordele:**

* **Skalerbarhed:** Muligheden for at skalere op eller ned baseret på behov uden betydelige investeringer i fysisk infrastruktur.
* **Fleksibilitet:** Adgang til ressourcer og services on-demand, hvilket tillader hurtigere implementering af nye løsninger.
* **Omkostningsbesparelser:** Betalingsmodellen "pay-as-you-go" eliminerer behovet for store kapitalinvesteringer i hardware.
* **Tilgængelighed:** Tjenesterne er tilgængelige fra hvor som helst med en internetforbindelse, hvilket understøtter fjernarbejde og mobilitet.
* **Vedligeholdelse:** Udbyderen håndterer vedligeholdelse og opdateringer, hvilket frigør interne ressourcer til andre opgaver.

**Ulemper:**

* **Sikkerhed:** Potentielle risici ved dataopbevaring uden for virksomhedens egne servere, hvilket kræver tillid til udbyderen.
* **Afhængighed af internetforbindelse:** Adgang til tjenester afhænger af en stabil og hurtig internetforbindelse.
* **Compliance:** Overholdelse af lovgivning og reguleringer kan være mere kompleks, især med hensyn til datalagring og -overførsel på tværs af landegrænser.
* **Omkostningsstyring:** Selv om cloud kan være omkostningseffektiv, kan uforudsete brugsmønstre føre til høje omkostninger.

**Cloud services, on-premise, IaaS, PaaS, SaaS:**

* **On-premise:** Traditionel IT-infrastruktur, hvor hardware og software administreres internt i virksomheden. Fordele inkluderer kontrol og sikkerhed, men det er ofte dyrt og mindre fleksibelt.
* **IaaS (Infrastructure as a Service):** Tilbyder virtuelle maskiner og anden infrastruktur via cloud. Eksempel: **Microsoft Azure VMs**. Det giver kontrol over operativsystemer, netværk og storage, hvilket minder om traditionel IT-infrastruktur.
* **PaaS (Platform as a Service):** Tilbyder en platform, der gør det muligt at udvikle, teste og implementere applikationer uden at håndtere den underliggende infrastruktur. Eksempel: **Azure App Service**. Det sparer tid og ressourcer på infrastrukturadministration.
* **SaaS (Software as a Service):** Tilbyder færdige applikationer, der køres og administreres af tjenesteudbyderen. Eksempel: **Microsoft Office 365.** Brugere tilgår applikationerne via internettet, hvilket gør det let at anvende og administrere.

**NIST essentielle karakteristika for cloud computing:**

* **On-demand self-service:** Brugere kan automatisk tildele computing-ressourcer såsom servere og storage efter behov uden menneskelig interaktion med tjenesteudbyderen.
* **Bred netværksadgang:** Tjenester er tilgængelige over nettet og kan tilgås via standardiserede mekanismer af forskellige klienttyper, såsom mobiltelefoner, tablets, laptops og arbejdsstationer.
* **Ressource pooling:** Computing-ressourcerne udbydes ved hjælp af en multi-tenant model, hvor flere kunder deler en pool af ressourcer. Disse ressourcer tildeles dynamisk efter kundernes efterspørgsel.
* **Hurtig elasticitet:** Ressourcer kan hurtigt og elastisk tilpasses op eller ned i forhold til efterspørgsel. For brugerne virker ressourcerne ubegrænsede og kan tilgås i enhver mængde til enhver tid.
* **Målt service:** Cloud-systemer kontrollerer og optimerer ressourcer automatisk ved at bruge en måling på et abstraktionsniveau, der er passende for den type tjeneste (f.eks. storage, processing, båndbredde). Brug kan monitoreres, kontrolleres og rapporteres, hvilket giver gennemsigtighed for både udbyder og forbruger af den anvendte tjeneste.

**Uddybning af relaterede emner:**

1. **On-premise vs. cloud computing:**
   * **On-premise:** Inkluderer fysisk kontrol og sikkerhed over hardware og data, men kræver stor kapitalinvestering og løbende vedligeholdelse. **Så ved on-premise tænker vi hele tiden økonomi, da alt koster penge og tid**
   * **Cloud computing:** Tilbyder skalerbarhed og fleksibilitet med mindre initial investering, men afhænger af en tredjeparts udbyder og internetforbindelse. **Ved cloud computing, har vi også fokus på økonomi, men hvis vi ved hvad vi laver kan vi starte i en lavere prisklasse og skalere efter behov, dette spare også tid på opsætning af maskiner. Dertil er vi afhægige af oppetid og service-niveau fra udbyder**
2. **IaaS, PaaS, SaaS modeller:**
   * **IaaS:** Tilbyder grundlæggende infrastruktur som virtuelle maskiner, storage, og netværk. Brugeren har kontrol over operativsystemer og applikationer.
   * **PaaS:** Tilbyder en platform til at udvikle, teste og implementere applikationer. Brugeren fokuserer på applikationsudvikling frem for infrastrukturadministration.
   * **SaaS:** Tilbyder færdige applikationer, som brugerne kan tilgå via internettet. Udbyderen håndterer alt vedrørende infrastruktur, middleware, og applikationer.
3. **NIST definitioner og karakteristika:**
   * NIST definerer cloud computing og dets essentielle karakteristika, der gør det muligt at standardisere forståelsen og implementeringen af cloud-tjenester. Disse karakteristika sikrer, at tjenesterne er fleksible, skalerbare og omkostningseffektive.
4. **Omkostningsbesparelser ved cloud computing:**
   * Cloud computing reducerer behovet for store initialinvesteringer i hardware. Med "pay-as-you-go" modellen betaler virksomheder kun for de ressourcer, de faktisk bruger, hvilket reducerer driftsomkostninger.
5. **Sikkerhedsudfordringer i cloud computing:**
   * Sikkerheden i cloud computing indebærer udfordringer som databeskyttelse, overholdelse af lovgivning, og afhængighed af udbyderens sikkerhedsforanstaltninger. Det er vigtigt at implementere stærke sikkerhedsprotokoller og vælge pålidelige udbydere.

## Modul 2: Virtuelle maskiner og storage

**Beskrivelse:** Virtuelle maskiner (VM'er) og storage er grundlæggende komponenter i cloud computing. VM'er muliggør drift af forskellige operativsystemer og applikationer på en enkelt fysisk maskine, hvilket optimerer ressourceudnyttelse. Azure tilbyder en række storage-løsninger, som kan bruges til at gemme data i forskellige formater og til forskellige formål. Designovervejelser ved oprettelse af en VM inkluderer valg af passende ressourcer og konfigurationer for at sikre optimal ydeevne og omkostningseffektivitet.

**Virtuel maskine med en webapp:** En virtuel maskine kan hoste en webapplikation, hvilket indebærer installation af et operativsystem, en webserver (f.eks. IIS, Apache), og den nødvendige software til at køre applikationen. Azure tilbyder VM'er med præinstallerede konfigurationer for at lette denne proces.

**Azure storage:** Azure storage tilbyder forskellige typer af lagring, såsom blob storage (for ustrukturerede data), file storage (for delte filer), queue storage (for messaging mellem komponenter), og table storage (for strukturerede NoSQL data).

**Designovervejelser ved oprettelse af en ny virtuel maskine:** Når man opretter en VM i Azure, skal man overveje flere faktorer, herunder:

* **Størrelse og type af VM:** Bestemmes af applikationens krav til CPU, hukommelse og disk I/O.
* **Beliggenhed:** Vælg den geografiske region for at reducere latens og opfylde lovgivningsmæssige krav.
* **Netværkskonfiguration:** Sikrer korrekt opsætning af IP-adresser, subnets, og netværkssikkerhedsgrupper.
* **Skalerbarhed:** Planlæg for fremtidig vækst og muligheden for at skalere op eller ned baseret på behov.
* **Sikkerhed:** Implementer sikkerhedsforanstaltninger som firewalls, sikker adgang (SSH/RDP), og regelmæssige sikkerhedsopdateringer.

**Uddybning af relaterede emner:**

1. **Azure VM oprettelse og konfiguration:**
   * **Oprettelse:** Brug Azure Portal, Azure CLI eller ARM templates til at oprette en ny VM. Vælg operativsystem, VM-størrelse og andre konfigurationsindstillinger.
   * **Konfiguration:** Installer nødvendige applikationer og tjenester. Konfigurer netværksindstillinger og opsætning af sikkerhedsgrupper for at beskytte VM'en.
2. **Typer af Azure storage (blob, file, queue, table):**
   * **Blob Storage:** Designet til at gemme store mængder ustrukturerede data som billeder, videoer og logfiler.
   * **File Storage:** Giver fuldt administrerede filshares, der kan tilgås via SMB-protokollen.
   * **Queue Storage:** Bruges til messaging og kommunikation mellem forskellige applikationskomponenter.
   * **Table Storage:** Tilbyder NoSQL lagring til strukturerede data og er optimeret til hurtig adgang og skalering.
3. **Webapp deployment på en VM:**
   * **Webserver installation:** Installer en webserver som IIS (Windows) eller Apache (Linux).
   * **Applikationsdeployment:** Upload og konfigurer webapplikationen. Sørg for at inkludere alle nødvendige biblioteker og afhængigheder.
   * **Netværksadgang:** Konfigurer netværksindstillinger for at tillade trafik til webapplikationen. Brug netværkssikkerhedsgrupper for at beskytte mod uønsket adgang.
4. **Designovervejelser for VM'er:**
   * **Ressourceoptimering:** Vælg VM-størrelse baseret på ydeevnekrav og budget. Overvåg ressourcforbrug og juster størrelsen efter behov.
   * **Redundans og failover:** Implementer høj tilgængelighed ved at bruge flere VM'er og konfigurere failover-mekanismer.
   * **Backup og genoprettelse:** Opsæt regelmæssige backup-politikker og genoprettelsesplaner for at beskytte data.
5. **Omkostningsoptimering af storage:**
   * **Valg af lagringstype:** Vælg den mest omkostningseffektive lagringstype baseret på dataadgangsmønstre (hot, cool, eller archive tiers).
   * **Lagerstyring:** Implementer politikker for at arkivere eller slette gamle og ubrugte data.
   * **Overvågning og analyse:** Brug Azure Monitor og Cost Management til at holde styr på storageforbruget og identificere besparelsesmuligheder.

Disse uddybningspunkter giver en dybere forståelse af, hvordan man effektivt bruger og administrerer virtuelle maskiner og storage i Azure, samt hvilke overvejelser man skal have i tankerne for at optimere ydeevne og omkostninger.

## Modul 3: Virtuelle netværk og netværkssikkerhed

**Beskrivelse:** Virtuelle netværk (VNet) i Azure er fundamentet for at bygge en sikker og skalerbar infrastruktur i skyen. Det inkluderer administration af IP-adresser, private netværk, og integration af netværkssikkerhedsgrupper (NSG'er) for at beskytte ressourcer. Azure Resource Manager (ARM) bruges til at administrere og implementere ressourcer ved hjælp af deklarative templates. Budgettering og omkostningsstyring er afgørende for at sikre en økonomisk effektiv drift.

**Virtuelle netværk – IP-adresser, tildelte IP-adresser, private netværk:** VNets giver mulighed for at oprette private netværk i Azure, som kan segmentere og isolere ressourcer. Hver VNet kan opdeles i subnets, og IP-adresser kan tildeles enten statisk eller dynamisk afhængigt af behovet.

**Azure networking basics:** Azure networking inkluderer konfiguration af VNets, subnets, og NSG'er, samt opsætning af peering mellem VNets for at muliggøre kommunikation på tværs af netværk. Det er også vigtigt at forstå brugen af offentlige og private IP-adresser for at sikre korrekt adgang og sikkerhed.

**Virtuelt netværk i en cloud løsning:** Virtuelle netværk i Azure muliggør kommunikation mellem Azure-ressourcer som VM'er, databaseservere, og webapps. VNets kan også forbindes med on-premise netværk via VPN eller Azure ExpressRoute.

**Netværkssikkerhedsgrupper og hvordan de integreres i virtuelle netværk:** NSG'er bruges til at styre netværkstrafik til og fra Azure-ressourcer på subnet- eller netværksinterface-niveau. De indeholder regler, der tillader eller nægter trafik baseret på IP-adresse, port og protokol.

**SSH med nøgler:** SSH (Secure Shell) bruges til sikker adgang til VM'er i Azure. Ved at anvende nøglepar (offentlig og privat nøgle) i stedet for adgangskoder, forbedres sikkerheden betydeligt.

**Azure Resource Manager (ARM):** ARM er Azure's implementerings- og administrationsservice, der muliggør gruppering af relaterede ressourcer, så de kan administreres som en enhed. ARM-templates bruges til at definere infrastrukturen som kode.

**Udregning af budget:** Budgettering i Azure indebærer overvågning af forbrug, estimering af fremtidige omkostninger og opsætning af advarsler for at undgå overskridelser. Azure Cost Management and Billing værktøjer hjælper med dette.

**ARM-templates (”bicep” også, men vi har ikke snakket ret meget om det):** ARM-templates er JSON-filer, der definerer infrastrukturer og tjenester. Bicep er et domænespecifikt sprog (DSL), der forenkler skrivningen af ARM-templates ved at reducere kompleksiteten og forbedre læsbarheden.

**Uddybning af relaterede emner:**

1. **IP-adressering og subnets:**
   * **IP-adresser:** Kan være offentlige (til ekstern adgang) eller private (til intern kommunikation). Dynamiske IP'er tildeles automatisk, mens statiske IP'er tildeles manuelt og forbliver uændrede.
   * **Subnets:** Del netværket i mindre segmenter for bedre organisering og sikkerhed. Hver subnet har sin egen IP-adresseblok.
2. **Azure networking basics:**
   * **VNet konfiguration:** Opret og administrer VNets via Azure Portal eller CLI. Konfigurer subnets og tildel IP-adresser.
   * **Peering:** Forbind VNets for at tillade ressourcekommunikation på tværs af forskellige netværk. Peering er som en bro mellem to VNets, hvilket sikrer lav latens og høj båndbredde.
3. **Netværkssikkerhedsgrupper (NSG):**
   * **Opsætning:** Opret NSG'er og definer regler, der styrer indgående og udgående trafik. Regler specificerer kilde og destination IP-adresser, protokoller og porte.
   * **Integration:** Tildel NSG'er til subnets eller individuelle netværksinterfaces for at beskytte specifikke ressourcer.
4. **SSH med nøglepar:**
   * **Nøgleoprettelse:** Generer et nøglepar ved hjælp af SSH-keygen eller lignende værktøjer. Den offentlige nøgle placeres på VM'en, mens den private nøgle opbevares sikkert af brugeren.
   * **Sikker adgang:** Brug SSH-klienter (som PuTTY eller OpenSSH) til at oprette forbindelse til VM'en ved hjælp af den private nøgle.
5. **ARM-templates og budgetstyring:**
   * **ARM-templates:** Brug JSON-filer til at deklarere og implementere Azure-ressourcer. Definer ressourceafhængigheder og konfigurationsdetaljer.
   * **Bicep:** En enklere syntaks til at skrive ARM-templates, der gør det lettere at oprette og vedligeholde komplekse deployment-konfigurationer.
   * **Budgetstyring:** Brug Azure Cost Management and Billing til at spore udgifter, opsætte budgetter og modtage advarsler ved overskridelser. Optimer omkostninger ved at analysere ressourceforbrug og justere tildelinger efter behov.

Disse uddybningspunkter giver en dybere forståelse af, hvordan virtuelle netværk og netværkssikkerhed fungerer i Azure, samt hvilke overvejelser man skal have i tankerne for at opnå optimal ydeevne og sikkerhed.

## Modul 4: Azure Web Apps og Load Balancing

**Beskrivelse:** Azure Web Apps er en PaaS-tjeneste, der gør det muligt at bygge, implementere og skalere webapplikationer hurtigt og nemt uden at håndtere den underliggende infrastruktur. Load balancing er en teknik til at distribuere indgående netværkstrafik på tværs af flere servere for at sikre høj tilgængelighed og pålidelighed af applikationer. Dette modul fokuserer på fordelene ved at bruge Azure App Service fremfor virtuelle maskiner (VM'er) til webapplikationer, samt konfiguration og integration af storage.

**Azure Web Apps og load balancing:** Azure Web Apps muliggør hosting af webapplikationer uden at administrere servere. Load balancing sikrer, at trafik distribueres jævnt mellem flere instanser af webapplikationen, hvilket forbedrer ydeevne og tilgængelighed.

**Web app configuration:** Web apps kan konfigureres gennem Azure Portal, Azure CLI eller ARM templates. Konfigurationen omfatter indstillinger for applikationsindstillinger, forbindelsesstrenge, SSL-certifikater, skaleringsindstillinger og diagnosticering.

**Web app med storage:** Webapplikationer kan integreres med Azure Storage for at gemme data som blob storage (til filer og multimedier), table storage (til strukturerede data) eller queue storage (til messaging). Dette gør det muligt at håndtere store datamængder effektivt og skalerbart.

**Fordele og ulemper ved at bruge Azure App Service fremfor en VM i en webbaseret løsning:**

**Fordele:**

* **Ingen serveradministration:** Azure håndterer opdateringer, skalering og vedligeholdelse af den underliggende infrastruktur.
* **Indbygget skalerbarhed:** Automatiske skaleringsmuligheder gør det let at tilpasse ressourcer baseret på belastning.
* **Integration med DevOps-værktøjer:** Understøtter CI/CD pipelines og integration med GitHub, Azure DevOps og andre værktøjer.
* **Høj tilgængelighed:** Indbygget load balancing og redundans sikrer høj oppetid.
* **Sikkerhed:** Understøtter SSL-certifikater, IP-begrænsninger og integration med Azure Active Directory.

**Ulemper:**

* **Mindre kontrol:** Begrænset adgang til underliggende servere og operativsystemer, hvilket kan være nødvendigt for visse avancerede konfigurationer.
* **Begrænsninger i tilpasning:** Visse applikationer kan kræve specifikke softwarekonfigurationer, der ikke understøttes af App Service.
* **Omkostninger:** Kan være dyrere end VM'er ved meget store skalaer eller særlige brugerdefinerede behov.

**Uddybning af relaterede emner:**

1. **Oprettelse og konfiguration af Azure Web Apps:**
   * **Oprettelse:** Brug Azure Portal, CLI eller ARM templates til at oprette en ny Web App. Vælg den ønskede runtime stack (f.eks. .NET, Node.js, Python).
   * **Konfiguration:** Indstil applikationsindstillinger, forbindelsesstrenge til databaser, og implementer SSL-certifikater for sikker kommunikation.
2. **Load balancing teknikker:**
   * **Azure Load Balancer:** Distribuerer indgående trafik på tværs af VM'er. Anvendes typisk for IaaS-løsninger.
   * **Azure Application Gateway:** En layer 7 load balancer, der tilbyder yderligere funktioner som SSL offloading, URL-baseret routing og WAF (Web Application Firewall).
   * **Traffic Manager:** En DNS-baseret load balancer, der dirigerer trafik baseret på DNS-forespørgsler, anvendelig til global skalerbarhed.
3. **Storage integration med Web Apps:**
   * **Blob Storage:** Bruges til at gemme store filer som billeder, videoer og dokumenter. Web Apps kan let tilgå og administrere blob data via API'er.
   * **File Storage:** Tilbyder delte filsystemer, som kan mountes direkte til Web Apps for delte data.
   * **Queue Storage:** Hjælper med messaging og køhåndtering mellem forskellige komponenter af applikationen, hvilket sikrer løs kobling og skalerbarhed.
4. **DevOps integration:**
   * **CI/CD pipelines:** Implementer kontinuerlig integration og deployment ved at konfigurere pipelines i Azure DevOps eller GitHub Actions. Automatiser build, test og deployment processer.
   * **Application Insights:** Bruges til at overvåge applikationens ydeevne, fejlfinding og analysere brugeradfærd. Integrer Application Insights i Web Apps for at få dyb indsigt i applikationens helbred og ydeevne.
5. **Sikkerhed og overholdelse:**
   * **SSL/TLS:** Implementer SSL/TLS for at sikre kommunikation mellem klienter og webapplikationen.
   * **Azure Active Directory (AAD):** Brug AAD til autentificering og autorisation af brugere, og integrer AAD B2C for at håndtere kundeadgang.
   * **Compliance:** Sørg for, at applikationen overholder lovgivningsmæssige krav ved at bruge Azure's compliance tilbud, som hjælper med at opfylde globale, regionale og branchespecifikke krav.

Disse uddybningspunkter giver en dybere forståelse af, hvordan Azure Web Apps og load balancing fungerer, samt hvordan man kan drage fordel af disse teknologier for at bygge skalerbare, pålidelige og sikre webapplikationer.

## Modul 5: Cloud Databaser

**Beskrivelse:** Cloud databaser, såsom Azure Cosmos DB, tilbyder skalerbarhed, høj tilgængelighed og global distribution, hvilket gør dem ideelle til moderne applikationer. Dette modul fokuserer på Cosmos DB, en globalt distribueret, multi-model database-tjeneste fra Microsoft Azure, og dækker designovervejelser, service level agreements (SLA'er), og nødvendige trin for at bringe en Cosmos DB i drift.

**Cloud databaser:** Cloud databaser giver mulighed for at gemme, administrere og skalere data uden at håndtere den underliggende infrastruktur. De tilbyder høj tilgængelighed, automatiske backups, og fleksible skaleringsmuligheder.

**Cosmos DB:** Azure Cosmos DB er en fuldt administreret NoSQL database-tjeneste, der understøtter flere datamodeller som dokumenter, grafer og kolonner. Den er designet til høj ydeevne og global distribution.

**Cosmos DB .Net:** Cosmos DB kan integreres med .NET-applikationer ved hjælp af Azure SDK'er. Dette gør det muligt for udviklere at interagere med Cosmos DB ved hjælp af velkendte .NET-biblioteker og API'er.

**Designovervejelser ved oprettelse af en Cosmos DB løsning:** Når man opretter en Cosmos DB løsning, bør man overveje følgende:

* **Datamodel:** Vælg den mest passende datamodel (dokument, graf, tabel, eller kolonne) baseret på applikationens behov.
* **Partitionering:** Planlæg partitioneringsstrategien for at sikre optimal ydeevne og skalerbarhed.
* **Konsistensniveau:** Vælg mellem fem konsistensniveauer (strong, bounded staleness, session, consistent prefix, eventual) baseret på krav til dataintegritet og latency.
* **Global distribution:** Konfigurer replikation til flere regioner for at forbedre ydeevne og tilgængelighed.

**Service Level Agreement (SLA):** Azure Cosmos DB tilbyder SLA'er for gennemløb (throughput), latency, konsistens og høj tilgængelighed. Dette giver garanti for en pålidelig og højtydende tjeneste.

**Azure “Request Units”:** Request Units (RU'er) er en valuta for at måle gennemløb i Cosmos DB. Hver forespørgsel, indsættelse, opdatering eller sletning koster et bestemt antal RU'er, baseret på operationens kompleksitet og datamængde.

**Nødvendige trin til at bringe en Cosmos DB op og køre:**

1. **Opret en Cosmos DB konto:** Brug Azure Portal, CLI eller ARM templates.
2. **Vælg API:** Vælg den passende API (SQL, MongoDB, Cassandra, Gremlin, Table) baseret på applikationskrav.
3. **Opret database og container:** Konfigurer databasen og containere, og definer partitioneringsnøgle.
4. **Indstil gennemløb:** Tildel gennemløb (RU'er) til databasen eller containere.
5. **Konfigurer global distribution:** Vælg regioner for replikation og opsæt multi-region writes, hvis nødvendigt.
6. **Implementer sikkerhed:** Anvend adgangskontrol ved hjælp af tokens og nøglebaseret autentificering.
7. **Monitor og optimer:** Brug Azure Monitor og Application Insights til at spore ydeevne og justere konfigurationer efter behov.

**Uddybning af relaterede emner:**

1. **Datamodeller og API'er i Cosmos DB:**
   * **SQL API:** Bruges til dokumenter og tilbyder SQL-lignende forespørgselssprog.
   * **MongoDB API:** Giver kompatibilitet med MongoDB-protokollen.
   * **Cassandra API:** Understøtter Cassandra Query Language (CQL).
   * **Gremlin API:** Bruges til grafdatabaser og understøtter Gremlin traversal language.
   * **Table API:** Tilbyder en NoSQL key-value lagringsmodel.
2. **Partitionering og skalerbarhed:**
   * **Partitioneringsnøgle:** Vælg en nøgle, der fordeler data jævnt for at undgå hot spots.
   * **Gennemløbsstyring:** Juster RU'er baseret på applikationens krav for at sikre tilstrækkelig kapacitet og ydeevne.
3. **Konsistensniveauer:**
   * **Strong:** Garanterer den højeste konsistens, men med højere latency.
   * **Bounded Staleness:** Garanterer en maksimal forsinkelse mellem skrivninger og læsninger.
   * **Session:** Sikrer konsistens inden for en session.
   * **Consistent Prefix:** Garanterer, at læsninger aldrig ser et ude af rækkefølge sæt af opdateringer.
   * **Eventual:** Garanterer den laveste latency, men med mulighed for at læsninger kan være inkonsistente i en kort periode.
4. **Integration med .NET:**
   * **Azure SDK:** Brug Azure Cosmos DB SDK for .NET til at interagere med databasen. Implementer CRUD-operationer, forespørgsler og transaktioner i .NET-applikationer.
   * **Dependency Injection:** Konfigurer Cosmos DB klienten som en tjeneste i ASP.NET Core for lettere brug og testbarhed.
5. **Overvågning og optimering:**
   * **Azure Monitor:** Spor ydeevne, throughput, latency og fejlrate. Brug Metrics og Logs til at identificere og løse problemer.
   * **Application Insights:** Integrer med Application Insights for dybdegående applikationsydelse og diagnostik. Optimer RU-forbrug ved at analysere forespørgselsmønstre og justere indeksering.

Disse uddybningspunkter giver en dybere forståelse af, hvordan man effektivt bruger og administrerer Azure Cosmos DB, samt hvilke overvejelser der er nødvendige for at opnå optimal ydeevne, skalerbarhed og omkostningseffektivitet i cloud databaser.

## Modul 6: Containers

**Beskrivelse:** Containere er en teknologi, der gør det muligt at pakke applikationer og deres afhængigheder i en enkelt enhed, hvilket sikrer, at de kører ensartet på tværs af forskellige miljøer. Azure Containers tilbyder en række tjenester til at administrere og orkestrere containere i skyen. Dette modul fokuserer på forskellen mellem containere og virtuelle maskiner, fordele og ulemper ved at bruge containere, og workflowet for at bringe en container i operation.

**Containers:** Containere er letvægtsenheder, der indeholder alt, hvad der er nødvendigt for at køre en applikation, inklusiv kode, runtime, systemværktøjer, biblioteker og indstillinger. De kører isoleret fra hinanden på den samme værtskerne, hvilket giver effektiv ressourceudnyttelse.

**Azure Containers:** Azure tilbyder flere tjenester til containeradministration, herunder Azure Kubernetes Service (AKS), Azure Container Instances (ACI), og Azure Container Registry (ACR). Disse tjenester hjælper med implementering, skalerbarhed og vedligeholdelse af containeriserede applikationer.

**Forskellen på en container og en virtuel maskine, samt fordele og ulemper:**

**Containere:**

* **Letvægts:** Deler værtsoperativsystemets kerne, hvilket gør dem mindre og hurtigere at starte.
* **Bærbarhed:** Ensartet miljø uanset hvor de kører (udviklerens maskine, testmiljø, produktion).
* **Isolering:** Selvom de deler værtskerne, kører de isoleret, hvilket forbedrer sikkerheden.

**Virtuelle maskiner (VM'er):**

* **Tunge:** Indeholder en hel OS-installation, hvilket gør dem større og langsommere at starte.
* **Fleksibilitet:** Fuldt isolerede og kan køre forskellige operativsystemer på samme hardware.
* **Overhead:** Mere ressourcekrævende pga. behovet for hypervisor og fuldt OS.

**Fordele ved containere:**

* **Ressourceeffektivitet:** Mindre ressourceforbrug sammenlignet med VM'er.
* **Hurtig opskalering:** Hurtigere opstart og opskalering.
* **Ensartethed:** Samme miljø fra udvikling til produktion.
* **Microservices:** Ideel til at køre microservices-arkitekturer.

**Ulemper ved containere:**

* **Sikkerhedsrisici:** Deling af værtskerne kan udgøre sikkerhedsrisici.
* **Kompleksitet:** Administration af mange containere kan blive komplekst, kræver avancerede orkestreringsværktøjer.

**Workflowet for at bringe en container i operation:**

1. **Byg container image:**
   * Opret en Dockerfile, der specificerer applikationen og dens afhængigheder.
   * Byg image'et ved hjælp af Docker CLI: docker build -t myapp:latest .
2. **Test container lokalt:**
   * Start containeren lokalt for at sikre, at den fungerer som forventet: docker run -p 8080:80 myapp:latest
3. **Push image til container registry:**
   * Log ind på Azure Container Registry (ACR): az acr login --name myregistry
   * Tag image'et og push det til ACR:

bash

Kopier kode

docker tag myapp:latest myregistry.azurecr.io/myapp:latest

docker push myregistry.azurecr.io/myapp:latest

1. **Deploy container til Azure:**
   * Brug Azure Kubernetes Service (AKS) eller Azure Container Instances (ACI) til at implementere containeren.
   * For AKS:

bash

Kopier kode

kubectl create deployment myapp --image=myregistry.azurecr.io/myapp:latest

kubectl expose deployment myapp --type=LoadBalancer --port=80 --target-port=8080

* + For ACI:

bash

Kopier kode

az container create --resource-group myResourceGroup --name myapp --image myregistry.azurecr.io/myapp:latest --dns-name-label myapp --ports 80

1. **Monitor og skaler:**
   * Overvåg ydeevnen og sundheden af containerne ved hjælp af Azure Monitor.
   * Skalér antallet af containere efter behov ved hjælp af Kubernetes auto-scaling eller ACI's skaleringstjenester.

**Uddybning af relaterede emner:**

1. **Azure Kubernetes Service (AKS):**
   * **Orkestrering:** Administrerer distribution, skalering og drift af containere ved hjælp af Kubernetes.
   * **Integration:** Indbygget integration med andre Azure-tjenester som Azure Monitor og Azure Active Directory.
2. **Azure Container Instances (ACI):**
   * **Enkelhed:** Kør containeriserede applikationer uden behov for at administrere den underliggende infrastruktur.
   * **Fleksibilitet:** Hurtig og nem måde at køre containere på, ideel til batchjob og burst workloads.
3. **Azure Container Registry (ACR):**
   * **Private registries:** Sikrer, at container images er tilgængelige for dine Azure-tjenester.
   * **Automatisering:** Understøtter CI/CD pipelines for automatisk build og deployment.
4. **Docker og Docker Compose:**
   * **Docker:** Platform til at bygge, teste og implementere containere.
   * **Docker Compose:** Værktøj til at definere og køre multi-container Docker-applikationer. Brug docker-compose.yml til at specificere tjenester, netværk og volumener.
5. **Sikkerhed og governance:**
   * **Container Security:** Implementer bedste praksis for sikkerhed, som at scanne images for sårbarheder, brug af begrænsede adgangsrettigheder og regelmæssige opdateringer.
   * **Policy Management:** Brug Azure Policy til at håndhæve governance og compliance krav for containerressourcer.

Disse uddybningspunkter giver en dybere forståelse af, hvordan containere og Azure container-tjenester fungerer, samt hvilke overvejelser der er nødvendige for at effektivt implementere og administrere containeriserede applikationer.

## Modul 7: Serverless Computing

**Beskrivelse:** Serverless computing er en cloud-udviklingsmodel, hvor cloud-udbyderen dynamisk administrerer tildeling af maskinressourcer. Azure Functions er en serverless compute-tjeneste, der gør det muligt at køre små kodestykker (funktioner) uden at bekymre sig om den underliggende infrastruktur. Dette modul fokuserer på fordele og ulemper ved serverless computing sammenlignet med containerbaserede løsninger, opbygningen af Azure Functions, og integration med Cosmos DB.

**Serverless computing – fordele og ulemper i forhold til en containerbaseret løsning:**

**Fordele:**

* **Ingen serveradministration:** Udviklere behøver ikke at administrere eller provisionere servere.
* **Automatisk skalering:** Funktionen skalerer automatisk baseret på antallet af indkommende anmodninger.
* **Betal kun for forbrug:** Betaling er baseret på den faktiske udførelsestid for koden, hvilket kan reducere omkostningerne.
* **Hurtig udvikling og implementering:** Gør det muligt at fokusere på at skrive kode uden at bekymre sig om infrastruktur.

**Ulemper:**

* **Koldstart:** Der kan være en forsinkelse (kaldet koldstart) når en funktion, der ikke har været brugt i et stykke tid, bliver kaldt.
* **Begrænsninger i køretid:** Funktioner kan have begrænsninger i maksimal køretid og ressourceforbrug.
* **Kompleks debugging:** Fejlfinding og lokal test af serverless funktioner kan være mere kompleks end traditionelle applikationer.

**Azure Functions – beskriv opbygningen:** Azure Functions er en event-drevet serverless compute-tjeneste, der består af flere nøglekomponenter:

* **Function App:** En container til en eller flere funktioner, der deler samme konfiguration og skaleringsindstillinger.
* **Triggers:** Angiver, hvordan funktionen aktiveres. Eksempler inkluderer HTTP-anmodninger, timer, og meddelelser fra køer.
* **Bindings:** Gør det nemt at integrere med andre Azure-tjenester som Cosmos DB, Storage, og Event Hubs. Bindings kan være input eller output.
* **Kørselstid:** Miljøet hvor funktionen kører. Azure Functions understøtter flere sprog som C#, JavaScript, Python, og Java.

**Functions med Cosmos DB:** Azure Functions kan integreres med Cosmos DB ved hjælp af bindings, der muliggør læsning og skrivning af data uden eksplicit forbindelse eller konfiguration i koden. For eksempel kan en funktion trigges af ændringer i en Cosmos DB-collection, hvilket gør det let at reagere på dataændringer.

**Uddybning af relaterede emner:**

1. **Triggers og bindings:**
   * **Triggers:** Definerer begivenheder, der aktiverer funktionen. Eksempler inkluderer HTTP-triggers for webhooks og Timer-triggers for planlagte opgaver.
   * **Bindings:** En nem måde at forbinde funktioner til andre tjenester uden at skrive omfattende integrationskode. Inputbindings leverer data til funktionen, mens outputbindings sender data til en tjeneste.
2. **Skaleringsmodeller i serverless computing:**
   * **Event-driven skalering:** Funktionen skalerer automatisk i realtid baseret på antallet af indkommende begivenheder.
   * **Forbrugsmæssig plan:** Betalingsmodellen, hvor man betaler kun for den faktiske kørselstid af funktionerne.
   * **Premium og Dedicated plans:** Tilbyder yderligere funktioner som forudsigelig ydeevne og længere køretid for enterprise applikationer.
3. **Udvikling og deployment af Azure Functions:**
   * **Local development:** Brug Azure Functions Core Tools til at udvikle og teste funktioner lokalt.
   * **CI/CD pipelines:** Integrer med Azure DevOps eller GitHub Actions for at automatisere build og deployment processer.
   * **Deployment slots:** Muliggør staging og produktion slots for at sikre problemfri deployment og rollback.
4. **Overvågning og fejlfinding:**
   * **Azure Monitor:** Overvåg funktionens ydeevne og fejlhåndtering ved hjælp af Azure Monitor.
   * **Application Insights:** Integrer med Application Insights for avanceret logging, metrics og diagnosticering af funktioner.
5. **Best practices for serverless arkitektur:**
   * **Stateless functions:** Design funktioner til at være stateless for bedre skalerbarhed.
   * **Modularize functions:** Opdel funktionalitet i små, fokuserede funktioner for bedre genbrugelighed og vedligeholdelse.
   * **Error handling:** Implementer robust fejlhåndtering og retries for at sikre pålidelighed.
   * **Security:** Brug Managed Identities og Azure Key Vault til at sikre adgang til secrets og konfigurationsindstillinger.

Disse uddybningspunkter giver en dybere forståelse af, hvordan serverless computing og Azure Functions fungerer, samt hvilke overvejelser og bedste praksis der er nødvendige for effektivt at implementere og administrere serverless applikationer i Azure-miljøet.

## Modul 8: Messaging og Sikring af Information

**Beskrivelse:** Messaging og sikring af information er vigtige aspekter af cloud computing, der håndterer kommunikation og sikkerhed mellem forskellige applikationskomponenter. Dette modul fokuserer på Azure's messagingplatforme, brugen af Azure Key Vault til sikring af nøgler og hemmeligheder samt implementering af load balancing i webapplikationer.

**Azure Messaging Platforms:** Azure tilbyder forskellige messagingtjenester, herunder Azure Service Bus og Azure Event Grid, der muliggør pålidelig og skalerbar kommunikation mellem forskellige applikationskomponenter.

**Vault:** Azure Key Vault er en cloudtjeneste, der fungerer som et sikkert lager til lagring af adgangs-tokens, kryptografiske nøgler og andre hemmeligheder. Det giver en centraliseret og sikker måde at administrere adgang til disse ressourcer.

**Service Bus – hvordan den anvendes i et eksempel:** Azure Service Bus er en fuldt administreret tjeneste til asynkron kommunikation mellem applikationskomponenter og tjenester. Det understøtter køer og emner til at udveksle meddelelser på en pålidelig og skalerbar måde.

**Azure Key Vault:** Azure Key Vault tilbyder sikker opbevaring af kryptografiske nøgler, certifikater og andre hemmeligheder, samt håndtering af deres livscyklus. Det integreres sømløst med andre Azure-tjenester og applikationer.

**Hvad er messaging:** Messaging refererer til udveksling af meddelelser mellem forskellige applikationskomponenter eller systemer. Det kan være synkron eller asynkron og bruges til at muliggøre løs kobling og skalerbarhed.

**Forskellene på Azure messagingplatformene:**

* **Azure Service Bus:** Ideel til pålidelig asynkron kommunikation mellem applikationskomponenter.
* **Azure Event Grid:** Optimeret til høj skalerbarhed og event-baseret kommunikation mellem tjenester.
* **Azure Event Hubs:** Designet til indsamling, lagring og behandling af store mængder af streamingdata fra enheder og applikationer.

**Load Balancing:** Load balancing er en teknik til at distribuere indgående netværkstrafik på tværs af flere servere eller virtuelle maskiner for at forbedre tilgængelighed og ydeevne af en applikation.

**Fordele og ulemper ved at bruge Azure App Service fremfor en virtuel maskine i en webbaseret løsning:** Samme fordele og ulemper, som tidligere diskuteret i Modul 4.

**Uddybning af relaterede emner:**

1. **Azure Service Bus:**
   * **Køer:** Bruges til asynkron kommunikation mellem applikationskomponenter, hvor beskeder gemmes i køen, indtil de bliver behandlet af modtagende applikationer.
   * **Emner:** Tillader publikation/abonnement-mønstre, hvor flere modtagere kan lytte efter og modtage meddelelser fra samme emne.
2. **Azure Event Grid:**
   * **Events:** Representerer ændringer eller hændelser i Azure eller i kundens applikationer, som kan udløse automatiske handlinger.
   * **Topics and Subscriptions:** Events publiceres til topics, og interesserede parter (subscribers) abonnerer på bestemte topics for at modtage relevante events.
3. **Azure Event Hubs:**
   * **Streaming Data Platform:** Optimeret til indsamling, aggregering og analyse af store mængder af streamingdata fra enheder og applikationer.
   * **Partitions:** Data partitioneres for at muliggøre parallel behandling og skalerbarhed.
4. **Azure Key Vault:**
   * **Secrets Management:** Lagrer og administrerer følsomme oplysninger som adgangskoder, API-nøgler og SSL-certifikater.
   * **Role-based Access Control (RBAC):** Giver mulighed for at styre adgang til nøgler og hemmeligheder baseret på brugerroller.
5. **Load Balancing Techniques:**
   * **Azure Load Balancer:** Distribuerer indgående trafik på tværs af flere virtuelle maskiner for at sikre høj tilgængelighed og skalerbarhed.
   * **Azure Application Gateway:** En layer 7 load balancer, der tilbyder yderligere funktioner som SSL offloading, URL-baseret routing og WAF (Web Application Firewall).

Disse uddybningspunkter giver en dybere forståelse af, hvordan messaging og sikring af information fungerer i Azure-miljøet, samt hvordan man effektivt integrerer og administrerer disse tjenester i cloud-applikationer.

## Modul 9: Tilgængelighed og Skalering af Applikationer

**Beskrivelse:** Tilgængelighed og skalering af applikationer er afgørende for at sikre, at applikationer kan håndtere varierende belastninger og forblive tilgængelige for brugerne. Dette modul fokuserer på teknikker og værktøjer til at skalere og sikre tilgængelighed af Azure-baserede applikationer, herunder skalering af applikationer, messaging med DTO (Data Transfer Objects), og implementering af skalering med load balancer.

**Skalering af applikationer:** Skalering af applikationer indebærer justering af antallet af ressourcer (f.eks. virtuelle maskiner, containere) for at imødekomme ændringer i belastningen. Dette kan være vertikal (opgradering af eksisterende ressourcer) eller vandret (tilføjelse eller fjernelse af ressourcer).

**Messaging med DTO:** Data Transfer Objects (DTO'er) er objekter, der bruges til at overføre data mellem lag eller systemer i en applikation. Ved hjælp af DTO'er kan datastrukturer standardiseres og optimeres til effektiv kommunikation.

**Skalering med Load Balancer:** Load balancing er en teknik til at distribuere netværkstrafik jævnt på tværs af flere servere eller virtuelle maskiner for at forbedre ydeevnen og tilgængeligheden af en applikation. Azure tilbyder load balancer-tjenester, der automatisk distribuerer trafik baseret på forskellige algoritmer.

**Uddybning af relaterede emner:**

1. **Autoskalering:**
   * **Vertikal autoskalering:** Automatisk tilpasning af ressourcestørrelsen baseret på belastningskravene. Dette kan omfatte opgradering af CPU, hukommelse eller diskplads.
   * **Vandret autoskalering:** Dynamisk justering af antallet af instanser eller ressourcer baseret på belastningen. Dette kan omfatte tilføjelse eller fjernelse af virtuelle maskiner, containere eller funktioner.
2. **Azure Autoscale:**
   * **Skalering af virtuelle maskiner:** Konfigurer autoskalering for virtuelle maskiner ved hjælp af Azure Monitor-metriker som CPU-belastning eller hukommelsesforbrug.
   * **Skalering af AKS:** Konfigurer autoskalering for Azure Kubernetes Service (AKS) ved hjælp af horisontal pod autoscaler (HPA) baseret på CPU- eller hukommelsesforbrug.
3. **Messaging Patterns:**
   * **Point-to-point (P2P):** Direkte kommunikation mellem to applikationskomponenter gennem en midlertidig beskedkø.
   * **Publish/subscribe:** Flere abonnenter modtager kopier af samme besked, når den offentliggøres i et emne.
4. **DTO Design:**
   * **Data Transfer Objects:** Defineret til at overføre data mellem forskellige lag eller komponenter i en applikation.
   * **Serialization:** Konvertering af DTO'er til et format, der kan transporteres over netværket (f.eks. JSON eller XML).
5. **Implementering af Load Balancer:**
   * **Azure Load Balancer:** Distribuerer indgående trafik jævnt på tværs af flere virtuelle maskiner for at forbedre tilgængeligheden og skalerbarheden af en applikation.
   * **Application Gateway:** En layer 7 load balancer, der tilbyder avancerede funktioner som SSL-termination, URL routing og WAF.

Disse uddybningspunkter giver en dybere forståelse af, hvordan man effektivt håndterer tilgængelighed og skalerbarhed i Azure-miljøet, samt hvilke værktøjer og teknikker der er nødvendige for at sikre en pålidelig og responsiv cloud-applikation.