Inlämningsuppgift 2: **Om driftsättning av projekt**

1. Vilken är din valda tjänst, och vilken utgivare står bakom?

Mitt projekt som ska driftsättas är gjort med ASP.NET Core i kombination med ett UI som är gjort som en SPA och med React js.
Tjänsten som jag valt att driftsätta projektet med är en IaaS (Infrastrukture as a Service).
Företaget som levererar tjänsten är DigitalOcean (https://www.digitalocean.com).
DigitalOcean kallar just denna typen av tjänst för ”Droplets” (https://www.digitalocean.com/products/droplets).
Tjänsten innebär att jag köper en virtuell maskin (VPS), som jag via deras webbsida konfigurerar utifrån behov, och som därefter skapas på fysiska servrar i en befintlig maskinpark som DigitalOcean driver och underhåller.

2. Vilken typ av molnsystem är din tjänst?

Molnsystemet är ett publikt molnsystem (Public Cloud). Det innebär att det drivs av en tredjepart, i detta fall DigitalOcean och är publikt för alla att kunna köpa olika tjänster som delas på samma hårdvara.
Själva lösningen som jag valt att skapa i detta moln är en **IaaS** lösning. En IaaS-lösning innebär stor frihet, är kraftfull och dynamisk men kräver ofta en hel del konfiguration och till viss del löpande underhåll. I detta fall är det en VPS som behöver konfigureras, skapas, installera program för applikationen, hantera miljövariabler och skapa deamon services och ladda upp de körbara applikationsfilerna via t.ex. sFTP. Eftersom det i detta fallet inte heller finns något grafiskt gränssnitt för detta, görs det via terminalen och ssh. För överföring av filer är det dock möjligt att använda en FTP-client vilket har ett grafiskt användargränssnitt.

3. Vad är fördelarna och nackdelarna med att driftsätta applikationen på ditt valda molnsystem?

Fördelarna är många med att använda sig av ett molnsystem för driftsättning. Tar vi mitt valda exempel d.v.s. att skapa en virtuell server med tjänsten Droplets på DigitalOcean, så listar jag nedan exempel på både fördelar och nackdelar.

Fördelar:

- Du skapar din droplet med några få knapptryckningar och den är direkt uppkopplad mot internet med en fast IP.
- Du betalar för just den storlek (konfiguration) du behöver för ditt behov.
- Du behöver inte själv hålla en fysisk server uppdaterad, varken mjukvaru- eller hårdvarumässigt.
- Du kan enkelt och snabbt öka eller minska storleken på servern beroende på förändrade behov.
- Prestandan är alltid den samma även vid hög belastning.
- Du kan välja vart i världen din server ska skapas och dessutom skapa två på olika ställen för att säkerställa driften om det skulle hända en katastrof på ett av ställena.
- Du kan enkelt utöka med fler tjänster utifrån behov. T.ex. lägga till en databas eller backup.
- Driftsäkerheten är mycket hög, d.v.s. skulle strömmen försvinna finns det system som gör att maskinerna (servrarna) fortsätter att köras.

Nackdelar:

- Du måste lita på företaget du anlitar för din tjänst faktiskt håller vad de lovar ang. t.ex. datasäkerhet. D.v.s. du har själv ingen insyn eller möjlighet att veta vem/vilka som kan komma åt din data.
- För att komma åt din tjänst kräver det remote access. Du behöver alltså själv vara uppkopplad mot internet för att kunna göra justeringar eller hämta data. Skulle kunna vara aktuellt i krigssituationer eller katastrofer.
- Denna droplet saknar ett grafiskt gränssnitt, viket kan upplevas som svårare att hantera.

Jämför vi med att själv konfigurera sin egna fysiska server på plats, så har det också både fördelar och nackdelar. Här kommer några exempel.

Fördelar:

- Du har själv kontroll på din data och behöver inte lita på något annans tjänster.
- Du kommer åt din data utan att vara uppkopplad mot internet.
- En fast inköpskostnad som inte ändras.

Nackdelar:

- En hög startkostnad.

- Du behöver installera operativsystem själv.
- Du behöver hålla hårdvaran uppdaterad.
- Svårt att uppdatera hårdvara utan att stänga ner servern.
- Själv hantera strömbortfall.
- Själv hantera backup av systemet och data vid t.ex. en krasch.
- Hantera internetuppkoppling med fast IP.
- Ev. hantera säkerheten på ditt lokala nätverk, där kanske servern ingår och är publik.
- Ofta används bara en liten del av kapaciteten från början, och gör det till en onödigt hög startkostnad.

4. Vilka sätt finns tillgängliga för dig att driftsätta din applikation med ditt valda molnsystem?

Jag skulle också kunna driftsätta min applikation med DigitalOceans tjänst App. Det är en PaaS-tjänst och innebär inte att driftsätta en virtuell server (droplet).
Dock klarar inte den tjänsten ännu att bygga .NET Core applikationer, utan då skulle man behöva förpacka applikationen som en Docker-img och sedan istället bygga den med tjänsten App. Blir ett extra steg, men möjligt. Dock blir det både billigare och förhållandevis enklare att driftsätta denna applikationen på en VPS, DigitalOcean ”Droplets”, vilket därför det är att föredra.
För att driftsätta applikationen på en droplet (VPS) krävs dessa steg:

- Skapa en Droplet på DigitalOcean med operativsystemet Ubuntu.
- Logga in som root-användare på servern med ssh via terminalen på en internetuppkopplad dator.
- Med ”snap” installera .NET’s SDK i den version som applikationen är skriven för och skapa ett alias för dotnet.
- Bygg och samla (publish) din applikationskod till bytekod, så den blir körbar från en mapp på servern.
- Med sFTP kopplar du upp dig mot servern och överför dina byggda och samlade filer till en mapp på servern. Förslagsvis en www-mapp i var-mappen i root.
- I terminalen, gör din applikationsfil .dll körbar med [chmod +x]
- För att din applikation inte ska sluta fungera när du startat den så behövs det skapas en ”unit” för att starta en process, som håller igång applikationen. Detta görs genom att

skapa en fil med ändelsen [.service], samt konfigurera den för att ändra miljövariabler, sätta arbetsmapp, sökväg och vilken fil som ska startas, omstartskonfiguration m.m. Och sedan med sFTP överföra den till mappen /etc/systemd/system/. Om kunskap finns går det sjävklart skapa filen via terminalen med något valt textediteringsprogram.

- Uppdatera sedan ”units” med [systemctl daemon-reload] och sedan starta processen med [systemctl start {filnamn}] och [systemctl enable {filnamn}].
- Nu är applikationen driftsatt och går att nå på dropletens (VPS:ens) IP.
- Dock är ssl-sertifikatet självsignerat och när applikationen öppnas i en webbläsare får man en varning att sidan kan vara osäker. Vill man göra något åt detta, bör man först koppla en url till VPS:ens IP, skaffa ett ssl-certifikat för den specifika url:en och sedan går det använda t.ex. NGINX för att lägga in ssl-certifikat samt använda den som en reverse proxy om fler applikationer driftsätts på samma VPS.

5. På vilka sätt måste din applikation anpassas till den valda tjänsten?

Eftersom det är en IaaS-tjänst så kräver det en hel del konfiguration av ”Dropleten” istället för anpassningar i applikationen. Dessutom använder applikationens API och UI samma webbserver, Kestrel i .NET, så det blir inte mycket som behöver specifikt anpassas för just denna droplet-tjänsten (VPS). Dock finns det några övergripande delar som är bra att ta hänsyn till.

- Applikationen behöver vara en ASP.NET Core applikation, om den ska köras på operativsystemet Ubuntu på dropleten. DigitalOcean har inte Windows som val.
- I UI:t vid API-requests så skrivs inte hela url:en med, utan bara den sista delen. T.ex. /api/rooms/ och inte https://localhost:5001/api/rooms/ eftersom de ändras med miljövariablerna.
- I .csproj-filen skapa ett flöde som t.ex. hämtar node_modules med [npm ci] vid [dontet build] samt bygger dist-filer till UI:t med [npm run build] vid en ”publish”.
- Vid en ”publish” av applikationen behöver också utvecklaren veta om det ska vara en [--self-containd] ”publish” och i så fall vilket operativsystem den ska köras på. Eller om .NET installeras på dropleten (VPS:en). I det senare fallet behöver .NET versionen synkas mellan den version som applikationen görs för och att samma installeras på dropleten. Eller vise versa om dropleten redan existerar och applikationen behöver göras för en specifik version.