1. Välj ut ett stort, molnbaserat system/app/tjänst som ni känner till och som vi inte diskuterat noga under lektionstid. Det kan vara allt från appar till webtjänster till multiplayer-backends till myndighetslösningar.

Cloud Gaming: Google Stadia

2. Beskriv vad det är och vad den har för tekniska (molnbaserade) behov baserat på geografi, användarmängd, datatrafik, säkerhet, tillgänglighet, etc. Inga siffror behöver (eller kan) vara exakta - utan försök vara i rätt område. Att streama en timmes video på Netflix tar definitivt inte upp bara 1 MB, men inte heller 1 TB. 1 GB däremot, är rimligt.

Vad är cloud gaming?

Istället för att ha en kraftfull speldator eller spelkonsol i ditt hus, har du fjärråtkomst till en speldator i ett serverställ långt borta. Dessa servrar strömmar spelet till dig precis som en YouTube-video.

Spel kräver tvåvägsinteraktion av en spelare som skickar indata till molnet samtidigt som den tar emot video och ljud från servern och det måste göras väldigt snabbt så att du inte märker en fördröjning.

Det är därför det inte räcker att bara ha en snabb internetanslutning, den måste också vara stabil och du måste vara tillräckligt nära dessa servrar.

Molnspeltjänster kan förbruka tiotals gigabyte per timme eftersom du ständigt laddar ner videobildrutor.

Stadia

Stadia är en molntjänst för cloud gaming utvecklat av google. För att kunna använda tjänsten behövs en stabil internetanslutning på minst 10 mb/s upp till 35 mb/s beroende på videoupplösning. Stadia förbrukar mellan 5 till 20 GB data per timme.

På en dator körs spelen i webbläsaren google Chrome.

När spelaren väljer ett spel startar chrome en WebRTC video session, webbkommunikation i realtid (Det tillåter ljud- och videokommunikation att fungera på webbsidor genom att tillåta direkt peer-to-peer-kommunikation) och användaren kan börja spela.

Servern överför både video och ljud, medan användaren sänder sina inmatningar. På det här sättet, har videoström och inmatningsström olika trafikbelastningar.

För att minska latens och öka stream-hastigheten finns stadias servrar i utkanten av googles nätverk så nära spelarna som möjligt, i s.k. egde-nodes.

Varje Stadia server innehåller en anpassad x86-processor på 2,7 GHz, 16mb ram och en anpassad amd gpu som kan prestera 10,7 teraflops för att bearbeta videospel för flera individer.

Stadia körs på operativsystemet Debian och använder Vulkan, ett grafik och applikationsprogrammeringsgränssnitt, skräddarsytt av Google för molnbaserat spel.

Varje spel instans körs i en container. Containers gör det enkelt att dela CPU-, minnes-, lagrings- och nätverksresurser på operativsystemnivå och erbjuder en logisk paketeringsmekanism.

3. Lista de generella delar som behövs för detta system. Tänk på servrar, lagring, nätverk, säkerhet, monitorering, m.m.... (Gör ett enkelt diagram med ett förslag på utformning.) Systemet behöver inte vara perfekt - det kommer alltid finnas flaskhalsar - men kommentera gärna brister som ni ser.

Google har datacenter i 3 olika steg:

Data Centers:
Det finns 16 stycken datacenter som Google använder för hela sin verksamhet, innehåller den mesta processorkapaciteten och back-end-lagringen.
Dessa är identiska med varandra, om något händer med center A, kan det återskapas från center B.
För att logga in på Stadia använder man sitt Google-konto t ex som vid inloggning på Gmail, dessa användaruppgifter ligger här på datacentret. Säkerhetsmässigt hanteras kontona på dessa center.
(Stadias spelservrar ligger inte här, de håller till vid Edge-nodes.)

Edge Points of Presence, EDGE-pops.
Dessa är själva kopplingen mellan Googles datacenter och resten av internet, trafiken levereras alltid från en station som ligger så nära användaren som möjligt, för att minska lagg och spara på resurser.
Det finns totalt circa 190 st av dessa, ungefär lika många som det finns länder.
I praktiken innebär det här att Google har stationer vid varje stor anslutning till det globala internetnätverket.
Google har sina egna DNS-servrar på dessa stationer.

Edge-Nodes eller GGC, Google Global Cache.
Det finns över 7500 stycken av dessa världen över, inklusive en på Grönland.
En Edge-Node består huvudsakligen av mycket cache-minne för att hantera YouTube, Google play – tjänster och sökningar, detta för att avlasta databaserna och snabbt leverera populärt innehåll.
Själva spelservern för Stadia ligger på samma plats som dessa stationer, för att vara så nära spelaren som möjligt.
Denna punkt är den närmaste kopplingen mellan en användare och Google, eller en spelare och Stadia.
Stadia-kontrollen i sig är hårdkodad att ansluta via Wifi direkt till Stadia-servern via deras egna DNS.
Detta sänker fördröjningen ytterligare.

Google Stadia blev inte en så stor framgång som de först hoppats, men då mycket av infrastrukturen ändå används till övriga tjänster, kunde de lediga Stadia-resurserna istället hyras ut till andra spelutvecklare.

Några av säkerhetsmetoderna som används inom Google Cloud är:
Intrångsdetektering.
Flera lager av lastbalansering, både mjukvarumässigt och hårdvarumässigt för att motverka ev DDoS och för att jämna ut arbetsbördan mellan datacentren. Lastbalanseraren kan släppa eller strypa trafiken om det behövs.
All lagrad data i datacentrens databaser krypteras.
Kommunikationen mellan samtliga delar i systemet krypteras och kontrolleras hela tiden.
Använder Googles konton för inloggning inklusive för Stadia.
Internt förlitar sig systemet inte på brandväggar eller segmentering, istället används filtrering/kontroll på flera ställen i nätverket för att förhindra fejkade IP-adresser och anslutningar.
Detta maximerar tillgängligheten och hastigheten på deras tjänster.
Varje tjänst i infrastrukturen blir tilldelad en krypterad ID som den använder när den kommunicerar med andra tjänster och klienter, detta gör att varje tjänst kan identifiera sig själv och därmed

kommunicera med rätt motpart.
Varje spelare i Stadia är anonym för utomstående så det går inte att ”doxa” en annan spelare, dvs ta reda på deras IP-adress.

5. Vilka fördelar och nackdelar skulle finnas med att köra detta projekt "traditionellt", dvs med lokala fysiska servrar? Tänk på kostnad, prestanda, geografi, underhåll, säkerhet, etc. Ni behöver inga tydliga siffror här, utan beskriv mer generellt vad skulle kunna fungera sämre (eller bättre)?

Eftersom Google Stadia körs genom Google Cloud så hamnar detta projekt i något av en gråzon. Det är en molntjänst, men då molnet och alla dess fysiska servrar ägs av Google själva så körs det tekniskt sett även ”traditionellt” vilket ger dem möjlighet att baka in mer än bara vad de erbjuder till andra Cloud-kunder i projektet (som t.ex. företagets interna säkerhet och övervakning).
Det finns dock många fördelar för Google att huvudsakligen köra Stadia som en molntjänst.

Nummer ett: det är kostnadseffektivt! De kan som vi tidigare nämnt mycket enklare erbjuda resurserna som deras egen tjänst inte använder till andra molnkunder för ökad inkomst. Dessutom tillkommer som sagt ingen extra kostnad varken för underhåll eller säkerhet för Stadia då detta redan finns implementerat.

Nummer två: ökad prestanda! Med lokala fysiska servrar så påverkas användarnas ping beroende på hur nära eller långt bort de befinner sig. Genom att kunna sprida ut tjänsten på ett moln med servrar över hela världen så kan pingen hållas låg för de flesta användare.

Nummer tre: ökad skalbarhet! Flera tjänster, speciellt spel, har en fast maxkapacitet som ibland tvingar användare att få vänta i inloggningsköer. Även när spel som t.ex. World of Warcraft kan förutse en ökad användning, som veckorna efter lanseringen av en ny expansion, så förekommer ibland timmeslånga köer.
Trots att de temporärt ökar taket på sina egna servrars maxkapacitet genom att hyra in extra resurser från molntjänster så förändras inte faktumet att det fortfarande finns ett tak på hur många användare de kan ha aktiva samtidigt utan att det påverkar användarupplevelsen.
Här har Stadia det helt klart lättare då de enklare och med mindre begränsningar kan skala upp och ner, så deras användare behöver inte vänta på att resurser ska bli lediga innan de får spela. Även Stadia har såklart ett tak, men ett mer flexibelt ett.

Nummer fyra: ökad tillgänglighet! Genom att inte metaforiskt lägga alla äggen i en korg så krävs det mer än att en lokal serverhall går upp i brand eller på annat sätt sätts ur spel för att hela tjänsten ska ligga nere för alla användare.

Så sammanfattningsvis kan vi bara se fördelar och inga nackdelar för Google att köra Stadia i molnet istället för helt traditionellt!

6. Sammanställ ovanstående i en presentation som ni kort (ca 10 min inkl frågor) ger inför resten av klassen den 7/1.

Notera - Det finns såklart inget facit och dessa företag har hela team som jobbar heltid med att planera projekt som dessa. Gör ert bästa utifrån kursens innehåll och er ambitionsnivå.

1. Gemensamt inledande (kort rubrik i talform)
2. Marianne 2 minuter
3. Matti 2 minuter
4. Gemensamt 2 minuter av tekniska detaljer etc.
5. Ronja 2 minuter

+ 2 minuter öppet för frågor = 10 minuter briljans!

Tekniskt nonsens (courtesy of Marianne):

Videokodningen väljs automatiskt i början av sessionen. videokodningsformat:

H.264: Det är det videoformat som stöds mest nuförtiden, med 92 % av videoutvecklarna som använde det under 2018, följde av dess efterträdare H.265 med 42 %. Medan H.265 utlovar halva bithastigheten av H.264 vid samma bild kvalitet, H.264 har stöds i telefoner, surfplattor och webbläsare i flera år. Alltså för Stadia och andra innehållsleverantörer, fungerar det som en reserv för att säkerställa att nej användare kommer att få problem med att avkoda sina media.

VP9: Utvecklad av Google 2013 som efterträdare till VP8. Den är royaltyfri, till skillnad från H.264 och H.265, och har en prestanda jämförbar med H.265. Det används redan av youtube, och Google rapporterar att VP8 och VP9 utgör 90 % av WebRTC-kommunikationen via Google Chrome.

Ljudkodning görs genom Opus, ett öppet ljud codec släpptes 2012 och standardiserad av IETF. av-signerad med röst över IP och livedistribuerad musik framförd i åtanke kan den skala ljud från 6 kbps till 510 kbps. Det används flitigt av applikationer som WhatsApp2 och Jitsi3.

Google Stadia använder WebRTC för att tillhandahålla sina tjänster, vilket använder följande protokoll:

ICE, STUN och TURN: Interactive Connectivity Es-tablishment (ICE) är ett protokoll utformat för att underlätta peer-to-peer-funktioner i UDP-mediasessioner via Nätverksadressöversättare (NAT). Den använder Session Traversal Utilities för NAT (STUN) och Traversal med Relay NAT (TURN). STUN används för att fråga en server om användarens offentliga IP. När båda användarna vet deras respektive IP och port, kan de sedan dela denna information med den andra användaren för att upprätta en direkt förbindelse. TURN används när direktanslutning inte är möjligt (på grund av en symmetrisk NAT). Vid ett sådant tillfälle används en TURN-server som mellanhand mellan användarna.

I signaleringsprocessen, används Session Description Protocol (SDP) för att förhandla om parametrarna för sessionen (ljud- och videokodekar, IP och portar för RTCP-feedback, etc). Dessa byts ut tillsammans med ICE-kandidaterna, som informerar båda par om sina anslutningsalternativ (IP och port, direktanslutning eller via en TURN-server, transportprotokoll som används, etc). UDP är det föredragna protokollet för de flesta livestreaming applikationer, men Transmission Control Protocol (TCP) stöds också av ICE.

DTLS: Datagram Transport Layer Security (DTLS) används för att tillhandahålla säkerhet i datagrambaserad kommunikationer. Det designades som en implementering av TLS som krävde en pålitlig transportkanal (dvs designad för datautbyte som inte använder TCP), som en konsekvens av den ökade användningen av User Datagram Protokoll (UDP) för livestreamingapplikationer, som prioriterar snabb mottagning framför tillförlitlighet. DTLS har blivit standarden för den här typen av livestreaming applikationer, och det används i WebRTC för att säkra RTP-kommunikation.

RTP och RTCP: Real-Time Protocol (RTP) och Real-Time Control Protocol (RTCP) används för att sända media över den säkrade kanalen. RTP används av avsändare av media (Stadia), medan RTCP används av mottagande klient för att ge feedback på mottagandet kvalitet. RTP-paket skickas vanligtvis via UDP, och innehåller ett sekvensnummer och en tidsstämpel. De endpoints implementerar en jitterbuffert, som är van vid ordna om paket, samt att eliminera paketdubbletter, eller kompensera för olika mottagningstid. RTCP tillhandahåller mätvärden som kvantifierar kvaliteten på den mottagna strömmen. Vissa av dessa mätvärden är paket förlust, jitter, latens och det högsta sekvensnumret tas emot i ett RTP-paket. RTCP-paket är tidsinställda dynamiskt, men rekommenderas endast att ta hänsyn till 5 % av RTP/RTCP-trafiken. De har också ett maximum sändningsintervall på 5 sekunder.

När ICE-anslutningen och DTLS-kryptering är på plats, anslutningen består mestadels av RTP- och RTCP-paket för videoströmmen och applikationsdata (som vi antar inkluderar användarinmatningar) skickade genom DTLS-paket och STUN bindande förfrågningar skickas med jämna mellanrum för att säkerställa att peer till peer-anslutning kan upprätthållas.

Congestion control

Google Congestion Control har två huvudelement: en fördröjningsbaserad styrenhet på klientsidan och en förlustbaserad kontroller på serversidan. Den fördröjningsbaserade styrenheten använder fördröjningarna mellan sändning av videoram hos avsändaren och dess ankomst till mottagaren för att uppskatta buffertarnas tillstånd längs vägen (dvs. den jämför tiden det tog att sända en hel videoram hos avsändaren, med den tid det tog avsändaren att ta emot alla paket som bildar den ramen). Med denna information, såväl som den mottagande bithastigheten i den sista 500 ms, beräknar den nödvändiga bithastigheten Ax och vidarebefodrar det till avsändaren. Aviseringsmeddelanden från klienten till servern skickas varje sekund om det inte sker en betydande förändring (dvs. en skillnad högre än 3%) i den beräknade räntan med i förhållande till den föregående, i vilket fall de vidarebefordras omedelbart. Den förlustbaserade kontrollern fungerar på servern sida. Den uppskattar hastigheten Ay baserat på andelen paket förlorade från RTCP-meddelanden. Dess funktion är enkel: Om paketförlusten är under 0,02, ökas Ay. Tvärtom, om den är över 0,1 minskas takten Ay. I fall paketförluster ligger mellan 0,02 och 0,1, förblir Ay densamma. Avsändaren använder sedan minimum av de två hastigheterna, dvs min(Ax, Ay) till den aktuella hastigheten med vilken paket sänds.

Slut punkt 4.