R: Dan heb ik nu de opname gestart. Allereerst bedankt dat je me wilt joinen bij deze sessie. Allereerst zou ik uitleggen waar mijn onderzoek over gaat. En welke taak we het komende uur gaan voltooien. En welke achtergrond informatie erbij hoort. Dus ik doe onderzoek naar software product kwaliteit. En specifiek hoe je software product kwaliteit op een zo objectief en kwantitatief mogelijke manier kan meten. Daarvoor maak ik gebruik van de ISO25010 standaard. Een internationale standaard die software product kwaliteit voorschrijft aan de hand van acht kwaliteits karakteristieken. Wat ik tijdens het eerste deel van mijn studie heb gedaan is samen met engineers in discussiegroepen gekeken naar mogelijke measures. Dus metrics, data die we kunnen meten om iets te zeggen over die kwaliteits karakteristieken. En in het tweede deel, het deel wat wij samen gaan doen is we gaan kijken naar 1 a 2, afhankelijk van de tijd van die kwaliteits karakteristieken. En die gaan we beoordelen op eigenlijk drie dimensies. Allereerst gaan we kijken naar de source. Dus waar kan je dit data punt mogelijk vandaan halen? Moet je kijken naar de codebase in een repository? Moet je data vanuit bijvoorbeeld Github projects of Jira halen?
Zijn er dashboards in de tooling die je gebruikt die dit al voor je doen? Dat soort dingen. Vervolgens gaan we kijken naar de difficulty of obtaining data. Dus hoe moeilijk is het om de data te verkrijgen? Ik zal die ook tijdens het interview gewoon op het scherm laten. Ingeschat in drie dimensies, eigenlijk drie, drie waardes. Low, moderate en high.
Low betekent dat de data bijna gelijk available is en het maximaal een dag duurt om de data beschikbaar te maken. Dan heb je moderate. Moderate houdt in dat de data mogelijk beschikbaar is. Maar er is wel wat effort nodig om deze beschikbaar te maken. Door bijvoorbeeld wat stuff te preprocessen en cleanen. Gemiddeld dus één dag tot een week. En high is dat de data over het algemeen nog niet beschikbaar is. En er echt complexe effort nodig is om dit te doen. Door bijvoorbeeld in de codebase wat implementeren. Of door een proces te veranderen zodat deze data wel beschikbaar komt. Over het algemeen zou die dus als high worden ingeschat wanneer de effort om de data available te maken meer dan een week is. En dan de tweede dimensie is de technical expertise required. Low, wanneer alleen basic technical skills nodig zijn. Moderate, wanneer er technical skills nodig zijn die door de meeste software developers wel bezit worden. Dus kennis over bepaalde tooling. En high, wanneer geavanceerde technische skills nodig zijn. Dus echt specifieke in-depth knowledge over een specifieke tool. Is dat duidelijk voor jou? En is het ook duidelijk welke taken we zo meteen gaan doen?
P: Ja.
R: Oké, perfect. Laten we dan doorgaan naar het tweede gedeelte.
Dus allereerst wil ik jou vragen, we gaan al deze dingen bekijken binnen een specifieke context. Het is natuurlijk heel lastig om generiek iets te zeggen over deze measures. Omdat ze overal anders worden gemeten. Dus zou je mij eerst in een paar zinnen kunnen vertellen wat je doet bij <naam organisatie> en welke tooling je daarvoor gebruikt?
P: Ja. Ik ben lead solution software engineer. Dat is mijn officiële taak. Ik leid viertal developers binnen <naam organisatie>. En ontwikkel daarbij zelf ook. En we hebben een heel breed scala aan dingen die we doen. Die bij elkaar heel goed kunnen werken.
Maar ook heel goed te isoleren zijn van elkaar. Dus voor de context van deze vragen is het makkelijker als ik één stukje pak. Dus ik pak het stukje GraphQL Gateway. Waarbij wij een open source Docker container draaien. Op AWS serverless in de ECS met Fargate. En wij kunnen dan die Docker container configureren met YAML's om het te configureren. Of wij kunnen plug-ins maken in een soort verkapte rust variant. Om de functionaliteit wat meer uit te breiden. En dat is dan de router. En die zorgt ervoor dat requests naar één of meerdere subgraphs gaan die gemaakt zijn in lambdas. Met typescript code.

R: Check. Oké, duidelijk voor mij. In ieder geval de dimensie die wij met z'n tweeën zullen gaan benaderen is portability. Dus de draagbaarheid van het systeem. Dus in hoeverre kan je een systeem porten van één omgeving naar een andere. En de standaard die definieert portability als the degree of effectiveness and efficiency to which a product or system can be transferred from one hardware or software operational or usage environment to another. Dus tijdens de discussiegroepen zijn er een aantal metrics uitgekomen. En het is specifiek hierbij belangrijk dat we dus niet gaan kijken of je het eens bent met deze metrics. Of jij portability ook op die manier zou meten. Maar we gaan specifiek kijken naar waar zou je deze data kunnen verkrijgen.
P: Yes, check.
R: Oké. De eerste measure die men heeft gevonden.
Men zei dat portability te benaderen is aan de hand van lines of code die specifiek is aan je platform. Divided by de totale lines of code. En om dat nog verder iets uit te leggen, de platform specific lines of code zijn bijvoorbeeld interacties met een AWS SDK.
P: Ja.
R: Dus dan is de eerste vraag die erbij hoort, waar zou je deze data punten kunnen verzamelen?
P: Waar zou ik de platform specific lines of code kunnen vinden?Nou, de totale lines of code kunnen we makkelijk in github vinden. Platform specific lines of code moeten we nagaan wat precies platform specific lines of code is.
R: Exact, ja. Kan per context ook verschillen overigens?
P: Ja. Ik denk dat als je dingen zoals Docker meeneemt en die mee kan nemen naar verschillende hardware, waar het ook voor gebruikt is, dan hebben we geen platform specific lines of code. In het geval van de subgraphs hetzelfde. Dus dan zitten we eigenlijk te kijken naar de lines of code specifiek. Want wij gebruiken dan CDK.
R: Ja, exact.
P: Eigenlijk alleen de CDK code, want dat is platform specifiek. Dus dan moeten we dat om gaan bouwen naar een andere platform om het te deployen.
R: Exact, ja. Dus hoe zou je die platform specific lines of code kunnen gebruiken?
Dan is denk ik de volgende vraag, hoe zouden we die lines of code specifiek kunnen tellen? Hoe zouden we die kunnen identificeren? Dan gaan we natuurlijk over naar het puntje difficulty of obtaining data. Als dat iets wat gelijk gedaan kan worden, heb je daar een complexe oplossing voor nodig om die lines te identificeren? En dan daarna, welke technical expertise is required om deze measurement op te zetten?
P: Ik denk niet dat het heel moeilijk is om dat te doen. Ik denk niet dat het heel moeilijk is om het op te halen. Bij ons zit het allemaal specifiek in één stukje code, in één folder.
Dus alle lines of code vinden in één folder is niet zo lastig. Het kan zeker minder dan een dag duren. Ja, dan zouden we de difficulty of obtaining data onder low kunnen scharen.
Ik heb het specifiek nooit getest, maar als GitHub ook al die functionaliteit heeft,
van hoeveel lines of code zit er in deze folder, dan is het zelfs een paar minuten.
En dan vervolgens de technische expertise om dit op te zetten. Dus om deze measurement op te kunnen halen en dit te kunnen doen. Ik denk dat iedere developer wel regeltjes code moet kunnen testen. Tenminste, kunnen tellen.
R: Zou je die dan eerder onder low of onder moderate scharen?
P: Low. Ik vind git clonen of in git kijken een basis skill van een developer.
R: Oké, vervolgens hebben de deelnemers gezegd dat portability gemeten kan worden door de number of major platforms you can run on. Without modifications hebben ze erbij gezegd.
P: Wat is een major platform?

R: Ze bespraken het specifiek in de context van cloud platformen. Dus in hoeveel cloud platformen kan jouw oplossing draaien.
P: Ja, check. Zonder veranderingen. Dan moet je ook naar de code gaan kijken.
Dus wederom gewoon in je repository. Voor ons is het ook specifiek dat wij gebruiken cdk, dus je moet dingen gaan aanpassen. In dit geval bij jullie zou deze waarde gewoon vrij snel 1 zijn. Ik denk dat op het moment dat je infrastructures code gebruikt, dat je vrij snel zit aan 1. Ja, de difficulty of obtaining data and the technical expertise required is simpelweg door te kijken naar de code, vrij makkelijk te bepalen. Zit er iets van platform specific calls of resources in? Is het automatisch 1. Dus dat zou dan ook low zijn. Zeg ik dat goed?
R: Ja, ik heb geen idee wat mijn toetsen aan het doen.
P: Het is een beetje low slash moderate. Je moet natuurlijk wel weten wat is platform specific code en als je bijvoorbeeld niet weet wat cdk is en dat dat specifiek alleen op AWS geldt. Ja, dan is het lastig te achterhalen. Dat is dan voor de technical expertise required.
R: Maar het duurt geen dag?
P: Nee, laten we dat inderdaad gewoon op low zetten.
R: Oké, vervolgens noemde men the number of features functional on a platform divided by the number of features still working for a new platform. Dus je hebt een aantal functionaliteiten in je product zitten. Die kan je van tevoren definiëren van hij moet xyz kunnen. Vervolgens poort je het naar een nieuwe platform.
Werken al je functionaliteiten nog? Dat is eigenlijk wat men hiermee probeert te zeggen.
P: Dat is een mix van github repo, maar ook documentatie.
R: Dat had ik er inderdaad nog niet bij gezegd. Alleen het is inderdaad ook mogelijk dat je meerdere sources met elkaar moet combineren om een bepaalde measure te vinden.
We volgen weer het gestructureerde format. Werkt dit format? Is het duidelijk wat we aan het doen zijn?
P: Ja, zeker. Ik denk hetzelfde als bij de vorige, low low.
R: En waarom zijn ze hier low?
P: Omdat je gebruikt dezelfde points of information als de vorige. Je kijkt naar welke features bij de code horen. En dan vergelijk je dat. Vanuit hoeveel features je hebt is dat lastiger. En hoeveel documentatie je hebt kan het langer duren dan een dag. Maar het wordt er niet moeilijker van. Als je een beetje goed documenteert is het een stuk makkelijker.
R: Dat snap ik inderdaadDit komt uit een andere discussiegroep en daar stelden de deelnemers dat de applicatie is beter portable als je meer cloud agnostic services gebruikt. Hoe zou je deze mogelijk kunnen tellen?
P: En daar moet je ook goed definiëren wat valt onder een cloud agnostic service. Ja, dat weet ik ook niet. Bij mij is het allemaal specifiek allemaal AWS. Dus hoe je dan überhaupt een cloud agnostic service kan gebruiken zou ik niet weten.
R: Ja, als je het niet weet dan zetten we gewoon not applicable. Gewoon binnen jouw context niet gebruikt.
P: Dat kan natuurlijk ook. Anders zou het eigenlijk hetzelfde zijn. Je kijkt naar de code. Als je het kan porten dan kan je het porten.Als je infrastructures code gebruikt dan moet je een cloud agnostic infrastructures code gebruiken.
R: Nu komen we wellicht een aantal die voor jou ook not applicable kunnen zijn.
Maar misschien kun je uit ervaring wel wat over zeggen. Men heeft gesteld dat je applicatie beter portable is als een groter deel van je potentiële user base jouw applicatie kan draaien op hun device. Dus the number of supported devices divided by the number of potential users. Waar zou je deze informatie vandaan kunnen halen? Zou het lastig zijn om te verkrijgen? En is er technische expertise nodig voor de measurement?

P: Als je dit wilt doen dan moet je zoiets hebben als. Wat heet dat nou ook weer? Je kan Google Analytics gebruiken. Of goede user agent headers neerzetten. Die standaard gebruikt worden. En je moet een goed. Als je de number of potential users wel hebben dan moet je wel goede, ja user analyse doen. Dan heb ik service alleen om te kijken wat heb je nu? En ja. Hoeveel devices zijn er? Maar als het niet werkt dan werkt het niet.
R: Oké, kunnen we deze twee dan ook invullen? De difficulty of obtaining data and the technical expertise required.
P: Ja. Number of supported devices zou ik moderate zetten. Of difficult of obtaining data zou ik moderate zetten. Want je moet wel weten hoe analytics werkt en dat allemaal opzetten. Of in ieder geval zorgen dat. Die user agent goed wordt neergezet. Dan zou ik die ook wel moderate zetten.
R: Ik zie dat ze ook vaak inderdaad hand in hand gaan met elkaar. Hoe moeilijk is om te verkrijgen en hoe moeilijk het is om hem te implementeren.
R: Vervolgens komen er een aantal die heel erg op elkaar lijken. De number of supported browser versions. De number of supported mobile platforms. De number of supported OS versions. En de number of supported clouds. Deze werden ook allemaal binnen hetzelfde discussiegroep opgebracht. Hoe zou je kunnen aantonen hoeveel browserversions, hoeveel mobile platforms. Hoeveel OS versions en hoeveel clouds gesupport worden door jouw applicatie.
P: Ja, de user agent string. Omdat die ook echt gebruikt wordt.
R: En geldt dat voor al deze vier measures? Of geldt dat puur voor de browserversions?
P: Allemaal. Tenminste een of meerdere headers. Je hebt vaak een user agent, maar je kan ook <inaudible> erin zetten. Dan kan je zelf iets verzinnen. En dan is dat alleen maar het verkrijgen van de data. En dan moet je tenminste het tot je houden van de data.
Dan moet je natuurlijk ook nog queries gaan schrijven of een dashboard gaan maken.
Om al die data te aggregaten.
R: Exact ja, exact ja. Dus hier, als ik het zo hoor bij jou, zit er mogelijk wel een verschil van de difficulty of obtaining data and the technical expertise required.
P: Ik denk het wel. Ja. Want als je een string ergens toevoegen. Dat zit overal wel in.
Dus dat is wel makkelijk. Dat zou ik low zeggen. Technical expertise required is dan moderate
R: En geldt dit voor eigenlijk al deze verschillende smaakjes?
P: Ik zou zeggen van wel.
R: Dus de number of supported browserversions. De number of supported mobile platforms. De number of supported OAS versions. En de number of supported clouds.
P: Dan ga ik er nu al vanuit dat de clients, de user clients, goed gekoppeld zijn van de backend.
R: Dat is inderdaad een aanname die we kunnen maken. Hier lijkt deels herhaling te vallen voor wat we eerder hebben beschoken.
Iemand die zei, je kan puur portability meet aan de hand van de lines of code. Want des te meer lines of code je hebt, des te moeilijker het wordt om het over te dragen. Is wat deze participant opdracht.
P: Interessant. Dan moet je naar GitHub kijken. Volgens mij is dit gewoon simpelweg een subset van deze. Dus we kunnen ook dezelfde invullen.
R: Ja, zeker. In een andere discussiegroep werd opgebracht de tijd die het kost om te porten naar een andere platform. En wat dat andere platform is, is dan gespecificeerd in je requirements. Dus stel je wil je product porten van AWS naar Azure.
Je meet gewoon de tijd, dus in FTE, in uren, die het kost om dit te kunnen doen.

P: Ja, dan moet je echt een analyse gaan maken van je platform specifieke code.
Dus eigenlijk die andere measure die je hebt. En dan moet je ook een design gaan maken van hoe wil ik dat om gaan gooien. Dus dat is ook weer kijken naar je code. Dus de source zit dan in je GitHub repo. Ik denk wel dat dit moeilijk. Het is sowieso moeilijk om dit te gaan doen dus difficulty of obtaining data is high.
R: Omdat?
P: Je moet echt een heel ontwerp gaan maken. Wat zit er in een andere platform? Dan moet je dus. Het is sowieso al super complex om alle features van een andere platform te kennen en hoe je dat moet gaan gebruiken. Laat staan dat je dat al voor meerdere platformen hebt.
R: Dat kan ik begrijpen inderdaad.
P: Dus ik zou het bij technical expertise ook high zetten.
R: Ik zie dat ze vaak gewoon hand in hand met elkaar gaan. Vervolgens eigenlijk een beetje dezelfde dimensie die we net hebben bekeken. Alleen dan een andere waarde die je uitmeet in plaats van de tijd meet je het aantal acties. En acties zijn nog gespecificeerd op een zelfbepaalde granulariteitsniveau. Dus bijvoorbeeld is het. Bijvoorbeeld het moet herschrijven van een functie kan een actie zijn. Het runnen van een Github workflow kan een actie zijn et cetera. Dus je telt het aantal acties dat je moet doen. En hierbij stelden de deelnemers meer achtergrond te geven over deze measure. Dat meer acties zorgen voor een hogere cognitive load. En dat dat dan een applicatie minder portable zou maken.
P: Ik denk dat als je deze measure wil pakken. Dat je precies hetzelfde hebt als hiervoor.
Ik denk dat het wel wat makkelijker is dan kijken van hey hoe lang duurt dit. Want ik denk dat je inherent aan het kijken van hey hoe lang duurt dit. Dat je dan acties onderliggend hebt gemaakt. Dus dan kan je hem eerder afkappen. Maar ik denk niet dat het dan echt...
Misschien het idee is dan acties laten. Acties laten misschien een spoor achter.
Je kan zien wanneer een Github workflow is afgetrapt. Je kan het zien wanneer code is gemodificeerd. Waarbij tijd het lastig is om te kunnen zien hoeveel tijd iemand aan een taak heeft besteed. Maar dat is zeg maar altijd. Ik zit wel met deze measures van. Kijk als je wil weten of je deze measure hebt. En of je echt portable bent.
Dan ga ik wel ervan uit dat je dat van tevoren doet. En niet dat je daarna hebt. Want ja je gaat niet porten.
R: Nee dat is zeker de goede aanname hier.
P: Oké. Je gaat niet portability. Ik denk van ik wil heel portable zijn. Dus ik ga mijn code allemaal porten naar different platforms. Om te kijken hoe lang het duurt. Ik denk dat het makkelijker is dan de time. Maar ik denk niet dat het zodanig makkelijker is. Dat het dan opeens minder lang. Dat het dan een moderate wordt.
R: Exact ja.
P: Dus ze zouden me steeds gewoon als high high classificeren.
R: Dan hebben we nog het number of functions covered by documentation.
Divided by the total number of functions. Dus hoeveel van jouw functionaliteit staat gedocumenteerd?
P: Dan moet je naar je documentatie kijken. En naar je functies. Dus dat is vrij laag. Meestal is het al available. Maar dit zou available zijn want je hebt je functies. Je hebt in ieder geval je functies. En je hebt je documentatie. Als je ook nog docs hebt is het vrij makkelijk om te tellen. Ja en als je geen docs hebt dan is het nog makkelijker om te tellen. Want dan is het gewoon nul. Je hebt wel functies maar je hebt geen documentatie.
Dan heb je nul numbers of functions covered by documentation.

R: Vervolgens hebben we ook nog de allerlaatste onder portability. The number of code changes needed per function. Als je wilt porten naar een andere omgeving. Binnen een specifieke context.
P: Wat moet ik bijzien van een code change? Is dat een line die verandert?
R: Dat hebben de deelnemers niet uitgesproken in de discussiegroep. Dus dat is iets wat ik helaas niet erbij kan zeggen.
P: Dan moet je de documentatie kijken van je nieuwe platform. Tenminste kijken wat platform specifiek is aan je functie. Dan moet je de documentatie kijken van je nieuwe platform. Dan kijken wat er allemaal moet veranderen dan. Dus ja. Die difficulty of obtaining data is denk ik low. En dan wel een moderate to high technical expertise.
Omdat je wel iets van de target platform moet weten.
R: En dependent op hoeveel complexiteit er in de target platform zit.
Zou dit dan moderate of high zijn?
P: Ja. Als je hele basis functies hebt. En dan is het moderate want je hebt twee dingen nodig. Als je hele in-depth functies hebt. Waarbij heel veel van het nieuwe platform is. En dan moet je de documentatie kijken van je nieuwe platform.
R: Ja, Oké.Dat waren ze voor portability.
Ik zie dat we nog even hebben. Dus laten we dan ook nog doorgaan naar een tweede karakteristiek. En dat is compatibility. En compatibility is in de standaard. The degree of effectiveness and efficiency with which. A product or system can exchange information. with other products of systems. Soms werd die ook geconfused met een ander type van compatibiliteit. Dus in hoeverre rund het op verschillende platforms. Maar dit gaat specifiek over de informatie exchange. Het bestaat in verschillende platforms. De eerste die werd voorgesteld is simpelweg. Een boolean. Gestandardiseerde API documentatie, ja, nee. Misschien deels inherent aan de measure self-explanatory?
P: De measure self-explanatory, hoe bedoel je?
R: Ja, dat hij zichzelf uitlegt.
P: Heeft het documentatie, waar vind je dat? Ja, in de documentatie.
R: Ja, zeker.
P: Met documentatie is het ook soms dat als je gestandardiseerde API documentatie hebt, dan laat je dat vaak autogenereren en dan zet je dat in je GitHub repo, maar dan in dit geval de source docs zou dan waarschijnlijk ook in je GitHub repo zetten.
R: Ja, dat is wel een goede om er nog inderdaad bij te zetten. Oké, het verkrijgen van deze measure, is deze data meestal wel beschikbaar? Is deze data meestal wel beschikbaar? Is het soms niet beschikbaar?
P: Ja, het is wel beschikbaar. Dit is al lastig, hè? Het ligt eraan hoe je het standardiseert. Als je zegt van, ik maak een eigen standaard op Confluence en ik laat het niet automatisch genereren.Ja, het Confluence paginaatje invullen is vrij low.
Als je het echt laat automatisch genereren samen met Open API 3 bijvoorbeeld of Async API,dan heb je wel wat meer skill nodig.Maar puur als je naar een product gaat kijken en je moet over het product, want het gaat natuurlijk over het meten van de measure,
dus heeft die standaard API documentatie niet hoe die is opgezet, dan zou je dat in principe basically kunnen bekijken of door in de repo te kijken of door bijvoorbeeld in Confluence te kijken.
R: Nee, klopt.
P: Dus in dat geval is je *inaudible* vrij basis.
R: De volgende die we hebben is, iemand stelde dat je systeem is beter compatible als je functionaliteiten door middel van een API ontsloten worden. Dus hoeveel functionaliteiten

worden exposed through een API, gedeeld door de hoeveelheid functionaliteiten dat beschikbaar is.
P: Ik vind het wel een hele interessante measure. Dus ik zou het zelf heel anders aanpakken, maar oké. Ja, dan moet je naar je code kijken. En vooral naar de code waarin je je API opzet.
R: Is dit makkelijk te vinden in de code, wederom dezelfde vragen?
P: Ja. Het is makkelijk te vinden. Je kan het gewoon vinden in de Git Repo. Eigenlijk dingen die inherent zijn aan de source zijn makkelijk te meten. Als je nu de sources met elkaar moet combineren. Dus dan gaan we hier weer omlodig. Naar de technical expertise kan we modelleren. Ik vind het vrij basis, maar als je bij ons in de context zit. Is je een API gaat in AWS opzet, dan moet je wel weten hoe die linking gaat naar de functie. Dat is. Ik vind het vrij basis, maar ja, als je bij ons *inaudible* zit. Ja, er is wel tooling specifieke kennis nodig om dit te doen. Nodig om dit te doen? Ja, dat wel. Je hebt wel AWS kennis nodig, of in ieder geval bij ons CDK-kennis, om te weten van, hier in deze repo kan je zien van, dit is een API Gateway en deze functies zijn allemaal exposed.
R: Ja. Oké, cool.
P: Schat ik eens onder met moderate.
R: Oké. Wederom een soort gelijke. The number of API functions divided by the number of documented API functions. Dus hoeveel van jouw function in een tijd is gedocumenteerd? Waarbij men stelde dat als een functie gedocumenteerd is, maakt het je systeem meer compatible, omdat het makkelijker maakt voor een consumer om zich aan te sluiten op je platform.
P: We zouden het eerder onder usability pakken, dat maar ook goed.
R: Oké, vervolgens werd gesteld van, als nu een functie gedocumenteerd is, Vervolgens werd gesteld van als niet alle systemen die mogelijk jou zou connecten, dan ben je minder compatible.
P: Ja. Dus de hoeveelheid systeem in dat geconnect is, versus de wens.
Hoeveel systeem in wil *inaudible*. Dan moet je weer iets met headers doen denk ik, of source IP's van requests die binnenkomen zodat je kan identificeren hoeveel verschillende systemen connecten met dat van jou en je moet ook denk ik een proces in kaart brengen dat op het moment dat iemand wil connecten dat dat fout gaat.
R: Ja, wat bedoel je daarmee?
P: Ja, een soort van question shared-achtig dat wij hebben, van ik kan connecten maar ik wil connecten. Of je moet gewoon zelf zeggen zoveel goals, goal of number of connected system. Die moet je kunnen identificeren.
R: Oké, is dit lastig om op te zetten? Is dit al readily available?
P: Binnen onze context zou dat wel kunnen. Maar je hebt wel technische expertise nodig om die te aggregeren. Omdat je waarschijnlijk naar de input-output audit logs moet gaan kijken van je API gateway. En dan daar IP-adressen uit halen. Dat zijn niet de meest basis logs. Dus ik zou high zeggen omdat je best wel wat in-depth kennis moet hebben van de standard logging die ze hebben. En dan moet je ook nog naar CloudWatch of een vergelijkbare tool. Ja, dus er is in-depth knowledge over verschillende tools nodig om dit te kunnen doen.
R: Maar als je die kennis hebt kan je de data vrij snel available maken als ik het goed begrijp.

P: Ja. Bij ons in ieder geval is die data de standard. Ja, de data is er, alleen het aggregeren kost gewoon technisch effort.

R: Oké, dan is de volgende vraag, zijn systemen verbonden door middel van standaard authenticatie? Dus gebruik je bijvoorbeeld een OAUTH om je endpoints te beveiligen?
P: Is dit ook echt daadwerkelijk dezelfde technologie of is dit hetzelfde account?
R: Hoe bedoel je met hetzelfde account?
P: Nou, je kan OAuth gebruiken, maar dan wordt het niet per definitie makkelijker van om het te gaan gebruiken. Als je dan voor elke request een ander account gebruikt, dan is het niet makkelijker om het te gaan gebruiken. Dus als je dan voor elke request een ander account moet gaan ophalen, moet het graag gebruiken.
R: Niet gespecificeerd in die context, maar ik denk dat men het per account hier bedoelt. Dat is ook een aanname, dat is ook een kanttekening die ik hier dan bij ga maken, omdat het niet weer gespecificeerd is.
P: Ja, dus er wordt meer met SSO bedoeld, de technologie is hetzelfde.
R: Dat is gewoon de aanname, maar kan je ook hetzelfde account gebruiken.
P: Ja, dan moet je naar. Bij ons zou je dan naar de code moeten kijken, van de API Gateway. En dan kijken hoe de autorisatie daar geregeld is. Ja, en dan simpel inherent, net als bij die andere, is de data readily available.
R: Heb je nog technische kennis nodig om hierachter te kunnen komen of is het eigenlijk wel iets wat je met.
P: Je hebt wel technische kennis nodig. Specifiek om erachter te komen of dit zeg maar de SSO is, dan is het gestandaardiseerd of. Want je kan dezelfde technologie gebruiken om gewoon zelf iets aan te kloten.
R: Ja, exact, ja.
P: Maar je hebt wel technische kennis nodig om te kijken van, hey, wij gebruiken dan Cognito. Is dat geïntegreerd met het centrale <naam organisatie>. Dan is het de vraag, valt die onder moderate of valt die onder high te scharen? Hebben we echt in-depth knowledge nodig over deze technologie of. Ik zou high zeggen. We hebben ook Cognito nodig als, wat is nou echt de SSO binnen onze context? En hoe kan ik dat zien?
Zodat we wel domain-context-kennis als specifieke technologie-kennis nodig, vandaar wij.
R: Ja, Oké. Vervolgens zei men dat compatibility te meten is aan de hand van het data exchange format dat je gebruikt versus hoeveel je geaccepteerd zijn door de industrie.
En daarmee probeer je men te zeggen, stel dat je zelf een data format uitvindt en die stuur je uit naar mogelijke consumers. Dat maakt je als *inaudible* omdat het niet geaccepteerd is binnen de industrie.
P: En bedoelt het een beetje JSON versus XML of
R: Ja Beiden zou je dus kunnen scharen onder *inaudible* geaccepteerd, maar bijvoorbeeld als je zou kijken naar de medische industrie, is het natuurlijk een heel specifieke format om medische gegevens over te sturen. Als je dat zelf gaat uitvinden, dan zou het niet heel compatible zijn. Dat is eigenlijk wat men hier probeert te zeggen.
P: Ja. Check. Je kan in je GitHub repo vinden welke data exchanges je vindt en gebruikt. Dan moet je nog kijken wat zijn geaccepteerde data exchanges. Bij de industrie. Ik denk niet zozeer dat dat ergens opgeschreven staat. Als je bijvoorbeeld kijkt naar algemene dingen, kan je dat gewoon online vinden. Wat je het meestal gebruikt, dat is vaak JSON en dan mag je er zelf wat in gooien. Als je dan echt in een specifieke tak van een industrie kijkt, medische gegevens, hypotheken, dan moet je gewoon domein kennis hebben.
R: Ja, zeker.

P: Dat kan natuurlijk ook gewoon een source zijn of een niet specifiek gedocumenteerde

source. Nou, als je in die tak van sport zit, waarin je domein kennis nodig hebt, denk ik wel dat het redelijk makkelijk is om te achterhalen, moeten we iets doen? Bijvoorbeeld in het geval van hypotheken, zal je waarschijnlijk in je bedrijf wel een business analyst hebben of misschien een architect die dat weet, dan kan je gewoon rondvragen. Nou, afhankelijk van hoe actief mensen zijn op je vragen, duurt het een dag of een week of langer, maar ik zou dan zeggen, moderate, dat is niet readily available. Je moet er wel eventjes achteraan, maar het hoeft ook weer niet maanden te duren.
R: Nee, precies.
P: Ja, het is een vraag die niet heel veel facetten bevat. Het is een vrij simpel vraag.
R: Nee, precies.
P: Technisch kennis is denk ik wel, om de vraag te stellen heb je niet de technische kennis nodig, maar om het antwoord te geven wel. Is moderate to high. Ja, laten we dan high zeggen omdat het specifiek is voor een given domein.
R: Ja, oké. We hebben er nog een, twee, drie, vier, vijf. Dat gaat prima uitkomen met de tijd. Oké, vervolgens eigenlijk een eentje een beetje in hetzelfde straatje. Misschien voor hetzelfde en misschien ook niet, dat is aan jou. Hier was de idee om een extra penalty te geven voor de hoeveelheid dataformats die je gebruikt
die niet geaccepteerd zijn. Dus waar deze eigenlijk zegt van, hoe meer je de gebruik die geaccepteerd zijn, hoe beter. Deze zegt, hoe meer je de gebruik die niet geaccepteerd zijn, hoe slechter.
P: Ik denk dat het precies hetzelfde is waar je naar moet kijken en hoe moeilijk het is. Je gebruikt precies dezelfde data punten.
R: Ja, het zijn dezelfde data punten inderdaad, ja. Vervolgens zei iemand, je systeem is meer compatible als alle endpoints die je aanbiedt voldoen aan organisatie standaarden. Dus, hoeveel API endpoints zijn compliant aan organisatie standaarden? Hoe zouden, wederom dezelfde vraag.
P: Het begint een. Github repo. En company knowledge. Om deze data te verkrijgen.
R: We hebben eerder vastgesteld over het algemeen wanneer je iets simpelweg uit een repo kan plukken omdat het vrij eenvoudig is om op te halen. Maar what about the company knowledge hier?
P: Ik zou zeggen moderate.
R: Oké. Omdat?
P: Ja, over het algemeen is het vinden van company standards niet de meest makkelijke dingen om te vinden. Het staat altijd wel ergens weggewerkt en je moet het dan wel vinden die pagina. Op een of andere manier wordt dat niet heel erg gemarket. Maar je kan er wel achter komen door even een beetje te vragen. Technical expertise is denk ik low. Nee, ik denk wel moderate. Oké. Of misschien zelfs high. Aan de hand van welke compliance regels er zijn binnen je organisatie moet je wel weten of je er aan wel of niet aan voldoet.
Als je bijvoorbeeld kijkt naar je mag niet buiten je VPC handelen. Moet je wel weten wanneer je er buiten gaat.
R: Laten we hier dan inderdaad high assumem. Assume dat er inderdaad complexe organisatie standaarden zijn. Maar ik snap je punt van de afhankelijkheid daarvan wel, ja. Goed, deze is wellicht een beetje inherent aan wat we eerder hadden gehad. Daar hebben we gekeken naar the number of connected systems divided by the goal. En hier kijkt men naar the number of connected systems, waarbij de mensen, de deelnemers, argueden. Hoe meer systemen verbonden zijn, hoe meer compatible je wel zou zijn.

P: Ja, hetzelfde. Ik denk dat het vinden van de number of connected systems het moeilijkste is. De goal systems is misschien moeilijker op difficulty in obtaining data. Maar, ik denk dat expertise is vooral belangrijk.
R: Ja, oké. Vervolgens de ene laatste die we zullen bespreken is de successful request divided by the total number of requests. Waarin iemand zei van als men requests op jouw systemen afhuren die niet succesvol zijn. Weet of de client niet hoe hij met jouw system moet communiceren. Of jij weet niet hoe je client requests moet afvangen.
Dus dan zou je systeem vast niet heel compatible zijn. Dus wederom de vraag hoe zou je dit kunnen meten?
P: Ja, dan moet je request logging in gaan doen. Dus op je API Gateway kan je dit aanzetten bij ons. Het staat meestal aan. Dan moet je het gaan aggregeren. En bij ons hebben we dat er al standaard inzitten.
R: Omdat dat eigenlijk. Standaard inzitten door?
P: Nou, we hebben het wel zelf erin gezet, maar bij ons zit het er al in. Dus in AWS kan je heel makkelijk de errors eruit pikken. In API Gateway zelf van AWS zit het volgens mij zelfs al een van de standaard metrics erin. Die je dan, als je hem enabled, gewoon in cloud kunt zien. Dus binnen onze context is het heel makkelijk. En ik denk dat het vrij...
Een succesrate of een fail rate bij bijna elke standaard API Gateway wel erin zit. Of erin gezet kan worden.
R: Ja, oké. Dus assume dat het nog niet eerder is gedaan. Hoe moeilijk zou het zijn om data te verkrijgen? Hoeveel technische expertise heb je nodig om dit op te zetten?
P: Ik zou zeggen difficulty of obtaining data low.Ja, want loggen of iets goed is gegaan of fout is vrij basis. Technical expertise required is dan ook low. Iets loggen is vrij makkelijk. Het aggregreren is niet heel complex als je zelf je logging doet.
R: Ja, oké. Dan de allerlaatste die we zullen bespreken is number of packages built divided by the number of packages that can be shared with different techniques. Iemand zei je systeem is beter compatible als je constructs uitbrengt die ook binnen andere technieken te gebruiken zijn. Hoe meetbaar is dit? Zou dit zijn binnen jullie context?
P: Dan moet je kijken naar weer GitHub repo. En dan specifiek kijken naar wat haal je binnen met andere packages. Dus dat is vrij makkelijk. Je kan vrij makkelijk achterhalen. Ja, de data is readily available want het staat gewoon al in je repo. Ja, je kan in je packages en kijken welke dingen je binnen haalt en welke constructs je er vanaf allemaal gebruikt. En dan moet je eigenlijk alleen nog even kijken of andere mensen hem ook gebruiken. Ja, in GitHub zelf kan je daar best wel makkelijk in. Want je kan je package namen gewoon in GitHub zetten en dan zie je welke repos je ook allemaal gebruikt en dan filter je er wat uit.
Ja. Dus het is ook vrij makkelijk om het te doen want ja, het zit een beetje op hetzelfde niveau als googlen. Existing tooling. GitHub search. Je gooit je package naam erin en je krijgt de repositories eruit.
R: Ja, check. Oké. Thanks. Dat waren ze voor nu. Dan wil ik je super erg bedanken en dan ga ik mijn scherm in ieder geval stoppen met delen en dan ga ik de opname stoppen.