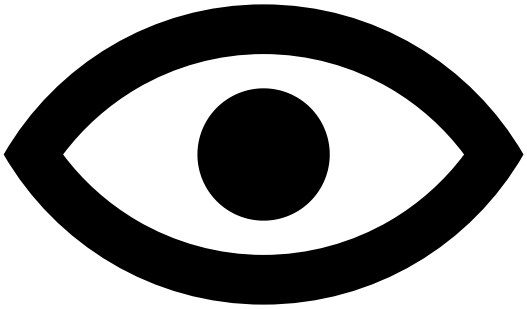

27,1K

Channel

Tech

Achtergrond - 10 mei 2017 - 12:00

Microservices: wanneer wel, wanneer niet?

Microservices, misschien wel het modewoord in de software-ontwikkeling van dit moment. Grote bedrijven als Amazon en Netflix zweren erbij, en er is zeer veel aandacht voor in de techcommunities. Veel bedrijven met een online propositie worden daarom geconfronteerd met de vraag of zij hier ook niet iets mee moeten. Deze vraag kan bijvoorbeeld ontstaan doordat het development team dit op de agenda zet, of dat een vendor met een nieuwe oplossing komt. Hoe zinvol is het gebruik van microservices?

Wat zijn microservices?

Een software-applicatie is onderhevig aan een architectuur: regels die op grote lijnen specificeren hoe de software technisch gestructureerd is. Deze architectuur geeft onder andere richting hoe nieuwe functionaliteiten geprogrammeerd moeten worden, maar bepaalt ook wat er wel en niet mogelijk is binnen de software, en tegen welke prijs.

Er bestaan verschillende architectuurstijlen die allen voor- en nadelen hebben. Afhankelijk van de context waarin de software leeft past een bepaalde stijl goed of minder goed. Een microservices architectuur is zo'n architectuurstijl. Helaas bestaat er geen eenduidige definitie hoe deze architectuurstijl precies in elkaar steekt, maar de essentie van het verhaal is vaak dat één grote applicatie (een zogenaamde monoliet) wordt opgeknipt in meerdere, kleinere applicaties (de microservices). Al deze losse applicaties praten dan met elkaar om uiteindelijk functioneel hetzelfde te doen als de monolitische applicatie. Voor de eindgebruikers verandert er (als het goed wordt uitgevoerd) dus niets.

Waarom microservices?

Als er functioneel niets verandert, waarom zou je dit dan doen? Hier worden een aantal argumenten voor genoemd, waarvan de meest voorkomende zijn:

1. Betere performance
Amazon heeft ons al lang geleden geleerd dat een sneller systeem meer verkoopt. Betere performance van de software zorgt dus direct voor business value. De theorie is dat je met een microservices-architectuur beter inzicht hebt welk deel van je systeem niet performed, en dan de individuele microservice kunt optimaliseren of opschalen.

2. Beter onderhoudbare software
Dit betekent concreet dat software sneller doorontwikkeld kan worden, en dat aanpassingen en nieuwe features dus sneller bij de eindgebruiker terecht komen. De gedachtengang hier is dat kleine applicaties makkelijker te begrijpen zijn, en daarmee makkelijker aanpasbaar zijn en er minder bugs geïntroduceerd worden.

3. Duurzame velocity
Naarmate een softwareproject groter wordt, en er meer mensen aan werken, groeit de benodigde communicatie tussen mensen in het project. Dit kan zo ernstig worden dat developers en product managers alleen nog maar aan het praten zijn, maar daadwerkelijk niets meer opleveren. Bij een microservices-architectuur kun je teams rond een microservice bouwen. Omdat de services per definitie klein zijn vermijd je dit probleem van de verlammende communicatie-overhead, en kan de velocity behouden worden.

4. Ontwikkelen in meerdere talen
Doordat alle applicaties hun eigen codebase hebben, en alleen via standaardprotocollen met elkaar praten is het mogelijk dat elke microservice in een andere taal geschreven wordt. Dit biedt de mogelijkheid *the right tool for the right job* te kiezen, en biedt je ook de mogelijkheid om uit een grotere pool developers te vissen.

5. Herbruikbaarheid
Dit is wat Amazon heeft gedaan met hun AWS. Door hun IT-infrastructuur in services op te knippen, was het mogelijk om deze vervolgens ook als diensten aan te bieden voor externe klanten. Hierdoor kon én de software én de hardware herbruikt worden. Nu zijn de meeste bedrijven natuurlijk geen Amazon, maar zelfs als je het niet extern kunt aanbieden kun je het in ieder geval voor intern gebruik aanbieden.

In de praktijk worden er nog meer voordelen genoemd, maar dit zijn wel de meest voorkomende.

Waarom geen microservices?

Als je bovenstaande leest, dan krijg je het idee dat je eigenlijk wel gek moet zijn als je *geen* microservices-architectuur gebruikt. Alle voordelen lijken immers behoorlijke business value op te leveren. Er zijn echter ook behoorlijke nadelen, en daarnaast zijn niet alle bovengenoemde redenen zonder meer legitiem.

Het grote probleem met een microservices-architectuur is dat dit een vorm van een gedistribueerd systeem oplevert, en het is een welbekend onderzoeksresultaat dat gedistribueerde systemen juist een stuk complexer zijn dan niet-gedistribueerde systemen. Hier zijn meerdere oorzaken voor, en ze zijn vrij technisch, maar in de basis komt het omdat er veel meer bewegende onderdelen zijn.

In het geval van een microservices-architectuur zijn deze bewegende delen de microservices, maar ook nadere infrastructuur als messaging systemen of databases. Dit maakt dat er op meer plekken iets fout kan gaan, en dat het ook moeilijker is om het overzicht te hebben van wat er gebeurt. Het is waar dat de individuele applicaties minder complex worden (voordeel #2), maar het systeem als geheel is juist complexer geworden.

Wat het geheel nog problematischer maakt is dat developers die traditioneel gezien monolitische applicaties hebben gebouwd, niet bekend zijn met de principes van gedistribueerde systemen en de problemen die daarbij komen kijken. Zij zetten het systeem daarom op alsof het een niet-gedistribueerd systeem is (mede mogelijk gemaakt door tooling die ook z'n uiterste best doen om de problemen te negeren). Voeg daar aan toe dat alles goed lijkt te werken op de lokale ontwikkelomgevingen van developers, en je hebt een recept voor een moeizaam ontwikkeltraject met veel technische problemen en juist een lage velocity.

Een ander probleem is dat het heel moeilijk is om de grenzen van microservices te kiezen: welke microservice is waar verantwoordelijk voor? Waar je misschien zal verwachten dat dit een technische beslissing is, is dit juist een business beslissing. Microservices dienen de contouren van het bedrijf te volgen, bijvoorbeeld op het gebied van verantwoordelijkheidsgebieden of operatie.

Dit betekent dat een alignment van business en development een vereiste is (als development al een microservices architectuur heeft, maar daar niet over overlegd heeft met de business, dan is dat zeker een red flag). Daarnaast moet de organisatie al enigszins stabiel zijn zodat het mogelijk is zinvolle en duurzame service grenzen te definiëren. Als deze voorwaarden niet aanwezig zijn, dan is het dus geen goed idee om er mee te beginnen.

Wanneer microservices?

Hoe maak je nu de afweging? Er kan duidelijk veel business value zitten in het gebruiken van microservices, maar aan de andere kant hangt er ook een groot risico aan. In mijn ogen ligt het belangrijkste voordeel aan microservices in punt #3 (duurzame velocity met grotere teams): het biedt een methode om je development capaciteit op te schalen. Software development teams schalen namelijk niet. Afhankelijk van de specifieke situatie, zal dit bij een teamgrootte zijn van tussen de 5 en 10 personen. Onder die grootte kunnen teams prima in dezelfde code base werken, en is het de overhead dus niet waard. Daarnaast zijn alle andere voordelen niet inherent aan microservices, en dus ook op een andere manier te behalen (hoewel het soms wel *iets* makkelijker wordt met microservices).

Als je eenmaal die grootte bereikt hebt, zorg dan dat je mensen aan boord hebben met training op het gebied van het bouwen van gedistribueerde systemen (een goede training is Udi Dahan's Advanced Distributed Systems Design), en zorg ervoor dat er korte communicatielijnen zijn tussen development en business.

Conclusie

Het is onweerlegbaar dat grote bedrijven succes hebben met microservices. Het belangrijke woord in die zin is *grote*. Ik zie om mij heen veel startups en kleinere bedrijven die aan de slag gaan met microservices, terwijl ik eigenlijk niet zie hoe dat de business ondersteunt. In tegenstelling, ik zie juist dat het de business schaadt omdat heel veel development effort in de infrastructurele kant gaat zitten, in plaats van in het oplossen van de business problemen. Ik kan me niet aan de indruk onttrekken dat hier meegegaan wordt met de hype train, zonder de nadelen goed af te wegen. Wat overigens ook wel weer te begrijpen is, want vanuit een technisch perspectief is het best wel interessant. Laat dit echter de strategie van jouw bedrijf niet bepalen, want dat is wat het kiezen voor deze architectuur is: een strategische beslissing.

Dit bericht is 46 keer gedeeld