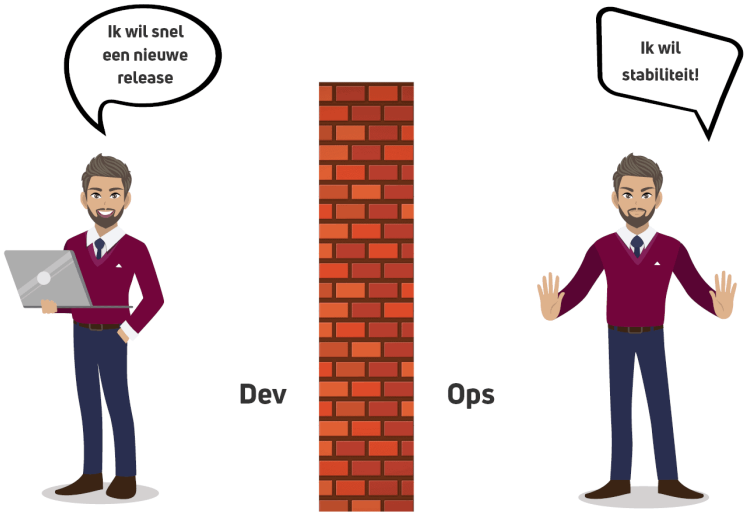
Traditioneel zijn softwareontwikkeling en systeembeheer ondergebracht in twee verschillende afdelingen, met ieder hun eigen belang. Ontwikkelteams worden afgerekend op hun productiviteit, met andere woorden: hoe snel kan nieuwe functionaliteit worden opgeleverd? Voor de beheerders geldt een heel andere (tegengestelde) performance indicator, namelijk een zo hoog mogelijke



beschikbaarheid van het systeem met een zo laag mogelijk security risico. Deze tegenstelling in belangen leidt tot conflicten bij de overdracht van ontwikkeling naar beheer. Het statement “Ik moet ’s nachts mijn bed uit omdat JULLIE niet goed hebben getest”, zal in ieder bedrijf wel eens in de chat worden gezet. Gevolgd door: “JULLIE remmen de business af”, als antwoord van het ontwikkelteam. Het type persoon in een ontwikkelteam is gemiddeld genomen anders dan in het beheerteam. Dit versterkt de polarisatie en maakt de communicatie tussen de teams lastiger. In het verleden is geprobeerd de samenwerking tussen Development (Dev) en Operations (Ops) te optimaliseren met steeds verdergaande processen en procedures overgewaaid vanuit het Verenigd Koninkrijk (denk aan: stage containment, Prince2, Change Advisory Boards en ITIL). Het is duidelijk dat dit niet het gewenste resultaat heeft opgeleverd en moderne IT afdelingen richten zich nu op kort-cyclisch werken en een andere verdeling van de verantwoordelijkheden.

Het vakgebied van systeembeheer is sterk in beweging met de komst van cloud diensten. In het verleden bestond het grootste gedeelte van de werkzaamheden uit manuele activiteiten. Denk aan het inrichten van een server, het beheer van harde schijven en het installeren van nieuwe applicaties. Een deel van de werkzaamheden wordt nu ‘as a service’ uitbesteed aan Amazon, Microsoft of Google. Het vervangen van een kapotte harde schijf is daarmee een anekdote uit het verleden. Maar de impact op het vakgebied is nog veel groter, doordat infrastructuur wordt beheerd via scripts (bijvoorbeeld Terraform, Ansible of Kubernetes). Alle handmatige handelingen worden vervangen door geautomatiseerde, van te voren vastgelegde instructies. Met Ikde komst van serverless computing, waarbij het beheer volledig is uitbesteed aan een cloud provider en de ‘server’ pas wordt geactiveerd op het moment dat een request wordt gedaan (bijvoorbeeld AWS Lambda), is traditioneel

systeembeheer niet eens meer mogelijk.

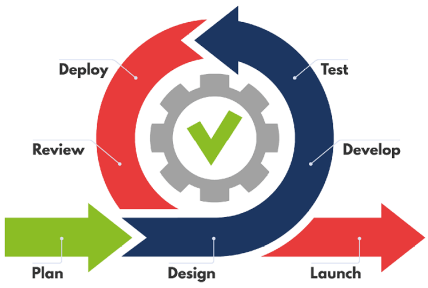
De grenzen tussen applicatie, middleware en infrastructuur zijn aan het verwateren. De verantwoordelijkheid voor database engines, cache oplossingen, monitor/alerting, load balancers, rate limiting, etc. is steeds lastiger toe te wijzen aan Development of Operations omdat beide teams hierbij een rol spelen.

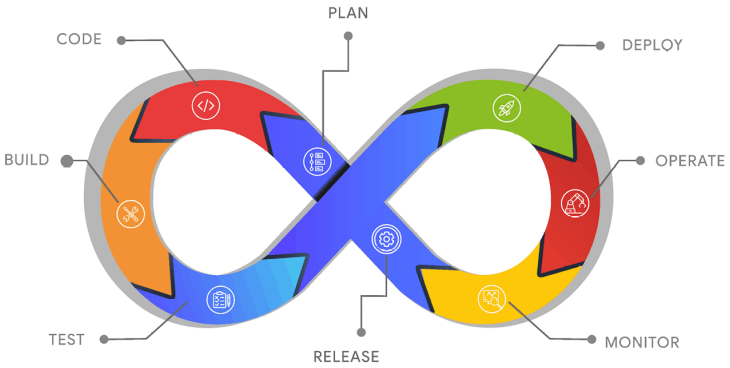
Deze drie trends hebben geleid tot de populariteit van DevOps; het samenvoegen van Development en Operations tot één team.

Vervangt DevOps Agile?

Agile is een manier van samenwerken die veel gebruikt wordt binnen ontwikkelteams. Voor de meeste organisaties wordt Agile ingevuld door de Scrum of Kanban methode, ondersteund door visualisatie door middel van ‘geeltjes’ of met Jira boards. Er bestaan tientallen vergelijkbare boeken, trainingen en coaches op het gebied van Agile werken. De rollen (bijvoorbeeld Product Owner) en het Agile manifesto zijn algemeen geaccepteerde begrippen.

Voor DevOps verschilt de implementatie in organisaties sterk. Voor sommigen is het een verandering in mindset, een cultuurverschuiving, terwijl voor anderen de organisatiestructuur en verantwoordelijkheden wezenlijk veranderen. In onze optiek zijn de fundamentele voordelen van DevOps alleen te bereiken door een combinatie van cultuur- en organisatieverandering met Agile en automation als voorwaarde.





Agile vs DevOps

Er is geen kant-en-klaar recept voor DevOps, maar er zijn al wel publicaties en voorbeelden uit de praktijk. Veel ontwikkelteams werken momenteel al Agile, waarin nieuwe functionaliteit in kort-cyclische iteraties (bijvoorbeeld tweewekelijks) wordt opgeleverd. Deze releases moeten niet alleen in een testomgeving worden beoordeeld door de opdrachtgevers/gebruikers, maar worden ook direct in productie gebruikt en beoordeeld door

de markt. De achterliggende gedachte is om zo snel mogelijk feedback te krijgen en verbeteringen door te voeren in volgende iteraties. In de DevOps gedachte behelst deze feedback zowel de functionele aspecten, als ook de non-functional aspecten zoals beschikbaarheid, performance en security. Door het samenvoegen van de verantwoordelijkheden van Dev en Ops teams wordt het DevOps team verantwoordelijk voor alle aspecten van het product. Het DevOps team moet continu een afweging maken tussen nieuwe features en de stabiliteit van het product in productie. Het investeren in nieuwe tests of betere monitoring is soms belangrijker dan het toevoegen van een extra feature.

Als je als DevOps teamlid uit je bed wordt gebeld omdat het systeem plat ligt als gevolg van de laatste release, dan kun je dit de volgende ochtend in de stand-up van je eigen team bespreken. Het DevOps team kan de oplossing dan hoog op het bord zetten, waardoor de oplossing ’s avonds in productie kan worden gezet. Hierdoor hoef je de nacht erna er niet meer uit.

In hoeverre is DevOps nu anders dan Agile? Agile werken gaat over de manier van samenwerken en het proces (het ‘hoe’) en niet over de scope van het product (het ‘wat’). Agile werken binnen ontwikkelteams is voornamelijk gericht op het toevoegen van nieuwe business functionaliteit. De bekende story points worden vaak alleen toegekend aan toegevoegde waarde gezien vanuit de ogen van de klant. In een DevOps team spelen de non-functional requirements een meer prominente rol. Denk hierbij aan het uitvoeren van security of performance tests, het analyseren van logfiles uit productie, het optimaliseren van de build en deployment processen en het beoordelen van het resource gebruik. De toepassing van Agile is ook evident: kort-cyclisch werken, bijsturen op basis van feedback, ‘doen’ in plaats van discussiëren en analyseren. Concluderend kunnen we stellen dat DevOps de scope van een team vergroot, maar weinig verandert aan de Agile werkwijze.

Binnen DevOps is continue samenwerking en feedback erg belangrijk. Het succes van DevOps vraagt om een samenwerkingscultuur die gericht is op continue verbetering. ‘Continue’ of ‘continuous’ is sowieso een term die je heel vaak tegenkomt in het kader van DevOps. Continuous integration, continuous delivery, maar ook: continu testen, continue monitoring, continue feedback en continue verbetering. Het uiteindelijke doel van deze continue samenwerking tussen alle belanghebbenden is de time-to-market te versnellen, de kwaliteit van de software te verhogen, sneller te kunnen inspelen op veranderingen in de markt en (beter) te voldoen aan de steeds veranderende behoeften van klanten. Nieuwe trends zoals “latest greatest” als opvolger van het werken met conventionele release branches & patch releases en “canary deployments” als middel om de “fail fast and early” strategie te implementeren, zijn mede tot stand gekomen door de samenwerking tussen Dev en Ops.

Het belang van (geautomatiseerd) testen binnen DevOps

Stel dat elke wijziging na een peer-review en het doorlopen van de geautomatiseerde tests direct gedeployed wordt op de productieomgeving (CI/CD), dan betekent dat voor een gemiddeld ontwikkelteam meerdere releases per dag. Handmatige activiteiten zijn dan eigenlijk niet meer mogelijk. Stel dat jouw systeem een beschikbaarheid moet hebben van 99,99%. Dat geeft het DevOps team per maand ruim vier minuten de tijd om een incident op te lossen. Ook hier geldt dat het bijna niet meer mogelijk is om dit handmatig op te lossen. Deze trend (geautomatiseerde tests en zelfherstellende systemen) is jaren geleden al ingezet met Agile en past goed in de DevOps gedachte. Dit is echter alleen mogelijk als alle stakeholders kunnen vertrouwen op het testframework van het DevOps team. Testen speelt dus een cruciale rol binnen DevOps.

Automatisering is het toverwoord, maar zijn alle tests te automatiseren? Ondanks het feit dat teams streven naar een zo hoog mogelijke test coverage van hun software, blijft testen gebaseerd op een risico-afweging. Alles testen is een utopie. De variëteit van laptops, tablets en mobiele telefoons is dermate groot dat niet iedere combinatie van hardware, browser en operating systeem te testen valt. Ook de visuele beoordeling van een user interface is lastig te valideren in een geautomatiseerde test.

In een DevOps cultuur is iedereen verantwoordelijk voor de kwaliteit en stabiliteit, en daarmee voor het succes van de organisatie. Testen en Quality Assurance (QA) leveren hieraan een belangrijke bijdrage.

De tester in een DevOps team

DevOps betekent werken in een cross-functional team waarin alle disciplines vertegenwoordigd zijn en waarin men gezamenlijk verantwoordelijk is voor de continue ontwikkeling en het beheer van software. Je bent als team zelf eigenaar van het product, en dus ben je ook verantwoordelijk voor de eventuele incidenten.

De grenzen tussen de verantwoordelijkheden van een ontwikkelaar en een tester vervagen. Het ontwikkelen van geautomatiseerde testscripts en het beheren van testdata is niet langer het domein van alleen de tester. In de meeste teams worden deze activiteiten door zowel de tester als de ontwikkelaar uitgevoerd. De rol die zij spelen is even belangrijk, echter de regie over de testactiviteiten (gebaseerd op de risico-analyse) is de verantwoordelijkheid van de tester. Voor een traditionele tester en software ontwikkelaar betekent de overgang naar DevOps een grote verandering. Van beide kanten vraagt dit om aanpassing. Het betekent onder andere een verschuiving in de vaardigheden, methoden, tools, timing en de aanpak van de tester.

Ervaren DevOps teams zullen bij aanvang van nieuwe ontwikkelingen al direct nadenken over de teststrategie en de benodigde tools. De ontwikkeling van de tests zal parallel lopen met de softwareontwikkeling, of het team gaat nog een stapje verder en werkt volgens het Behaviour Driven Development principe.

De gereedschapskist van de tester is de laatste jaren gevuld met vele nieuwe (open source) tools. In nauwe samenwerking met de ontwikkelaars zal de tester bestaande tests aanvullen of nieuwe frameworks toevoegen. Het gaat hierbij niet alleen om de functionele tests, bijvoorbeeld ontwikkeld in Cucumber, Behat of Selenium, maar ook om performance (bijvoorbeeld Gatling) en security (zoals Netsparker of Tenable io) tests. Fans van het ‘Shift Left’ concept zullen dit beamen. De achterliggende gedachte van dit concept is het zo vroeg mogelijk ontdekken van fouten, omdat de kosten van een fout steeds hoger worden naarmate deze fout later in de lifecycle wordt ontdekt.

Meer en meer varen teams op dashboards als kompas. Bij het ontwikkelen van metrieken op dergelijke dashboards speelt de tester een belangrijke rol. Welke tests zijn cruciaal om op dagelijkse basis te monitoren? Op basis van het gebruik van de functionaliteit geeft het dashboard ook inzicht in de werking van het systeem. Denk bijvoorbeeld aan de metriek ‘gefaalde betaling’ versus ‘succesvolle betaling’ in een webshop.

T-shaped

Naast multidisciplinair is het belangrijk dat de DevOps teamleden T-shaped zijn. Dit geldt ook voor de tester in dit team. Uiteraard is hij specialist op het gebied van testen en neemt hij hierin de lead, maar daarnaast wordt verwacht dat hij bijvoorbeeld ook een businessanalyse kan uitvoeren, de operatie kan helpen met incidenten in productie of beschikt over development skills. De tester in het DevOps team is dus van alle markten thuis, is technisch

onderlegd en voelt zich thuis binnen de diverse disciplines. En misschien is er nóg wel een belangrijke rol voor een DevOps tester weggelegd: die van coach die zijn eigen testkennis en -vaardigheid deelt om zodoende de andere teamleden te helpen om beter te testen of ontwikkelaars te leren hoe ze betere tests kunnen schrijven.

Ontwikkelaars hebben minder focus op kwaliteitsproblemen en meestal vinden ze testen ook geen leuk werk. Er zijn zeker ook developers die goed kunnen testen, maar waar zij vanuit hun oplossingsgerichte mindset vooral op zoek zijn naar oplossingen, heeft een tester ‘van nature’ een meer risicogedreven mindset en kan hij inzicht (en overzicht) verschaffen in de werkelijke status van de nieuwe functionaliteit. De kritische blik en de natuurlijke drang van de tester om te zoeken naar problemen en risico’s stellen de overige teamleden in staat betere producten te maken en slimmere beslissingen te nemen. De tester van vandaag is een allrounder en helpt net zo makkelijk bij een deployment als bij een GDPR audit.

De tester speelt vaak ook een rol bij het beoordelen van de feedback (vanuit gebruikersacceptatie tests, maar ook vanuit monitoring systemen). Deze feedback geeft de mogelijkheid om de testaanpak continu te verbeteren. Een deel van de tijd zou ook moeten besteed aan het volgen van de nieuwste trends op het gebied van tooling en risico’s. De ontwikkelingen blijven elkaar in hoog tempo opvolgen en een technical debt kan ook ontstaan in de testuitvoering. Recente voorbeelden met hoge impact binnen het testdomein zijn de opkomst van IoT en AI.

Conclusie

DevOps lijkt het passende antwoord op de vraag uit de markt om IT systemen te beheren, waarbij het mogelijk is om sneller en goedkoper nieuwe functionaliteit toe te voegen. DevOps is alleen te bereiken door een combinatie van cultuur- en organisatieverandering met Agile en automation als voorwaarde. Cruciaal is het vervangen van handmatige acties door geautomatiseerde procedures. Dit zie je terug in de ontwikkelstraat (CI/CD), het beheer van de infrastructuur (IaC) maar zeker ook in het testdomein. Denk hierbij aan geautomatiseerde tests, profiling tools en security assessments.

In een DevOps cultuur is iedereen verantwoordelijk voor de kwaliteit en stabiliteit, dus werkt het team echt samen aan betrouwbare systemen die snel kunnen worden aangepast. De tester acteert als regisseur van de testactiviteiten en werkt in de uitvoering samen met de ontwikkelaars. Juist bij DevOps gaan Testen en QA als verbindende factor tussen ontwikkeling en operations een meer strategische rol spelen.