Continuous Delivery in de praktijk

Continuous Delivery is een manier om software snel en herhaalbaar in productie te brengen. Dit is te bereiken door het volledige software voortbrengingsproces te standaardiseren en te automatiseren. Na een korte inleiding behandelt dit artikel drie cases vanuit organisaties die Continuous Delivery in de praktijk hebben toegepast.

Eelco Meuter is Senior Java Architect bij Ordina

Walter van Iterson is Senior Java Architect bij Ordina

Het voortbrengingsproces wordt het best weergegeven in **figuur 1**. De stappen die je software tot aan productie doorloopt, wordt een deployment pipeline genoemd. Door deze stappen te automatiseren, wordt software sneller opgeleverd en worden mogelijke bugs sneller en consistenter gevonden. De kans dat de oplossing van een probleem tot een nieuw probleem leidt, wordt op deze manier gereduceerd. Het automatiseren van het proces vereist standaardisatie, wat weer tot gevolg heeft dat software een hogere kwaliteit heeft.

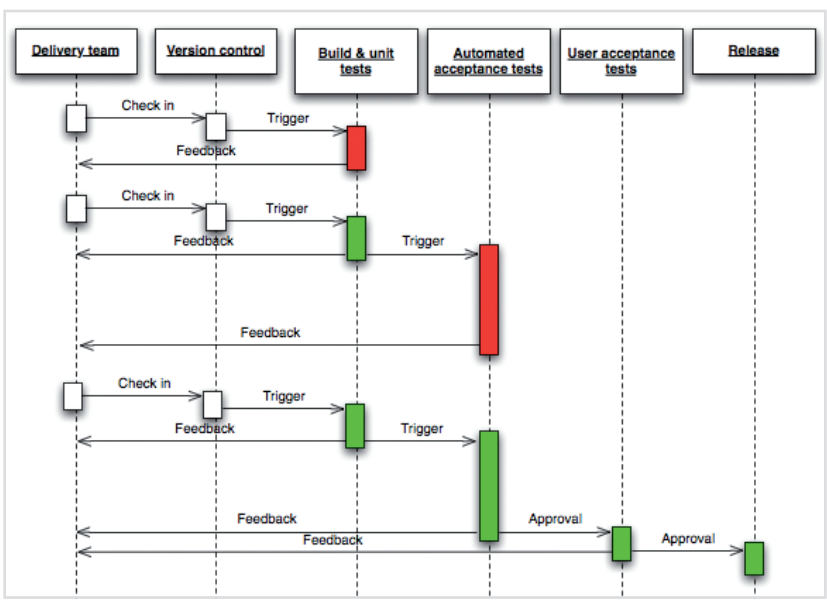
Een gestructureerde aanpak

De wens voor zo'n deployment pipeline komen wij vaak tegen in de praktijk. Het software voortbrengingsproces is een veel omvattend begrip. In veel organisaties zijn veel verschillende afdelingen en mensen betrokken bij het tot stand brengen van software. Een wijziging in een bestaand proces is een uitdaging en je zult dikwijls obstakels tegenkomen. Het is daarom goed om eerst te inventariseren wat je al doet aan Continuous Delivery. Daarna kan je realistische doelen stellen ten aanzien van wat je graag wilt bereiken.

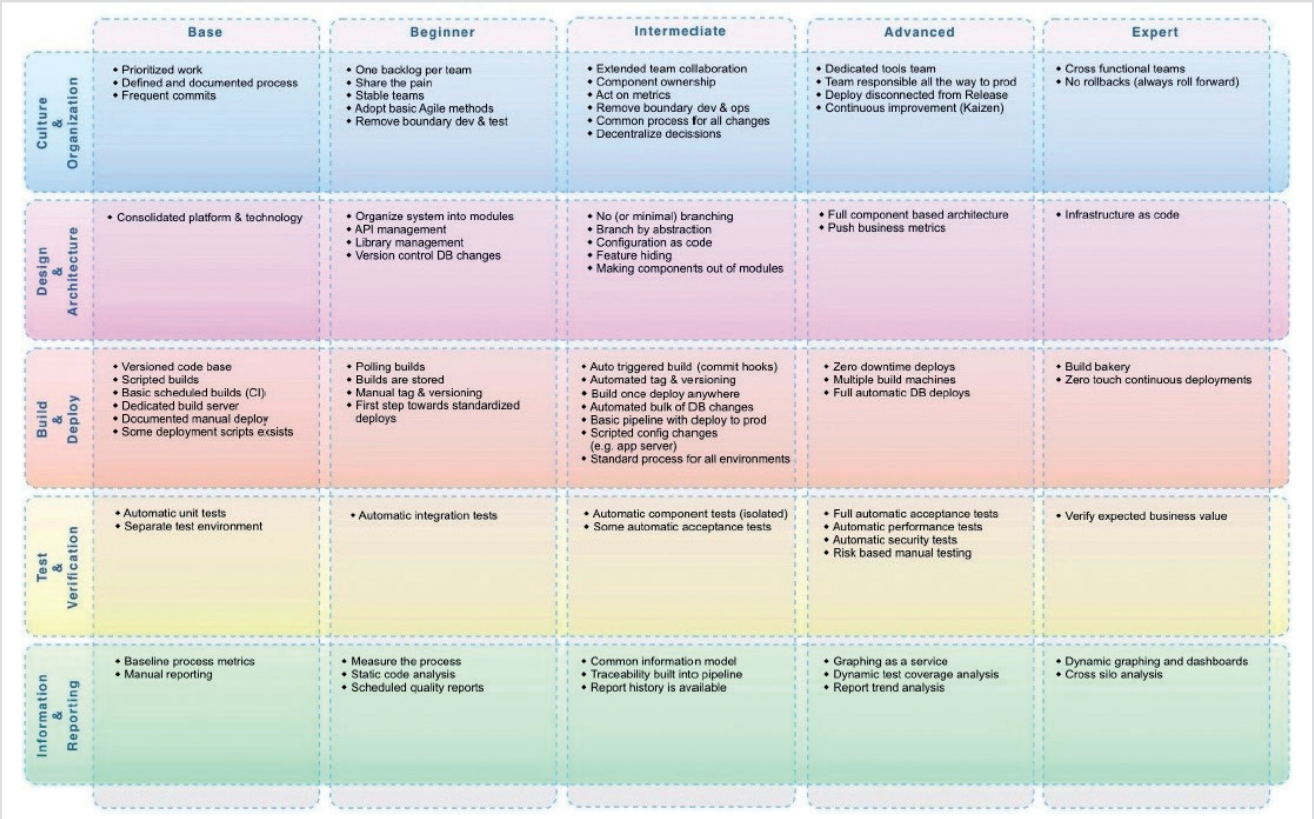
Een gestructureerde aanpak helpt enorm in de communicatie en het stellen van realistische doelen. Een veelgebruikt model is het 'Continuous Delivery Maturity Model' . Hierin worden vijf pijlers benoemd waarop je voortbrengingsproces steunt. Elke pijler heeft een vijftal niveaus van volwassenheid. Deze niveaus variëren van basis tot expert. Op deze manier ontstaat een matrix waarbij het meteen duidelijk is waar je als organisatie staat en waar je naar toe wilt.

Er valt veel te zeggen over een maturity model. Elk punt in de matrix kan op zichzelf al het onderwerp zijn voor een flinke discussie. Het is dan ook goed om je te realiseren dat de gepresenteerde punten een indicatie zijn waar je concreet aan moet denken als je over volwassenheid praat in de context van Continuous Delivery. Je bent geheel vrij om deze punten aan te passen, mits je die aanpassing staaft aan de huidige standaarden in de industrie. Het is niet zinvol om voor elke pijler het niveau expert te ambiëren, want het is belangrijk om realistisch te zijn. De software moet immers waarde toevoegen aan de bedrijfsvoering. Het behalen van een expert niveau is vaak zeer kostbaar, wat niet opweegt tegen de waarde die de software brengt voor het bedrijf.

Als je ervoor kiest om een maturity model te gebruiken, dan begin je met twee modellen. Het



Figuur 1: Deployment pipeline



Figuur 2: Een Continuous Delivery Maturity Model

eerste model beschrijft de huidige situatie en de tweede model geeft het ideaalplaatje weer. Vervolgens stippel je een pad uit hoe je van het eerste model naar het tweede model gaat. Dit pad is vol uitdagingen en het vergt veel zendingswerk binnen een organisatie. Je zal het pad dan ook moeten bijstellen naar gelang de ontwikkelingen gedurende het transitieproces. Door constant uit te leggen waarom Continuous Delivery een goed idee is en wat je wil bereiken, maak je de stappen in de transitie steeds concreter.

Het is belangrijk om in het transitieproces kleine stappen te nemen en de resultaten zichtbaar te maken. Hiervoor kan je dezelfde matrix weer gebruiken, maar je kan ook demo sessies geven. Deze werken enorm motiverend. Je laat dan concreet zien dat iets werkt.
Tooling helpt om Continuous Delivery te implementeren en te faciliteren, maar het optimaliseren en borgen van processen is net zo belangrijk om jouw Continuous Delivery avontuur een succes te laten zijn. ■

Continuous Delivery bij de Kamer van Koophandel

De nadruk bij het implementeren van Continuous Delivery bij de Kamer van Koophandel ligt op het verhogen van de software kwaliteit, het versnellen van het opleverproces en het verkrijgen van zero downtime. Zero downtime is de term die we gebruiken voor het leveren van service zonder ooit down te gaan.

Software kwaliteit

Het begrip softwarekwaliteit is bij de Kamer van Koophandel gefocust op de snelheid van opleveringen en het gemak van onderhoud. Er zijn standaarden gedefinieerd en beschreven die deze focus specificeren. Op basis van deze

Frank Verbruggen is Senior Java Architect bij Ordina

standaarden worden verschillende metrics verzameld en geanalyseerd. Het doel van de metrics is om op een objectieve manier inzicht te krijgen in de kwaliteit van de software en waar het mogelijk is om automatisch beslissingen te maken.
Om voortdurend bij te blijven, goede beslissingen te maken en compliance te vergroten in de organisatie heeft de KvK een expertgroep in het leven geroepen.
Deze expertgroep is in sterke mate betrokken bij het bepalen wat kwaliteit nou precies is en zij zorgen ervoor dat het als zodanig in de organisatie ingebed wordt.

Testautomatisering

Testautomatisering is een belangrijk onderdeel van het meten van de softwarekwaliteit. Goede tests verifiëren zowel het functionele aspect van je code als dat ze het design verifiëren. Een moeilijk te maken test impliceert een code smell. Door de testautomatisering centraal te stellen in het proces zijn regressietesten goed geborgd.
Daarnaast heb je het voordeel dat iedere bijdrage van iedere tester iedere testrun opnieuw gedraaid wordt, waardoor de testbase van de producten steeds verder groeit. Het is belangrijk om ervoor te zorgen dat tests in een continuous delivery omgeving zowel goed als snel uit te voeren zijn.

Deployment-proces

Het deployment-proces wordt geborgd middels tooling. Hiervoor worden tools als Jenkins, Puppet en DeployIt gebruikt. Het automatisch deployen naar de ontwikkel- en testserver was een kwestie van gewoon doen. Voor de andere omgevingen was het niet zozeer een technische uitdaging, maar eerder een organisatorische uitdaging. Er zijn projecten die al hun releases

tot en met productie geautomatiseerd uitrollen. Om de zero downtime te realiseren, zijn een aantal stappen gemaakt. We zijn overgegaan op rolling updates. Tevens werden de servers gevirtualiseerd en is het nu mogelijk om automatisch horizontaal te schalen naar gelang de load op een cluster.

Conclusie

Als we terugkijken op het afgelopen jaar dan is al veel bereikt. De successen zijn bereikt door telkens kleine stappen te nemen en inzichtelijk te maken dat deze manier van werken veel voordelen heeft. We zijn er nog niet. Zo lang we telkens weer op zoek gaan naar concrete optimalisatie, komen we steeds dichterbij een optimaal proces in het opleveren van software. Het is vooral belangrijk dat de voordelen zichtbaar zijn. Dit heeft een zeer motiverend effect. ■

Continuous Delivery bij Malmberg

Malmberg behoort tot de top drie van educatieve uitgeverijen van Nederland in het basis-, voortgezet- en middelbaar onderwijs. Binnen Malmberg werken een aantal teams aan nieuwe, digitale leerplatformen voor het voortgezet onderwijs en MBO. Hierbij wordt ruim een jaar Continuous Delivery toegepast. Deze case deelt de ervaringen vanuit het oogpunt van die development teams.

Wat gaat er goed

Malmberg werkt Agile en heeft Scrum omarmd. De organisatie biedt een zeer goede suite collaboratietools aan die overal te gebruiken zijn, waaronder JIRA Agile, Confluence en Hipchat. Ontwikkelteams kunnen

en mogen veel zelf, waaronder het uitrollen van nieuwe omgevingen in de Amazon cloud. Als je naar de kosten kijkt, is dat niet zo gek. Een uur tijd verliezen met wachten, kost al snel meer dan de maandhuur van een gemiddelde Amazon-omgeving.
We werken met een moderne technologiestack: Java 8, Vert.x, AngularJS en MongoDB. Projecten werken samen aan een set generieke componenten, zowel voor de backend (Vert.x modules) als voor de frontend (AngularJS directives).
Alle broncode staat in private Github repositories. We gebruiken de pullrequest-flow van Github voor het ontwikkelen en reviewen van nieuwe features in branches. Een Jenkins job bouwt en test pull requests en met één druk op de kop kunnen we een branch mergen.
De Jenkins buildserver is het hart van het delivery-proces. Build slaves schalen on demand bij wanneer dit nodig is. Het uitgangspunt voor het build- en deploymentproces is dat alle stappen Maven modules zijn:

- builds, zowel van Java- als van frontend-modules (via NPM/Grunt/Gulp);
- deployments, via een zelf ontwikkelde deployment-module;
- inladen van testdata in databases van testomgevingen;
- releases: bouwen, versioneren en deployen naar Nexus;
- unit-, integratie- en end-to-end tests.

Dit zorgt ervoor dat we de stappen los kunnen uitvoeren, maar ook gemakkelijk aan elkaar kunnen koppelen in builds of build pipelines. Alle builds starten automatisch (via commit hooks) of met een druk op de knop. Dat bespaart tijd en voorkomt fouten. Zowel de ontwikkelteams als de beheerorganisatie gebruiken Nexus als artifact store voor snapshots en releases. Het opleveren van een release aan beheer betekent daardoor slechts het deployen van de betreffende artifacts vanuit de ene Nexus naar de andere.
Op productie en productie-like omgevingen gebruiken we Amazon ELB load balancers met minimaal twee applicatie-instanties erachter. Onze deployment tooling koppelt met de ELB API waardoor we 'zero downtime' deployments kunnen doen: eerst de ene instance down brengen en upgraden, daarna de andere. Deployments zijn daardoor voor gebruikers nauwelijks merkbaar.
We hebben flink wat automatische tests: Java en JavaScript-unittests, integratietests op REST-endpoints en end-to-end tests met Fitnesse, Browserstack en Protractor. De tests geven vertrouwen bij het refactoren van code en voorkomen regressie. JIRA Agile biedt inzicht in de status en voortgang van sprints, behaalde resultaten en groei die teams doormaken. Op technisch vlak hebben we inzicht in codekwaliteit en testdekking via Sonar en tools als JSHint en Checkstyle. Teams gebruiken die input actief om aan onderhoudbaarheid te werken.

Wat kan er nog beter

Het blijft een uitdaging om organisatiebreed op een Agile manier met stroomlijning en prioritering van werkstromen voor projecten om te gaan. De teams moeten daarover intern, onderling en richting stakeholders heel helder communiceren. Op het DevOps-vlak kunnen we ook nog groeien. Er is een samenwerking tussen projecten en beheer, maar nog niet op een niveau waarbij op dagelijkse basis in teams samengewerkt wordt.
Een ander aandachtspunt is de continuïteit van het CD-proces. Lang niet alle developers zijn op detailniveau op de hoogte van alle tools, jobs en omgevingen. En alhoewel we met een microservices-architectuur werken, deployen we onze applicaties nu nog als geheel naar productie. Het zou beter zijn om op serviceniveau uit te rollen, zodat we sneller en met minder impact wijzigingen live kunnen zetten. Op testomgevingen doen we dat al wel.
Veel testomgevingen worden nog met

Bert Jan Schrijver is Software craftsman bij JPoint, momenteel werkzaam bij Malmberg. Hij is erg geïnteresseerd in Java, open source en Continuous Delivery.

de hand beheerd. Een 'hands off' werkwijze met automatische provisioning van omgevingen zou tijd besparen en fouten voorkomen. Een van de teams is dat nu met Amazon OpsWorks aan het inrichten. Zodra we dat goed voor elkaar hebben, kunnen we ook met automatisch bijschalen van omgevingen aan de slag om zo beter op incidentele load te reageren.
Om een stap verder te komen richting continuous deployment moeten ook onze automatische tests nog beter en dan met name de acceptatietests. Die hebben we nu nog weinig en daar moet ook meer vertrouwen vanuit de organisatie in gaan komen. Momenteel testen we nog teveel handmatig.
Op het gebied van 'information & reporting' missen we metrieken over performance, gebruik en trends. Feedback daarop zou ons beter in staat stellen om onze gebruikers optimaal te bedienen en door te groeien in het CD-proces.

Conclusie

Continuous Delivery is maatwerk. Kijkend naar de invulling van het maturity model hebben we in een jaar al veel bereikt. Door de compleet geautomatiseerde build- en deploymentstraat hebben we een zeer korte time-to-market en kunnen we snel inspelen op feedback. De beperkende factor daarin is op het moment nog de tijd die de menselijke verificatiestappen innemen. Daar kunnen we door meer automatisering nog tijd en kwaliteit winnen. Een belangrijk kenmerk van Continuous Delivery is continuous improvement: blijf jezelf evalueren en verbeteren. Daar ligt onze focus de komende periode op. ■

Continuous Delivery bij de Nederlandse Spoorwegen

IT is voor de Nederlandse Spoorwegen key als het gaat om het leveren van voorspelbare en hoogwaardige diensten aan haar klanten. Ook de NS heeft behoefte aan een steeds kortere time-to-market voor nieuwe en verbeterde IT-diensten met hoge kwaliteit. Continuous Delivery en DevOps worden ook binnen deze enterprise gezien als een enabler en hier wordt dan ook sterk op ingezet. Het project dat dient als case voor dit artikel, is verantwoordelijk om een bestaand legacy systeem te vervangen, waarmee zogenaamde bijstuurders in staat zijn om treinen bij te sturen tijdens calamiteiten op het spoor.

Wat gaat goed

NS biedt met haar interne organisatie NSR-IT-OPS een goede basis voor iedereen die aan de slag gaat binnen een NS IT project. Het heeft een compleet platform met tools zoals Jira, Sonar, Jenkins, Nexus, Hipchat, Enterprise Architect en daarbij horende (proces) practices. Projecten kunnen hun eigen specifieke
behoeftes hierbinnen aanvullen.
Sinds medio 2009 is een transitie gestart naar multidisciplinaire teams die steeds meer verantwoordelijkheid dragen. De missie van elk team verschoof van discipline georiënteerd naar klantwaarde georiënteerd (het realiseren van klantwaarde op de meest optimale wijze). De teamleden werden gecoacht om meer T-Shape te worden; een verandering die tijd vergt! Testers en ontwikkelaars gingen intensiever samenwerken om te bepalen welke testen het beste geautomatiseerd kunnen worden en hoe. Daarbij hanteren ze vooral de theorie achter de Agile test-piramide. Daardoor werden langdurige en handmatige testperioden overbodig en kon sneller gereleased worden. Ook kwam vanuit de teams sterk de behoefte om de build feedback cycle zo kort mogelijk te houden. Dit heeft geleid tot een uitgebreide Jenkins build farm die zo snel mogelijk na een code checkin een nieuwe build maakt en onze gehele delivery pipeline doorloopt. In een pipeline gaan meerdere gespecialiseerde build verschillende stages af, waar mogelijk parallel. Hierin wordt automatisch gecompileerd (totaal 500.000 regels code), getest (unit, UI, integratie; totaal 15.000 testen), kwaliteit gemeten (Sonar, performance), geïnstalleerd (inrollen servers, deployen van de applicatie) en gereleased (release bouwen, taggen en in artifact repository Nexus zet-

ten). Een succesvolle build is dus meteen een release candidate.
Momenteel is de feedback cycle dermate snel dat de teams gestart zijn met het hanteren van een bijna 'zero-defect policy'. Een nieuw defect wordt toegevoegd aan de defect backlog als deze nog niet twintig defects bevat of als het nieuwe defect belangrijker is dan één van de huidige twintig. In de afgelopen twee maanden hebben we gemerkt dat dit model prima werkt en de juiste discussies bevordert. Daarnaast wordt voorkomen dat defecten die voor de klant niet relevant zijn, toch belangrijk worden (bijvoorbeeld minieme kleurverschillen). Let wel: dit werkt alleen als er niet al veel defecten aanwezig zijn.
Na deze optimalisaties bleek de bottleneck de uitrol naar acceptatie en productie omgevingen. De teams zijn dan ook gestart met het introduceren van Infrastructure as Code, om scripting voor de gehele OTAP te ontwikkelen met behulp van Puppet. Uitgangspunt is om deze scripts over de hele OTAP-omgeving zoveel mogelijk gelijk houden. Daarbij wordt momenteel zowel de client applicatie als de Oracle Weblogic middleware (Weblogic binaries, domains, en deployment artefacten) volledig uitgerold.
De productieomgeving is een 7x24 omgeving, met 99,9999% uptime! Met het oog daarop zijn de principes 'fail hard, fail fast', 'automate to the max' en 'if it hurts, do it more often' uitermate belangrijk.

Wat kan er nog beter

Een aanpak van eerst de applicatie deployen naar productie en vervolgens monitoren en daarop bijstellen geeft teveel risico, omdat het betekent dat bij het falen van het systeem miljoenen reizigers vertraging kunnen krijgen. Daarom wordt momenteel hard gewerkt aan het opzetten van continuous performance testing. Oftewel, hoe kunnen de scrumteams waarde blijven leveren terwijl ook continue gemeten wordt of de niet-functionele requirements (zoals performance, schaalbaarheid en 24X7 beschikbaar) gehaald worden. De teams opereren vanuit zogenaamde architecturele, significante scenario's en zetten tools als ContiPerf, JMeter als LoadRunner hiervoor in.
De verwachting is dat deze aanpak als het nieuwe 'normaal' zal worden ervaren binnen een half jaar in tegenstelling tot het huidige 'normaal'. Het huidige normaal is dat de performance pas wordt getest als de volledige scope gerealiseerd is.

Om beheer van en de gang naar productie nog effectiever en efficiënter te realiseren, is een intensieve samenwerking tussen de teams en beheer cruciaal. Hiervoor worden de mogelijkheden verkend om de huidige multidisciplinaire teams verder uit te breiden met Operations. Ultiem daarin zou zijn dat ieder teamlid ook echt om twee uur 's nachts uit bed gebeld kan worden bij applicatie problemen. Daarmee verandert onze Definition of Done van 'klaar voor productie' naar 'klaar om uit productie te nemen'!

Conclusie

Op veel aspecten heeft dit project Continuous Delivery ver gevorderd geïmplementeerd. Echter, uiteraard zijn er stappen te maken. Wanneer wij een nieuwe stap gaan maken, is het uitgangspunt altijd om eerst te bepalen wat de grootste bottleneck is in de Kanban flow van 'idee tot aan productie'. Vanaf daar starten we stapsgewijs de verbetering waarbij we altijd de winkel open houden. We willen wel waarde kunnen blijven creëren, terwijl we procesverbeteringen doorvoeren. Dit maakt acceptatie door stakeholders van het project en/of het product eenvoudiger en zorgt ervoor dat we ook continu kunnen evalueren of de visie en de stappen daar naartoe nog wel de juiste zijn. Tijd is daarbij wel een noodzakelijk aspect om een organisatie in zijn geheel te verbeteren, want let op: Continuous Delivery en DevOps betekent niet alleen een nieuw team inhuren! ■

Tim Prijn is Competence Lead Java en senior software engineer bij Info Support

Harald van Teeffelen is werkzaam als Senior Software Developer / Integrator bij InfoSupport, en op dit moment ingezet bij NS.