Continuous Integration

Softwarekwaliteit verbeteren met Continuous Integration.
Door Bart de Best

Context:
Deze blog is ontleend aan mijn ervaringen als DevOps trainer, coach en auditor met het concept Continuous Integration. Als trainer, coach en auditor krijg zie ik vele varianten van Continuous Integration toepassingen. Deze blog beschrijft mijn ervaringen met dit onderdeel van Continuous Everything.

Uitdaging:
De uitdaging van de toepassing van Continuous Integration is dat het diep snijdt in de werkwijze van software programmeren. Dit is al decennia een lastig onderdeel van kwaliteit management omdat iedere DevOps engineer op zijn of haar eigen wijze wil programmeren en daar energie aan ontleent. Aan de andere kant varieert de kwaliteit van de geschreven software van DevOps engineers sterk. Zo zijn leesbaarheid, onderhoudbaarheid, uitbreidbaarheid, integreerbaarheid, schaalbaarheid, beveiliging, performance kwaliteitsaspecten belangrijke factoren die een rol spelen in de time-to-market en het halen van SLA-normen. Continuous Integration richt zich vooral op de integreerbaarheid van de programmatuur van DevOps engineers en de versnelling van de time-to-market door het verlagen van de verspilling (waste).

Oplossing:
De oplossing voor deze uitdaging is gevonden in het concept van Continuous Integration waarin DevOps engineers meer keren per dag hun programmeerwerk delen en controleren of deze goed samenwerken. Deze blog bespreekt het concept Continuous Integration aan de hand van de volgende stappen:

1. De Definitie
2. De principes
3. De werkwijze
4. De ervaringen

1. De definitie

Continuous integration is als volgt te definiëren:

Continuous Integration
Continuous Integration is een holistische Lean software ontwikkelaanpak die beoogt om op een incrementele en iteratieve wijze continue software te produceren en in productie te nemen waarbij waste reductie hoog in het vaandel staat.

De kernwoorden zijn Lean en continue. Het woord ‘Lean’ verwijst naar de reductie van verspilling die bijvoorbeeld ontstaat door defecten dan wel incidenten die pas laat in de CI/CD secure Pipeline ontdekt worden. Het woord ‘continue’ verwijst naar het streven van een soepel branch (lokale copy van sourcecode maken) en merge (samenvoegen met de centrale sourcecode).

Dit wil zeggen dat een DevOps engineer een deel van de software kan aanpassen of uitbreiden en dat deze eenvoudig en zonder problemen samen te voegen (integratie) is met de rest van de software, ook als die tegelijkertijd door andere DevOps engineers wordt aangepast of uitgebreid. Als de branch leidt to merge conflicten dan wordt ook wel gesproken over de merge hell. Dit wil zeggen dat het heel complex kan zijn om de mutaties en extra functionaliteit samen te voegen. Vergelijk het met een MS Word document van 1 pagina die door 20 reviewers is aangepast en die samengevoegd moeten worden. Er zijn software ontwikkelprojecten gestopt omdat het oplossen van de merge hell langer zou duren dan alles opnieuw te programmeren.

2. De principes

De volgende principes zijn van toepassing op Continuous Integration:

- Er worden alleen kleine brokken software ontwikkeld
- Lokaal geïsoleerd ontwikkelen van software
- Gebruik van een centrale (remote) versiebeheer repository
- Hoge frequentie van samenvoegen van sourcecode
- Korte levensduur van een branch

Kleine brokken software

Bij Continuous Integration werken DevOps engineers met kleine, beheersbare stukjes software. Dit maakt het gemakkelijker om wijzigingen te integreren en problemen snel op te sporen en te verhelpen.

Lokaal

Elke DevOps engineer heeft een eigen omgeving om in te werken. Dit stelt hen in staat om nieuwe features of fixes te ontwikkelen zonder dat dit direct impact heeft op collega DevOps engineers of op de stabiele versie van de software. Software wordt alleen gedeeld als deze getest is.

Centraal

Een gedeelde repository, zoals GitHub, wordt gebruikt als de centrale bron waar alle sourcecode samenkomt. Dit zorgt voor een enkel punt van integratie en faciliteert samenwerking en tracking van veranderingen.

Frequentie

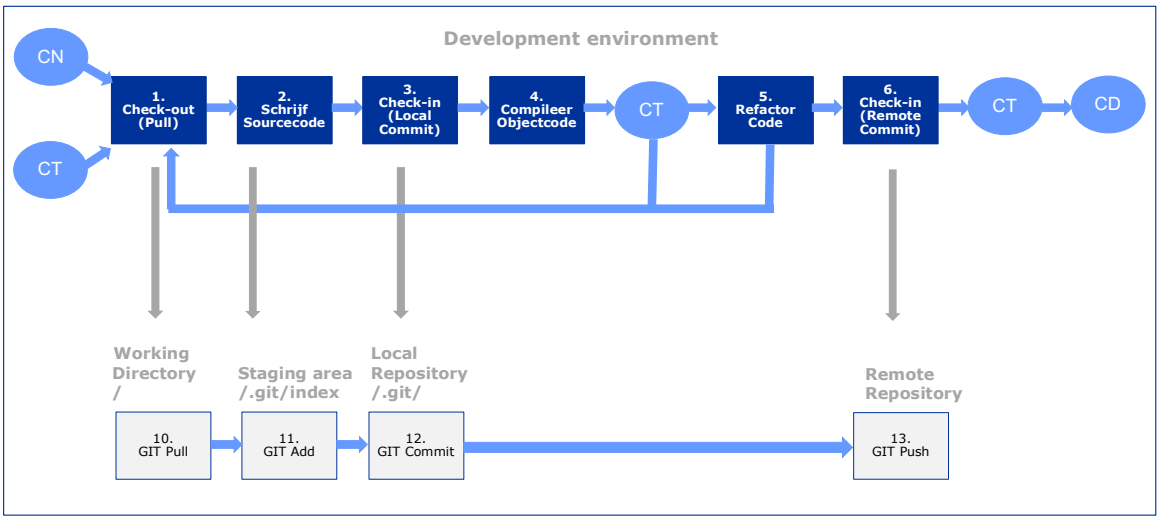
Door frequent wijzigingen samen te voegen, zoals meerdere keren per dag, worden integratieproblemen geminimaliseerd dan wel voorkomen. Dit zorgt ervoor dat defecten en conflicten snel worden ontdekt en opgelost.

Levensduur branch

Lange levensduur van branches kan leiden tot complexe merge-conflicten en integratieproblemen. Korte levenscycli van branches bevorderen snelle integratie en verminderen de kans op afwijkingen tussen verschillende brokken sourcecode.

3. De werkwijze

In figuur 1 is de value stream weergegeven van Continuous Integration. De stappen 1 tot en met 6 worden cyclisch doorlopen.



Figuur 1, Continuous Integration value stream.

De start van de value stream is Continuous Design (CN) die diverse design objecten aanlevert als ook de requirements. Continuous Testing (CT) is de value stream die een essentiële rol speelt in Continuous Integration (zie ook de Blog Productiviteits-verhoging door Continuous Testing). Tenslotte wordt de applicatie gedeployed en gereleased door Continuous Deployment (CD).

Stap 1. Check-out
De eerste stap is het clonen van de sourcecode die in een centrale repository is opgeslagen naar een lokale repository. Daarna maak wordt lokaal een branch aangemaakt.

Stap 2. Schrijf Sourcecode
De DevOps engineer kan nu aan de slag in zijn of haar eigen repository. Daarmee is de wijziging alleen lokaal te zien voor de eigen DevOps engineer en niet voor de collega's.

Stap 3. Check-in (local commit)
De local commit betekent dat de DevOps engineer de wijziging in de branch samenvoegt (merged) in de software in de lokale repository.

Stap 4. Compileer
De compilatie stap vertaalt de sourcecode naar objectcode. Hierdoor is de applicatie lokaal uit te voeren. Hierdoor kan middels Continuous Testing de applicatie getest worden. Er wordt net zolang door stappen 1, 2, 3, 4 gelopen totdat de testcases succesvol zijn.

Stap 5. Refactoring
De sourcecode wordt opgeschoond en hulp sourcecode (scaffolding) wordt verwijderd.

Stap 6. Check-in (remote commit)
Ten slotte wordt de locatie repository gesynchroniseerd met de centrale repository. Dit is het moment van integratie. Als deze integratie meer keren per dag gebeurt, dan is er spraken van Continuous Integration.

4. De ervaringen

In de loop van de jaren heb ik verschillende ervaringen mogen opdoen met de Continuous Integration die ik graag met u deel.

Eigen ervaring
Het lokaal werken aan sourcecode geeft een veilig gevoel omdat er geen werk van collega's verloren kan gaan. De neiging is wel dat je zo veel focus hebt op de realisatie van de ontwerp en de requirements dat je vergeet om te controleren of alles wel samenwerkt met wat anderen aan het maken zijn.

Het gebruik van pair programming waarbij twee DevOps engineers samen achter het beeldscherm werken om software te schrijven zorgt voor een grote daling van fouten. Hierbij type de ene DevOps engineer software terwijl de ander aangeeft wat moet worden gecodeerd. Dit doet geen afbreuk aan de eis een hoge mate van integratie.

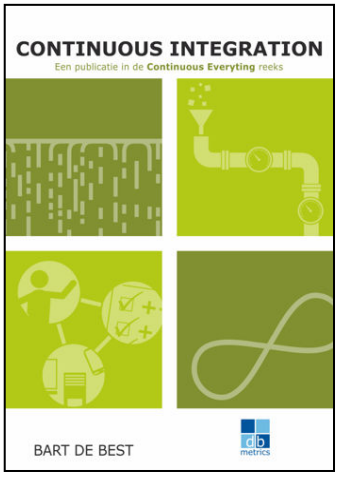
Het samenvoegen in de centrale repository vereist wel dat er gelijk systeem integratie testen en systeem testen worden uitgevoerd die de gehele applicatie testen. Als dit niet het geval is dan is het de bedoeling dat iedereen stopt met programmeren tot de build en de daarop volgende testen wel succesvol zijn. Dit heet het herstel van een broken build. Door iedereen mee te laten werken ontstaat ook een positief leereffect van welke fouten voorkomen moeten worden.

Training ervaringen
In trainingen vinden vaak discussies plaats of het wel noodzakelijk is om vaak te integreren. Zeker als je gebruik maakt van microservices waarvan de interfaces zijn gedefinieerd en collega's gebruik kunnen maken van mocking waarbij elkaars microservices gesimuleerd worden.
Dit is inderdaad een situatie waarbij er minder conflicten zullen ontstaan bij een merge. Maar de werking van de applicatie is meer dan software compatibiliteit. Ook het gedrag van de verschillende delen van de applicatie en de wijze waarop informatie wordt verwerkt vereist een goede en hoog frequentie integratie.

Audit ervaringen
Tijdens audits merk ik dat de vragen omtrent branching en merging vaak discussies opleveren. Nogal eens blijken DevOps teams geen goed gedefinieerd ontwikkel-proces te hebben. Het is dan meer de willekeur van de DevOps engineer. De ene DevOps engineer heeft een hoge merge frequentie en de ander merged dagen tot wekenlang niets. Dit is natuurlijk niet wat verstaan wordt onder Continuous Integration. Het hele DevOps team dat aan een applicatie werkt moet zich houden aan de principes en de Continuous Integration value stream om succesvol te zijn.

Met deze toepassing van de Continuous Integration is het mogelijk om continu vast te stellen of de applicatie goed functioneert en worden fouten snel gevonden en zijn ook snel op te lossen. Daarmee kan ook de frequentie van deployen verhoogd worden. Daarom is op deze wijze van software ontwikkeling een goed voorbeeld van de toepassing van Continuous Integration.

Door Bart de Best
DutchNordic.Group



https://www.dbmetrics.nl/ce-nl/continuous-integration-nl/