Introductie

Mijn naam is Rick Sollman. Ik ben werkzaam bij CGI en heb daar in 2015 een intern talent ontwikkel programma gevolgd. Als afsluiter van dit programma kon men kiezen uit een viertal opdrachten, ik heb gekozen om een visie document te schrijven.

Dit document beschrijft mijn visie over het releasen van software. Vanaf de ontwikkeling van een regel code tot het in productie nemen van die code, desbetreffende primaire en secundaire voorwaarden inclusief een best practice van de software stack.

Mijn visie over software releases is gebaseerd op jarenlange ervaring met dit onderdeel van software ontwikkeling. Geïnspireerd hoe releases van software in de open source wereld plaatsvinden, ervaringen van mijn huidige opdracht bij het UWV en vorige opdracht bij CAK zijn in deze visie verwerkt.

Pre conditie

Om enigszins de scoop te bepalen voor mijn visie wordt de software stack bepaald. Aangezien ik Java ontwikkelaar beaam te zijn is het erg waarschijnlijk dat het tools en methoden behelst door de Java community zijn omarmt. Hieruit is direct mijn visie qua tools af te leiden. De stack bestaat uit:

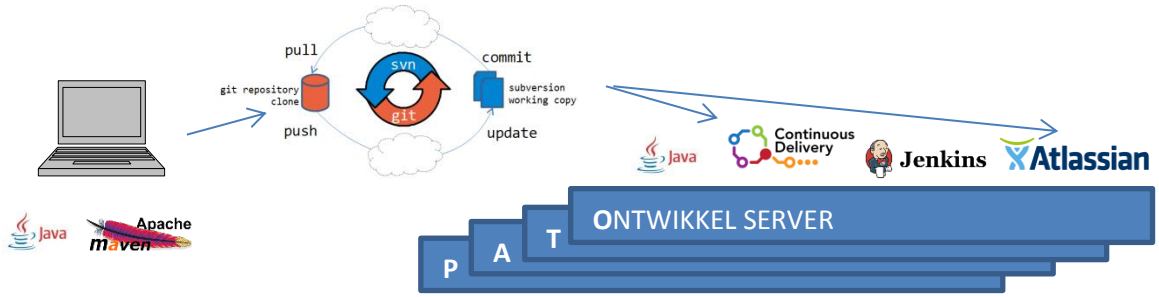
- Java als programmeertaal
- Maven[1] voor project modellering
- SVN[2] / GIT[3] voor versiebeheer
- Jira[4] voor issue tracking
- Jenkins[5] als integratie en oplever platform

Voorwaarde voor een applicatie server stel ik niet echt, maar als ik mag kiezen dan zou Apache Tomcat[6] als container fungeren. Een database management systeem is in mijn visie ook buiten beschouwing gelaten.

Tevens prefereer ik een Apple of Linux besturingssysteem boven Windows aangezien de standaard tools en gebruik van de hardware aanzienlijk beter is bij eerstgenoemde.

Overzicht

Dit hoofdstuk geeft een beeld van hoe mijn visie over het releasen van software het beste tot stand komt. Zoals eerder gemeld van het coderen tot in productie nemen in een globaal overzicht.



Een ontwikkelaar codeert in zijn/haar favoriete IDE[7] zoals de gewenste functionaliteit. Maven ondersteunt het modelleren van het project en stimuleert hergebruik van bestaande open source bibliotheken. Nieuwe en aangepaste code wordt in versiebeheer geplaatst. Typisch onder een uniek nummer van het issue tracking systeem Jira.

Hooks vanuit versiebeheer richting desbetreffende issue zorgen voor een correcte audit trail. Tevens wordt het bouwproces automatisch getriggerd in Jenkins om continue bouwproces en integraal teamwork te waarborgen.

Code reviews worden gedaan in Crucible[8] van Atlassian en zorgen voor continue kwaliteit van code dat uit de release komt. Het geheel wordt in iteraties herhaald in de zogenaamde OTAP[9] straat. Hierdoor wordt kwaliteit van installatie, testen, acceptatie en zorgeloze in productie name gewaarborgd.

In de volgende hoofdstukken wordt er verder verdiept in de verscheidene onderdelen van mijn visie over een software release.

Programmeren

De eerste gedachte zou kunnen zijn dat je kan denken, wat heeft een programmeur nou met software release te maken. Die programmeur codeert toch slechts alleen en communiceert nauwelijks. In mijn visie is dit zeker niet het geval en is de programmeur, of liever gezegd ontwikkelaar, in staat om test gedreven te ontwikkelen. Het schrijven van unit testen die falen en “groen” gemaakt worden. Idealiter worden ook behaviour driven tests[10] gemaakt waarmee functioneel getest kan worden en regressie wordt voorkomen. De testen vind ik een essentieel onderdeel van een release, software kan en mag niet worden gereleased zonder dat alle testen slagen.

Laatstgenoemde testen worden gemaakt met behulp van Selenium[11] en Cucumber[12]. Beide open source oplossingen die met een paar eigen Java klassen kunnen leiden tot een goede geoliede motor waaruit software komt rollen die te allen tijde voldoet aan de gestelde eisen en wensen.

Ingericht met Jenkins, die bijvoorbeeld midden in de nacht het apart gemodelleerde maven module voor de functionele en integratie testen automatisch uitvoert, leiden tot overzichtelijke rapportages van de geautomatiseerde test resultaten. Uit eigen ervaring kan ik beamen dat dit ook voor de ontwikkelteams erg stimuleert om goede code op te leveren met bijbehorende testen. Ook belonen voor foutieve of juiste goede build en/of automatische testen is erg inspirerend en komen mijn inziens de harde en zachte kanten van software releases heel mooi samen.

Wanneer in een ontwikkelteam de testers dicht aansluiten bij de business en de gewenste functionaliteiten beschrijven in de zogenaamde “given-when-then” methode dan is de ontwikkelaar in staat om deze beschrijving al dusdanig te coderen in een test. Dit leidt weer tot een Agile[13] werkwijze die oplevering van software tot een mooi en vooral leuk samenspel maakt.

Maven

Maven is een open source tool dat gebruikt wordt voor modellering van voornamelijk Java projecten. Het is momenteel de standaard en wordt uiteindelijk vervangen door hoogstwaarschijnlijk gradle. Maven is een zeer krachtig hulpmiddel tijdens software ontwikkeling en vooral tijdens het release van de software. In mijn visie bestaat een software project uit verscheidene zogenaamde POMs, xml bestanden, die het project structureren, de afhankelijkheden (dependencies) beheert en zorg voor een bouw of package van het gehele project met één druk op de knop.

Door het gehele project worden eigen software artifacts slechts met één versie nummer uitgerust. Een versienummer bestaat uit de naam van de op te leveren software gevolgd door drie cijfers met als suffix een build nummer van het versiebeheer systeem, bijvoorbeeld *MijnSoftware-4.2.6-r78045*. De release plugin van maven wordt gebruikt voor het release van de software, ophogen van de versienummers en taggen in het versiebeheersysteem. Tussentijds wordt gewerkt op de zogenoemde SNAPSHOT versie en deze wordt als niet stabiel beschouwd.

Versiebeheer

Voor versiebeheer wordt een systeem als subversion of GIT gebruikt. Er wordt gewerkt volgens de best practice bekend onder de *trunk/branches/tags* structuur. De trunk is voor de ontwikkeling en er is een maintenance branch voor onderhoud van productie. Merging wordt tot een minimum beperkt, slechts bugfixes van de maintenance branch of een eventuele feature branch richting de trunk. Parallel levende versies op verschillende branches is uit den boze, leidt tot eventuele regressie en extra niet waardevolle werkzaamheden.

Over de inrichting van subversion zijn talloze best practices te vinden, mijn persoonlijke voorkeur is een heldere logische inrichting van de “takken” en ik vermijd altijd technische namen. Ik deel ze altijd in op functioneel vlak, dit helpt om snel te herkennen waar wat staat en men is niet afhankelijk van de toegepaste techniek.

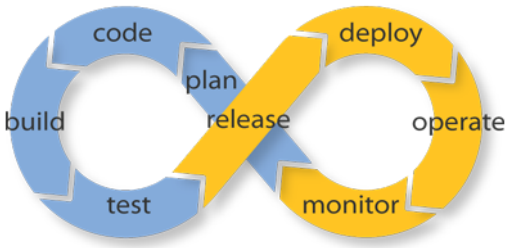
Wanneer het mogelijk is om nog te kiezen in het toe te passen versiebeheer systeem dan is in mijn visie GIT in gebruik. Dit omdat GIT aanzienlijk sneller is omdat alleen de verschillen worden

bijgehouden. Tevens is de lokale branching mogelijkheid behoorlijk toegevoegde waarde voor ontwikkelaars.

Documentatie

Documenteren van de opgeleverde software, doorvoeren van wijzigingen en oplossen van bugfixes etc is een must in mijn visie. De wijze van documenteren valt over te twisten, mijn visie is het *lean and mean* te houden en in een tekst formaat. Dit stelt ontwikkelaars, testers en anderen in staat de revisies uit versie beheer in te zien. Mijn persoonlijke voorkeur gaat uit naar het markdown[14] formaat. Documenten die typisch bij een release horen zijn installatie documentatie, mits benodigd en natuurlijk de welbekende release notes.

Continue integratie



Om continue integratie en bouwen van de software af te dwingen is een zogenaamde build server zoals Jenkins gewenst. Bamboo is ook een erg goede build server, maar daar is een licentie voor nodig en Jenkins doet het werk net zo goed en gratis. Typisch zijn de hooks vanuit een versiebeheer die na een commit een complete bouw van het software project bouwen volgens plan. Een typisch ander bouwplan is er één voor functionele testen. De output van deze testen is een rapport waarin men in één oogopslag kan zien of de testen succesvol zijn en of er bijvoorbeeld regressie is.

OTAP

Dit is een welbekend concept in het ontwikkel en release landschap. De afkorting staat voor ontwikkel, test, acceptatie en productie. Doel is om tot een release te komen in productie waarvan men zeker kan zijn dat er geen onverwachte fouten optreden. Kortom, er zijn vier omgevingen waarbij de ontwikkelomgeving dient als speeltuin, waar de initiële release wordt uitgevoerd en men in iteraties tot een punt komt dat de release gereed is voor de test omgeving.

Mijn persoonlijke voorkeur is dat er in de test omgeving wordt gewerkt op basis van SNAPSHOT versies. Dit vereist enige extra communicatie tussen de testers en ontwikkelaars, maar kan het ontwikkel en release proces, dus de time to market, behoorlijk versnellen.

Acceptatie en productie worden idealiter door operationele beheerders beheerd. Tevens voeren zij de releases uit op die omgeving. Hierdoor kijken er een extra paar ogen en is de acceptatie omgeving een goede oefening, tevens een mooie scheiding tussen de ontwikkelaars en beheerders.

Conclusie

Begrippenlijst

1. Maven is een softwaregereedschap voor Java-projectmanagement en geautomatiseerde softwarebouw.
2. Subversion (SVN) is een versiebeheersysteem en in 2000 opgezet door CollabNet Inc. Subversion is de opvolger van CVS, een alternatief versiebeheersysteem. Subversion is uitgebracht onder de Apache License, waardoor het opensourcesoftware is.
3. Git is een vrij gedistribueerd versiebeheersysteem. Het wordt ook wel een softwarebroncode-managementproject genoemd.
4. Jira is een eigen issue tracking product, ontwikkeld door Atlassian.
5. Jenkins is een open source continue integratie hulpmiddel geschreven in Java. Het project werd gevorkte uit Hudson na een geschil met Oracle.
6. Apache Tomcat is een opensource webcontainer ontwikkeld door de Apache Software Foundation (ASF). Tomcat voert servlets en JavaServer-pagina's uit, het verzorgt de communicatie tussen JSP-pagina's en een webserver.
7. Integrated Development Environment, een software-ontwikkelomgeving voor programmeurs.
8. Crucible is a collaborative code review application by Australian software company Atlassian.
9. Ontwikkeling Test Acceptatie en Productie, afgekort OTAP is de naam van een methodiek die wordt gebruikt in de ICT. De hoofdwoorden in de naam geven de fases aan die onder andere in de softwareontwikkeling doorlopen worden.
10. Behaviour Driven development (BDD, letterlijk vertaald: gedragsgedreven ontwikkeling) is een manier van programmeren waarbij eerst het gedrag beschreven wordt alvorens men daadwerkelijk gaat programmeren.
11. Selenium is een draagbare software toetsingskader voor web applicaties. Selenium biedt een record / playback tool voor authoring proeven zonder het leren van een test scripttaal.
12. Komkommer is een software tool die programmeurs gebruiken voor het testen van andere software. Het loopt geautomatiseerde acceptatietesten geschreven in een BDD stijl.
13. Agile-softwareontwikkeling is een manier van softwareontwikkeling. Het Engelse woord agile betekent: behendig, lenig.
14. Markdown is een lichtgewicht opmaaktaal volle tekst opmaak syntax ontworpen dat het kan worden omgezet in HTML en vele andere formaten met een gereedschap met dezelfde naam.

Referenties

1. SVN best practice: https://svn.apache.org/repos/asf/subversion/trunk/doc/user/svn-best-practices.html
2. Software releases: https://en.wikipedia.org/wiki/Software_release_life_cycle
3. Boek: clean code…. The bible ☺