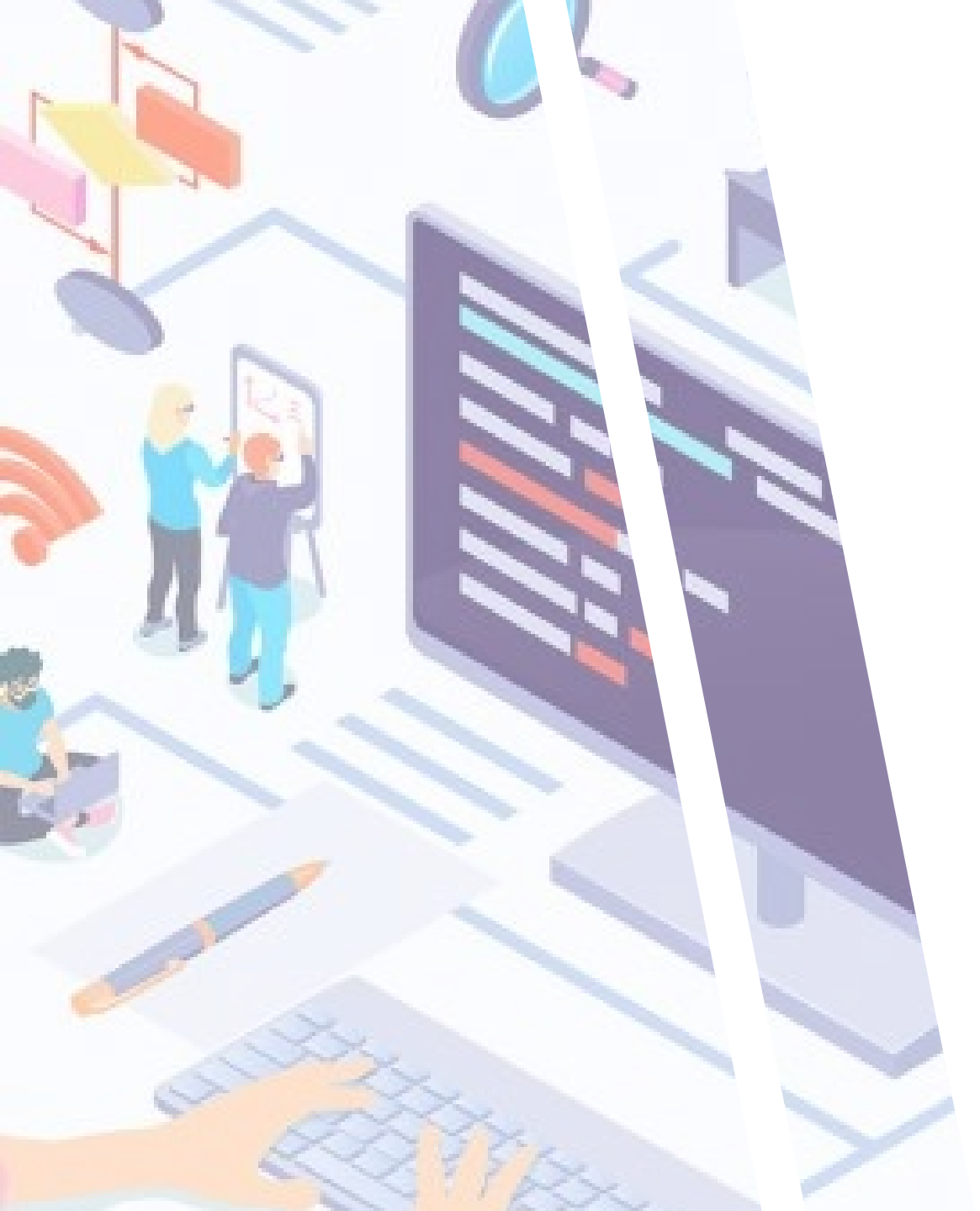


SDLC en DevOps

IT-Organisation 2

Cursusinhoud

- Hoofdstuk 1: Software development lifecycle
- Hoofdstuk 2: Ontwikkelomgevingen
- Hoofdstuk 3: DevOps

Leerdoelen

- Hoofdstuk 1: Software development lifecycle
  - De student kan SDLC in eigen woorden uitleggen
  - De student kan de domeinen van SDLC benoemen
  - De student kan de domeinen van SDLC in eigen woorden uitleggen
  - De student kan de domeinen van SDLC herkennen in een casus
  - De student kan de domeinen van SDLC toepassen in een casus
- Hoofdstuk 2: Ontwikkelomgevingen
  - De student kan de ontwikkelomgevingen benoemen
  - De student kan de ontwikkelomgevingen in eigen woorden uitleggen en weet waarvoor elke omgeving dient en wat erin gebeurt
  - De student kan de ontwikkelomgeving toepassen op een casus
- Hoofdstuk 3: DevOps
  - De student kan uitleggen wat DevOps betekent en kan voorbeelden geven van veelgebruikte technologieën
  - De student kent de voordelen van een DevOps werking en kan het vergelijken met Agile en Waterval

Hoofdstuk 1

Software development lifecycle

Wat is de Software Development Lifecycle? SDLC

De Software Development Life Cycle (SDLC) beschrijft de **stappen** die nodig zijn bij de **ontwikkeling** van software in elke fase.

Het geeft aan **welke fases** je moet doorlopen voor het **bouwen**, **implementeren** en **onderhouden** van software.

Software development lifecycle - Algemeen

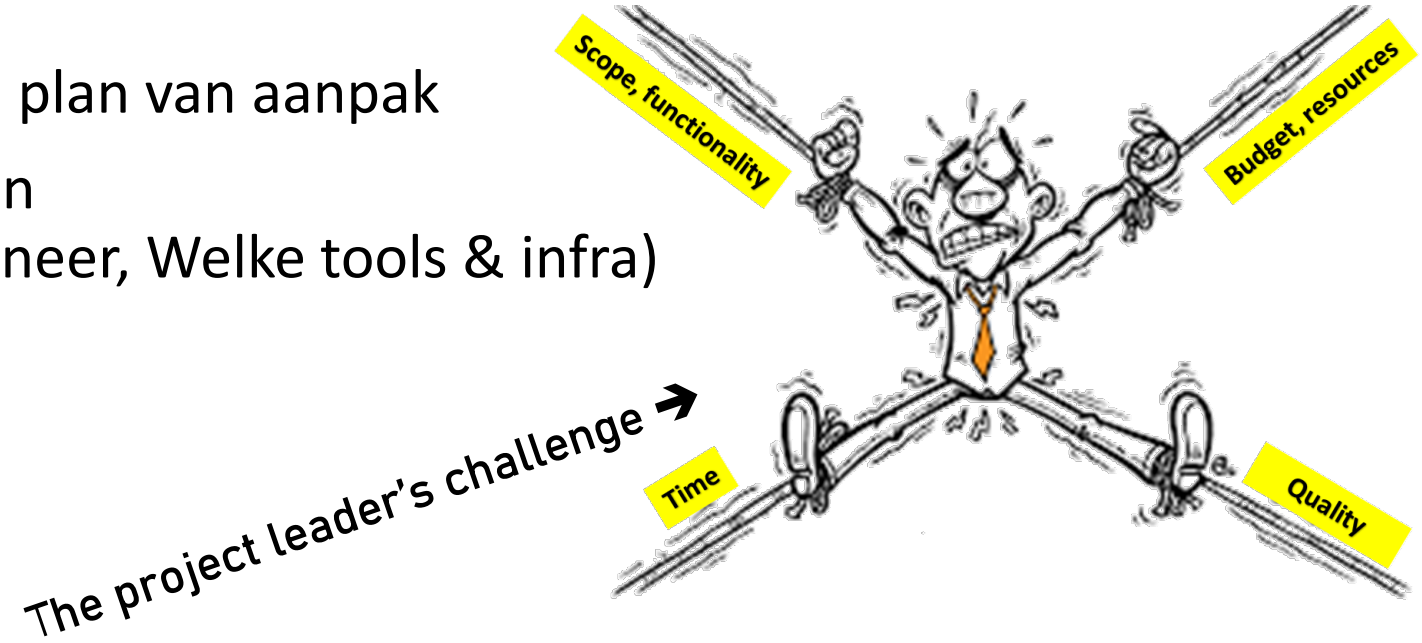
1. Planning
2. Analyse
3. Design
4. Implemen-tatie
5. Testing & integratie
6. Maintenance

Software development lifecycle – Planning

Planning

Planning

- Stappen bepalen
- Zorgen voor de juiste resources
- Project plannen, hoe lang het duurt
- Kosten inschatten
- Opstellen projectplan, plan van aanpak
- Antwoord op W-vragen
  (Wie, Wat, Waar, Wanneer, Welke tools & infra)



Software development lifecycle – Analyse

Analyse

Analyse

- Requirements duidelijk maken
  - Wat moet de software kunnen?
  - Bevraging van de klanten of eindgebruikers
  - Basis voor een haalbaarheidsonderzoek
- Al de requirements door de klant laten goedkeuren, zodat er geen misverstanden zijn
- Een analyse wordt genoteerd in een **SRS** (Software Requirements Specification*), dit is een template om alle **functionele en niet-functionele vereisten** van een product vast te leggen

* IEEE 830, Internationale standard

Wat zijn functionele en niet-functionele vereisten?

- **Functionele vereisten :**
  - Beschrijven wat het verwacht gedrag is van het systeem.
  - **WAT** het systeem moet doen voor de gebruiker.

Bijvoorbeeld :

- Webshop: Ik wil van een product de detailgegevens kunnen raadplegen en de prijs.
- Fitness app: Ik wil een overzicht van hoeveel calorieën ik vandaag verbrand heb.
- Ticketverkoop: We willen de mogelijkheid bieden om tickets te kunnen kopen voor een event.
- De wagen moet een maximum snelheid halen van 150 km/u.

Wat zijn functionele en niet-functionele vereisten?

- **Niet-functionele vereisten :**
  - Geven aan op welke manier het systeem de werking moet aanbieden.
  - **HOE** het systeem moet werken, met verwachtingen naar :
    - Performantie
    - Schaalbaarheid
    - Maximum aantal gebruikers
    - Backups
    - Security / privacy
    - Compatibiliteit

Bijvoorbeeld:

- Binnen 30 seconden moet je kunnen ingelogd zijn op de website
- 500 gebruikers moeten tegelijkertijd kunnen inloggen zonder impact te hebben op performantie
- Als een document geprint wordt, mag het maximum 5 seconden duren voor het document uit de printer rolt
- Gebruikers in een agentschap mogen geen toegang hebben tot de product beheersmodule.

Wat zijn functionele en niet-functionele vereisten?

Manier van formuleren van requirements
kan via user stories:

Voorbeeld functionele user story

As a lecturer,

I want to be able to understand my colleagues progress,

So that I can better report their success and points of attention.

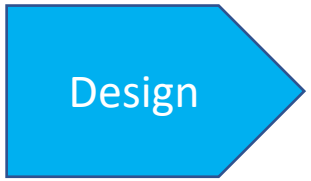
Voorbeeld niet-functionele user story

As a user of Blackboard,

I want Blackboard to be available 99.999 percent of the time I try to access it,

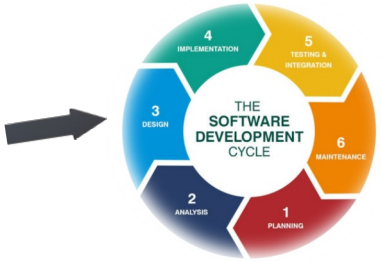
so that I don't get frustrated.

Software development lifecycle - Design



Design

- SRS vormt de basis voor de architecten
- Een ontwerp of design wordt genoteerd in een DDS – Design Document Specification
- Een ontwerpdocument is een meer gedetailleerde beschrijving van het product of de software (met beschrijvingen van schermen, benodigde data of tabellen, beschrijving van technische metingen systeem security en performantie te verzekeren, …)
- Bij agile werkvormen kan een design tijdens de loop van een project nog gemakkelijk gewijzigd worden



Software development lifecycle - Implementatie

Implemen-tatie

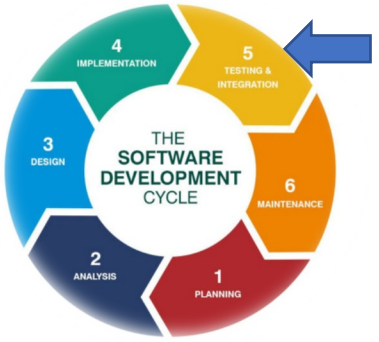
Implementatie

- DDS vormt de basis voor de programmatie
- Naast DDS kunnen er ook coding guidelines zijn: naamgeving van klasses, naamgeving tests, structuur code, aantal code reviewers, …
- Hier wordt er gebruik gemaakt van *ontwikkelomgevingen*
- Bij waterval gaat men op het einde van implementatie naar de testomgeving
- Bij agile werkvormen kan men op het einde van iedere sprint naar productie gaan
  - Klant heeft dan een direct voordeel
  - In de praktijk gebeurt dit soms niet iedere sprint, maar op afgesproken releasemomenten

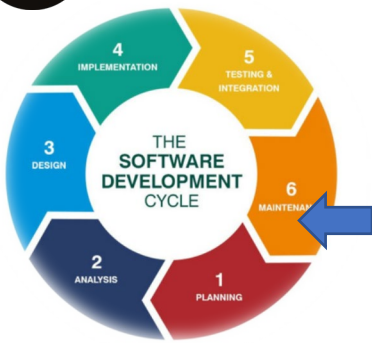
Software development lifecycle - Testen

Testing

- **Testen** na ontwikkeling is een belangrijke fase.
- Afhankelijk van of je met een klassieke projectmethode of agile werkt, kan testen op **verschillende momenten** ingepland worden in je project
  - Bij agile werken in elke iteratie
  - In klassieke ontwikkeling na de ontwikkelfase
- Doel van testen is **kwaliteit** na te streven
  - Testen of de requirements, opgesteld in analysefase, voldaan zijn (nakijken SRS). Men maakt **testscenario's** op.
  - Testen is **functioneel** (werkt alles zoals verwacht) zowel als **technisch** (bv performantie, …)
  - Testen gebeurt best door **verschillende mensen** en vanuit verschillende invalshoeken.
  - Testen is **tijdsintensief** en mag niet onderschat worden.
- Verschillende soorten testen: zie hoofdstuk over omgevingen.



Software development lifecycle - Maintenance

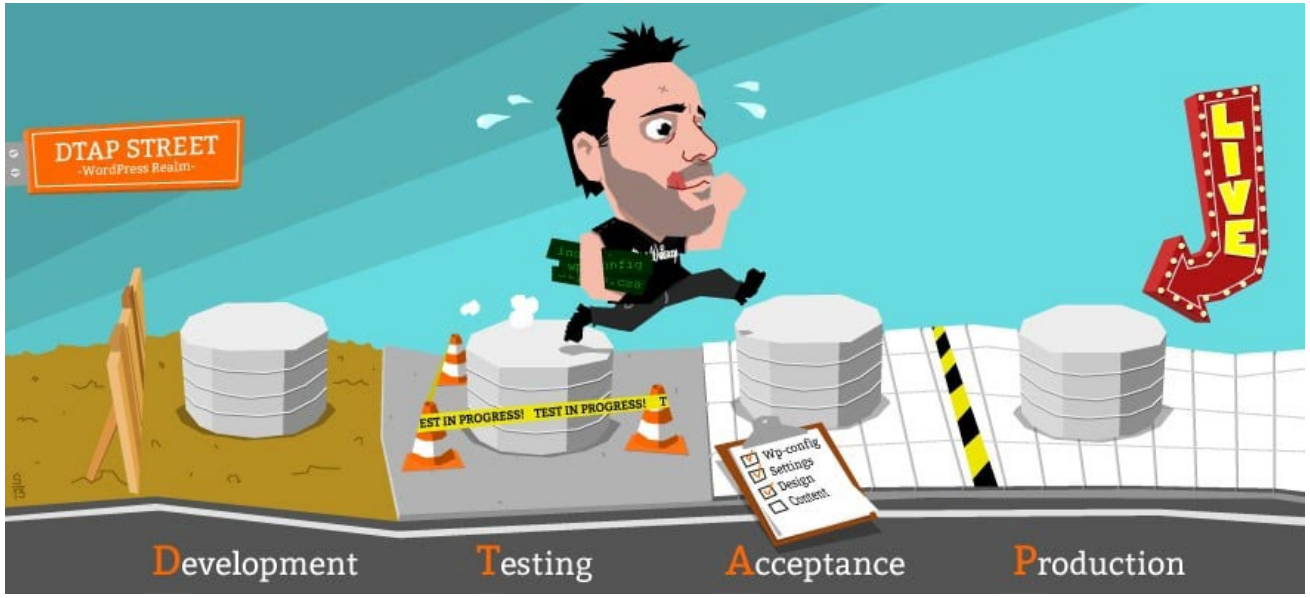


6. Maintenance

- Het product is klaar en gaat **live/ naar productie**
  - Alle requirements uit SRS geïmplementeerd en getest
  - Alle processen en integratie met andere toepassingen is getest
  - Het product kan “in productie” gezet worden. Men zegt ook “gereleased” worden. Het wordt overgezet naar productie-omgeving.
- Soms wordt een product **gefaseerd** uitgerold
  - Bv. eerst naar een kleinere doelgroep. Men test zo of de in productiestelling goed werkt, kan feedback van eindgebruikers capteren en nog aanpassingen uitvoeren alvorens het voor een groot publiek open te stellen.
  - Bv. een deel van het product, om meer controle te hebben over de impact of het risico dat er zaken fout lopen
- In een eerste **garantieperiode** wordt er gratis support aangeboden.
- Er worden **afspraken gemaakt met de klant** over het onderhoud van het product (bv. een jaarlijks budget voor het oplossen van fouten en kleine aanpassingen, of betalen per aanvraag, of een budget voor het oplossen van bugs maar niet voor nieuwe functionaliteiten…)

Hoofdstuk 2

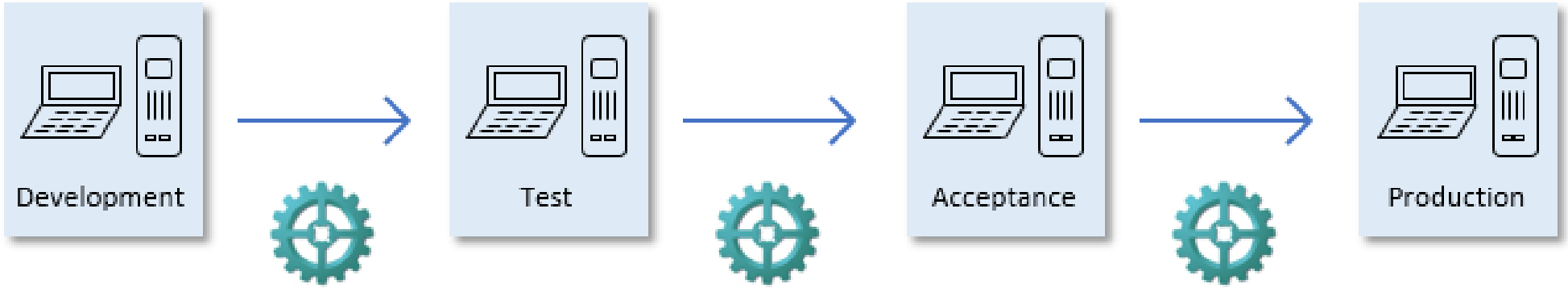
Ontwikkelomgevingen



Ontwikkelomgevingen

- Idealiter wordt er met deze omgevingen gewerkt tijdens de ontwikkeling (DTAP)
  - Development
  - Test
  - Acceptatie
  - Productie

- Afhankelijk van de organisatie worden al deze omgevingen gebruikt, of enkel een subset
  - Een productie omgeving is er altijd! ➔ OTAP, teststraat



Ontwikkelomgevingen

Development

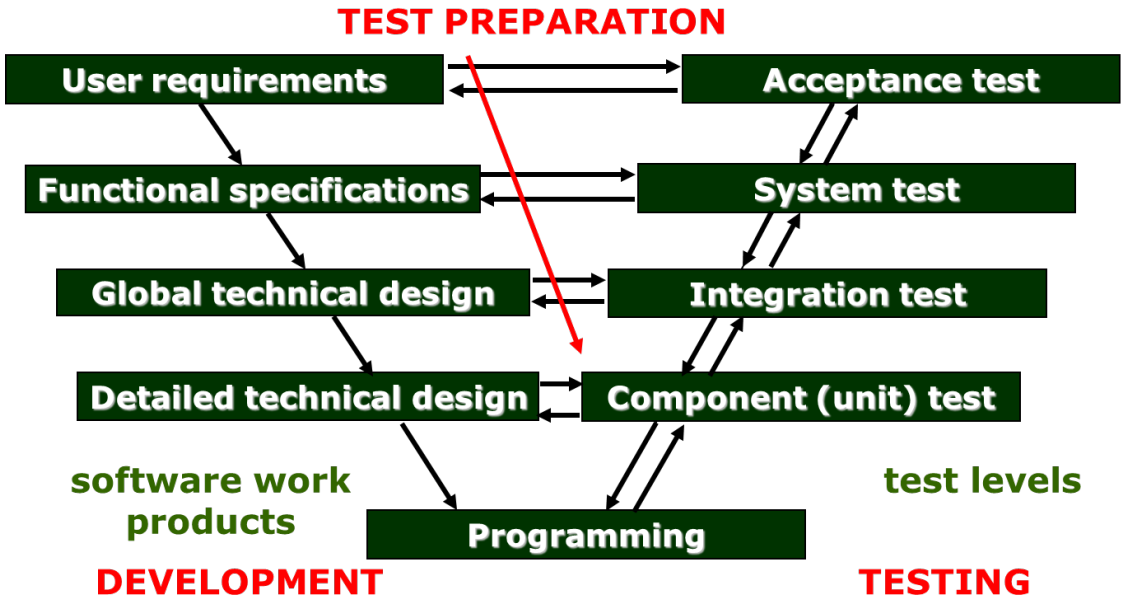
- HW/SW* platform en tools die de ontwikkelaar gebruikt voor ontwikkeling van de code
- Naast development, ook eerste testen en/of review mogelijk
- Meestal nog heel instabiel, omdat er nog geen uitgebreide testen zijn gebeurd

*HW/SW: Hardware/Software

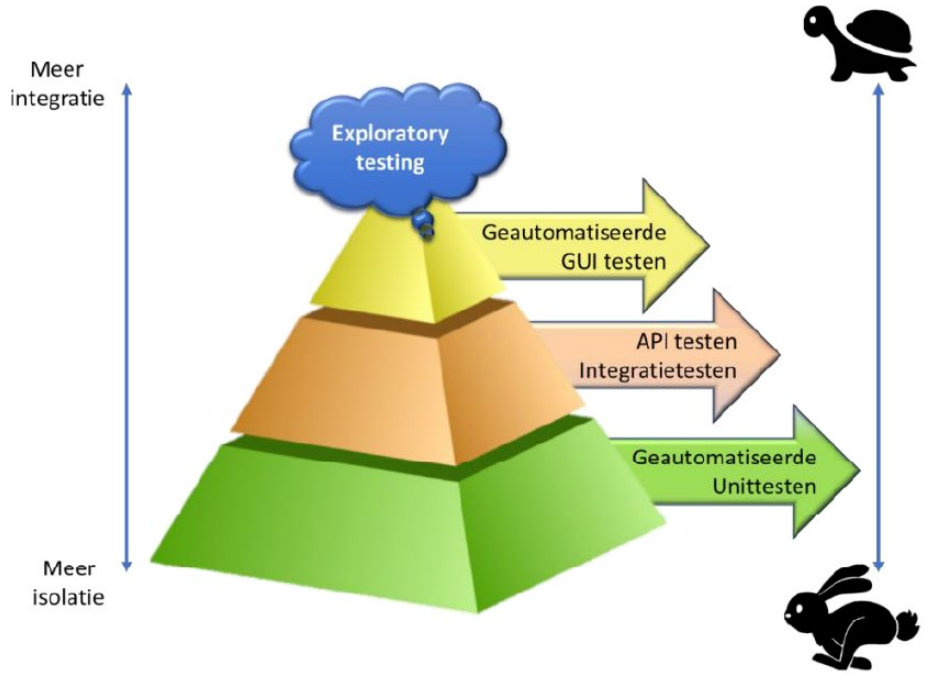
Ontwikkelomgevingen

Test

- Wanneer de development omgeving getest werd door de ontwikkelaars, worden alle functionaliteiten van de development omgeving ‘gekopieerd’ naar de testomgeving
- Stabieler dan de ontwikkelomgeving, omdat de unittesten hebben plaats gevonden



Testing volgens traditionele V-model



Agile Testing Piramide

Ontwikkelomgevingen

Acceptatie (“staging environment”)

- Als alle noodzakelijke features op de testomgeving goed werken, wordt de testomgeving ‘gekopieerd’ naar de acceptatie omgeving
- Op deze omgeving gaat de klant zijn goedkeuring geven voor de gebouwde features
- De klant of een vertegenwoordiger van de klant, voert de testen uit op deze omgeving. Er kunnen ook specifieke/andere testers ingeschakeld worden.
- Dit is de laatste stap voor het product toegankelijk is voor de eindgebruikers
- De acceptatie-omgeving is qua software, hardware en gegevens zoveel mogelijk gelijk aan de productie omgeving (bijna een replica van de productie-omgeving zodat je productie zo goed mogelijk kan simuleren)

Ontwikkelomgevingen

Acceptatie

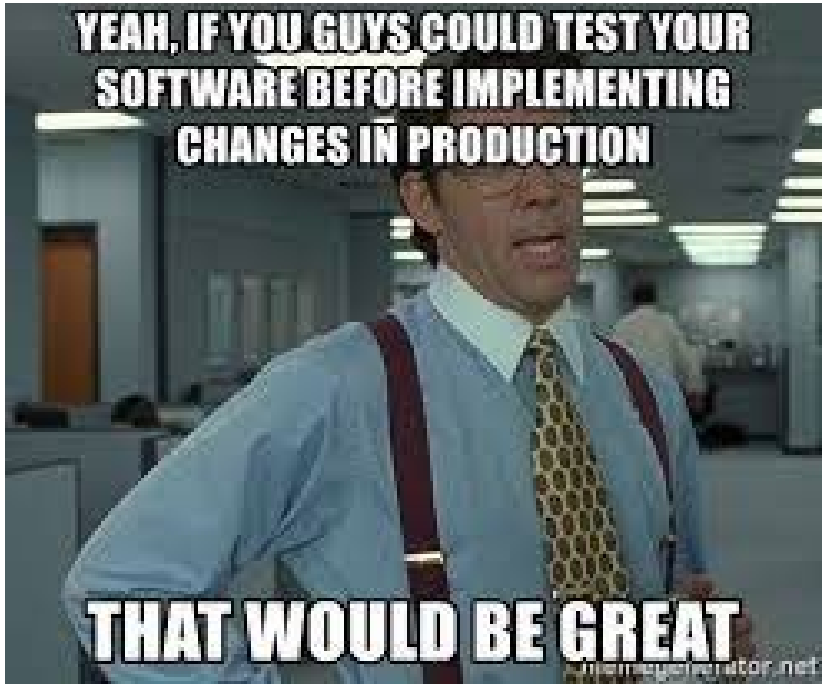
Volgende soorten testen kunnen uitgevoerd worden:

- Functionele test
  - Testen of de functionaliteiten werken
  - Bv: Als de toepassing een webshop is, testen of alles werkt zoals verwacht
- Piek- of stresstest
  - Testen of het systeem tegen piekbelasting kan
  - Bv: veel verkeer op de website, moeilijke berekeningen,…
- Monkey-proof test
  - Is het systeem bestand tegen ongeoorloofde en onjuiste handelingen
  - Bv: testen met random inputs en random klikken door random gebruikers
- Pen test of penetratietest (belangrijk voor security)
  - Proberen binnen te breken in de applicatie door kwetsbaarheden uit te buiten
  - Bv.: Wachtwoorden kraken, phishing, scannen van kwetsbaarheden

Ontwikkelomgevingen

Productie

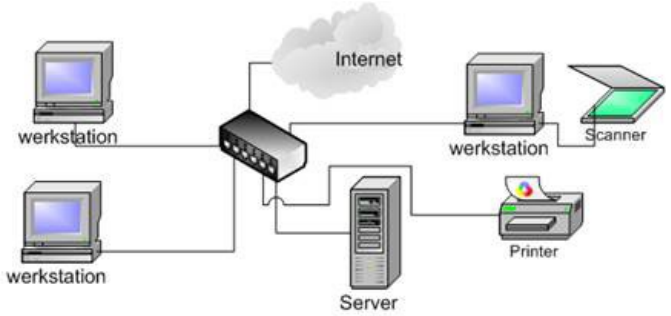
- Deze omgeving is toegankelijk voor de eindgebruiker
- Als de klant zijn akkoord gegeven heeft op de acceptatie omgeving (door te testen), wordt het project op deze omgeving gezet



System engineers beheren deze omgevingen

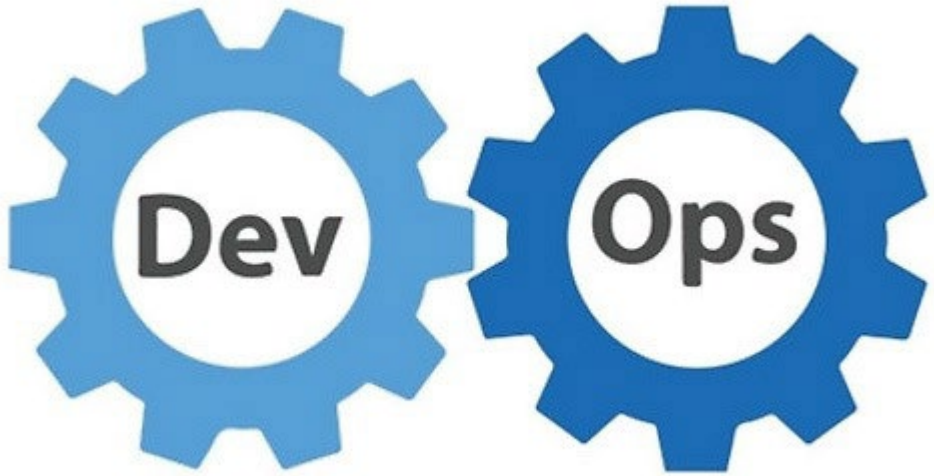
Wat doen system engineers in het beheer van deze ontwikkel-, test- en productie omgevingen?

- Opzetten of zorgen voor de omgevingen (on premise of in the cloud)
- Zorgen voor de overgang tussen de omgevingen DTAP
- Beheer van toegangen tot de omgevingen
- Zorgen voor de infrastructuur voor application development
- Zorgen dat de infrastructuur voorzien is op de *non functional requirements*
- Zorgen dat de applicatie kan draaien op de systemen
- Uitrollen en monitoren van de systemen

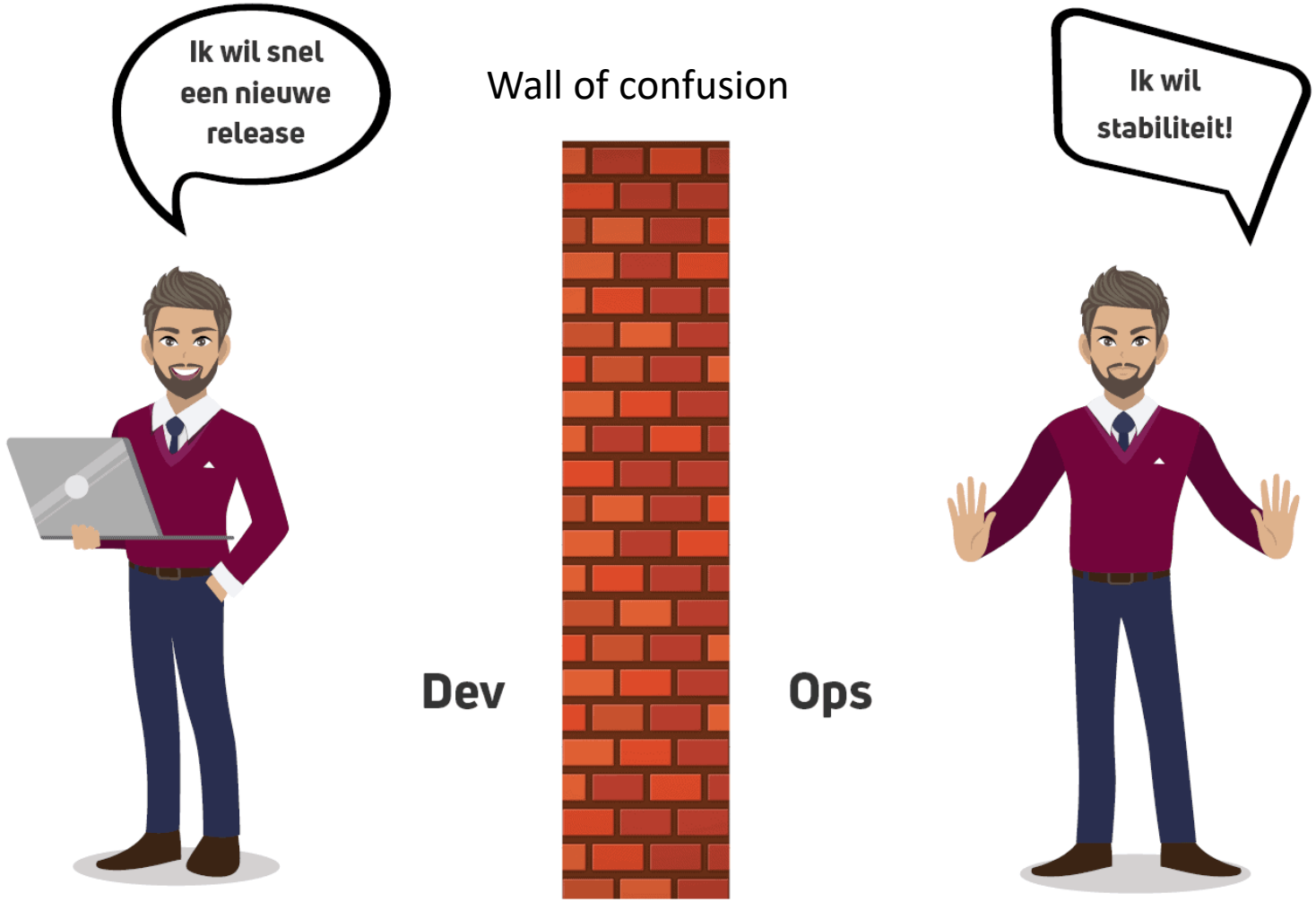


Hoofdstuk 3

DevOps



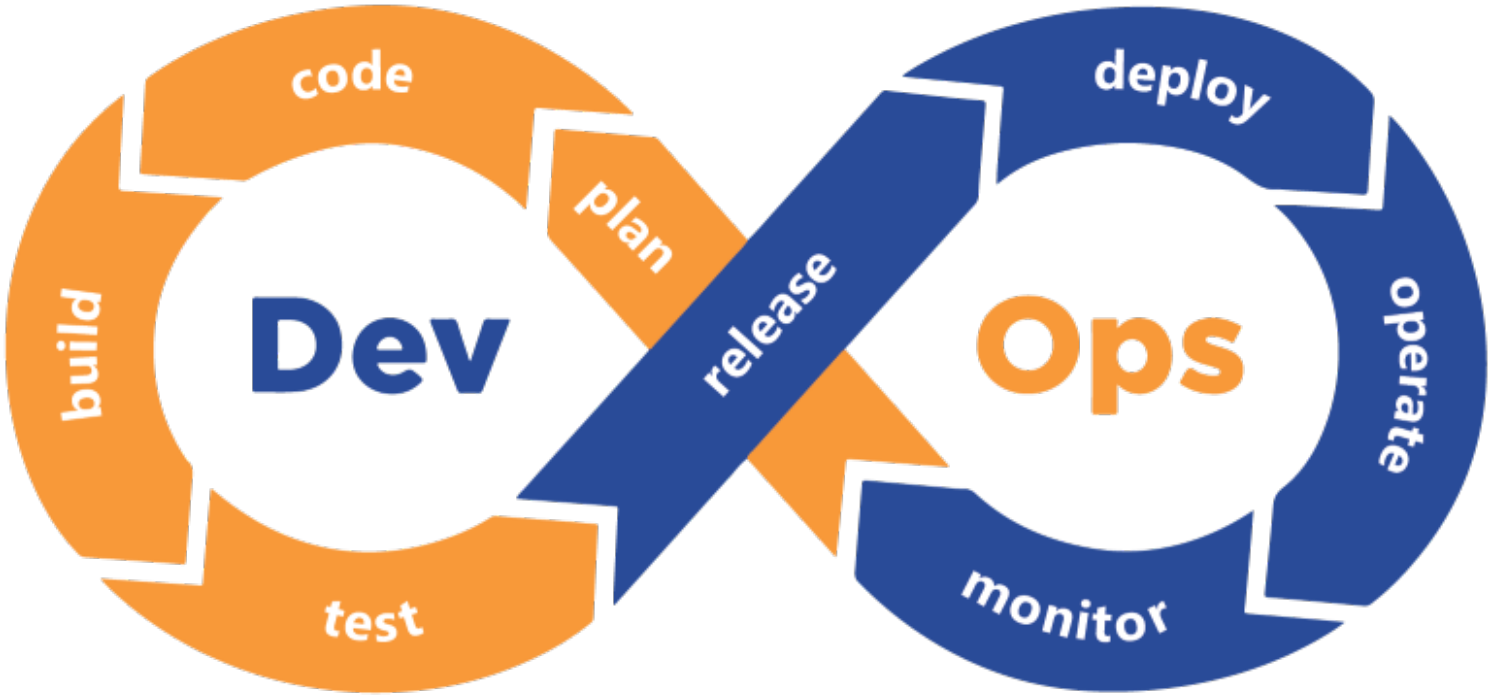
DevOps


Ik wil snel een nieuwe release
Wall of confusion
Ik wil stabiliteit!
Dev
Ops

DevOps

De meest efficiënte en effectieve manier om informatie te delen in een ontwikkelteam is de muur af te breken en met elkaar te praten en nauw samen te werken.

DevOps



https://www.delta-n.nl/devops/wat-is-devops/

DevOps betekenis

- **Nauwe samenwerking** tussen **ontwikkelteams (Dev/Development)** en teams die **het beheer van de applicaties doen (Ops/Operations)** om de ontwikkeling te verkorten en het onderhoud beter te beheersen
- **Automatisering en monitoring** in alle onderdelen bij het bouwen van software, integratie, testen, release tot infrastructuurmanagement. Testen automatiseren waar mogelijk, is belangrijk.
- Devops probeert om **zeer korte cycli** te krijgen van ontwikkeling, met snellere, meer frequente maar wel betrouwbare opleveringen, in lijn met de doelstellingen van de business
- Werkt goed samen met de Agile/Scrum methodologieën

NI.Wikipedia.org/wiki/DevOps

DevOps technologieën

PXL DIGITAL

- Door betere afstemming tussen ontwikkeling en infrastructuur, schrijven ontwikkelaars niet langer specifieke versies voor specifieke hardware of infrastructuurconfiguraties.
- Ze rollen continu nieuwe versies uit dankzij **CI/CD (Continous Integration / Continuous Deployment)**.
- Gebruikte tools zijn bv. Gitlab en Jenkins

DevOps technologieën

- Als je continu nieuwe versies uitrolt, moet je aan de infrastructuurkant ook duidelijke procedures en maximale automatisering hebben.
- Dit kan met container-technologie zoals **Docker** of een automatiseringsplatform zoals **Kubernetes**:

Docker:

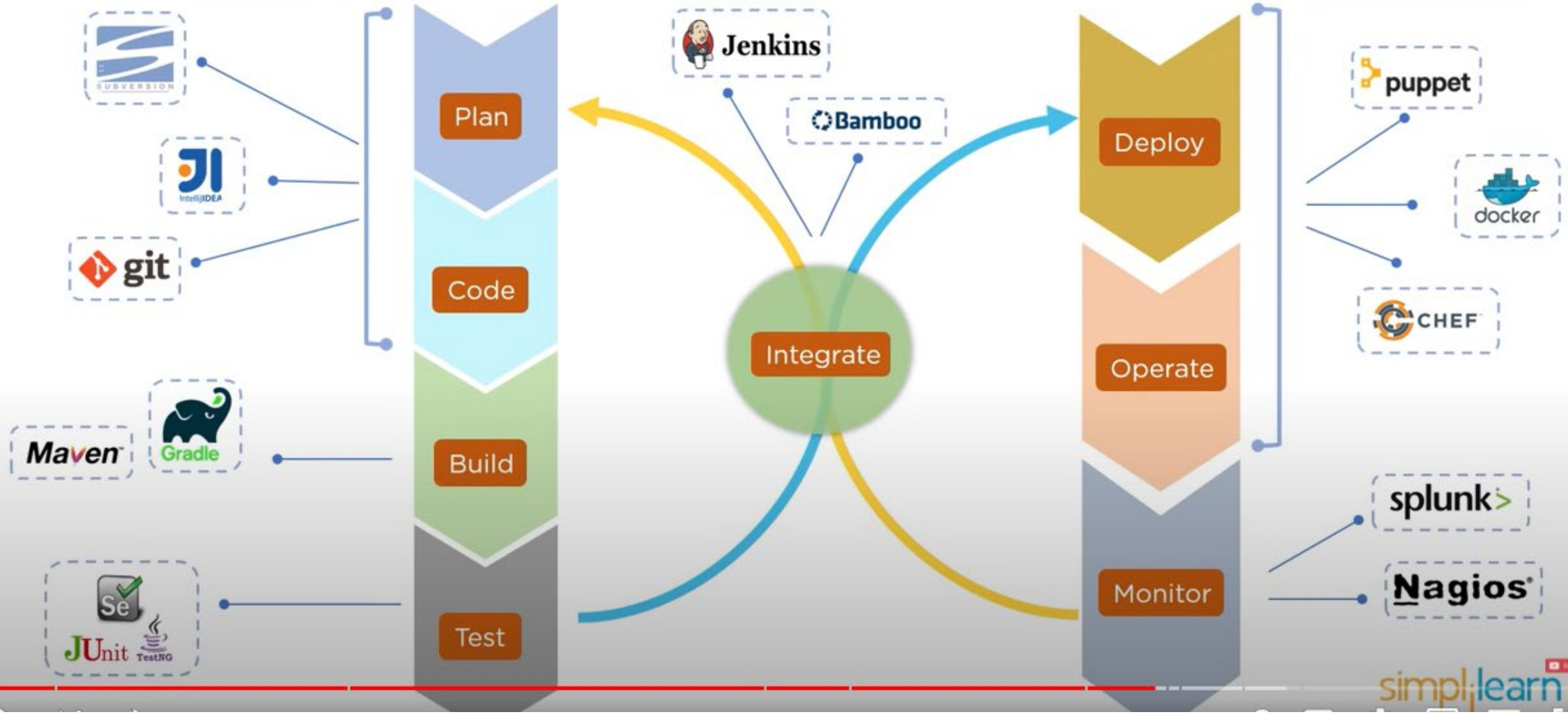
- Vereenvoudigt de levering van applicaties met containers.
- Je moet niet langer rekening houden met de hardware en configuratie parameters.
- Containers bevatten alle benodigde pakketten, kunnen gemakkelijk in zijn geheel getransporteerd en geïnstalleerd worden. Ze zorgen voor een scheiding en het beheer van bronnen die op een computer gebruikt worden bv. Code, runtime-modules, systeemtools en –bibliotheken, …

DevOps technologieën

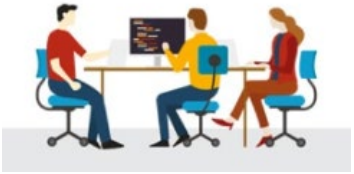
Kubernetes: komt uit het Grieks en betekent “Stuurman”.

- Kubernetes is een systeem voor het beheren van grote groepen containers
- Het zorgt voor orkestratie en automatisatie en regelt de administratie, monitoring en communicatie tussen containers.
- Het is een schaalbare manier om meerdere containers tegelijk te draaien en te laten aanpassen aan tijdelijke toename en afname van gebruik.

DevOps Tools



DevOps voordelen



- **Geen silo's** tussen Development en Operations maar nauwe samenwerking
- **IT-organisatie** wordt beter, **productiever** en minder gevoelig voor fouten

- **Minder tijd nodig** om software te creëren en op te leveren

- **Minder complex om applicaties te onderhouden**

- **Veiligheid:** minder tijd nodig om beveiligingsproblemen op te lossen doordat de informatie over beveiligingsdoelstellingen beter in het dagelijkse werk geïntegreerd is

Devops voordelen

- Bedrijven die DevOps volgen, releasen meer producten en features in een veel kortere tijdsspanne.

- Voorbeelden:

NETFLIX

1000’en dagelijkse
productieveranderingen
voor 1 000 000’en klanten

Iedereen heeft toegang
tot de volledige productie omgeving

amazon

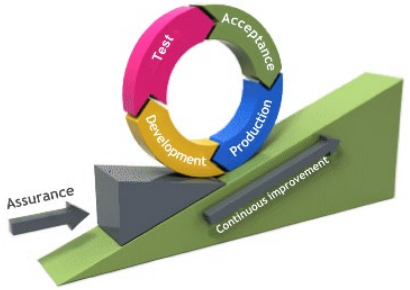
Bij Amazon worden elke 10 sec.
deployments gedaan

We moeten altijd sneller, geavanceerder en innovatiever worden.

Conclusie

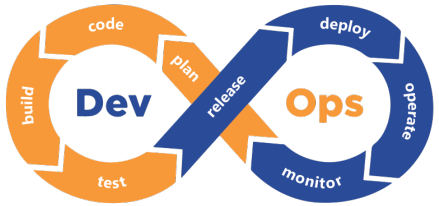
- De **software development lifecycle** geeft de fases aan die je moet doorlopen voor het ontwikkelen en onderhouden van software

  (Planning - Analyse – Design – Implementatie – Test – Maintenance)



- Om deze fases kwalitatief te doorlopen, hebben we verschillende **ontwikkelomgevingen** nodig. => “DTAP” voor het releasen van nieuwe software versies

  (DTAP – Development – Test – Acceptatie – Productie)



- Software moet voortdurend aangepast worden. Dit leidde tot het agile development model. Maar niet alleen agile ontwikkelaars moeten wendbaar reageren, ook het operationele team dat de nieuwe toepassingen moet uitrollen en monitoren. Dit leidt dan weer tot de **DevOps** aanpak.

Duidelijk? Vragen?