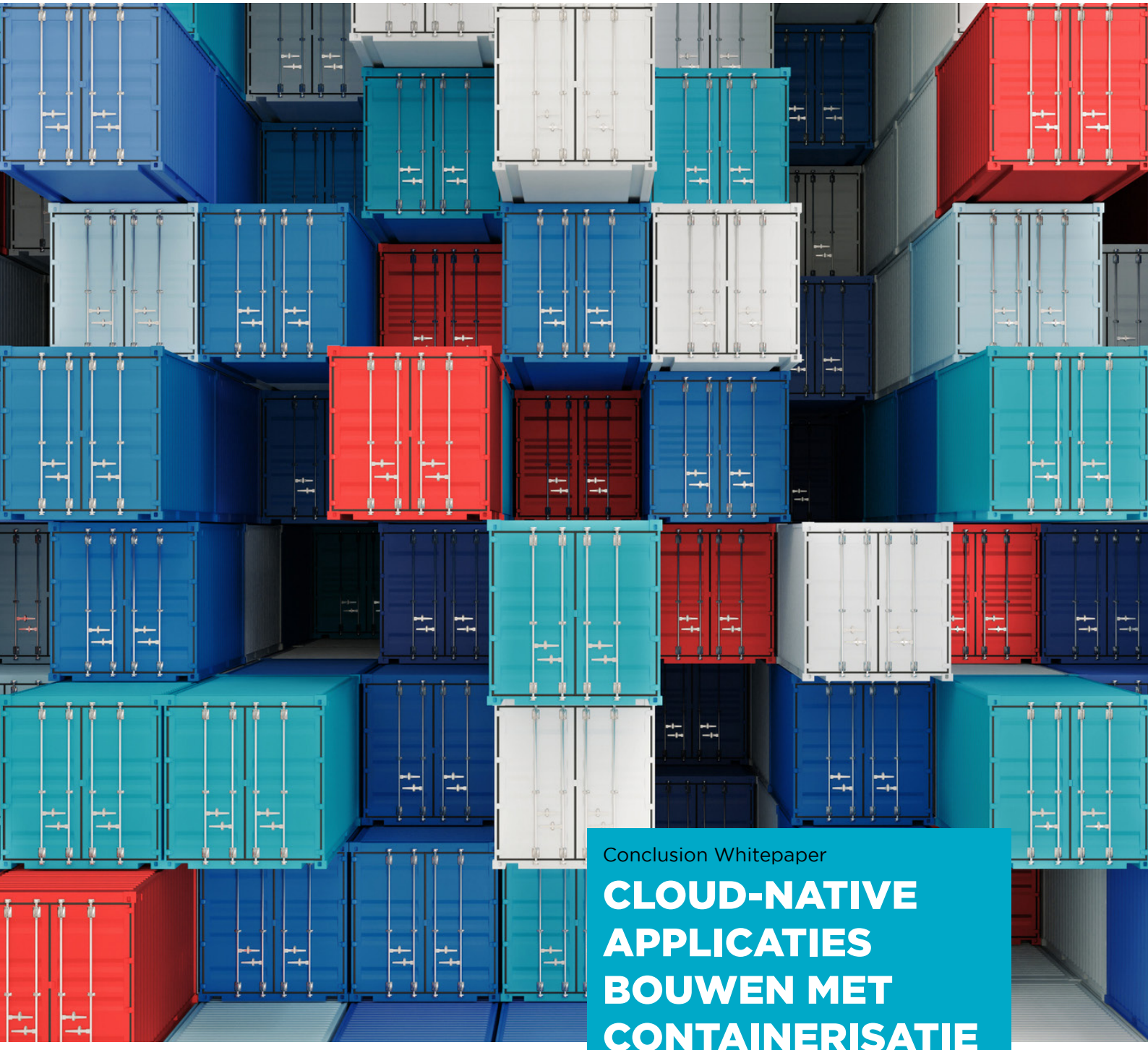
CONCLUSION
XFORCE



Conclusion Whitepaper

CLOUD-NATIVE APPLICATIES BOUWEN MET CONTAINERISATIE

4 belangrijke basisprincipes

INHOUD

Conclusion Whitepaper
CLOUD-NATIVE APPLICATIES
BOUWEN MET CONTAINERISATIE

1. Inleiding

2. Selecteer geschikte applicaties voor containerisatie

Begrijp de applicatie

Bepaal je doelen

3. Volg de basisprincipes voor container design

Principe 1. Kies de juiste base image voor een container

Principe 2. Zorg dat je containers stateless zijn

Principe 3. Maak de status van je containers inzichtelijk

Principe 4. Houd je containers klein en simpel

4. Maak je organisatie klaar voor containerisatie

Werk samen binnen de CI/CD-pipeline

Fail fast, fail often

Blijf dicht bij de business

1. INLEIDING

De populariteit van containertechnologie is sterk toegenomen in de afgelopen jaren. Dat is niet zo verwonderlijk, want containers bieden grote voordelen. Denk aan een korte time-to-market, flexibiliteit, schaalbaarheid, standaardisatie en beheergemak. Hoewel veel organisaties begrijpen waarom het zo belangrijk is cloud-native te worden en containerisatie in te zetten, weten ze vaak niet goed hoe ze eraan moeten beginnen.

Waar het om gaat, is de cloud-native platformen en de gecontaineriseerde applicaties zo te bouwen dat deze naadloos samenwerken. Zo kun je anticiperen op storingen en zowel stabiel draaien als schalen wanneer dat nodig is – zelfs wanneer de onderliggende infrastructuur uitvalt.

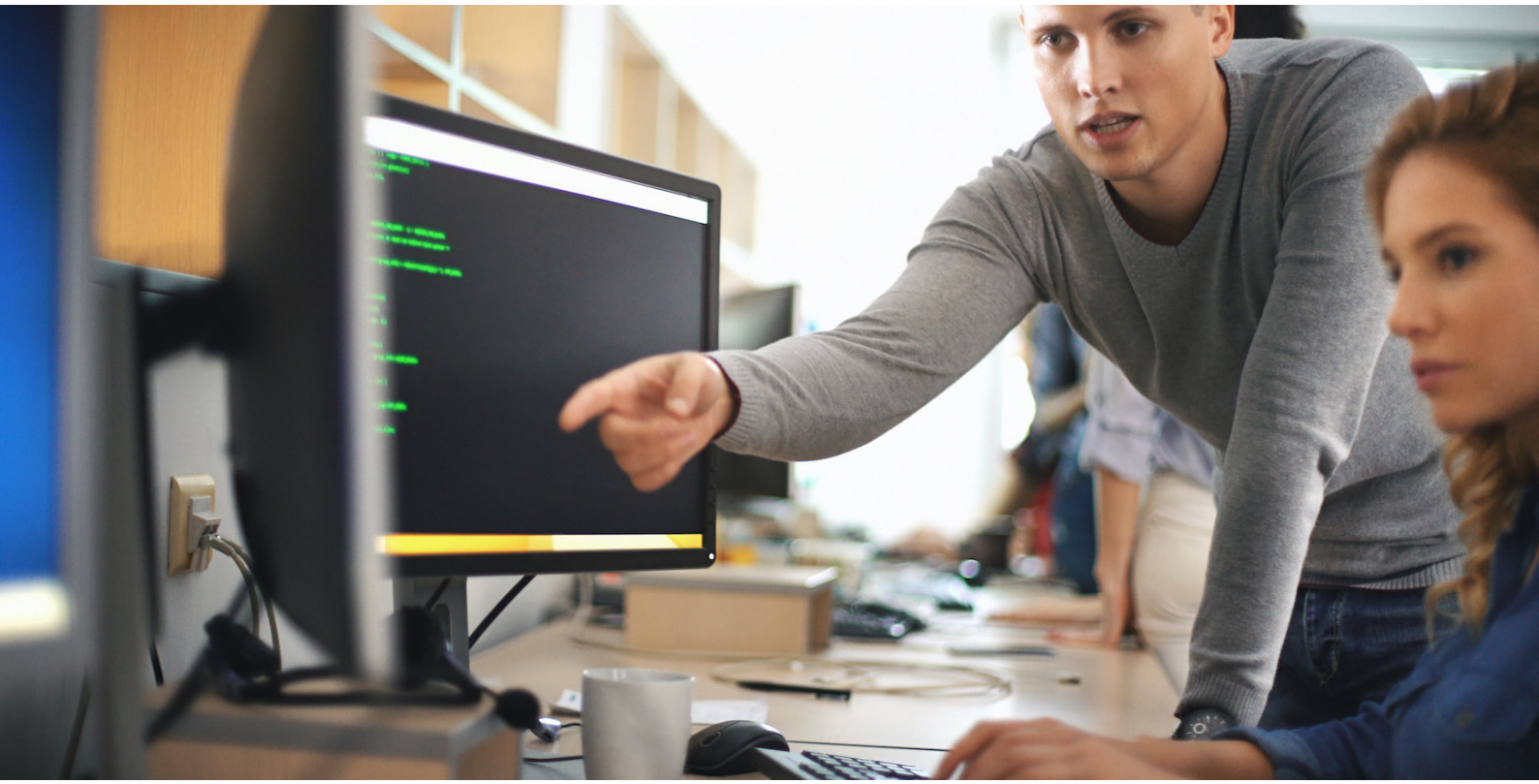
Containerisatie biedt veel kansen, maar het is geen toverwoord. Deze whitepaper beschrijft een aantal basisprincipes waaraan gecontaineriseerde applicaties moeten voldoen om goede cloud-native applicaties te worden. Door deze principes te volgen, zorg je dat je applicaties geschikt zijn voor automatisering in cloud-native platforms zoals Kubernetes. Je voorkomt ondermijning van de voordelen van containerisatie door een ineffectief design en vergroot de kans op succes. Maar net zo belangrijk is het om je organisatie aan te passen en voor te bereiden op containerisering.

2. SELECTEER GESCHIKTE APPLICATIES VOOR CONTAINE-RISATIE

De eerste stap bij het containeriseren van een workload, is de keuze van je applicatie. Tegenwoordig is het mogelijk bijna elke applicatie in een container te plaatsen. Maar een gecontaineriseerde applicatie maken die je effectief kunt automatiseren en orkestreren met een cloud-native platform zoals Kubernetes, vraagt een extra inspanning. Hoe bepaal je welke applicatie daarvoor geschikt is?

Begrijp de applicatie
Wil je een bestaande applicatie naar containers migreren, dan is het allereerst belangrijk om te weten hoeveel moeite dit kost. Daarvoor moet je die applicatie begrijpen. Wat is het precies voor applicatie? Hoe zit het met de opbouw en complexiteit? En hoe is de applicatie verweven in de bedrijfsvoering? Een simpele applicatie die de kern van je organisatie raakt en een grote impact op de business heeft, is misschien niet de juiste applicatie om een containerinitiatief mee te starten. Wat vaak werkt, is relatief makkelijke applicaties als first mover te kiezen. Applicaties die werken op basis van HTTP(s)-calls, zonder lokale dataopslag en met statische configuratie en wachtwoorden zijn uitermate geschikt om als container te draaien en dan ook ideaal om als eerste te migreren. Daarnaast is het makkelijker om open source-applicaties te migreren, dan commerciële off-the-shelf applicaties waarbij je rekening moet houden met de restricties en eisen van een softwarebedrijf.

Bepaal je doelen
Kijk bij het selecteren van de te containeriseren applicaties samen met de business naar de doelen die je wilt bereiken. Hoe draagt de containerisatie bij aan het bedrijfsresultaat? En zijn de doelstellingen wel realistisch? Naast het hoofddoel is er een aantal bijkomende voordelen te behalen, zoals het opbouwen van kennis en ervaring rond containeriseren. Daarmee kun je toekomstige containerinitiatieven beter inschatten en valkuilen voorkomen. Snellere releases maken je organisatie bovendien wendbaarder en helpen je sneller waarde toe te voegen voor interne of externe klanten – met een hogere klanttevredenheid als resultaat. Zorg trouwens dat je resultaten altijd meetbaar zijn en vergeet ook zeker niet om de behaalde successen met je team te vieren.



3. VOLG DE BASIS-PRINCIPES VOOR CONTAINER DESIGN

Net zoals bij ieder ontwerp, zijn er altijd redenen om af te wijken van de geldende basisprincipes. Dat is ook zo bij het ontwerpen van containers. Maar met elke afwijking ondermijn je de kracht van containerisatie. Houd je daarom zo veel mogelijk aan de volgende vier designprincipes.

Principe 1.

Kies de juiste base image voor een container

De basis van elke container is een container image. Vaak is deze base image een uitgeklede versie van een Linux-distributie. Er zijn verschillende soorten base images. Zo is Alpine een light-weight, minimalistische Linux-distributie. Terwijl de Universal Base Image (UBI) van Red Hat bijvoorbeeld een kleinere versie van RHEL is, met getunede images voor verschillende programmeertalen. Let er bij het kiezen van een base image op dat deze past bij het type applicatie dat je wilt containeriseren. Gaat het om een light-weight applicatie met veel custom code die veel tuning vereist en waarvan er tientallen versies draaien? Dan is het de moeite om een maatwerk image te bouwen op basis van Alpine. Is het een applicatie met veel generieke code? Heeft je organisatie weinig specifieke containerkennis? En is enterprise support belangrijk? In dat geval is het verstandig om de UBI van Red Hat als basis te gebruiken bij het bouwen van de containers. Een algemene regel is altijd de specifieke versie van de base image te gebruiken en nooit de laatste versie. Want deze 'latest version' wordt continu geüpdatet. Dat maakt het bouwen van je eigen applicatiecontainer onvoorspelbaar. Je krijgt niet altijd hetzelfde resultaat.

Principe 2.

Zorg dat je containers stateless zijn

Je moet een container op elk moment kunnen starten of stoppen en makkelijk kunnen verplaatsen van het ene containerplatform naar het andere. Update een draaiende container dan ook niet, maar maak een nieuwe versie van de container image en zorg dat die wordt opgestart. Belangrijk daarbij is dat de image zo min mogelijk informatie bevat over het platform en de omgeving waarin hij draait. Neem bij het bouwen van een container dus bijvoorbeeld geen URL's, config files of wachtwoorden mee. Je kunt dit soort gerelateerde informatie meegeven via de environment variabelen van de container, dat is de meest gebruikte oplossing. Het is ook mogelijk om externe files te mounten, zoals bij configmaps of secrets in Kubernetes. Je container mag geen applicatiedata of 'state' bevatten. Bij

het stoppen en starten van een container image gaan data die niet zijn opgeslagen in een gedeelde resource namelijk verloren. Sla daarom de state van de applicatie op in een gedeelde resource, zoals een database, cache, object storage of storage volume.

Principe 3.

Maak de status van je containers inzichtelijk

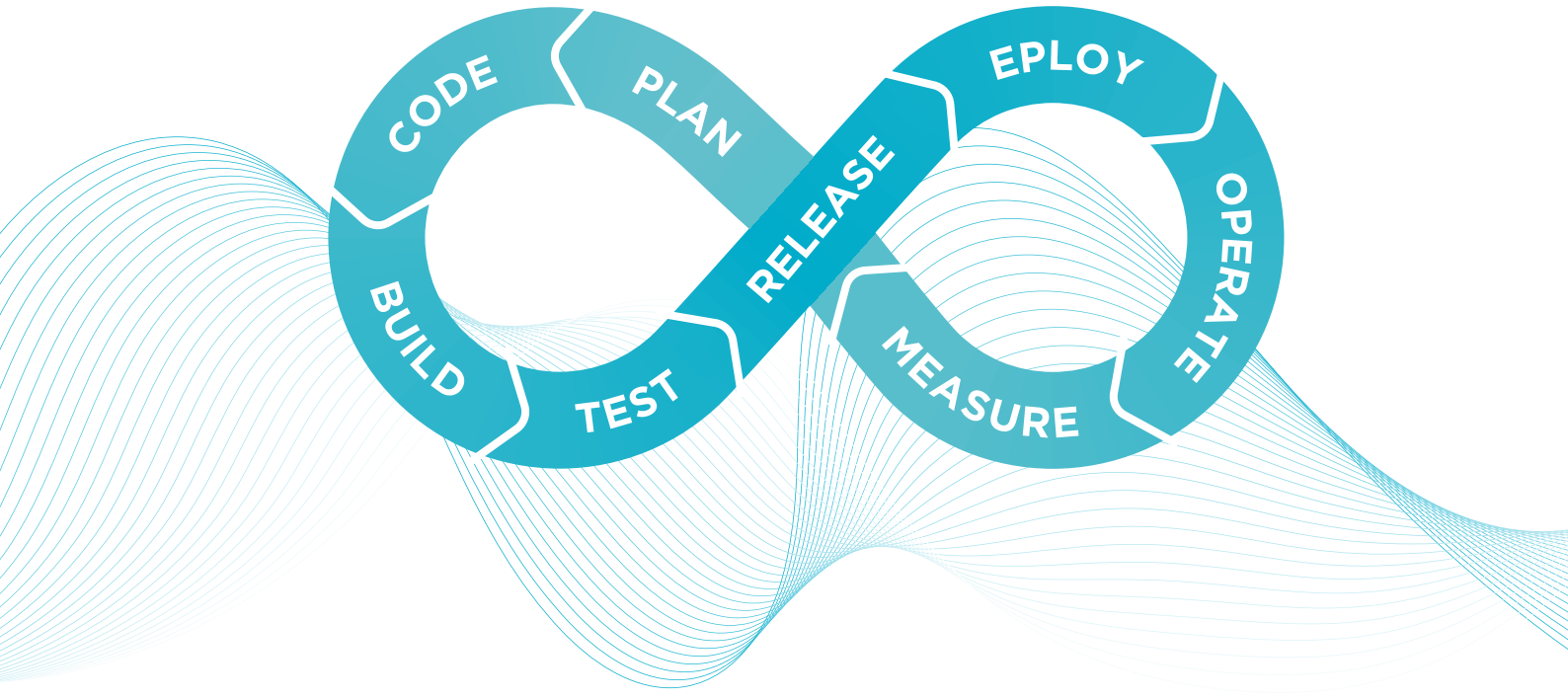
Met containers start je applicaties snel op in een andere omgeving of op een andere machine. Maak het daarom eenvoudig om onafhankelijk van de locatie inzicht in de werking van een container te krijgen. Logging is een van de manieren om het functioneren van een container inzichtelijk te maken. Een veelgebruikte oplossing voor logging is het sturen van logberichten naar de 'standard out' van de container. De meeste platformen hebben op elke node een container draaien die de standard out van de andere containers op die node leest en doorstuurt naar een centrale logopslag. Zo hoeft de applicatie geen eigen mechanismen voor logging te ondersteunen. Behalve dat je met logging inzicht hebt in de werking van de container, wil je ook weten wat de performance van de applicatie is. Hoeveel berichten verwerkt de applicatie bijvoorbeeld op dit moment? En wat zijn de gemiddelde verwerkingstijden? Dit soort informatie maak je via HTTP endpoints beschikbaar. HTTP endpoints stellen standaardoplossingen zoals Prometheus in staat prestatie-informatie op te halen bij verschillende containers en die informatie op te slaan in een centrale database. Voordeel van zo'n centrale database is dat je containers aan elkaar kunt relateren en verbanden inzichtelijk maakt. Veel gebruikte frameworks zoals Quarkus, .NET core of Symfony hebben standaardoplossingen om snel en eenvoudig endpoints op te zetten.

Principe 4.

Houd je containers klein en simpel

Containers worden vaak in één adem genoemd met een korte time-to-market. Maar dat betekent wel dat je nieuwe features snel, efficiënt en doelmatig moet kunnen testen en releasen op productie. Daarbij is het van belang je containers zo veel mogelijk te beperken tot één proces of in ieder geval tot één functie. Op die manier zijn problemen snel te pinpointen en kun je gecontroleerd aan lifecycle management doen, container per container.

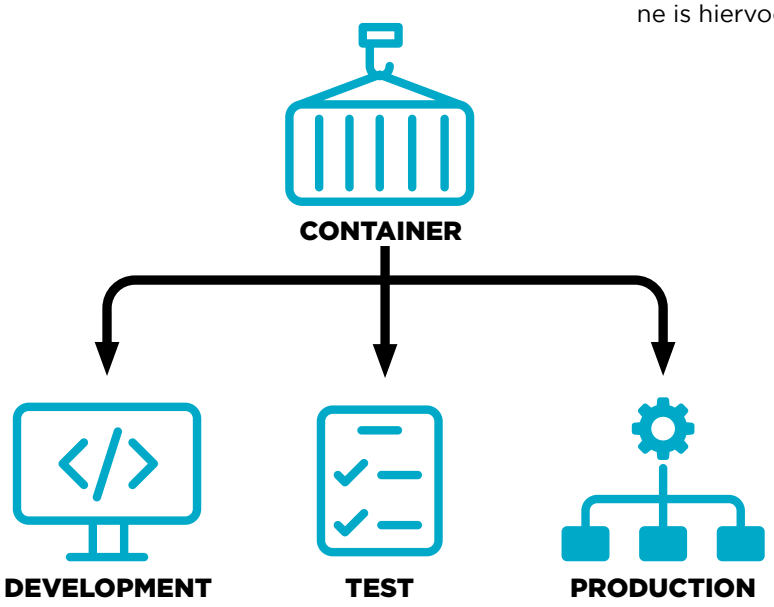
Twee belangrijke succesfactoren bij het life cycle management van je containers zijn Continuous Integration/Continuous Delivery (CI/CD) en automatisering. Neem bij een containerinitiatief de manier waarop je je code naar productie brengt onder de loep en zet een CI/CD-proces op. Zo'n proces kan er als volgt uitzien:



De mate van automatisering binnen het CI/CD-proces is een sleutel tot het succes van containeriseren. Hoe meer je automatiseert, hoe beter. Zorg dus dat je alles wat er in het proces gebeurt zo veel mogelijk in code vat. Door het ontwikkelen, uitbrengen, beheren en functioneren van een containerapplicatie in code en pipelines te verwerken, kun je stabiel, minder foutgevoelig en vaker releasen. De bedoeling is dat je containerapplicaties niet meer verandert nadat je ze hebt gemaakt. Zo kun je containers makkelijk deployen in verschillende omgevingen.

Daarnaast helpt een goed CI-proces je om bij elke wijziging een nieuwe container te bouwen en zowel de functionaliteit van de applicatie als de integratie van die functionaliteit in de bestaande applicatie te testen. Met CI/CD-tools zoals Azure Devops, GitLab of Openshift pipelines automatiseer je het testen van applicaties eenvoudig met standaard bouwblokken.

Doordat je containers met Continuous Integration makkelijk test, kun je aan Continuous Deployment doen. Wil je daar maximaal gebruik van maken, dan moet je zo vaak mogelijk naar productie deployen. Op die manier zijn je wijzigingen minimaal en kun je eventuele risico's goed inschatten. Een geautomatiseerde CI/CD-pipeline is hiervoor een voorwaarde.



4. MAAK JE ORGANISATIE KLAAR VOOR CONTAINERISATIE

Met het volgen van de vier basisprincipes voor container design en de juiste technologische tools ben je er nog niet. Er is meer nodig om echt te profiteren van containerisering. Ook je organisatie moet er klaar voor zijn en dat vraagt de nodige veranderingen.

Werk samen binnen de CI/CD-pipeline
In het traditionele ontwikkelproces worden bijvoorbeeld testers, change managers en security officers er pas aan het einde van de rit bijgehaald. Bij containerisering moeten ze hun werk kunnen doen binnen de CI/CD-pijplijn van de ontwikkelaars. Dat wil zeggen: testen lopen automatisch bij het bouwen van een container, changes worden geautomatiseerd vanuit de pipeline aangemaakt en ook de security scanning is onderdeel van de pipeline. Door deze partijen vooraf hun eisen te laten stellen, al dan niet in de vorm van kant-en-klare modules die je in de CI/CD-pipeline draait, kun je mogelijke issues vooraf en proactief oplossen.

Fail fast, fail often
Containers maken niet alleen de releaseprocessen snel en efficiënt. Als er onverhoopt iets misgaat, kun je een mislukte release bovendien snel terugdraaien. Nieuwe ontwikkelingen maken het zelfs mogelijk om bijvoorbeeld slechts één procent van het verkeer naar de nieuwe applicatie door te sturen. Op die manier hebben eventuele problemen een minimale impact. Doordat je fouten snel signaleert en terugdraait, kun je meer risico nemen. Geef je ontwikkelteams het mandaat om dat risico zelf in te schatten, zodat ze zich verantwoordelijk voelen voor het maken van de juiste keuze.

Blijf dicht bij de business
Containerisatie maakt een eenvoudige integratie met de infrastructuur, geautomatiseerd testen en snel en zelfstandig releasen mogelijk, maar je hebt er weinig aan als het de business niet echt iets oplevert. Geef de business dus directe invloed op de prioritering van nieuwe functionaliteit. Welke features zijn het belangrijkst en willen ze eerst hebben? Dit helpt je ontwikkelteams een realistische inschatting te maken van de (functionele) winst en het bijbehorende risico dat ze daarbij kunnen nemen.

CONCLUSION
XFORCE

MEER WETEN?

Wil je meer weten over de inzet van containerisatie en wat daarbij komt kijken? Neem contact op met Conclusion Xforce.

Jeroen van de Lockand
M: 06- 11 16 56 91
E: jeroen.vandelockand@conclusionxforce.nl

Bronnen (klik op onderstaande links)

PRINCIPLES OF CONTAINER-BASED APPLICATION DESIGN

THE TWELVE-FACTOR APP

BEST PRACTICES FOR OPERATING CONTAINERS - METRICS HTTP ENDPOINT

CONTAINERIZED ARCHITECTURE: COMPONENTS AND DESIGN PRINCIPLES

BEST PRACTICES FOR MIGRATING TO CONTAINERIZED APPLICATIONS