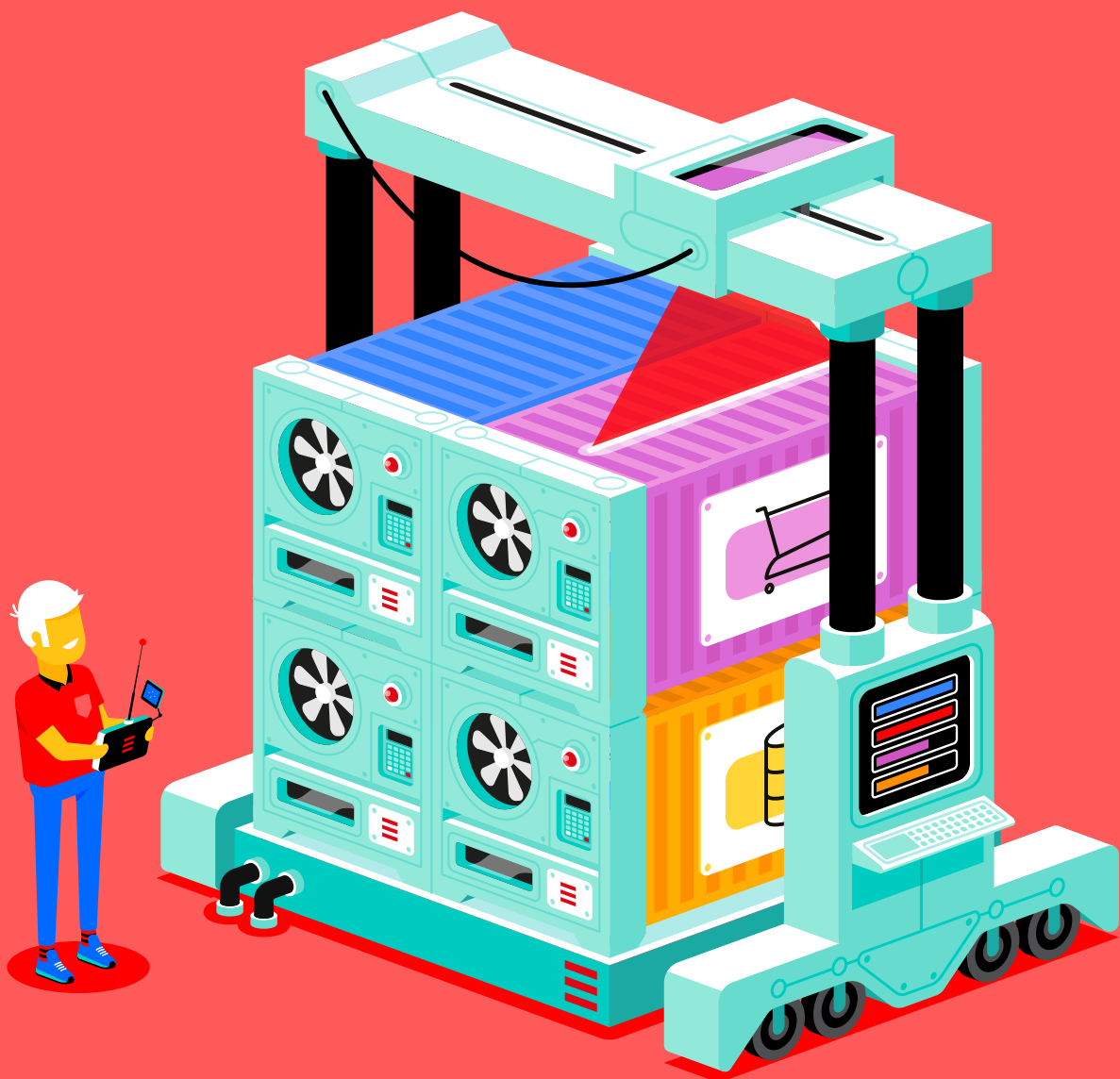


WHITEPAPER

# Top 10 onmisbare ontwerppatronen om te starten met Kubernetes



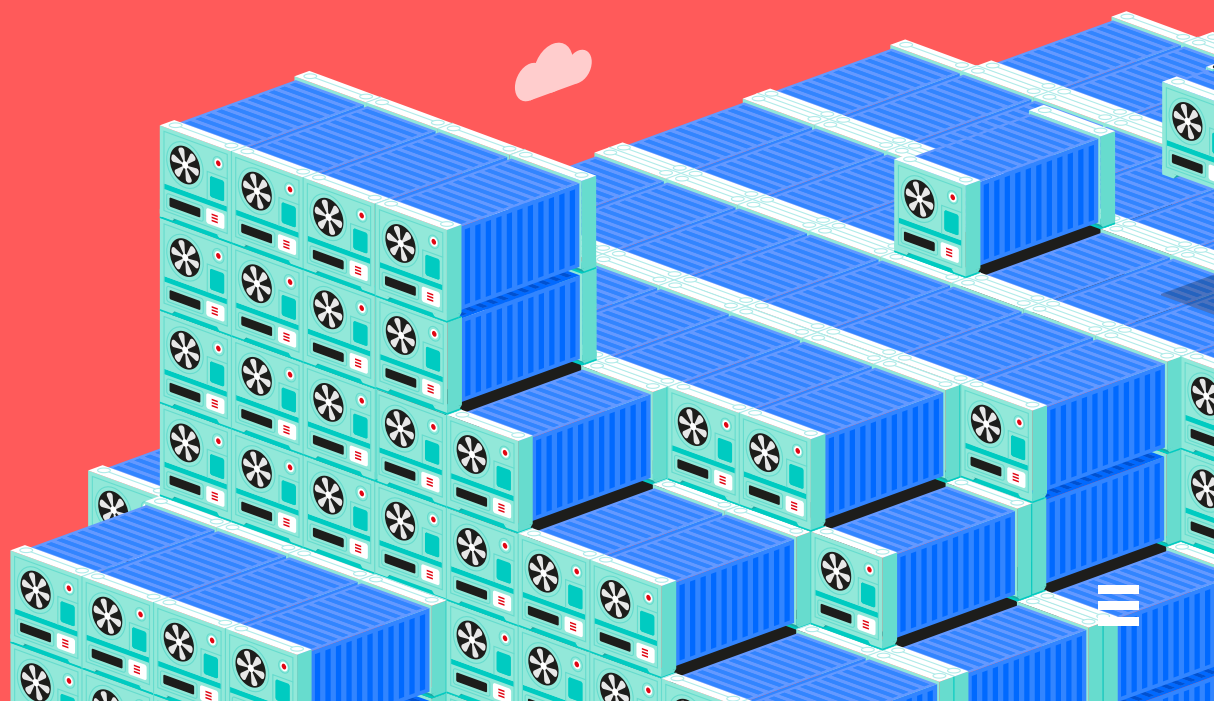
Het softwarelandschap verandert continu en razendsnel. Hedendaagse softwareontwikkeling kenmerkt zich door drastisch snellere releasecycli van applicaties. Waar softwareversies traditioneel volgens een tijdschema werden uitgebracht, worden updates nu steeds vaker de hele dag continu geleverd en direct geïmplementeerd. Gebruikers profiteren hierdoor direct van de nieuwe mogelijkheden of doorgevoerde verbeteringen.

De aanleiding voor deze verschuiving is meerledig. Enerzijds gaat het om technologische ontwikkeling, waaronder de groeiende populariteit van de cloud en containers. Denk echter ook aan stijgende verwachtingen van gebruikers. Zij hebben behoefte aan meer nieuwe functies, snellere probleemoplossing en een product dat zich continu blijft ontwikkelen.

Voor het continu vernieuwen en updaten van applicaties worden zij steeds vaker ondergebracht in containers. En worden systemen ingezet die de implementatie en orkestratie van deze containers automatiseren, zoals Kubernetes. Deze platforms stellen bedrijven in staat tot het gelijktijdig beheren van honderden actieve containers.

Kubernetes is een project dat in 2014 door Google open source is gemaakt. Het container-orkestratieplatform helpt bij het automatiseren van de implementatie, het schalen en het beheren van gecontaineriseerde applicaties.

In deze whitepaper bieden we je handvatten voor het draaien van cloud-native applicaties op Kubernetes. We zoomen in op onder meer cloud-native principes, de belangrijkste begrippen rondom cloud-native en helpen je op weg met het omarmen van Kubernetes.



# Wat is cloud-native?

Het begrip cloud-native is hot en komt met grote regelmaat voorbij. Cloud-native is een benadering voor softwareontwikkeling. Software wordt hierbij specifiek ingericht op een wijze die aansluit op de principes van cloud computing. Maar wat zijn deze principes precies? Google definieert deze als volgt:

## 1. Ontwerp met het oog op automatisering

Automatisering is ook bij traditionele software populair. De cloud maakt het automatiseren van de infrastructuur en de componenten die hierop draaien echter nog eenvoudiger. Een van de cloud-native principes is volgens Google dan ook het ontwerpen van cloud-native applicaties met automatisering in het achterhoofd. Automatisering kan onder meer worden ingezet op het gebied van infrastructuur, CI/CD, het schalen van de omgeving en monitoring en recovery.

## 2. Geef de voorkeur aan stateless

Een van de complexere onderdelen van het ontwerpen van een cloud-native architectuur is het opslaan van de 'state'. De 'state' kan gebruikersdata omvatten zoals producten in een winkelwagentje, maar ook de system state. Een stateless applicatie kenmerkt zich door het niet opslaan van gebruikersdata. Hierdoor is je applicatie niet afhankelijk van data uit eerdere gebruikerssessies. Google adviseert het gebruik van stateless componenten waar mogelijk. Zo zijn stateless componenten eenvoudiger schaalbaar en repareerbaar, maar ook geschikter voor load balancing en rollbacks.

## 3. Kies waar mogelijk voor managed services

De cloud is complex en veel meer dan uitsluitend infrastructuur. Managed services kunnen veel werk uit handen nemen en jouw organisatie in belangrijke mate ontlasten. Hiermee geef je bijvoorbeeld het beheer van managed databases uit handen. Je kunt echter ook via managed services geautomatiseerde back-ups laten uitvoeren en het schalen van de onderliggende infrastructuur uitbesteden. Veel bedrijven zijn echter terughoudend met het omarmen van managed services, onder meer omdat zij een vendor lock-in willen voorkomen. Google stelt dat deze zorgen terecht zijn, maar wijst tegelijkertijd op de grote winst die managed services kunnen opleveren. Zo kan een managed service provider je niet alleen veel tijd besparen, maar ook veel kosten in de vorm van overhead schelen.



#### 4. Kies de juiste verdediging

Security speelt bij traditionele architecturen een belangrijke rol. Organisaties creëren hierbij een netwerk met vertrouwde apparaten en houden niet-vertrouwde apparaten en gebruikers buiten de deur. Perimeter security staat al langer onder druk. Onder meer door de behoefte aan mobiel werken, waardoor het netwerk uit meer externe componenten bestaat. Een cloud-native architectuur maakt dit nog complexer en bestaat uit meerdere diensten die met internet zijn verbonden. Dit maakt perimeter security minder geschikt. Veel cloud-native architecturen zetten daarom in op authenticatie en het minimaliseren van trust tussen individuele componenten, ook indien deze componenten intern zijn. Een dergelijke architectuur is ook bekend als een zero-trust architectuur. Google adviseert dit door te trekken naar bredere maatregelen voor het segmenteren van componenten, zoals rate limiting en script injectie.

#### 5. Blijf de architectuur aanpassen

Een cloud-native systeem is dynamisch en continu in ontwikkeling, waarmee het meebeweegt met onder meer de business of het applicatielandschap. Dat geldt ook voor de onderliggende architectuur. Een belangrijk ontwerpprincipe is daarom dat je altijd blijft inzetten op het verder verfijnen, vereenvoudigen en verbeteren van de cloud-native architectuur. Alleen indien IT-systemen meebewegen met de business blijven zij immers ook op de lange termijn relevant.

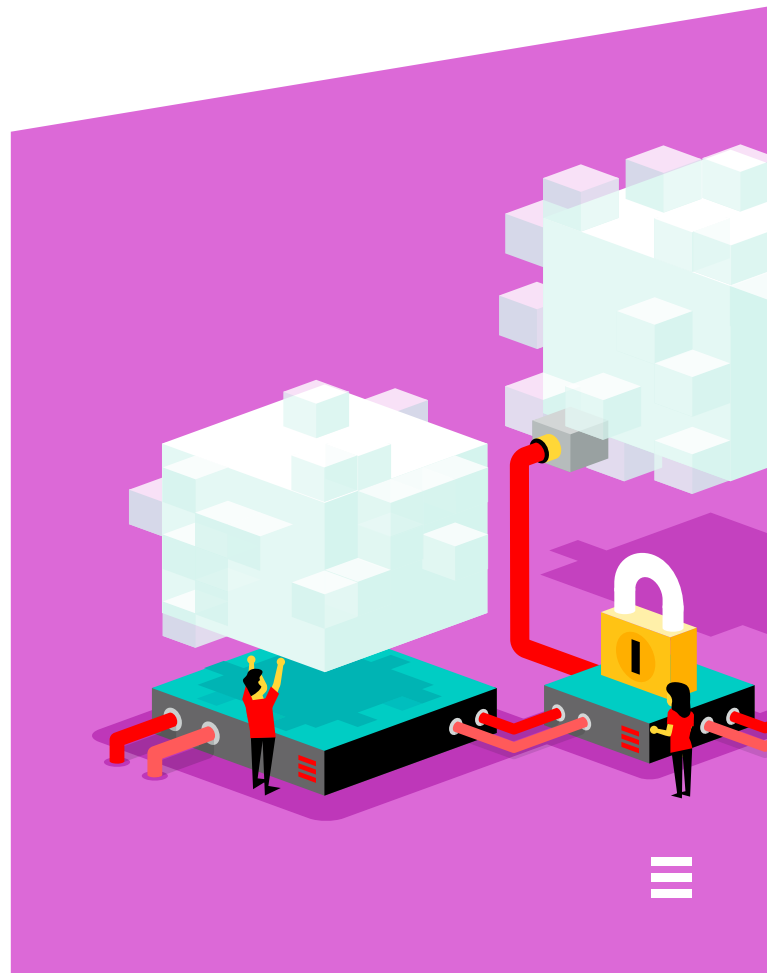
#### Wat is een cloud-native applicatie

De Cloud Native Computing Foundation (CNCF) fungeert als hoeder van cloud-native software en het Kubernetes project. Een cloud-native applicatie moet volgens de CNCF in ieder geval de volgende kenmerken hebben:

- Gecontaineriseerd: Ieder onderdeel van de applicatie is verpakt in een eigen container.
- Dynamisch georkestreerd: Containers worden actief ingepland en beheerd, wat het gebruik van resources optimaliseert.
- Gericht op microservices: Applicaties zijn opgesplitst in microservices, die bijvoorbeeld een specifieke functionaliteit omvatten. Dit verhoogt de portabiliteit van applicaties en vereenvoudigt onderhoud.

Wil je meer weten over het verschil tussen monolithische applicaties en microservices?

Lees dan onze blog '[Van monolithische applicaties naar microservices](#)'.



# Waarom cloud-native?

Steeds meer bedrijven kiezen voor cloud-native. Wat maakt dit voor hen de beste optie om hun architectuur en applicaties op in te richten? En wat is het verschil tussen cloud-first en cloud-native?

## Cloud-first vs cloud-native

De cloud is populair. Veel organisaties die een nieuwe app ontwikkelen zetten daarom in op cloud-first. Bij een cloud-first aanpak verhuizen bedrijven hun apps en workloads zoveel mogelijk naar de cloud. Wie optimaal wil profiteren van de mogelijkheden en voordelen die de cloud biedt, kan echter beter inzetten op cloud-native.

Bij cloud-first worden bestaande applicaties in veel gevallen verhuisd naar de cloud. Cloud-native applicaties zijn echter volledig ontwikkeld voor het maximaal benutten van de voordelen de cloud. Onder meer op het gebied van schaalbaarheid, aanpasbaarheid en portabiliteit. We zoomen in op deze factoren.

## Schaalbaarheid

Kenmerkend voor cloud-native applicaties is het gebruik van software-defined infrastructuur. Hierbij worden [commodity-servers](#) gebruikt die met behulp van software diverse functies kunnen vervullen. De werkwijze zorgt voor een aanzienlijk grotere schaalbaarheid ten opzichte van een traditionele infrastructuur. Zo kan een [commodity-server](#) worden ingezet voor het toevoegen van meer rekenkracht of opslagruimte, maar kan als switch ook worden gebruikt voor het vergroten van de netwerkcapaciteit. Afhankelijkheden van hardware worden verminderd of volledig geëlimineerd.

## Aanpasbaarheid

Veel cloud-native applicaties maken gebruik van microservices, waarbij individuele functionaliteiten als mini-applicaties worden neergezet. Deze mini-applicaties communiceren met elkaar met behulp van API's. Cloud-native applicaties zijn hierdoor zeer aanpasbaar en kunnen eenvoudig meegroeien en –bewegen met de ontwikkelingen binnen een organisatie. Wil je bijvoorbeeld een betaalfunctionaliteit of koppeling met een ERP-systeem aanpassen? Dan hoeft alleen de microservice die hiervoor verantwoordelijk is op de schop, terwijl de rest van de applicatie onaangepast kan blijven functioneren.

## Portabiliteit

Ook het gebruik van containers is kenmerkend voor cloud-native applicaties. Je hebt containers nodig om microservices goed te ondersteunen en je hebt Kubernetes nodig om die containers goed te kunnen managen.

Deze containers bieden belangrijke voordelen en zorgen onder meer voor eenvoudige portabiliteit. Doordat alle afhankelijkheden en benodigde componenten zijn geïntegreerd kan de container eenvoudig worden verplaatst naar een andere cloud- omgeving, een on-premise datacenter of andere gewenste locatie. Dit zonder dat deze verhuizing aanpassingen aan de code vereist.



# Cloud-native glossary

De belangrijkste cloud-native begrippen op een rijtje.



## Microservices

Een microservice-architectuur breekt een applicatie op in verschillende functies of processen. Zij worden als losstaande mini-applicatie ontwikkeld en ook wel een microservice genoemd. De microservices werken samen en vormen gecombineerd de applicatie waarvan de eindgebruiker gebruik maakt. Wil je één functionaliteit aanpassen? Dan hoeft alleen deze microservice op de schop.

## Containers

Een container is een image waarin alle benodigde componenten, afhankelijkheden en instructies voor het draaien van een applicatie - vaak een microservice - aanwezig zijn. Een container kan hierdoor eenvoudig worden uitgerold op een breed scala aan platforms en eenvoudig tussen deze platforms worden verplaatst.

## Docker

Docker is een bekend en veelgebruikt format voor containers. Een runtime instance van een Docker-image bestaat uit drie componenten: een Docker-image, een omgeving waarin deze image draait en een reeks instructies voor het draaien van deze image.

## Kubernetes

Kubernetes is een bekend orkestratiesysteem dat het uitrollen en beheren van gecontaineriseerde applicaties automatiseert. Met de software kunnen containers onder meer worden gegroepeerd en eenvoudiger worden gemanaged. Dat is belangrijk, want indien toepassingen uit meerdere containers bestaan neemt de operationele complexiteit al snel toe. Zeker indien deze containers op meerdere servers draaien.

## Continuous Integration & Continuous Development/Deployment (CI/CD)

CI/CD is een methode voor het ontwikkelen, testen en uitrollen van software-updates naar gebruikers. Vrijwel alle stappen zijn hierbij geautomatiseerd, wat veel tijd scheelt. CI in de afkorting CI/CD staat vrijwel altijd voor Continuous Integration. CD kan staan voor Continuous Delivery, maar ook voor Continuous Deployment. Continuous Delivery betekent doorgaans dat aanpassingen aan een applicatie geautomatiseerd worden getest en klaargezet in een code repository zoals Gitlab. Vanuit hier wordt de code over het algemeen handmatig uitgerold door een expert. Bij Continuous Deployment wordt deze handmatige stap overgeslagen en de code na een reeks geautomatiseerde tests zonder menselijk handelen uitgerold.

# Cloud-native & Kubernetes

Wie cloud-native gaat kan eigenlijk niet om Kubernetes heen. Kubernetes is een containerorkestratieplatform en helpt bij het beheer van containers aanzienlijk.

Het is een open source-systeem dat het onder meer mogelijk maakt containers te groeperen in zogeheten Pods, wat het beheer vereenvoudigt. Dat wil echter niet zeggen dat aan de slag gaan met Kubernetes niet zeer complex is. Wil jij met Kubernetes aan de slag? Diverse ontwerppatronen helpen zeker stellen dat jouw cloud-native applicatie optimaal geschikt is voor Kubernetes. We doen een greep uit de belangrijkste ontwerppatronen zoals Red Hat die beschrijft en stippen deze patronen kort aan:

## De basis

**Health Probe:** het Health Probe-ontwerpprincipe stelt dat iedere container specifieke API's moet implementeren, zodat Kubernetes de applicatie zo goed mogelijk kan observeren en beheren. Zo kan Kubernetes onder meer detecteren of een applicatie beschikbaar is.

**Predictable Demands:** het Predictable Demands-principe zorgt voor een voorspelbaar gebruik van beschikbare resources. Het ontwerpprincipe

stelt dat iedere container zijn resourceprofiel duidelijk moet maken. Denk hierbij aan de applicatievereisten, wat zowel runtime dependencies als vereiste resources kunnen zijn.

**Automated Placement:** Automated Placement zorgt voor een juiste distributie van workloads in een multi-node cluster. Het principe helpt zeker stellen dat iedere workload over de juiste resources beschikt.

## Structuur

**Init Container:** Init Containers zijn gespecialiseerde containers die als allereerste draaien in een Pod, nog voordat appcontainers worden uitgerold. De gespecialiseerde containers omvatten tools en setup-scripts die niet aanwezig zijn in de containerimage.

**Sidecar:** het Sidecar-principe omschrijft hoe de functionaliteit van een bestaande container kan worden uitgebreid, zonder het wijzigen van de container zelf.



## Gedrag

**Batch Job:** dit ontwerpprincipie omschrijft hoe een kleine repetitieve taken geautomatiseerd kunnen worden uitgevoerd en beheerd in een gedistribueerde omgeving.

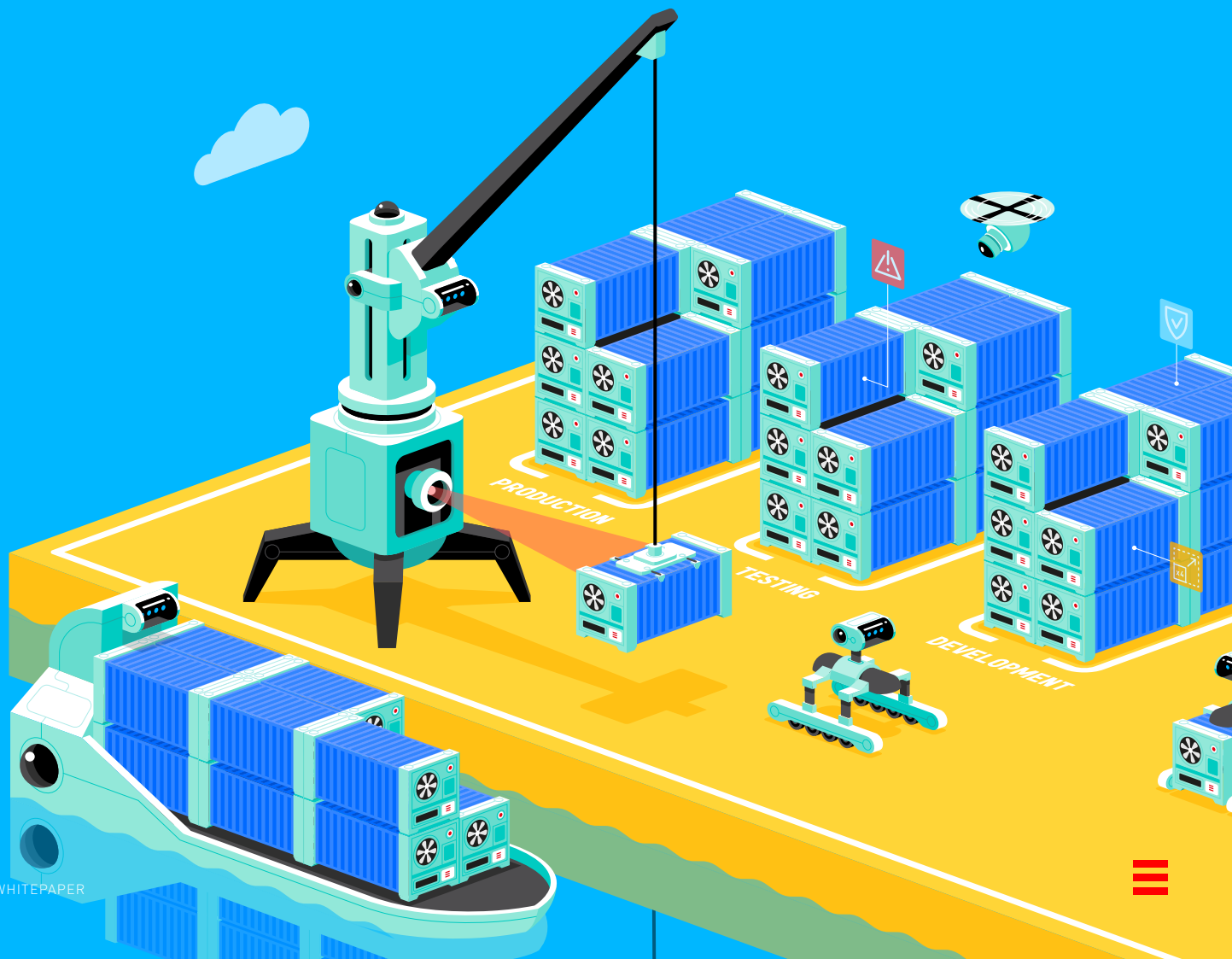
**Stateful Service:** Stateful Service omschrijft hoe gedistribueerde stateful applicaties kunnen worden gecreëerd en beheerd met behulp Kubernetes.

**Service Discovery:** het Service Discovery-ontwerpprincipie definieert hoe klanten toegang kunnen krijgen tot instances die applicatiediensten leveren en deze instances kunnen

## Hoger niveau

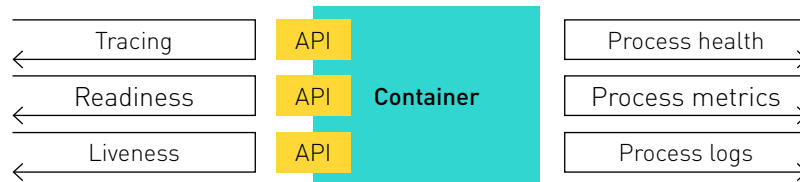
**Controller:** een controller monitort en beheert een reeks Kubernetes-resources, en controleert of hun staat overeenkomt met de gewenste staat.

**Operator:** een operator is een controller die beheerders helpt bij het automatiseren van repetitieve taken gerelateerd aan het beheer van Kubernetes. Het operator-patroon omschrijft hoe code geschreven kan worden voor het automatiseringstaken die Kubernetes standaard niet omvat.

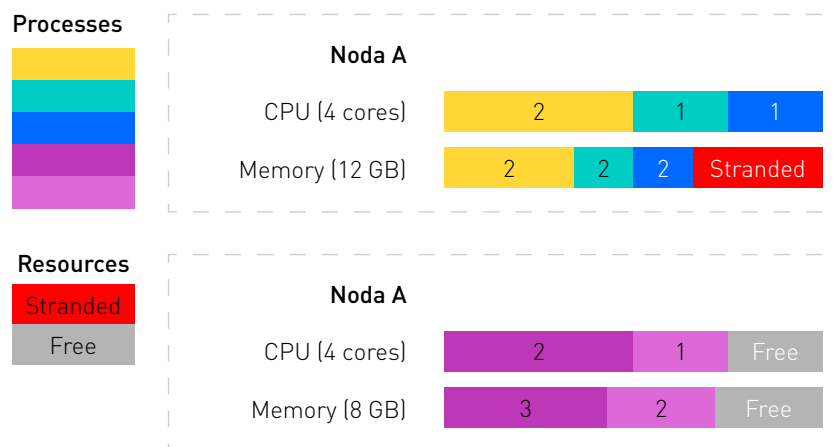


# Top 10 onmisbare ontwerppatronen om te starten met Kubernetes <sup>(1/4)</sup>

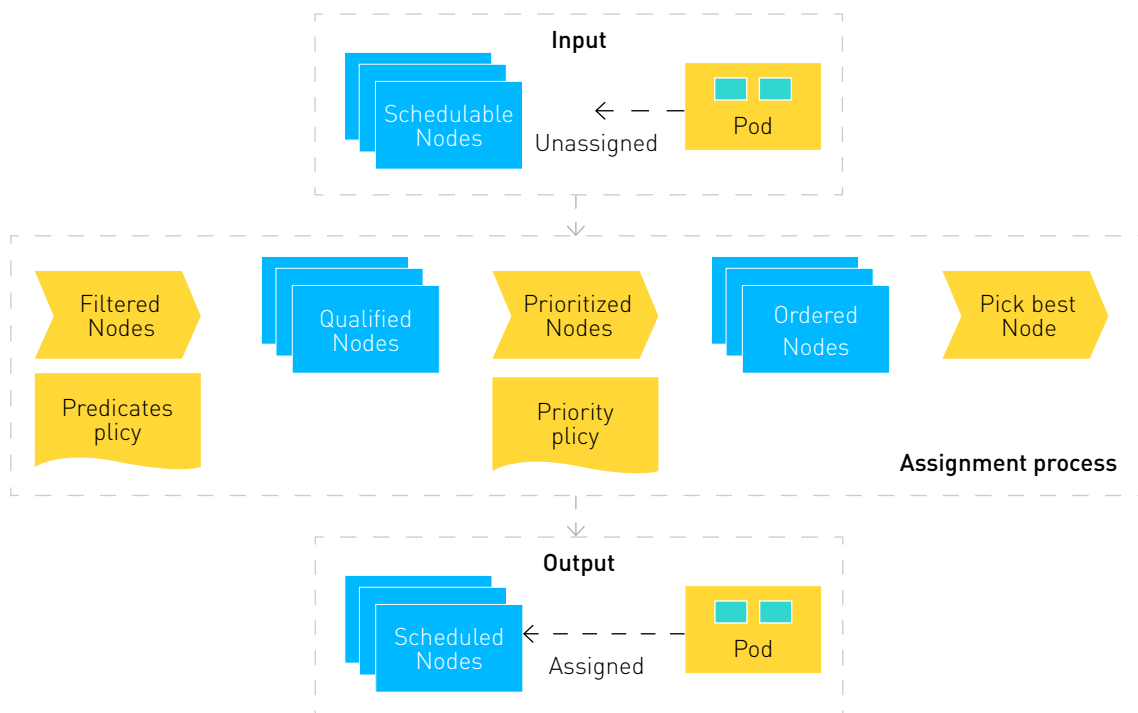
## Foundational



## Health probe



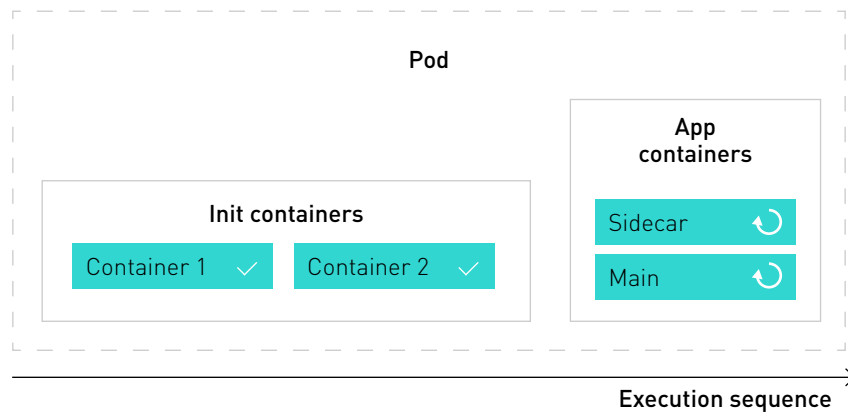
## Predictable demands



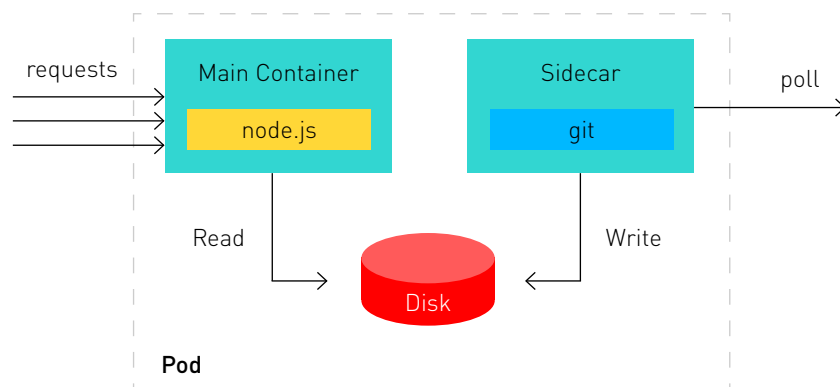
## Automated Placement

# Top 10 onmisbare ontwerppatronen om te starten met Kubernetes <sup>(2/4)</sup>

## Structural



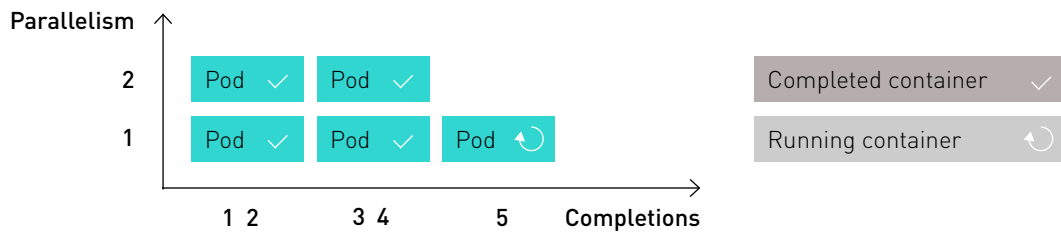
## Init Container



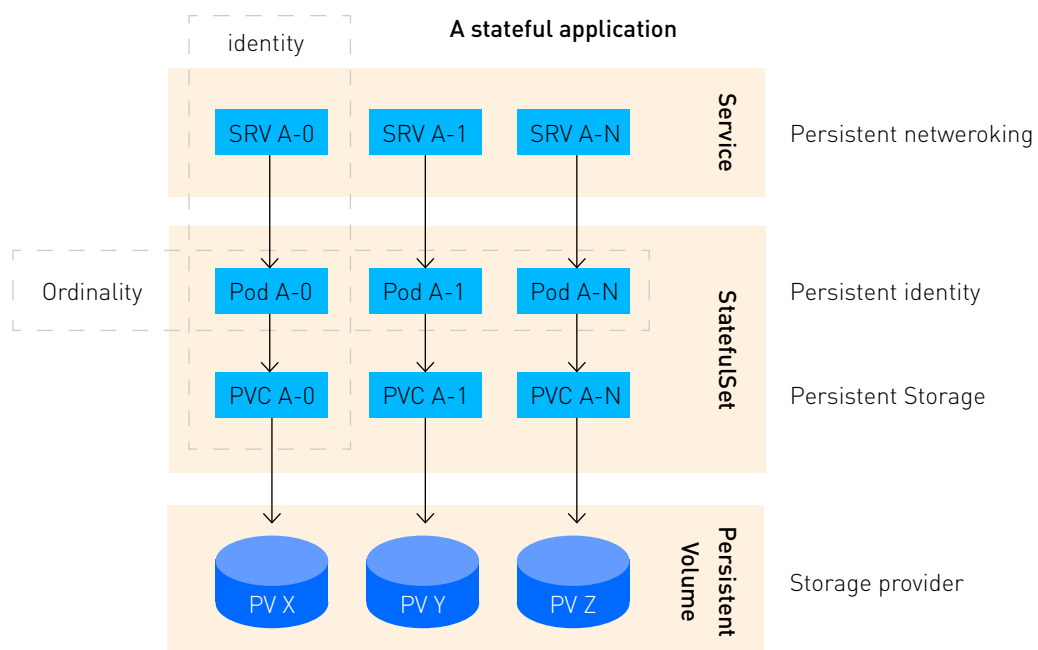
## Sidecar

# Top 10 onmisbare ontwerppatronen om te starten met Kubernetes <sup>(3/4)</sup>

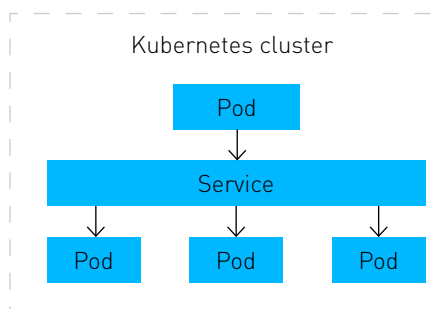
## Behavioural



## Batch Job



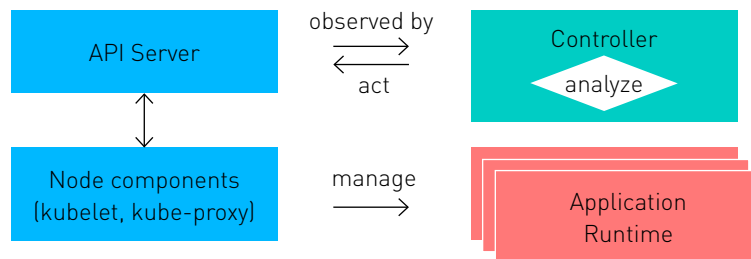
## Stateful service



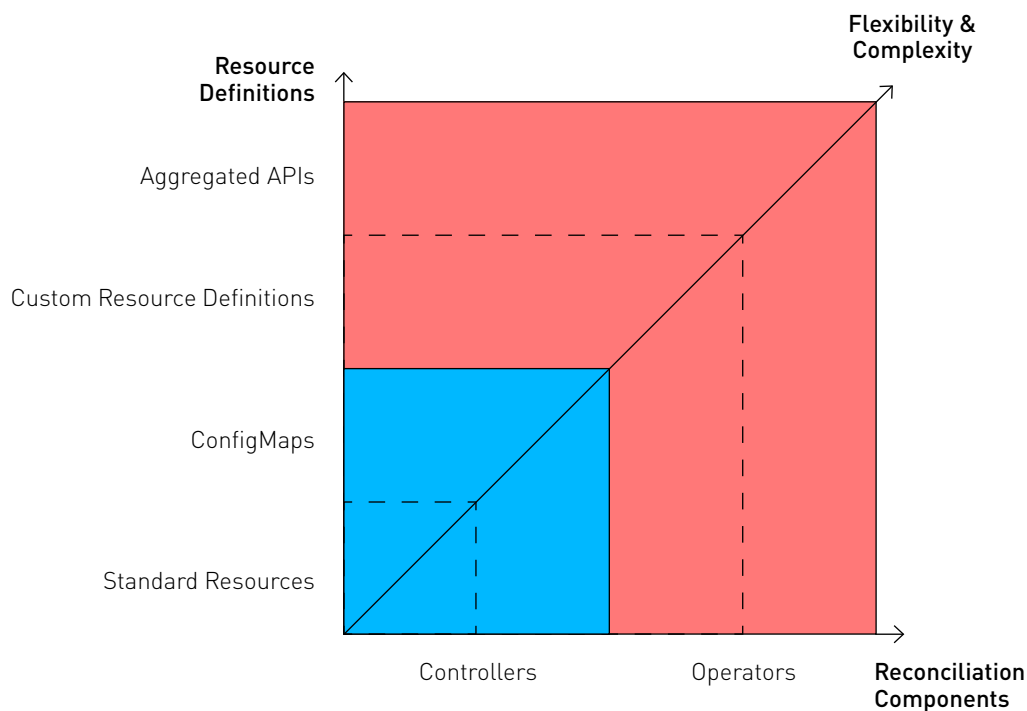
## Service Discovery

# Top 10 onmisbare ontwerppatronen om te starten met Kubernetes <sup>(4/4)</sup>

## Higher-level



## Controller



## Operator

# Waar begin je?

Indien je voor het eerst aan de slag gaat met cloud-native technologie komt er veel op je af. Zo zijn er allerlei aandachtspunten en afwegingen die je aandacht vragen bij het bouwen van nieuwe apps voor containers en Kubernetes. We bekijken enkele belangrijke overwegingen.

## **Zorg dat containerimages zo klein mogelijk zijn**

De omvang van een containerimage is van invloed op onder meer de prestaties, maar ook security. Minimaliseer daarom het aantal packages in een containerimage. Iedere package kan daarnaast in potentie een beveiligingsprobleem bevatten. Doordat microservices via API's met elkaar communiceren kunnen deze problemen ook impact hebben op de veiligheid van de cloud-native applicatie als geheel. Verwijder daarom ongebruikte packages, waarmee je het aanvalsoppervlak minimaliseert. Daarnaast wordt de image kleiner, wat de prestaties ten goede komt.

## **Kies de meest geschikte programmeertalen en frameworks**

Niet iedere programmeertaal is even geschikt voor het ontwikkelen van iedere functie. Cloud-native technologie biedt hierin veel vrijheid. Zo kunnen microservices in uiteenlopende programmeertalen en met behulp van een breed scala aan frameworks worden ontwikkeld. Kies dan ook de programmeertaal en framework die het meest geschikt is voor het specifieke soort functie die je bouwt.

## **Omarm CI/CD en automatisering**

Een cloud-native applicatie bestaat vaak uit een groot aantal microservices, die individueel van

elkaar draaien en met elkaar communiceren.

Daarnaast wordt vaak gewerkt met korte ontwikkelcycli. Ontwikkelaars rollen hierbij met grote regelmaat kleine updates uit naar productieservers. Dit maakt automatisering een cruciaal onderdeel van cloud-native technologie. Het eerder besproken CI/CD is hiervoor populair. Het automatiseren van een groot deel van de stappen bij het ontwikkelen, testen en uitrollen van software-updates naar gebruikers bespaart veel tijd.

## **Integreer inzicht, telemetrie en monitoring vanaf het begin van de ontwikkeling**

De microservices waaruit een cloud-native applicatie bestaat draaien allen in containers. Door het grote aantal microservices loopt het aantal containers snel op. Integreer daarom inzicht, telemetrie en monitoring vanaf het begin van het ontwikkelproces in iedere microservice, zodat je een volledig overzicht hebt. Kubernetes is daarnaast voorzien van ingebouwde mechanismen die containers onder meer automatisch herstarten indien hun werking wordt onderbroken. Hoewel dit de veerkracht van de containers vergroot, kan dit ook problemen verbergen. Monitoring zorgt dat je problemen tijdig opmerkt en kunt verhelpen.

## Onderschat cloud-native technologie niet

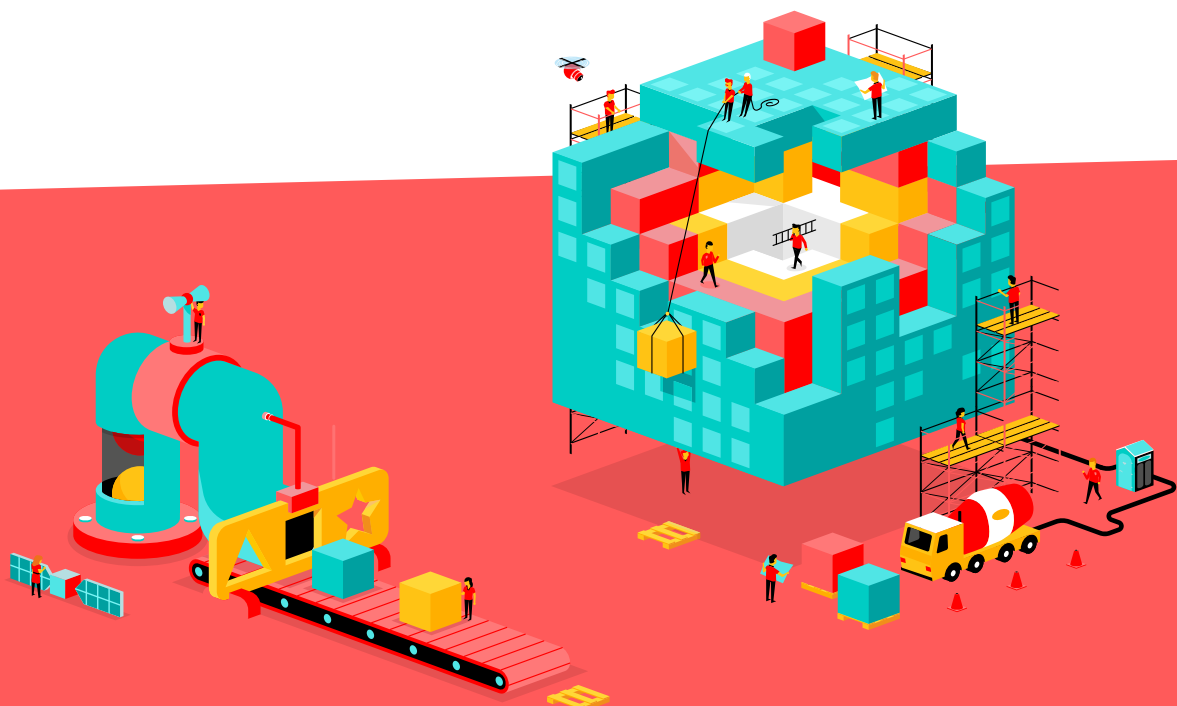
Cloud-native technologie is complex. Houd daarom rekening met fouten en problemen, en leer daarvan. Hoewel veel bedrijven met cloud-native applicaties aan de slag willen gaan, beschikt lang niet iedereen over de juiste experts. Managed services kunnen uitkomst bieden en veel werk uit handen nemen. Ook zijn er providers zoals True waarbij je zelf aan de slag kunt met Kubernetes, maar kunt terugvallen op ervaren experts. Zo kan je zelf experimenteren met cloud-native technologie en ervaring opdoen.

Wil je aan de slag met het draaien van cloud-native applicaties op Kubernetes? De onderstaande drietal webinars helpen je op weg:

→ [Cloud-native apps uitrollen op Kubernetes met behulp van werf](#): in deze webinar leer je welke voordelen het gebruik van werf kan opleveren in infrastructuur-, releasemanagement- en CI/CD-pipelines. werf is een command-line interface (CLI) tool die verschillende populaire oplossingen combineert voor het distribueren van applicaties.

→ [De volgende generatie cloud-native applicaties draaien met behulp van het Open Application Model \(OAM\)](#): Het OAM definieert een gestandaardiseerde manier voor het bouwen en draaien van cloud-native applicaties. In deze webinar word je bijgepraat over de laatste ontwikkelingen rondom dit model.

→ [Cloud-native applicaties uitrollen in Kubernetes met Hazelcast](#): Databases worden steeds groter. Hazelcast is een in-memory computing platform dat de toegang tot data versneld door gegevens op te slaan in het geheugen van een computeromgeving. In deze webinar leer je hoe je Hazelcast kunt inzetten voor het bouwen van cloud-native applicaties.



# Huawei kiest voor cloud-native applicaties op Kubernetes



Allerlei grote spelers gingen jouw organisatie al voor en profiteren inmiddels van de voordelen van cloud-native applicaties. Een praktijkvoorbeeld is Huawei, een multinational en één van de grootste fabrikanten van telecomapparatuur wereldwijd. Het bedrijf telt ruim 180.000 werknemers. Zij gebruiken ruim 800 applicaties, die in meer dan 100.000 virtuele machines verdeeld over een achttal datacenters draaien.

Huawei rolde voorheen applicaties uit op virtuele machines. Het opstarten van een virtuele machines nam echter te veel tijd in beslag. Huawei kiest daarom voor containertechnologie

De interne IT-afdeling van Huawei ontwikkelt nagenoeg wekelijks meerdere nieuwe applicaties. Ontwikkelaars rollen tienduizenden containers uit, met taken die draaien op duizenden nodes wereldwijd. De gedistribueerde aard van het systeem maakt het op een consistente wijze beheren van al deze taken een grote uitdaging.

Huawei zet daarom Kubernetes in. Het cloudorquestratieplatform lost nagenoeg alle uitdagingen op waar Huawei tegenaan liep. Met behulp van Kubernetes zijn de uitrolcycli versneld van weken naar minuten. Ook zorgt Kubernetes voor een efficiëntere levering van applicaties en kostenbesparingen, die in sommige gevallen oplopen tot 20 tot 30 procent.

[Lees hier verder.](#)

## Onze experts denken graag met je mee

Wil jij aan de slag met cloud-native applicaties en Kubernetes?

Onze experts denken graag met je mee en helpen je op weg.

Neem contact met ons op voor meer informatie!