# Slide 2 – Agenda

Wat zijn containers en waarom hebben we ze nodig en wat is docker

Nadien Kubernetes, de architectuur en de verschillende resources

Nadien nog kort over Helm

# Slide 3 Why containers?

Begin en midden jaren 2000

Per app een nieuwe server

Geen idee hoeveel resources, dus liever zo groot/veel mogelijk

Echter weinig van de grote hoeveelheid gebruikt meestal, bv kleine website

# Slide 4 Why containers? Wat was de oplossing?

# Slide 5 Why containers?

Nieuwe virtual machine per app

Vaste percentages van de virtual machine: elke virtual machine gebruikt 25% van de totale resources van de server

Een Server, met daarop een Hypervisor om de VMs te draaien, daarop de VM’s met elk hun OS en daarop een app, database of iets anders

# Slide 6 Why containers? Maar dit had andere problemen,,,

# Slide 7 Why containers?

Het was nog steeds gokken hoeveel van de resources er nodig zal zijn

Er was ook een verlies van resources, aangezien overal dezelfde OS Meestal geinstalleerd wordt, wat eigenlijk niet nodig is

Het OS heeft ook licentiekosten, updates en patches, antivirus installaties die up to date gehouden moeten worden.

# Slide 8 Why containers? Hiervoor is de oplossing…

# Slide 9 Why containers?

Een server, waarop 1 OS draait, linux of windows

Apps nemen in wat ze nodig hebben van resources van de hele OS in ipv fixed size

Dan is er een container per app

Al de rest is free resources, waarop nieuwe containers kunnen draaien

Nog een groot pluspunt ten opzichte van VMs: container starten duurt maar enkele seconden

# Slide 10 Docker Dus Docker

# Slide 11 Docker

Er zijn containers, die klein zijn en zeer snel opgestart

En Docker is de software om deze te runnen

# Slide 12 Docker compose

Docker compose wordt gebruikt om meerdere docker containers tegelijk op te starten, bv frontend, backend en DB.

Hierbij kan je instellen wat de dependencies zijn van andere containers, bv. een backend die een database nodig heeft voordat deze gestart wordt

Ook kan je de argumenten die je gebruikt om een docker container te starten, zoals de ports of de volumes, opslaan in een yaml file

# Slide 13 Kubernetes

Omdat het manueel aanmaken en beheren van deze containers veel werk vereist, hebben we een orchestrator nodig om dit te automatiseren

# Slide 14 Kubernetes

K8s is afkorting van Kubernetes, staat voor helmsman (stuurman)

Docker is bezig met het starten, stoppen, deleten, … van de containers

Kubernetes beheert de docker instances en zegt welke up moeten, down moeten, …

Azure, Amazon, Google, … hebben eigen Kubernetes cluster, maar werken allemaal met dezelfde commands

Kan ook gewoon op je laptop draaien, bv via docker desktop

Auto scaling: meer instances als er meer traffic is, minder als er minder load is

Self healing: Als een instance crasht, wordt er nieuwe opgestart

Zero downtime updates van een app met rolling updates

# Slide 15 Kubernetes Cluster Architecture

Een Kubernetes cluster bestaat uit **master node(s)**, kunnen meerdere zijn en kunnen Linux of Windows zijn, beheren de worker nodes

Bevat een controller manager die verschillende controllers bevat om de resources in de worker nodes te beheren

Bv. Een ReplicaSet controller, Deployments controller, …

Een scheduler om containers op de nodes te plaatsen

Een ETCD database waarin Kubernetes al zijn informatie opslaat

Een Api server die we kunnen aanspreken met REST of met de kubectl command line en met YAML manifests

Meerdere worker nodes bestaan uit: Kubelet: Kubernetes agent, beheert de pods

Container runtime: bv. Docker, beheert de containers

Kube-proxy: beheert het netwerk tussen de verschillende Kubernetes componenten

Ik ga niet te veel in detail, meer is te vinden online

# Slide 16 Kubernetes Controller Pattern

De controllers in de control plane gaan ervoor zorgen dat steeds de desired state wordt behaald

Dus als er een pod is die faalt en er maar 1 meer in de current state is, gaat de pods controller de reconciliation loop doorlopen om terug een pod op starten, omdat er 2 replicas desired zijn

Omdat de aangevragen gescheduled worden, duurt het aanmaken of deleten steeds even, dit werkt als een soort queue system

Dit in tegenstelling tot commands sturen naar Docker. Deze wacht steeds als een commando aan het uitvoeren is tot het gedaan is.

# Slide 17 Kubernetes Resources

Hier is een overzicht van de meest gebruikte resources die we gaan overlopen.

Er zijn er nog enkele anderen, maar deze zijn vooral voor special cases. Die zoek je dus best op als je ze nodig hebt.

# Slide 18 Pod

Het kleinste onderdeel in een Kubernetes cluster is een pod

Deze bevat meestal maar 1 container, dewelke gebuilt is vanuit een docker image

Kunnen er meerdere zijn, wordt er meestal gesproken over een Sidecar (bv om files aan te passen in de pod zelf)

Een pod zal altijd op dezelfde workernode aangemaakt worden waarop deze is gestart.

Horizontal scaling van pods gebeurd met ReplicaSets

Wanneer een pod sterft, wordt een nieuwe gestart (dankzij ReplicaSets en Deployments), nooit dezelfde

Elke pod heeft dus steeds een ander IP adres

Hiervoor heb je service nodig om een “static ip” in de cluster te krijgen

Readiness probe: is de pod klaar om traffic te ontvangen, wordt vooral gebruikt bij deployments en updates

Liveness probe: pod wordt hermaakt als deze probe faalt

Je kan resources instellen dat een pod maximum kan gebruiken, zodat deze de hele cluster niet neer haalt

# Slide 19 ReplicaSet

Een ReplicaSet bevat pods, deze zorgt voor het automatisch herstarten van pods om tot de desired state te geraken, bv 3 replicas

Daardoor ook fault tolerant

Zorgt ook voor de horizontal scaling

Meestal wordt deze niet gebruikt zonder een deployment

# Slide 20 Deployment

Bevat ReplicaSets en gebruikt deze om de autoscaling te doen

Worden ook gebruikt voor zero-downtime deployments

Gebruik steeds een deployment om je app te deployen, zodat je deze features kan gebruiken

StatefulSet is gelijkaardig aan een deployment, maar de pods gaan steeds eenzelfde soort IP, soort naam en dergelijke krijgen. Wordt voornamelijk gebruikt als je databases in een pod zet

# Slide 21 Zero-downtime deployment

Om zero-downtime deployments te bereiken hebben we 4 manieren

Rolling update is de standaard, 1 per 1 down en up

Blue green, met 2 environments, andere wordt actief gezet als deze volledig is getest

Canary, percentage van gebruikers krijgt nieuwe versie, zodat deze kunnen laten weten of alles OK is om dan te wisselen

Rollback: Als er iets fout gaat, kunnen we rollbacken zonder downtime

Om deze deployments te kunnen doen, hebben we Services nodig om de load balancing te doen, dus…

# Slide 22 Services

Pods hebben unieke IP’s, dus we hebben services nodig om deze op een statische manier aan te kunnen spreken

Labels in de yaml worden gebruikt om een pod te linken met een service

Als dus lijkt dat je service niet gekoppeld is, is er waarschijnlijk geen koppeling tussen de labels van de service en de pods

Om snel een pod op je local machine te testen, kan je een port forward open zetten. Niet echt een service, maar heeft wel hetzelfde doel.

# Slide 23 Services

Er zijn 4 soorten services

ClusterIP is om de pod in de cluster zelf te bereiken, bv een backend app die een database moet gebruiken

NodePort is om aan een pod te geraken vanop je local machine, maar is in tegenstelling tot een port forward wel configureerbaar via Yaml, poort tussen 30000-30767

LoadBalancer gebruikt de load balancer van de cloud provider, kan je dus niet gebruiken op je local machine, tenzij je zelf 1 opzet, niet zo simpel

Brengt ook veel kosten mee, omdat een cloud load balancer per service kosten aanrekent

Ingress controller is niet echt een service. Een Ingress Controller, bv Nginx, Traefik of Envoy Proxy, draait een reverse proxy. Dan maak je services aan van het type Ingress Rules die de routing rules bevatten voor reverse proxy

Gebruik dit in Productie

# Slide 24 Storage

Storage kan je overal aan, alle nodes, pods, …

Er zijn verschillende soorten storage resources., beginnen met…

# Slide 25 PV-PVC

**Persistent volume** Static provision: op voorhand gemaakte storage

**Storage class** Dynamic: je kan een StorageClass aanmaken en pas nadien wanneer je het nodig hebt, de storage locatie zelf aanmaken

Claim: Hiermee zegt een pod hoeveel plaats het vereist en de cluster zal dan beslissen welke PV het hiervoor gaat gebruiken

Je zegt dus niet welke PV je wil gebruiken, enkel hoeveel plaats

Er zijn een heleboel types van PVs, zoals in azure, AWS, google cloud, nfs, …

# Slide 26 Storage Config & Secrets

Dan heb je ook nog config maps secrets

Config maps kan je gebruiken om environment variables te injecteren in een pod, config file te laten aanmaken,…

Een Secret is hetzeflde als een config map, maar dan voor passwords, omdat deze encrypted zijn als deze niet in gebruik zijn.

Belangrijk is wel als je secrets gebruikt, dat je de RBAC rollen op een cluster goed instelt, want als een gebruiker een pod kan aanmaken, kan deze ook eender welke secret gebruiken in die pod en dus de waardes uitlezen.

Goed nadenken dus

# Slide 27 Namespaces

Je hebt ook nog namespaces

Namespaces kan je gebruiken om je resources te groeperen, bv alles wat te maken heeft met monitoring, kan je in een monitoring namespace plaatsen

De default namespace noemt gelukkig default

# Slide 28 Kubernetes Resources

Deze resources hebben we allemaal overlopen

# Slide 29 Helm

Omdat als je Kubernetes resources aanmaakt, je een hele hoop yaml files hebt die je 1 per 1 moet installeren en ook niet echt variabelen kan gebruiken in de yaml files, bestaat er een package manager, Helm

# Slide 30 Helm

Helm is dus de package manager van Kubernetes

Even snel over de versies die je online tegenkomt. Versie 2 gebruikte Tiller dat geinstalleerd werd in de Kubernetes cluster, maar dit zorgde voor security issues.

Gebruik dus als je kan versie 3

# Slide 31 Helm

Deze werkt een beetje als docker-compose

Je kan resources groeperen en tegelijk starten

De templates gebruiken de Go language, dus je kan in de templates gebruik maken van if en loop statements, variabelen, …

Je kan dependencies instellen voor de resources

# Slide 32 Next steps…

Onze workshop die hierna volgt

Zelf leren, ik heb dit vooral gedaan op Pluralsight

Veel mee spelen, zo heb ik het meeste geleerd, vooral ook troubleshooting, leren van fouten

# Slide 33 Questions?

Bedankt voor jullie aandacht, zijn er nog vragen?