Automatisering van scaling

Horizontal Insights

Teun van der Kleij (s1188351), Bas Plat (s1184827), Ties Greve (s1128147) en Jeroen Terpstra (s1189144)

Abstract:

In dit onderzoek wordt de automatisering van het schalingsproces binnen Kubernetes clusters bij Vultr onderzocht, een hostingpartij gebruikt door de startup van Ernst Bolt. Het doel is om een methodologie te ontwikkelen voor tijdige en efficiënte opschaling van nodes en nodepools, zodat applicaties consistent presteren zonder overbodige resourcegebruik. Het onderzoek identificeert de meetdata, voorspelt de benodigde resources met behulp van modellen zoals Prophet en Neuralprophet, en test implementaties door middel van prototyping in een gecontroleerde omgeving.

De resultaten tonen aan dat het voorspellingsmodel Prophet een goede balans biedt tussen accuraatheid en snelheid. Daarbij zal het model bepalen of er geschaald moet worden of niet doormiddel van het CPU en RAM gebruik van de verschillende nodes. Verder wordt een gemiddeld opstarttijdraamwerk van tien minuten vastgesteld, wat de basis vormt voor de schaalstrategieën. Daarnaast worden beperkingen in de testomvang en de beschikbaarheid van real-world data besproken. Het onderzoek benadrukt het belang van robuuste datasystemen en beveelt verdere verfijning van voorspellingsmodellen, zoals LSTM-netwerken, aan zodra meer gegevens beschikbaar zijn.

1 Introductie

In dit onderzoeksverslag wordt ingegaan op de automatisering van de scaling binnen Kubernetes van nodes en nodepools bij Vultr, de hosting partij van de startup van Ernst Bolt. Hier zal voornamelijk gekeken worden naar hoe dit proces tijdig gestart kan worden.

Het doel van dit onderzoek is om een effectieve manier te vinden om de scaling van nodepools automatisch binnen Vultr te laten verlopen. Het onderzoek is van belang aangezien er op het moment geen manier is om dit proces tijdig te starten. Daarbij is een groot missend component bij de huidige scaling mogelijkheden, dat de scaling die plaatsvindt ervan uitgaat dat alle nieuwe nodes dezelfde specificaties hebben.

Om deze doelstelling te bereiken, worden een aantal van de aspecten, die binnen dit proces ter sprake komen, onderzocht. Dit proces begint bij het verkrijgen van een beeld van de data die gemeten moet worden voor de scaling. Daarna wordt gekeken naar hoe met deze data voorspellingen gemaakt kunnen worden om tijdig nieuwe nodes op te kunnen opstarten.

Dit onderzoek is van primair belang voor de product owner van de startup van Ernst Bolt. Secundair, kan het mogelijke inzichten geven aan individuen en bedrijven die tegen hetzelfde probleem aanlopen.

Dit onderzoek zal een tijdsduur hebben van twee weken.

1.1 Vraagstelling

1.1.1 Hoofdvraag

De hoofdvraag van het onderzoek luidt als volgt:
Hoe kan het op- en afschalingsproces van nodes over nodepools met Kubernetes geautomatiseerd kunnen worden voor Vultr?

1.1.2 Deelvragen

De deelvragen van dit onderzoek zijn:
Welke data moet gemeten worden om te bepalen hoe op- en afgeschaald moet worden?

- Hoe vaak moet de data gemeten worden?

Welke methode(s) moet(en) gebruikt worden voor het voorspellen van het op- en afschalen?

Hoe wordt het schalingsproces tijdig ingestart?

- *Hoelang duurt het voor nodes en nodepools om aangemaakt te worden?*

Hoe zal het schalingsproces worden gerealiseerd?

2 Methoden

2.1 Literatuuronderzoek

In dit onderzoek wordt gebruik gemaakt van onder andere literatuuronderzoek. Het doel van het literatuuronderzoek is om te onderzoeken wat de gebruikte hosting partij, die de startup van Ernst Bolt gebruikt, al aanbiedt aan informatie over het zelf instellen van het automatisch schalen. Verder zal gezocht worden naar manieren om voorspellingen uit te voeren op basis van gegevens weggezet tegen tijd.

Voor het literatuuronderzoek wordt voornamelijk gebruik gemaakt van betrouwbare onlinebronnen. Onder betrouwbare bronnen worden wetenschappelijke artikelen en onderzoeken verstaan. Elke bron wordt gecontroleerd door te kijken of deze objectief is. Verder is het belangrijk dat wordt verwezen naar andere onderzoeken of bronnen ter validatie dat de informatie als betrouwbaar wordt gezien door anderen. Als laatste wordt gekeken naar de auteurs en de publicatiedatum.

Verder zullen als bronnen ook technische bronnen, zoals als GitHub repositories en de documentatie van geverifieerde partijen en programma's, worden gebruikt.

2.2 Interview

Als tweede onderzoeksmethode wordt gebruik gemaakt van een interview. Er is een interview geregeld met Mischa Mol. Hij kan helpen inzicht te geven over het voorspellen, zoals welke methodes gebruikt kunnen worden en zaken waar rekening mee gehouden moet worden.

2.2.1 Interviewvragen

Voor het interview met Mischa Mol zijn de volgende vragen opgesteld. Deze vragen zijn een richtlijn voor het interview en op basis van de gegeven antwoorden zullen verdere vragen gesteld kunnen worden.

- Welke methodes kunnen gebruikt worden om te zorgen dat op tijd nieuwe capaciteit aangezet wordt?

    - Moet ook rekening gehouden worden met het uitvoeren van deze methodes? Duren deze bijvoorbeeld langer of hebben deze meer resources nodig?
- Hoe kan gezorgd worden dat de voorspellingen betrouwbaar zijn?
    - Is bijvoorbeeld historische data nodig voor betere betrouwbaarheid? Zo ja, welke?
- Wat is een goede manier om eventuele prototypes te testen?
    - Wat is een goede bron voor testdata als dit nodig is?
- Wat moet er gebeuren als er straks echte data in het systeem komt?
    - Is het mogelijk om hier van tevoren rekening mee te houden?
- Is het verstandig/mogelijk om meerdere waardes tegelijk te voorspellen, bijvoorbeeld CPU en geheugen gebruik?

2.3 Beschikbare product analyse

Als derde onderzoeksmethode wordt gebruik gemaakt van beschikbare product analyse. Het doel van deze onderzoeksmethode is om bekende vergelijkbare systemen te zoeken, te onderzoeken welke data bij andere partijen gebruikt wordt om te schalen en hoe deze partijen voorspellen wanneer er geschaald moet worden.

2.4 Prototyping

Het laatste onderzoeksmethode die gebruikt gaat worden voor het onderzoek is prototyping. Door middel van testen zullen gegevens verzameld worden over opstarttijden van nodepools en nodes. Op basis van deze resultaten kunnen conclusies getrokken worden over de tijd die genomen moet worden om op tijd het schalingsproces in te starten.

3 Resultaten

3.1 Welke data moet gemeten worden om te bepalen hoe op- en afgeschaald moet worden?

Het effectief beheren van Kubernetes clusters vereist een goed doordachte aanpak om te bepalen wanneer nodes en nodepools op- of afgeschaald moeten worden. Door gebruik te maken van preventieve schaalstrategieën kan worden voorkomen dat applicaties onderpresteren of resources onnodig worden verspild. Hieronder wordt uiteengezet welke data gemeten moet worden, hoe deze data gebruikt kan worden om schaalbeslissingen te nemen en welke fallback-strategieën kunnen worden toegepast in onverwachte situaties.

3.1.1 Data om te meten voor schaalbeslissingen

Om het schalen van Kubernetes nodes accuraat te plannen, zijn de volgende gegevens essentieel (Ju, 2021):

- **CPU-gebruik**: Het meten van het CPU-gebruik van de nodes geeft inzicht in de belasting van het cluster. Hoge en langdurige CPU-belasting kan een indicatie zijn dat er meer capaciteit nodig is. Daarnaast kan het aantonen dat er te veel nodes zijn aangemaakt als de CPU-belasting langdurig laag is. Hierbij moet ook de request CPU in gedachte worden genomen. Het kan gebruikelijk zijn dat CPU niet gebruikt wordt, maar wel gereserveerd is.
- **Geheugengebruik**: Net als CPU-belasting is het belangrijk om het geheugengebruik te monitoren. Als het geheugengebruik structureel hoog is, kunnen applicaties falen door geheugenuitputting. Dit wijst op de noodzaak tot opschaling. Als er geheugenuitputting plaats vindt, kan Kubernetes ook geen pods meer schedulen.
- **Requests**: Het aantal inkomende requests naar de applicaties binnen de cluster geeft een directe indicatie van de werkdruk. Een stijgende trend in requests kan anticiperen op toekomstige resourcebehoeften. (Coralogix, 2024)

3.1.2 Modeloutput

Bij het modelleren van schaalbeslissingen kunnen de volgende parameters worden voorspeld:

- **Nodes**: De voorspelde hoeveelheid nodes die nodig is voor een optimale werking van de cluster.
- **Totaal aantal virtuele CPU's (vCPU's)**: Het aantal vCPU's dat nodig is om de voorspelde werkdruk op te vangen zonder prestatieverlies.
- **Totale hoeveelheid RAM**: De voorspelde hoeveelheid geheugen die nodig is voor een optimale werking van de cluster.
- **Requests**: De verwachte toename of afname van inkomende requests, zodat resources hierop kunnen worden afgestemd.

3.1.3 Fallback- strategieën

In het geval van onvoorziene omstandigheden of wanneer de voorspellingen niet accuraat genoeg is, kunnen de volgende indicatoren gebruikt worden om schaalacties te initiëren (DavidW, 2024):

- **Responsetijd**: Een toenemende responstijd van applicaties kan wijzen op overbelasting, wat directe opschaling vereist.
- **CPU-gebruik**: Als de CPU-belasting plotseling sterk stijgt en langdurig hoog blijft, kan dit een trigger zijn om extra nodes toe te voegen.
- **Geheugengebruik**: Hoge geheugendruk zonder directe voorspellingen kan een reden zijn voor opschaling om crashes te voorkomen. Hierbij moet ook de request geheugen in gedachte worden genomen. Het kan gebruikelijk zijn dat geheugen niet gebruikt wordt, maar wel gereserveerd is.
- **Ongeplande pods**: Pods die niet kunnen worden ingepland door resourcebeperkingen zijn een indicatie dat er meer capaciteit nodig is (Rabih, 2022).

3.2 Welke methode(s) moet(en) gebruikt worden voor het voorspellen van het op- en afschalen?

Twee veelgebruikte modellen voor tijdreeks-voorspellingen zijn ARIMA en SARIMA (Brownlee, 2020). Deze twee modellen zijn voornamelijk naar voren gekomen tijdens het interview met Mischa Mol, het interview is verder uitgewerkt in Bijlage 1: uitwerking interview Mischa mol. Er zijn ook andere, vaak nauwkeurigere, maar zwaardere modellen zoals Random Forest Regression (AnalytixLabs, 2023), LSTM (Long Short-term Memory) (Alhamid, 2021) en Neuralprophet (Laptev, Triebe, & Hewamalage, 2021). Daarnaast zijn er andere opties zoals Prophet (Brownlee, 2020), AutoTS (AutoTS, 2024), Darts (Time Series Made Easy in Python, 2024) en eenvoudige modellen zoals naive forecasts.

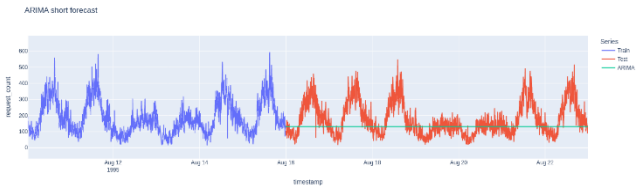
Tijdens het interview (Bijlage 1: uitwerking interview Mischa mol) werd aangegeven dat de focus vooral moest liggen op Prophet en Neuralprophet. Er werd geadviseerd om specifieke/gespecialiseerde versies van modellen achterwege te laten, omdat er nog geen echte data beschikbaar was en deze modellen hierdoor niet goed vergeleken konden worden.

Er zijn verschillende prototypes van modellen gemaakt. De data is afkomstig van het Kennedy Space Center (Dumoulin, 1996). Deze data is bewerkt om de modellen (ARIMA, SARIMA, Prophet en Neuralprophet) te trainen. Oorspronkelijk bevatte de dataset per regel de tijd en een request dat in een periode van twee maanden was verstuurd. De data is omgezet naar een tabel met tijdsintervallen van vijf minuten en het aantal requests per interval. Voor perioden met weinig requests is een tweede versie van de set gemaakt waarin de data aangevuld is door te interpoleren. Hierdoor is het mogelijk onderhoud of downtime niet meegenomen voor modellen die niet met incomplete data om kunnen gaan.

De dataset is opgesplitst vanaf 16 augustus. Ongeveer 30% van de data werd niet gebruikt voor training, maar om de modellen te testen. Deze datum ligt midden in een werkweek, zodat er een goed beeld van weekpatronen ontstaat.

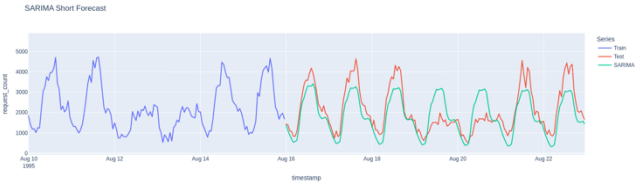
Tijdens het testen is de tijd gemeten die de modellen nodig hadden voor het trainen en voorspellen. De testen zijn binnen een tijdslot van vijf minuten op dezelfde machine uitgevoerd voor de meest accurate data. Na de testen is gekeken welke modellen het nauwkeurigst waren. Bij de onderstaande figuren (1 t/m 4) gelden dat de blauwe lijn historische data is, de groene lijn de voorspelde data en de rode lijn de daadwerkelijke data is.

Figuur 1 toont een grafiek met de voorspelde data vanuit het ARIMA-model. In dit figuur is te zien dat het model geen rekening houdt met de variatie van dagen, waardoor de voorspelde lijn horizontaal blijft.



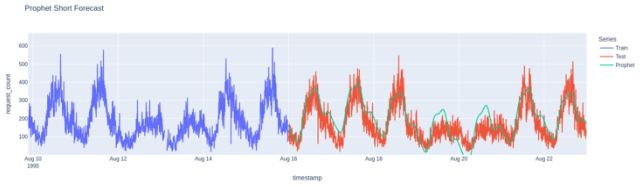
Figuur 1: Grafiek van voorspelde data ARIMA

In Figuur 2 staat de grafiek met de voorspelde data vanuit het SARIMA-model. Dit model was echter zwaarder om te trainen, daarom is dit gedaan met intervallen van een uur in plaats van vijf minuten.

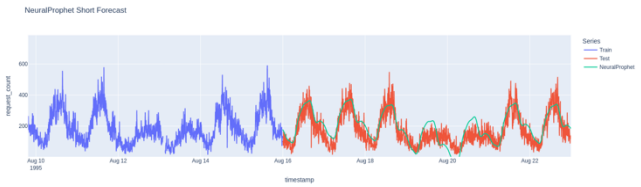


Figuur 2: Grafiek van voorspelde data SARIMA

In Figuur 3 wordt Prophet en Figuur 4 wordt Neuralprophet laten zien. Deze twee modellen hebben een vergelijkbare voorspelling gedaan met een verschil van 1%. Echter heeft Neuralprophet een langere trainingstijd.



Figuur 3: Grafiek van voorspelde data Prophet



Figuur 4: Grafiek van voorspelde data Neuralprophet

De resultaten van de prototyping is te vinden in Tabel 1. Hierin worden de boven geteste modellen vergeleken op basis van de trainingstijd en accuraatheid.

Modelnaam	ARIMA	SARIMA	Prophet	Neuralprophet
Trainingstijd (in seconden)	5.1	37.1	2.9	19.4[1]
Voorspellingstijd – 1 week (in seconden)	0.0	0.0	2.7	0.0
Voorspellingstijd – 16 dagen (in seconden)	0.1	0.0	2.1	0.0
MAPE[2] – 1 week	53%	31%[3]	44%	43%
MAPE[2] – 16 dagen	50%	40%[3]	45%	44%

Tabel 1: Resultaten prototyping

Uit deze resultaten blijkt dat ARIMA het snelste model is terwijl Neuralprophet het accuraatste is. Prophet is minder accuraat maar wel sneller, waardoor Prophet de keuze wordt voor het voorspel model.

3.3 Hoe wordt het schalings-proces tijdig ingestart?

Als opgeschaald moet worden, moet dit proces tijdig ingestart worden. Als gewacht wordt totdat elke node vol zit en daarna opgeschaald wordt, dan zal er een tijdperiode zijn waarin het cluster geen nieuwe requests aan kan. Dit is niet een gewenste situatie.

Door proactief te schalen blijft de belasting op de nodepool in balans en voorkom je overbelasting van de nodes.

3.3.1 Hoelang duurt het voor nodes en nodepools om aangemaakt te worden?

Om een beter beeld te krijgen hoelang het duurt om nodes en nodepools aan te maken, worden er testen uitgevoerd. Het plan bijhorend bij de testen staat verder in deze paragraaf omschreven.

3.3.1.1 Doel

Het doel van het testen van de opstarttijden van nodepools en nodes binnen Vultr is om te bepalen wat de gemiddelde tijd is voor de opstartprocessen. Met deze gegevens kan een inschatting gemaakt worden hoe vroeg te op- of afschaling moet plaatsvinden. Er wordt een verschil gemaakt tussen de opstarttijden op verscheidene dagen en tijdsmomenten.

3.3.1.2 Scope

Er wordt getest met drie verschillende soorten nodetypes vanuit Vultr (Regular performance, AMD High performance en Intel High Performance). Het nodeplan wat gekozen zal worden bij elk type zal bestaan uit twee verschillende specificaties, 2CPU met 2GB RAM en 4CPU met 8GB RAM.

Er zijn drie verschillende starttijden op de dag gekozen om te testen. De keuze voor deze tijden is gemaakt op basis van de start- en eindtijd van een gemiddelde werkdag (vanaf 9:00 en 16:00)[4] en nog één moment verspreid over de werkdag (vanaf 13:00)[4].

3.3.1.3 Testomgeving

Om de testen uit te voeren is gebruik gemaakt van de hosting omgeving op Vultr van Ernst Bolt. Hier is het mogelijk om op clusters nodepools en nodes aan te maken.

3.3.1.4 Testuitvoering

De testen zullen uitgevoerd worden op de eerder beschreven tijdsmomenten. Daarnaast zullen de testen op twee verschillende dagen gedaan worden. De dagen dat de testen uitgevoerd gaan worden zijn vrijdag 13 december 2024 en zondag 15 december 2024.

De testen zijn uitgevoerd op een cluster met als uitgangssituatie één node met één nodepool draaiend op de cluster.

Op beide dagen worden testen uitgevoerd waar één nodepool op aangevraagd wordt met één, twee en drie nodes. Deze testen worden op elk tijdsmoment uitgevoerd.

[1] Getraind op een machine waar een GPU aanwezig was.

[2] MAPE staat voor *Mean Absolute Percentage Error.* Dit is een benchmarking metric die gebruikt wordt om de nauwkeurigheid van modellen aan te geven. Een lagere MAPE duidt op een nauwkeuriger model.

[3] MAPE is lager omdat dit model met intervals van 1 uur getraind is wat betekend dat er relatief minder noise is.

[4] De testen worden uitgevoerd in de tijdzone van CET (+1 GMT)

De testen zijn uitgevoerd door een API-call te maken naar de API van Vultr.

3.3.1.5 Testresultaten

De resultaten van deze testen zijn opgenomen in Bijlage 2: Testresultaten van opstarttijd testen. De testen zijn niet volledig uitgevoerd door externe factoren, uitgelegd in 4.2.1 Testen van het discussie hoofdstuk. In overleg met de product owner is bepaald dat er alsnog een conclusie getrokken zal worden uit deze resultaten.

3.3.1.6 Conclusie

Uit de testresultaten is de volgende ranking gemaakt. Deze ranking is als volgt:

- 1: staat gelijk aan de snelste
- 2: staat gelijk aan de middelste
- 3: staat gelijk aan de langzaamste

	Intel High performance	Regular performance	AMD High performance
Nodepool aanmaaktijd (2cpu2gb)	1	3	2
Node aanmaaktijd (2cpu2gb)	3	1	2
Nodepool+Node aanmaaktijd (2cpu2gb)	1	3	2
Nodepool aanmaaktijd (4cpu8gb)	2	1	3
Node aanmaaktijd (4cpu8gb)	1	3	2

Tabel 2: Ranking van nodetypes

Uit Tabel 2 komt voort dat “Intel High performance” het best uit de test komt. Een punt om hierbij aan te merken is dat de verschillen minimaal zijn.

Uit de testresultaten is gebleken dat er geen significant verschil zit tussen weekdagen en weekenddagen voor zowel het aanmaken van nodepools als nodes. Verder bleek de opstarttijd van nodepools met verschillende hoeveelheid nodes een inconsistente factor op te leveren, waardoor deze niet meegenomen zal worden in de uiteindelijke conclusie.

Wel waren de tijden voor het opstarten zo dicht op elkaar dat de conclusie getrokken worden dat het opstarten van nodes een asynchroon proces is.

Verder had de gehele opstarttijd (van nodepools en nodes samen) geen significant verschil tussen verschillende hoeveelheden aan nodes. Hierbij was het grootste verschil acht seconden op een tijd van 8:48min en 8:40min.

De opstarttijd van nodepools verschilt wel per nodetype en nodeplan. Dit gegeven maakt verder geen verschil, doordat de opstarttijd van nodes samen met nodepools dus niet een significant verschil heeft tussen de nodeplannen en -typen.

De gemiddelde tijd van het opstarten van (een) node(s) met een nodepool was acht minuten en 43 seconden.

3.4 Hoe wordt het schalings-proces gerealiseerd?

Het voorspellende model maakt deel uit van een module die verantwoordelijk is voor het opschalen van het aantal nodes. De architectuur en opzet van deze module vallen buiten de scope van dit onderzoek. In dit onderzoek wordt alleen gekeken naar de implementatie van het voorspellende model binnen deze module.

Voor dit voorspelmodel zijn de talen Python en R het handigst om te gebruiken. Dit komt voort de grote hoeveelheid aan packages die beschikbaar zijn en de grootte van de community voor statistieken (Wainaina, 2023).

Het resultaat van het voorspellende model kan in een nieuwe eenheid die wij Node Ranking Index gaan noemen. Deze eenheid is een ratio op basis van de kracht van een node.

Index	Plan	vCPUs	Memory	Hourly Price
2	vhp-2c-2gb-intel	2	2GB	$0.027 /hr
4	vhp-4c-8gb-intel	4	8GB	$0.071 /hr
8	vhp-8c-16gb-intel	8	16GB	$0.143 /hr

Tabel 3: Node Ranking Index (NRI)

4 Discussie

4.1 Conclusie

Om geautomatiseerd het op- en afschalingsproces van nodes en nodepools in gang te zetten zijn verschillende onderdelen nodig. Het eerste onderdeel is het voorspelmodel. Dit model zal een Prophet model zijn. Deze zal ontwikkeld in Python, omdat Python de packages met nodige statistische functies en modellen voor het voorspellen heeft en de ontwikkelaars die bezig gaan met dit model hebben ervaring met Python en niet met R. Door de gelimiteerde realisatie tijd voor dit model is gekozen voor de taal met de meeste ervaring.

De data die wordt gebruikt voor voorspellingen kan het beste worden opgeslagen in een aparte database. In deze database wordt meerdere keren per minuut informatie opgeslagen over het cluster. Deze data bevat het aantal nodes en informatie per node, zoals het CPU-gebruik, het RAM-gebruik en het aantal requests. Hierdoor kan een accuraat beeld van deze gegevens worden opgebouwd.

Omdat het model invloed heeft op de data waarmee het in de toekomst getraind gaat worden, is het nodig om maatregelen te nemen om foutaccumulatie te voorkomen. Het model bepaalt hoeveel nodes actief zijn en deze beslissingen beïnvloeden de toekomstige trainingsdata. Om te voorkomen dat deze beïnvloeding gebeurt, wordt een combinatie van een fallback-mechanisme en een downward adjustment toegepast. Hierbij wordt gecontroleerd of er pods zijn die niet gescheduled kunnen worden en wordt het CPU-gebruik van de nodes meegenomen.

Door te lage voorspellingen te corrigeren met een fallback en te hoge voorspellingen met een downward adjustment, wordt voorkomen dat het model in een feedbackloop terecht komt. Door dit beveiligingsmechanisme wordt het een stabieler en betrouwbaarder model.

Uit de testen over opstarttijden zijn verschillende conclusies getrokken. Het aanmaken van nodes is een asynchroon proces en heeft geen relaties met de aanmaaktijd van nodepools. Verder, het nodetype “Intel High performance” is het beste uit de test gekomen voor beide nodeplannen. Uit de resultaten kwam de conclusie dat de gemiddelde opstarttijd van nodes acht minuten en 43 seconden. Om genoeg speling te hebben tussen het voorspellen en het opstarten, zal het voorspelmodel altijd minimaal tien minuten in de toekomst kijken.

Deze voorspellingen kunnen opgeslagen worden om een beeld te geven van de betrouwbaarheid van het model.

4.2 Discussie

4.2.1 Testen

De uitgewerkte testen behandeld bij het onderdeel over tijdig schalen hebben beperkte data. Dit heeft te maken met de limitaties van zowel Vultr als tijd.

Voor de tweede testdag was het de bedoeling om een volledige testdag te hebben, maar hier kwamen errors vanuit Vultr naar boven over limieten. De limieten gingen over dat het account te veel nodes had aangemaakt en verwijderd in een te korte periode. Hierdoor was het niet meer mogelijk om via de API nodes en nodepools aan de maken. Hier is overlegd met de product owner om niet verder te gaan met de testen uit te voeren en een conclusie te trekken uit de data die al verzameld was. Oorspronkelijk zouden nog twee dagen meer getest worden, maar deze zijn uit het testplan geschrapt.

Verder was het testen qua tijd ook gelimiteerd. Voor betere data zou over meerdere dagen getest moeten worden met meerdere nodepools om meer vergelijkingen te maken en objectievere data te genereren.

Door beide limieten zijn ook testen van het toevoegen van een node op een bestaande nodepool niet uitgevoerd.

Als resultaat van deze limieten is het gelukt om te testen op één doordeweekse dag en één dag in het weekend. Op de doordeweekse dag is het gelukt om een volledige dag aan data te verzamelen en bij de dag in het weekend niet. De uiteindelijke verzamelde data staat vermeld inBijlage 2: Testresultaten van opstarttijd testen Bijlage 2: Testresultaten van opstarttijd testen.

4.2.2 Onderzochte modellen

Het aantal modellen dat voor deelvraag 2 getest is, is beperkt. Met meer tijd hadden meer modellen getest kunnen worden. Dit had kunnen leiden tot betere resultaten. Toch zouden deze extra testen geen compleet accuraat beeld geven, omdat nog geen data beschikbaar is om deze goed te testen.

4.2.2.1 Weinig expertise

Een tweede interview met een andere machine learning-expert had kunnen helpen. Met een tweede interview had meer feedback en bevestiging verzameld kunnen worden voor dit onderzoek.

4.3 Advies

Wij adviseren om gegevens, zoals geheugengebruik, responsetijd, kubernetes scheduling tijd en het aantal requests, op te slaan in de database. Deze gegevens kunnen waardevol zijn voor toekomstige uitbreidingen van het voorspelingsmodel. Door deze informatie proactief te verzamelen, wordt het eenvoudiger om in de toekomst over te stappen naar een complexer en geavanceerder model.

Bovendien adviseren wij om in de toekomst over te stappen naar een neuraal netwerk, zoals een Long Short-Term Memory-model (LSTM). LSTM-netwerken zijn geschikt voor het verwerken en analyseren van tijdsafhankelijke gegevens en sequenties, zoals responsetijden, planningstijden en trends in verzoeken. Door gebruik te maken van een LSTM-model kunnen complexe patronen en afhankelijkheden in de gegevens beter worden geïdentificeerd en voorspellingen worden verfijnd.

4.4 Vervolgonderzoek

Een vervolgonderzoek kan worden gedaan zodra er echte data beschikbaar is over de load op het cluster. Met deze data kunnen meer geavanceerde en complexe modellen worden getest, zoals neurale netwerken. Deze modellen kunnen mogelijk accuratere voorspellingen maken en helpen om het cluster beter te begrijpen.

Verder zouden in een vervolgonderzoek grotere testen uitgevoerd kunnen worden voor de opstarttijden. Hierbij zouden wij adviseren om de testen uit te breiden met het testen van verschillende en meer testdagen, verschillende nodeplannen voor de verschillende nodetypes en het testen van alleen nodes toevoegen op al bestaande nodepools en hierbij de invloed op de opstarttijd.

Bibliografie

Alhamid, M. (2021, Juni 26). *LSTM and Bidirectional LSTM for Regression*. Opgehaald van Medium: https://towardsdatascience.com/lstm-and-bidirectional-lstm-for-regression-4fddf910c655

AnalytixLabs. (2023, December 26). *Random forest regression*. Opgehaald van Medium: https://medium.com/@byanalytixlabs/random-forest-regression-how-it-helps-in-predictive-analytics-01c31897c1d4

AutoTS. (2024, December 9). Opgehaald van ouro: https://ouro.foundation/posts/chronos/018fcfc1-e07e-7cbb-bbf8-26a166429bda

Brownlee, J. (2020, Mei 8). *Time Series Forecasting With Prophet in Python*. Opgehaald van Machine learning mastery: https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/

Coralogix. (2024, December 17). *Top 6 Kubernetes Metrics & How to Monitor Them*. Opgehaald van Coralogix: https://coralogix.com/guides/kubernetes-monitoring/kubernetes-metrics/)

DavidW. (2024, Maart 14). *Mastering Predictive Scaling in Kubernetes*. Opgehaald van Medium: https://overcast.blog/mastering-predictive-scaling-in-kubernetes-6e09501afbec

Dumoulin, J. (1996, May). *NASA-HTTP.* Opgehaald van Traces In The Internet Traffic Archive: https://ita.ee.lbl.gov/html/contrib/NASA-HTTP.html

Greve, T., & van der Kleij, T. (2025, Januari 14). *Research-Scaling-Automatization*. Opgehaald van https://github.com/Scaling-Insights/Research-Scaling-Automatization

Ju, L. (2021). *Proactive auto-scaling for edge computing systems with Kubernetes.* New York: Association for Computing Machinery.

Laptev, N., Triebe, O., & Hewamalage, H. (2021, November 30). *NeuralProphet: The neural evolution of Meta's Prophet*. Opgehaald van Meta: https://ai.meta.com/blog/neuralprophet-the-neural-evolution-of-facebooks-prophet/

Rabih, E. (2022, Maart 22). *6 Metrics to Watch for on Your Kubernetes Cluster*. Opgehaald van Komodor: https://komodor.com/learn/6-metrics-to-watch-for-on-your-kubernetes-cluster/

Time Series Made Easy in Python. (2024, December 9). Opgehaald van unit8co: https://unit8co.github.io/darts/

Wainaina, P. (2023, Oktober 24). *The Complete Guide to Time Series Forecasting Models*. Opgehaald van Medium: https://medium.com/@wainaina.pierre/the-complete-guide-to-time-series-forecasting-models-ef9c8cd40037

Tijdens de voorbereiding van dit werk is er ChatGPT gebruikt om eigen zinnen/stukken tekst te herformuleren en eigen stukken tekst samen te vatten.

Bijlage 1: uitwerking interview Mischa mol

Welke methodes kunnen gebruikt worden om te zorgen dat op tijd nieuwe capaciteit aangezet wordt? Moet ook rekening gehouden worden met het uitvoeren van deze methodes? Duren deze bijvoorbeeld langer of hebben deze meer resources nodig?

Het klinkt als een tijdsreeks, het gaat waarschijnlijk niet van 0 naar honderd.

Eerste ingeving, het gaat om een tijdreeks. Daarom dus SARIMA, deze heeft seizoensinvloed, anders ARIMA, een model met nog betere accuracy is prophet. Nog een stap verder is neuralprophet, maar deze heeft misschien een videokaart nodig, en komt ook een neuralnetwerk lichtelijk bij. Probeer deze drie methodes eerst. SARIMA doet het niet per se heel goed, dus kijk naar prophet of neuralprophet.

Er zijn ook een aantal hele specialistische gebaseerd op deze modellen, maar zonder trainingsdata is dit heel moeilijk.

Welke beter is tussen prophet en neuralprophet durf ik niet te zeggen.

Hoe kan gezorgd worden dat de voorspellingen betrouwbaar zijn? Is bijvoorbeeld historische data nodig voor betere betrouwbaarheid? Zo ja, welke?

Zorgen voor echte data en data die er veel op lijkt. Verder is er niet per se iets wat je kan doen behalve meer trainingsdata.

Wat is een goede manier om eventuele prototypes te testen? Wat is een goede bron voor testdata als dit nodig is?

Je kan op Kaggle zoeken voor een dataset die lijkt op de data waar je mee gaat werken.

Wat moet er gebeuren als er straks echte data in het systeem komt? Is het mogelijk om hier van tevoren rekening mee te houden?

Het is het handigste om te kijken naar een dataset die veel lijkt op die van jullie. Dat is het meest gemakkelijk wanneer er de daadwerkelijke data gebruikt moet worden.

Wat je ziet is dat met elke data punt dat je hebt wordt het meer betrouwbaarder. Dus misschien wel goed om eerst data te verzamelen (zoals een maand) voordat het model daadwerkelijk gebruikt kan worden.

Is het verstandig/mogelijk om meerdere waardes tegelijk te voorspellen, bijvoorbeeld CPU en geheugen gebruik?

Zou kunnen, maar kijk naar de complexiteit. Hoe meer waardes erbij komen hoe complexer het wordt, dan kom je ook echt veel dichter bij een neuraal netwerk. Dus zou het nu bij een model met enkele output laten.

Hoe kan je het model relatief snel trainen en met wat voor data?

Is een beetje moeilijk, hoeveel trainingspunten zijn er voor de huidige data?

Twee maanden aan data met seasonality van 288.

Heb je iedere 5 minuten aan data punten nodig, waarom niet 10 minuten of 15 minuten. Wel vreemd dat het zo lang duurt op het moment dit komt mogelijk door de hoge seasonality. Alle ARIMA afgeleiden zijn leuk voor het begrip, maar kijk snel naar de andere modellen omdat deze accurater zijn.

Bijlage 2: Testresultaten van opstarttijd testen

Voor elke test staan de waarden in seconden met nauwkeurigheid op milliseconden.

Doordeweekse Testen

Intel High Performance

2cpu2gb	**9:00**	**13:00**	**16:00**
Nodepool	87,872	67,476	103,769
1 Node	522,095	520,988	516,212
Nodepool	62,309	28,439	139,388
2 Nodes	525,403	519,449	519,065
Nodepool	41,700	51,801	110,837
3 Nodes	535,620	522,479	528,669

4cpu8gb	**9:00**	**13:00**	**16:00**
Nodepool	43,792	78,094	134,021
1 Node	517,285	521,543	536,671
Nodepool	86,740	102,003	96,810
2 Nodes	521,988	521,622	531,142
Nodepool	58,020	126,682	126,004
3 Nodes	524,325	525,225	524,483

Regular Performance

2cpu2gb	**9:00**	**13:00**	**16:00**
Nodepool	125,197	90,815	99,143
1 Node	524,769	522,895	526,818
Nodepool	48,026	127,538	90,551
2 Nodes	525,261	525,887	524,672
Nodepool	59,680	123,413	86,078
3 Nodes	528,259	531,067	531,052

4cpu8gb	**9:00**	**13:00**	**16:00**
Nodepool	28,221	32,581	62,879
1 Node	518,673	518,278	527,489
Nodepool	103,083	61,628	94,233
2 Nodes	523,607	527,649	528,189
Nodepool	141,713	66,175	55,465
3 Nodes	528,573	532,870	531,851

AMD High Performance

2cpu2gb	**9:00**	**13:00**	**16:00**
Nodepool	41,718	73,201	46,612
1 Node	527,201	521,523	520,390
Nodepool	68,736	53,223	100,807
2 Nodes	519,123	523,422	522,170
Nodepool	62,454	83,890	89,964
3 Nodes	528,774	527,795	527,480

4cpu8gb	**9:00**	**13:00**	**16:00**
Nodepool	73,197	73,201	81,644
1 Node	524,441	521,523	524,279
Nodepool	36,831	53,223	139,158
2 Nodes	522,599	523,422	532,105
Nodepool	142,231	83,890	147,863
3 Nodes	533,421	527,785	555,875

Weekend Testen

2cpu2gb	**Intel**	**Regular**	**AMD**
Nodepool	96,858	106,552	102,129
1 Node	520,905	530,938	527,507
Nodepool	76,889	108,531	103,298
2 Nodes	529,895	531,447	527,507
Nodepool	71,822	108,153	102,823
3 Nodes	525,702	533,631	532,546

Bijlage 3: Uitleg testproject voor opstarttijden testen

Het testproject dat gebruikt is voor het uitvoeren van de testen voor de opstarttijden is een C# project. Hierin zijn de benodigde endpoints van de Vultr API. Dit project staat in het Scaling-Insights organisatie in de repository “Research-Scaling-Automatization”. Het project heeft de naam “StartUpTimeTestingProject” (Greve & van der Kleij, 2025).

In de tekst hierna zullen methodes aangegeven worden met een onderlijning en italic (*methode*) en properties met alleen italic (*property*).

Properties en methodes

Er zijn verschillende instellingen die gedaan kunnen worden bij de properties:

- *vkeId*: Dit is het ID van het cluster.
- *token*: Dit is de API key van het Vultr account
- *backendImage*: Dit is de link naar de docker image van de backend op de container registery van het account.

Verder staan drie soorten nodeplannen met twee variaties per nodeplan als properties in de class. Dit zijn de nodeplannen waarmee getest kan worden. De property van *nodePlan* wordt gezet met een keuze van één van de nodeplan properties.

Om de testen uit te voeren moet in program.cs de static methode *TestProcess* worden uitgevoerd. Hierbij is het vereist dat de gewenst hoeveelheid pools en nodes, die aangemaakt moeten worden, worden doorgegeven.

Bij de *TestProcess* methode wordt in de console gelogd hoelang het duurt voor de nodes en pools om aangemaakt en verwijderd te worden.

Er is ook een methode aangemaakt waarmee clusters kunnen worden aangemaakt om testen synchroon uit te voeren zonder invloed te hebben op de testen op een ander cluster (*CreateCluster*).