JDriven Tech Radar Najaar 2026

Onze kijk op recente ontwikkelingen in het veld

14e editie

Commit to know.

Develop to grow.

Share to show.

jdriven conflux

Voorwoord

Met veel trots presenteren we de 14e editie van de JDriven Tech Radar.

Het afgelopen jaar stonden AI en digitale soevereiniteit centraal in gesprekken op kantoor en tijdens techconferenties. Deze twee onderwerpen worden steeds meer met elkaar verweven: ontwikkelaars die geïnteresseerd zijn in AI-assisted coding kijken naar het lokaal hosten van modellen en het bouwen van hun eigen harnasses, terwijl bedrijven hun tokenbudgetten inkorten en terughoudend zijn met afhankelijk worden van de grote AI-aanbieders of met gevoelige data delen met ze.

Gelukkig zien we ook dat "tokenmaxing" en "sovereigntywashing" de gesprekken niet domineren. In plaats daarvan is er hernieuwde aandacht voor vakmanschap, architectuur en de keuzes die ten grondslag liggen aan het gebruik van technologie. Tools en platforms die de facto standaarden zijn geworden, worden kritisch bekeken, maar men moet zich ervan bewust zijn dat alternatieven mogelijk onderhevig zijn aan dezelfde risico's.

Ik raad aan speciale aandacht te besteden aan de blips die bewegen of waarvan je zelf verwacht dat ze binnenkort zullen bewegen, wanneer je deze Tech Radar bekijkt. Zoek niet alleen naar een nieuwe tool of technologie om je huidige selectie te vervangen, maar bekijk het grotere plaatje, waarin toegenomen volatiliteit de nieuwe norm is. Dit is een gevolg van de combinatie van huidige geopolitieke instabiliteit en technologische disruptieve innovatie. Het ongemak dat gepaard gaat met deze houding zal verdwijnen naarmate we hier meer ervaring mee opdoen.

Het maakt ons vakgebied van softwareontwikkeling spannender dan ooit tevoren.



Jasper Bogers
Consultant JDriven

Commit. Develop. Share.

Introductie

Onze ervaren specialisten werken dagelijks mee aan tal van softwareprojecten in Nederland en zijn betrokken in wereldwijde community's. Halfjaarlijks komen wij vanuit JDriven bij elkaar om te bespreken wat wij aan nieuwe trends en ontwikkelingen zien. Deze trends proberen wij te vangen in een technologieradar. Elke editie van de radar laat verschuivingen zien t.o.v. een vorige editie. Een verschuiving kan betekenen dat wij een technologie interessanter zien worden waar van toepassing, of juist minder geschikt ongeacht de toepassing. Indien een trend niet meer voorkomt in een latere editie, dan zijn er geen nieuwe ontwikkelingen en/of ervaringen die ons eerdere beeld zouden hebben bijgesteld. In dit document willen we toelichten welke verschuivingen we de afgelopen periode hebben waargenomen, om op basis daarvan weer richting te geven aan wat wij inzetten en aanraden.

Tech Radar

Het idee voor het opstellen van een Tech Radar komt voort vanuit Thoughtworks. Zij dragen al langer periodiek met een radar hun visie uit op nieuwe trends en ontwikkelingen. Bovendien raden zij iedereen aan een eigen radar op te stellen.

Bij JDriven onderschrijven we dat. Het opstellen van een Radar is een leerzame en waardevolle ervaring waarin onderling kennis wordt gedeeld en een technisch bewustzijn wordt gecreëerd. Wij geloven erin dat specialisten zelf in staat moeten zijn om het gereedschap voor hun werkzaamheden samen te stellen. Met het opstellen van een radar faciliteer je discussies over technologie, om als organisatie de juiste balans te vinden in wat voor risico's en voordelen innovatie kan opleveren. Wij kunnen je helpen dit op te starten binnen je organisatie. Laat je teams elkaar inspireren tot innovatie en gezamenlijk komen tot een set aan technologieën en technieken die de ontwikkeling in je bedrijf versnellen.

Indeling

Een radar bestaat uit kwadranten en ringen, met daarbinnen blips om interessante technologieën en technieken aan te duiden.

De kwadranten verdelen de verschillende onderwerpen in categorieën.

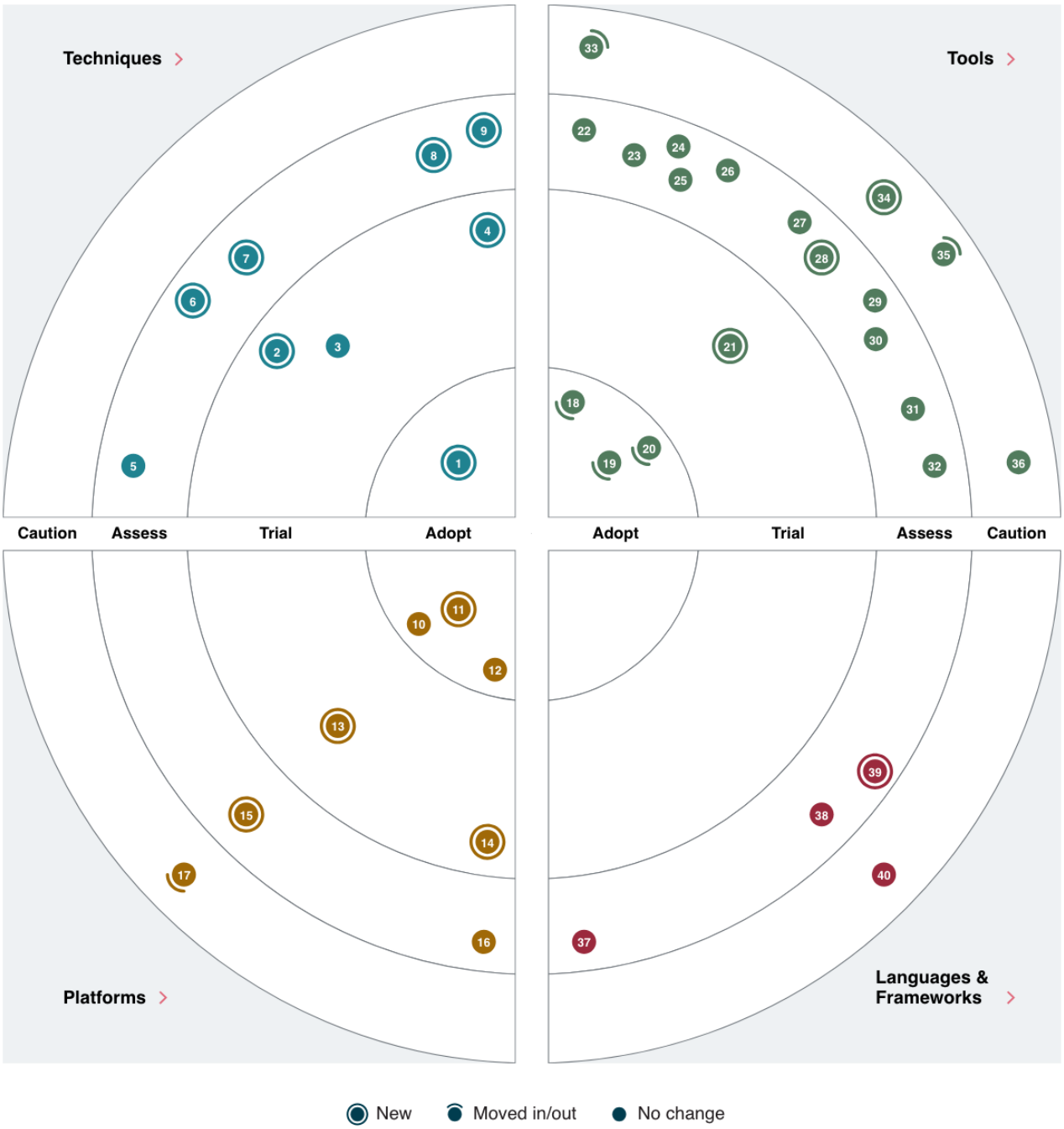
- **Technieken** die je helpen betere software te maken
- **Platformen** waarop je software kunt uitvoeren
- **Tools** ter ondersteuning van je ontwikkel- en delivery-proces
- **Talen & frameworks** die je ondersteunen bij de ontwikkeling van software

De ringen in elk kwadrant geven aan in welk stadium van adoptie wij denken dat die technologie zich bevindt.

- **Adopt** → Wij raden sterk aan deze technologie te gebruiken, waar van toepassing.
- **Trial** → Interessant om alvast ervaring mee op te doen (in een project dat het risico kan dragen).
- **Assess** → Goed om beter te begrijpen en toekomstige impact in te schatten, maar nog niet om toe te passen.
- **Caution** → Ga voorzichtig te werk; brengt risico's of nadelen met zich mee die een zorgvuldige afweging vereisen.

In de volgende secties zullen we onze kijk op de recente ontwikkelingen per kwadrant toelichten.

Commit. Develop. Share.



Commit. Develop. Share.

Technieken	Tools
ADOPT	**ADOPT**
1. Post-Quantum Cryptography	18. NullAway
TRIAL	19. Agents.md
2. Architecture Advice Process	20. IntelliJ Spring Debugger
3. Open Space Technology	**TRIAL**
4. Certificate Lifecycle Management	21. KYAML
ASSESS	**ASSESS**
5. Dynamic Consistency Boundary (DCB)	22. Agent skills
6. Local LLM	23. Danger
7. Rules as Code	24. Devbox
8. Spec-Driven-Development	25. Jujutsu
9. Tokenomics	26. Kepler
CAUTION	27. Kro
-	28. LiteLLM
	29. Matrix
	30. OpenCode.AI
	31. Pulp
	32. Pulumi
	CAUTION
	33. Anubis
	34. Hermes
	35. Mattermost
	36. OpenClaw

Platforms	Talen & frameworks
ADOPT	**ADOPT**
10. Docker Hardened Images	-
11. AWS S3 Vectors	**TRIAL**
12. European Clouds	-
TRIAL	**ASSESS**
13. AWS AgentCore	37. Embabel Agent Framework
14. Temporal	38. Arconia
ASSESS	39. EU Cloud Sovereignty Framework
15. AWS Websearch Connector	**CAUTION**
16. Pigsty minio fork	40. Scala
CAUTION	
17. Codeberg	

Commit. Develop. Share.

Technieken



ADOPT

1. Post-Quantum Cryptography

TRIAL

2. Architecture Advice Process
3. Open Space Technology
4. Certificate Lifecycle Management

ASSESS

5. Dynamic Consistency Boundary (DCB)
6. Local LLM
7. Rules as Code
8. Spec-Driven-Development
9. Tokenomics

Post-Quantum Cryptography

ADOPT

Quantum computers verschillen van klassieke computers in hoe ze - in theorie - exponentiële problemen reduceren tot polynomiale problemen. Een bekend voorbeeld hiervan is het traveling salesman probleem: het berekenen van de optimale route voor een handelaar tussen diverse steden wordt exponentieel complexer met iedere stad die wordt toegevoegd aan de route. Een quantumcomputer is simpel gezegd in staat om zo'n probleem sneller op te lossen, doordat het alle mogelijke routes tegelijkertijd kan berekenen, waarbij de optimale route als oplossing komt bovendrijven. Er zijn redenen om aan te nemen dat dit in de praktijk nog altijd een exponentieel karakter zal hebben, maar het belangrijkste gegeven is dat de quantumcomputer het tijdspad van de oplossing van een dergelijke uitdaging van "niet haalbaar in een mensenleven" kan reduceren tot "de moeite waard om op te wachten".

Een ander voorbeeld van iets dat exponentieel moeilijker wordt, is het breken van sommige asymmetrische public-key algoritmes die o.a. gebruikt worden bij het encrypten van data zoals wachtwoorden. Voorbeelden van dergelijke algoritmes zijn RSA en ECDSA. Iemand met een quantumcomputer kan Shor's algoritme gebruiken om dusdanig geëncrypte data veel sneller te decrypten dan mogelijk met een klassieke computer. Post-Quantum Cryptography draait om het vinden van encryptie-algoritmes die hier niet gevoelig voor zijn. Voorbeelden van quantum-veilige algoritmes zijn ML-KEM en ML-DSA.

Alhoewel er nog geen quantumcomputer bestaat die sterk genoeg is om dit nu te kunnen doen, bestaat het gevaar van *harvest now, decrypt later*: niet-volatiele data, die nu verzameld wordt, kan mogelijk jaren later, wanneer zo'n quantumcomputer wel voorhanden is, nog altijd waardevol zijn. En dus is het belangrijk om nu te beginnen met gegevensclassificatie en data, die over tien jaar nog altijd geheim moet blijven, nu al te beveiligen met encryptie die bestand is tegen Shor's algoritme. Platformen als Cloudflare en AWS en software libraries als OpenSSL en Bouncy Castle bieden hier inmiddels quantum-veilige algoritmes voor en we raden aan er gebruik van te maken.

Commit. Develop. Share.

Architecture Advice Process

TRIAL

Architecture Advice Process is een decentrale aanpak rond architecturele beslissingen. Iedereen kan architecturele keuzes maken. Voordat een keuze gemaakt wordt, moet de beslisser twee groepen consulteren: allereerst iedereen die wezenlijk door het besluit wordt geraakt en ten tweede personen met expertise op het gebied waarop de beslissing wordt genomen. Men wordt hierbij aangemoedigd om weerstand op te zoeken door juist ook advies in te winnen van mensen die een andere mening hebben. Dit proces bevordert een cultuur van samenwerking en kennisdeling binnen een organisatie, waardoor architecturele beslissingen sneller en effectiever kunnen worden genomen.

In Scaling the Practice of Architecture, Conversationally definieert Martin Fowler vier elementen als cruciaal voor het Architecture Advise Process:

1. ADR (Architecture Decision Records), als een denk- en vastleggingsmechanisme. Lichtgewicht documenten waarin besluiten worden vastgelegd en eerdere besluiten kunnen worden geraadpleegd.
2. AFF (Architecture Advisory Forum), als een tijd en plaats voor gesprekken over besluiten die voorbereid worden.
3. Team-sourced architectuur principes, zodat iedereen begrijpt wat een besluit grondig maakt.
4. Build your own radar, om te borgen dat technische trends en inzichten gedeeld worden binnen je organisatie.

Het staat in contrast met Architecture Review Boards, die zich kenmerken door een centrale aanpak waarbij een groep experts (de 'Ivoren Toren') beslissingen neemt over de architectuur van de organisatie.

Thoughtworks merkte in hun Tech Radar van april 2025 op dat de traditionele aanpak met Architecture Review Boards vaak vertragend werkt en zette destijds Architecture Advice Process in op *Trial*. Wij herkennen dit. Ook wij zien dat de aanpak met de 'Ivoren Toren' van een Architecture Review Board nog altijd voorkomt, met implicaties op langere doorlooptijden en minder slagkracht van ontwikkelteams. Daarom zijn wij voor een decentrale aanpak door middel van het Architecture Advice Process en plaatsen we deze techniek in *Trial*.

Open Space Technology

TRIAL

Open Space Technology (OST) is een interactieve en ongedwongen vergadervorm die experts in staat stelt om kennis te delen waarbij de agenda ter plekke wordt opgesteld. Deelnemers zijn vrij om te kiezen welk onderwerp voor hen interessant is om bij te wonen en kunnen gedurende de sessie ook elders aanschuiven. Hierdoor faciliteert het de mogelijkheid om vrij onderwerpen te bespreken waarbij je mensen bij elkaar brengt voor constructieve en positieve discussies.

Hoewel Open Space Technology als format al langer bestaat en ook beschreven staat in "Liberating Structures", een collectie aan technieken om samenwerking en innovatie te bevorderen, zien we dit de laatste tijd meer gebruikt worden bij congressen en binnen organisaties. Onze ervaringen zijn hier positief over en we hebben dit dan ook op *Trial* gezet om anderen eens aan te raden dit te proberen.

Commit. Develop. Share.

Certificate Lifecycle Management

TRIAL

Nog niet zo lang geleden was een (TLS) beveiligingscertificaat twee jaar geldig. Sinds dit jaar is de maximale geldigheidsduur nog maar zes maanden. Deze zal de komende jaren verder worden verkort naar drie maanden en vervolgens naar één maand. Dit betekent dat iets wat voorheen eens per twee jaar moest worden gedaan, nu elke maand gedaan moet worden, oftewel 24 keer vaker. Het proces van het identificeren, aanvragen, distribueren en intrekken van certificaten wordt **Certificate Lifecycle Management** genoemd. Traditioneel werd dit proces handmatig en ad hoc uitgevoerd, idealiter voordat het actieve certificaat verliep. Steeds meer leveranciers bieden tooling om dit proces te automatiseren. Zo'n tool heet vaak simpelweg (vendor/cloud) Certificate Manager.

Jouw organisatie gebruikt wellicht al ACME (Automatic Certificate Management Environment) -protocoltooling om certificaten automatisch te vernieuwen. Deze tooling beheert typisch één enkel systeem. Hierdoor is het geheel van certificaatmanagement verspreid over vele systemen zonder duidelijke controle. Bij JDriven zijn we ervan overtuigd dat het tijd is om nu al te experimenteren met automatisch Certificate Lifecycle Management, inclusief tooling, om voorbereid te zijn op de deadline van 15 maart 2029. Dat is de datum waarop de geldigheidsduur van certificaten teruggebracht wordt naar één maand (+ een halve maand marge). Elk niet-geautomatiseerd proces dat op dat moment nog in gebruik is, zal extreem tijdrovend en foutgevoelig zijn, wat resulteert in hogere kosten en grotere risico's voor de beschikbaarheid.

Dynamic Consistency Boundary (DCB)

ASSESS

Dynamic Consistency Boundary (DCB) is een aanpak voor het handhaven van consistentie in event-sourced systemen. In klassiek DDD bewaakt een **aggregate** de regels die gelden voor data (entities met business rules) binnen een logische grens (je consistency boundary). Zodra je architectuur een regel moet implementeren die logischerwijs meerdere aggregates moet bevragen of beïnvloeden, heb je een **saga** of **process manager** nodig om de coördinatie te regelen. En je bent aangewezen op het gebruiken van corrigerende events wanneer de transactie in één aggregate slaagt maar in de andere faalt. Je kunt overwegen je aggregates anders te modelleren, met het risico dat de modellen gekunsteld overkomen of elders gaan wringen. Bovendien is het hermodelleren van je aggregates nadat je systeem data heeft verwerkt geen eenvoudige exercitie.

Met DCB stap je af van aggregates als statische afbakening en wordt de saga overbodig. In plaats daarvan kijk je per uit te voeren opdracht (command) naar alle events in een event store die tot de huidige staat van het systeem hebben geleid en creëer je een tijdelijke consistentie-"bubbel". Dit is mogelijk doordat je bij DCB events van één of meerdere tags voorziet. Na het interpreteren van de huidige staat en de opdracht schrijf je de nieuwe events weg, waarbij je een voorwaarde (**append condition**) kunt opgeven waarmee je controleert of er niet in de tussentijd conflicterende events zijn weggeschreven. Hiermee bewaakt het systeem zijn consistentie.

Waar je een systeem met aggregates ook zonder event sourcing kunt werken, werk je met DCB met event sourcing zonder aggregates. Er zijn al frameworks en eventstores die het ondersteunen, maar het is een fundamenteel andere benadering van DDD die een leercurve kent. Binnen de DDD-gemeenschap wordt voorzichtig positief gereageerd op het verlaten van het beproefde, doch moeilijke concept van aggregates te verruilen voor DCB. Een overstap zal niet eenvoudig zijn maar de voordelen zijn veelbelovend en de moeite waard om te verkennen.

Commit. Develop. Share.

Local LLM

ASSESS

Lokale LLMs (Large Language Models) zijn neurale taalmodellen die binnen de eigen infrastructuur draaien.

Dit kan bijvoorbeeld op een laptop van een ontwikkelaar of binnen de infrastructuur van een organisatie. Voorbeelden zijn Llama (Meta), Qwen (Alibaba) en Gemma (Google). Met hulpmiddelen zoals Ollama en llama.cpp kunnen deze modellen lokaal worden uitgevoerd.

Het lokaal uitvoeren van een model geeft meer controle over data, privacy en de infrastructuur waarop het model draait. Het kan ook de afhankelijkheid van externe modelleveranciers zoals OpenAI of Anthropic verminderen. Welk model geschikt is, hangt sterk af van de toepassing; kleinere of gespecialiseerde modellen kunnen juist zeer geschikt zijn voor specifieke taken.

Argumenten voor een lokale LLM kunnen via het 'BOKSBAL' acroniem worden opgesomd:

BOKSBAL	**Waarom lokale LLMs?**
Beschikbaarheid	Modellen kunnen binnen de eigen infrastructuur draaien zonder afhankelijk te zijn van een externe dienst.
Onafhankelijkheid	Lokale modellen verminderen de afhankelijkheid van limieten, prijsmodellen, modelwijzigingen en voorwaarden van externe leveranciers.
Kostenbeheersing	Hardware kan een aanzienlijke investering vragen, maar aanschaf en afschrijving maken de kosten beter voorspelbaar.
Soevereine data	Gevoelige gegevens kunnen binnen de organisatie blijven, wat kan helpen bij privacy- en compliance-eisen.
Beheer in eigen hand	De organisatie bepaalt het model, de versie en de runtime, zoals Ollama of llama.cpp.
Aanpasbaarheid	Configuratie, quantisatie en context kunnen worden afgestemd op de toepassing.
LoRA-training	LoRA kan een model efficiënt aanpassen voor specifieke taken, gedrag of schrijfstijlen zonder het volledige model opnieuw te trainen.

We zien dat lokale LLMs zich snel ontwikkelen en verwachten dat ze voor steeds meer toepassingen relevant worden. Tegelijkertijd verschillen de benodigde hardware, modelkeuze en operationele complexiteit sterk per toepassing. Grotere modellen kunnen veel rekenkracht en geheugen vragen, terwijl kleinere modellen voor specifieke taken een praktische optie kunnen zijn.

Lokale modellen zijn ook interessant bij het ontwikkelen van software die gebruikmaakt van agents, Model Context Protocol-servers of probabilistische uitvoer. Een lokaal model kan bijvoorbeeld worden gebruikt tijdens ontwikkeling en testen zonder afhankelijk te zijn van een externe modelleverancier.

Het is momenteel belangrijk om ervaring op te doen en de mogelijkheden en beperkingen beter te begrijpen. Benodigde hardware, modelkeuze en operationele aspecten zijn sterk afhankelijk van de toepassing. Met name voor toepassingen waarbij privacy, data-eigenaarschap of onafhankelijkheid van externe modelleveranciers belangrijk zijn, zien we nu al aanleiding om lokale LLMs verder te onderzoeken.

Commit. Develop. Share.

Rules as Code

ASSESS

Rules as Code (RaC) is een vernieuwende benadering om wetten en beleid te ontwerpen en implementeren. Het maakt het mogelijk wet- en regelgeving op te schrijven in een format dat direct kan worden geïnterpreteerd door machines. Dit brengt grotere nauwkeurigheid en transparantie van wetgeving en maakt het mogelijk wetgeving automatisch uit te rollen, zonder risico op misinterpretatie in de samenwerking tussen betrokken developers en beleidsmakers.

Onder het initiatief Interoperable Europe van de Europese Commissie is er aandacht voor Digital-ready policymaking (DRPM). De EU en haar lidstaten onderzoeken of we wet- en regelgeving kunnen uitgeven in een zgn. 25e taal, die direct geïnterpreteerd kan worden door machines. De Nederlandse Belastingdienst is met haar programma Wendbare Wetsuitvoering een voorloper in dit domein. De Duitse methode van Rulemapping is een ander voorbeeld van hoe het concept van Rules as Code is toegepast.

Hoewel men bij falende IT-projecten vaak al gauw de overheid noemt als voorbeeld, is verschil in de manier waarop beleid en IT worden ontwikkeld een probleem voor organisaties van allerlei slag. We zien IT-afdelingen meer en meer moeite doen om eenheid en structuur aan te brengen in de interpretatie van beleid door in de business analyse op de stoel van de beleidsmaker te gaan zitten, wat tot de genoemde misinterpretatie leidt. Is deze worsteling herkenbaar, overweeg dan Rules as Code om beleid ondubbelzinnig op te stellen en het proces van interpretatie en -validatie tot software implementatie te standaardiseren en transparant te maken.

Spec-Driven-Development

ASSESS

Spec-Driven Development (SDD) is een werkwijze waarbij je eerst een specificatie schrijft en pas daarna de code. In die specificatie leg je vast wat de software moet doen, hoe het ontwerp eruitziet en welke taken er uitgevoerd moeten worden. Met de komst van AI-codingagents is SDD weer actueel. Een duidelijke specificatie geeft de agent context, waardoor hij minder zelf hoeft in te vullen en de code beter aansluit op wat de business wil. Tools als GitHub Spec Kit, Kiro en Tessl ondersteunen deze werkwijze. Ze verschillen in hoever ze gaan: bij de een is de specificatie een startpunt, bij de ander wordt de code er volledig uit gegenereerd.

De tools zijn nog jong en veranderen snel. Het is nog onduidelijk of specificatie en code op de lange termijn met elkaar in lijn blijven. Ook is het proces voor kleine wijzigingen al snel te zwaar. We denken dat SDD AI-ondersteund ontwikkelen voorspelbaarder kan maken, maar willen eerst meer ervaring opdoen. Daarom zetten we het op Assess.

Commit. Develop. Share.

Tokenomics

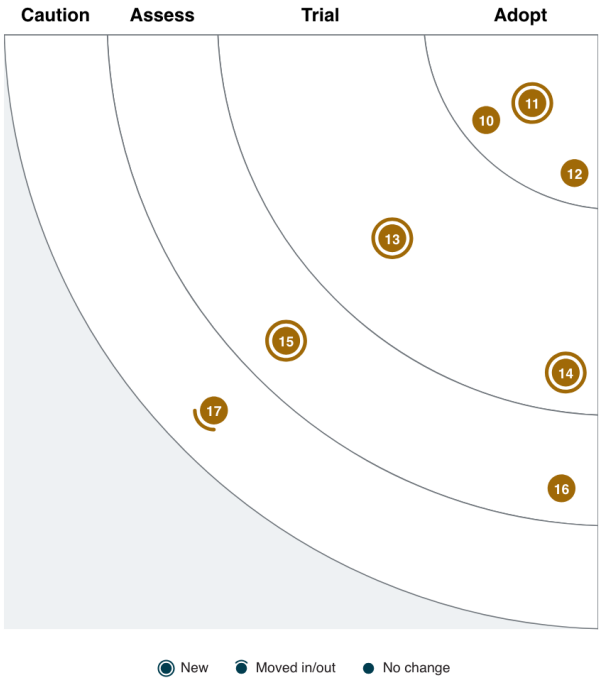
ASSESS

Tokenomics is het bewust beheren van het tokenverbruik van AI-modellen. Tokens zijn de eenheden waarin AI-modellen tekst verwerken, waarbij het aantal de kosten bepaalt. Het onderwerp wordt urgenter door AI-agents, die zelfstandig taken in meerdere stappen uitvoeren en daarbij veel tokens kunnen verbruiken. Technieken, zoals een passend model per taak kiezen en alleen relevante context meesturen, helpen het verbruik te beheersen.

Het doel is niet minimaal verbruik maar een bewuste afweging tussen kosten en kwaliteit. Er verschijnen steeds meer tools die besparing beloven, maar hun effect verschilt sterk per toepassing. Wij raden daarom aan om zulke tools eerst in de eigen context te testen en te meten. Omdat prijzen, modellen en limieten snel veranderen, bestaan er nog geen vaste werkwijzen. Om deze redenen plaatsen wij Tokenomics in *Assess*.

Commit. Develop. Share.

Platforms



ADOPT

10. Docker Hardened Images
11. AWS S3 Vectors
12. European Clouds

TRIAL

13. AWS AgentCore
14. Temporal

ASSESS

15. AWS Websearch Connector
16. Pigsty minio fork

CAUTION

17. Codeberg

Docker Hardened Images

ADOPT

Containerisatie, ofwel OS-virtualisatie, is voor veel developers de standaard manier om software te bouwen, testen en deployen. Het is daarom belangrijk dat de containers waar hun software op draait veilig zijn. Daartoe worden container images gehard, wat wil zeggen dat er extra veiligheidsmaatregelen worden toegevoegd om het aantal kwetsbaarheden te minimaliseren. Voorheen was dit een taak die developers zelf moesten uitvoeren, maar Docker biedt sinds december 2025 gratis 'geharde' versies van bepaalde images aan, zogenoemde Docker Hardened Images (DHI). Voor JVM-ontwikkelaars is er bijvoorbeeld een hardened versie van de populaire Eclipse Temurin container image (Eclipse Temurin DHI).

We zijn enthousiast over Docker Hardened Images, omdat het een eenvoudige manier biedt om de veiligheid van container images te verbeteren zonder dat developers zelf aan de slag hoeven met hardening. DHI's zijn drop-in replacements voor reguliere images, wat betekent dat je ze gemakkelijk kunt gebruiken in plaats van de standaardversies. Bovendien worden CVE's proactief gepatcht in DHI's, waardoor je minder zorgen hebt over kwetsbaarheden in je container images. Daarom plaatsen we DHI in *Adopt* en stellen we voor bij beschikbaarheid altijd de geharde versie te verkiezen.

Commit. Develop. Share.

AWS S3 Vectors

ADOPT

RAG-applicaties hebben een plek nodig om embeddings op te slaan. Het gebruikelijke antwoord is een aparte vector database: nog een systeem om op te zetten, te beheren en te betalen. Met S3 Vectors voegt AWS vector storage en search toe aan S3 zelf: je maakt een bucket, slaat je embeddings op en bevraagt ze, zonder iets te provisionen. Het schaalt zoals S3 dat doet, met volgens AWS tot twee miljard vectoren in één index. Dankzij het vertrouwde pay-per-use-prijsmodel van S3 is het voor de meeste workloads aanzienlijk goedkoper dan een aparte database. De afweging is dat queries wat trager zijn, al blijven ze ruim onder een seconde, en dat de feature set basaler is. Dit past bij eenvoudige vector storage, zonder op pure performance te concurreren met gespecialiseerde databases. Voor organisaties die al op AWS draaien is het een eenvoudige toevoeging die naadloos aansluit op hun bestaande storage en tooling.

European Clouds

ADOPT

De EU stelt via de General Data Protection Regulation (GDPR) eisen aan bedrijven die data van EU-inwoners verzamelen en verwerken. De Amerikaanse Cloud Act, wet FISA702 en Executive Order 12.333 stellen de Amerikaanse overheid echter in staat een bedrijf met Amerikaans hoofdkantoor te dwingen om de GDPR te negeren. Ze kunnen inzage in data eisen, zelfs als de servers buiten de VS liggen. Deze bedrijven hoeven hun klanten hier niet over te informeren. Ook kan de dienstverlening zelf aan banden gelegd worden, een reëel risico, zoals het Internationaal Strafhof ondervond toen de Amerikaanse politiek Microsoft opdroeg hun e-mail te blokkeren. Steeds meer bedrijven zoeken daarom Europese alternatieven voor hun data en diensten. Veel daarvan staan op european-alternatives.eu. Initiatieven om dergelijke beweging in gang te zetten, worden gevat binnen het concept digitale soevereiniteit.

Vanuit developer- en operationsperspectief kijkt JDriven met name naar cloudproviders. Bij het kiezen van een Europese cloud is het van belang te letten op een aantal aspecten:

- Wat biedt het aan compute-, networking- en storage-services? Veel van deze providers bieden niet het volledige serviceaanbod van de grootste Amerikaanse hyperscalers. Maar dat is minder relevant dan de vraag of het serviceaanbod voldoende is voor de actuele behoefte.
- Wat zijn de Service Level Agreements (SLA) voor beschikbaarheid?
- Wat zijn de prijzen? Veel providers bieden een prijscalculator om snel inzicht te kunnen krijgen en een vergelijking te maken. Afhankelijk van welke services gebruikt worden, kunnen verschillen per maand hier behoorlijk zijn.
- Is er een free tier? Een free tier kan enorm helpen met het opdoen van ervaring en dientengevolge ook met het vinden van ervaren developers.
- Biedt het infrastructure as code en een SDK om de services reproduceerbaar op te tuigen? Bestaan er bijvoorbeeld Terraform-providers voor?
- Hoe toegankelijk zijn de leermiddelen voor developers?
- Biedt het een vorm van managed Kubernetes? Dit is het containerplatform waar meer en meer klanten hun software op uitrollen, wat van groot belang is bij een exitstrategie waar een migratie naar een andere provider bij komt kijken.
- Ondersteunt het Gaia-X? Gaia-X is een project voor federatie van digitale ecosystemen waarbij data-soevereiniteit, veiligheid en naleving van wetgeving centraal staan. Een dergelijk streven naar samenwerking en integratie is een aspect van Europese cloudproviders dat we bij de gevestigde Amerikaanse providers niet zien.

Nemen we bovenstaande in overweging, dan zien we vooral grote stappen bij Scaleway, STACKIT en OVHcloud. Ook Exoscale, IONOS en het Nederlandse Cyso Cloud zijn interessant.

Commit. Develop. Share.

AWS AgentCore

TRIAL

Een AI-agent naar productie brengen levert problemen op die gewone applicaties niet kennen. Agents draaien minuten of uren in plaats van milliseconden, ze genereren onderweg artifacts zoals Word-documenten of spreadsheets, en ze voeren code uit en roepen tools aan namens een gebruiker. Dat maakt security een serieus aandachtspunt. En het schalen van langlopende taken die moeten bijhouden waar ze gebleven zijn, is iets waar een doorsnee web stack niet goed mee overweg kan.

AWS AgentCore Runtime lost dit op door voor elk gesprek een kleine, eigen virtual machine te starten. Elk gesprek krijgt een geïsoleerde omgeving met eigen rekenkracht, memory en file system, zodat een agent gegenereerde bestanden gewoon naar de lokale disk kan schrijven zonder dat die in het gesprek van een andere gebruiker terecht kunnen komen. Zodra het gesprek eindigt, wordt de omgeving weggegooid. Omdat AWS de provisioning verzorgt, vraagt schalen geen extra werk. Een gesprek kan tot acht uur duren.

De runtime is framework-agnostisch en neemt je gepackagede applicatie as-is, zodat bestaande agents zonder herschrijven kunnen worden gedeployd. Vergelijkbare producten bestaan bij andere vendors, maar wie al op AWS-infrastructuur draait, zal dit het meest natuurlijk vinden aansluiten.

Temporal

TRIAL

Temporal is een platform dat robuuste processen implementeren makkelijk maakt door automatisch bij te houden waar een proces (workflow) precies is gebleven.

Dit concept is inmiddels in de markt bekend als *Durable Execution*. Je schrijft code die een software-proces van arbitraire looptijd implementeert; dit mag in principe oneindig lang duren. Je code *is* de workflow bij Temporal. Het platform houdt onder water precies bij welke stappen al zijn afgerond en wat het resultaat van die stappen was. Dit zorgt ervoor dat het proces indien nodig kan worden hervat, waarbij alle reeds succesvol uitgevoerde stappen automatisch worden overgeslagen.

Er zijn meerdere platformen en libraries die expliciet inzetten op ondersteuning van het *Durable Execution* concept, waarvan Temporal de meest volwassen is. Temporal heeft SDK's beschikbaar voor praktisch alle gangbare programmeertalen, waarvan JDriven de Java-support heeft geëvalueerd en goed bevonden. Met de opkomst van generatieve AI is het schrijven van *stateful* orkestratielogica in en over agents heen de komende tijd steeds vaker nodig. Omdat Temporal een volwassen platform is en zowel een *managed-cloud* als een *opensource*, *zelfgehoste* variant biedt, hebben we ervoor gekozen het in de *Trial* ring te plaatsen.

AWS Websearch Connector

ASSESS

Een AI-agent toegang geven tot web search maakt hem aanzienlijk nuttiger, maar het zet ook een deur open: queries verlaten je omgeving, komen bij search API's van derden terecht en brengen uitgaand verkeer met zich mee. Dit leidt tot compliancevragen. Web Search op AgentCore vermijdt dit door queries uit te voeren op Amazons eigen web index. De agent doorzoekt het web, maar het request verlaat de AWS-infrastructuur nooit en raakt het publieke internet niet.

De resultaten kunnen iets achterlopen op gespecialiseerde publieke search-producten, maar voor organisaties waar security en compliance vooropstaan, is dat een afweging die de moeite waard is. Daar staat tegenover dat je je verbindt aan de web index van Amazon, wat vanuit soevereiniteitsperspectief mogelijk bezwaarlijk kan zijn.

Commit. Develop. Share.

Pigsty minio fork

ASSESS

MinIO was een populair opensource product om on-premise S3 buckets te hosten zonder AWS. Maar sinds enkele jaren heeft MinIO Inc de licentievoorwaarden strenger gemaakt en sinds 2026-02-13 is de MinIO repository gearchiveerd. MinIO Inc doet zelf nog wel onderhoud, maar maakt die code niet meer open source.

Organisaties die MinIO gebruiken maar niet afhankelijk willen zijn van een betaalde versie van MinIO Inc kunnen nu overwegen om uit te wijken naar de Pigsty MinIO fork. Die zou als een drop-in replacement gebruikt moeten kunnen worden. Images zijn via Docker Hub beschikbaar; binaries via APT en Yum.

Een ander alternatief is Ceph. Dat is open source en zonder licentiekosten te gebruiken. Voor de applicaties die gebruikmaken van MinIO is dit ook een drop-in replacement. Maar voor de beheerders niet, aangezien Ceph een complexer product is. Omdat Ceph onder de vlag van de Linux Foundation valt, is de kans dat het “opeens” closedsource wordt (zoals bij MinIO of Redis gebeurde) nagenoeg nul. De governance-structuur is erop ingericht dat de community en de industrie samen eigenaar zijn. Nog een opensource alternatief is SeaweedFS. Minder complex dan Ceph, maar niet zo eenvoudig als MinIO.

Codeberg

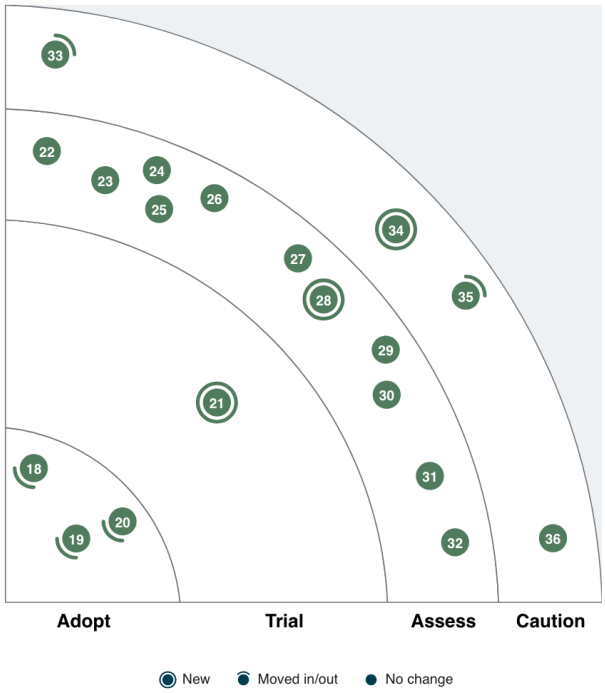
CAUTION

Als je graag een Europees alternatief voor GitLab of GitHub als clouddienst zoekt, is Codeberg een interessant alternatief om te overwegen. Het biedt ongeveer dezelfde functionaliteiten, zij het minder uitgebreid, zoals: code storage, static site hosting, pull requests met daarbij reviewmogelijkheden, pipelines, etc. Meertaligheid wordt ondersteund via Weblate. Het is gebaseerd op het Forgejo (Gitea fork) codeplatform.

Codeberg is een in Duitsland gevestigde non-profitorganisatie. Iedereen kan gratis een account aanmaken, code uploaden en samenwerken. Als je de continuïteit van Codeberg wilt versterken is het ook mogelijk om lid te worden (je betaalt daarvoor een kleine contributie) en/of eenmalige of structurele donaties te doen. Alhoewel het hoge beschikbaarheid nastreeft kan het niet dezelfde formele uptime Service Level Agreements (SLA’s) garanderen als commerciële platformen. Recentelijk heeft Codeberg zijn gebruiksvoorwaarden aangepast. Het staat niet langer toe dat code, die (deels) gegenereerd is met AI, wordt gehost in een Codeberg repo, wegens het risico op copyrightschendingen. Het is een begrijpelijke doch gewaagde zet en kan betekenen dat Codeberg niet de plek is om je code te hosten, indien je nu of in de toekomst AI-assisted coding wilt toepassen.

Commit. Develop. Share.

Tools



ADOPT

18. NullAway
19. Agents.md
20. IntelliJ Spring Debugger

TRIAL

21. KYAML

ASSESS

22. Agent skills
23. Danger
24. Devbox
25. Jujutsu
26. Kepler
27. Kro
28. LiteLLM
29. Matrix
30. OpenCode.AI
31. Pulp
32. Pulumi

CAUTION

33. Anubis
34. Hermes
35. Mattermost
36. OpenClaw

NullAway

ADOPT

NullAway is een statische analysetool voor Java die op basis van nullability-annotaties een mogelijke `NullPointerException` tijdens de build detecteert.

Het draait als een plugin op ErrorProne, wat een compiler plugin is voor Java die tijdens de build statische codechecks uitvoert. NullAway geeft daardoor snelle feedback in de normale build. Recente versies ondersteunen ook JSpecify en vereisen JDK 17 of hoger.

Wij verplaatsen NullAway van *Trial* naar *Adopt*. Het is een goede tool om je null-safety discipline mee op orde te brengen, zeker met de opkomst van null-restricted value class types (met ! and ? typemarkeringen) als onderdeel van Project Valhalla, zelfs als de uiteindelijke officiële Java notatie zal verschillen van NullAway's annotaties.

Commit. Develop. Share.

Agents.md

ADOPT

Agents.md is een open standaard voor het meegeven van projectspecifieke instructies aan AI-codingagents. Het is een Markdown-bestand dat je in de root van je repository plaatst, vergelijkbaar met een README.md maar dan gericht op AI-assistenten in plaats van menselijke developers. In het bestand leg je vast hoe het project gebouwd en getest moet worden, welke codeconventies er gelden en welke architectuurpatronen gevolgd moeten worden.

Het initiatief is gestart door Sourcegraph en valt nu onder de Agentic AI Foundation binnen de Linux Foundation. AGENTS.md is tool-agnostisch en wordt ondersteund door vrijwel alle grote AI-codingtools, zoals Claude Code, GitHub Copilot, OpenAI Codex, Cursor en Google Antigravity. Voor Claude Code geldt dit pas sinds versie 2.1.277; eerder was hier een apart CLAUDE.md-bestand voor nodig.

Vanuit JDriven zien wij dat een AGENTS.md de kwaliteit van AI-gegenereerde code merkbaar verbetert, doordat de AI beter begrijpt hoe het project in elkaar zit en welke patronen het team hanteert. Een bijkomend voordeel is dat dit ook voor nieuwe teamleden waardevolle documentatie oplevert. Het opstellen van een goed AGENTS.md-bestand vergt wel een initiële investering en het vinden van de juiste balans: je moet bewust nadenken over je conventies en deze helder formuleren.

In korte tijd is AGENTS.md uitgegroeid tot de standaard in de markt. Eén bestand volstaat nu voor het hele team, ongeacht welke AI-assistent iemand gebruikt. Daarom verplaatsen wij AGENTS.md naar *Adopt* en raden wij het aan voor elk project waarin AI-codingtools worden gebruikt.

IntelliJ Spring Debugger

ADOPT

De IntelliJ Spring Debugger tilt Spring debugging naar een heel ander niveau dan standaard debugging met breakpoints. Sinds de vorige radar is deze plugin van Trial naar Adopt gegaan: de meerwaarde is duidelijk en we zien weinig nadelen.

Tijdens een debugsessie geeft de Spring Debugger je toegang tot de volledige application context. Alle actieve beans zijn inzichtelijk, ook buiten de scope van je huidige breakpoint. Je kunt methods aanroepen op beans die elders in de applicatie leven. Zo kun je onder andere caches invalideren, direct queries uitvoeren via je repositories en tabelinhoud inzien of on-the-fly muteren, zonder de sessie te onderbreken. Het maakt de Spring *magie* inzichtelijker, wat met name de minder ervaren Spring developer zal helpen.

Daarnaast zijn omgevingsvariabelen zichtbaar precies zoals de applicatie ze heeft ingelezen. Je krijgt meer inzicht in hoe beans geïnstantieerd worden en waar hun waarden vandaan komen. Dat maakt de internals van Spring een stuk transparanter en concreter, zeker voor ontwikkelaars die minder vertrouwd zijn met het framework.

De hoeveelheid informatie die je tijdens een debugsessie tot je beschikking hebt, is aanzienlijk groter dan wat standaard IntelliJ biedt. Het leidt niet tot merkbare vertraging of andere negatieve neveneffecten.

Werk je dagelijks met Spring Boot in IntelliJ IDEA Ultimate? Dan mag deze plugin eigenlijk niet ontbreken. De plugin is alleen beschikbaar voor Ultimate en is primair gericht op Spring Boot; voor plain Spring zonder Boot kunnen de ervaringen wisselvallig zijn.

Commit. Develop. Share.

KYAML

TRIAL

KYAML is een veiligere en minder ambigue subset van YAML. KYAML is specifiek ontworpen voor Kubernetes en pakt veelvoorkomende valkuilen van YAML aan, zoals gevoeligheid voor witruimte en impliciete typeconversie, terwijl het volledig compatibel blijft met bestaande YAML-parsers en tooling. Het lost bijvoorbeeld het 'Norway problem' op, waar een NO zonder quotes zowel als de boolean `false` als de countrycode voor Noorwegen kan worden geïnterpreteerd. KYAML biedt specifieke voordelen bij het gebruik van templating, bijvoorbeeld in Helm Charts.

JDriven vindt dat de sterke stimulans vanuit het Kubernetes-ecosysteem rechtvaardigt om KYAML direct in de *Trial* ring te plaatsen. Merk daarbij op dat KYAML een subset van YAML is en daarom overal kan worden gebruikt waar YAML kan worden gebruikt, dus ook buiten Kubernetes.

Agent skills

ASSESS

Agent Skills is een open standaard voor het verpakken van herbruikbare AI-agentcapabilities als modulaire eenheden, oorspronkelijk ontwikkeld door Anthropic en in december 2025 vrijgegeven als open specificatie. Het stelt developers in staat om domeinkennis, instructies, scripts en referentiemateriaal te bundelen in een mapstructuur met een `SKILL.md`-bestand, dat agents on-demand kunnen ontdekken en laden. De standaard wordt inmiddels breed ondersteund door tools als Claude Code, VS Code, GitHub Copilot, OpenAI Codex, Cursor en Gemini CLI, en is ook geïntegreerd in het Spring AI-ecosysteem via de spring-ai-agent-utils toolkit.

Vanuit JDriven zien wij Agent Skills als een interessant stuk gereedschap voor teams die werken met AI-agents in hun ontwikkelproces. De kracht zit in de eenvoud: een skill is niet meer dan een map met Markdown en optioneel scripts, waardoor het laagdrempelig is om in gebruik te nemen en in versiebeheer te plaatsen bij je applicatiecode. Voor Java-teams is de integratie via Spring AI bijzonder interessant: skills werken modelonafhankelijk en vereisen geen architectuurwijzigingen in bestaande applicaties.

De standaard is echter jong en evolueert snel. Er is nog geen ingebouwd versiebeheermechanisme voor skills, en scripts worden zonder sandboxing uitgevoerd op de lokale machine, wat security-aandacht vereist. Daarnaast ontbreekt een standaard human-in-the-loop-mechanisme, waardoor agents skills en scripts automatisch aanroepen kunnen doen zonder expliciete goedkeuring. Dat grote spelers dit in gebruik nemen is bemoedigend, maar best practices voor het organiseren en onderhouden van skills op schaal moeten zich nog bewijzen. Om deze redenen plaatsen wij Agent Skills op Assess.

Commit. Develop. Share.

Danger

ASSESS

Danger is een opensource automatiseringstool die draait tijdens CI/CD om code review-standaarden af te dwingen. De tool automatiseert repetitieve code review-taken door pull requests te controleren tegen vooraf gedefinieerde regels. Danger draait als een extra stap in je pipeline en integreert met platforms zoals GitHub, GitLab en Bitbucket. De tool ondersteunt meerdere talen waaronder Ruby, JavaScript, Swift, Python en sinds 2022 ook Kotlin.

Teams kunnen Danger configureren om changelog-entries te verifiëren, issue tracker-referenties te vereisen in PR-omschrijvingen of architecturale anti-patterns te signaleren. Hiermee detecteert Danger veel voorkomende omissies, op basis van door het team in code gedefinieerde standaarden, automatisch.

Hoewel Danger al bestaat sinds 2016, maakt de in 2022 toegevoegde ondersteuning voor Kotlin het opnieuw relevant voor het Java- en Kotlin-ecosysteem. JDriven ziet dat organisaties Danger succesvol inzetten om hun code review-proces te standaardiseren. De tool helpt teams om reviewnormen vast te leggen en consistentie te behouden over pull requests heen. We plaatsen Danger in *Trial*: de tool is volwassen, maar Kotlin-support is relatief recent en teams moeten valideren hoe het past in hun specifieke workflow en reviewproces.

Devbox

ASSESS

Devbox is een command-line tool om reproduceerbare ontwikkelomgevingen met projectspecifieke tools en versies te definiëren.

Met Devbox leg je in versiebeheer vast welke tools een project nodig heeft, zoals Node, Python, Java of Pandoc, waarna dezelfde omgeving lokaal en in CI (simpelere build pipelines) gebruikt kan worden. Het voorkomt ook de "het werkt op mijn machine"-issues. Daarmee is het een lichter alternatief voor volledig gecontaineriseerde ontwikkelomgevingen. Devbox draait op Nix en moet niet verward worden met Microsoft Dev Box.

Wij plaatsen Devbox in *Assess*. Het biedt duidelijke voordelen voor reproduceerbare ontwikkelomgevingen, maar de Nix-basis en de bijbehorende leercurve maken het verstandig om eerst ervaring op te doen voordat het breed wordt ingezet.

Jujutsu

ASSESS

Jujutsu is een versiebeheersysteem (VCS) dat een eigen user interface biedt bovenop bestaande storage systems van andere versiebeheersystemen. Het is gestart als project binnen Google en een aantal mensen binnen Google werken er ook aan, maar Google is niet de maintainer. Jujutsu wil een makkelijkere manier bieden voor ontwikkelaars om hun dagelijks bezigheden met VCS te doen. Het is geheel compatibel met Git, wat momenteel het enige geïmplementeerde achterliggende storage system is. Qua features heeft Jujutsu het een en ander afgekeken van Git, Mercurial, Sapling en Darcs zoals snelheid, anonieme branches, geavanceerde merge conflict resoluties. Daarnaast ziet het iedere change automatisch als een commit, houdt het een uitgebreide log bij van acties op de repository en maakt een undo eenvoudig.

Jujutsu wordt actief doorontwikkeld, maar zit nog niet op een 1.0 versie en noemt zichzelf nog experimenteel. Het is zeker iets om in de gaten te houden en daarom staat het in *Assess* op de radar, want het combineren van de krachten van diverse versiebeheersystemen in een eenvoudigere user interface zonder van Git af te hoeven stappen klinkt veelbelovend.

© 2026 JDriven

Kepler

ASSESS

Kepler is een Prometheus exporter die energieverbruik in Kubernetes van containers, pods en nodes kan meten.

Door daadwerkelijk resourcegebruik en hardwaresensordata te meten, geeft Kepler concretere signalen voor prestaties, capaciteit en energieverbruik dan statische analyses of code scanners.

Wij plaatsen Kepler in *Assess*. Het is een interessante aanvulling voor teams die sustainability en efficiency meetbaar willen maken in Kubernetes, maar als CNCF sandbox project is het nog niet klaar voor brede toepassing.

Kro

ASSESS

Kube Resource Orchestrator (KRO) is een opensource, Kubernetes-native project bedoeld om complexe definities van resources te beheren middels Resource Graph Definitions (**RGD**'s). Deze RGD's zijn templates waarmee daadwerkelijke resource-groepen eenvoudig en uniform beheerd en uitgerold kunnen worden.

kro vereenvoudigt beheer van k8s resources door simpeler en centraal definities te aggregeren; een brug tussen complexiteit en ontwikkelaarvriendelijke self-service, zonder verdere banden met een specifieke leverancier. Waar **helm** distributie van definities makkelijker en beheersbaar maakt, zorgt **kro** voor een lokale kant van beheer van complexe resources.

LiteLLM

ASSESS

LiteLLM geeft je een uniforme, OpenAI-compatible interface richting meer dan honderd verschillende LLM-providers, waaronder Anthropic, OpenAI, Gemini en Amazon Bedrock. De tool werkt ook in combinatie met lokale modellen.

LiteLLM kan gebruikt worden als Python SDK of als zelfhosted proxy server (ook wel AI-gateway genoemd). Met de SDK kun je eenvoudig van model of provider wisselen in je applicatie zonder code te herschrijven. De proxy voegt extra mogelijkheden toe, zoals het centraal regelen van toegang tot providers, beschikbaarheid opties (fallbacks, retries etc.) en het verzamelen van logging en metrics.

De proxy is ook te gebruiken in combinatie met bestaande AI-agent harnesses zoals OpenCode en Claude Code. Het maakt in die opzet het gebruik van LLM's binnen een organisatie beter beheersbaar. Je deelt virtuele API-keys uit per team of project, met eigen budgetten en limieten. Hiermee voorkom je dat API-keys van providers verspreid raken over talloze applicaties en ontwikkelaars.

Features zoals single sign-on en geavanceerde audit logging zitten achter een enterpriselicentie.

Werk je met meerdere LLM-providers, of wil je het LLM-gebruik binnen je organisatie centraal beheren en inzichtelijk maken? Dan is het de moeite waard om een test met LiteLLM te doen. We plaatsen LiteLLM daarom in de *Assess* ring. De ontwikkeling van het project gaat erg snel (met regelmatig breaking changes) en ook worden er soms beta features geïntroduceerd die uiteindelijk alleen toegankelijk blijven via een enterpriselicentie. LiteLLM is eerder dit jaar slachtoffer geworden van een supply chain attack (door een hack bij Trivy) waarbij tijdelijk versies van de software met malware zijn gepubliceerd op PyPI. Er zijn maatregelen genomen om dit te voorkomen, zoals gebruik van Trusted Publishing op PyPI en het ondertekenen van container images met cosign.

Commit. Develop. Share.

Matrix

ASSESS

Matrix is een open protocol voor veilige, gedecentraliseerde instant messaging.

Matrix ondersteunt federation, end-to-end encryptie en bridges naar andere chatplatforms. Het wordt al gebruikt in Frankrijk bij de overheid met Tchap en in Duitsland bij de Bundeswehr en zorginstellingen. Hierdoor is het een interessant alternatief voor organisaties die meer controle willen over hun communicatie-infrastructuur.

Wij plaatsen Matrix in *Assess*. Het protocol is sterk voor open en soevereine communicatie, maar operationele complexiteit en wisselende kwaliteit van client software maken het nog geen vanzelfsprekende keuze voor brede inzet.

OpenCode.AI

ASSESS

OpenCode.ai is een opensource AI-coding assistant die onafhankelijk werkt van providers en modellen. De tool kan worden gebruikt als command-line interface, desktop-applicatie en IDE-extensie. OpenCode.ai draait lokaal en stelt je in staat om meerdere agents te definiëren, bijvoorbeeld voor Planning, Building of Reviewing. De tool begrijpt je code met behulp van language server protocols (LSP). Je kunt skills of MCP-based tools integreren met een expliciet permission model. Developers kunnen modellen gebruiken van 75+ LLM providers, bijvoorbeeld OpenAI, Anthropic, Google of zelfs lokale modellen.

OpenCode.ai bevordert consistent Agentic AI gebruik binnen teams door de configuratie te delen in je team. Code en context blijven lokaal. Configuratie en governance blijven onder eigen controle via version control en lokale environment variabelen.

JDriven ziet dat OpenCode.ai volwassen genoeg is voor pilots, maar de sprong naar brede toepassing vraagt eerst validatie en feedback op je eigen workflows. De tool moet zich nog bewijzen in diverse team- en projectcontexten. We plaatsen OpenCode.ai daarom in *Trial*: geschikt voor experimentatie en workflow standaardisatie nu, maar nog te nieuw om organisatiebreed in te zetten als standaard development tooling.

Pulp

ASSESS

Pulp is een gratis self-hosted opensource container- en artifact-repository. Het is uit te rollen als enkele container, of in meerdere containers via podman of docker compose. Bovendien is er Pulp Operator, waarmee Pulp horizontaal schaalbaar is uit te rollen in een kubernetes cluster. Net als Nexus en Artifactory is Pulp te gebruiken als een proxy voor publieke container en artifact repositories. Het is niet zo uitgebreid als die producten; zo is de *pull through cache* voor NPM nog in tech review en heeft het weliswaar een vulnerability reporter voor Python packages, maar niet voor andere repositories. Pulp biedt wel de mogelijkheid om zelf plugins toe te voegen voor additionele vulnerability reporting.

We zien dat organisaties worstelen met oplopende licentiekosten voor Nexus en Artifactory. Ook laat de beheersbaarheid van deze producten soms te wensen over. JDriven ziet dat Pulp een interessant alternatief kan zijn, maar waarschuwt dat het qua features mogelijk achterblijft en dat alhoewel het gratis is om te gebruiken, het qua onderhoudbaarheid wel kosten met zich meebrengt.

© 2026 JDriven

Pulumi

ASSESS

Pulumi is een opensource infrastructure-as-code-tool (IaC-tool) waarmee je cloudinfrastructuur definieert in algemene programmeertalen zoals TypeScript, Python, Go, .NET, Java en desgewenst YAML.

De opensource IaC-tool staat los van het commerciële Pulumi Cloud-platform, al kan dat platform wel worden gebruikt voor teamsamenwerking en als backend voor het opslaan van de infrastructuurstatus. Wie Pulumi zonder Pulumi Cloud wil gebruiken, kan state ook opslaan in een eigen backend, zoals S3, Azure Blob Storage, Google Cloud Storage, PostgreSQL of het lokale filesystem.

Voor development teams is Pulumi interessant omdat infrastructuurcode met dezelfde taal, libraries en ontwikkeltools kan worden geschreven als applicatiecode. Dat verlaagt de drempel voor developers om infrastructurele wijzigingen mee te ontwikkelen en maakt hergebruik, testing en abstrahering mogelijk met bekende programmeerconstructies.

Algemene programmeertalen maken het eenvoudig om nette abstraheringen te bouwen, maar ook om onnodig complexe deployment logica te introduceren en teams moeten bewuste keuzes maken rond statebeheer, secrets, governance en samenwerking. Voor organisaties die al zwaar op Terraform of OpenTofu leunen betekent Pulumi bovendien een extra IaC-model en ecosysteem om te beheren.

Daarom plaatsen we Pulumi in *Assess*: het is wellicht volwassen genoeg om op echte projecten te beproeven en kan developers productiever maken, maar we adviseren eerst te verkennen wat het vereist aan team afspraken, governance en statebeheer alvorens het in te zetten.

Anubis

CAUTION

Anubis is een softwareprogramma dat een op JavaScript gebaseerde proof-of-work challenge aan websites toevoegt voordat gebruikers deze kunnen bezoeken. In principe kunnen alleen moderne browsers die proof-of-work challenge uitvoeren en fungeert dit dus als een alternatief voor CAPTCHA's, waar geen gebruikersinteractie voor nodig is. Het is ontworpen om het internet te beschermen tegen de eindeloze stroom requests van AI-bedrijven. Een scraperbot kan standaard geen proof-of-work challenge uitvoeren en dus geen zinvolle verbinding opzetten. Anubis fungeert zodoende als een zogenaamde 'tar pit': het remt met name scrapers af, die data zoeken voor AI-training.

Van belang is om Anubis te configureren zodat bots van bijvoorbeeld zoekmachines doorgelaten worden, indien gewenst. De standaardafbeelding van Anubis, die bezoekers te zien krijgen bij een bezoek, zal niet bij het thema van alle websites passen, maar die kan aangepast worden. Anubis is een jong project en toen we er op stuitten, was het afwachten of deze aanpak stand zou houden gezien de snelle ontwikkelingen op het AI-vakgebied. Het aanpassen van een scraper, zodanig dat hij de proof-of-work challenge wel kan verrichten, zou hem langzamer maken, waardoor hij minder requests zou kunnen versturen, was de redenatie.

Inmiddels lijken AI-bots in staat eenvoudig workarounds, zoals anubis-fetch, te kunnen verzinnen om het te omzeilen. Websites als Codeberg rapporteren dat AI-crawlers Anubis steeds vaker weten te omzeilen en kijken naar additionele maatregelen. Het toepassen van Anubis is een relatief kleine investering, maar op zichzelf waarschijnlijk niet voldoende, terwijl het wel een merkbare vertraging kan opleveren voor menselijke bezoekers.

Commit. Develop. Share.

Hermes

CAUTION

Hermes Agent van Nous Research is een open-source AI-agent die zichzelf verbetert naarmate je hem langer gebruikt. Het is een alternatief voor OpenClaw, dat we in de vorige radar al op 'Caution' zetten.

Hermes gebruikt een 'learning loop' waarbij het na het uitvoeren van een taak zelf herbruikbare skills vastlegt en deze over een langere periode verfijnt. Daarnaast houdt de agent een eigen geheugen bij en kan hij makkelijk informatie terugzoeken uit eerdere sessies. De AI-agent is te gebruiken vanuit de terminal (text user interface) en vanuit chatplatforms als Telegram, Discord, Slack, WhatsApp en Signal. Voor parallel werk kan de agent geïsoleerde subagents starten. Met een ingebouwde scheduler voert hij terugkerende taken uit. Hermes kan gebruikt worden met een LLM-provider naar keuze.

Hermes heeft echter dezelfde problemen als OpenClaw. De agent voert zelfstandig shell-commando's uit, heeft toegang tot credentials en is via chatplatforms van buitenaf bereikbaar. Ook hier vormen tekstinstructies aan het taalmodel een risico vanwege de kans op promptinjecties en invoer van onbetrouwbare data.

Voor specifieke, afgebakende use cases kun je Hermes inzetten in een gecontroleerde omgeving. Denk aan een dedicated device voor de agent, met beperkte (netwerk)toegang tot zelfgehoste services. Geef de agent ook geen eigen API keys van providers, maar laat een gateway zoals LiteLLM de keys toevoegen aan LLM requests. Twijfel je over hoe je dit veilig opzet, begin er dan voorlopig niet aan. Vanwege deze risico's zetten we Hermes Agent, net als OpenClaw, op 'Caution' voor professioneel gebruik.

Mattermost

CAUTION

Mattermost is een opensource online chatdienst die zelf te hosten is. De dienst is ontworpen als interne chat voor organisaties en bedrijven, en positioneert zich voornamelijk als alternatief voor Slack en Microsoft Teams.

De laatste versies ondersteunen bellen, videobellen en schermdelen. Er is een desktopapplicatie voor Windows, macOS en Linux. Ook op je Android- of iOS-telefoon is er een app beschikbaar. We zien dat grote partijen Mattermost al enige tijd naar tevredenheid inzetten om developers, beheerders en gebruikers met elkaar te laten communiceren.

Maar hierbij moet wel worden opgemerkt dat Mattermost recentelijk de licentievoorwaarden restrictiever heeft gemaakt, wat vooral de functionaliteit en inzetbaarheid van de gratis zelf te hosten variant heeft ingeperkt. Het zet de toon voor waar Mattermost als bedrijf heen wil met de dienst. Zeker als het gratis zelf hosten een argument is om het in te zetten, is het aan te raden de voorwaarden goed door te nemen. Of om een volledig opensource alternatief te overwegen zoals bijvoorbeeld Matrix/Element, wat populair is bij Europese overheden (zoals de Franse overheid met Tchap en de Duitse Bundeswehr).

Commit. Develop. Share.

OpenClaw

CAUTION

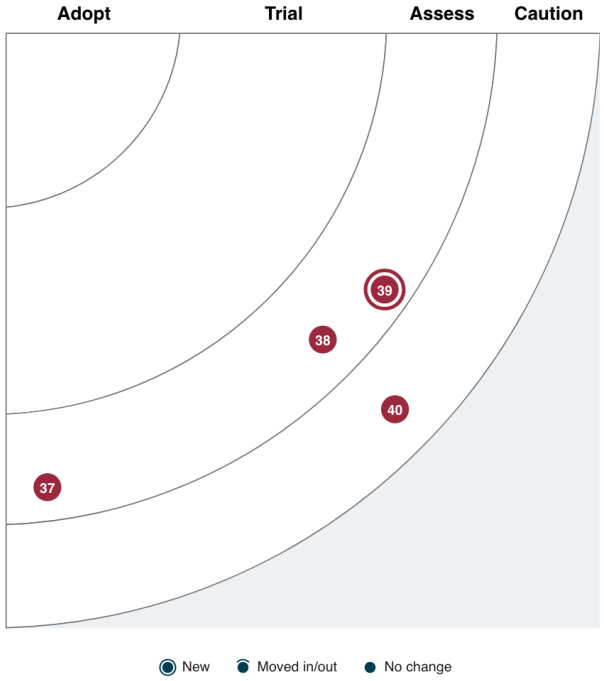
OpenClaw is een lokaal draaiende, autonome AI-agent die direct acties op een besturingssysteem uitvoert. Via de marktplaats ClawHub is de functionaliteit van de agent uit te breiden met zogeheten 'skills'. Deze skills omvatten diverse functionaliteiten, zoals het aanroepen van externe API's en het uitvoeren van prompts.

De directe systeemtoegang van de agent brengt echter risico's met zich mee, en in combinatie met een publieke marktplaats ontstaat hier een supply-chain risico vergelijkbaar met `npm` of `PyPI`. Op de marktplaats was er in eerste instantie geen controle op malafide skills, met als gevolg kwaadaardige scripts die zich richtten op lokale privédata. Inmiddels scant en blokkeert een nieuwe VirusTotal-integratie nu automatisch schadelijke code. Toch leunt de beveiliging nog te zwaar op systeem-prompts om het gedrag van de agent in te perken. Omdat een taalmodel gevoelig is voor misleiding via onbetrouwbare invoer, vormen tekstinstructies geen harde beveiligingsgrens. Wanneer de output van het model direct wordt omgezet in uitvoerbare commando's, blijft het risico op schadelijke prompt-injecties gewoon te groot.

Binnen ons vakgebied hechten we veel waarde aan de verantwoorde inzet van dit soort nieuwe technologie. Hoewel het veelbelovend is en recente updates helpen, blijft de standaardconfiguratie risicovol. Het lokaal draaien van deze software zonder de juiste isolatie is eigenlijk onverantwoord vanuit een security-perspectief. We vergelijken dit effectief met het uitvoeren van onbetrouwbare externe code via bijvoorbeeld een scriptrunner met volledige root-rechten. Het platform is werkbaar onder strikte voorwaarden, zoals strakke netwerkisolatie binnen een virtuele machine met minimale permissies. Totdat de standaardimplementaties van dit soort platforms harde beveiligingsgrenzen afdwingen, zetten wij OpenClaw momenteel op 'Caution' voor professioneel gebruik.

Commit. Develop. Share.

Talen & frameworks



ASSESS

37. Embabel Agent Framework
38. Arconia
39. EU Cloud Sovereignty Framework

CAUTION

40. Scala

Embabel Agent Framework

ASSESS

Embabel is een opensource AI-agentframework voor de JVM, gecreëerd door Rod Johnson, de grondlegger van het Spring Framework. Het stelt Java- en Kotlin-developers in staat om agentic flows te bouwen die LLM-interacties naadloos combineren met getypeerde domeinmodellen en bestaande applicatiecode. Een onderscheidend kenmerk is het gebruik van Goal-Oriented Action Planning (GOAP), een planningsalgoritme uit de gamesindustrie, waarmee agents dynamisch stappen combineren om doelen te bereiken.

Vanuit JDriven zien wij Embabel als een veelbelovende ontwikkeling voor het Java-ecosysteem. Het brengt AI-agentcapabilities naar de JVM op een manier die aansluit bij hoe Java-teams gewend zijn te werken: met sterke typering, compile-time checking en naadloze integratie in het Spring-ecosysteem. Dit verlaagt de drempel om GenAI te integreren in bestaande enterprise-applicaties, zonder over te hoeven stappen naar een Python-stack. De GOAP-aanpak biedt voordelen ten opzichte van frameworks die volledig op LLM's leunen voor planning: het is efficiënter, voorspelbaarder en beter testbaar.

Wel is Embabel een jong project dat zich nog moet bewijzen in productieomgevingen. De community is klein vergeleken met alternatieven zoals LangChain4j en Spring AI, waar Embabel overigens bovenop bouwt als hogere abstractielaag. Daarnaast vereist het begrip van concepten als GOAP en agentic flows, wat een leercurve met zich meebrengt. Om deze redenen plaatsen wij Embabel op *Assess*.

Commit. Develop. Share.

Arconia

ASSESS

Arconia is een add-on framework voor Spring Boot dat cloud-native capabilities toevoegt zonder van framework te wisselen.

Het framework richt zich op developer experience met onder andere Dev Services voor lokaal ontwikkelen en testen, Kubernetes-ondersteuning, service binding, observability en multi-tenancy. Dat maakt het interessant voor teams die al Spring Boot gebruiken en minder libraries en maatwerkconfiguratie willen combineren.

Wij plaatsen Arconia in *Assess*. Het framework is veelbelovend, maar werkt nog toe naar een eerste stabiele release begin 2026 en is daarom nog niet klaar voor brede inzetbaarheid. Daarbij komt ook dat het project voor nu door één developer is gestart en het afwachten is wat voor support er komt.

EU Cloud Sovereignty Framework

ASSESS

De Europese Commissie (EC) heeft een raamwerk laten samenstellen waarmee getoetst kan worden of cloudaanbieders voldoen aan het idee van soevereiniteit binnen Europa. Het heet het EU Cloud Sovereignty Framework. Het werkt met acht gewogen soevereiniteitscategorieën en vijf niveaus (Sovereignty Effectiveness Assurance Levels, of SEALS) van borging van de soevereiniteitseffectiviteit. Per categorie is een minimumborgingsniveau te besluiten dat als afwijzingsdrempel kan dienen, wanneer het raamwerk wordt gebruikt om een digitale dienstverlener te overwegen. De resultaten zijn ook samen te voegen tot een gemiddelde soevereiniteitsscore. Tezamen kan het gebruikt worden om inzicht te krijgen in de assen waarlangs een organisatie zijn "SEAL level" zou kunnen verhogen.

De EC heeft het framework in eerste instantie opgesteld voor de beoordeling van een eigen uitvraag voor digitale services. Op basis van die ervaring beveelt de EC Europese bedrijven aan om deze toetsing als een soort benchmark over te nemen voor het beoordelen van een clouddienstverlener, al heeft dat geen wetmatig karakter Op een vorige radar hebben we de Digital Sovereignty Readiness Assessment van Red Hat genoemd als een middel om soevereiniteit van de eigen organisatie te toetsen. We denken dat het EU Cloud Sovereignty Framework ook in die hoedanigheid te gebruiken is, en wellicht een betere graadmeter is voor het toetsen van bestaande en toekomstige digitale services. De categorieën, scoringsniveaus en wegingen van het Cloud Sovereignty Framework zijn namelijk fijnmaziger en gedetailleerder. En het neemt expliciet zaken als GDPR, NIS2 en DORA mee, die specifiek voor de EU van belang zijn, maar waar de assessment van Red Hat niet naar vraagt. JDriven raadt daarom aan het advies van de Europese Commissie mee te wegen en te kijken of het EU Cloud Sovereignty Framework zou kunnen helpen bij het beoordelen van soevereiniteitsdoelstellingen van de organisatie en die waar nodig naar een hoger plan te tillen.

Commit. Develop. Share.

Scala

CAUTION

Scala is ontstaan uit een behoefte om principes van functioneel programmeren te kunnen toepassen binnen het JVM ecosysteem. Het faciliteert ook objectgeoriënteerd programmeren en breidt de standaard features van Java op dat gebied uit, zoals bijv. via extension methods en operator overloading.

In 2020 plaatsten we Scala nog op Adopt ondanks de steile leercurve van Scala en de Scala Build Toolkit (sbt). De reden daarvoor was dat versie 3 uitkwam, die de taal niet alleen rijker maakte, maar ook gericht was op het vereenvoudigen ervan. Sindsdien zien we de vraag naar Scala in de markt echter significant afnemen, ten gunste van met name Kotlin. Expertise is lastig te vinden en is bovendien vaak specifiek gericht op ZIO, Cats of Akka, wat niet uitwisselbaar is. Een Scala-project is daardoor moeilijker en daarmee duurder te bemannen. Aangezien er een sterk alternatief als Kotlin bestaat en Java zelf ook is doorontwikkeld, kunnen wij Scala niet meer aanbevelen voor nieuwe projecten.

Commit. Develop. Share.



Commit.
Develop.
Share.