JDriven Tech Radar Najaar 2023

Onze kijk op recente ontwikkelingen in het veld

8e editie

Commit to know.

Develop to grow.

Share to show.

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 [https://www.thoughtworks.com/radar/byor].

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.

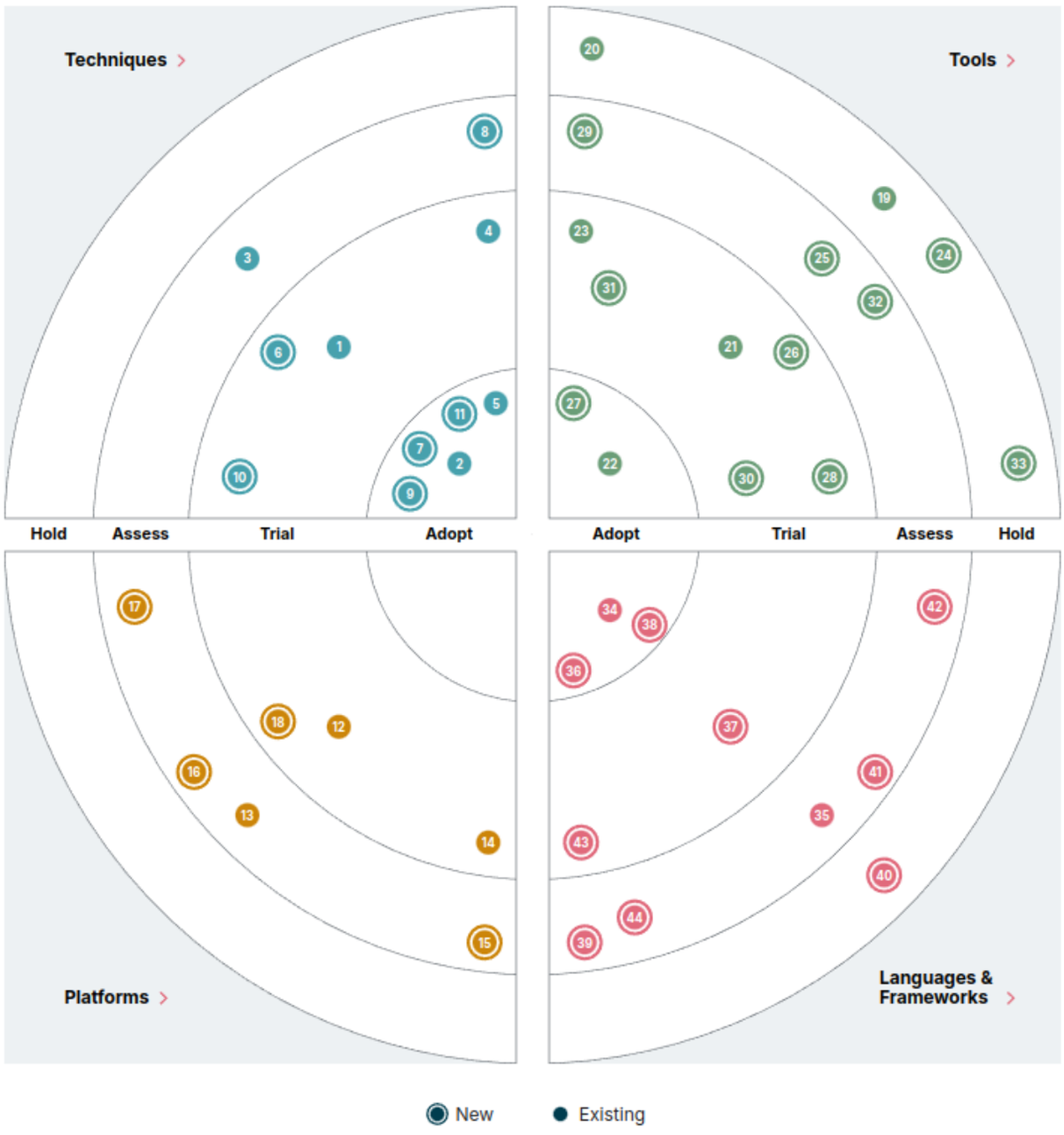
- **Talen & frameworks** die je ondersteunen bij de ontwikkeling van software
- **Platformen** waarop je software kunt uitvoeren
- **Technieken** die je helpen betere software te maken
- **Tools** ter ondersteuning van je ontwikkel- en delivery-proces

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.
- **Hold** → Niet (meer) gebruiken.

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

Commit. Develop. Share.

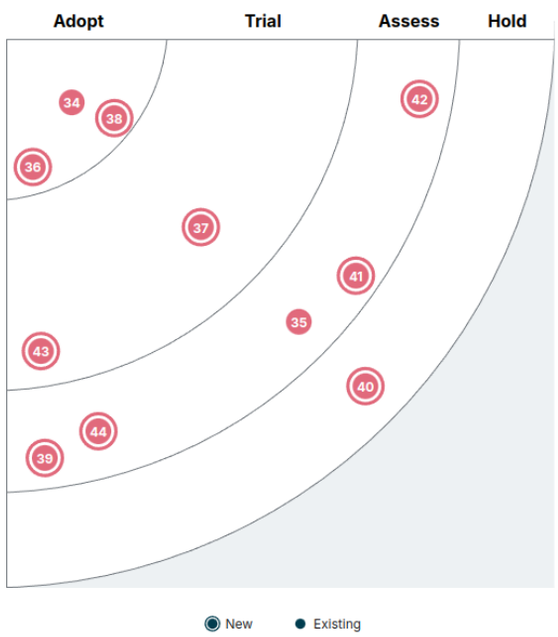


Commit. Develop. Share.

Tools	Talen & frameworks
ADOPT 22. k9s 27. Vector Databases **TRIAL** 21. Sigrid 23. Testkube 26. ChatGPT 28. Qodana 30. jlink 31. Maven Build Cache Extension **ASSESS** 25. Error Prone 29. Thunderdome Dev 32. Konsist **HOLD** 19. Docker 4 Desktop 20. Diffblue Cover 24. Lombok 33. JobRunr	**ADOPT** 34. Azure Bicep 36. Structured concurrency 38. Quarkus **TRIAL** 37. Langchain4j 43. Backstage **ASSESS** 35. Service Weaver 39. Jakarta 11 41. Terraform CDK 42. HTMX 44. JetBrains Compose **HOLD** 40. OpenTofu
Platforms	**Technieken**
ADOPT - **TRIAL** 12. Edge Computing 14. Apache Pulsar 18. Cloud events **ASSESS** 13. Dapr 15. Moderne platform 16. eBPF 17. Azure Container Apps **HOLD** -	**ADOPT** 2. DDD 5. Team Topologies 7. Dev Ex 9. Automatisch mergen van dependency updates 11. Prompt Engineering **TRIAL** 1. AI code assist 4. Developer Productivity Engineering 6. Inner Source 10. Unit testing voor alerting **ASSESS** 3. GPU Programming 8. Sustainable software engineering **HOLD** -

Commit. Develop. Share.

Talen & frameworks



Adopt

- 34. Azure Bicep
- 36. Structured concurrency
- 38. Quarkus

Assess

- 35. Service Weaver
- 39. Jakarta 11
- 41. Terraform CDK
- 42. HTMX
- 44. JetBrains Compose

Trial

- 37. Langchain4j
- 43. Backstage

Hold

- 40. OpenTofu

Azure Bicep

ADOPT

Azure Bicep [https://learn.microsoft.com/en-us/azure/azure-resource-manager/bicep/overview?tabs=bicep] is een domeinspecifieke taal (DSL) waarmee ontwikkelaars infrastructuren als code kunnen schrijven voor Azure-resources. Deze technologie vereenvoudigt het proces van het maken en implementeren van Azure-resources, door ontwikkelaars in staat te stellen declaratieve code te schrijven die de gewenste staat van hun infrastructuur definieert. Azure Bicep is een open-sourceproject ontwikkeld door Microsoft en is gebouwd op de bewezen Azure Resource Manager (ARM) infrastructuur.

Ontwikkelaars gebruiken Azure Bicep om Azure-resources zoals virtuele machines, opslagaccounts en netwerkinterfaces te beheren. Door Bicep te gebruiken, kunnen ontwikkelaars herbruikbare code-modules maken die kunnen worden gebruikt in verschillende omgevingen, waardoor de tijd en moeite die nodig zijn voor infrastructuurimplementatie worden verminderd. Azure Bicep maakt ook infrastructuurautomatisering mogelijk, waardoor het gemakkelijker wordt om Azure-resources op schaal te beheren en te onderhouden.

Wij geloven dat als je ervoor kiest om te werken met Infrastructure as Code, Azure Bicep de standaardkeuze is als je werkt met Azure. De adoptie ervan stelt organisaties in staat hun infrastructuur-implementatiesnelheid te verbeteren, de betrouwbaarheid te vergroten en het risico op menselijke fouten te verminderen. Bovendien zorgt de integratie met Azure Resource Manager ervoor dat Azure Bicep volledig compatibel is met bestaande ARM-templates, waardoor het gemakkelijk is aan te nemen voor organisaties die al Azure gebruiken. Ook is er een Azure Bicep Visual Studio-extensie die ontwikkelaars een gestroomlijnde ervaring biedt voor het schrijven van infrastructuren voor Azure-resources. Met deze extensie kunnen ontwikkelaars profiteren van IntelliSense en syntax highlighting, waardoor het gemakkelijker wordt om Bicep-code te schrijven, te onderhouden en te testen. Daarom raden wij aan dat bedrijven Azure Bicep gaan gebruiken om hun Azure-infrastructuurimplementatieproces te stroomlijnen en hun algehele infrastructuurbeheer te verbeteren.

Structured concurrency

ADOPT

Structured concurrency is een programmeerparadigma dat een gestructureerde aanpak biedt voor het parallel uitvoeren van taken. Het introduceert een aantal tools en abstracties om parallel uitgevoerde code te vereenvoudigen, met als doel de leesbaarheid, onderhoudbaarheid en het debuggen van code te verbeteren. In tegenstelling tot traditioneel thread management, zorgt structured concurrency ervoor dat alle gelijktijdige taken expliciet worden gescopet en beheerd, waardoor het eenvoudiger wordt om te redeneren over hun levenscyclus. Dit paradigma helpt ontwikkelaars om veelvoorkomende valkuilen van verkeerd thread management zoals resourcelekken en onvoorspelbaar gedrag te voorkomen, door een meer georganiseerde en deterministische manier te bieden om met concurrency om te gaan.

Wij raden aan om, waar mogelijk, gebruik te maken van structured concurrency bij het ontwikkelen van applicaties. In bijvoorbeeld Kotlin raden wij aan om gebruik te maken van Kotlin Coroutines bij code die baat heeft bij parallele uitvoering. In Java 21 is structured concurrency nog in preview. Toch raden we aan om deze nieuwe functie alvast uit te proberen en er vertrouwd mee te raken, zodat het kan worden gebruikt zodra het gereleaset wordt.

Quarkus

ADOPT

Quarkus [https://quarkus.io/] is een fijnwerkend framework voor developers die Context en Dependency Injection (CDI) en MicroProfile gewend zijn. Volgens hun eigen site is het: "A Kubernetes Native Java stack tailored for OpenJDK HotSpot and GraalVM, crafted from the best of breed Java libraries and standards." Het is vergelijkbaar met andere frameworks zoals Spring Boot, en is met name geschikt om serverapplicaties te bouwen.

Quarkus is ontwikkeld om het leven van developers gemakkelijk te maken. Er is weinig configuratie nodig om ermee te kunnen werken. Het framework biedt ondersteuning om eenvoudig in development-, test- of productiemodus te draaien. Quarkus gaat efficiënt om met geheugen. Met behulp van extensies is de functionaliteit eenvoudig uit te breiden. Het is een volwassen product geworden (inmiddels is versie 3 al uit). Quarkus biedt standaard ondersteuning voor het bouwen van Docker containers, zodat je die in Kubernetes of OpenShift kunt draaien.

Het gebruik van Smallrye Mutiny als Reactive Streams library maakt het mogelijk om imperatieve stijl naast reactive stijl te laten bestaan. Dit stelt developers in staat om laagdrempelig en stapsgewijs een applicatie volledig non-blocking te maken.

Testen wordt ook uitgebreid ondersteund. Met de `@QuarkusTest` annotatie kun je gemakkelijk integratietests opnemen in de build pipeline, waarbij ook test containers worden ondersteund. Het is ook mogelijk om jouw applicatie lokaal in developer mode te starten. Met de tool Skaffold [https://skaffold.dev/] (geen standaard onderdeel van Quarkus) kun je daarnaast snel een nieuwe deployment testen in een lokaal draaiend Kubernetes-cluster (bv. minikube of Rancher Desktop).

Het is zeer geschikt voor HTTP RESTful web development en integreert goed met gangbare message protocols, authentication protocols, datastores en platforms. Dankzij de rijkelijk aangeklede handleidingen is het bovendien relatief eenvoudig om Quarkus naar behoefte uit te breiden.

Het maken van native executables is al lang niet meer een onderscheidende factor voor Quarkus, maar door de brede ondersteuning en uitgebreide documentatie heeft Quarkus ons bewezen een serieuze speler te zijn in het aanbod van JVM CDI Frameworks. In het afgelopen jaar werd Quarkus bij diverse klanten succesvol ingezet (o.a. financiële sector). Hiermee heeft het zich als stabiel framework bewezen, waarmee Quarkus in deze editie de *Adopt* status heeft verkregen.

Langchain4j

TRIAL

Langchain4j [https://github.com/langchain4j/langchain4j] is een framework om eenvoudig applicaties te ontwikkelen met AI large language models (LLM).

Deze LLM's kunnen verscheidene taken met betrekking tot natuurlijke taalverwerking uitvoeren, zoals vertalingen maken en content samenvatten. Ze kunnen ook generatieve taken als contentcreatie uitvoeren en zijn ook extreem nuttig om vragen te beantwoorden.

Met dit framework kunnen LLM-providers eenvoudig worden uitgewisseld omdat er een abstractielaag tussen zit. Implementaties voor OpenAI GPT-4, Azure OpenAI en Google Vertex AI zijn beschikbaar, en ook lokale modellen kunnen gebruikt worden met Ollama [https://ollama.ai/]. Het is een tegenhanger van de Langchain [https://www.langchain.com/] library voor Python en Javascript. De Java-implementatie is volop in ontwikkeling en er komen steeds nieuwe functies bij.

Indien je een Java-applicatie wilt uitbreiden met AI-functies, dan is het bekijken van dit framework de moeite waard. Er kan dan ook eenvoudig gewisseld worden tussen en getest worden met verschillende modellen.

Wij bij JDriven geloven dat de abstractielaag van dit framework leidt tot flexibelere software zonder "vendor lock-in", wat een voordeel is, gezien het feit dat alle features en modellen nog volop in beweging zijn. Als er bijvoorbeeld wegens privacy of betere resultaten van model gewisseld moet worden, is dit eenvoudig te realiseren.

Backstage

TRIAL

Backstage [https://backstage.io/] is een open source developer portal. Het is door Spotify gemaakt en gedoneerd aan de Cloud Native Computing Foundation (CNCF). Het is in 2020 gepubliceerd en is dus een relatief nieuw platform.

Het doel van Backstage is het bieden van een centrale plek voor software, tooling en documentatie. Het biedt daarbij de mogelijkheid om templates te schrijven, waarmee snel een nieuw project is op te starten.

Backstage heeft veel integraties en plugins voor verschillende tools om dit doel te realiseren. De sterke punten van Backstage komen het beste naar voren als het bij grote projecten wordt gebruikt.

Voorbeelden van grote bedrijven die Backstage gebruiken:

- LinkedIn
- Vodafone
- Lego
- Siemens

Backstage staat in *Trial*, want het nut van een developer portal is niet altijd duidelijk en kan verschillen per project. De cultuur en omvang van een bedrijf zijn belangrijke factoren.

Service Weaver

ASSESS

Service Weaver [https://opensource.googleblog.com/2023/03/introducing-service-weaver-framework-for-writing-distributed-applications.html] is een framework voor het ontwikkelen van gedistribueerde applicaties. Het is ontworpen om ontwikkelaars te helpen bij het bouwen van complexe systemen die gebruikmaken van een gedistribueerde architectuur.

Service Weaver omvat een reeks abstracties en patronen die de definitie, communicatie en samenwerking van services mogelijk maken. Door een gedecentraliseerde aanpak te hanteren, kunnen services autonoom werken en communiceren via asynchrone berichten.

Service Weaver is een nieuw Framework dat veel potentieel biedt bij de ontwikkeling van gedistribueerde applicaties. Daarom is Service Weaver geplaatst in het *Assess* kwadrant van deze editie van onze Tech Radar.

Jakarta 11

ASSESS

Jakarta 11 [https://jakarta.ee/specifications/platform/11/] is de nieuwste (aankomende) versie van Jakarta EE. Deze nieuwste versie heeft een aantal interessante features zoals Virtual Threads en Jakarta Data (vergelijkbaar met Spring Data). Het ondersteunt alleen Java 21 en het verwijdert een aantal oude specificaties. Dit alles zorgt ervoor dat dit een interessante versie van Jakarta EE wordt wat het volgens ons de moeite waard maakt om te beoordelen.

Terraform CDK

ASSESS

Terraform maakt gebruik van HashiCorp Configuration Language (HCL) om op een cloudagnostische wijze de infrastructuur van een applicatie te beheren. HCL is een declaratieve taal om cloudresources te beschrijven. Echter betekent het gebruik hiervan dat developers een nieuwe taal moeten leren om met deze tooling overweg te kunnen. Door gebruik te maken van CDK for Terraform (CDKTF) kan men nu ook aan de slag met Java, C#, Python, TypeScript of Go voor het beheren van infrastructuur.

CDKTF verdient een plekje op de Tech Radar omdat Terraform onderhand een bewezen technologie is, waar deze toevoeging een aantal voordelen biedt ten opzichte van de conventionele aanpak. Zo krijgt men code completion en IDE-support cadeau, integreert de infrastructuurdefinitie beter met bestaande tooling zoals codeformatters en linters, en wordt het laagdrempeliger voor developers om Terraform te gaan gebruiken in hun projecten.

Op 21 augustus 2022 heeft HashiCorp de tooling als 'production ready' aangemerkt, met de kanttekening dat de tooling nieuw is en daarom nog niet volledig volwassen. Daarnaast is de documentatie omtrent bepaalde onderwerpen benedenmaats, en introduceert de CDK een aantal nieuwe concepten en dus wat extra complexiteit. Al met al een interessante ontwikkeling om in de gaten te houden, zeker als Terraform al gebruikt of overwogen wordt.

HTMX

ASSESS

htmx is een JavaScript-library gericht op het bouwen van een hypermedia-driven application [https://hypermedia.systems/hypermedia-systems/] (HDA). HDA en htmx zijn een reactie op de zware Single Page Application JavaScript-frameworks als React en Angular, en leggen de aandacht terug op de kracht van HTTP, HTML, hyperlinks en REST, samengevat als *hypermedia architecture*. De opkomst van SPA's kan verklaard worden door het feit dat het moeilijk was om in web 1.0 vlotte UI-overgangen en interactiviteit te bouwen zonder trage en merkbare page reloads. HTML biedt nog altijd slechts 2 HTTP methods (GET en POST) en 2 HTML-elementen (anchor en form) voor interactiviteit met server content.

De oplossing, die SPA-frameworks voor dit gebrek aan HTML-interactiviteit hebben geïntroduceerd, heeft echter wat nadelen: ze verandert RESTful backend-API's in data-API's die JSON serveren, maakt JavaScript leidend om HTML te tonen, en introduceert een complexe frontend-architectuur leunend op vele JavaScript-libraries voor schijnbaar eenvoudige zaken zoals URL's, web content indexering (SEO), de back button, caching, progressive enhancement en loose coupling. htmx gebruikt JavaScript om de bovengenoemde tekortkomingen van HTML te adresseren, om daarna uit de weg te gaan en websites weer hypermedia-gedreven te kunnen maken, en daarmee de tekortkomingen van SPA's te vermijden.

We zien een toename van interesse in htmx en HDA onder full-stack developers, en zelfs onder frontend developers die hun carrière begonnen zijn met het toepassen van SPA-frameworks en de genoemde problemen herkennen. JDriven adviseert om bij het opzetten van een frontend-architectuur te kijken of een SPA-architectuur wel de beste oplossing is, of dat HDA en htmx een goede fit zouden kunnen zijn. Let daarbij op de toepasbaarheid van tools in het frontend-ecosysteem zoals design systems, Tailwind CSS en Playwright. Omdat htmx in feite een terugkeer naar serverside rendering is, is het ook belangrijk om de ontwikkelingen op het gebied van Edge Computing, die ook spelen op het gebied van SPA's, mee te nemen in de overweging.

JetBrains Compose

ASSESS

JetBrains zet met Compose Multiplatform in op het ontwikkelen van businessfunctionaliteit in één codebase, met een op diverse platforms toegespitste UI. Het combineert de krachten van Jetpack Compose en Kotlin Multiplatform om ontwikkeling voor servers, Android, iOS, web (via Kotlin/Wasm), en desktop (Linux, macOS, Windows) vanuit een enkele codebase mogelijk te maken.

Voor developers die reeds bekend zijn met de wereld van Kotlin en/of Android development, is Compose Multiplatform veelbelovend. Het biedt de mogelijkheid om business logic en UI-code bruikbaar te maken voor alle genoemde platforms. De vraag blijft altijd, of de effectiviteit van het werken met één codebase opweegt tegen de extra lagen van abstractie, die nodig zijn om te komen tot een grote gemene deler in de code. Ook is het zo, dat de iOS-functionaliteit nog in alpha-fase is, en Kotlin/Wasm voor web nog experimenteel. Maar het biedt een interessante optie voor organisaties die willen investeren in business logic, een homogene developer skillset en een uniforme UI over meerdere platforms.

OpenTofu

HOLD

HashiCorp heeft de sourcecode-licentie van zijn producten, zoals Terraform, aangepast naar een Business Source License [https://www.hashicorp.com/bsl]. Dat heeft onder gebruikers van Terraform geleid tot zorgen, dat het omringende ecosysteem van tools daaronder gaat lijden. Op 23 augustus 2023 is er een fork gemaakt van Terraform, genaamd OpenTofu. Het doel is om dit als opensourceproject te laten voortbestaan onder beheer van de Linux Foundation.

Nadere inspectie van de uitleg [https://www.hashicorp.com/license-faq] van HashiCorp over deze licentievoorwaarden wijst echter uit, dat de wijziging naar een BSL alleen bedrijven treft, die commerciële producten maken die concurreren met HashiCorp's producten, door gebruik te maken van HashiCorp's eigen producten. Dat is voor de grootste groep gebruikers van Terraform niet van toepassing. Daarom zien wij voor hen geen reden om uit te wijken naar een alternatief voor Terraform zoals OpenTofu, wat we daarom op *Hold* plaatsen.

Techniques

Trial

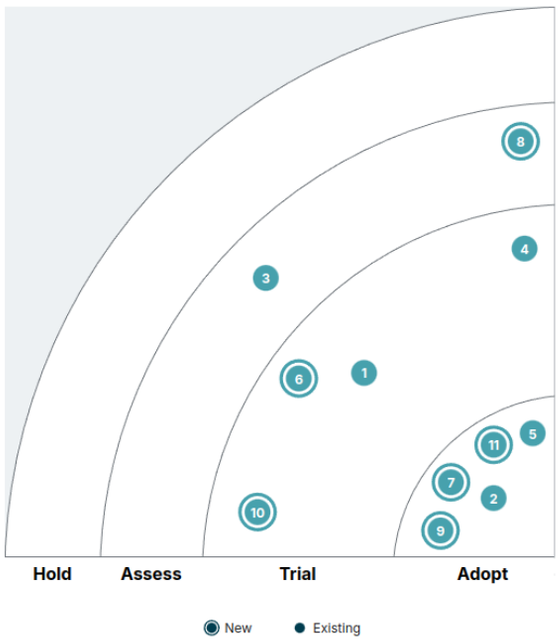
- 1. AI code assist
- 4. Developer Productivity Engineering
- 6. Inner Source
- 10. Unit testing voor alerting

Adopt

- 2. DDD
- 5. Team Topologies
- 7. Dev Ex
- 9. Automatic merging of dependency updates
- 11. Prompt Engineering

Assess

- 3. GPU Programming
- 8. Sustainable software engineering



DDD

ADOPT

Domain Driven Design (DDD) is een techniek die onderverdeeld kan worden in strategisch en tactisch ontwerp. Het eerste biedt manieren om een probleemdomein te benaderen en bounded contexts en hun onderlinge afhankelijkheden te identificeren, alsmede een ubiquitous language. Het tweede biedt manieren en building blocks om deze bounded contexts te modelleren en implementeren.

Alhoewel DDD als een concept reeds 20 jaar oud is, zien we dat het recentelijk weer in de schijnwerpers staat. Tools voor CQRS, Event Sourcing en message-driven architectures bestaan al en blijven doorontwikkeld worden, en technieken zoals event storming en domain storytelling worden vaker gebruikt. Maar belangrijker nog, is dat architecten en developers ondervinden dat de overstap van monolieten naar microservices hun problemen soms heeft verergerd; in het bijzonder wanneer bounded contexts en de afhankelijkheden daartussen onvoldoende zijn gemodelleerd en begrepen. Zij grijpen nu naar DDD voor hulp.

Team Topologies

ADOPT

Team Topologies [https://teamtopologies.com/key-concepts] is een praktisch model voor organisatorisch ontwerp en teaminteractie, gebaseerd op vier fundamentele teamtypes en drie interactiepatronen. Het is een model dat teams ziet als de basis van delivery, en dat teamstructuren en communicatielijnen laat groeien met technologische en organisatorische volwassenheid. Team Topologies biedt een duidelijke manier voor teams om met elkaar samen te werken, teneinde de resulterende softwarearchitectuur helderder en beter onderhoudbaar te maken. Daarbij interpreteert het problemen tussen teams als waardevolle signalen die kunnen helpen bij een zelfsturende organisatie.

Team Topologies benadrukt het optimaliseren van teaminteractie, en vult daarmee het strategisch

ontwerp van Domain Driven Design aan bij het in kaart brengen van interactie tussen softwaresystemen. In combinatie met het bewustzijn van Conway's Law en team cognitive load, helpt dit bij het produceren van architecturen die onderhouden kunnen worden door een organisatie.

Dev Ex

ADOPT

DevEx, afgeleid van Developer Experience, is een cruciaal facet van softwareontwikkeling dat de algemene ervaring van ontwikkelaars tijdens het werken aan een project of in een specifieke ontwikkelingsomgeving omvat. Het is gericht op de bruikbaarheid van ontwikkelingstools, efficiënte workflows, duidelijke documentatie, en de tevredenheid en productiviteit van ontwikkelaars.

Binnen de context van softwareontwikkeling is DevEx de sleutel tot het optimaliseren van processen en het bevorderen van een positieve werkomgeving voor ontwikkelaars. Door te zorgen voor intuïtieve tools, gestroomlijnde workflows en effectieve ondersteuning, draagt het bij aan het versnellen van ontwikkelingscycli en het leveren van hoogwaardige softwareproducten. Het is de brug die ontwikkelaars verbindt met de mogelijkheden van technologie, en het stimuleert innovatie in de dynamische wereld van softwareontwikkeling.

Automatisch mergen van dependency updates

ADOPT

Het automatisch mergen van dependency updates is vandaag de dag steeds belangrijker aan het worden. Tegenwoordig komt het steeds vaker voor dat kwetsbaarheden ontdekt worden die meekomen met dependencies in projecten. Wij vinden dat het automatisch updaten en mergen van de desbetreffende dependencies de juiste manier is, zodat code zo up-to-date mogelijk blijft.

Enkele tools die het automatisch mergen van de dependency updates mogelijk maken zijn onder andere Renovate [https://github.com/renovatebot/renovate] en Dependabot [https://github.com/dependabot/dependabot-core]. Wat deze in essentie doen is het continu scannen van de dependencies in het project en hier merge requests voor aanmaken indien er updates beschikbaar zijn. Ontwikkelaars kunnen deze merge requests handmatig of automatisch mergen.

JDriven kiest ervoor om deze techniek op te nemen in de Tech Radar en in *Adopt* te plaatsen, omdat het teams efficiënt laat werken door het bijhouden van dependency versies (deels) te automatiseren, wat leidt tot kwalitatieve en veilige software.

Prompt Engineering

ADOPT

Prompt engineering is een discipline die draait om het vormgeven van nauwkeurige, effectieve aanwijzingen voor door AI aangedreven modellen, zoals GPT-X (Generative Pre-trained Transformer). Deze aanwijzingen sturen de reacties van het AI-model in de richting van het gewenste resultaat. Het is een proces dat experimenteren, iteratie, verificatie en afstemming omvat. Prompt engineering biedt verschillende voordelen die het potentieel hebben om software-engineering te verbeteren:

- Versnelde Ontwikkeling
- Verbeterde Creativiteit
- Consistentie en Betrouwbaarheid
- Leren en Groeien

Prompt engineering stimuleert een cultuur van continu leren en groeien. Ontwikkelaars kunnen

modelreacties analyseren, hun aanwijzingen aanpassen en hiervan leren. Het verstrekken van deze inzichten aan de makers van het AI-model kan resulteren in geoptimaliseerde modellen die betere en robuustere resultaten opleveren. Hoewel het gebruik van AI-modellen snelle resultaten kan opleveren, vereist het behalen van hoogwaardige resultaten een investering van tijd, moeite en het verwerven van nieuwe vaardigheden.

AI code assist

TRIAL

Een AI-codeassistent is een tool die gebruik maakt van machine learning-algoritmen en statistische modellen om code te voorspellen. Het helpt ontwikkelaars bij het schrijven van code door suggesties te doen voor het voltooien van codefragmenten, het corrigeren van fouten en het geven van aanbevelingen voor optimalisatie. Tevens kan er gebruik gemaakt worden van prompt-based engineering, waar met natuurlijke taal code wordt gegeneerd.

Als ontwikkelaar kun je een AI-codeassistent gebruiken door deze te integreren als plugin in je IDE (Integrated Development Environment). Hierdoor krijg je tijdens het typen van de code direct suggesties en correcties aangeboden, waardoor je productiever en efficiënter kunt werken. Bovendien kan een AI-codeassistent je helpen om nieuwe talen en frameworks sneller te leren, doordat het suggesties doet op basis van best practices en veelgebruikte patronen.

Op dit moment bevinden de AI-codeassistenten zich nog in de kinderschoenen. Zo hebben AI-codeassistenten weinig kennis van specifieke domeinen, genereren ze mogelijk foute code en kan het eigendomsrecht ook een probleem zijn. Wij geloven echter, dat ondanks deze reële bezwaren, binnen vijf jaar de AI-code assists mainstream zullen worden in software ontwikkeling. Tools als Tabnine [https://www.tabnine.com], GitHub Copilot [https://github.com/features/copilot/], Machinet [https://www.machinet.net] en IntelliCode [https://visualstudio.microsoft.com/services/intellicode/] strijden om de aandacht. De vraag is dan ook niet of je dit als ontwikkelaar gaat gebruiken, maar wanneer. Wij raden daarom aan om dergelijke tooling nu te onderzoeken, zodat wanneer AI-codeassistenten volwassen zijn, het meteen voor je productiecode gebruikt kan worden.

Developer Productivity Engineering

TRIAL

Developer Productivity Engineering (DPE) [https://gradle.com/developer-productivity-engineering/] is een tak van software engineering die zich richt op het verbeteren van de efficiëntie en productiviteit van ontwikkelaars op basis van meetbare data. DPE richt zich op het ontwerpen en bouwen van tools, processen en infrastructuren die de workflow van ontwikkelaars kunnen stroomlijnen en hun ontwikkelingscyclus kunnen versnellen. DPE omvat een breed scala aan activiteiten, zoals bijvoorbeeld het ontwerpen en implementeren van geautomatiseerde testsystemen en het optimaliseren van de build- en deploy-processen om de feedback loop te verkorten. Doordat het optimaliseren gedaan wordt op basis van meetbare data kan dit team bewijzen hoeveel tijd door hun activiteiten wordt bespaard.

Gradle is dit op het moment in de markt aan het brengen en vooral voor grotere bedrijven is dit een interessante trend om te volgen. Doordat het op meetbare resultaten is gebaseerd en het op verschillende niveaus kan worden geïmplementeerd, vinden we het geschikt om het op *Adopt* te zetten.

Inner Source

TRIAL

Inner source verwijst naar het toepassen van open source principes binnen een organisatie. Het gaat erom de samenwerkende, transparante en gedecentraliseerde aanpak die vaak wordt gezien in open source projecten toe te passen op de interne softwareontwikkeling van de organisatie. Het is een alternatief op closed source, waarbij normaliter repositories enkel beschikbaar zijn voor het verantwoordelijke team.

In een omgeving waar inner source wordt toegepast, zijn teams in staat binnen het bedrijf openlijk code repositories in te zien. Dit bevordert het samenwerken, verbetert de codeanalyse, en geeft de mogelijkheid voor het bijdragen van codewijzigingen via Pull Requests over verschillende projecten heen. Het bevordert een grotere transparantie, stimuleert kennisdeling en stelt ontwikkelaars in staat om over teams en afdelingen heen te werken.

Dit model kan de kwaliteit van software verbeteren, de ontwikkeling versnellen en een cultuur van samenwerking en innovatie binnen de organisatie bevorderen door gebruik te maken van de gezamenlijke expertise en inspanningen.

JDriven staat voor commit, develop, share met de opensourceprincipes in gedachten. Daarom vinden wij dat organisaties moeten overwegen de broncode open te stellen voor interne ontwikkelaars.

Unit testing voor alerting

TRIAL

Binnen het domein van software-observability en -monitoring is het opzetten van robuuste alertmechanismen cruciaal. De onvoorspelbaarheid van events vereist nauwkeurige en responsieve alerts, vaak gebouwd op complexe regels. Echter, de validatie van deze regels voltrekt zich vaak pas in productie wanneer er onverwachts echte situaties voordoen.

De techniek 'unit testing voor alering' is ontworpen om de precisie van de regels en alerts te verhogen door proactieve evaluatie en verfijning. Deze aanpak stelt teams in staat om regels grondig te definiëren en te valideren voordat ze live getriggerd worden door incidenten. Dit verhoogt het vertrouwen in de regels en alerts aanzienlijk.

Het primaire doel heeft een dubbele functie: het verminderen van valse alarmen en ervoor zorgen dat echte problemen snel en nauwkeurig worden benadrukt. Door verschillende scenario's en omstandigheden te simuleren, kunnen teams beoordelen hoe goed hun rules presteren en deze verfijnen voor optimale responsiviteit.

Belangrijke tools zoals Prometheus bieden speciale ondersteuning voor het testen van rules. Door deze preventief te valideren, hebben we een aanzienlijke afname van valse positieven. Hierdoor is een snellere en nauwkeurigere reactie op echte problemen mogelijk.

JDriven is van mening dat teams en organisaties die met rules en alerts werken, zouden moeten beginnen met het toevoegen van unit tests voor alerts.

GPU Programming

ASSESS

General purpose programming on graphics processing units (GPU) is een techniek waarmee generieke programma's op een GPU kunnen worden uitgevoerd. Deze techniek wordt veelal gebruikt voor toepassingen die veel verwerkingskracht nodig hebben, zoals machine learning, data-analyse en complexe wetenschappelijke en financiële simulaties. Oorspronkelijk zijn GPU's ontworpen voor het verwerken en renderen van computer graphics, maar GPU's bieden grote snelheidsvoordelen voor alle type toepassingen die een grote mate van parallelle verwerking toestaan.

We hebben GPU-programmering geselecteerd voor deze editie van onze Tech Radar, omdat het snel populairder wordt binnen verschillende sectoren. De hogere verwerkingssnelheid kan taken, waarvoor eerder niet de tijd beschikbaar was om op het resultaat te wachten, binnen bereik brengen. De specifieke architectuur van deze processoren maakt het echter ingewikkeld om software te ontwikkelen. De veelgebruikte GPGPU-programmeertalen CUDA en OpenCL zijn bovendien geïnspireerd op C en redelijk low-level. Om effectieve code te schrijven, veronderstellen deze talen een gedegen kennis van het GPGPU-platform. Hierdoor is het nog steeds te vroeg voor universele adoptie binnen de industrie.

Desondanks geloven we dat, zelfs voor Java-ontwikkelaars, GPU-programmeren blijft groeien in belang. Een mooi voorbeeld is onder andere de snelle opkomst van Large Language Models (LLM's), zoals ChatGPT. Deze zijn sterk afhankelijk van GPU's om een gebruiker binnen een acceptabele tijd van een antwoord te voorzien.

Tot slot is GPU-programmeren een krachtige technologie die de manier waarop we gegevens verwerken en analyseren transformeert.

Naarmate GPU's blijven evolueren en toegankelijker worden, kunnen we in de toekomst nog meer wijdverspreide adoptie van deze technologie verwachten.

Voor nu raden we bedrijven aan om te beoordelen of er taken en/of services zijn die kunnen profiteren van de GPU en hiermee nieuwe diensten te kunnen aanbieden.

Sustainable software engineering

ASSESS

Groene software is een opkomende discipline op het snijvlak van klimaatwetenschap, softwareontwerp, elektriciteitsmarkten, hardware en datacenterontwerp. Groene software is CO_2 -efficiënte software, wat betekent dat het zo min mogelijk CO_2 uitstoot. Slechts drie activiteiten verminderen de CO_2-uitstoot van software; energie-efficiëntie, koolstofbewustzijn en hardware-efficiëntie.

Een softwarearchitectuur die rekening houdt met koolstofefficiëntie is er één waarin ontwerp- en infrastructuurkeuzes zijn gemaakt om het energieverbruik en dus de koolstofuitstoot te minimaliseren. De meetinstrumenten en het advies op dit gebied worden steeds volwassener, waardoor het voor teams haalbaar wordt om CO_2-efficiëntie te overwegen naast andere factoren zoals prestaties, schaalbaarheid, financiële kosten en veiligheid.

De cloud heeft nu een grotere ecologische voetafdruk dan de luchtvaartsector. Wij vinden dat we als software engineers en architecten ook een verantwoordelijkheid hebben in het verminderen van deze voetafdruk en het verbeteren van het klimaat. Door hierin bewuste keuzes te maken, kunnen we die stappen ook zetten. Daarnaast hebben efficiënte systemen een bijkomend voordeel: minder resources zorgt voor minder kosten.

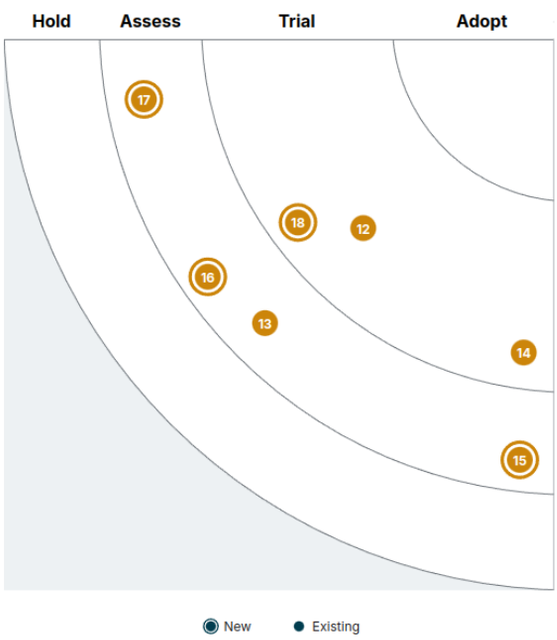
Platforms

Trial

- 12. Edge Computing
- 14. Apache Pulsar
- 18. Cloud events

Assess

- 13. Dapr
- 15. Moderne Platform
- 16. eBPF
- 17. Azure container apps



Edge Computing

TRIAL

Edge computing brengt rekenkracht en dataopslag dichter bij de eindgebruiker, die daardoor snellere berekeningen en responsetijden mag verwachten. Dit is vergelijkbaar met hoe content delivery networks de performance van statische websites verbeteren. Recentelijk hebben diverse cloud providers edge functions aan hun edge nodes toegevoegd. Edge functions introduceren de optie om bepaalde functionaliteit van de client te verplaatsen naar deze edge nodes.

Vendors zoals Netlify, AWS en Vercel zijn hun portfolio rondom edge functions en bijbehorende middleware aan het uitbreiden. Vanuit de frontend bezien is er nu een plek dichter bij de client, die een deel van de server side rendering kan verzorgen. Dit heeft het voordeel van snelheid, maar het introduceert ook het beveiligingsrisico van meer contactpunten in je architectuur, en kan een toename van complexiteit van je architectuur betekenen. Dientengevolge zien we potentie en risico's, wat het rechtvaardigt om met deze nieuwe oplossingen te experimenteren op het punt tussen Single Page Application (SPA) frameworks en de backend.

Apache Pulsar

TRIAL

Apache Pulsar [https://pulsar.apache.org/] is een open-source gedistribueerd pub/sub-messagingsysteem dat een hoge mate van schaalbaarheid, betrouwbaarheid en prestaties biedt. Vergelijkbaar met Apache Kafka en RabbitMQ biedt Pulsar een manier voor applicaties om met elkaar te communiceren door berichten uit te wisselen via een broker.

Pulsar heeft echter enkele verschillen in vergelijking met Kafka en RabbitMQ. Een van de belangrijkste voordelen van Pulsar is de mogelijkheid om zowel een queue- als pub/sub-model te ondersteunen in één systeem, wat flexibiliteit biedt voor verschillende typen berichtenverkeer. Pulsar biedt ook geavanceerde functies zoals dynamische partitionering, wat zorgt voor betere load balancing en resourcegebruik in gedistribueerde omgevingen.

In termen van schaalbaarheid is de architectuur van Pulsar ontworpen om miljoenen berichten per seconde te verwerken met lage latentie en hoge doorvoer, waardoor het een goede keuze is voor real-time streamingtoepassingen. Pulsar heeft ook een ingebouwd gelaagd opslagsysteem waarmee gegevens efficiënt kunnen worden bewaard en opgehaald.

Hoewel Apache Kafka en RabbitMQ hun eigen sterke punten en gebruiksmogelijkheden hebben, is Apache Pulsar een aantrekkelijke optie voor bedrijven die op zoek zijn naar een zeer schaalbaar, duurzaam en flexibel berichtensysteem dat zowel queue- als pub/sub-berichtensystemen aankan.

Bij JDriven zien we dat steeds meer klanten experimenteren met Apache Pulsar. Gezien bovengenoemde voordelen hebben wij Pulsar op *Trial* gezet. Voor nieuwe landschappen kan Apache Pulsar een interessant alteratief voor Kafka of RabbitMQ zijn. Wanneer er geen problemen zijn op bestaande platforms zie wij echter nog geen redenen om deze te migreren naar Pulsar.

Cloud events

TRIAL

Cloud Events [https://cloudevents.io/] is een open standaard die is ontwikkeld om de compatibiliteit en samenwerking tussen event-gedreven systemen in de cloud te verbeteren. Het biedt een uniforme manier om events te specificeren, ongeacht de verschillende cloudplatformen, services of systemen die betrokken zijn. Deze standaardisatie stelt organisaties in staat om events te beschrijven in een formaat dat door verschillende cloudproviders kan worden begrepen en verwerkt, waardoor de complexiteit van event-verwerking tussen systemen wordt verminderd.

Cloud Events zijn gebaseerd op een set van metadata en standaardformaten voor events beschreven in JSON of YAML. De metadata van Cloud Events biedt een consistente en gestructureerde manier om informatie over events vast te leggen. Dit omvat details zoals de identificatie van het event, de tijd waarop het plaatsvond en de bron waar het event vandaan komt. Daarnaast biedt Cloud Events de mogelijkheid om zelf optionele metadatavelden naast de standaard metadatavelden vast te leggen.

Cloud Events biedt flexibiliteit en draagbaarheid door een gemeenschappelijke taal te bieden voor het beschrijven van gebeurtenissen, ongeacht de onderliggende technologieën, protocollen en infrastructuur. De voordelen van het implementeren van Cloud Events zijn onder meer een verbeterde interoperabiliteit, verhoogde flexibiliteit en vereenvoudigde integratie tussen verschillende cloudservices en applicaties. Dit stelt ontwikkelaars in staat om gebeurtenisgestuurde architecturen te bouwen die gemakkelijk schaalbaar, flexibel en onderling uitwisselbaar zijn in verschillende cloudomgevingen.

JDriven zet Cloud Events op *Trial* omdat wij denken dat het het onderzoeken waard is voor met name grote enterprise-omgevingen, die baat hebben bij een bredere standaard.

Dapr

ASSESS

Dapr [https://dapr.io/] (Distributed Application Runtime) is een door Microsoft ontwikkeld platform dat de connectiviteit van microservices vereenvoudigt. Het biedt een abstractielaag over technieken die vaak in elke microservice moeten worden geïmplementeerd. Denk hierbij aan pub/sub-messaging, state management, secrets management, configuratie, tracing en logging. Dapr draait in een sidecar naast de microservice. De microservice communiceert vervolgens met de sidecar via een SDK die beschikbaar is voor o.a. Java, .NET, Python, Javascript en Rust. Wij hebben Dapr opgenomen in het *Assess* kwadrant van deze editie van de Tech Radar omdat het de ontwikkeling van een microservicelandschap sterk vereenvoudigt. Een nadeel dat wij zien is dat het waarschijnlijk lastig weer uit een applicatie te krijgen is.

Moderne platform

ASSESS

Het Moderne Platform is een SaaS-oplossing gebouwd bovenop de open-source software OpenRewrite. Het biedt geautomatiseerde code-refactoringstools waarmee ontwikkelaars met een eenvoudige klik of CLI-opdracht snel en moeiteloos complete codebases kunnen migreren. Dit wordt gedaan met behulp van meegeleverde recepten die bijvoorbeeld Spring Boot-apps kunnen upgraden van versie 2 naar 3.

In onze Tech Radar hebben we eerder OpenRewrite genoemd. Een SaaS-platform dat de sterke punten van OpenRewrite combineert met tools om de ervaring van ontwikkelaars rond het gebruik van OpenRewrite te verbeteren, is iets waar we positief tegenover staan. Concurrenten zoals Snyk, Dependabot en Sonar waarschuwen alleen voor softwarekwetsbaarheden en bevindingen. Wat het Moderne Platform voor ons vooral interessant maakt, is dat het deze problemen ook onmiddellijk oplost, waardoor ontwikkelaarstijd vrijkomt.

Vanwege onze beperkte ervaring met het Moderne Platform zijn we nog niet zeker of het geschikt is voor missiekritieke toepassingen. Wanneer de CLI niet wordt gebruikt, maakt een Moderne-agent verbinding met in-house code- en artefactopslagplaatsen, wat voor sommige organisaties een obstakel kan zijn. Bovendien hangt de effectiviteit van het platform af van het aantal recepten en de taalondersteuning die ze hebben, iets waar ze momenteel hard aan werken om uit te breiden.

eBPF

ASSESS

eBPF (extended Berkeley Packet Filter) is een techniek om programma's in een sandbox binnen de Linux-kernel te draaien, door middel van een Just-in-Time compiler. Dit stelt gebruikers in staat om programma's te schrijven die systeem- en netwerkgedrag observeren en filteren, of extra veiligheidsmaatregelen toepassen, zonder restricties op systeemtoegang. De JIT-compiler zorgt ervoor dat de programma's niet buiten de geprogrammeerde context kunnen opereren. Bovendien verifieert de compiler bepaalde eigenschappen voor veiligheid en "liveness" voordat een eBPF-programma wordt uitgevoerd. Traditioneel wordt dergelijke aangepaste kernelfunctionaliteit verkregen door het schrijven van custom patches voor de Linux-kernel, of het schrijven van kernelmodules. Het nadeel van deze methoden is dat dit erg bewerkelijk is en de stabiliteit van het gehele besturingssyteem in gevaar kan brengen. Bovendien is de in-kernel API van Linux niet stabiel, waardoor dergelijke aanpassingen vaak voor een beperkt aantal kernelversies werken. Daardoor is frequent onderhoud en een grote kennis van de kernel nodig.

We hebben er dit najaar voor gekozen deze techniek op *Assess* te zetten. eBPF lost een beperkte klasse van problemen op, met name voor organisaties die een grote hoeveelheid infrastructuur beheren of speciale eisen aan veiligheid stellen. Ook vereist het schrijven van dergelijke programma's veel specialistische kennis. Een gebruiker heeft daarnaast specifieke rechten op een doelsysteem nodig, en een Linux-kernel waarbij de JIT-functionaliteit beschikbaar is gemaakt. Deze functionaliteit is daardoor meestal alleen beschikbaar voor organisaties die hun eigen systemen beheren; publieke cloudproviders zullen deze functionaliteit veelal uitgeschakeld hebben voor gebruikers om hun eigen infrastructuur te beschermen.

Azure Container Apps

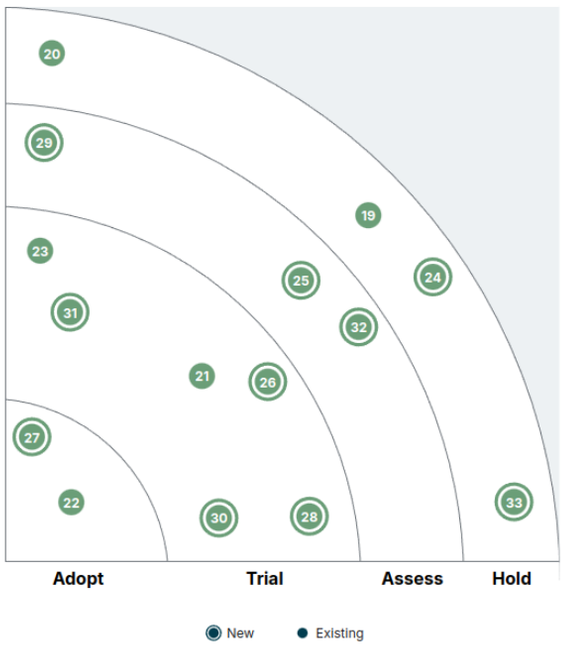
ASSESS

Azure Container Apps is een nieuwe service op Microsoft Azure die het inzetten en schalen van containerized applicaties in een serverless omgeving vereenvoudigt. Het is gepositioneerd tussen Azure App Service, dat vooral gericht is op eenvoudige webapplicaties, en Azure Kubernetes Service, wat een beheerd Kubernetes-platform is.

Container Apps is gebouwd op Kubernetes, KEDA en Dapr. Hierdoor maakt het event-driven applicatiearchitecturen mogelijk door ondersteuning van geavanceerde autoscaling (inclusief scale-to-zero en traffic splitting), ondersteuning voor langlopende processen en achtergrondtaken. Er is geen directe toegang tot de Kubernetes API's beschikbaar.

Wij denken dat Azure Container Apps een goed alternatief is voor teams die Azure App Service ontgroeid zijn, maar niet hun eigen Kubernetes-cluster willen of hoeven te beheren. Daarom plaatsen wij Azure Container Apps in *Assess*.

Tools



Hold

- 19. Docker 4 Desktop
- 20. Lombok
- 24. Diffblue Cover
- 33. JobRunr

Trial

- 21. Sigrid
- 23. Testkube
- 26. ChatGPT
- 28. Qodana
- 30. jlink
- 31. Maven Build Cache Extension

Adopt

- 22. k9s
- 27. Vector Databases

Assess

- 25. Error Prone
- 29. Thunderdome Dev
- 32. Konsist

k9s

ADOPT

k9s [https://k9scli.io/] is een product om Kubernetes-clusters te beheren via de commandline. Het geeft je de mogelijkheid om zowel je cluster te beheren als real-time inzicht te krijgen in de status van het cluster. Dit kan door middel van het inzien van logbestanden, het wijzigen van deployments en ook verbinding maken met de pods. k9s is een gratis alternatief voor Lens Desktop [https://k8slens.dev/desktop.html] dat inmiddels een commercieel product is geworden, waarvoor de overgangsperiode in januari 2023 is afgelopen. Gezien k9s dezelfde functionaliteit biedt als Lens Desktop, plaatsen wij het op *Adopt*.

Commit. Develop. Share.

Vector Databases

ADOPT

Vector databases zijn gespecialiseerde databases die efficiënt hoog-dimensionale vectoren opslaan, indexeren en ophalen. Deze vectoren worden gebruikt voor semantisch zoeken naar tekst, afbeeldingen en meer, waardoor toepassingen zoals ChatGPT en GitHub Copilot kunnen functioneren.

Bij JDriven zijn we van mening dat bedrijven zouden moeten overwegen om vector databases te gebruiken wanneer ze de mogelijkheden van semantisch zoeken verkennen of zich begeven in het AI-domein.

Hoewel er talloze leveranciers van vector databases zijn, zijn de meeste, op basis van onze ervaring, vergelijkbaar, zonder groot onderscheidend kenmerk. Daarom hebben we ervoor gekozen om elk van deze databases niet afzonderlijk op te nemen in onze Tech Radar. De meest populaire databases zijn Pinecone, Chroma, pgVector en Weaviate. Bovendien hebben bedrijven zoals Elasticsearch, Redis en Neo4j vectorfunctionaliteiten geïntegreerd in hun gereedschapskist.

Sigrid

TRIAL

Sigrid [https://www.softwareimprovementgroup.com/solutions/sigrid-software-assurance-platform/] is een code quality benchmarking tool, gericht op het verbeteren van de kwaliteit van softwarecode door middel van statische codeanalyse. Het bevat benchmarks op basis van best practices en patterns in code, opgebouwd vanuit onder andere opensourceprojecten. Het beoordeelt hoe gescande code hiertegen afsteekt. Sigrid geeft ontwikkelaars en ook management inzicht in de technische kwaliteit van de code. Het richt zich daarbij wat sterker op het management, maar levert waardevolle inzichten voor zowel de ontwikkelaars als de stakeholders. Analyse gebeurt na het aanmaken van een pull request in de repository. Het kent daarbij geen early warning systeem of een plug-in voor een IDE. Volgens Sigrid zal een plug-in ook niet worden ontwikkeld, er wordt aangeraden om Sigrid naast bijvoorbeeld SonarLint en SonarQube te gebruiken. Het wordt bij diverse klanten van JDriven ingezet waarbij we samen met de klant onderzoeken wat de voor- en nadelen zijn. Om deze reden voegen we Sigrid toe aan het *Trial* kwadrant van deze editie van onze Tech Radar.

Testkube

TRIAL

Testkube [https://testkube.io] is een framework om testen uit te voeren en te coördineren op Kubernetes. Testkube definieert testen als Kubernetes Custom Resource Definitions (CRD) en biedt ondersteuning voor tools als Maven en Gradle. Zo is het mogelijk dat integratietesten worden uitgevoerd als er bijvoorbeeld een nieuwe versie van een microservice wordt gedeployed op Kubernetes. Door middel van een dashboard, command-line tools en/of configuratiebestanden is het mogelijk om testen te beheren. Testkube is onderdeel van het Cloud Native Interactive Landscape (CNCF). Het is een product dat het uitvoeren van testen anders doet dan in de Continuous Integration (CI) pipeline en we plaatsen het daarom in *Trial* zodat ervaring kan worden opgedaan met deze andere manier van testen uitvoeren.

ChatGPT

TRIAL

Het bekende taalmodel dat in staat is om complexe gesprekken te voeren met gebruikers, is intussen meer dan een jaar oud. ChatGPT kan ontwikkelaars helpen bij het oplossen van problemen, verbeteren van code, detecteren van bugs, uitvoeren van statische codeanalyse en meer.

Echter kunnen we niet blindelings vertrouwen op ChatGPT. Er zijn een aantal redenen om zorgvuldig om te gaan met de resultaten. Allereerst zal de tool code met vertrouwen presenteren, zelfs als het fouten bevat, niet geoptimaliseerd is of constructies gebruikt waarvoor betere alternatieven beschikbaar zijn. De gratis versie heeft gegenereerde code opgeleverd met problemen zoals delen door nul, n+1 selectieproblemen en refactors die de kwaliteit van de code verminderden. De gegenereerde code door de betaalde versie is van hogere kwaliteit, maar nog steeds niet vlekkeloos.

Naast functionele problemen kan de tool een beveiligings- of privacyrisico vormen. Chatgeschiedenis wordt gelogd en aan het model gevoed. Daarom is het belangrijk om alleen code bloot te stellen die van generieke aard is en geen bedrijfsspecifieke details of gevoelige informatie bevat. Er is geen garantie dat gegevens veilig zijn.

Al met al is ChatGPT een nuttige en capabele tool, vooral voor het leren van een nieuwe taal, concept of framework. Gebruik het om aanwijzingen te krijgen voor het oplossen van problemen, maar vertrouw niet blindelings op de oplossingen zonder precies te weten wat de gegenereerde code doet. Gebruik het voor refactoring, maar houd er rekening mee dat het mogelijk geen passende patronen of benaderingen toepast.

Qodana

TRIAL

Qodana is een Static Code Analysis tool van JetBrains. Het wordt meegeleverd met de IntelliJ IDE, maar moet apart geactiveerd worden. Los van IntelliJ-integratie kent het veel CI/CD- en IDE-integraties en kan het o.a. Java, Kotlin, JavaScript en TypeScript codebases scannen. Het biedt een Qodana Cloud dashboard voor het in kaart brengen van de codekwaliteit over meerdere projecten.

Na een paar jaar als preview beschikbaar te zijn geweest, is Qodana officieel uitgekomen in 2023. Het kent een gratis community-model, maar voor JavaScript/TypeScript- en Spring-ondersteuning ben je aangewezen op een betaald model. Dat betaalde model is echter ook wat het interessant maakt t.o.v. SonarQube, want in Qodana betaal je niet voor de hoeveelheid lines of code (LoC), maar voor het aantal actieve committers. Bovendien zit de CI/CD-integratie bij Qodana in de gratis variant. Dat betekent dat, afhankelijk van de aard van je project en hoe het onderhouden wordt, Qodana aanzienlijk goedkoper zou kunnen uitvallen. Het is de moeite waard om die vergelijking te maken en de tool te proberen.

jlink

TRIAL

jlink is een tool om een op maat gemaakte Java Runtime Environment (JRE) te bouwen. Deze tool doet dit door het strippen van componenten uit de JRE die de doelapplicatie niet nodig heeft. Te denken valt onder andere aan overbodige modules, documentatie en losse tools voor de Java-omgeving. Een op maat gemaakte JRE biedt twee voordelen: het leidt tot bruikbare artefacten (zoals Docker images) van een kleiner formaat, en vermindert de hoeveelheid potentiële ingangen voor aanvallen. Een kleiner formaat images leidt tot kortere uitroltijden, met name binnen cloud omgevingen, en mogelijk ook tot lagere kosten bij het beheren en hosten van images en omgevingen. Door onnodige modules niet toe te voegen, kunnen kwaadwillende partijen geen gebruik maken van mogelijke veiligheidslekken in deze modules. Dergelijke lekken kunnen in catastrofale gevallen leiden tot privilege escalation en niet-geautoriseerde datatoegang.

We hebben jlink onder *Trial* ingedeeld, omdat wij duidelijke voordelen zien bij het gebruik hiervan. Zoals met praktisch alle tooling zijn er echter ook een aantal keerzijden. De tool om te bepalen welke modules nodig zijn voor een gegeven applicatie, jdeps, geeft niet altijd een eenduidige rapportage van welke modules nodig zijn. Vaak moet met de hand nog een aantal modules aan de lijst worden toegevoegd. Een serieuzer probleem is dat afwezigheid van benodigde modules kan leiden tot cryptische foutmeldingen en veranderd gedrag van de applicatie. Dit zorgt ervoor, dat een op maat gemaakte JRE een klein risico kan vormen, wat het voor sommige toepassingen ongeschikt maakt.

Maven Build Cache Extension

TRIAL

Maven Build Cache [https://maven.apache.org/extensions/maven-build-cache-extension/] is een extensie voor Maven (beschikbaar vanaf versie 3.9) waarmee je je Maven builds efficiënter kunt maken door slim gebruik te maken van caching (vergelijkbaar met de Gradle Build Cache).

De build cache kan op verschillende momenten in het build-proces de outputs, zoals gegenereerde code of gecompileerde classes, in een cache plaatsen. Als op een later moment de build nogmaals wordt uitgevoerd met (deels) ongewijzigde inputs, kunnen de eerder gecachete outputs worden hergebruikt. Hierdoor kunnen delen van de build, zoals het compileren van code en het uitvoeren van tests, worden overgeslagen, waardoor de totale buildtijd afneemt. Bijkomend voordeel is dat build caches ook gedeeld kunnen worden door gebruik te maken van een remote cache.

Door de Maven Build Cache extension te introduceren is het in veel gevallen mogelijk om de buildtijd van je Maven-projecten significant te verminderen. Doordat dit een vrij nieuwe feature van Maven is, bestaat de mogelijkheid dat nog niet elke third-party Maven-plugin hier goed mee samenwerkt. Daarnaast is elk Maven-project anders, dus zal het enig uitzoekwerk kosten om de build cache op de juiste manier te laten werken. Een verkeerd geconfigureerde build cache kan leiden tot instabiele, niet-reproduceerbare builds. Om deze redenen plaatsen wij het in *Trial*.

Error Prone

ASSESS

Error Prone [https://errorprone.info/index] is een Java-library die een reeks checks en statische analysemogelijkheden biedt tijdens de build-fase.

Door OutOfRange en NullPointerExceptions te identificeren, evenals ineffectieve tests, maakt Error Prone directe feedback op diverse problemen mogelijk. Error Prone biedt ontwikkelaars ook de mogelijkheid om de checks aan te passen aan specifieke codingstijlen, projectvereisten en best practices uit het vakgebied.

Door Error Prone te integreren in de ontwikkelworkflow hopen we de codekwaliteit te verbeteren, het risico op defecten te verminderen en de productiviteit van ontwikkelaars te verhogen. De proactieve aanpak van Error Prone bij het identificeren van potentiële problemen is de reden waarom we het hebben geplaatst in het *Assess* gedeelte van deze editie van onze Tech Radar.

Thunderdome Dev

ASSESS

Thunderdome.dev [https://thunderdome.dev/] is een opensource-samenwerkingstool. Een onderscheidende feature is als planning poker-tool. Het kan ook worden gebruikt bij andere processen, zoals retrospectives, team checkins, en feature/story mapping. Het proces van planning poker werkt goed in de context van persoonlijke aanwezigheid van teamleden. In een hybride werkomgeving mist die context, waardoor het ontwikkelteam het proces op een andere manier moet vormgeven. Thunderdome.dev zorgt bij planning poker voor een basisstructuur waarmee dit makkelijker wordt.

Aan de ene kant is het een erg toegankelijke tool, aan de andere kant is de toepassing sterk afhankelijk van de werkwijze en behoeftes van het ontwikkelteam. We plaatsen Thunderdome.dev daarom in *Assess*. Houd bij een assessment rekening met alle features van deze tool. Met name voor de mogelijkheid als aanvulling op, of vervanging van een gevestigde tool als Jira.

Konsist

ASSESS

Konsist [https://docs.konsist.lemonappdev.com/getting-started/readme] is een *static code-analyzer* library voor Kotlin die helpt codeconventies te garanderen en gedefinieerde architecturele grenzen kan controleren. Controles worden uitgevoerd in unittests en kunnen als zodanig worden uitgevoerd tijdens pull requests om de architectuur van een project te beschermen. Konsist ondersteunt Kotlin-projecten, waaronder Android, Spring en Kotlin Multiplatform, terwijl ArchUnit, een vergelijkbare tool, alleen (JVM) bytecode-analyse ondersteunt.

Wij bij JDriven geloven in codekwaliteit en een helder gedefinieerde software-architectuur. Konsist is een zeer nuttig hulpmiddel in onze gereedschapskist om codeconventies en architecturale grenzen te formaliseren en te verifiëren. Omdat Konsist nog in de kinderschoenen staat, houden we de ontwikkeling ervan in de gaten voordat we de adoptie ervan volledig aanraden.

Docker 4 Desktop

HOLD

Docker for desktop [https://www.docker.com/products/docker-desktop/] is een product om makkelijk met containers te werken op desktop computers of laptops en wordt vooral gebruikt op Microsoft Windows- en macOS-systemen. De licentievoorwaarden voor Docker for desktop zijn in 2022 gewijzigd, waardoor bedrijven mogelijk geld moeten betalen om Docker for desktop te gebruiken. In 2023 is de overgangsperiode van één jaar verstreken en gaan de licentievoorwaarden nu echt gelden. Daarom plaatsen we Docker for desktop nu op *Hold*. Dit betekent dat voor de inzet van Docker for desktop eerst goed moet worden nagedacht of andere tools misschien een beter alternatief zijn.

Lombok

HOLD

Lombok [https://projectlombok.org/] is een codegenerator om boilerplate-code zoals getters, setters en constructors in Java code te laten genereren. De code wordt hoofdzakelijk vanuit annotaties gegenereerd. De wildgroei aan annotaties, de benodigde kennis van de Lombok-implementatie en de opkomst van nieuwere Java language features, zoals records, maakt dat JDriven terughoudend is bij het inzetten van Lombok voor nieuwe projecten.

Er zijn veel Lombok-annotaties, waarvan niet elke developer zich bewust is van de consequenties. Lombok genereert code waarvan een developer moet weten dat deze er is, maar het feitelijk niet kan zien omdat het in de codegeneratie-stap onzichtbaar aan de bytecode wordt toegevoegd. Als developer zijn we tien keer zo vaak code aan het lezen als aan het schrijven. Dat maakt juist dat we het lezen moeten vergemakkelijken om het zo makkelijker schrijfbaar te maken.

Bedenk bij het gebruik van bijvoorbeeld @Getter, @Setter, @NoArgsConstructor en @Builder of het daadwerkelijk bijdraagt aan beter leesbare code, of dat het juist belemmerend is bij het lezen van de code. Het gebruik van meerdere Lombok-annotaties kan overlappende functionaliteit hebben, waardoor de code overbodige annotaties bevat die de leesbaarheid niet verhogen. Soms is het volledig uitschrijven van de Java-code in een POJO (Plain Old Java Object) zonder deze Lombok-annotaties beter om de leesbaarheid te verhogen.

Wij als JDriven adviseren om simpele voorbeelden die vaak met Lombok zijn uitgeschreven niet klakkeloos in projecten als voorbeeld te gebruiken, maar goed na te denken over de toepassing van Lombok-annotaties. Om deze reden plaatsten wij Lombok in het *Hold* kwadrant van deze editie van onze Tech Radar.

JobRunr

HOLD

JobRunr [https://www.jobrunr.io/] is een Java-library voor het draaien en aansturen van achtergrondprocessen. Met JobRunr kun je jobs plannen, batches uitvoeren, of workflows maken met behulp van Java Lambda-functies, met ingebouwde functies voor het opnieuw proberen van jobs die gefaald zijn. JobRunr komt met een webinterface waarmee je de jobs in de gaten kunt houden, jobs kunt aansturen, en het kan je inzicht geven wanneer er fouten zijn opgetreden. Alhoewel de gratis versie erg interessant is en het overwegen waard om te gebruiken, hebben we JobRunr op *Hold* geplaatst vanwege de pro versie. JobRunr Pro is namelijk niet opensource en er zijn maar een aantal mensen die eraan werken, wat voor wat risico zorgt dat sommige bedrijven liever vermijden. Dus ons advies is om hier zorgvuldig een keuze te maken.

Diffblue Cover

HOLD

Diffblue Cover [https://www.diffblue.com/products/], een geautomatiseerde tool voor het genereren van unittesten voor Java code, toont sterke punten in het begrijpen van de te testen code en het creëren van geschikte mocks. Echter, het vermogen om complexe mockdata te genereren is beperkt, en de benadering van het genereren van testen komt mogelijk niet overeen met bestaande projectstijlen.

Hoewel Diffblue Cover nuttig kan zijn voor het genereren van boilerplatecode en het opzetten van eerste testen, zal het handmatige testen niet volledig vervangen. Tevens is de compatibiliteit beperkt tot specifieke versies van Spring, Java en JUnit, en het heeft moeite met methoden zonder parameters of klassen zonder constructors. Verder kan Diffblue Cover niet worden uitgevoerd op een heel project of zelfs op pakketten, wat de toepasbaarheid ervan beperkt.

Daarom wordt Diffblue Cover in het kwadrant *Hold* geplaatst van deze editie van onze Tech Radar.

Commit.
Develop.
Share.