JDriven Tech Radar Voorjaar 2025

Onze kijk op recente ontwikkelingen in het veld

11e editie

Commit to know.

Develop to grow.

Share to show.

jdriven

Commit. Develop. Share.

Voorwoord

Met veel trots presenteren we de 11e editie van de JDriven Tech Radar. Het is de eerste editie na de jubileumeditie. Daarom ging het samenstellen weer als vanouds, en prijkt er op de kaft deze keer geen kunstwerk gemaakt door alle collega's. Desalniettemin is de manier waarop we de Tech Radar samenstellen iedere keer weer bijzonder, en iets om eens bij stil te staan.

De JDriven-collega's werken bij diverse klanten, in diverse sectoren, met diverse tools, technieken, talen, frameworks en platforms. We wonen bovendien conferenties bij als sprekers, bezoekers en organisatoren, en krijgen zo aardig wat mee van wat er speelt in ons vakgebied. We spreken elkaar daarnaast op kantoor, via Slack en op JDriven evenementen, en daar delen we wat we zien, doen en vinden. Regelmatig wijdt een collega er een blog, workshop of hackathon aan.

Tweemaal per jaar komen we bij elkaar om het net op te halen, op een rijtje te zetten wat we het afgelopen half jaar hebben gesignaleerd en wat we daarvan willen delen met onze vakgenoten en onze klanten. We kijken naar wat we eerder noemenswaardig vonden, en iedere JDriven collega kan nieuwe ideeën aandragen om samen te bespreken. Dat kan een nieuw framework zijn dat niet de nukken heeft van een gevestigd alternatief, een oude methodologie die goed blijkt uit te pakken in combinatie met een nieuwe technologie, of een tool die het leven van een developer makkelijker maakt. Hierbij blijven we scherp op relevantie, actualiteit, granulariteit en vanzelfsprekendheid. Een nieuwe versie van Java of Kotlin zal niet op de radar verschijnen. En liever noemen we specifieke producten dan algemene concepten die door die producten worden toegepast, tenzij dat concept zelf hetgeen is dat we willen analyseren en bediscussiëren.

Het interessante van zulke Tech Radar sessies is dat geen van de aanwezigen een expert is op alle genoemde onderwerpen, maar dat we elkaar kunnen meenemen in de argumentatie om het op de radar te plaatsen. De onderwerpen, die we uiteindelijk kiezen als "blips" voor op de radar, schrijven we samen uit. Daarbij letten we altijd op een duidelijk onderscheid tussen de objectieve beschrijving en de mening van JDriven. Dat laatste is een risico: we kunnen ernaast zitten, iets afraden wat iemand wellicht met veel succes inzet, of juist iets aanprijzen dat uiteindelijk flopt. Maar dat is inherent aan ons vakgebied, dat altijd in beweging is. En wat we in afgelopen jaren gelukkig hebben gemerkt, is dat het altijd goed is om stof tot nadenken te bieden, de dialoog aan te gaan en tot nieuwe inzichten te komen.

Bedankt aan eenieder die dit mogelijk heeft gemaakt. Laten we samen verder bouwen aan een actieve en groeiende community voor software engineering!

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 [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.

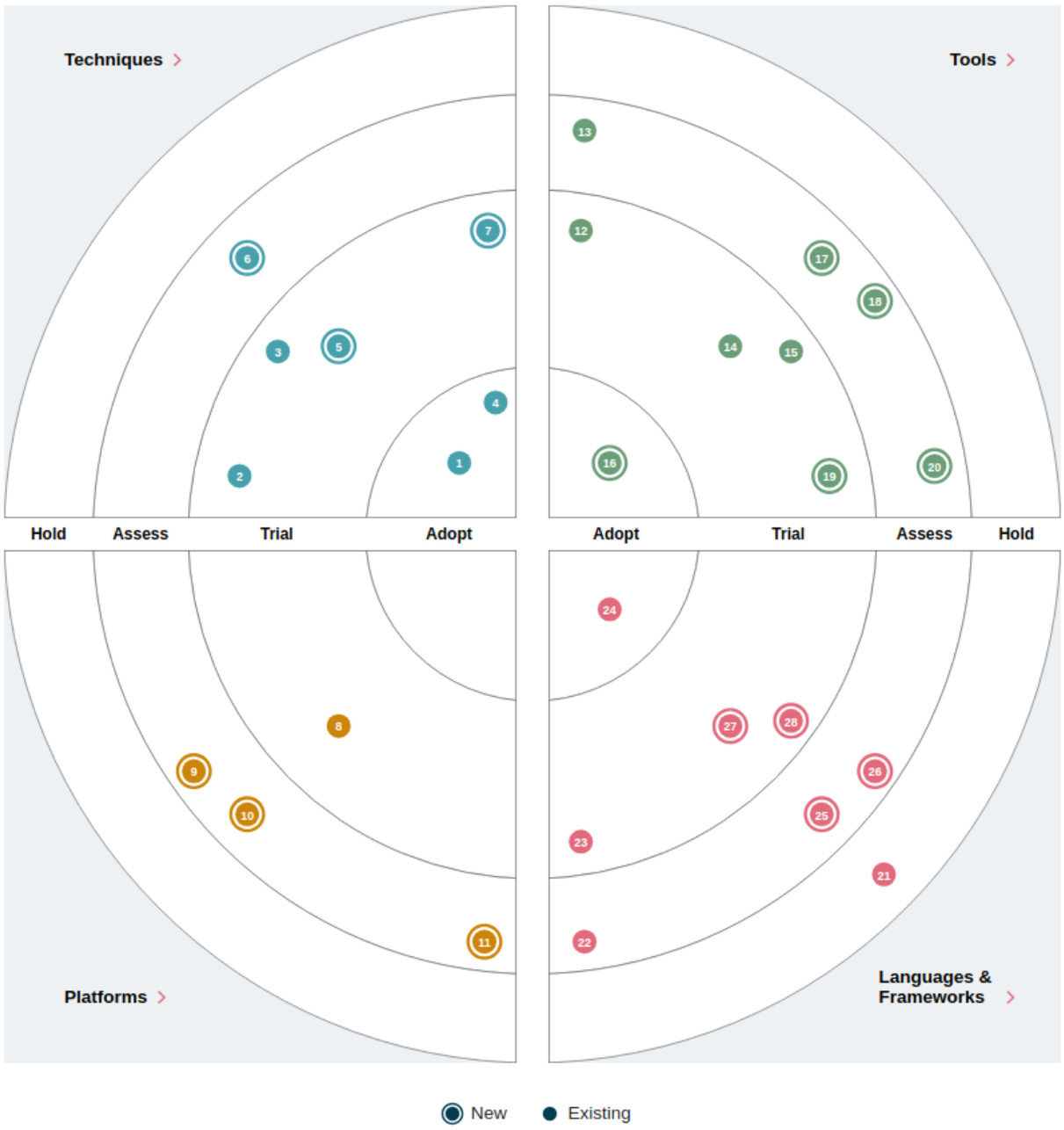
De kwadranten verdelen de verschillende onderwerpen in categorieën.

- **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.

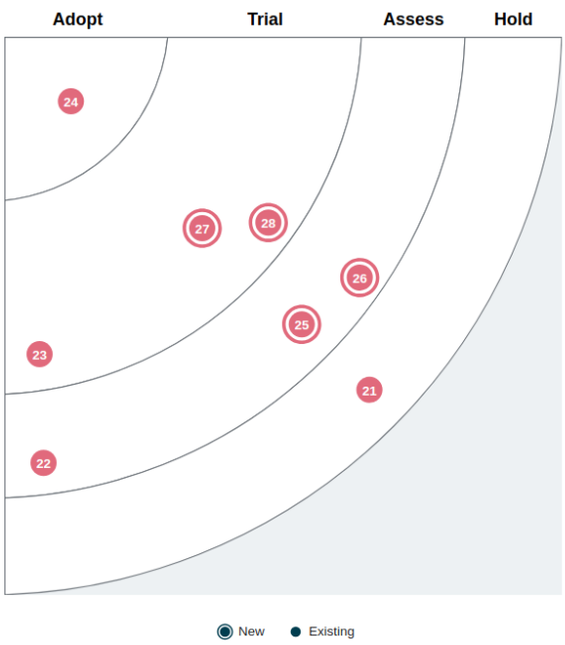


Commit. Develop. Share.

Tools	Talen & frameworks
ADOPT IntelliJ HTTP client **TRIAL** Bruno Devoxx Genie Nushell warp.dev **ASSESS** CRaC Cursor IDE DeepSeek JetBrains AI **HOLD** -	**ADOPT** Spring Modulith **TRIAL** Fabrikt JSpecify TimeFold AI **ASSESS** Declarative Gradle Helidon Model Context Protocol (MCP) **HOLD** Micronaut SourceGen
Platforms	**Technieken**
ADOPT - **TRIAL** AsyncAPI **ASSESS** AWS European Sovereign Cloud EU Cloud alternatives Proxmox **HOLD** -	**ADOPT** 15-factor app GitOps **TRIAL** Embrace Shadow IT Event Modeling Passkeys Shape Up **ASSESS** AI Agents **HOLD** -

Commit. Develop. Share.

Talen & frameworks



Hold

21. Micronaut SourceGen

Assess

22. Helidon
25. Declarative Gradle
26. Model Context Protocol (MCP)

Trial

23. TimeFold AI
27. Fabrikt
28. JSpecify

Adopt

24. Spring Modulith

Spring Modulith

ADOPT

Domain Driven Design heeft developers doen inzien dat softwarearchitectuur kan en moet helpen met het faciliteren van de evolutie van softwaresystemen bij wijzigende inzichten in business requirements. Microservices kunnen daarbij ingezet worden om functionele units te isoleren, maar dat introduceert uitdagingen op het gebied van service-orkestratie en extra benodigde infrastructuur voor onderlinge communicatie. Dit heeft developers doen heroverwegen of diezelfde modulariteit is aan te brengen en af te dwingen in monolithische applicaties.

Spring biedt als framework technische stereotypen zoals `@Controller`, `@Repository`, etc. zonder een mening op te leggen over hoe componenten die er gebruik van maken van elkaar afhankelijk zijn. Met Spring Modulith [https://spring.io/projects/spring-modulith] beoogt het juist wel een modulaire structuur aan te moedigen in een Spring Boot-applicatie. In de praktijk lijkt dit te betekenen dat een goed opgezette Spring Boot & Modulith-applicatie kandidaten voor dependency injection aan banden legt op basis van packagestructuur-conventies. Daarbij biedt het wel weer manieren om moduledetectie aan te passen t.o.v. die conventies.

Spring Modulith biedt een aantal annotaties om aan te duiden welke packages onderdeel zijn van een gekozen architectuurpatroon. Naast de annotaties biedt het ook een aantal ArchUnit testen, om vervolgens de gekozen architectuur af te dwingen. Standaard ondersteunde architectuurpatronen zijn: Layer Architecture, Onion Architecture en Hexagonal Architecture. Daarnaast ondersteunt het de generatie van verschillende soorten documentatiestandaarden. Spring Modulith biedt een nieuwe manier voor het integratietesten van individuele modules, die naar verwachting een stuk sneller zal uitpakken dan het gebruik van `@SpringBootTest`, omdat het alleen de modules instantieert die voor de test relevant zijn. Tevens stuurt Spring Modulith aan op het gebruik van events voor communicatie tussen de modules.

Recent heeft de officiële release van Spring Modulith plaatsgevonden, en wij zien Spring Modulith als een goed framework om modulariteit aan te brengen in monolithische applicaties.

Fabrikt

TRIAL

OpenAPI, voorheen bekend als Swagger, is een veelgebruikte interfacespecificatie voor het definiëren en documenteren van API's. De meest bekende tool die gebruikt wordt om op basis van een OpenAPI-specificatie code te genereren is waarschijnlijk OpenAPITools' openapi-generator. Fabrikt is een alternatief dat zich puur richt op het genereren van Kotlin-code op basis van de OpenAPI v3-specificatie. Het biedt ondersteuning voor zowel Spring, Micronaut als Ktor, en biedt bovendien een Gradle plugin.

In tegenstelling tot openapi-generator kan Fabrikt goed overweg met complexere structuren die OpenAPI v3 kent, met *anyOf* als bekendste voorbeeld. Fabrikt weet bovendien, mede dankzij de focus op OpenAPI v3 en Kotlin, openstaande issues en merge requests goed bij te benen; iets waar openapi-generator mee worstelt. Dit zijn redenen waarom wij aanraden om Fabrikt te gebruiken voor op OpenAPI v3 gebaseerde Kotlin code. Omdat Fabrikt complexere specificaties aankan en andere code kan genereren dan openapi-generator, raden we aan klein te beginnen en te evalueren wat de impact is van een migratie.

JSpecify

TRIAL

JSpecify [https://jspecify.dev/] is een annotation framework dat de afhandeling van nullability in Java verbetert door expliciete type-informatie te bieden, geïnspireerd door het nullability-systeem van Kotlin. Het helpt null pointer excepties te voorkomen door nullability-contracten op compile-tijd te standaardiseren. Het project is al meer dan vijf jaar in ontwikkeling en is ontworpen door grote partijen zoals Google, JetBrains, Meta, Microsoft, Oracle en Uber. Met de release van versie 1.0 is de specificatie gestabiliseerd, wat een solide basis biedt voor adoptie. IntelliJ IDEA, Project Lombok, NullAway en het Checker Framework bieden momenteel al ondersteuning voor JSpecify.

Migration guides zijn beschikbaar om over te stappen van JSR-305 of Checker Framework annotaties naar de JSpecify varianten. Nu Spring Framework 7 JSpecify omarmt en bredere ondersteuning in tooling in opkomst is, wordt het een steeds logischere keuze om null safety in Java te verbeteren. Hoewel de adoptie binnen het ecosysteem nog gaande is, is JSpecify stabiel en klaar voor productiegebruik. Daarom plaatsen we het in de Trial-ring.

TimeFold AI

TRIAL

Timefold AI [https://timefold.ai/] is een open-source constraint solver beschikbaar voor Java, Python en Kotlin. Het is ontworpen om complexe planningsproblemen zoals roosteren, routering en resource-allocatie te optimaliseren. Met behulp van constraint satisfaction programming kan een score aan een oplossing worden toegewezen, en kan het probleem worden geoptimaliseerd door een van de ingebouwde zoekalgoritmes te gebruiken. Timefold AI is een fork van OptaPlanner, een project dat niet langer actief wordt ontwikkeld, met verbeteringen die zich richten op prestaties, waaronder een enterprise-versie die multithreading biedt voor verhoogde efficiëntie.

Bij JDriven zien we een toenemende vraag naar constraint solvers bij onze klanten, en we beschouwen Timefold AI als een veelbelovend alternatief voor maatwerkoplossingen, vooral bij problemen van beperkte omvang. Daarom plaatsen we Timefold AI in de *Trial* ring en moedigen we teams aan om het uit te proberen.

Declarative Gradle

ASSESS

Bij gebruik van Gradle zijn er veel verschillende manieren om je build-scripts in te richten. Dit kan voor veel verwarring zorgen, vooral als je met meerdere projecten werkt die verschillende manieren van werken hebben. Bij Gradle zijn ze zich hiervan ook bewust en daarom hebben ze eind 2023 aangekondigd begonnen te zijn met een experiment om Gradle declaratief te maken. Dit project heet Declarative Gradle [https://declarative.gradle.org] en is nog in een experimentele fase. Het doel van dit project is om een vaste structuur aan te bieden voor Gradle build-scripts, zodat deze makkelijker te lezen en te onderhouden zijn. Dit moet ook de herkenbaarheid tussen projecten verbeteren.

We plaatsen deze nieuwe standaard in de *Assess*-ring, omdat we het de moeite waard vinden ontwikkelingen op dit vlak te volgen en te bekijken welke gevolgen dit heeft voor Gradle build-scripts.

Helidon

ASSESS

Helidon [https://helidon.io/] is een open-source microservices framework, vergelijkbaar met Spring Boot en Quarkus, en biedt een lichtgewicht alternatief zoals Ktor en Vert.x. Het ondersteunt zowel reactieve als imperatieve programmeerstijlen en is beschikbaar in twee versies: Helidon SE (Standalone Engine), een lichtgewicht, reactief framework, en Helidon MP (MicroProfile), dat volledige compatibiliteit biedt met Jakarta EE MicroProfile. In versie 4.0 heeft Helidon belangrijke verbeteringen doorgevoerd, waaronder native ondersteuning voor virtuele threads, verbeterde gRPC-integratie, en performance optimalisaties die gericht zijn op het verminderen van het resourceverbruik.

Helidon wordt ondersteund door Oracle, wat zowel een kracht als een mogelijke beperking kan zijn. Aan de ene kant zorgt Oracle's ondersteuning voor sterke enterprise-support en langdurige stabiliteit. Aan de andere kant kunnen sommige teams zich zorgen maken over vendor lock-in of afhankelijkheid van een grote corporate partij voor innovatie.

Gezien de ontwikkelingen in versie 4.0 adviseren wij teams, die op zoek zijn naar alternatieve op Java gebaseerde microservice frameworks, om Helidon te overwegen. Daarom plaatsen we Helidon in de *Assess* ring.

Model Context Protocol (MCP)

ASSESS

Model Context Protocol (MCP) [https://modelcontextprotocol.io/introduction] is een open standaard die applicaties in staat stelt om op een consistente manier context te bieden aan large language models (LLM's). MCP heeft integraties met databases (PostgreSQL, SQLite), platforms, bestandssystemen en vele andere systemen. Door de afhankelijkheid van propriëtaire oplossingen te verminderen, verbetert MCP de interoperabiliteit met meerdere LLM-modellen. Het heeft ook de potentie om niet-technische gebruikers te helpen bij het opvragen en begrijpen van hun data door AI-modellen dynamisch toegang te geven tot relevante context.

De adoptie hangt echter af van industriebrede ondersteuning, en er blijven uitdagingen bestaan bij het standaardiseren van contextoverdracht over diverse systemen, terwijl ook beveiligings- en prestatievraagstukken moeten worden aangepakt. We plaatsen MCP in *Assess*, omdat het veel potentie toont, maar dit zich nog wel in de praktijk moet bewijzen.

Micronaut SourceGen

HOLD

Micronaut SourceGen [https://micronaut-projects.github.io/micronaut-sourcegen/latest/guide/#introduction] is een library waarmee je broncode kunt genereren voor Kotlin en Java. Het is een fork van JavaPoet [https://github.com/square/javapoet] en is bijgewerkt om nieuwere Java-constructies zoals records te gebruiken. Micronaut SourceGen biedt een gebruiksvriendelijke API om de structuur van de te genereren broncode te definiëren. De library gebruikt een annotatieverwerker om de broncode te genereren tijdens de Java- of Kotlin-compilatie.

Er zijn annotaties om builders en withers voor records te genereren in de library. Dit zou de builder-annotaties van Lombok in een project kunnen vervangen.

Het beste kenmerk van de bibliotheek is dat het integreert met de standaard annotation processor van de compiler. Er worden alleen nieuwe bronbestanden gemaakt en het zal niet knoeien met bestaande classes. De API om de broncode te maken is eenvoudig te gebruiken. De library is in *Assess* geplaatst om uit te proberen en mee te experimenteren.

Technieken

Adopt

1. GitOps
4. 15-factor app

Trial

2. Shape Up
3. Event Modeling
5. Passkeys
7. Embrace Shadow IT

Assess

6. AI Agents

15-factor app

ADOPT

In 2011 publiceerde Heroku een set van principes voor het ontwikkelen van SaaS-applicaties (Software as a Service). Deze set aan principes staat bekend als de 12-factor app [https://12factor.net]. Naast een aantal principes voor het ontwikkelen van de services en het managen van de dependencies lag de focus vooral op het cloud-agnostisch ontwikkelen van de applicatie zodat deze eenvoudig in diverse omgevingen kan draaien.

De 15-factor app [https://developer.ibm.com/articles/15-factor-applications/] voegt 3 principes toe die de kwaliteit en onderhoudbaarheid van cloud native-applicaties kan verbeteren.

- **API-first:** Cloud native-applicaties draaien in een ecosysteem met andere services, managed services en faciliteiten die cloudomgevingen aanbieden. Door de focus op goede API's te leggen kunnen uitdagingen in integratie worden voorkomen.
- **Telemetry:** In de originele 12-factor app was "logging" al een factor. Logging geeft inzicht in de interne staat van de applicatie. Telemetry is toegevoegd om de aandacht ook te leggen op het resourceverbruik, scaling, integratie met andere applicaties en de integratie met de cloud.
- **Authentication en Authorization:** Cloud native-applicaties kunnen makkelijk gedistribueerd worden over diverse datacenters of zelfs diverse clouds. Het bereik van dergelijke apps is enorm en zowel het aantal als de verschillende typen consumers vergroten de complexiteit op het gebied van security. Authentication en Authorization legt de focus op security bij het ontwikkelen van cloud native-applicaties.

We hebben de 15-factor app in *Trial* gezet. De 12-factor app geeft al een set aan goede handvatten voor het ontwikkelen van cloud native-applicaties, maar de 15-factor app geeft 3 extra factoren die niet mogen ontbreken in moderne cloud native-applicaties.

GitOps

ADOPT

GitOps [https://about.gitlab.com/topics/gitops/] is een techniek om je DevOps-automation te versterken door best practices binnen software development, zoals versiebeheer, samenwerking, CI/CD, en compliance toe te passen.
De techniek bestaat simpel gezegd uit het gebruik van Git en Infrastructure as Code, in samenwerking met een CI/CD-pipeline voor alles.
Dat geldt dus ook voor **elke** configuratieaanpassing in een draaiend systeem.

Met Git [https://git-scm.com/] wordt elke wijziging bijgehouden in de Git-historie waardoor er een audit trail ontstaat, en er een centraal punt is waar men kan samenwerken door middel van merge requests om wijzigingen door te voeren.
Infrastructure as Code [https://en.wikipedia.org/wiki/Infrastructure_as_code] zorgt dat alle infrastructuurconfiguratie in code beschreven staat, die centraal bijgehouden wordt in een Git-repository
Door een CI/CD-pipeline op de Git-repository aan te sluiten is het mogelijk om de infrastructuur automatisch te laten deployen zonder menselijke interventie.

We zien dat veel bedrijven het al vanzelfsprekend vinden om GitOps te gebruiken. Wij adviseren dan ook om GitOps te omarmen en hebben het daarom in *Adopt* gezet voor onze technologieradar.

Embrace Shadow IT

TRIAL

Shadow IT is een fenomeen waarbij software developers binnen een organisatie IT-faciliteiten optuigen buiten officieel toezicht en regulering door de verantwoordelijke IT-supportafdeling. Dit gebeurt typisch wanneer IT-support er niet voldoende in slaagt om de behoeften van de developers te faciliteren voor het leveren van software. Dat kan liggen in het ongeoorloofd in gebruik nemen van network, storage & compute resources voor te draaien applicaties of voor CI/CD-tooling voor het uitrollen daarvan. Soms komt het tot uiting in zelfgebouwde tooling die buiten audit trails valt, of zelfs in onder de radar in gebruik genomen third party softwarepakketten die via het publieke internet benaderd worden. Shadow IT kan leiden tot een versnipperd softwarelandschap met blinde vlekken. Onderhoud, security en compliance van cruciale onderdelen kan onderbelicht zijn, en management beschouwt shadow IT daarom doorgaans terecht als een risico.

We zien een trend waar platformteams, geboren uit eerdere DevOps-inzichten, gaandeweg weer ondergebracht worden in eigen afdelingen, en weer gestuurd worden op het stabiel en voorspelbaar houden van infrastructuur en CI/CD-tooling, en een overeenkomstige toename van shadow IT door software developers. JDriven moedigt organisaties aan om shadow IT - waar mogelijk, vandaar *Trial* op de Tech Radar - niet krampachtig te beteugelen, en juist de verbinding te zoeken. Developers moeten inzicht hebben in de redenen waarom IT-support grenzen stelt aan wat het faciliteert, en IT-support moet de behoeftes van developers kennen voor het groeiende landschap en de eisen aan CI/CD-tooling om software regelmatig te verbeteren. Zoals DevOps het antwoord was op de wrijving ontstaan in een softwarelandschap waarin developers agile software moeten leveren maar wat operations zo stabiel en controleerbaar mogelijk wil houden, zo is shadow IT aan te grijpen voor gecontroleerde innovatie en blijvende effectiviteit van je IT-support. Het omarmen van shadow IT moet dan ook onderdeel zijn van de DevOps-mentaliteit.

Event Modeling

TRIAL

Event Modeling [https://eventmodeling.org/] is een methode waarbij het beschrijven van gebeurtenissen in een businessproces centraal staat. Deze gebeurtenissen leiden weer tot het veranderen van informatie.

Het centraal stellen van een businessproces en deze op een eenduidige manier beschrijven, helpt zowel de business als de architecten en ontwikkelaars om een probleem en oplossing te doorgronden. Bij Event Modeling wordt ook de gebruikerservaring meegenomen. Zo ontstaat een compleet beeld van interacties van gebruikers met het systeem en hoe de informatie door het systeem vloeit.

Bij JDriven zien we dat Event Modeling een goede aanvulling op of vervanging van Big Picture Event Storming kan zijn, om vanuit een gedeeld begrip van het businessdomein sneller tot concrete implementaties ervan te komen.

Passkeys

TRIAL

Hoewel we niet volledig overtuigd zijn dat passkeys inmiddels een algemeen geaccepteerd begrip zijn, hebben we besloten om deze techniek in *Trial* te plaatsen. De passkey-techniek is wat ons betreft klaar om als alternatief aangeboden te worden naast traditionele authenticatietechnieken, zoals wachtwoorden. Het idee is dat passkeys veilig zijn door het "iets dat je hebt" principe van de "authentication factors" (https://en.wikipedia.org/wiki/Authentication#Authentication_factors), omdat de private key altijd op het apparaat van de gebruiker blijft. Ten opzichte van wachtwoorden voorkomt dit een aantal kwetsbaarheden, die vanwege de menselijke factor vaak kwetsbaar zijn door bijvoorbeeld lage wachtwoordsterkte, hergebruik, datalekken, social engineering en onvoorzichtigheid. Door adoptie op apparaten van grote IT-bedrijven zoals Apple en Google is het gebruiksgemak voor de gemiddelde gebruiker inmiddels zo hoog dat passkeys een realistische optie voor primaire beveiliging zijn geworden.

Mocht je als ontwikkelaar het gebruik van passkeys overwegen, heeft Hanko.io een handige website die het nut van passkeys demonstreert: https://www.passkeys.io/.

Denk uiteraard goed na over het proces van accountherstel als je passkeys wilt adopteren. Geef gebruikers voldoende gelegenheid om met vertrouwde processen hun authenticatie aan te passen.

Shape Up

TRIAL

Shape Up [https://basecamp.com/shapeup] beschrijft een werkwijze waarbij een bepaalde oplossing dusdanig "bijgeschaafd" is, dat het in een iteratie past.

Bij veel projecten staat iteratief werken centraal, waarbij wordt gekeken hoeveel werk in een iteratie past, bijvoorbeeld door middel van het inschatten van Story Points. Shape Up keert dit om, door de oplossing constant bij te schaven, zodat het in een iteratie past. Centraal staat dat de oplossing voor een probleem weinig details bevat, compleet is en duidelijke kaders heeft. Weinig details betekent dat bijvoorbeeld een gebruikerservaring uit een grove schets bestaat. Compleet betekent dat valkuilen geïdentificeerd en weggenomen zijn. Duidelijk kaders betekent dat het duidelijk is, wat het beginpunt en het eindpunt van de oplossing is.

Blijkt dat de oplossing niet in een iteratie past, dan kan moet de oplossing dusdanig bijgeschaafd worden, zodat het nog steeds de beoogde waarde oplevert én in de iteratie past.

AI Agents

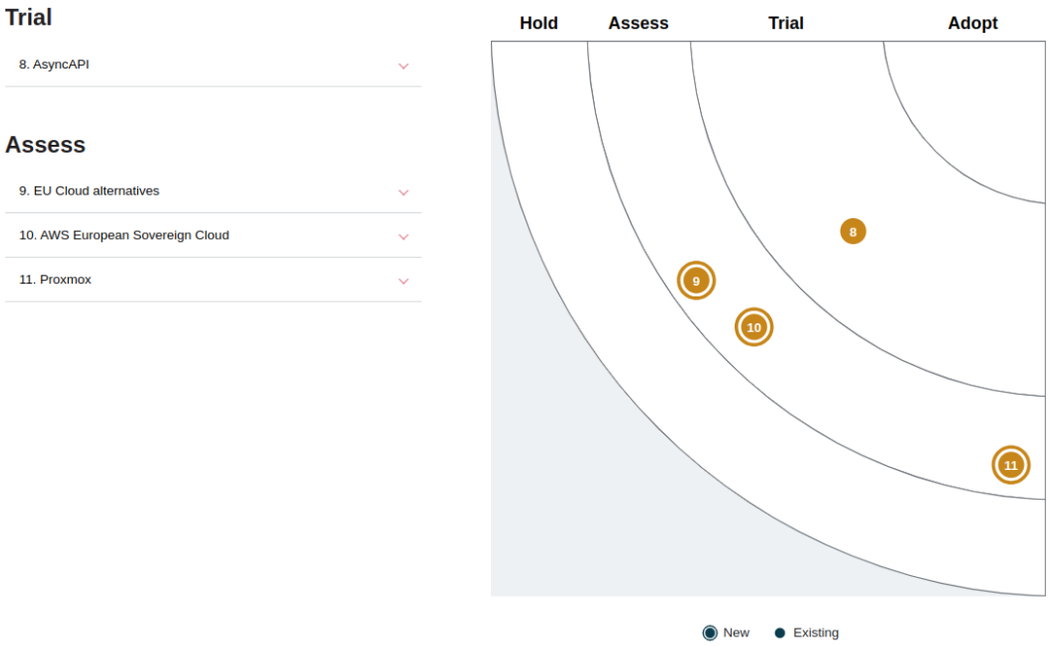
ASSESS

LLM's maken interactie tussen mens en machine mogelijk op een niveau dat we voorheen alleen zagen in films. De mogelijkheid om een vraag te stellen of een taak te definiëren via "natuurlijke" taal is misschien wel de grootste innovatie sinds het internet. Toch communiceren LLM's niet direct met de echte wereld. Ze verwerken een input en produceren een output, niets meer. Moderne AI Agents [https://www.ibm.com/think/topics/ai-agents] zijn programma's die LLM's gebruiken om zelfstandig te redeneren welke acties nodig zijn om hun taak te voltooien. Ze zijn niet gebonden aan een specifiek script, wat hen in staat stelt om zich dynamisch aan te passen, bijvoorbeeld op basis van het resultaat van een eerdere actie.

Een digitale assistent voor een webshop is een concreet voorbeeld van een agent die je misschien zelf weleens hebt gebruikt. Als een gebruiker wil weten hoe het staat met zijn bestelling, kan hij de assistent simpelweg vragen: "Wat is de status van mijn bestelling?" en hoeft hij geen specifieke formulieren in te vullen. De agent kan vervolgens afleiden dat hij, om deze vraag te beantwoorden, informatie uit de database moet ophalen, maar ook dat hij hiervoor eerst aan de gebruiker het nummer van zijn bestelling moet vragen.

Tegelijkertijd staan AI-agents nog in de kinderschoenen. De onderliggende modellen zijn krachtig, maar foutgevoelig. Betrouwbare inzet vereist aandacht voor robuustheid, beveiliging en ethiek. Toch is de potentie in praktische toepassingen nu al groot genoeg om actief mee te experimenteren — daarom plaatsen we AI-agents in de *Assess* ring.

Platforms



AsyncAPI

TRIAL

De AsyncAPI-specificatie [https://www.asyncapi.com/en], vergelijkbaar met de OpenAPI-specificatie voor RESTful-diensten, biedt een gestandaardiseerde manier om event-driven en message-driven API's te beschrijven. Deze specificatie vergemakkelijkt het ontwerp, de documentatie en het gebruik van asynchrone API's, waardoor ontwikkelaars formaten van berichten, kanalen en publish-subscribe patronen kunnen definiëren op een taal-onafhankelijke manier. Naarmate organisaties steeds vaker event-driven architecturen adopteren, ontwikkelt de AsyncAPI-specificatie zich tot een goed hulpmiddel voor het handhaven van consistentie en interoperabiliteit tussen gedistribueerde systemen. De AsyncAPI biedt een standaard en we kunnen die gebruiken om documentatie en code te genereren.

Voor projecten, die event-driven API's gebruiken, bevelen we aan AsyncAPI zeker eens te overwegen.

Dependency on US services

De EU stelt via General Data Protection Regulation (GDPR) eisen aan bedrijven die data, gerelateerd aan inwoners van de EU, verwerken en verzamelen. De Amerikaanse wet FISA702 en Executive Order 12.333 stellen de Amerikaanse overheid echter in staat een Amerikaans bedrijf, of een internationaal bedrijf met een kantoor in Amerika, te dwingen om de GDPR te negeren en inzage in data te geven, zelfs als de betreffende servers buiten de VS liggen. Een additionele overeenkomst tussen de EU en de VS, TADPF, is daarom opgesteld, die in de VS wordt bewaakt door het Privacy and Civil Liberties Oversight Board (PCLOB). Recente grillen van de Amerikaanse politiek omtrent het PCLOB doen diverse Europese bedrijven vrezen, dat dit onvoldoende is om te borgen dat hun data en diensten veilig zijn bij Amerikaanse service providers.

Commit. Develop. Share.

AWS European Sovereign Cloud

ASSESS

Om zorgen van Europese bedrijven rondom PCLOB en GDPR te adresseren, heeft AWS de AWS European Sovereign Cloud [https://aws.amazon.com/compliance/europe-digital-sovereignty/] aangekondigd, wat een AWS Cloud is die geheel in de EU gehuisvest moet zijn, bemand en bediend door inwoners van de EU. De eerste locatie van de AWS European Sovereign Cloud zal in Duitsland geplaatst worden en is gepland voor eind 2025. AWS is via een FAQ en whitepapers op de site van deze nieuwe service transparant omtrent het willen naleven van de GDPR en over de impact van FISA702, en benadrukt dat ieder verzoek van een overheid om data in te zien kritisch bekeken zal worden.

Voor een bedrijf dat voor zijn software hevig heeft geïnvesteerd in AWS-kennis en -implementatie, en op zoek is naar een manier om GDPR compliant te blijven zonder die investeringen af te schrijven, denkt JDriven dat het de moeite waard is om te analyseren of deze AWS European Sovereign Cloud een veilig alternatief is. Daarbij moet opgemerkt worden, dat de AWS European Sovereign Cloud aanzienlijk minder services biedt dan de reguliere AWS-cloud. Wat er bovendien in de voorwaarden te lezen valt, is dat AWS uiteindelijk toch zal moeten voldoen aan wettelijke verzoeken om data in de AWS European Sovereign Cloud in te zien, wat de vraag doet rijzen of het uiteindelijk uitmaakt of medewerkers en servers Europees zijn, zolang het moederbedrijf AWS Amerikaans is en dus aan die wetten is gebonden. Dit alles maakt, dat we dit als Assess op de radar plaatsen.

EU Cloud alternatives

ASSESS

Meer dan een decennium geleden besloten diverse Amerikaanse bedrijven als Amazon, Google en Microsoft, dat ze hun serverparken konden uitbreiden en verhuren aan andere bedrijven. Platform-as-a-service (PaaS), infrastructure-as-a-service (IaaS) en simpelweg "de cloud" waren geboren. Over de jaren breidden cloudleveranciers hun diensten uit en kwamen er ook "cloud native" services bij: als klant installeer je bijvoorbeeld niet meer een database op een gehuurde server, maar neem je een databaseservice af, waarbij de cloud provider ook installatie en onderhoud van de software voor z'n rekening neemt. Dit heeft organisaties geholpen om kostenefficiënt infrastructuur schaalbaar te maken en heeft ze bovendien in staat gesteld zich te concentreren op hun eigen unieke competitieve kerncompetenties. Softwareconsultants waarschuwen echter ook al tijden voor vendor lock-in: cloudleveranciers willen klanten lokken met een uniek aanbod, maar dat maakt dat een klant ook lastiger weg kan migreren van een cloud, als ze die services gebruiken.

De laatste paar maanden wordt duidelijk dat er een additionele overweging te maken is rondom vendor lock-in. Amerikaanse cloudleveranciers moeten zich schikken naar een regering die zich grillig of zelfs agressief opstelt richting het buitenland, en daarbij een handelsoorlog heeft ontketend met Europese landen. Europese organisaties moeten zich daarom afvragen, hoe kwetsbaar ze zijn als ze hun software en data onderbrengen bij Amerikaanse cloudleveranciers. Cloud-agnostische platforms als Kubernetes kunnen helpen met ontkoppelen van de softwarearchitectuur en de onderliggende infrastructuur, en zodoende het risico enigszins afdekken. Maar JDriven denkt dat het ook tijd is om te kijken naar de diverse Europese cloudleveranciers en te overwegen in hoeverre hun cloudaanbod voldoende aansluit op hun software- en infrastructuurbehoeftes. Bekendere spelers op de Europese markt zijn bijvoorbeeld Cyso cloud, OVH Cloud, Exoscale, en Hetzner.

Proxmox

ASSESS

Proxmox [https://www.proxmox.com/en/] biedt een alternatief voor het veelgebruikte VMWare-platform. De Proxmox Virtual Environment (PVE) is een opensource-virtualisatieplatform voor het draaien van containers en/of virtual machines met verschillende operating systems. De PVE ondersteunt high availability over meerdere servers, en heeft efficiënte backupfunctionaliteit.

Ondanks dat het geen nieuwe techniek is, is het in de context van groeiende onzekerheid rondom internationale clouddiensten belangrijk om het zelf hosten en in beheer nemen van IT-diensten op eigen servers opnieuw te overwegen. Wij zien Proxmox als een interessantere optie dan VMWare, dat sinds de overname door Broadcom almaar stijgende licentiekosten kent. We hebben Proxmox in *Assess* gezet omdat de toepasbaarheid volledig afhankelijk is van de situatie. De afwegingen zijn lagere (licentie-)kosten en meer controle, tegenover het verlies aan schaalbaarheid en beschikbare support.

Tools

Trial

- 12. warp.dev
- 14. Devoxx Genie
- 15. Bruno
- 19. Nushell

Assess

- 13. CRaC
- 17. JetBrains AI
- 18. DeepSeek
- 20. Cursor IDE

Adopt

- 16. IntelliJ HTTP client

IntelliJ HTTP client

ADOPT

De IntelliJ HTTP-Client is een geïntegreerde functionaliteit binnen IntelliJ IDEA die ontwikkelaars in staat stelt om HTTP-verzoeken direct vanuit de IDE op te stellen en uit te voeren. Deze client maakt gebruik van `.http` of `.rest` bestanden, waarin http requests eenvoudig kunnen worden gedefinieerd. Dit maakt het mogelijk om API's snel en efficiënt te testen zonder afhankelijk te zijn van externe tools zoals Postman.

De HTTP-Client ondersteunt het gebruik van variabelen, scripting en diverse vormen van authenticatie, wat het geschikt maakt voor zowel eenvoudige als geavanceerde use-cases. Responses worden overzichtelijk weergegeven en opgeslagen naast je request bestanden, waardoor het analyseren van deze responses eenvoudig is.

Een belangrijk voordeel is dat deze request-bestanden kunnen worden opgeslagen in de projectstructuur, naast de broncode. Dit bevordert herbruikbaarheid, versiebeheer en samenwerking binnen teams, en draagt bij aan een beter gedocumenteerde en reproduceerbare API-ontwikkeling. Omdat IntelliJ een veel gebruikte IDE is, is het ook een logische keuze voor ontwikkelaars die al vertrouwd zijn met de omgeving, en niet nog een aparte tool willen gebruiken voor API-testing.

De client bestaat al enige tijd, maar er is recent veel goed werkende functionaliteit toegevoegd, waaronder het gebruik van environment-files, voor het veilig gebruik van authenticatie, OAuth2-flows en het gebruik van variabelen. Voor ontwikkelaars die al met IntelliJ werken, biedt het alle features die nodig zijn om API's te testen en te ontwikkelen, zonder dat ze daarvoor naar een aparte tool dienen te switchen.

Bruno

TRIAL

Bruno [https://www.usebruno.com] is een open-source API-client vergelijkbaar met bekende alternatieven zoals Postman [https://www.postman.com/], Insomnia [https://insomnia.rest/] en Hopscotch [https://hoppscotch.io/]. Volgens Bruno's manifesto [https://www.usebruno.com/manifesto] moeten API-collecties worden opgeslagen in de broncode repository, zodat ze als een levende set voorbeelden dienen voor het gebruik van de API, waarmee het meteen "levende documentatie" wordt. Collecties worden volwaardige 'burgers', die samen met de bijbehorende informatie worden bewaard en eenvoudig met versiebeheer kunnen worden beheerd.

Bruno's UI is vergelijkbaar met die van Postman, wat een bekende omgeving biedt als je gewend bent aan andere API-clients. Het ondersteunt uiteraard verschillende HTTP-methoden, request body-formaten en response parsing. Bruno heeft plugins voor Visual Studio Code, en ondersteuning voor andere IDE's zijn in ontwikkeling. Belangrijke functies zijn onder meer environment-variabelen, collectie-variabelen en de mogelijkheid om pre-request en post-request scripts uit te voeren.

Vanuit het oogpunt van een ontwikkelaar is Bruno een fris alternatief voor bestaande commerciële producten, zonder dat je meteen verplicht een account nodig hebt en collecties in een cloud omgeving op moet slaan en delen. Door API-collecties naast de broncode te plaatsen, zijn import/export-processen ook niet meer nodig. Hierdoor verbetert automatisch de samenwerking binnen de ontwikkelteams, en de API-documentatie groeit samen met de codebase.

Hoewel Bruno enkele geavanceerde functies mist die in de bekendere API-platforms te vinden zijn, maakt de focus op gebruiksgemak en integratie met ontwikkelworkflows het een interessante keuze. En omdat Bruno open-source is, zijn verbeteringen en aanpassingen vanuit de community mogelijk (laten we hopen dat de IntelliJ plugin beschikbaar is op het moment dat je dit leest). We zouden het zeker adviseren om Bruno uit te proberen, en hebben Bruno daarom in Trial gezet.

Devoxx Genie

TRIAL

Devoxx Genie is een volledig op Java gebaseerd large language model (LLM) Code Assistant-plugin voor IntelliJ IDEA, ontworpen om te integreren met lokale LLM-providers. Een aantal bekende tools wordt al ondersteund: Ollama, LMStudio, GPT4All, Llama.cpp en Exo. Het kan ook verbinden met cloud-gebaseerde LLM's (OpenAI, Anthropic, Mistral, Groq, Gemini, DeepInfra, DeepSeek en OpenRouter). Enkele van de belangrijkste functies zijn een lokale chathistorie, een tokenkostencalculator, webzoekopdrachten en code-highlighting. De belangrijkste functie is de mogelijkheid om je hele project of pakket toe te voegen aan de context van je prompt wanneer je vragen stelt aan een LLM naar keuze.

Door online LLM's te gebruiken wordt er normaal gesproken data naar servers gestuurd die onder beheer vallen van deze LLM-providers. Bij de projecten waar we als JDriven betrokken bij zijn, wordt nauwelijks gebruik gemaakt van cloud-gebaseerde AI-coding assistants omdat er onvoldoende controle is over wat er met gevoelige data gebeurt. Daarom hebben we Devoxx Genie in de *Trial* ring gezet. De mogelijkheid om nog steeds de kracht van coding assistants lokaal, binnen je IDE, te benutten is iets wat veel ontwikkelaars willen. We geloven dat, hoewel Devoxx Genie nog in ontwikkeling is en zijn eigenaardigheden heeft, het een goed alternatief is voor cloud-gebaseerde oplossingen.

Nushell

TRIAL

Nushell [https://www.nushell.sh] is een nieuw type shell. Het doel van Nushell is om de Unix-filosofie van shells, waarbij pipes eenvoudige commando's met elkaar verbinden, te combineren met een krachtige programmeertaal. Zo wordt Nushell zowel een shell als een programmeertaal in één pakket, dat draait op Linux, macOS en Windows. Nushell combineert een volledige shell met een uitgebreide programmeertaal. Code geschreven met Nushell kan worden uitgevoerd op Linux, MacOS en Windows. Nushell gebruikt overal datastructuren, bijvoorbeeld de output van een `ls` commando, maar ook de output van een REST API call. Er zijn standaard commando's die gecombineerd kunnen worden voor het vinden, sorteren, filteren en converteren van data. Bijvoorbeeld om bestanden te vinden die groter zijn dan 10 MB, en vervolgens te sorteren op de laatste wijziging: `ls | where size >= 10mb | sort-by modified`.

Nushell lijkt al langer te bestaan, maar het is nog wel een `0.x` versie. Daarom is het wel interessant om Nushell te onderzoeken en uit te proberen, om te kijken wat de voor- en nadelen zijn ten opzichte van bestaande shells en tools.

warp.dev

TRIAL

Warp [https://www.warp.dev/] is een terminal-applicatie die de ervaring van de command-line interface verbetert voor ontwikkelaars. De applicatie biedt mooie command-line completion, een geïntegreerde command-palette en een block-based outputsysteem voor verbeterde resultatenvisualisatie en interactie. Maar de meest interessante feature is de mogelijkheid om met natuurlijke taal shell-commando's te genereren met behulp van AI. Ook foutuitvoer van een shell-commando kan worden ingevoerd en wordt AI gebruikt om met een oplossing te komen.

Warp biedt een gratis plan met een beperkt aantal AI-aanroepen per maand. Om onbeperkte AI-aanroepen te hebben, kun je upgraden naar het betaalde plan. Er is ook een optie om voor een team te betalen en dan kan het team gebruik maken van extra opties om samen te werken via de shell. Een enterprise versie is ook beschikbaar, waarbij het bijvoorbeeld mogelijk is om een eigen LLM te gebruiken.

De shell is een krachtige tool voor ontwikkelaar, maar het kan moeilijk zijn om alle command-line opties te onthouden. Warp maakt het gebruik van de shell makkelijker door een natuurlijketaal-interface te gebruiken. Als ontwikkelaar hoef je de shell niet verlaten om alle informatie die je nodig hebt te krijgen en dat is zeer krachtig. Aangezien het nog in bèta is, hebben we het toegevoegd aan *Assess* in onze Tech Radar.

CRaC

ASSESS

CRaC [https://openjdk.org/projects/crac/], Coordinated Restore at Checkpoint maakt het mogelijk om een snapshot te maken van een draaiende JVM en deze op te slaan op de disk. Vervolgens kun je een JVM laten starten op basis van deze snapshot, wat de opstarttijd flink verkort. Het biedt qua opstarttijd een alternatief voor native compilatie met behoud van de flexibiliteit van de JVM, wat voor complexere applicaties een gulden middenweg kan betekenen. Het Spring Framework [https://docs.spring.io/spring-framework/reference/integration/checkpoint-restore.html] heeft hier onder andere flink op ingezet en heeft CRaC volledig geïntegreerd. Er moet echter wel rekening mee gehouden worden dat alle waarden die gezien zijn door de JVM (denk vooral aan secrets) in deze snapshot terecht komen.

Wij vinden het een veelbelovende JVM-toevoeging die een quick win kan leveren voor complexere microservices die veel variatie in load zien, en daarom plaatsen wij CRaC in *Assess*.

Cursor IDE

ASSESS

Cursor IDE [https://www.cursor.com/] is een AI code-editor die gestoeld is op Visual Studio Code (VS Code). Het biedt geïntegreerde AI waarbij het gebruik van AI-support in het ontwikkelproces ingebakken zit. Verder bevat het functies zoals code completion, code generatie, code review, etc.

We zien Cursor recentelijk regelmatig gebruikt worden door sprekers op conferenties, en het is een interessant alternatief voor VS Code. Cursor onderscheidt zich door de geïntegreerde AI-support ten opzichte van AI-plugins die je in andere IDE's kunt gebruiken. Vanuit JDriven zien wij voordelen, zoals het beter meenemen van AI in de developerworkflow en de mogelijke toename in productiviteit doordat met AI randzaken sneller opgezet kunnen worden. Wel merken we dat er een leercurve is om er efficient mee te werken. Bovendien zien we in het algemeen dat zgn. *vibe coding* ertoe kan leiden dat een developer meer bezig is met het reviewen en corrigeren van door AI gegenereerde code, dan met het zelf uitdenken en begrijpen van oplossingen. Dit is een risico, omdat ontwikkelaars zelf verantwoordelijk blijven voor de code die op productie staat. Het raadplegen van de LLM's werkt met credits en wachttijden, die omzeild kunnen worden door te betalen per request. Het is belangrijk te weten, dat requests naar Cursor servers worden gestuurd, wat privacy- en beveiligingsrisico's met zich meebrengt. Deze voor- en nadelen doen ons besluiten Cursor IDE op *Assess* te plaatsen.

DeepSeek

ASSESS

DeepSeek [https://tweakers.net/nieuws/231566/nederlandse-ambtenaren-mogen-deepseek-niet-gebruiken-van-kabinet.html] is een alternatief LLM voor ChatGPT dat kan worden gebruikt voor verschillende taken, waaronder code-assistentie. Het redeneermodel en de training stellen het in staat om nauwkeurig te presteren, zelfs met minder resources, waardoor het deel uitmaakt van een nieuwe generatie efficiënte open-source LLM's die lokaal kunnen draaien en toch sterke prestaties leveren.

Ondanks zijn potentieel is DeepSeek niet zonder controverse. De rechtmatigheid van de data-acquisitie roept, net als bij ChatGPT, vragen op, en er zijn zorgen over mogelijke Chinese censuur. We hebben gemerkt dat veel van onze klanten de voorkeur geven aan het vermijden van cloud-gebaseerde LLM's vanwege het risico dat hun gegevens worden gedeeld. Een lokaal draaiende LLM zoals DeepSeek pakt dit probleem aan door ondersteuning te bieden zonder informatie extern te versturen.

Wanneer het wordt gecombineerd met tools zoals Devoxx Genie, kan DeepSeek in-IDE suggesties bieden voor code. Zijn mogelijkheden reiken echter verder dan alleen coderen, waardoor het een veelzijdige AI-tool is.

We plaatsen DeepSeek in *Assess*, omdat het lokaal kan worden gebruikt en een interessant alternatief biedt, maar nog verder onderzoek vereist.

JetBrains AI

ASSESS

JetBrains [https://www.jetbrains.com/ai/] heeft twee AI-technologieën geïntroduceerd: Mellum (oktober 2024) en Junie (januari 2025). Mellum is een gespecialiseerde LLM, geoptimaliseerd voor code-completion met ondersteuning voor Java, Kotlin, Python en Go. Het heeft drie keer snellere responstijd en 40% acceptatiegraad. Junie is een AI coding agent binnen de Jetbrains Pro en Ultimate lijn die 53,6% (volgens SWEBench) van de ontwikkelingstaken zelfstandig kan oplossen. Waar Mellum functioneert als intelligente code-aanvuller, is Junie een semi-autonome assistent die complexere taken voltooit zoals het genereren van code, het uitvoeren van inspecties en het schrijven van tests. Beide tools zijn ontworpen met privacy als prioriteit – Mellum is getraind op publiek beschikbare code met permissieve licenties – en geven ontwikkelaars controle over voorgestelde wijzigingen. JetBrains' visie met deze tools is om softwareontwikkeling productiever en plezieriger te maken door directe assistentie (Mellum) te combineren met een AI-partner die de ontwikkelingsworkflow fundamenteel kan veranderen (Junie).

Voor een ontwikkelaar is vaak het verzinnen en uitwerken van oplossingen het plezierige deel van ons werk. We besteden al veel tijd aan het lezen en begrijpen van code die anderen (inclusief AI) hebben geschreven, en niet actief betrokken zijn bij het zelf uitwerken van oplossingen maakt dat werk moeilijker. Het is belangrijk om te evalueren of deze technologieën daadwerkelijk de juiste problemen aanpakken, zoals het reduceren van boilerplate code en het bieden van een efficiënt alternatief voor StackOverflow. Ook is het zo dat deze zogenaamde cloud completion tooling je code naar JetBrains' servers stuurt, wat een dealbreaker kan zijn voor sommige organisaties. Bovendien is het goed om te onthouden dat JetBrains AI meer omvat dan alleen deze twee technologieën. JetBrains AI Assistant is met de huidige code completion en code generation opties nog steeds in volle ontwikkeling en ondersteunt ook lokaal draaiende LLM's (net als Devoxx Genie, zie elders in deze radar). Dit is waarom wij Mellum en Junie in de *Assess* ring hebben geplaatst.

Commit. Develop. Share.

Commit.
Develop.
Share.