JDriven Tech Radar Spring 2026

Our view on recent developments in our field

13th edition

Commit to know.

Develop to grow.

Share to show.

jdriven conflux

Opening Remarks

We proudly present the 13th edition of the JDriven Tech Radar.

As many will have noticed, two subjects have been dominating conferences and media in our profession: AI and digital sovereignty in Europe. In such a situation, it's tempting to follow along with the sentiment and unquestionably copy or reject criticism. The way we bring the JDriven Tech Radar about protects us from this. First of all, we need to present convincing arguments to our colleagues about whether a subject and our opinion on it belong on the radar. Once that's been ascertained, we follow a style guide for writing the texts, where each blip always consists of an objective description followed by a subjective opinion. This distinction helps with taking various perspectives into consideration and separating fact and opinion. Once the texts have been written, they're treated the same way we treat any merge request: colleagues review each other's contributions. This way, we can follow trends and interpret them with nuance and substance.

I think we've succeeded at that once more.

Thank you to everyone who made this possible.

Jasper Bogers
Consultant JDriven

Commit. Develop. Share.

Introduction

Our experienced specialists participate every day in the development of many different software projects all over the Netherlands, and are involved in worldwide professional communities. Each half year our engineers gather to discuss the latest emerging trends and developments in software engineering. We seek to capture these trends in a technology radar. Each edition of the radar shows the changes in these trends, compared to the previous edition. A shift can indicate that we see a technology is becoming more, or less, relevant for certain use cases. If a trend does not show up in later editions, it signifies that there are no relevant developments or experiences that cause us to shift our assessment. With this document we will discuss the shifts we have observed over the past half year. This analysis guides us in deciding which technologies we use and recommend.

Tech Radar

The idea to create a Tech Radar was initiated by ThoughtWorks. They periodically showcase their view on new trends and development. In addition, they advise everyone to define their own radar.

At JDriven we fully support that view. Building a Tech Radar is an instructive and valuable experience, where we mutually share our knowledge and create a common awareness around technology. We believe that specialists should be able to create and maintain their own toolset to perform their job to their best ability. When composing a radar, you facilitate a broad discussion on technology, so organisations can strike the right balance between the risks and rewards of innovation. We can help you to kick-start these discussions in your organisation. Let your teams innovate and inspire each other. Together they can draw up a set of technologies and techniques that can accelerate your business.

Overview

The radar consists of quadrants and rings, containing blips to indicate technologies of interest.

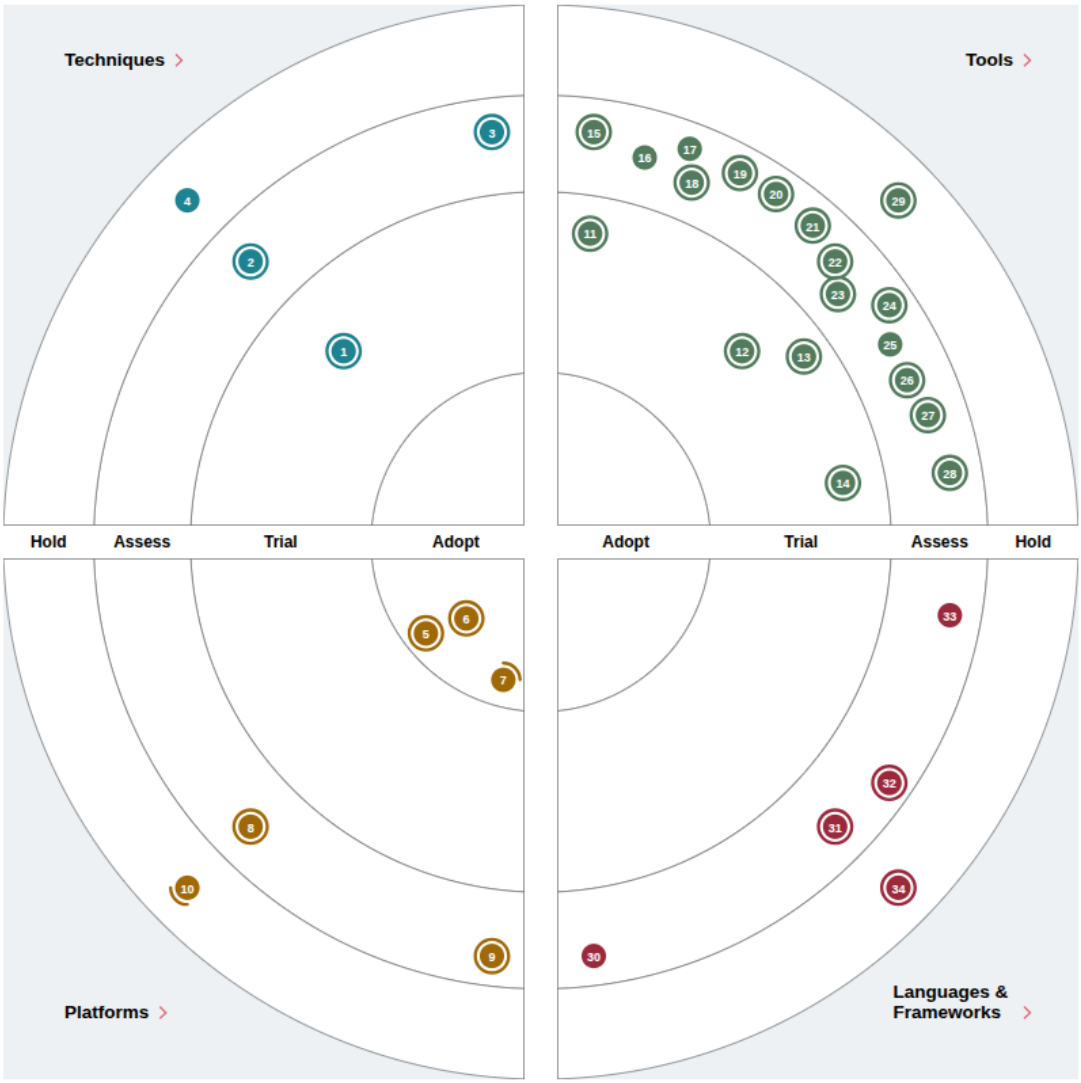
The quadrants subdivide the different subjects into categories:

- **Techniques** help developers create better software.
- **Platforms** to deploy and execute programs.
- **Tools** aid in the development and delivery process.
- **Languages & frameworks** that support developers in their daily tasks.

The rings in each quadrant indicate which phase of the adoption cycle we believe a technology is currently at:

- **Adopt**: We recommend to use this technology, wherever it fits the requirements.
- **Trial**: We advise to gain experience with this technology, wherever a project allows for a certain degree of risk.
- **Assess**: Interesting topic to learn more about and assess its future impact; yet too early to use in production.
- **Hold**: Do not deploy in a project not already using this technology.

In the subsequent sections we will delve into our views on recent developments in the field of software engineering more closely.



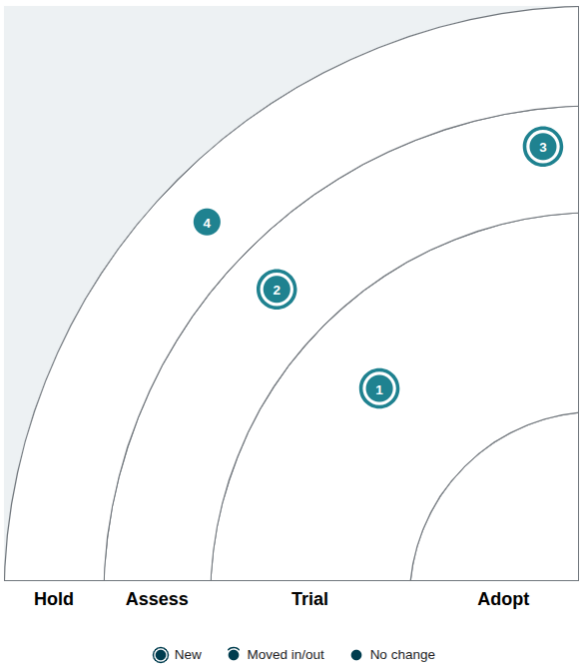
Commit. Develop. Share.

Techniques	Tools
ADOPT	**ADOPT**
-	-
TRIAL	**TRIAL**
1. Open Space Technology	11. Agents.md
ASSESS	12. IntelliJ Spring Debugger
2. AI Reasoning	13. Mattermost
3. Dynamic Consistency Boundary (DCB)	14. NullAway
HOLD	**ASSESS**
4. Vibe coding	15. Agent skills
	16. Anubis
	17. BlueDolphin
	18. Danger
	19. Devbox
	20. Digital Sovereignty Readiness Assessment
	21. Jujutsu
	22. Kepler
	23. Kro
	24. Matrix
	25. Maven 4
	26. OpenCode.AI
	27. Pulp
	28. Pulumi
	HOLD
	29. OpenClaw

Platforms	Languages & frameworks
ADOPT	**ADOPT**
5. Docker Hardened Images	-
6. European Clouds	**TRIAL**
7. Proxmox	-
TRIAL	**ASSESS**
-	30. Amber
ASSESS	31. Arconia
8. Codeberg	32. Embabel Agent Framework
9. Pigsty minio fork	33. Koog
HOLD	**HOLD**
10. AWS European Sovereign Cloud	34. Scala

Commit. Develop. Share.

Techniques



TRIAL

1. Open Space Technology

ASSESS

2. AI Reasoning
3. Dynamic Consistency Boundary (DCB)

HOLD

4. Vibe coding

AI Reasoning

ASSESS

AI Reasoning is an evolution of our earlier blips on AI Agents, specifically focused on the use of AI in business processes rather than in the software development process itself. Where "AI Agents" typically refers to autonomous systems that make decisions independently, AI Reasoning emphasizes support: AI that supports the reasoning process, while keeping humans in control.

The core distinction is *AI-assisted reasoning* versus *AI-based decisions*. AI takes over specific work: analysing, summarising, weighing options, drawing connections. Responsibility for the final decision remains with humans. That distinction is ethically uncontroversial and makes AI Reasoning practically applicable where AI Agents still raise questions.

An insurance company that uses AI to assess damage claims and formulate a recommendation, but where an employee makes the final decision, is a typical example of AI Reasoning in practice.

Caveats still apply here too: formulating requirements precisely, checking output, and correcting intermediate steps takes time — sometimes more than implementing business rules in the traditional way. In the right situations, however, the potential is large enough to warrant active experimentation.

Dynamic Consistency Boundary (DCB)

ASSESS

Dynamic Consistency Boundary (DCB) is an approach for maintaining data consistency in event-sourced systems. In classic DDD, an **aggregate** guards the rules that apply to data (entities with business rules) within a logical boundary (the consistency boundary). As soon as your architecture needs to accommodate a rule that logically requires and affects multiple aggregates, you can use a **saga** or **process manager** to manage coordination. You may also have to resort to using corrective events when a transaction succeeds in one involved aggregate but fails in another. It's worth considering remodelling your aggregates, but you risk models becoming artificial or becoming a poor fit elsewhere. Besides that, remodelling aggregates in a system that has processed data is no small task.

With DCB, the aggregate as a static consistency boundary is abandoned and the saga becomes unnecessary. Instead, when processing a command you take all relevant events in your event store into consideration that led to the current state of the system and you create a temporary consistency bubble. This is made possible by supplying events with relevant tags. After interpreting the current state of the system and processing the command you can emit and write a new event, which allows attaching an **append condition** which is checked by the system to ensure no conflicting events were written in the meantime. This keeps the system state consistent.

You don't need to use event sourcing when working with aggregates, but when working with DCB you use event sourcing without aggregates. There are frameworks and eventstores already that support it, but it's a fundamentally different approach to DDD that has a learning curve. Within the DDD community there's a cautiously positive response to trading the tried and tested, though difficult concept of aggregates for DCB. Making the switch won't be easy, but the benefits are promising and worth assessing.

Open Space Technology

TRIAL

Open Space Technology (OST) is an interactive, informal format that empowers experts to share knowledge through co-creating an agenda in real-time. Participants are free to choose which topics interest them and move between sessions as they please. This flexibility boosts open dialogue, bringing people together for constructive and inspiring discussions.

While Open Space Technology is a well-established format, featured in "Liberating Structures", a toolkit designed to boost collaboration and innovation, lately we have seen its adoption at conferences and within corporate environments. Given our positive experience, we've placed this in *Trial* and recommend others to explore its potential.

Vibe coding

HOLD

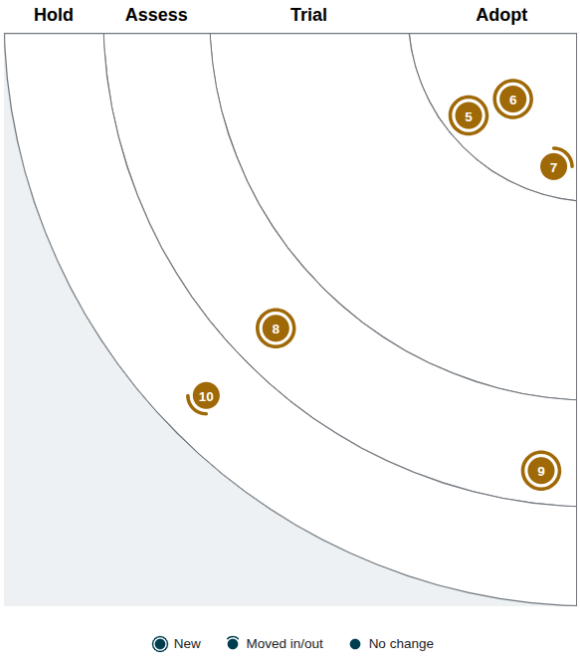
Vibe coding is a term that has recently gained traction in software development. It refers to a development methodology where programmers primarily rely on intuition and feeling when iteratively using AI tools, particularly large language models (LLMs). Instead of structured development processes, it revolves around repeatedly adjusting prompts and experimenting with AI-generated code until the result "feels right" or appears to function.

The practice is characterized by a cycle of prompt engineering, where developers refine instructions based on output, without necessarily having deep understanding of the underlying code, language, or frameworks. Proponents claim this enables faster prototyping, while critics point to risks around code quality, maintainability, and understanding of the software.

JDriven sides with the critics and places vibe coding on *Hold*. It involves a trial-and-error approach that goes against the practices of professional software development, where systematic work, clean code and quality assurance are central. This lack of systematicity leads to technical debt, security issues and poorly maintainable codebases. Fact is that software development involves a lot more than typing in code to get to a solid result, and vibe coding offers no solution for that.

On the other hand, deliberate use of LLMs in a professional development environment *can* add value. As part of a structured development process, AI tools can effectively support developers with code review, documentation, generating boilerplate code, exploring design patterns and offering assistance with explaining technical error messages. The key difference lies in accountable, methodical integration versus the ad hoc experimentation that characterizes vibe coding.

Platforms



ADOPT

5. Docker Hardened Images
6. European Clouds
7. Proxmox

ASSESS

8. Codeberg
9. Pigsty minio fork

HOLD

10. AWS European Sovereign Cloud

Docker Hardened Images

ADOPT

Containerization, or OS virtualization, is the standard way for many developers to build, test, and deploy software. It is therefore important that the containers on which this software runs are secure. To this end, container images may be hardened to minimize vulnerabilities. Previously, this was a task developers had to perform themselves, but since December 2025, Docker has offered free hardened versions of certain images, known as Docker Hardened Images (DHIs). For JVM developers, for example, there's a hardened version of the popular Eclipse Temurin container image (Eclipse Temurin DHI).

We're excited about Docker Hardened Images because it offers a simple way to improve the security of container images without requiring developers to handle hardening themselves. Also, DHIs are drop-in replacements for regular images. Furthermore, CVEs are proactively patched in DHIs, reducing your concerns about vulnerabilities in your container images. That's why we're placing DHI in *Adopt* and suggest using a hardened version of an image whenever it's available.

European Clouds

ADOPT

The EU requires companies who collect and process data related to its citizens to comply with the General Data Protection Regulation (GDPR). The American Cloud Act, law FISA702 and Executive Order 12.333 however allow the US government to force any company that is legally American to ignore the GDPR. The US can demand insight into data, even when these companies' servers are outside US territory. There is no obligation to inform customers of such demands. Another real risk is that services can be restricted, as the International Criminal Court in The Hague found out, when the US forced Microsoft to block its email. This is why more and more companies are searching for European alternatives for their data and services. Many are mentioned on european-alternatives.eu. Initiatives to set this in motion are referred to as working towards digital sovereignty.

From the perspective of development and operations, JDriven looks at cloud providers in particular. When considering a European cloud, it's important to look at the following aspects:

- What does it offer with regard to compute, networking and storage services? Many of these providers can't match the full offering of the major American hyperscalers. But that's less relevant than whether what they do offer is enough to meet the actual needs.
- What are the Service Level Agreements (SLA) for availability?
- What are the prices? Most providers offer a price calculator offering a quick insight and allowing cost comparisons. Depending on which services are used, the differences in expenses per month can be considerable.
- Is there a free tier? A free tier is a major help for building up experience and consequently finding experienced developers.
- Does it offer infrastructure as code and an SDK for reproducible setup? Does it support Terraform, for example?
- How accessible are the learning resources such as tutorials?
- Does it offer some form of managed Kubernetes? This is the container platform that more and more customers use to deploy and run their software, which benefits an exit strategy which involves migrating to a different provider.
- Does it support the Gaia-X project? Gaia-X is a project for federation of digital ecosystems that focuses on data sovereignty, security and compliance. Striving for collaboration and integration under the banner of Gaia-X is an important trait of European cloud providers that American providers do not have.

Taking these things into consideration, we see Scaleway, STACKIT and OVHcloud taking huge steps. Exoscale, IONOS and the Dutch Cyso Cloud are good to keep an eye on as well.

Proxmox

ADOPT

Proxmox offers an alternative to the commonly-used VMware platform.

Proxmox Virtual Environment is a complete, open source server management platform for enterprise virtualization. It tightly integrates the KVM hypervisor and Linux Containers (LXC), software-defined storage and networking functionality, on a single platform. With the integrated web-based user interface you can manage VMs and containers, high availability for clusters, or the integrated disaster recovery tools with ease.

Proxmox Backup Server is an enterprise backup solution for backing up and restoring VMs, containers, and physical hosts. By supporting incremental, fully deduplicated backups, Proxmox Backup Server significantly reduces network load and saves valuable storage space. With strong encryption and methods of ensuring data integrity, you can feel safe when backing up data, even to targets which are not fully trusted.

Proxmox Datacenter Manager is a centralized management solution to oversee and manage multiple nodes and clusters of Proxmox-based virtual environments. Designed for large-scale enterprise environments, it gives IT teams a complete overview of their distributed Proxmox virtualization infrastructures.

Although this technology is not new, in the context of ever-growing uncertainty surrounding international cloud services it may be worth considering hosting and operating IT services on self-maintained servers once more. We think Proxmox is a more interesting option than VMware, as the latter has seen increasing licence costs since its acquisition by Broadcom. We've moved Proxmox to *Adopt* because we're seeing wide adoption in the market.

Codeberg

ASSESS

If you're looking for a European alternative to GitLab or GitHub as a cloud service, Codeberg is an interesting alternative to assess. It offers approximately the same functionalities, albeit less extensive, such as: code storage, static site hosting, pull requests with review capabilities, pipelines, etc. Multilingualism is supported via Weblate. It's based on the Forgejo (Gitea fork) code platform.

Codeberg is a non-profit organization based in Germany. Anyone can create an account, upload code, and collaborate for free. If you do wish to strengthen the continuity of Codeberg, it is possible to become a member (for which you pay a small membership fee) and/or you can make one-time or recurring donations. While aiming for high availability, it does not offer the same formal, guaranteed uptime Service Level Agreements (SLAs) as commercial platforms.

Pigsty minio fork

ASSESS

MinIO was a popular open source product for hosting on-premise S3 buckets without AWS. However, in recent years, MinIO Inc. has tightened its licensing terms, and since February 13, 2026, the MinIO repository has been archived. MinIO Inc. itself still performs maintenance, but no longer makes that code open source.

Organizations using MinIO but not wanting to be dependent on a paid version from MinIO Inc. may consider switching to the Pigsty MinIO fork. This should be usable as a drop-in replacement. Images are available via Docker Hub; binaries via APT and Yum.

Another alternative is Ceph. That is open source and can be used without licensing costs. For applications that use MinIO, this is also a drop-in replacement. But not for administrators, since Ceph is a more complex product. Because Ceph falls under the umbrella of the Linux Foundation, the chance that it will "suddenly" become closed-source (as happened with MinIO or Redis) is virtually zero. The governance structure is set up so that the community and the industry are co-owners. Another open source alternative is SeaweedFS. Less complex than Ceph, but not as simple as MinIO.

AWS European Sovereign Cloud

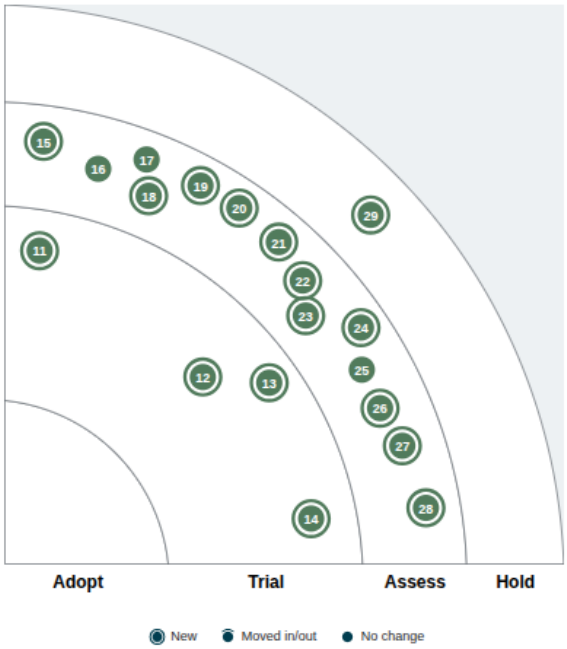
HOLD

To address the concerns of European companies with regard to PCLOB and GDPR, AWS has announced AWS European Sovereign Cloud, which is an AWS Cloud located entirely in the EU, operated solely by European employees. The first location of the AWS European Sovereign Cloud is in Germany within a separate German company, Amazon Web Services EMEA SARL. AWS is transparent about how they mean to comply with GDPR, and about the impact of FISA702, stressing that any lawful government request for data insight would be held to the highest scrutiny. Initially it offers fewer services than the American AWS Cloud, because it wants to be separate from its American parent company on an organisational, technical and physical level.

For any company seeking a way to remain compliant with GDPR without having to write off investments in AWS knowledge and implementation, AWS European Sovereign Cloud looks like a welcome alternative. It's important to note however that the European daughter company ultimately still is a part of the American Amazon.com Inc, which will have to comply with US law. Whether or not the parent company will go through the German court when an American court orders it to force its own daughter company to comply, and what it will do when it gets stuck in the middle, is a legally unclear situation. It remains to be seen how that would play out. It's a risk that is smaller than when you use the American AWS cloud, but it's a risk that needs to be weighed: is the cost justifiable of a migration from AWS US to AWS EU, with its smaller service offering, if sovereignty is a leading argument, or is it worth considering a European alternative?

All things considered we're moving this from *Assess* to *Hold*. And it's worth mentioning that given the current events regarding sovereignty, your *capability* of moving between cloud providers deserves attention.

Tools



TRIAL

11. Agents.md
12. IntelliJ Spring Debugger
13. Mattermost
14. NullAway

ASSESS

15. Agent skills
16. Anubis
17. BlueDolphin
18. Danger
19. Devbox
20. Digital Sovereignty Readiness Assessment
21. Jujutsu
22. Kepler
23. Kro
24. Matrix
25. Maven 4
26. OpenCode.AI
27. Pulp
28. Pulumi

HOLD

29. OpenClaw

NullAway

TRIAL

NullAway is a static analysis tool for Java that detects potential `NullPointerException`s at build time based on nullability annotations.

It is a plugin for ErrorProne, which is a Java compiler plugin that runs static code checks during the build. This way NullAway gives fast feedback in the normal build. Recent versions also support JSpecify and require JDK 17 or higher.

We place NullAway in *Trial*. It can deliver quick value in Java codebases. Adoption still requires a consistent annotation strategy based on JSpecify and integration with ErrorProne.

Commit. Develop. Share.

Agents.md

TRIAL

Agents.md is an open standard used to provide project-specific instructions to AI coding agents. It is a Markdown file that you place in the root of your repository, comparable to a README.md, but intended for AI assistants rather than human developers. Within the file you describe how the project should be built and tested, which coding conventions apply, and which architectural patterns should be followed.

The initiative was originally started by Sourcegraph and is now maintained by the Agentic AI Foundation within the Linux Foundation. AGENTS.md is tool-agnostic and is supported by several AI coding tools such as Kiro, Cursor, GitHub Copilot, and OpenAI Codex. This makes it an appealing alternative to tool-specific files like .cursorrules or CLAUDE.md.

From our perspective at JDriven, we see that an AGENTS.md file noticeably improves the quality of AI-generated code, because the AI gains a better understanding of how the project is structured and which patterns the team follows. An additional advantage is that it also becomes valuable documentation for new team members.

Creating a well-written AGENTS.md file does require an initial investment and a careful balance. Teams need to consciously reflect on their conventions and describe them clearly. The broad adoption of the standard, with tens of thousands of repositories on GitHub, combined with its open nature, gives us enough confidence to place AGENTS.md in *Trial* on our Tech Radar.

OpenCode.AI

ASSESS

OpenCode.ai is an open-source AI coding assistant that works independently of providers and models. The tool runs as a command-line interface, desktop application and IDE extension. OpenCode.ai runs locally and allows you to define multiple agents, for instance for Planning, Building or Reviewing. It understands your code using language server protocols (LSP). You can incorporate skills or MCP-based tools with an explicit permission model. Developers can use models from 75+ LLM providers, for instance OpenAI, Anthropic, Google or even local models.

OpenCode.ai promotes consistent Agentic AI usage within teams by sharing the configuration in your team. Code and context remain local. Configuration and governance remains under your own control through version control and local environment variables.

JDriven sees that OpenCode.ai is mature enough for pilots, but the leap to broad adoption first requires validation and feedback on your own workflows. The tool still needs to prove itself across diverse team and project contexts. We therefore place OpenCode.ai in *Trial*: suitable for experimentation and workflow standardization now, but still too new to deploy organization-wide as standard development tooling.

Agent skills

ASSESS

Agent Skills is an open standard for packaging reusable AI agent capabilities as modular units, originally developed by Anthropic and released as an open specification in December 2025. It enables developers to bundle domain knowledge, instructions, scripts and reference materials into a directory structure with a `SKILL.md` file, which agents can discover and load on demand. The standard is now broadly supported by tools such as Claude Code, VS Code, GitHub Copilot, OpenAI Codex, Cursor and Gemini CLI, and has also been integrated into the Spring AI ecosystem via the spring-ai-agent-utils toolkit.

At JDriven we see Agent Skills as an interesting development for teams working with AI agents in their development process. Its strength lies in its simplicity: a skill is nothing more than a folder with Markdown and optional scripts, making it easy to adopt and version-control alongside your application code. For Java teams the integration via Spring AI is particularly interesting: skills work model-agnostically and require no architectural changes to existing applications.

The standard is still young, however, and evolving rapidly. There is no built-in versioning mechanism for skills, and scripts execute without sandboxing on the local machine, which requires security attention. Additionally, there is no standard human-in-the-loop mechanism, meaning agents can invoke skills and scripts automatically without explicit approval. The broad adoption by major players is encouraging, but best practices for organising and maintaining skills at scale have yet to emerge. For these reasons we are placing Agent Skills on *Assess*.

OpenClaw

HOLD

OpenClaw is a locally running, autonomous AI agent that performs actions directly on an operating system. The functionality of the agent can be expanded using so-called 'skills' via the ClawHub marketplace. These skills cover various functionalities, such as calling external APIs and executing prompts.

However, the agent's direct system access entails risks, and combined with a public marketplace, this creates a supply-chain risk comparable to `npm` or `PyPI`. Initially, there were no checks for malicious skills on the marketplace, resulting in malicious scripts targeting local private data. A new VirusTotal integration now automatically scans for and blocks harmful code. Nevertheless, the security still relies too heavily on system prompts to constrain the agent's behavior. Because a language model is susceptible to deception via untrusted input, text instructions do not constitute a hard security boundary. When the model's output is directly translated into executable commands, the risk of harmful prompt injections simply remains too high.

Within our field, we place a high value on the responsible implementation of such emerging technologies. Although the technology is promising and recent updates are helpful, the out-of-the-box configuration remains risky. Running this software locally without proper isolation is essentially irresponsible from a security perspective. We effectively compare this to executing untrusted external code through, for example, a script runner with full root privileges. The platform can be viable under strict conditions, such as rigorous network isolation within a virtual machine with minimal permissions. Until standard implementations of such platforms enforce hard security boundaries, we are currently placing OpenClaw on 'Hold' for professional use.

IntelliJ Spring Debugger

TRIAL

The IntelliJ Spring Debugger is an IntelliJ IDEA Ultimate exclusive plugin. Its primary focus is Spring Boot projects. Other users report that regular Spring project support can be flaky. It boosts Spring debugging beyond the standard breakpoint debugging.

Spring Debugger's unique selling point is being able to query and manipulate the application context while running. You can interact with all active Spring Beans, including the ones outside of the class where the breakpoint is active. You can call methods on the bean in the plugin window so you can query the repository, invalidate a cache or query the environment as interpreted by Spring. All this functionality makes Spring less of a black box, especially for less seasoned Spring developers.

If you have IntelliJ IDEA Ultimate and are working with Spring Boot, this plugin is highly recommended as it really has no drawbacks. The amount of information it gives you during debugging is significantly bigger than what IntelliJ offers by default. No slow downs or other unwanted side effects were observed while testing.

Maven 4

ASSESS

Maven 4 is currently in Release Candidate stage and brings significant improvements for Java projects. The main gains are in performance: parallel dependency resolution and improved caching can reduce build times significantly, especially noticeable in large multi-module projects. Additionally, Maven 4 introduces the separation between build and consumer POMs, allowing developers to use more flexible configurations without affecting dependent projects.

Improved error messages and automatic parent version inference make working with Maven more intuitive. For teams working with modern CI/CD pipelines, the better integration capabilities with container-based builds and cloud native environments are a welcome addition. Maven 4 introduces a new dedicated `bom` packaging type with support for exclusions and classifiers, simplifying dependency management in multi-module projects. The long-awaited built-in support for CI-friendly versioning with `$+{revision}+`, `$+{sha1}+`, and `$+{changelist}+` properties now works without the Flatten Maven Plugin, along with automatic parent and subproject versioning.

The lifecycle has evolved from a graph to a tree structure with new phase types like `before:all` and `after:all` for more granular control. Profile management has become more flexible with optional profiles using a `?` prefix, preventing builds from breaking when a profile doesn't exist. These improvements make Maven 4 significantly more user-friendly for modern development workflows. There are some breaking changes, particularly around plugin compatibility and stricter POM validation, requiring careful migration. Note that Java 17 is the minimum requirement for Maven 4.

We expect that when Maven 4 reaches GA (General Availability), it will be a solid choice for projects, given the significant performance improvements and the enhanced developer experience. Therefore, it's important to already look at what's involved in a migration. Given the significant performance improvements and enhanced developer experience, we place Maven 4 in the *Assess* ring.

Danger

ASSESS

Danger is an open-source automation tool that runs during CI/CD to enforce code review standards. The tool automates repetitive code review tasks by checking pull requests against predefined rules. Danger runs as an additional step in your pipeline and integrates with platforms like GitHub, GitLab and Bitbucket. It supports multiple languages including Ruby, JavaScript, Swift, Python and since 2022 also Kotlin.

Teams can configure Danger to verify changelog entries, require issue tracker references in PR descriptions or flag architectural anti-patterns. Rather than having reviewers manually note the same issues repeatedly, Danger catches common oversights programmatically. Teams implement their specific standards through code rather than relying on manual enforcement.

While Danger has existed since 2016, the Kotlin support added in 2022 makes it newly relevant for the Java and Kotlin ecosystem. JDriven has seen organizations use Danger to standardize their code review process successfully. The tool helps teams codify review norms and maintain consistency across pull requests. We place Danger in *Trial*: the tool is mature, but Kotlin support is relatively recent and teams need to validate how it fits their specific workflow and review process.

Pulumi

ASSESS

Pulumi is an open source infrastructure-as-code (IaC) tool that lets you define cloud infrastructure in general-purpose programming languages such as TypeScript, Python, Go, .NET, Java, and optionally YAML.

The open source IaC tool is separate from the commercial Pulumi Cloud platform, although that platform can be used for team collaboration and as backend for storing the infrastructure state. Teams that do not want to use Pulumi Cloud can also store state in their own backend, such as S3, Azure Blob Storage, Google Cloud Storage, PostgreSQL, or the local filesystem.

Pulumi is interesting for development teams because infrastructure can be written using the same language, libraries, and development tooling as application code. That lowers the barrier for developers to contribute infrastructure changes and enables reuse, testing, and abstraction through familiar programming constructs.

General-purpose languages make it easy to build clean abstractions, but also to introduce unnecessarily complex deployment logic and teams also have to make deliberate choices around state management, secrets, governance, and collaboration. For organisations that already rely heavily on Terraform or OpenTofu, Pulumi also introduces an additional IaC model and ecosystem to manage.

For these reasons we place Pulumi in *Assess*: it may be mature enough for real projects and can make developers more productive, but we recommend first aligning on team conventions, governance, and state management before trying it on.

Devbox

ASSESS

Devbox is a command-line tool for defining reproducible development environments with project-specific tools and versions.

With Devbox, teams declare the tools a project needs, such as Node, Python, Java or Pandoc, in version control so the same environment can be used locally and in CI (simplifying build pipelines). This eliminates the "works on my machine" issues. It makes a lighter alternative to fully containerised development environments. Devbox is built on Nix and should not be confused with Microsoft Dev Box.

We place Devbox in *Assess*. It offers clear benefits for reproducible development environments, but its Nix foundation and the related learning curve make it worth evaluating before broad adoption.

Kro

ASSESS

Kube Resource Orchestrator (KRO) is an open-source, Kubernetes-native project designed to simplify the management of complex resources by allowing teams to define custom Resource Graph Definitions (**RGD**s). It acts as an abstraction layer that enables the grouping of multiple Kubernetes objects into a single, manageable unit.

kro reduces the operational burden on platform teams while accelerating delivery for developers. It bridges the gap between complex infrastructure management and user-friendly self-service, without tie-in with a specific vendor. Like **helm** simplifies distribution of definitions, **kro** provides a local perspective on complex resource management.

Pulp

ASSESS

Pulp is a free self-hosted open source container and artifact repository. It can be deployed as a single container or in several containers via podman or docker compose. There is also Pulp Operator, which can be used to make it horizontally scalable in a kubernetes cluster. Like Nexus and Artifactory, Pulp can be used as a cache for public container and artifact repositories. It's not as advanced as those products though. For example the *pull through cache* for NPM is still in tech review and although it features a vulnerability reporter for Python packages, it doesn't do so for other repositories. Pulp does offer the possibility of adding your own plugins for additional vulnerability reporting.

We see organizations struggle with increasing licence fees for Nexus and Artifactory. And we hear complaints about the stability and operability. JDriven see Pulp as an interesting alternative, but warns that it lags behind in features. And though it's free to use, there is the extra human and infrastructure cost of maintaining and operating it yourself.

Kepler

ASSESS

Kepler is a Prometheus exporter that attributes energy usage in Kubernetes to containers, pods and nodes.

By measuring actual runtime usage and hardware sensor data, Kepler provides more direct signals for performance, capacity and energy consumption than static analysis or code scanners.

We place Kepler in *Assess*. It is an interesting option for teams that want to make sustainability and efficiency measurable in Kubernetes, but as a CNCF sandbox project it is not yet ready for broad adoption.

Jujutsu

ASSESS

Jujutsu is a version control system (VCS) that offers its own user interface on top of existing storage systems of other VCSes. It started as a Google project and some Google employees continue to contribute, but Google is not the maintainer. Jujutsu strives to ease the daily use of a VCS for developers. At this point in time it's entirely compatible with Git, which is the only implemented backing storage system. It's inspired by features of Git, Mercurial, Sapling and Darcs such as speed, anonymous branches and advanced merge conflict resolution. On top of that it treats every single change as a commit, maintains an extensive log of all actions on the repository and makes performing an undo action simple.

Jujutsu is actively maintained, but calls itself experimental and hasn't reached a 1.0 version yet. It's definitely something to keep an eye on, which is why it's placed in the *Assess* ring on the radar, because combining the strengths of various VCSes in a simple user interface without having to abandon Git seems promising.

Anubis

ASSESS

Anubis is a web AI firewall utility written in JavaScript that presents proof-of-work challenges to the browser in order to protect upstream resources from scraper bots. Modern browsers are able to perform this proof-of-work challenge, which functions as an alternative to CAPTCHAs without requiring real human interaction. A default web scraper bot on the other hand won't be equipped to fulfill the proof-of-work challenge and will be unable to effectively scrape and extract the wanted data. Anubis thus acts as a 'tar pit' to slow down web scrapers, commonly used to gather data for training AI models, to avoid smaller web pages from being over-encumbered by these scraper bots.

Anubis is a young project, and it remains to be seen whether its approach will stay effective once scrapers adjust. But in adapting so that they can perform the proof-of-work challenge, scrapers would slow down considerably, leading to fewer requests. Because implementing Anubis is a relatively small investment, we think it's worth giving it a try. It's important to configure Anubis to let friendly bots like search indexers through, if so desired. The default image served by Anubis, that visitors will briefly see, won't fit with every website theme, but it can be adjusted to one's needs.

Matrix

ASSESS

Matrix is an open protocol for secure, decentralised instant messaging.

Matrix supports federation, end-to-end encryption and bridges to other chat platforms. It is already used by organisations such as the French government via Tchap and in Germany by the Bundeswehr and healthcare initiatives. That makes it an interesting option for organisations that want more control over their communication infrastructure.

We place Matrix in *Assess*. The protocol is strong for open and sovereign communication, but operational complexity and uneven client maturity mean it is not yet an obvious choice for broad adoption.

Mattermost

TRIAL

Mattermost is an open-source online chat service that can be self-hosted. The service is designed as an internal chat for organizations and businesses, positioning itself primarily as an alternative to Slack and Microsoft Teams.

The latest versions support calling, video calling, and screen sharing. There is a desktop application available for Windows, macOS, and Linux, as well as mobile apps for Android and iOS. We see large organisations have been using Mattermost successfully for some time to facilitate communication between developers, administrators, and users.

However, it should be noted that Mattermost has recently made their license terms more restrictive, limiting available functionality and use of the free self-hosted variant. It's an indication of where Mattermost as a company is heading with its service. Especially if free self-hosting is the primary consideration for using it, it's worth investigating whether those terms are still suitable for your situation.

Digital Sovereignty Readiness Assessment

ASSESS

The Digital Sovereignty Readiness Assessment is an open source and vendor-neutral tool to help organizations evaluate their digital sovereignty posture and which steps they can consider to possibly become more sovereign.

Red Hat has released the Digital Sovereignty Readiness Assessment under the Apache 2.0 license on GitHub. It can be installed locally.

Within approximately 10-15 minutes it generates professional reports based on your answers to 21 questions in Yes/No/"Don't Know" format. It covers domains like your data, technical, operational and sovereignty, open source strategy, managed services and executive oversight. The tool calculates your organization's maturity levels in real-time and provides actionable recommendations based on your assessment results. It also provides research questions for follow-up investigation into your "Don't Know" responses.

We believe that the extent to which an organization wishes to be dependent on external parties is a strategic and tactical policy issue. We are therefore placing this Digital Sovereignty Readiness Assessment in *Assess*, as it is one of the first clear tools we've seen that can help answer this question.

BlueDolphin

ASSESS

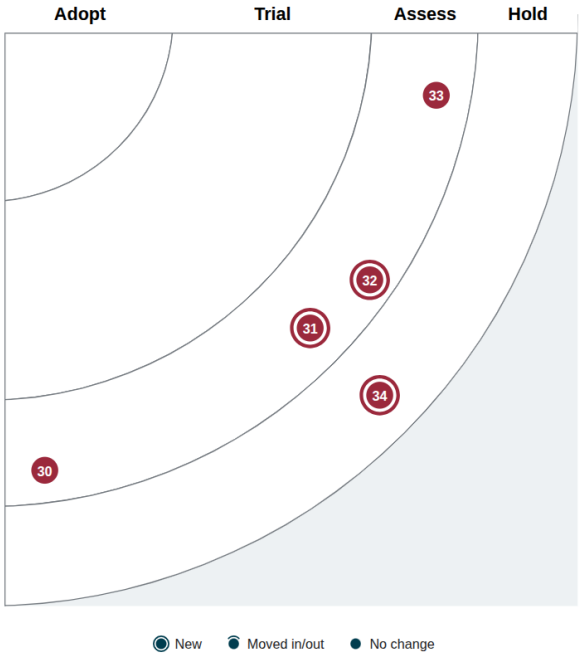
BlueDolphin is a platform for enterprise architecture and process modeling. It offers a comprehensive set of features for visualizing organizational structures, application landscapes, and information flows. Its strength lies in supporting collaboration between various roles within an organization, enabling teams to shape both the current and the desired architecture in a visual and structured way. This enhances understanding of the impact of changes and helps to substantiate architectural decisions.

However, the abundance of features also comes with a downside. Due to the large number of options and configuration possibilities, it is easy for new users, and sometimes even experienced ones, to lose their way. The learning curve is steep, and without clear guidelines or governance, the use of BlueDolphin can quickly become unstructured or inconsistent.

While BlueDolphin has the potential to help organizations structure and communicate their architecture, it requires a clear vision and guidance to be used effectively. It is therefore important to carefully consider the purpose, scope and training needs related to the use within the organization.

For these reasons, BlueDolphin is placed in the *Assess* quadrant of this edition of our Tech Radar.

Languages & frameworks



ASSESS

30. Amber
31. Arconia
32. Embabel Agent Framework
33. Koog

HOLD

34. Scala

Embabel Agent Framework

ASSESS

Embabel is an open-source AI agent framework for the JVM, created by Rod Johnson, the founder of the Spring Framework. It enables Java and Kotlin developers to build agentic flows that seamlessly combine LLM interactions with typed domain models and existing application code. A distinguishing feature is its use of Goal-Oriented Action Planning (GOAP), a planning algorithm originating from the gaming industry, which allows agents to dynamically combine steps to achieve goals.

At JDriven we see Embabel as a promising development for the Java ecosystem. It brings AI agent capabilities to the JVM in a way that aligns with how Java teams are accustomed to working: with strong typing, compile-time checking and seamless integration into the Spring ecosystem. This lowers the barrier for integrating GenAI into existing enterprise applications, without having to switch to a Python stack. The GOAP approach offers advantages over frameworks that rely entirely on LLMs for planning: it is more efficient, more predictable and easier to test.

However, Embabel is a young project that has yet to prove itself in production environments. The community is small compared to alternatives like LangChain4j and Spring AI, on top of which Embabel builds as a higher-level abstraction layer. Additionally, it requires understanding concepts like GOAP and agentic flows, which introduces a learning curve. For these reasons, we are placing Embabel on *Assess*.

Arconia

ASSESS

Arconia is a Spring Boot add-on framework that adds cloud-native capabilities without requiring a switch to another framework.

The framework focuses on developer experience with features such as Dev Services for local development and testing, Kubernetes support, service binding, observability and multitenancy. That makes it interesting for teams already using Spring Boot that want fewer separate libraries and less custom configuration.

We place Arconia in *Assess*. The framework looks promising, but it is still working toward its first stable release in early 2026 and is therefore not yet ready for broad adoption. Also, the project was started by one developer and time will tell what kind of support can be given.

Koog

ASSESS

With Koog, JetBrains has launched an open source framework that helps developers overcome various challenges regarding the creation of AI agents. Our definition of an AI agent here is: An independent application that accepts loosely defined input, sends that to an LLM for interpretation and then returns output that is principally non-deterministic. It's hard to build an application that makes one or more LLM calls in a way that is predictable, reusable and generally reliable. Koog offers various mechanisms for dealing with that, like creating workflows, memorizing conversations, compressing history, and consequently building up a knowledge base for the AI-agent to build on, rather than having to query the LLM every time. It also allows switching to a different LLM during a conversation, for example to only call a slower or more expensive LLM when a faster cheaper one yields insufficient answers. Koog is Kotlin-oriented and supports OpenTelemetry, Model Context Protocol (MCP), Spring Boot and Ktor.

JDriven sees in Koog a framework that might help mitigate the big risks of unpredictable outcomes and possible high costs of using an LLM. It does this up to a point where one may wonder if an application, that goes to such lengths to put safeguards around an AI, wouldn't be better built without AI entirely. But it's there, where developers are reluctant to use AI in their applications and are in search for a solution architecture where non-deterministic components might shine, that such safety is necessary. We think Koog can play a part there, although it will have to compete with LangChain4j and Spring AI.

Amber

ASSESS

Amber is a modern programming language that compiles to Bash shell scripts. It addresses the challenges of error-prone and hard-to-maintain shell scripts by adding compile-time type checking, a modern ECMAScript-like syntax, and a comprehensive standard library. This makes shell scripts more reliable and prevents runtime errors. In 2024, the project won the award for "The Best Engineering Project in Technology" at Wrocław University. The language supports five data types (`Text`, `Num`, `Bool`, `Null`, and `Array`s), provides modern programming constructs such as functions, loops, and conditions, and integrates seamlessly with existing Bash scripts.

Amber is currently at version 0.4.0-alpha and still under active development. While the concept is promising for improving shell scripting, there are still limitations, such as the lack of nested array support due to Bash constraints, and the risk of breaking changes between versions. The language is worth watching for teams that rely heavily on shell scripts, but is not yet mature enough for production use. Therefore, we place it in the *Assess* ring.

Scala

HOLD

Scala originated from a need to apply functional programming principles in the JVM ecosystem. It facilitates object-oriented programming and extends Java's standard features in that area, for example through extension methods and operator overloading.

In 2020, we still placed Scala in the Adopt ring, despite the steep learning curve of Scala and the Scala Build Toolkit (sbt). The reason for this was that version 3 was released, which not only made the language richer, but was also focused on simplifying it. Since then, however, we've seen demand for Scala in the market decrease significantly, particularly in favor of Kotlin. Expertise is difficult to find and is moreover often specifically focused on ZIO, Cats, or Akka, which is not interchangeable. As a result, staffing a Scala project is more difficult and therefore more expensive. Since a strong alternative exists in the form of Kotlin, and Java itself has also continued to evolve, we can no longer recommend Scala for new projects.

Commit.
Develop.
Share.