JDriven Tech Radar Fall 2026

Our view on recent developments in our field

14th edition

Commit to know.

Develop to grow.

Share to show.

jdriven conflux

Opening Remarks

We proudly present the 14th edition of the JDriven Tech Radar.

The past year AI and digital sovereignty have been at the center of conversations in the office and in tech conferences. These two topics are becoming more and more intertwined: developers interested in AI-assisted coding are looking into hosting models locally and engineering their own harnasses, while companies are cutting tokenbudgets and are reluctant to becoming reliant on the big AI-providers or sharing sensitive data with them.

Thankfully, we're also seeing that "tokenmaxing" and "sovereigntywashing" aren't dominating the narrative. Instead there has been growing attention to craftsmanship, architecture and the choices underpinning the use of technology. Tools and platforms that have been de facto standards are being challenged, but people should be aware that a replacement may be susceptible to the same risks.

I recommend paying special attention to the blips that are moving or that you yourself expect to move soon, when looking at this Tech Radar. Do not just look for a new tool or technology to replace your current selection, but consider the bigger picture, in which increased volatility is the new normal. This is a consequence of the combination of current geopolitical instability and technological disruptive innovation. The discomfort that accompanies this stance will disappear as we gain experience with it.

It sure is making our field of software development more exciting than ever.



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.

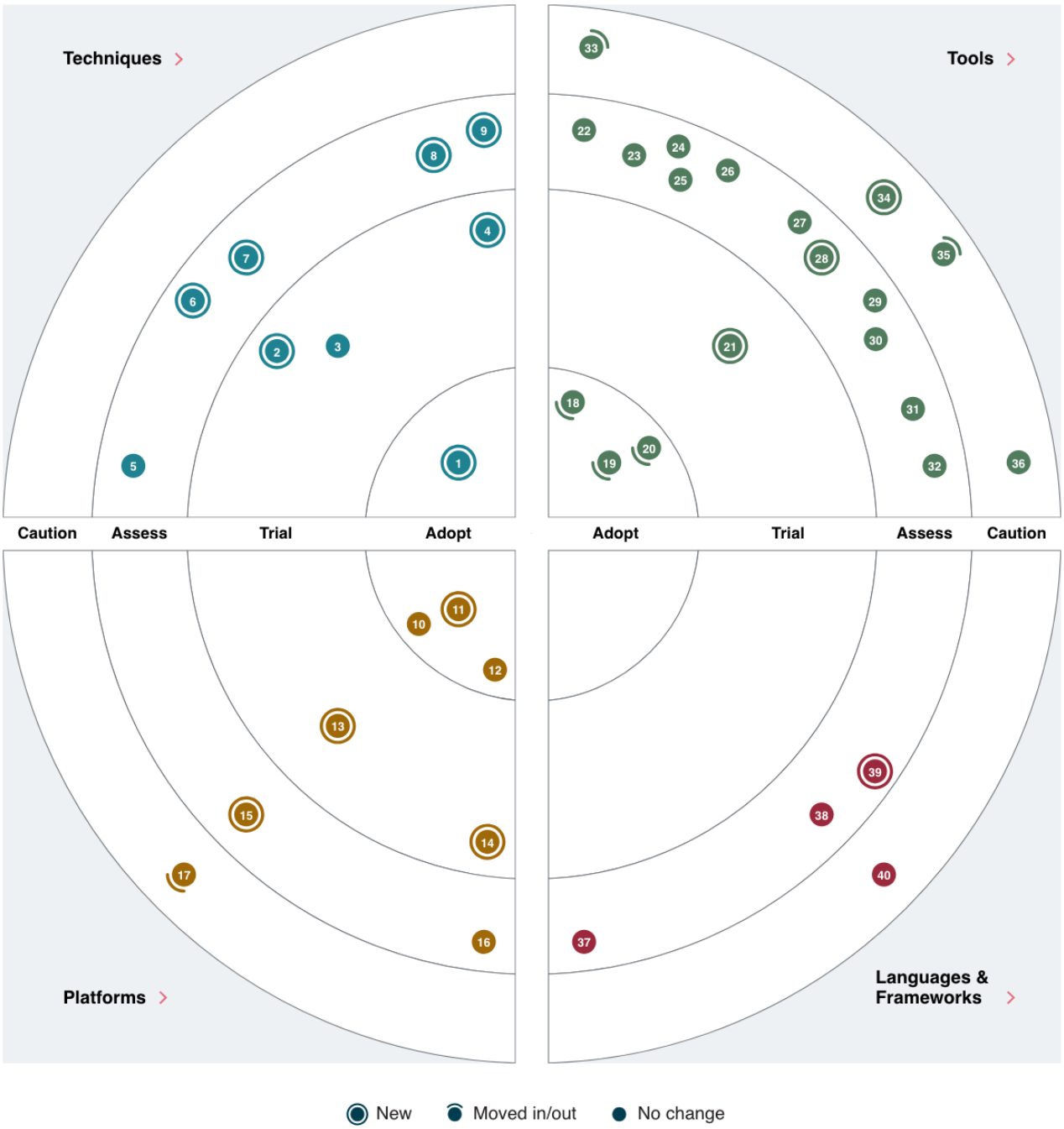
The rings in each quadrant indicate which phase of the adoption cycle we believe a technology is currently at:

- **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.
- **Caution**: Proceed with caution; risks or trade-offs require careful evaluation for your specific context.

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

Commit. Develop. Share.

© 2026 JDriven



Commit. Develop. Share.

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

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

Commit. Develop. Share.

Techniques



ADOPT

1. Post-Quantum Cryptography

TRIAL

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

ASSESS

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

Post-Quantum Cryptography

ADOPT

Quantum computers differ from classical computers in how they - in theory - reduce exponential problems to polynomial ones. A well-known example of this is the traveling salesman problem: calculating the optimal route for a traveler between various cities becomes exponentially more complex with every city added to the route. In simple terms, a quantum computer is able to solve such problems faster because it can calculate all possible routes simultaneously, causing the optimal route to emerge as the solution. There are reasons to believe that this will still have an exponential character in practice, but the key point is that the quantum computer can reduce the timescale for solving such a challenge from "not feasible within a human lifetime" to "worth waiting for."

Another example of something that becomes exponentially more difficult is breaking certain asymmetric public-key algorithms that are used for encrypting data such as passwords. Examples of such algorithms include RSA and ECDSA. Someone with a quantum computer can use Shor's algorithm to decrypt such encrypted data much faster than possible with a classical computer. Post-Quantum Cryptography focuses on finding encryption algorithms that are not vulnerable to this. Examples of quantum-safe algorithms include ML-KEM and ML-DSA.

Although no quantum computer currently exists that is powerful enough to do this yet, there is the danger of harvest now, decrypt later: non-volatile data collected now may still be valuable years later when such a quantum computer becomes available. And so it is important to begin with data classification now, and to already secure data that should remain confidential in ten years' time, using encryption that is resistant to Shor's algorithm. Platforms such as Cloudflare and AWS, as well as software libraries like OpenSSL and Bouncy Castle, now offer quantum-safe algorithms for this, and we recommend using them.

Commit. Develop. Share.

Architecture Advice Process

TRIAL

The Architecture Advice Process is a decentralized approach to architectural decisions, where anyone can make an architectural decision. Before making the decision, the decision-maker must consult two groups: first and foremost, everyone who will be meaningfully affected by the decision, and second, people with expertise in the area the decision concerns. It is encouraged to actively seek out opposing views by soliciting advice from those who hold different opinions. This process fosters a culture of collaboration and knowledge sharing within an organization, enabling architectural decisions to be made more quickly and effectively.

In Scaling the Practice of Architecture, Conversationally Martin Fowler defines four elements crucial to the Architecture Advise Process:

1. ADR (Architecture Decision Records), as a thinking and recording tool. Lightweight documents to capture the decisions and read previous decisions.
2. AFF (Architecture Advisory Forum), as a time and place for conversionations on decisions in the making.
3. Team-sourced architectural principles, to make sure everyone understands what makes a thorough decision.
4. Build your own radar, to capture and share insight on technology trends within your organization.

It stands in contrast to Architecture Review Boards, which are characterized by a centralized approach where a group of experts (the "Ivory Tower") makes decisions regarding the organization's architecture.

In its April 2025 Tech Radar, Thoughtworks noted that the traditional Architecture Review Board approach often causes delays and consequently placed the Architecture Advice Process in the *Trial* ring. We recognize this. We, too, observe that the "Ivory Tower" approach of Architecture Review Boards still persists, resulting in longer lead times and reduced agility for development teams. We therefore advocate for a decentralized approach via the Architecture Advice Process and are placing this technique in the *Trial* ring.

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.

Commit. Develop. Share.

Certificate Lifecycle Management

TRIAL

Not so long ago, a (TLS) security certificate would be valid for two years. As of this year, the maximum validity is only six months. This will continue to decrease to 3 months and then 1 month over the next couple of years. This means that something that used to be done once every two years will now need to be done 24 times more often. The process of identifying, requesting, distributing, and retiring certificates across your environment is called **Certificate Lifecycle Management**. Traditionally, this process was done manually as an ad hoc process, ideally before the active certificate expired. Increasingly, vendors provide automated tooling to facilitate this process. Such a tool is usually simply called (vendor/cloud) Certificate Manager.

You may already be using tooling to automatically renew certificates using the ACME (Automatic Certificate Management Environment) protocol. These tools typically manage the certificate for a single server, which requires distributing configuration as well as credentials. At JDriven we believe you should try out centralized Certificate Lifecycle Management including tooling, in order to be ready for the deadline of March 15, 2029, when certificate lifetimes will be reduced to one month (+ 0.5-month margin). Any nonautomated process still in use at that time will be extremely time-consuming and error-prone, resulting in high costs and a high risk to availability.

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.

Commit. Develop. Share.

Local LLM

ASSESS

Local LLMs (Large Language Models) are neural language models that run within your own infrastructure. This can be on a developer's own laptop or within the organization's own infrastructure. Examples include models such as Llama (Meta), Qwen (Alibaba), and Gemma (Google), which can be run locally using tooling such as Ollama or llama.cpp.

Running a model locally provides more control over data, privacy, and the infrastructure on which the model runs. It can also be interesting when dependency on an external model provider such as OpenAI or Anthropic is undesirable. On the other hand, running larger models can require significant computing power and memory. Which model is suitable depends strongly on the application; smaller or specialized models can be very well suited for specific tasks. The capabilities, performance, and operational characteristics also vary between models and deployment approaches.

Reasons for using a local LLM can be summed up via the 'LOCALAI' acronym:

LOCALAI	**Why local LLMs?**
Local availability	Available immediately within the organization's own infrastructure, without depending on an external service.
Operational independence	No rate limits or dependency on an external provider's models, pricing, or terms.
Cost predictability	Hardware can be expensive, but acquisition and depreciation make costs more predictable, without variable per-prompt fees.
Autonomous data handling	Sensitive data can remain within the organization, which can help with privacy and compliance.
Local control	You choose the model, version, and backend, such as LM Studio or Ollama.
Adaptability	Configuration, quantization, and context can be tailored to the specific application.
Individual tuning (LoRA)	LoRA can be used to efficiently adapt a model for specific tasks, behavior, or style without retraining the entire model.

We see the development of local LLMs progressing rapidly and expect them to become an interesting option for an increasing number of applications. At the same time, hardware requirements, model selection, and operational complexity still depend strongly on the application. Although acquiring or making the required infrastructure available can be costly, the costs are more predictable and can be accounted for in the budget.

Local models can also be interesting when developing software that makes use of agents, MCP (Model Context Protocol) servers, or probabilistic data. For example, a local model can be used for development and testing without depending on an external model provider.

At the moment it's key to gain experience and better understand the possibilities and limitations of local LLMs. The required hardware, model selection, and operational considerations still depend strongly on the application. Especially for applications where privacy, data ownership, or independence from external model providers are important, it may already be worthwhile to investigate local LLMs.

Commit. Develop. Share.

Rules as Code

ASSESS

Rules as Code (RaC) is an innovative approach to designing and implementing laws and policy. It makes it possible to write legislation and regulations in a format that can be interpreted directly by machines. This brings greater precision and transparency to legislation, and makes it possible to roll out legislation automatically, without the risk of misinterpretation in the collaboration between the developers and policymakers involved.

Under the European Commission's Interoperable Europe initiative, attention is being paid to Digital-ready policymaking (DRPM). The EU and its member states are investigating whether we can publish legislation and regulations in a so-called 25th language, one that can be interpreted directly by machines. The Dutch Tax and Customs Administration, with its Wendbare Wetsuitvoering (Agile Policy Implementation) programme, is a frontrunner in this domain. The German Rulemapping method is another example of how the concept of Rules as Code has been applied.

Although when IT projects fail people are quick to hold up the government as an example, differences in the way policy and IT are developed are a problem for organisations of every kind. We see IT departments making ever greater efforts to bring consistency and structure to the interpretation of policy by taking the policymaker's seat during business analysis — which is precisely what leads to the misinterpretation mentioned above. If this struggle sounds familiar, consider Rules as Code as a way to draft policy unambiguously and to standardise and make transparent the process of interpretation and validation through to software implementation.

Spec-Driven-Development

ASSESS

Spec-Driven Development (SDD) is a way of working in which you write a specification first and only then the code. The specification describes what the software should do, what the design looks like and which tasks need to be done. With the arrival of AI coding agents, SDD has become relevant again. A clear specification gives the agent context, so it has to fill in less by itself and the code better matches what the business wants. Tools such as GitHub Spec Kit, Kiro and Tessl support this way of working. They differ in how far they go: in one the specification is a starting point, in another the code is generated entirely from it.

The tools are still young and change quickly. It is not yet clear whether specification and code stay aligned in the long run. For small changes the process also quickly becomes too heavy. We think SDD can make AI-assisted development more predictable, but we want to gain more experience first. That is why we put it on *Assess*.

Commit. Develop. Share.

Tokenomics

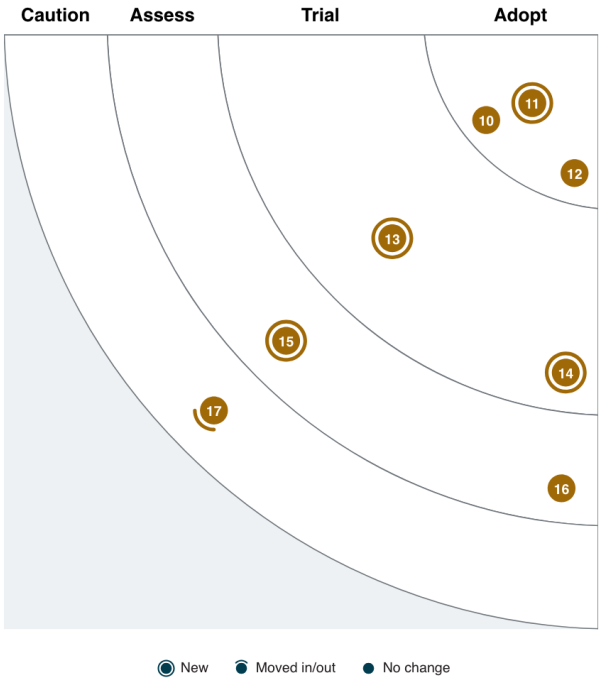
ASSESS

Tokenomics is the practice of managing token consumption by AI models. Tokens are the units in which AI models process text, with the number of tokens determining the cost. The topic is becoming more pressing with AI agents, which carry out tasks independently in multiple steps and consume many tokens in the process. Techniques such as choosing a suitable model per task and sending only relevant context help keep consumption in check.

The goal is not minimal consumption but a deliberate trade-off between cost and quality. A growing number of tools promise savings, but their effect varies greatly per use case. We therefore recommend testing and measuring such tools in your own context first. Because prices, models and limits change rapidly, there are no fixed practices yet. For these reasons we place Tokenomics in *Assess*.

Commit. Develop. Share.

Platforms



ADOPT

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

TRIAL

13. AWS AgentCore
14. Temporal

ASSESS

15. AWS Websearch Connector
16. Pigsty minio fork

CAUTION

17. Codeberg

Docker Hardened Images

ADOPT

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.

Commit. Develop. Share.

AWS S3 Vectors

ADOPT

RAG applications need somewhere to store embeddings. The usual answer is a dedicated vector database, which means another system to set up, run and pay for. With S3 Vectors AWS adds vector storage and search to S3 itself: you create a bucket, store your embeddings and query them, with nothing to provision. It scales the way S3 does, with AWS quoting up to two billion vectors in a single index. With S3's familiar pay-for-what-you-use pricing, for most workloads it is considerably cheaper than a dedicated database. The trade-off is that searches are somewhat slower, though still well under a second, and the feature set is more basic. This suits straightforward vector storage without competing with dedicated databases on raw performance. For organisations already running on AWS, it is an easy addition that fits naturally alongside the storage and tooling they have in place.

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.

Commit. Develop. Share.

AWS AgentCore

TRIAL

Deploying an AI agent to production raises issues that ordinary applications don't. Agents run for minutes or hours rather than milliseconds; they generate artifacts along the way, such as Word documents or spreadsheets, and they execute code and call tools on a user's behalf. That makes security a real concern, and scaling long-running tasks that need to keep track of where they are is not something a typical web stack handles well.

AWS AgentCore Runtime addresses this by starting a small, dedicated virtual machine for every user conversation. Each conversation gets its own isolated environment with its own processing power, memory and file system, so an agent can simply write generated files to local disk without any risk of them leaking into another user's conversation. When the conversation ends, the environment is discarded. Because AWS manages the provisioning, scaling requires no extra work. A conversation can run for up to eight hours.

The runtime is framework-agnostic and takes your packaged application as-is, so existing agents can be deployed without rewriting them. Similar products exist from other vendors, but if you are already running on AWS infrastructure, this is the one that will integrate most naturally with what you have.

Temporal

TRIAL

Temporal is a platform that makes it easy to implement robust processes by automatically tracking exactly where a process (workflow) is.

This concept is now widely known in the industry as *Durable Execution*. You write code that implements a software process with an arbitrary execution time, which can, in principle, run indefinitely. In Temporal, your code *is* the workflow.

Behind the scenes, the platform keeps track of exactly which steps have been completed and what the results of those steps were. This allows the process to be resumed when necessary, automatically skipping any steps that have already completed successfully.

Several platforms and libraries explicitly support the *Durable Execution* concept, with Temporal being the most mature option. Temporal provides SDKs for virtually all mainstream programming languages, and JDriven has evaluated its Java support and found it to be solid. With the rise of generative AI, writing stateful orchestration logic such as within or across agents is becoming increasingly relevant. Since Temporal is a mature platform that offers both a managed cloud solution and an open source, self-hosted version, we have decided to place it in the *Trial* ring.

AWS Websearch Connector

ASSESS

Giving an AI agent access to web search makes it considerably more useful, but as a downside queries leave your environment, hit third-party search APIs, and results in outbound traffic. This brings compliance into question. Web Search on AgentCore avoids this by running queries against Amazon's own web index. The agent searches the web, but the request never leaves AWS infrastructure and never touches the public internet.

Results may be slightly behind dedicated public search products, but for organisations where security and compliance come first, that is a trade-off worth making. On the other hand doing this does tie you to Amazon's web index, which in itself may be objectionable from a sovereignty perspective.

© 2026 JDriven

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.

Codeberg

CAUTION

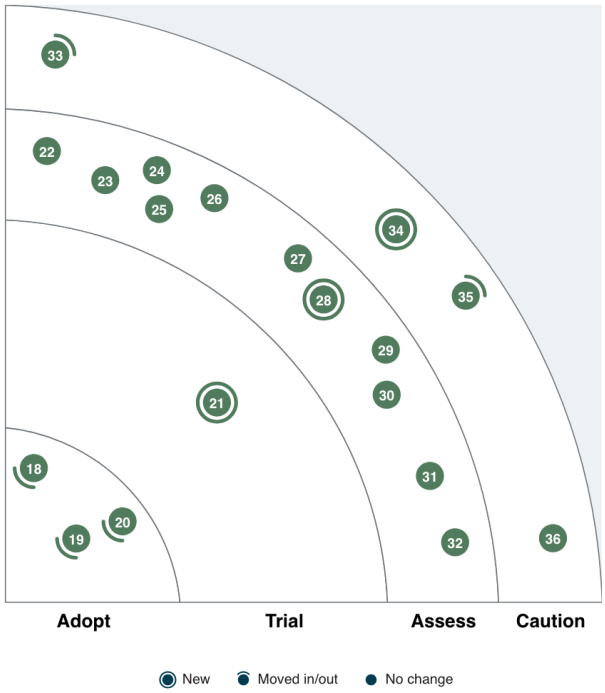
If you're looking for a European alternative to GitLab or GitHub as a cloud service, Codeberg is an interesting alternative to assess. It offers approximately the same functionalities, albeit less extensive, such as: code storage, static site hosting, pull requests with review capabilities, pipelines, 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.

Recently, Codeberg has updated its terms of service. It no longer allows for code that is (partially) generated with AI to be hosted in a Codeberg repository, due to the risk of copyright infringement. This is an understandable yet bold move and could mean that Codeberg is not the place to host your code if you want to use AI-assisted coding, now or in the future.

Commit. Develop. Share.

Tools



ADOPT

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

TRIAL

21. KYAML

ASSESS

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

CAUTION

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

NullAway

ADOPT

NullAway is a static analysis tool for Java that detects any potential `NullPointerException` 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 are moving NullAway from *Trial* to *Adopt*. It is a good tool to get your null-safety discipline in order, especially with the upcoming null-restricted value class types (with `!` and `?` markers) as part of Project Valhalla, even if the eventual official Java notation may end up differing from NullAway's annotations.

© 2026 JDriven

Agents.md

ADOPT

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 nearly all major AI coding tools, such as Claude Code, GitHub Copilot, OpenAI Codex, Cursor, and Google Antigravity. For Claude Code this has only been the case since version 2.1.277. Before that it required a separate CLAUDE.md file.

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.

In a short time AGENTS.md has become the market standard. A single file now serves the whole team, regardless of which AI assistant each member uses. We therefore move AGENTS.md to *Adopt* and recommend it for every project that uses AI coding tools.

IntelliJ Spring Debugger

ADOPT

The IntelliJ Spring Debugger takes Spring debugging to a whole new level compared to standard breakpoint debugging. Since the previous radar, this plugin has moved from Trial to Adopt: the added value is clear and we see few drawbacks.

During a debug session, the Spring Debugger gives you access to the full application context. All active beans are visible, including those outside the scope of your current breakpoint. You can call methods on beans living elsewhere in the application. This lets you, among other things, invalidate caches, run queries directly through your repositories, and inspect or modify table contents on the fly, all without interrupting the session. It makes the Spring *magic* easier to understand, which will especially help less experienced Spring developers.

Environment variables are also shown exactly as the application read them. You get more insight into how beans are instantiated and where their values come from. This makes the internals of Spring far more transparent and concrete, especially for developers who are less familiar with the framework.

The amount of information available during a debug session is significantly larger than what IntelliJ offers by default. We observed no noticeable slowdowns or other unwanted side effects.

Do you work with Spring Boot in IntelliJ IDEA Ultimate every day? Then this plugin belongs in your standard setup. It is only available for Ultimate and primarily targets Spring Boot; results with plain Spring without Boot may vary.

© 2026 JDriven

KYAML

TRIAL

KYAML is a safer and less ambiguous subset of YAML. Designed specifically for Kubernetes, KYAML addresses common YAML pitfalls such as whitespace sensitivity and implicit type coercion while maintaining full compatibility with existing YAML parsers and tooling. For example, it solves the infamous 'Norway problem': does NO in an enumeration mean `false` or the country code for Norway? KYAML has specific benefits when used in templating, such as in Helm Charts.

JDriven feels that the strong push from the Kubernetes ecosystem warrants pushing KYAML beyond directly into the *Trial* ring. Do note that, since KYAML is a subset of YAML, it can be used anywhere YAML can be used, so also outside of Kubernetes.

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*.

Commit. Develop. Share.

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.

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.

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.

© 2026 JDriven

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.

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.

LiteLLM

ASSESS

LiteLLM gives you a unified, OpenAI-compatible interface to more than a hundred different LLM providers, including Anthropic, OpenAI, Gemini and Amazon Bedrock. The tool also works with local models.

LiteLLM can be used in two ways: as a Python SDK or as a self-hosted proxy server (also known as an AI gateway). With the SDK you can easily switch models or providers in your application without rewriting code. The proxy adds further capabilities, such as centrally managing access to providers, availability options (fallbacks, retries, etc.) and collecting logging and metrics.

The proxy can also be used with existing AI agent harnesses such as OpenCode and Claude Code. In that setup it makes the use of LLMs within an organization easier to control. You hand out virtual API keys per team or project, each with its own budgets and limits. This prevents provider API keys from spreading across countless applications and developers.

Some features, including single sign-on and advanced audit logging, are behind an enterprise license.

Do you work with multiple LLM providers, or do you want to manage and gain insight into LLM usage across your organization centrally? Then it is worth running a test with LiteLLM. We therefore place LiteLLM in the *Assess* ring. The project is developing very quickly (with regular breaking changes), and beta features are sometimes introduced that ultimately remain available only through an enterprise license. Earlier this year LiteLLM fell victim to a supply chain attack (through a compromise of Trivy), in which malware-laden versions of the software were temporarily published on PyPI. Measures have been taken to prevent a repeat, such as using Trusted Publishing on PyPI and signing container images with cosign.

Commit. Develop. Share.

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.

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.

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.

© 2026 JDriven

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.

Anubis

CAUTION

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.

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. Anubis is a young project, and when we first encountered it, it remained to be seen whether its approach would stay effective once scrapers adjust. The reasoning was that modifying a scraper, so that it can perform the proof-of-work challenge, would make it slower, able to send fewer requests.

By now AI bots appear capable of easily devising workarounds, such as anubis-fetch, to circumvent it. Websites such as Codeberg report that AI crawlers are increasingly managing to bypass Anubis and are exploring additional countermeasures. Implementing Anubis represents a relatively modest investment, but is likely insufficient on its own, while it could still create noticeable delays for human visitors.

Commit. Develop. Share.

Hermes

CAUTION

Hermes Agent from Nous Research is an open-source AI agent that improves itself the longer you use it. It is an alternative to OpenClaw, which we already put on 'Caution' in the previous radar.

Hermes uses a 'learning loop'. After completing a task it records reusable skills on its own and refines them over a longer period. The agent also maintains its own memory and can easily search for information from earlier sessions. The agent can be used from the terminal (text user interface) and from chat platforms such as Telegram, Discord, Slack, WhatsApp and Signal. For parallel work it can spawn isolated subagents, and a built-in scheduler runs recurring tasks. Hermes can be used with an LLM provider of your choice.

Hermes has the same drawbacks as OpenClaw. The agent executes shell commands autonomously, has access to credentials and is reachable from the outside via chat platforms. Here too, text instructions to the language model pose a risk because of the possibility of prompt injection and untrusted input data.

For specific, well-defined use cases you can run Hermes in a controlled environment. Think of a dedicated device for the agent, with limited (network) access to self-hosted services. Also, don't give the agent its own provider API keys, but let a gateway such as LiteLLM add the keys to LLM requests. If you are unsure how to set this up safely, don't start with it for now. Because of these risks, we place Hermes Agent, like OpenClaw, on 'Caution' for professional use.

Mattermost

CAUTION

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. Or one could consider a fully open-source alternative such as Matrix/Element, which is popular among European government entities (such as the French government with Tchap and the German Bundeswehr).

Commit. Develop. Share.

OpenClaw

CAUTION

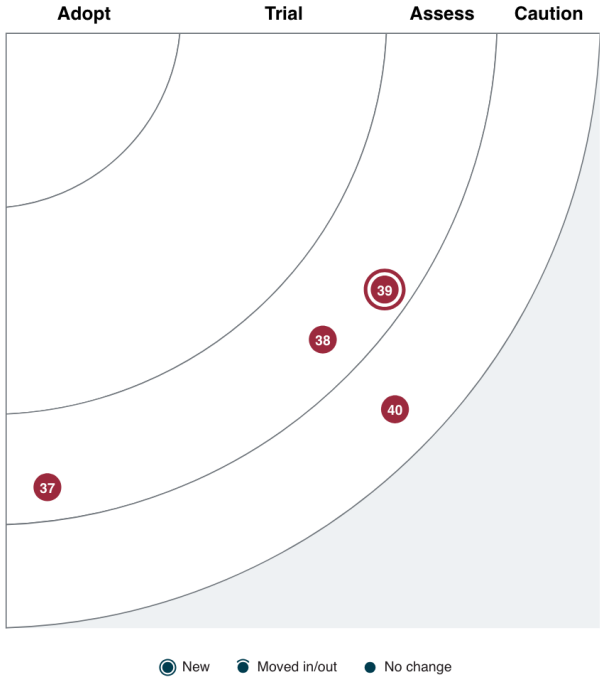
OpenClaw is a locally running, autonomous AI agent that performs actions directly on an operating system. The functionality of the agent can be expanded using so-called 'skills' via the ClawHub marketplace. These skills cover various functionalities, such as calling external APIs and executing prompts.

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 'Caution' for professional use.

Commit. Develop. Share.

Languages & frameworks



ASSESS

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

CAUTION

40. 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*.

Commit. Develop. Share.

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.

EU Cloud Sovereignty Framework

ASSESS

The European Commission (EC) has commissioned a framework to test whether cloud providers meet the concept of sovereignty within Europe. It is called the EU Cloud Sovereignty Framework. It uses eight weighted sovereignty categories and five levels (Sovereignty Effectiveness Assurance Levels, or SEALS) for assuring sovereignty effectiveness. For each category, a minimum assurance level can be determined that may serve as a rejection threshold, if the framework is used for considering a digital service provider. The results can also be aggregated into an average sovereignty score. Together, these elements can be used to gain insight into the dimensions along which an organization could increase its "SEAL level".

The EC initially developed the framework to assess its own tender for digital services. Based on that experience, the EC recommends that European companies adopt this assessment as a kind of benchmark for evaluating a cloud service provider, although this carries no legal force. In a previous radar, we mentioned the Digital Sovereignty Readiness Assessment from Red Hat as a means to test the sovereignty of one's own organization. We believe the EU Cloud Sovereignty Framework can also serve in that capacity and may even be a better yardstick for assessing existing and future digital services. Its categories, scoring levels, and weightings are more granular and more detailed. And it explicitly takes into account matters such as GDPR, NIS2, and DORA, which are specifically important in the EU and which the Red Hat assessment does not address. JDriven therefore recommends taking the European Commission's advice into consideration and examining whether the EU Cloud Sovereignty Framework could help in assessing the organization's sovereignty objectives and, where necessary, elevating them to a higher level.

Commit. Develop. Share.

Scala

CAUTION

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.



Commit.
Develop.
Share.