JDriven Tech Radar Spring 2025

Our view on recent developments in our field

11th edition

Commit to know.

Develop to grow.

Share to show.

jdriven

Commit. Develop. Share.

Opening Remarks

We proudly present the 11th edition of the JDriven Tech Radar. Contrary to the last milestone edition, we've foregone the special artwork. But as always, putting this Tech Radar together was a remarkable process that's worth highlighting.

Our JDriven colleagues work for various customers, in various sectors, with various tools, techniques, languages, frameworks and platforms. We attend conferences as guests, speakers and organisers, and in doing so we manage to get a pretty decent impression of what's going on in our field of work. We also stay in touch with each other in the office, via Slack and JDriven events, and share our observations, experience and opinions. Every now and then, a colleague will give a subject special attention through a blog, workshop or hackathon.

Twice a year we meet up to see where we stand with regard to all of this. We'll discuss what we've spotted coming at us, and consider whether it's worth sharing with our peers and our customers. We look back at what we found noteworthy before, and every JDriven colleague can put new ideas up for discussion. This could be a new framework that doesn't suffer the drawbacks of a well-known alternative, an old methodology that proves to work well with a new technology, or a tool that makes life easer for developers. As we consider these tools, we take into consideration relevance, actuality, granularity and self-evidence. For example, a new version of Java or Kotlin isn't going to appear on the radar. And we would rather mention specific products than generic concepts that those products implement, unless such a concept itself is something to analyse, assess and discuss first.

What's interesting about our Tech Radar sessions is that none of the attendees is an expert on every single subject. Yet the setting for sharing insights and arguments allows us to make the decision to include things on the radar as a team. These subjects, or 'blips' on the radar, are then distributed among our colleagues to elaborate and write down concisely. In that process, we always take care to separate objective description and the opinion of JDriven. The latter is a risk: we could be wrong, advise against something that someone is using to great success, or recommend something that turns out to be a failure. But this is inherent to our field of work, that is always in motion. And thankfully, what we've noticed in the past few years, is that it's good to offer food for thought, to engage with each other on these matters, and gain new insights.

Thank you to everyone who made this possible. Let's keep building an active and growing future for software engineering!

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

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.

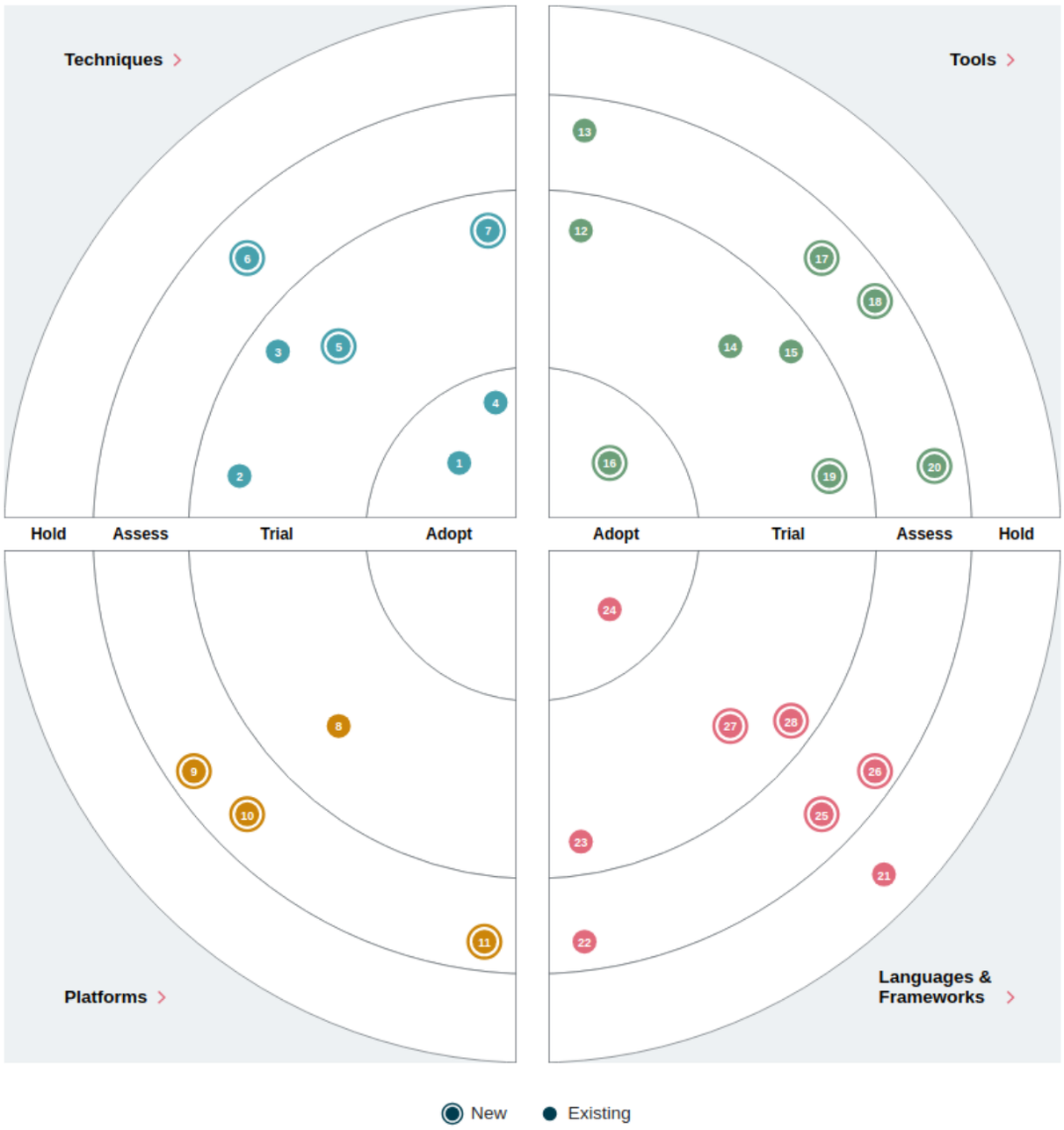
The quadrants subdivide the different subjects into categories:

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

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.

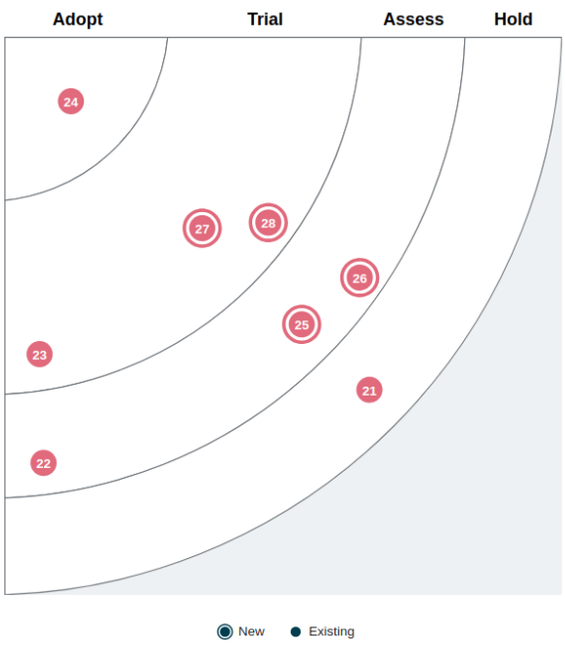


Commit. Develop. Share.

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

Commit. Develop. Share.

Languages & frameworks



Hold

21. Micronaut SourceGen

Assess

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

Trial

23. TimeFold AI
27. Fabrikt
28. JSpecify

Adopt

24. Spring Modulith

Spring Modulith

ADOPT

Domain driven design has helped developers understand that a software architecture can and must facilitate the evolution of software systems to follow changes in business requirements and insights. Using microservices is a way of isolating functional modules to that end, but it introduces challenges with regard to service orchestration and HTTP-based communication. This has made developers reconsider whether this modularity can be achieved and enforced in monolithic applications.

Spring has been a framework that provides technical stereotypes, such as `@Controller`, `@Repository`, etc. without regard for how components depend on each other. With Spring Modulith [https://spring.io/projects/spring-modulith], it aims to be more opinionated on the domain-aligned structure of a monolithic Spring Boot application. In practice, it seems to mean that a properly set up, well-structured Spring Boot application using Spring Modulith will limit candidates for dependency injection based on package structure convention. It offers ways to deviate from the defaults by customizing module detection.

Spring Modulith has additional annotations available to mark packages as part of a chosen architectural pattern. Besides this, Spring Modulith also supplies a number of ArchUnit tests to verify the chosen architectural pattern. By default, it supports Layered, Onion and Hexagonal Architecture and has the ability to generate different kinds of documentation standards. Spring Modulith also provides a new way of integration testing modules independently, which will be a lot faster than the old `@SpringBootTest` because it only instantiates the classes of the module under test. Furthermore, they recommend using events to provide communication between the modules.

Recently, Spring Modulith has been released officially, and we see Spring Modulith as a good way to add modularity to a monolithic application.

Fabrikt

TRIAL

OpenAPI, formerly known as Swagger, is a popular interface specification for defining and documenting APIs. The best-known tool in use for generating code based on an OpenAPI specification is probably OpenAPITools' openapi-generator. Fabrikt is an alternative, that focuses purely on generating Kotlin code from an OpenAPI v3 specification. It supports Spring, Micronaut Ktor, plus it offers a Gradle plugin.

Contrary to openapi-generator, Fabrikt deals well with the more complex structures of OpenAPI v3, with *anyOf* being a well known example. Besides that, the smaller technical focus helps the team behind Fabrikt manage to keep up with open issues and merge requests, where openapi-generator lags behind and has built up a substantial backlog. These are reasons why we advise using Fabrikt for OpenAPI v3 based Kotlin code. Because Fabrikt can handle more complex specifications and can generate code different from openapi-generator, we recommend starting small and evaluating the impact of a migration.

JSpecify

TRIAL

JSpecify [https://jspecify.dev/] is an annotation framework designed to improve nullability handling in Java by providing explicit type information, inspired by Kotlin's nullability system. It helps prevent null pointer exceptions by standardizing nullability contracts at compile-time. The project has been in development for over five years and is backed by major players such as Google, JetBrains, Meta, Microsoft, Oracle, and Uber. With the release of version 1.0, the specification has stabilized, providing a solid foundation for adoption. Currently, IntelliJ IDEA, Project Lombok, NullAway, and the Checker Framework already provide support for JSpecify.

Migration guides are available to transition from JSR-305 or Checker Framework annotations to their JSpecify variants. With Spring Framework 7 embracing JSpecify and broader toolchain support emerging, it is becoming a practical choice for improving Java's null safety. While ecosystem-wide adoption is still ongoing, JSpecify is stable and production-ready, which is why we place it in the Trial ring.

TimeFold AI

TRIAL

Timefold AI [https://timefold.ai/] is an open-source constraint solver available for Java, Python, and Kotlin. It is designed to optimize complex planning problems such as scheduling, routing, and resource allocation. Using constraint satisfaction programming, a score can be assigned to a solution, and the problem can be optimized by utilizing one of the built-in search algorithms. Timefold AI builds on the foundation of OptaPlanner, a project that is no longer under active development, with improvements focused on performance, including an enterprise version that offers multithreading for enhanced efficiency.

At JDriven, we have observed recurring demand for constraint solvers in client projects and see Timefold AI as a promising alternative to custom-built solutions, especially for problems of moderate size. Hence, we place Timefold AI in the *Trial* ring, encouraging teams to experiment with it.

Declarative Gradle

ASSESS

When using Gradle, there are many different ways to structure your build scripts. This can cause a lot of confusion, especially when working with multiple projects that have different approaches. Gradle is aware of this, and at the end of 2023, they announced that they had started an experiment to make Gradle declarative. This project is called Declarative Gradle [https://declarative.gradle.org] and is still in an experimental phase. The goal of this project is to provide a fixed structure for Gradle build scripts, making them easier to read and maintain. This should also improve consistency between projects.

We place this new standard in the *Assess* ring, because we find it worthwhile to follow developments in this area and see what impact it will have on Gradle build scripts.

Helidon

ASSESS

Helidon [https://helidon.io/] is an open-source microservice framework, comparable to Spring Boot and Quarkus, and offers a lightweight alternative akin to Ktor and Vert.x. It supports both reactive and imperative programming styles, with two versions: Helidon SE (Standalone Engine), a lightweight, reactive framework, and Helidon MP (MicroProfile), which provides full compatibility with Jakarta EE MicroProfile. In version 4.0, Helidon introduced significant enhancements, including native support for virtual threads, improved gRPC integration and performance optimizations aimed at reducing resource consumption.

Helidon is backed by Oracle, which can be seen as both a strength and a potential limitation. On the one hand, Oracle's backing provides strong enterprise support and long-term stability. On the other hand, some teams may have concerns over vendor lock-in or dependency on a large corporate entity for innovation.

Given the developments in version 4.0, we advise teams looking for alternative Java-based microservice frameworks to consider Helidon. Therefore, we place Helidon in the *Assess* ring.

Model Context Protocol (MCP)

ASSESS

Model Context Protocol (MCP) [https://modelcontextprotocol.io/introduction] is an open standard that enables applications to provide context to large language models (LLMs) in a consistent way. MCP has integrations with databases (PostgreSQL, SQLite), platforms, filesystems and many other systems. By reducing reliance on proprietary solutions, MCP improves interoperability with multiple LLM models. It also has the potential to help non-technical users query and understand their data more easily by allowing AI models to access relevant context dynamically.

However, adoption depends on industry support, and challenges remain in standardizing context delivery across diverse systems while addressing security and performance concerns. We are placing MCP in *Assess* as it shows a lot of potential, but still needs to be proven in practice.

Micronaut SourceGen

HOLD

Micronaut SourceGen [https://micronaut-projects.github.io/micronaut-sourcegen/latest/guide/#introduction] is a library that allows you to generate source code for Kotlin and Java. It is a fork of JavaPoet [https://github.com/square/javapoet] and has been updated to use newer Java constructs, like records. Micronaut SourceGen provides an easy-to-use API to define the structure of the source code that needs to be generated. The library uses an annotation processor to generate the source code during the Java or Kotlin compilation.

Included with the library are annotations to generate builders and withers for records. This could replace the builder annotations provided by Lombok in a project.

The nice feature of the library is that it uses the default annotation processor of the compiler. Only new source files are created and it will not mangle existing class files. The API to create the source code is easy to use. The library is placed in *Assess* to try out and experiment with.

Techniques

Adopt

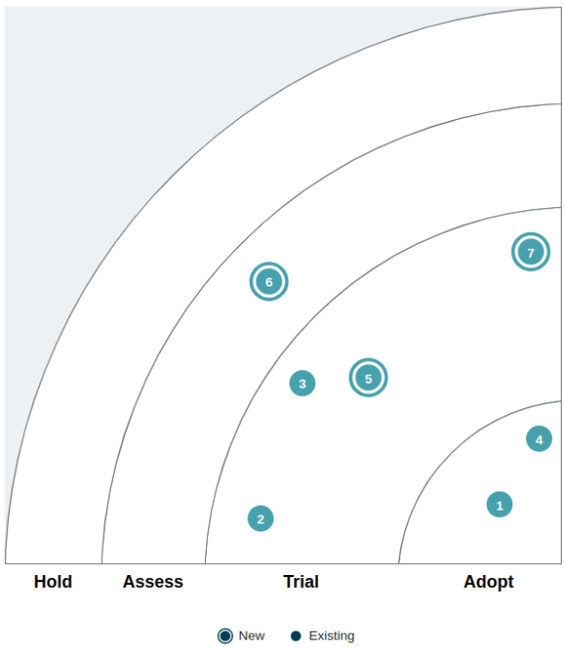
1. GitOps
4. 15-factor app

Trial

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

Assess

6. AI Agents



15-factor app

ADOPT

In 2011 Heroku published a set of principles for developing SaaS (Software as a Service) applications. This set is known as the 12-factor app [https://12factor.net]. Aside from providing suggestions for the development of services and managing software dependencies, the factors focused heavily on building cloud-agnostic applications.

The 15-factor app [https://developer.ibm.com/articles/15-factor-applications/] adds 3 principles to improve quality and maintainability of cloud native applications.

- **API-first:** Cloud native applications run in ecosystems with other services, managed services and facilities from the cloud environment. By focusing on well-defined APIs challenges in integration can be prevented.
- **Telemetry:** In the original 12-factor app "logging" was already a factor. Logging monitors the internal health of the applications. Telemetry is added to also watch the health of the application in its environment; monitoring resource usage, scaling, integration with other application and integration with the cloud environment.
- **Authentication and Authorization:** Cloud native applications are easily distributed over datacenters around the globe or even different clouds. The reach of such apps is enormous and both the number and the different types of consumers increase the complexity regarding Security. Authentication and Authorization draws attention to these concerns during development.

We have put the 15-factor app in *Trial*. The 12-factor app already provides a good list of principles for developing cloud native applications, but the 15-factor app adds 3 extra factors which add a lot of value in modern cloud native applications.

GitOps

ADOPT

GitOps [https://about.gitlab.com/topics/gitops/] is a technique that tries to enhance DevOps automation with the help of best practices within software development, like version control, compliance, collaboration and CI/CD.
Simply put, the technique exists of the usage of Git and Infrastructure as Code, in collaboration with a CI/CD pipeline for everything. This will include **every** change of configuration of a running system.

Git [https://git-scm.com/] tracks every change in the Git history which creates an audit trail. Also, there is a central hub for collaboration through merge requests to implement changes in a compliant way.
Infrastructure as Code [https://en.wikipedia.org/wiki/Infrastructure_as_code] will take care of all infrastructure configuration via code. Combined with Git, it provides centralized place that describes your infrastructure.
Git and Infrastructure as Code combined with a CI/CD pipelne will enable automatic deployments of everything without the need of any human intervention.

Currently, we see GitOps being used by many companies, and it is considered both pleasant and self-evident. We therefore recommend embracing GitOps, and we have included it in our technology radar in *Adopt*.

Embrace Shadow IT

TRIAL

Shadow IT is a phenomenon where software developers within an organization build up IT facilities outside official supervision and regulation by the IT department responsible. This typically occurs when IT support fails to facilitate everything that developers need to deliver their software solutions. It may concern unauthorized use of network, storage, and compute resources for running applications, or for CI/CD tooling for the delivery of those applications. Sometimes it's self-built tooling that escapes audit trails, or third party software packages that are accessed via public internet. Shadow IT can lead to a scattered software landscape with blind spots. Required maintenance, security and compliance of crucial parts can remain unknown and unaddressed, which is why management generally regards shadow IT as a risk.

We see a trend where platform teams, born from earlier DevOps insights, are gradually being moved back to separate departments, governed by goals of predictable and stable infrastructure and CI/CD tooling, and a corresponding increase of shadow IT among software developers. JDriven encourages organizations to not excessively restrain shadow IT - where possible, which is why it's on *Trial* on the Tech radar - and to seek to bridge the gap. Developers need insight into the reasons why IT support puts limits on what it facilitates, and IT support should know the developer needs for the growing software landscape and the demands for CI/CD tooling to continue building software. DevOps was the answer to the friction in the software landscape between software developers' agile software delivery and operations' drive to keep that landscape stable and controllable. Similarly, the DevOps mentality should be to embrace shadow IT and let it lead to controlled innovation and sustained effectiveness of IT support.

Event Modeling

TRIAL

Event Modeling [https://eventmodeling.org/] is a method of describing systems using an example of how information has changed within them over time.

An event-centric approach to a business process and describing it in a clear manner helps business, architects, and developers to have a deep understanding about the problem to be solved. Besides an event-centric approach, Event Modeling also takes the user experience into account. This combination results in a complete picture of user interaction with the system and how information flows through the system.

At JDriven we have seen that Event Modeling can be used alongside or instead of Big Picture Event Storming, to develop a common understanding of a business domain. It results in a more concrete picture for software implementations.

Passkeys

TRIAL

While we still have some reservations whether passkeys are generally accepted by the public, we decided to put this technique in *Trial*. We consider passkeys as a technique ready to be offered as an option together with the traditional authentication techniques such as passwords. The idea behind passkeys is that they are secure through the "something you have" principle of the "authentication factors" (https://en.wikipedia.org/wiki/Authentication#Authentication_factors), because the private key doesn't leave the users' device. Compared to passwords, this prevents some vulnerabilities. Because of the human factor, passwords are susceptible to things such as low password strength, reuse, data breaches, social engineering and carelessness. Large IT players, such as Apple and Google have broad adoption for passkeys on their devices, increasing ease of use for the average user. This has made passkeys a realistic option as primary authentication.

If you are considering adoption of passkeys as a developer, take a look at Hanko.io's website that demonstrates the benefits of passkeys: https://www.passkeys.io/.

Of course, be diligent in defining the account recovery process if you want to adopt passkeys. Give users enough opportunity to alter their authentication through familiar processes.

Shape Up

TRIAL

Shape Up [https://basecamp.com/shapeup] describes a way of working to "shape" a solution so it fits within one iteration.

Iterative working is a common way of working within many projects, whereby each iteration consists of an estimated amount of work, usually expressed in Story Points. Shape Up turns this approach the other way around, by "shaping" a solution so it fits within one iteration. Shape Up describes a solution on a high level, without too many details, which must be complete and must have concrete boundaries. High level means a user experience design must be a sketch, complete means pitfalls are taken away and concrete boundaries means there's a clear starting and ending of the solution.

If the solution does not fit into one iteration, it needs to be shaped in such a way that it will fit in one iteration and still has value when delivered.

AI Agents

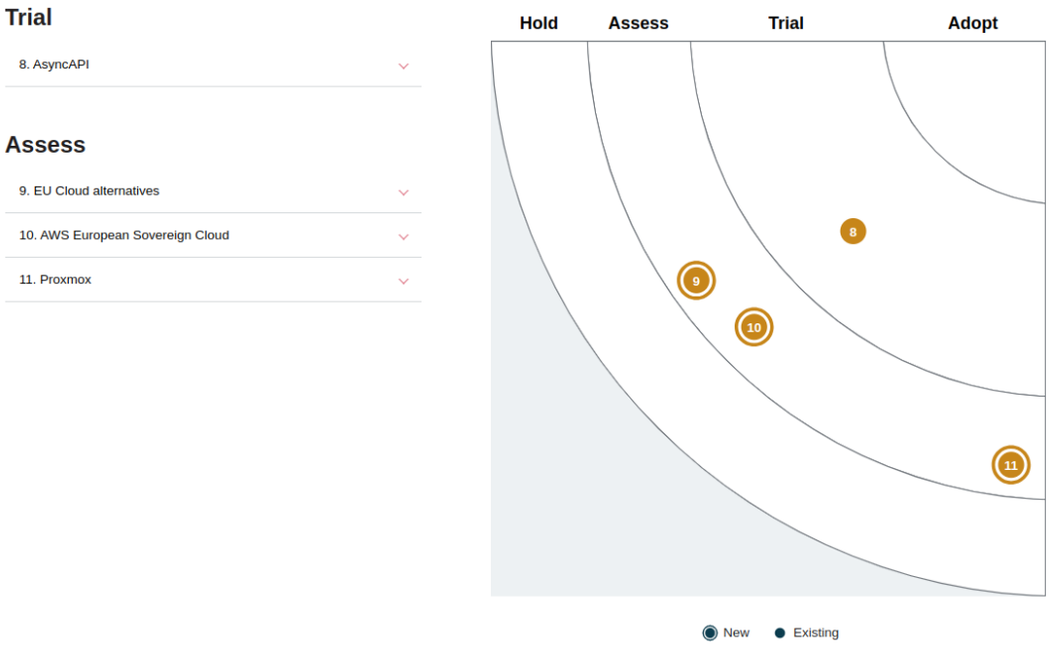
ASSESS

LLMs provide the capability to interact with a computer in a language that is natural to humans. Being able to pose a question or define a task via a prompt written in natural language might be the biggest game changer since the internet. Yet, they don't interact with the real world but take an input and produce some output. Modern AI Agents [https://www.ibm.com/think/topics/ai-agents] are programs that use LLMs to autonomously reason which sequence of actions is necessary to complete its goal. They are not restricted to a particular script and can dynamically utilize the results of previous actions to select the subsequent action.

What would this look like in practice? Imagine a digital assistant for an e-commerce platform. With this agent, instead of having to fill in a form, the user can ask, "What is the status of my order?". The assistant can reason that it should retrieve the status by querying the order database. Moreover, the agent can also infer that it must first request the order id from the customer.

At the same time, AI agents are still in their early stages. While the underlying models are powerful, they can be prone to errors. Reliable deployment requires careful attention to robustness, security, and ethics. Still, the potential for practical applications is already significant enough to warrant active experimentation — which is why we're placing AI agents in the *Assess* ring.

Platforms



AsyncAPI

TRIAL

The AsyncAPI specification [https://www.asyncapi.com/en], akin to the OpenAPI specification for RESTful services, provides a standardized way to describe event-driven and message-driven APIs. This specification facilitates the design, documentation, and consumption of asynchronous APIs, enabling developers to define message formats, channels, and publish-subscribe patterns in a language-agnostic manner. As organizations increasingly adopt event-driven architectures, the Async API specification is a good tool for maintaining consistency and interoperability across distributed systems. The AsyncAPI provides a standard and we can generate documentation and code from it.

For projects that use event-driven APIs the AsyncAPI specification may be good fit, and we recommend assessing it.

Dependency on US services

Via the General Data Protection Regulation (GDPR) the EU regulates how companies collect and process data related to EU citizens. However, the American FISA702 law and Executive Order 12.333 allow the US government to force any American company, or even an international company with an office in the US, to ignore the GDPR and allow it access to that data, even if the servers are not on US soil. An additional agreement called TADPF was put in place between the EU and the US to help safeguard data privacy, guarded in the US by the Privacy and Civil Liberties Oversight Board (PCLOB). The recent capricious American decisions surrounding the PCLOB are cause for concern among European companies that fear, that these safeguards for their data and services on American service providers may prove to be insufficient.

AWS European Sovereign Cloud

ASSESS

To address the concerns of European companies with regard to PCLOB and GDPR, AWS has announced AWS European Sovereign Cloud [https://aws.amazon.com/compliance/europe-digital-sovereignty/], 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 scheduled to be in Germany by the end of 2025. AWS offers an FAQ and whitepapers on its accompanying website to explain 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.

For any company seeking a way to remain compliant with GDPR without having to write off investments in AWS knowledge and implementation, JDriven recommends assessing whether the AWS European Sovereign Cloud is a safe alternative. It's important to note that the AWS European Sovereign Cloud service offering is considerably smaller than that of the regular AWS Cloud. Besides that, in their terms of use AWS mentions that they are bound by law to comply with legal requests for data insight, which raises the question if it makes any difference that AWS European Sovereign Cloud has European locations and employees, while the parent company is American, bound by American law. These are things that warrant a status of *Assess* on the radar.

EU Cloud alternatives

ASSESS

Over a decade ago, several American companies like Amazon, Google and Microsoft decided to expand their server-parks and rent them out to other companies. Platform-as-a-service (PaaS), infrastructure-as-a-service (IaaS) and simply "the cloud" were born. As the years progressed, cloud providers expanded and added "cloud native" services: for example, you no longer install a database on a rented server, but you rent a database service, which is maintained by the cloud provider. This has helped organizations with cost-efficient scalable infrastructure, and has also allowed them to focus on their unique core competitive capabilities. Software consultants however have been warning for vendor lock-in: cloud providers want to lure customers with their unique service offering, but using those unique services complicates efforts to depart from that cloud provider later.

In recent months it has become apparent that there's an additional aspect to vendor lock-in to consider. American cloud providers need to comply with a government that has been capricious or even hostile in their foreign relations, and has entered into a trade war with European countries. European organizations should therefor wonder, how vulnerable their software and data are when hosted with American cloud providers. Cloud-agnostic platforms like Kubernetes can help decouple the software architecture and underlying infrastructure, mitigating the risk to some extent. But JDriven thinks its time organizations look at the various European cloud providers and consider to what extent the service offerings matches their software and infrastructure needs. Examples of European cloud providers are Cyso Cloud, OVH Cloud, Exoscale and Hetzner.

Proxmox

ASSESS

Proxmox [https://www.proxmox.com/en/] offers an alternative to the commonly-used VMWare platform. The Proxmox Virtual Environment (PVE) is an open source virtualisation platform for running containers and/or virtual machines with varying operating systems. The PVE supports high-availability over multiple servers, and includes resource-efficient backup functionality.

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 have placed Proxmox in *Assess* because the applicability is entirely dependent on the situation. The gains in lower (licence) costs and increased control must be considered against the loss of scalability and available support.

Tools

Trial

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

Assess

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

Adopt

16. IntelliJ HTTP client

IntelliJ HTTP client

ADOPT

The IntelliJ HTTP Client is an integrated feature within IntelliJ IDEA that enables developers to create and execute HTTP requests directly from the IDE. It uses `.http` or `.rest` files in which HTTP requests can be defined in a straightforward manner. This allows for fast and efficient API testing without relying on external tools like Postman.

The HTTP Client supports the use of variables, scripting, and various authentication methods, making it suitable for both simple and advanced use cases. Responses are clearly displayed and saved alongside the request files, which simplifies analysis and debugging.

A key advantage is that request files can be stored within the project structure, alongside the source code. This enhances reusability, version control, and collaboration within teams, and contributes to better-documented and reproducible API development. Since IntelliJ is a widely used IDE, it is a logical choice for developers already familiar with the environment who prefer not to use a separate tool for API testing.

While the client has been available for some time, recent updates have introduced robust new features, including support for environment files, secure handling of authentication, OAuth2 flows, and variable usage. For developers already working in IntelliJ, it offers all the necessary functionality to develop and test APIs, without ever leaving the IDE.

Bruno

TRIAL

Bruno [https://www.usebruno.com] is an open-source API client similar to well-known alternatives like Postman [https://www.postman.com/], Insomnia [https://insomnia.rest/] and Hopscotch [https://hoppscotch.io/]. According to Bruno's manifesto [https://www.usebruno.com/manifesto], API collections should be co-located within the source code repository, serving as a living set of examples on how to use the API, and with that turning it into "living documentation". Collections become first-class citizens, co-located with related information and easily version controlled.

Bruno's UI is comparable to Postman, offering a known environment if you are used to other API clients. It supports obviously various HTTP methods, request body formats, and response parsing. Bruno has plugins for Visual Studio Code, and support for other IDEs are in development. Key features include environment variables, collection variables, and the ability to run pre-request and post-request scripts.

From a developer's perspective, Bruno is a fresh alternative to existing commercial products, without needing a required account and storing and sharing collections in a cloud environment. By placing API collections together with source code, separate import/export processes are not needed as well. This automatically improves collaboration within development teams, and documentation evolves with the codebase.

While Bruno lacks some advanced features found in the more well-known API platforms, its focus on ease of use and integration with development workflows makes it an interesting choice. And because Bruno is open-source, community-driven improvements and customizations are also possible (let's hope the IntelliJ plugin is available at the moment you are reading this). We would definitely advise you to give Bruno a try, which is why we have added Bruno in Trial.

Devoxx Genie

TRIAL

Devoxx Genie is a fully Java-based large language model (LLM) Code Assistant plugin for IntelliJ IDEA, designed to integrate with local LLM-providers. Some well-known tools are already supported: Ollama, LMStudio, GPT4All, Llama.cpp and Exo. It can also connect to cloud-based LLMs (OpenAI, Anthropic, Mistral, Groq, Gemini, DeepInfra, DeepSeek, and OpenRouter). Some of the key features are a local chat history, a token cost calculator, web search and code highlighting. The flagship feature is the ability to add your whole project or package to the context of your prompt when asking questions to an LLM of your choice.

Running online LLMs usually involves sending data to servers managed by these LLM providers. At the projects where we as JDriven are involved in, cloud-based AI coding assistants are hardly used because there is insufficient control over what happens to sensitive data. That is why we put Devoxx Genie in the *Trial* ring. The ability to still leverage the power of coding assistants locally, within your IDE, is something many developers want. We believe that, while Devoxx Genie is still in development and has its quirks, it is a good substitute for cloud-based solutions.

Nushell

TRIAL

Nushell [https://www.nushell.sh] is a new type of shell. The goal of Nushell is to take the Unix philosophy of shells, where pipes connect simple commands together, and combine it with a rich programming language. Making Nushell both a programming language and shell in one package, that runs on Linux, macOS, and Windows. Nushell connects both by bringing a rich programming language and a full-featured shell together into one package. Code written in Nushell can be run on Linux, macOS, and Windows. Nushell uses structured data everywhere, for example output of `ls` command, or REST API calls. Common commands to query, sort, filter and transform data are available and can be combined with each other. For example to find files larger than 10 MB and sort them by last modified: `ls | where size >= 10mb | sort-by modified`.

Nushell has been around for a couple of years, but it is still in the `0.x` release stage. Therefore, it is very interesting to play around with it and see what the advantages and disadvantages are compared to other shells and tools.

warp.dev

TRIAL

Warp [https://www.warp.dev/] is a terminal application that enhances the command-line interface experience for developers. The application provides nice command-line completion, an integrated command palette, and a block-based output system for improved result visualization and interaction. But the most interesting feature is the ability to use natural language to generate shell commands using artificial intelligence. Also error output from a shell command can be input for the AI to come up with a solution.

Warp provides a free plan with a limited number of AI calls per month. To have unlimited AI calls, you can upgrade to the paid plan. Warp also offers an option to pay for a team, with extra options to collaborate with other developers. An enterprise version is also available, where it is possible to use a private LLM.

The shell is a very powerful tool for developers, but it can be hard to remember all the commands and options. Warp makes the shell easier to use by providing a natural language interface. A developer doesn't have to leave the shell to get all the information they need, which is very powerful. As it is still in beta, we've added it to *Assess* in our Tech Radar.

CRaC

ASSESS

CRaC [https://openjdk.org/projects/crac/], Coordinated Restore at Checkpoint makes it possible to take a snapshot of a running JVM and store it to disk. This snapshot can then be used to start a JVM, resulting in massive startup performance gain. It offers an alternative to native compilation in terms of startup time but retains the flexibility of the JVM, which could be a middle ground for some more complex applications. The Spring Framework [https://docs.spring.io/spring-framework/reference/integration/checkpoint-restore.html] saw the potential and has fully integrated with CRaC. You do have to take into account that any value (secrets and such) seen by the JVM will be included in the snapshot.

We think it's a promising addition to the JVM which can provide quick gains to startup times for more complex microservices with big variation in load, and therefore we put CRaC in the *Assess* ring.

Cursor IDE

ASSESS

Cursor IDE [https://www.cursor.com/] is an AI code editor which is based on Visual Studio Code (VS Code). It offers integrated AI, so that AI support is embedded in the development process. It has different features like code completion, code generation, code review, etc.

We've been seeing speakers at various recent conferences use Cursor, and it's an interesting alternative for VS Code. Instead of having to rely on AI-plugins, Cursor distinguishes itself as an IDE with integrated AI support. JDriven sees the benefits for developer workflow and productivity by having an AI help with peripheral matters. There is a learning curve before being able to use it efficiently, however. And we notice that so-called *vibe coding* in general can cause a developer to have to spend more time reviewing and correcting AI-generated code, than on creating and understanding solutions. This is a risk, as developers remain responsible for the code that ends up in production. Requests to the LLMs involve credits and waiting times, which can be circumvented by pay-per-use. Either way, it's important to note that requests are sent to Cursor servers, which has privacy and security implications. All things considered we're placing Cursor IDE on *Assess*.

DeepSeek

ASSESS

DeepSeek [https://tweakers.net/nieuws/231566/nederlandse-ambtenaren-mogen-deepseek-niet-gebruiken-van-kabinet.html] is an alternative LLM to ChatGPT that can be used for various tasks, including code assistance. Its reasoning model and training allow it to perform accurately even with fewer resources, making it part of a new generation of efficient open-source LLMs that can run locally while still delivering strong performance.

Despite its potential, DeepSeek is not without controversy. The legitimacy of its data acquisition raises concerns, similar to ChatGPT, and there are also worries about potential Chinese censorship. We have seen that many of our clients prefer not to use cloud-based LLMs due to the risk of their data being shared. A locally running LLM like DeepSeek addresses this concern by providing assistance without sending information externally.

When combined with tools like Devoxx Genie, DeepSeek can offer in-IDE suggestions for code. However, its capabilities extend beyond coding, making it a versatile AI tool.

We place DeepSeek in *Assess*, as it can be used locally and presents an interesting alternative, but it still requires further investigation.

JetBrains AI

ASSESS

JetBrains [https://www.jetbrains.com/ai/] has introduced two AI technologies: Mellum (October 2024) and Junie (January 2025). Mellum is a specialized LLM optimized for code completion with support for Java, Kotlin, Python en Go. It has 3x faster response time and 40% acceptance rate. Junie is an AI coding agent within the JetBrains Pro and Ultimate line that can independently solve 53.6% (according to SWEBench) of development tasks. While Mellum functions as an intelligent code completer, Junie is a semi-autonomous assistant that can complete more complex tasks such as generating code, running inspections, and writing tests. Both tools are designed with privacy as a priority – Mellum is trained on publicly available, permissively licensed code – and give developers control over proposed changes. JetBrains' vision with these tools is to make software development more productive and enjoyable by combining immediate assistance (Mellum) with an AI partner that can fundamentally change the development workflow (Junie).

For a developer, designing and crafting solutions is often the enjoyable part of our work. We already spend quite some time reading and understanding code written by others (including AI), and not being actively involved in the solution process may make this part of the job more difficult. It is important to evaluate whether these technologies actually address the right problems, such as reducing boilerplate code and providing an efficient alternative to StackOverflow. It is also worth mentioning that this cloud completion tooling sends your code to JetBrains' servers, which may be a dealbreaker for some organizations. Additionally, it's good to remember that JetBrains AI encompasses more than just these two technologies. JetBrains AI Assistant, with its current code completion and code generation options, is still in active development and also supports locally running LLMs (just like Devoxx Genie, see elsewhere in this radar). This is why we have put Mellum and Junie in the *Assess* ring.

Commit. Develop. Share.

Commit.
Develop.
Share.