ThoughtWorks®

TECHNOLOGY RADAR VOL. 20

An opinionated guide to technology frontiers

thoughtworks.com/radar
#TWTechRadar

CONTRIBUTORS

The Technology Radar is prepared by the ThoughtWorks Technology Advisory Board

This edition of the ThoughtWorks Technology Radar is based on a meeting of the Technology Advisory Board in Shenzhen in March 2019

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Jonny LeRoy

Ketan Padegaonkar

Lakshminarasimhan Sudarshan

Marco Valtas

Mike Mason

Neal Ford

Ni Wang

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

ABOUT THE RADAR

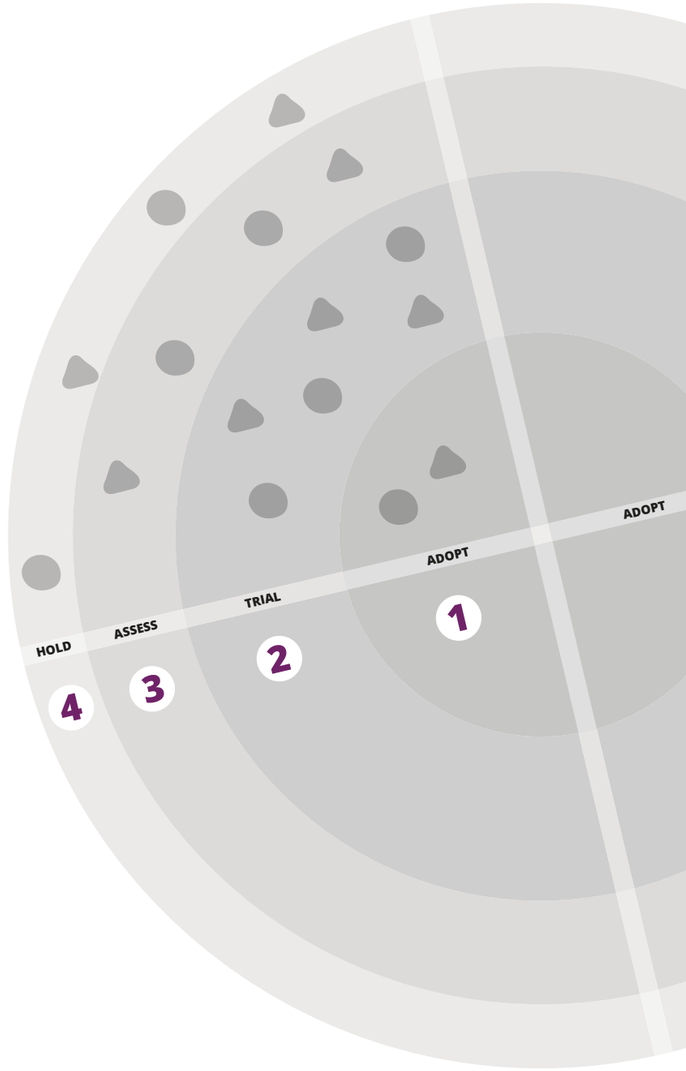
ThoughtWorkers are passionate about technology. We build it, research it, test it, open source it, write about it, and constantly aim to improve it — for everyone. Our mission is to champion software excellence and revolutionize IT. We create and share the ThoughtWorks Technology Radar in support of that mission. The ThoughtWorks Technology Advisory Board, a group of senior technology leaders in ThoughtWorks, creates the Radar. They meet regularly to discuss the global technology strategy for ThoughtWorks and the technology trends that significantly impact our industry.

The Radar captures the output of the Technology Advisory Board's discussions in a format that provides value to a wide range of stakeholders, from developers to CTOs. The content is intended as a concise summary.

We encourage you to explore these technologies for more detail. The Radar is graphical in nature, grouping items into techniques, tools, platforms, and languages & frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings to reflect our current position on them.

For more background on the Radar, see thoughtworks.com/radar/faq

RADAR AT A GLANCE



1 ADOPT

We feel strongly that the industry should be adopting these items. We use them when appropriate on our projects.

2 TRIAL

Worth pursuing. It's important to understand how to build up this capability. Enterprises can try this technology on a project that can handle the risk.

3 ASSESS

Worth exploring with the goal of understanding how it will affect your enterprise.

4 HOLD

Proceed with caution.

▲ NEW OR CHANGED
● NO CHANGE

Items that are new or have had significant changes since the last Radar are represented as triangles, while items that have not changed are represented as circles.

! Our Radar is forward looking. To make room for new items, we fade items that haven't moved recently, which isn't a reflection on their value but rather on our limited Radar real estate.

WHAT'S NEW

Highlighted themes in this edition:

The Shifting Shape of Data

A decade ago, *data* was synonymous with relational databases. Now, *data* can take on a staggering variety of shapes, including NoSQL, time series, SQL stores such as CockRoachDB and Spanner that offer global consistency as well as event streams that offer querying capabilities to aggregated log files. This is driven by businesses' desires for ever more real-time responses to increasingly larger, more varied and faster data sources. For developers, understanding the trade-offs inherent with each flavor of data usage can pose challenges. Architects and developers should be on the lookout for new capabilities offered by tools and paradigms while remaining diligent that they don't misuse new tools by treating them too much like familiar ones. We should embrace the fact that we're in the middle of a major shift in the data landscape and still searching for the proper strategies and tools.

Terraforming an Ecosystem

Developers love abstraction layers, for obvious reasons: by encapsulating complexity into an abstraction they can focus on higher-level concerns. We've seen this evolution across several editions of the Radar in the way teams deal with the intersections of clouds and containers. First, the action focused on Docker and its ecosystem. Then, focus shifted down the stack to Kubernetes. Now, the main activity we see is on infrastructure as code in general and the Terraform ecosystem in particular. Although we've recommended tools beyond Terraform, its adoption in the community of providers has been impressive. Highlights in this Radar include Terratest for testing infrastructure code and GoCD's new Provider, which lets you configure GoCD using Terraform.

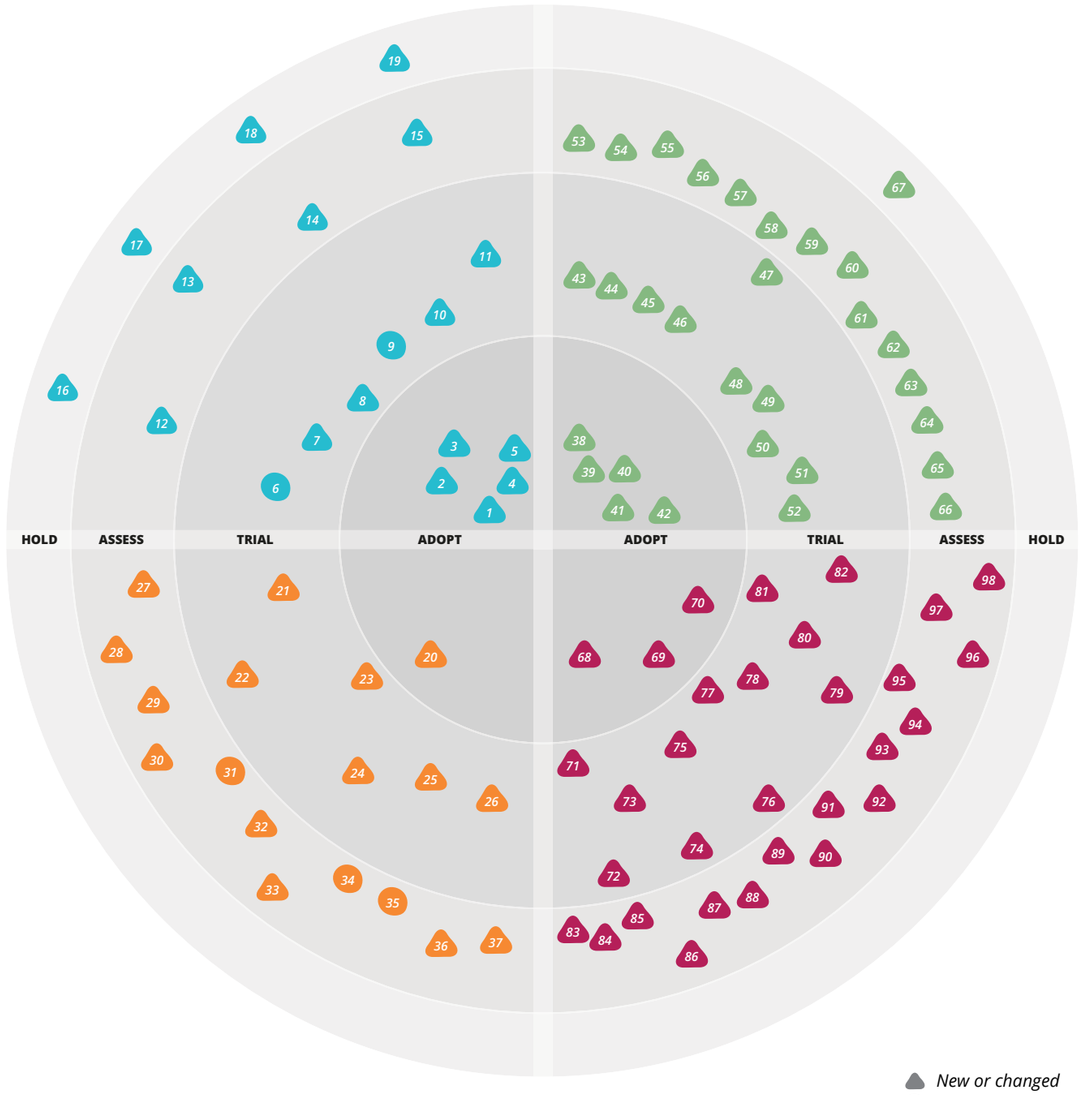
Kotlin Klimbing

Kotlin, the open-source language, continues to receive strong representation in our Radar as it expands beyond its Android stronghold. Created internally at JetBrains, because they didn't like the options available in the existing language landscape, and subsequently released under an open-source license, Kotlin seems to resonate with developers across a wide spectrum. It keeps appearing across platforms and tools as a general and special-purpose development language, and increasingly also in our Radar and on our project teams (e.g., Ktor, MockK, Detekt, HTTP4K). It's refreshing to see that pragmatic design, state-of-the-art tooling, and a burgeoning ecosystem allow an upstart language to succeed.

Leaking Encapsulation Boundaries

With the advent of "everything as code," almost everything — infrastructure, security, compliance and operations — formerly difficult to change becomes programmatically addressable, meaning that developers could apply good engineering practices; yet all too often we see either highly complex configuration subsystems or over-reliance on visual orchestration tools — logic creeping into configuration files, horrible syntax for conditionals in YAML, and the many orchestration frameworks we saw across a wide variety of technologies. With the advent of polyglot programming, infrastructure as code and *x*-as-a-service, teams end up with varying components that merge into a single cohesive system. Thus, logic that should reside within a system boundary leaks out into orchestration tools, configuration files, and other plumbing. While this is sometimes necessary, we encourage teams to carefully consider keeping such code in places where developers adhere to testing, version control, continuous integration, and other good engineering practices. Avoid adding business logic in configuration files (and avoid tools that require it), and try to keep necessary orchestration to a minimum rather than a dominant feature of your system.

THE RADAR



TECHNIQUES

ADOPT
1. Four key metrics
2. Micro frontends
3. Opinionated and automated code formatting
4. Polyglot programming
5. Secrets as a service

TRIAL
6. Chaos Engineering
7. Container security scanning
8. Continuous delivery for machine learning (CD4ML) models
9. Crypto shredding
10. Infrastructure configuration scanner
11. Service mesh

ASSESS
12. Ethical OS
13. Smart contracts
14. Transfer learning for NLP
15. Wardley mapping

HOLD
16. Productionizing Jupyter Notebooks
17. Puncturing encapsulation with change data capture
18. Release train
19. Templating in YAML

PLATFORMS

ADOPT
20. Contentful

TRIAL
21. AWS Fargate
22. EVM beyond Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

ASSESS
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

HOLD

TOOLS

ADOPT
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. UI dev environments

TRIAL
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

ASSESS
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Terraform provider GoCD
66. Terratest

HOLD
67. Handwritten CloudFormation

LANGUAGES & FRAMEWORKS

ADOPT
68. Apollo
69. MockK
70. TypeScript

TRIAL
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

ASSESS
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

HOLD

TECHNIQUES

Four key metrics
ADOPT

The thorough State of DevOps reports have focused on data-driven and statistical analysis of high-performing organizations. The result of this multiyear research, published in Accelerate, demonstrates a direct link between organizational performance and software delivery performance. The researchers have determined that only four key metrics differentiate between low, medium and high performers: lead time, deployment frequency, mean time to restore (MTTR) and change fail percentage. Indeed, we've found that these four key metrics are a simple and yet powerful tool to help leaders and teams focus on measuring and improving what matters. A good place to start is to instrument the build pipelines so you can capture the four key metrics and make the software delivery value stream visible. GoCD pipelines, for example, provide the ability to measure these four key metrics as a first-class citizen of the GoCD analytics.

Micro frontends
ADOPT

We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a frontend monolith — a large, entangled browser application that sits on top of the backend services — largely neutralizing the benefits of microservices. Since we first described micro frontends as a technique to address this issue, we've had almost universally positive experiences with the approach and have found a number of patterns to use micro frontends even as more and more code shifts from the server to the web browser. So far, web components have been elusive in this field, though.

Opinionated and automated code formatting
ADOPT

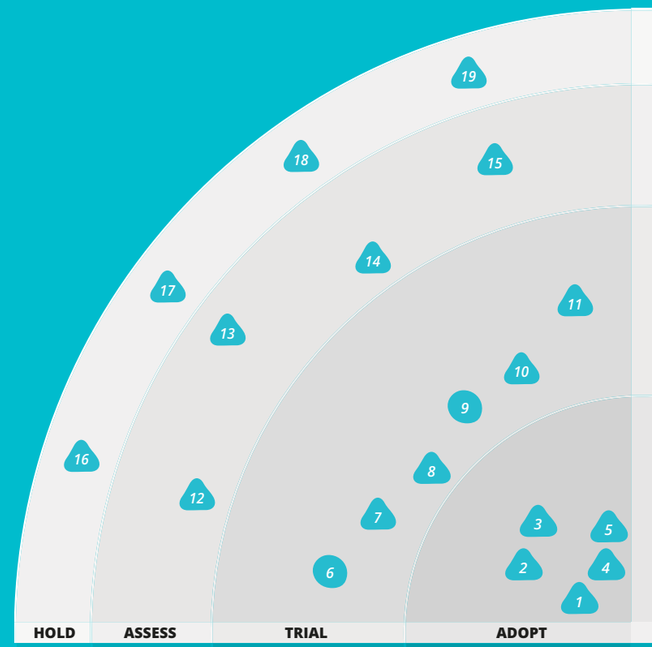
For as long as we can remember, what style to use for formatting code has been a matter of personal taste, company policy and heated debate. Finally, the industry appears to be tiring of this endless argument and teams are freeing up surprisingly large amounts of time by forgoing these discussions and just adopting opinionated and automated code formatting tools. Even if you don't agree 100% with the opinions of the various tools, the benefits of focusing on what your code does rather than how it looks is something most teams should be able to get behind. Prettier has been getting our vote for JavaScript, but similar tools, such as Black for Python, are available for many other languages and are increasingly being built-in as we see with Golang and Elixir. The key here is not to spend hours discussing which rules to enforce, but instead pick a tool that is opinionated, minimally configurable and automated — ideally as a pre-commit hook.

ADOPT
1. Four key metrics
2. Micro frontends
3. Opinionated and automated code formatting
4. Polyglot programming
5. Secrets as a service

TRIAL
6. Chaos Engineering
7. Container security scanning
8. Continuous delivery for machine learning (CD4ML) models
9. Crypto shredding
10. Infrastructure configuration scanner
11. Service mesh

ASSESS
12. Ethical OS
13. Smart contracts
14. Transfer learning for NLP
15. Wardley mapping

HOLD
16. Productionizing Jupyter Notebooks
17. Puncturing encapsulation with change data capture
18. Release train
19. Templating in YAML



TECHNIQUES

Choosing the right language for the job can significantly boost productivity, and supporting a language ecosystem is important to get systems live quickly and give developers tools to properly approach problems.

(Polyglot programming)

Traditional machine learning model development tends to have long cycle times between training models and actually deploying them to production, and we often end up with production models trained with (now) stale data. Applying continuous delivery practices to machine learning can address both of these issues.

(Continuous delivery for machine learning (CD4ML) models)

Polyglot programming

ADOPT

We put polyglot programming on Trial in one of our first Radars to suggest that choosing the right language for the job could significantly boost productivity, and there were new language entrants that were worthy of consideration. We want to reraise this suggestion because we're seeing a new push to standardize language stacks by both developers and enterprises. While we acknowledge that placing no restrictions on language uses can create more problems than it solves, promoting a few languages that support different ecosystems or language features is important for both enterprises to accelerate processes and go live more quickly and developers to have the right tools to solve the problem at hand.

Secrets as a service

ADOPT

Humans and machines use secrets throughout the value stream of building and operating software. The build pipelines need secrets to interface with secure infrastructures such as container registries, the applications use API keys as secrets to get access to business capabilities, and the service-to-service communications are secured using certificates and keys as secrets. You can set and retrieve these secrets in different ways. We've long cautioned developers about using source code management for storing secrets. We've recommended decoupling secret management from source code and using tools such as git-secrets and Talisman to avoid storing secrets in the source code. We've been using secrets as a service as a default technique for storing and accessing secrets. With this technique you can use tools such as Vault or AWS Key Management Service (KMS) to read/write secrets over an HTTPS endpoint with fine-grained levels of access control. Secrets as a service uses external identity providers such as AWS IAM to identify the actors who request access to secrets. Actors authenticate themselves with the secrets service. For this process to work, it's important to automate bootstrapping the identity of the actors, services and applications. Platforms based on SPIFFE have improved the automation of assigning identities to services.

Chaos Engineering

TRIAL

In the last year we've seen Chaos Engineering move from a much talked-about idea to an accepted, mainstream approach to improving and assuring distributed system resilience. As organizations large and small begin to implement Chaos Engineering as an operational process, we're learning how to apply these techniques safely at scale. The approach is definitely not for everyone, and to be effective and safe, it requires organizational support at scale. Industry acceptance and available expertise will definitely increase with the appearance of commercial services such as Gremlin and deployment tools such as Spinnaker implementing some Chaos Engineering tools.

Container security scanning

TRIAL

The container revolution around Docker has massively reduced the friction in moving applications between environments, fueling increased adoption of continuous delivery and continuous deployments. The latter, especially, has blown a rather large hole in the traditional controls over what can go to production. The technique of container security scanning is a necessary response to this threat vector. Tools in the build pipeline automatically check containers flowing through the pipeline against known vulnerabilities. Since our first mention of this technique, the tool landscape has matured and the technique has proven useful on development efforts with our clients.

Continuous delivery for machine learning (CD4ML) models

TRIAL

Continuous delivery for machine learning (CD4ML) models apply continuous delivery practices to developing machine learning models so that they are always ready for production. This technique addresses two main problems of traditional machine learning model development: long cycle time between training models and deploying them to production, which often includes manually converting the model to production-ready code; and using production models that had been trained with stale data.

A continuous delivery pipeline of a machine learning model has two triggers: (1) changes to the structure of the model and (2) changes to the training and test data sets. For this to work we need to both version the data sets and the model's source code. The pipeline often includes steps such as testing the model against the test data set, applying automatic conversion of the model (if necessary) with tools such as H2O, and deploying the model to production to deliver value.

Crypto shredding

TRIAL

Maintaining proper control over sensitive data is difficult, especially when it's copied outside of a master system of record for backup and recovery purposes. Crypto shredding is the practice of rendering sensitive data unreadable by deliberately overwriting or deleting encryption keys

used to secure that data. Considering there are systems, such as audit application or blockchain, that should not or could not delete historical records, this technique is quite useful for privacy protection and GDPR compliance.

Infrastructure configuration scanner

TRIAL

For some time now we've recommended that delivery teams take ownership of their entire stack, including infrastructure. This means increased responsibility in the delivery team itself for configuring the infrastructure in a safe, secure and compliant way. When adopting cloud strategies, most organizations default to a tightly locked-down and centrally managed configuration to reduce risk, but this also creates substantial productivity bottlenecks. An alternative approach is to allow teams to manage their own configuration and use an infrastructure configuration scanner to ensure the configuration is safe and secure. Options include open-source scanners such as prowler for AWS and kube-bench for Kubernetes installations. For more continuous detection, take a look at cloud platforms such as AWS Config Rules among other commercial services.

Service mesh

TRIAL

For some time now we've recommended Service mesh is an approach to operating a secure, fast and reliable microservices ecosystem. It has been an important stepping stone in making it easier to adopt microservices at scale. It offers discovery, security, tracing, monitoring and failure handling. It provides these cross-functional capabilities without the need for a shared asset such as an API gateway or baking libraries into each service. A typical implementation involves lightweight reverse-proxy processes, aka sidecars, deployed alongside each service process in a separate container. Sidecars intercept the inbound and outbound traffic of each service and provide cross-functional capabilities mentioned above. This approach has relieved the distributed service teams from building and updating the capabilities that the mesh offers as code in their services. This has led to an even easier adoption of polyglot programming in a microservices ecosystem. Our teams have been successfully using this approach with open source projects such as Istio and we will continue to monitor other open service mesh implementations such as Linkerd closely.

Ethical OS

ASSESS

As developers at ThoughtWorks we're acutely aware of the ethics of the work we do. As society becomes ever more reliant on technology, it's important that we consider ethics when making decisions as software development teams. Several toolkits have emerged that can help us think through some of the future implications of the software we're building. They include Tarot Cards of Tech and Ethical OS, which we've had good feedback on. Ethical OS is a thinking framework and a set of tools that drive discussions around the ethics of building software. The framework is a collaboration between the Institute for the Future and the Tech and Society Solutions Lab. It's based on a practical set of risk zones, such as addiction and the dopamine economy, plus a number of scenarios to drive conversation and discussion.

Smart contracts

ASSESS

The more experience we gain with using distributed ledger technologies (DLTs), the more we encounter the rough edges around the current state of smart contracts. Committing automated, irrefutable, irreversible contracts on ledger sounds great in theory. The problems arise when you consider how to use modern software delivery techniques to developing them, as well as the differences between implementations. Immutable data is one thing, but immutable business logic is something else entirely! It's really important to think about whether to include logic in a smart contract. We've also found very different operational characteristics between different implementations. For example, even though contracts can evolve, different platforms support this evolution to a greater or lesser extent. Our advice is to think long and hard before committing business logic to a smart contract and to weigh the merits of the different platforms before you do.

Transfer learning for NLP

ASSESS

Transfer learning has been quite effective within the field of computer vision, speeding the time to train a model by reusing existing models. Those of us who work in machine learning are excited that the same techniques can be applied to natural language processing (NLP) with the publication of ULMFiT and open source pretrained models and code examples. We think transfer learning for NLP will significantly reduce the effort to create systems dealing with text classification.

TECHNIQUES

As society becomes ever more reliant on technology, it becomes even more important that we consider ethics when taking decisions as software development teams. Ethical OS is a thinking framework and a set of tools that drive discussions around the ethics of building software.

(Ethical OS)

Immutable data is one thing, but immutable business logic is something else entirely — think long and hard before committing business logic to a smart contract.

(Smart contracts)

TECHNIQUES

Using change data capture (CDC) to publish row-level change events and directly consuming these events in other services can lead to fragile integrations.

(Puncturing encapsulation with change data capture)

Although we wholeheartedly endorse discipline around regularly releasing and demoing working software, we've experienced serious drawbacks with this approach over the medium to long term: release trains reinforce temporal coupling around sequencing of changes and can degrade quality as teams rush to complete features.

(Release train)

Wardley mapping

ASSESS

We're usually wary of covering diagrammatic techniques, but Wardley mapping is an interesting approach to start conversations around the evolution of an organization's software estate. At their simplest, they're used to visualize the value chains that exist within an organization, starting with customers' needs and progressively plotting the different capabilities and systems used to deliver on those needs along with the evolution of those capabilities and systems. The value of this technique is the process of collaborating to create the maps rather than the artefact itself. We recommend getting the right people in the room to produce them, and then treat them as living, evolving things rather than a complete artefact.

Productionizing Jupyter Notebooks

HOLD

Jupyter Notebooks have gained in popularity among data scientists who use them for exploratory analyses, early-stage development and knowledge sharing. This rise in popularity has led to the trend of productionizing Jupyter Notebooks, by providing the tools and support to execute them at scale. Although we wouldn't want to discourage anyone from using their tools of choice, we don't recommend using Jupyter Notebooks for building scalable, maintainable and long-lived production code — they lack effective version control, error handling, modularity and extensibility among other basic capabilities required for building scalable, production-ready code. Instead, we encourage developers and data scientists to work together to find solutions that empower data scientists to build production-ready machine learning models using continuous delivery practices with the right programming frameworks. We caution against productionization of Jupyter Notebooks to overcome inefficiencies in continuous delivery pipelines for machine learning, or inadequate automated testing.

Puncturing encapsulation with change data capture

HOLD

Change data capture (CDC) is a very powerful technique for pulling database changes out of a system and performing some actions on that data. One of the most popular ways of doing this is to use the database's transaction log to identify changes and then publish those changes directly onto an event bus that can be consumed by other services. This works very well for use cases such as breaking monoliths into microservices but when used for first-class integration between microservices, this leads to puncturing encapsulation and leaking the source service's data layer into the event contract. We've talked about domain scoped events and other techniques that emphasize the importance of having our events model our domain properly. We're seeing some projects use CDC for publishing row-level change events and directly consuming these events in other services. This puncturing of encapsulation with change data capture can be a slippery slope leading to fragile integrations and we would like to call this out with this blip.

Release train

HOLD

We've seen organizations successfully move from very infrequent releases to a higher cadence by using the release train concept. The release train is a technique for coordinating releases across multiple teams or components that have runtime dependencies. All releases happen on a fixed and reliable schedule regardless of whether all expected features are ready (the train doesn't wait for you — if you miss it you wait for the next one). Although we wholeheartedly endorse discipline around regularly releasing and demoing working software, we've experienced serious drawbacks with the approach over the medium to long term as it reinforces temporal coupling around sequencing of changes and can degrade quality as teams rush to complete features. We prefer to focus on the architectural and organizational approaches necessary to support independent releases. Although the train can be a useful forcing function for speeding up slower teams, we've also seen it as imposing an upper limit on how quickly faster-moving teams can move. We believe that it is a technique that should be approached with a good degree of caution, if at all.

Templating in YAML

HOLD

As infrastructures grow in complexity, so do the configuration files that define them. Tools such as AWS CloudFormation, Kubernetes and Helm expect configuration files in JSON or YAML syntax, presumably in an attempt to make them easy to write and process. However, in most cases, teams quickly reach the point where they have some parts that are similar but not quite the same, for example, when the same service must be deployed in different regions with a slightly different setup. For such cases tools offer templating in YAML (or JSON), which has caused a huge amount of frustration with practitioners. The problem is that the syntax of JSON and YAML requires all sorts of awkward compromises to graft templating features such as conditionals and loops into the files. We recommend using an API from a programming language instead or, when this is not an option, a templating system in a programming language, either a general-purpose language such as Python or something specialized such as Jsonnet.

PLATFORMS

Contentful
ADOPT

Headless content management systems (CMSes) are becoming a common component of digital platforms. Contentful is a modern headless CMS that our teams have successfully integrated into their development workflows. We particularly like its API-first approach and implementation of CMS as code. It supports powerful content modeling primitives as code and content model evolution scripts, which allow it to be treated like other data store schemas and enable evolutionary database design practices to be applied to CMS development. Its robustness and a stream of new features, including a sandbox environment, have impressed our teams further and made Contentful our default choice in this space.

AWS Fargate
TRIAL

AWS Fargate, the docker-as-a-service option on AWS, is now widely available across regions. It's a great solution for situations in which teams want to run Docker containers, because AWS Lambda functions aren't powerful enough, without having to manage EC2 instances or Kubernetes clusters. Our teams report generally positive experiences with Fargate; however, the convenience of this managed service can come at a cost, in financial terms.

EVM beyond Ethereum
TRIAL

Ethereum Virtual Machine (EVM) was originally designed for the Ethereum main network. Nowadays, however, most teams no longer want to reinvent blockchain from scratch; instead, they'd like to take EVM beyond Ethereum. We've seen a lot of blockchain teams choose to fork Ethereum (e.g., Quorum) or implement the EVM spec (e.g., Burrow, Pantheon), adding their own designs. The intention is to not only reuse the Ethereum design but also leverage its ecosystem and developer community. To many developers, the concept of "smart contract" is almost equivalent to a smart contract written in Solidity. Although Ethereum itself has some constraints, the technology around the EVM ecosystem is booming.

InfluxDB
TRIAL

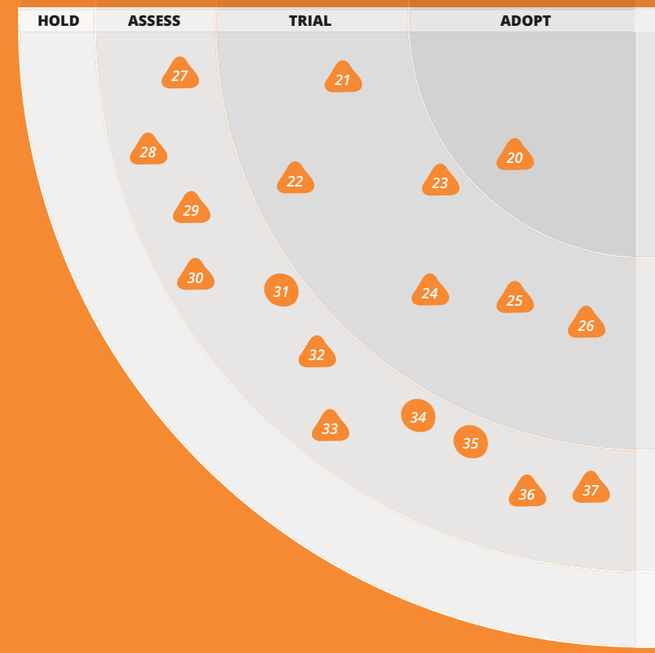
Time series databases (TSDBs) have been around for some time now. But increasingly they're becoming more mainstream as more use cases naturally fit the time series model. InfluxDB continues to remain a good choice for TSDBs with monitoring being one of its key use cases. TICK Stack is an example of a monitoring solution that has InfluxDB at its heart. Influx 2.0 alpha recently introduced Flux — a scripting language for querying and processing time series data. It's still early days for Flux and the jury's out on

ADOPT
20. Contentful

TRIAL
21. AWS Fargate
22. EVM beyond Ethereum
23. InfluxDB
24. Istio
25. Kafka Streams
26. Nomad

ASSESS
27. CloudEvents
28. Cloudflare Workers
29. Deno
30. Hot Chocolate
31. Knative
32. MinIO
33. Prophet
34. Quorum
35. SPIFFE
36. Tendermint
37. TimescaleDB

HOLD



PLATFORMS

Time series databases have been around for some time now, but increasingly they're becoming more mainstream as more use cases naturally fit the time series model. InfluxDB continues to remain a good choice in this space.

(InfluxDB)

Although events are a common FaaS triggering mechanism and every cloud provider supports them in some form, the current proprietary specifications prevent interoperability across clouds. CloudEvents is a burgeoning standard that attempts to address this issue.

(CloudEvents)

its broader adoption beyond InfluxDB, but it promises to be more powerful and expressive than InfluxQL and enables pushing time series analytic workloads to the database. However, clustering support for InfluxDB is only available with the enterprise version which has limited its adoption on some of our projects.

Istio

TRIAL

Istio is becoming the de facto infrastructure to operationalize a microservices ecosystem. Its out-of-the-box implementation of cross-cutting concerns — such as service discovery, service-to-service and origin-to-service security, observability (including telemetry and distributed tracing), rolling releases and resiliency — has been bootstrapping our microservices implementations very quickly. It's the main implementation of the service mesh technique we've been using. We've been enjoying its monthly releases and its continuous improvements with seamless upgrades. We use Istio to bootstrap our projects, starting with observability (tracing and telemetry) and service-to-service security. We're closely watching its improvements to service-to-service authentication everywhere in and outside of the mesh. We'd also like to see Istio establish best practices for configuration files to strike a balance between giving autonomy to service developers and control to the service mesh operators.

Kafka Streams

TRIAL

Kafka Streams is a lightweight library to build streaming applications. It supports basic streaming APIs such as join, filter, map and aggregate as well as local storage for common use cases such as windowing and sessions. Unlike other stream-processing platforms such as Apache Spark and Alpakka Kafka, Kafka Streams has been a good fit for scenarios that don't require large-scale distribution and parallel processing; hence we could get away without yet another piece of infrastructure such as cluster schedulers. Naturally, Kafka Streams has been a good choice when operating in the Kafka ecosystem. Kafka Streams is particularly useful when we have to process data strictly in order and exactly once. One particular use case of Kafka Streams is to build a change data capture (CDC) platform.

Nomad

TRIAL

HashiCorp continues to release interesting software. We've featured HashiCorp Vault in March 2017, and tools related to Terraform are all over this edition of the Radar. We've moved Nomad to Trial because we've had positive experiences using it. While Kubernetes continues to gain traction, we like Nomad's general applicability. It's not just limited to running containerized workloads but can be used to schedule just about anything. Java and Golang are supported natively as well as batch and distributed cron jobs. We like its focus on multi- and hybrid-cloud operations, something likely to become more important to avoid sticky clouds and the fact that it does scheduling well.

CloudEvents

ASSESS

Outside the function code itself, applications written as serverless functions are tightly coupled to the cloud platform on which they're hosted. Although events are a common FaaS-triggering mechanism, and every cloud provider supports them in some form, the current proprietary specifications prevent interoperability across clouds. The CloudEvents specification is a burgeoning standard that has been accepted into the CNCF Sandbox. The standard is still in active development but several language bindings exist and Microsoft has announced first-class support in Azure. We're hoping other cloud providers will follow suit.

Cloudflare Workers

ASSESS

Most modern server-side or serverless code execution platforms are centered around containers or VMs. Cloudflare Workers, however, takes a different approach to hosting a serverless computing offering. It uses V8 Isolates, the open source JavaScript engine developed for Chrome,

to run functions as a service (FaaS) on their extensive CDN network. Code can be written in JavaScript or anything that compiles to WebAssembly and data can be accessed from Cloudflare's cache or key-value store. The major benefit for developers is performance: by being on the edge network, close to end users, cold-starts take only five milliseconds. For the provider the benefits include both the ability to densely pack isolates because of their lower memory overhead and faster performance through reduced process context switching. This is definitely an intriguing approach to monitor and assess.

Deno

ASSESS

As a group we have mixed feelings about programming in JavaScript on the server side, especially when the rationale for doing so is simply to avoid polyglot programming. That said, if you decide to use JavaScript or TypeScript on the server, have a look at Deno. Written by Ryan Dahl, the inventor of Node.js, Deno aims to avoid what Ryan considers mistakes that were made in Node.js. It brings a strict sandbox system and built-in dependency and package management, and it supports TypeScript out of the box. Deno is built using Rust and V8.

Hot Chocolate

ASSESS

The GraphQL ecosystem and community keep growing. Hot Chocolate is a GraphQL server for .NET (core and classic). It lets you build and host schemas and then serve queries against them. The team behind Hot Chocolate has recently added schema stitching which allows for a single entry point to query across multiple schemas aggregated from different locations. Although there are plenty of ways to misuse this approach, it's worth assessing whether to add it to your toolkit.

Knative

ASSESS

The serverless architecture has popularized a FaaS style of programming among developers; it helps developers focus on solving core business problems with independently built and deployed functions that react to an event, run a business process, produce other events in the process and scale down to zero. Historically, proprietary serverless platforms such as AWS Lambda or Microsoft Azure Functions have enabled this programming paradigm. Knative is an open-source Kubernetes-based platform to run FaaS workloads. There are few things that stand out about Knative: it's open source and provider agnostic; it implements the serverless workflow as described in the CNCF Serverless Working Group whitepaper; it ensures cross-service interoperability by implementing its eventing interface consistent with CNCF CloudEvents specification; and, most importantly, it addresses a common challenge of operating a harmonized and yet hybrid FaaS and long-running container-based architecture. It easily integrates with both Istio and Kubernetes. For example, developers can take advantage of roll-out strategies that Istio implements by traffic splitting between different revisions of the functions. Developers can take benefit of Istio-provided observability not only for long-running container services but also for FaaS programs in the same Kubernetes environment. We anticipate that Knative open-source eventing interface will continue to enable new underlying source and destination event integrations.

MinIO

ASSESS

Object storage is a popular choice for storing unstructured data and in a few cases structured data in the cloud. We do discourage the use of generic cloud but if you want to minimize the risk of cloud stickiness for object storage, we've found MinIO quite helpful. With an S3-compatible API layer, MinIO abstracts object storage across cloud providers, including AWS, Azure and Google Cloud Platform (GCP), and we've used it successfully in products with flexible target infrastructures from data centers to cloud providers.

Prophet

ASSESS

Even in the era of deep learning, statistical models still play a role in business decision support. Time series models are widely used to forecast inventories, demand, customer traffic, and so on. Hand-crafting these models so that they're robust and flexible has typically been the role of either specialized statisticians or large commercial software vendors. Prophet is an open-source alternative to commercial forecasting packages that can be programmed in R or

PLATFORMS

If you decide to use JavaScript or TypeScript on the server, have a look at Deno which brings a strict sandbox system, built-in dependency and package management, and supports TypeScript out of the box.

(Deno)

Serverless architecture has popularized function-as-a-service style of programming among developers. Knative is an open source, provider agnostic, Kubernetes-based platform to run function-as-a-service workloads.

(Knative)

PLATFORMS

Tendermint is a Byzantine fault tolerance state machine replication engine that lets you implement your own blockchain systems.

(Tendermint)

Python. Facebook claims to use Prophet internally for business forecasting at scale and has made it available as an open-source package for anyone to use. We like that Prophet removes some of the tedium of model construction, maintenance and data manipulation so that human analysts and subject matter experts can focus on doing what they do best.

Quorum

ASSESS

Quorum is "an enterprise-focused version of Ethereum" that aims to provide network permissioning and transaction privacy as well as higher performance. One of our teams has worked deeply with Quorum; however, their experience so far hasn't been great. Some challenges result from complex smart contract programming and some come from Quorum itself. For example, it doesn't work well with load balancers and only has partial database support, which will lead to significant deployment burden. We faced some stability and compatibility issues especially on private transactions. Quorum recently attracted a lot of attention because of JPM Coin. However, from a tech perspective, we recommend being cautious when implementing Quorum while keeping an eye on its development.

SPIFFE

ASSESS

SPIFFE standardization of service identity has been an important step in enabling turnkey solutions for end-to-end encryption and mutual authentication between services. The SPIFFE standards are backed by the OSS SPIFFE Runtime Environment (SPIRE), which automatically delivers cryptographically provable identities to services. Istio also uses SPIFFE by default. SPIFFE enables many use cases, including identity translation, OAuth client authentication, mTLS "encryption everywhere" and workload observability. ThoughtWorks is actively working with the Istio and SPIFFE communities to bridge the gap between legacy service identity providers and SPIFFE-based identities so that mTLS can be used everywhere between services, inside a service mesh and outside.

Tendermint

ASSESS

Byzantine fault tolerance (BFT) is one of the fundamental problems in cryptocurrency and blockchain systems. It requires overall system agreement on a single data value in the presence of a number of arbitrary faulty processes, which includes malicious fraud. Tendermint is a BFT state machine replication engine that lets you implement your own blockchain systems. The consensus engine, Tendermint Core, takes over the peer-to-peer communication and consensus part, you just need to implement the rest of the application (e.g., construct transaction and verify cryptographic signature) and communicate with Tendermint Core through ABCI. Some blockchain implementations have already chosen Tendermint as their consensus engine.

TimescaleDB

ASSESS

In previous Radars we've discussed PostgreSQL for NoSQL. PostgreSQL's maturity and extensibility have led to a steady stream of innovative persistence stores built on the Postgres engine. One that caught our attention is TimescaleDB, a database that allows fast writes and optimized queries over time-series data. Albeit not (yet) as full-featured as InfluxDB, TimescaleDB offers an alternative data model and querying capability. You should evaluate TimescaleDB if you have modest scalability needs, prefer to use SQL and appreciate the stability and familiar administrative interface that PostgreSQL offers.

TOOLS

Cypress
ADOPT

We keep receiving positive feedback on "post-Selenium" web UI testing tools such as Cypress, TestCafe and Puppeteer. Running end-to-end tests can present challenges, such as the long duration of the running process, the flakiness of some tests and the challenges of fixing failures in CI when running tests in headless mode. Our teams have had very good experiences with Cypress by solving common issues such as lack of performance and long wait time for responses and resources to load. Cypress has become the tool of choice for end-to-end testing within our teams.

Jupyter
ADOPT

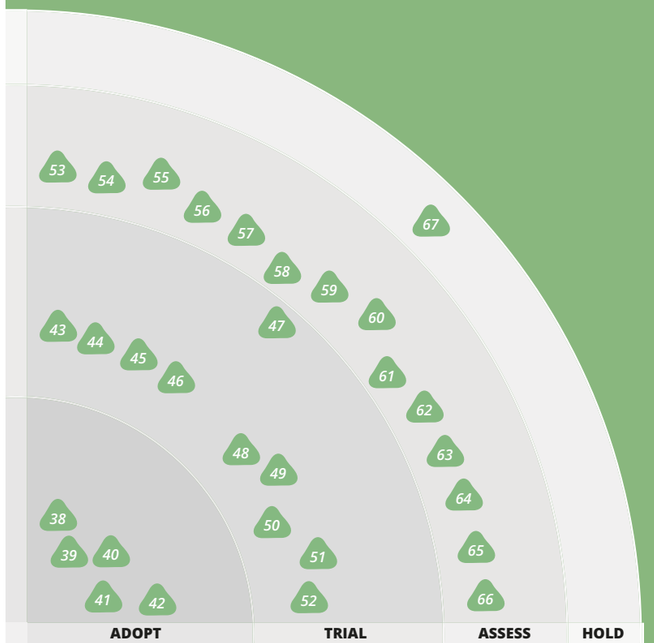
Over the past couple of years, we've noticed a steady rise in the popularity of analytics notebooks. These are Mathematica-inspired applications that combine text, visualization and code in a living, computational document. Jupyter Notebooks are widely used by our teams for prototyping and exploration in analytics and machine learning. We've moved Jupyter to Adopt for this issue of the Radar to reflect that it has emerged as the current default for Python notebooks. However, we caution to use Jupyter Notebooks in production.

LocalStack
ADOPT

One of the challenges of using cloud services is being able to develop and test locally. LocalStack solves this problem for AWS by providing local test double implementations of a wide range of AWS services, including S3, Kinesis, DynamoDB and Lambda. It builds on top of best-of-breed tools such as Kinesalite, dynalite and Moto and adds isolated processes and error injection functionality. LocalStack is very easy to use, ships with a simple JUnit runner and a JUnit 5 extension and can also run inside a docker container. For many teams, it has become the default for testing services that are deployed on AWS.

Terraform
ADOPT

Terraform, is rapidly becoming a de facto choice for creating and managing cloud infrastructures by writing declarative definitions. The configuration of the servers instantiated by Terraform is usually left to Puppet, Chef or Ansible. We like Terraform because the syntax of its files is quite readable and because it supports a number of cloud providers while making no attempt to provide an artificial abstraction across those providers. The active community will add support for the latest features from most cloud providers. Following our first, more cautious, mention of Terraform almost two years ago, it has seen continued development and has evolved into a stable product with a good ecosystem that has proven its value in our projects. The issue with state file management can now be sidestepped by using what Terraform calls a "remote state backend." We've successfully used AWS S3 for that purpose.



ADOPT
38. Cypress
39. Jupyter
40. LocalStack
41. Terraform
42. UI dev environments

TRIAL
43. AnyStatus
44. AVA
45. batect
46. Elasticsearch LTR
47. Helm
48. InSpec
49. Lottie
50. Stolon
51. TestCafe
52. Traefik

ASSESS
53. Anka
54. Cage
55. Cilium
56. Detekt
57. Flagr
58. Gremlin
59. Honeycomb
60. Humio
61. Kubernetes Operators
62. OpenAPM
63. Systems
64. Taurus
65. Terraform provider GoCD
66. Terratest

HOLD
67. Handwritten CloudFormation

TOOLS

Storybook, React Styleguidist, Compositor and MDX provide a comprehensive environment for quickly iterating on UI components, focusing on collaboration between user experience designers and developers.

(UI dev environments)

Much energy and effort continue to be wasted on configuring local development environments and troubleshooting the "works on my machine" problem. batect makes it easy to set up and share a build environment based on Docker, launching containers to perform build tasks.

(batect)

UI dev environments

ADOPT

As more and more teams embrace DesignOps, practices and tooling in this space mature. UI dev environments provide a comprehensive environment for quickly iterating on UI components, focusing on collaboration between user experience designers and developers. We now have a few options in this space: Storybook, React Styleguidist, Compositor and MDX. You can use these tools standalone in component library or design system development as well as embedded in a web application project. Many teams were able to decrease their UI feedback cycles and improve timing of UI work in preparation for development work, which has made using UI dev environments a reasonable default for us.

AnyStatus

TRIAL

As developers used to pushing many small commits daily, we rely on monitors to notify us when builds go green. AnyStatus is a lightweight Windows desktop app that rolls up metrics and events from various sources into one place. Examples include build results and releases, health checks for different services and OS metrics. Think of it as CCTray on steroids. It's also available as a Visual Studio plugin.

AVA

TRIAL

AVA is a test runner for Node.js. Even though JavaScript is single-threaded, IO in Node.js can happen in parallel because of its asynchronous nature. AVA takes advantage of this and runs your tests concurrently, which is especially beneficial for IO-heavy tests. In addition, test files are run in parallel as separate processes, giving you even better performance and an isolated environment for each test file. AVA is a lightweight option, when compared to full-featured frameworks such as Jest. It is opinionated and forces you to write atomic test cases.

batect

TRIAL

So much energy and effort continue to be wasted on configuring local development environments and troubleshooting the "works on my machine" problem. For many years our teams have adopted the "check out and go" approach where we use a scripted approach to ensure the local development environment is configured consistently. batect is an open source tool developed by a ThoughtWorker that makes it easy to set up and share a build environment based on Docker. batect becomes the entry point script for your build system, launching containers to perform build tasks that don't rely at all on local setup. Changes to build configuration and dependencies are simply shared through source control without requiring any changes or installations on local machines or CI agents. While we like Cage, among other tools, in this space, we see batect quickly growing in favor with our teams.

Elasticsearch LTR

TRIAL

One of the challenges of search is ensuring the most relevant results for the user appear at the top of the list. This is where learning to rank (LTR) can help. LTR is the process of applying machine learning to rank documents retrieved by a search engine. If you're using Elasticsearch, you can achieve search-relevant ranking with the Elasticsearch LTR plugin. The plugin uses RankLib for generating the models during the training phase. Then, when querying Elasticsearch, you can use this plugin to "rescore" the top results. We've used it in a few projects and have been happy with the results. There's also an equivalent LTR solution for Solr users.

Helm

TRIAL

Helm is a package manager for Kubernetes. It comes with a repository of curated Kubernetes applications that are maintained in the official Charts repository. Helm has two components: a command line utility called Helm and a cluster component called Tiller. Securing a Kubernetes cluster is a wide and nuanced topic, but we highly recommend setting up Tiller in a role-based access control (RBAC) environment. We've used Helm in a number of client projects and its dependency management, templating and hook mechanism has greatly simplified the application lifecycle management in Kubernetes. However, we recommend proceeding with caution — Helm's YAML templating can be difficult to understand, and Tiller still has some rough edges. Helm 3 is expected to address these issues.

InSpec

TRIAL

How does an organization give autonomy to delivery teams while still making sure their deployed solutions are safe and compliant? How do you ensure that servers, once deployed, maintain a consistent configuration without drift? InSpec is positioned as a solution for continuous compliance and security, but you can also

use it for general infrastructure testing. InSpec allows the creation of declarative infrastructure tests, which can then be continuously run against provisioned environments including production. Our teams particularly praise its extensible design with resources and matchers for multiple platforms. We recommend trialling InSpec as a solution to the problem of assuring compliance and security.

Lottie

TRIAL

Good UI animation could greatly improve user experience. However, to reproduce a designer's delicate animation on an app is usually a challenging task for developers. Lottie is a library for Android, iOS, web, and Windows that parses Adobe After Effects animations exported as JSON with Bodymovin and renders them natively on mobile and on the web. Both designers and developers can continue use their familiar tools and have a fluent collaboration.

Stolon

TRIAL

Setting up highly available PostgreSQL instances can be tricky, which is why we like Patroni — it helps us speed up the setup of PostgreSQL clusters. Stolon is another tool that we've used successfully to run high-availability (HA) clusters of PostgreSQL instances in production using Kubernetes. Although PostgreSQL supports streaming replication out of the box, the challenge in an HA setup is to assure that the clients always connect to the current master. We like that Stolon enforces the connection to the right PostgreSQL master by actively closing connections to unelected masters and routing requests to the active one.

TestCafe

TRIAL

We have good experience using "post-Selenium" web UI testing tools such as Cypress, TestCafe and Puppeteer. TestCafe lets you write tests in JavaScript or TypeScript and runs in-browser tests. TestCafe has several useful features that include out-of-the-box parallel execution and HTTP request mocking. TestCafe uses an asynchronous execution model with no explicit wait times, which results in much more stable test suites. Its selector API makes it easier to implement PageObject patterns. TestCafe recently released version 1.0.*x*, which improved stability and functionality.

Traefik

TRIAL

Traefik is an open-source reverse proxy and load balancer. If you're looking for an edge proxy that provides simple routing without all the features of NGINX and HAProxy, Traefik is a good choice. The router provides a reload-less reconfiguration, metrics, monitoring and circuit breakers that are essential when running microservices. It also integrates nicely with Let's Encrypt to provide SSL termination as well as infrastructure components such as Kubernetes, Docker Swarm or Amazon ECS to automatically pick up new services or instances to include in its load balancing.

Anka

ASSESS

Anka is a set of tools to create, manage and distribute build and test macOS reproducible virtual environments for iOS and macOS development. It brings Docker-like experience to macOS environments: instant start, CLI to manage virtual machines and registry to version and tag virtual machines for distribution. We discovered Anka when we proposed a macOS private cloud solution to a client. This tool is worth considering when applying DevOps workflow to iOS and macOS environments.

Cage

ASSESS

Cage is an open-source wrapper around Docker Compose that lets you configure and run multiple dependent components as a larger application. It lets you orchestrate the execution of components such as Docker images, service source code from repo, scripts to load datastores and pods, which are containers that run together as a unit. Cage uses the Docker Compose v2 configuration file format. It addresses some of the Docker Compose gaps such as supporting multiple environments, including the dev environment for running a distributed application on the local developer machine and the test environment for running integration tests and production.

Cilium

ASSESS

Traditional Linux network security approaches, such as iptables, filter on IP address and TCP/UDP ports. However, these

TOOLS

Trust, but verify. InSpec helps ensure that servers, once deployed, remain secure and compliant over their operational lifetime.

(InSpec)

Leveraging Linux eBPF, Cilium provides API-aware networking and security in a way that is based on service, pod or container identity, and is dynamic and microservices-ready.

(Cilium)

TOOLS

Detekt is a static code analysis for Kotlin. It finds code smells and code complexity. You can run from the command line or use its plugins for integration with popular developer tools.

(Detekt)

Humio is a fairly new player in the log management space. It was built from the ground up to be super fast at both log ingestion and query using its built-in query language on top of a custom-designed time series database.

(Humio)

IP addresses frequently churn in dynamic microservices environments. By leveraging Linux eBPF, Cilium provides API-aware networking and security by transparently inserting security in a way that is based on service, pod or container identity in contrast to IP address identification. By decoupling security from addressing, Cilium could play a significant role as a new network protection layer and we recommend you to check it out.

Detekt
ASSESS

Detekt is a static code analysis tool for Kotlin. It finds code smells and code complexity. You can run it from the command line or use its plugins for integration with popular developer tools such as Gradle (to perform code analysis via builds) or SonarQube (to perform code coverage in addition to static code analysis), and IntelliJ. Detekt is a great addition to build pipelines of Kotlin applications.

Flagr
ASSESS

Feature toggles are an important technique in continuous deployment scenarios. We've come across a number of good home-grown solutions, but we do like the approach Flagr takes: a complete feature toggle as a service, distributed as a Docker container. It comes with SDKs for all major languages, has a simple and well-documented REST API and provides a convenient frontend.

Gremlin
ASSESS

Gremlin is a SaaS solution for organizations to conduct chaos experiments and help test the resilience of their systems. It comes with a series of failure attacks — including resource, network and state failures — that can be run ad hoc or on schedule and require minimal setup (especially for Kubernetes users, who can run Helm to install Gremlin). The Gremlin client also has a nice web-based user interface, which makes it easy to execute and manage chaos experiments.

Honeycomb
ASSESS

Honeycomb is an observability tool that ingests rich data from production systems and makes it manageable through dynamic sampling. Developers can log large amounts of rich events and decide later how to slice and correlate them. This interactive approach is useful when working with today's large distributed systems because we've passed the point where we can reasonably anticipate which questions we might want to ask of production systems.

Humio
ASSESS

Humio is a fairly new player in the log management space. It's been built from the ground up to be super fast at both log ingestion and query using its built-in query language on top of a custom-designed time series database. It integrates with just about everything out there from an ingestion, visualization and alerting perspective. The log management space has been dominated by Splunk and the ELK Stack, so having alternatives is a good thing. We'll be watching Humio's development with interest.

Kubernetes Operators
ASSESS

We're excited about the impact Kubernetes has had on our industry but also concerned about the operational complexity that comes with it. Keeping a Kubernetes cluster up and running and then managing packages deployed on it requires special skills and time. Operational processes such as upgrades, migrations, backups, among others, can be a full-time job. We think that Kubernetes Operators will play a key role in reducing this complexity. The framework provides a standard mechanism to describe automated operational processes for packages running in a Kubernetes cluster. Although Operators were spearheaded and promoted by RedHat, several community-developed Operators for common open-source packages such as Jaeger, MongoDB and Redis have begun to emerge.

OpenAPM
ASSESS

One of the challenges in adopting an open-source alternative to popular commercial packages is sorting through the complicated landscape of projects to understand which components you need, which ones play nicely together and exactly which part of a total solution each component covers. This is particularly difficult in the world of observability, where the standard practice is to purchase one comprehensive but pricey package to do it all. OpenAPM makes the open-source selection process for observability tools easier. It displays the current crop of open-source packages classified by component roles, so you can interactively select

compatible components. As long as you keep the tool up to date, it should help you navigate through the confusing array of possible tools.

Systems
ASSESS

It's easy to think of many of the processes we work within as linear chains of cause and effect. Most of the time we are working within more complex systems where positive and negative feedback loops influence outcomes. Systems is a set of tools for describing, executing and visualizing systems diagrams. Using a compact DSL and running either standalone or within a Jupyter Notebook, it's super easy to describe fairly complex processes and the flow of information through them. It's pretty much a niche tool; but an interesting and fun one.

Taurus
ASSESS

Taurus is a handy application and service performance testing tool written in Python. It wraps many performance testing executors, including Gatling and Locust. You can run it from the command line and easily integrate it with continuous delivery pipelines to run performance tests at different stages of the pipeline. Taurus also has great reporting either as console text-based output or integrated with an interactive web UI. Our teams have found that configuring Taurus YAML files is easy because you can use multiple files to describe each test scenario and refer to underlying executer's scenario definitions.

Terraform provider GoCD
ASSESS

Terraform provider GoCD lets you build pipelines using Terraform, a mature and widely used tool in the infrastructure as code space. With this provider, you can write pipelines in the HashiCorp Configuration Language (HCL) that use all of the functionality Terraform provides, including workspaces, modules and remote state. This approach is an excellent alternative to Gomatic, which we highlighted in the Pipelines as code blip before. The Golang SDK used in this provider has automatic regression tests for the GoCD API which should minimize issues while upgrading.

Terratest
ASSESS

We widely use Terraform as code to configure a cloud infrastructure. Terratest is a Golang library that makes it easier to write automated tests for infrastructure code. A test run creates real infrastructure components (such as servers, firewalls or load balancers), deploys applications on them and validates the expected behavior using Terratest. At the end of the test, Terratest can undeploy the apps and clean up resources. This makes it largely useful for end-to-end tests of your infrastructure in a real environment.

Handwritten CloudFormation
HOLD

AWS CloudFormation is a proprietary declarative language to provision AWS infrastructure as code. Handwriting CloudFormation files is often a default approach to bootstrap AWS infrastructure automation. Although this might be a sensible way to start a small project, our teams, and the industry at large, have found that handwritten CloudFormation simply does not scale as the infrastructure grows. Noticeable pitfalls of handwritten CloudFormation files for large projects include poor readability, lack of imperative constructs, limited parameter definition and usage, and lack of type checking. Addressing these shortfalls has led to a rich ecosystem of both open-source and custom tooling. We find Terraform a sensible default that not only addresses shortfalls of CloudFormation but also has an active community to add the latest AWS features and fix bugs. In addition to Terraform, you can choose from many other tools and languages, including troposphere, sceptre, Stack Deployment Tool and Pulumi.

TOOLS

Systems is a set of tools for describing, executing and visualizing complex systems, where positive and negative feedback loops influence outcomes.

(Systems)

Terraform provider GoCD lets you build pipelines using Terraform, a mature and widely used tool in the infrastructure as code space. It has automatic regression tests for the GoCD API which should minimize issues while upgrading.

(Terraform provider GoCD)

LANGUAGES & FRAMEWORKS

Apollo
ADOPT

Our teams report that Apollo has become the library of choice when building a React application that uses GraphQL to access data from a back-end service. Although the Apollo project also provides a server framework and a GraphQL gateway, the Apollo client gets our attention because it simplifies the problem of binding UI components to data served by any GraphQL backend. Put simply, this means less code needs to be written than using REST backends and redux.

MockK
ADOPT

MockK is our go-to tool for mocks when writing tests for Kotlin applications. We like to use this library because of its first-class support for Kotlin language features such as coroutines or lambda blocks. As a native library, it helps our teams to write clean and concise code on testing Kotlin applications instead of using the inconvenient wrappers of Mockito or PowerMock.

TypeScript
ADOPT

TypeScript, a statically typed language and superset of JavaScript, has become our sensible default. Large-scale projects benefit most from the type safety. Our developers favor its minimal configuration management, well-integrated IDE support and its ability to refactor code safely and gradually adopt types. With its good repository of TypeScript-type definitions at hand, we benefit from all the rich JavaScript libraries while gaining type safety.

Apache Beam
TRIAL

Apache Beam is an open-source unified programming model for defining and executing both batch and streaming data parallel processing pipelines. The Beam model is based on the Dataflow model which allows us to express logic in an elegant way so that we can easily switch between batch, windowed batch or streaming. The big data-processing ecosystem has been evolving quite a lot which can make it difficult to choose the right data-processing engine. One of the key reasons to choose Beam is that it allows us to switch between different runners — a few months ago Apache Samza was added to the other runners it already supports, which include Apache Spark, Apache Flink and Google Cloud Dataflow. Different runners have different capabilities and providing a portable API is a difficult task. Beam tries to strike a delicate balance by actively pulling innovations from these runners into the Beam model and also working with the community to influence the roadmap of these runners. Beam has SDKs in multiple languages including Java, Python and Golang. We've also had success using Scio which provides a Scala wrapper around Beam.

ADOPT
68. Apollo
69. MockK
70. TypeScript

TRIAL
71. Apache Beam
72. Formik
73. HiveRunner
74. joi
75. Ktor
76. Laconia
77. Puppeteer
78. Reactor
79. Resilience4j
80. Room
81. Rust
82. WebFlux

ASSESS
83. Aeron
84. Arrow
85. Chaos Toolkit
86. Dask
87. Embark
88. fastai
89. http4k
90. Immer
91. Karate
92. Micronaut
93. Next.js
94. Pose
95. react-testing-library
96. ReasonML
97. Taiko
98. Vapor

HOLD

LANGUAGES & FRAMEWORKS

joi is a schema description language and validator for JavaScript objects, independent of any particular web application framework.

(joi)

Reactive systems come with improved scalability and resilience but with increased cost of debugging and a steeper learning curve. Some of our projects have observed significant improvements in scalability once they moved to Reactor and the rest of the Reactive stack.

(Reactor)

Formik
TRIAL

Formik is a useful higher-order component for making the surprisingly verbose and complex job of handling forms in React much easier. It localizes state management, assists with submission and optionally uses Yup to simplify data validation.

HiveRunner
TRIAL

HiveRunner is an open-source unit test framework for Apache Hadoop Hive queries based on JUnit4. When writing nontrivial analytics or data pipelines in Hive SQL, we found HiveRunner to be a good enabler for writing tests and even TDDing out some moderately complicated SQL. HiveRunner enables you to write Hive SQL as releasable tested artifacts.

joi
TRIAL

joi is a schema description language and validator for JavaScript objects. We like that joi is independent of any web application framework, so our teams can use the same schemas across different stacks. You can also use companion libraries to generate Swagger documentation for APIs that validate requests with joi schemas.

Ktor
TRIAL

Kotlin has demonstrated its value beyond mobile app development. When building microservices and shipping software to production, our teams have had good experiences with Ktor. Ktor is a framework that, unlike other web frameworks that support Kotlin, is written in Kotlin, using language features such as coroutines which allow for an asynchronous nonblocking implementation. The flexibility to incorporate different tools for logging, DI or a template engine — in addition to its lightweight architecture — makes Ktor an interesting option for creating RESTful services.

Laconia
TRIAL

Laconia is a framework for developing AWS Lambda functions in JavaScript. As interest and use of serverless tech has grown so has the complexity of the applications being built. Laconia is a small, lightweight framework that takes away some of the rough edges we often encounter. It uses dependency injection to isolate your application code from lower-level AWS APIs and provides adaptors for the different events that your application can respond to. It also plays nicely with the Serverless Framework at deploy time. We like small and simple frameworks and Laconia is just that.

Puppeteer
TRIAL

Much like Cypress and TestCafe, Puppeteer is one of the web UI testing tools garnering praise from our teams. Puppeteer can have fine-grained control over headless browsers, obtain time-trace for performance diagnostics and more. Our teams have found Puppeteer to be stable as well as faster and more flexible than alternatives based on WebDriver.

Reactor
TRIAL

We've talked about Reactor in the previous Radars. It has continued to gain traction in many of our projects. With the Spring ecosystem embracing Reactor, it has become the dominant implementation of Reactive Streams. Reactive systems come with improved scalability and resilience but with increased cost of debugging and a steeper learning curve. For those projects where this tradeoff is acceptable, Reactor has proven to be a good choice. Some of our projects have observed significant improvements in scalability once they moved to Reactor and the rest of the Reactive stack. With R2DBC we are starting to get reactive support for RDBMS drivers which addresses one of the weaknesses of reactive services.

Resilience4j
TRIAL

Resilience4j is a lightweight fault tolerance library inspired by Netflix Hystrix. We like its lightweight and modular structure where we pull in specific modules for specific capabilities such as circuit-breaking, rate-limiting, retry, and bulkhead. While service meshes are taking on some of the fault tolerance capabilities, fault tolerance libraries continue to remain a key component of our systems for more nuanced domain-specific fault tolerance behavior and for non-containerized services. With Hystrix going into maintenance mode, Resilience4j becomes a default choice in the Java ecosystem. It can work with synchronous APIs as well as reactive ones. It also surfaces metrics to dropwizard metrics, Prometheus and others using additional modules.

Room
TRIAL

Room is a persistence library to access SQLite on Android. It makes database access code much simpler, with minimal boilerplate code, and more robust, with compile-time verification of SQL queries. Our developers like its complete integration with observable queries, using LiveData. Room is one of the Android Jetpack components that were created to make application development on Android easier.

Rust
TRIAL

Since we last featured it on the Radar in January 2015, we've seen steadily increasing interest in Rust. Some of our clients are now using Rust, mostly in the context of infrastructure tooling but also in high-powered embedded devices. Interest was fuelled by a growing ecosystem as well as improvements to the language itself. The latter included straightforward performance improvements but also changes that make Rust more intuitive, for example the change to non-lexical scoping. Most of the significant changes are included in the Rust 2018 standard released last December.

WebFlux
TRIAL

WebFlux is the Spring Framework implementation of Reactive Streams. We see a rise in reactive programming models across our teams in general and the use of WebFlux in teams who are working in the Spring ecosystem. It's best used in large microservices ecosystems where the high performance of the requests is a major concern. It allows overlapping request processing asynchronously without the complications of using multiple threads. WebFlux uses Reactor as its reactive library but it is interoperable with other reactive libraries via Reactive Streams. It uses Netty as its underlying high-performance communications engine. Although we encourage using Reactive Streams, adopting this programming model requires a significant shift in thinking.

Aeron
ASSESS

Aeron is an efficient and reliable peer-to-peer message transport. It provides a replicated persistent log of messages via a number of media drivers, including HTTP, UDP and TCP. It also supports persistent storage of message streams for later replay. For many applications, Aeron may be overkill because it operates at a pretty low level (OSI Layer 4 conceptually), but it's peer-to-peer design and low (and predictable) latency are useful in a number of use cases. Indeed, we've found it to be useful in certain machine learning applications as well as playing a part in event-driven architectures. We think it's worth pointing out that alternative messaging protocols exist that don't require additional services such as Apache Kafka to be run.

Arrow
ASSESS

Arrow is a functional programming library for Kotlin, created by merging two existing popular libraries (kategory and funKTionale). While Kotlin provides building blocks for functional programming, Arrow delivers a package of ready-to-use higher-level abstractions for application developers. It provides data types, type classes, effects, optics and other functional programming patterns as well as integrations with popular libraries. With Arrow, existing libraries are unified, which should go a long way to avoid fractured communities in this space.

Chaos Toolkit
ASSESS

The Chaos Toolkit is one of a number of Chaos Engineering tools that made this edition of the Radar. You use the toolkit to describe and then run repeatable experiments on your infrastructure to understand its resilience in the event of failure. Many of our teams have been using homegrown tools to do this, so it's great to see an open-source project dedicated to the practice. The toolkit already has drivers for AWS, Azure Service Fabric and GCE (among others) and plays nicely with build tools which lets you experiment with automation. The usual caveats apply though, Chaos Engineering is a very powerful technique that is best used on resilience-aware systems, that is, systems that have been built to cope with failure. For that reason, we recommend starting using Chaos Toolkit in your nonproduction environments first.

Dask
ASSESS

Data scientists and engineers often use libraries such as pandas to perform ad hoc data analysis. Although expressive and powerful, these libraries have one critical limitation: they only work on a single CPU and don't provide horizontal scalability for large data sets. Dask, however, includes a lightweight, high-performance scheduler that can scale from a laptop to a cluster

LANGUAGES & FRAMEWORKS

Room makes Android database access becomes simpler, with minimal boilerplate code and more robust, compile-time verification of SQL queries.

(Room)

Dask provides ways to scale Pandas, Scikit-Learn, and Numpy workflows with minimal rewriting. This makes it an interesting choice for data scientists looking for horizontal scalability for large data sets.

(Dask)

LANGUAGES & FRAMEWORKS

fastai is an open-source Python library that simplifies training fast and accurate neural nets and has out-of-the-box support for computer vision, natural language processing (NLP) and more.

(fastai)

http4k is a lightweight HTTP toolkit written in pure Kotlin for serving and consuming HTTP services. It's elegant and simple with an emphasis on testability.

(http4k)

of machines. And because it works with NumPy, pandas and Scikit-learn, Dask looks promising for further assessment.

Embark
ASSESS

We've recommended Truffle for decentralized application (dapp) development in the past. Embark too can make your work easier. Embark provides features such as scaffolding, building, testing and debugging and integrates with decentralized storages such as IPFS. Through its declarative configuration, you can manage smart contract configuration, dependencies, artifact and deployment quite easily. Embark's interactive CLI dashboard is also impressive. We keep seeing people use Remix to write smart contracts and manually deploy their apps without automated testing, source control management or artifact management. We'd like to draw people's attention to dapp engineering practice by promoting tools such as Truffle and Embark.

fastai
ASSESS

fastai is an open-source Python library that simplifies training fast and accurate neural nets. It is built on top of PyTorch and has become a popular tool for our data scientists. fastai simplifies painful aspects of model training such as preprocessing and loading data down to a few lines of code. It's built on deep learning best practices and has out-of-the-box support for computer vision, natural language processing (NLP) and more. The founders' motivation has been to create an easy-to-use library for deep learning and an improved successor to Keras. GCP, AWS and Azure all have already included fastai in their machine images. The creators of fastai, acknowledging the speed and safety limitations of Python, have announced embracing Swift as an alternative language for deep learning. We'll be closely watching this space.

http4k
ASSESS

http4k is an HTTP toolkit written in pure Kotlin for serving and consuming HTTP services. One of the key ideas behind http4k is that HTTP apps are modelled by composing two simple functions — HttpHandler and Filter. They derive inspiration from Twitter's "Your Server as a Function" paper. It's very lightweight with the core module having no dependencies apart from Kotlin StdLib. Apart from its elegance and simplicity, we also like its emphasis on testability — given that the entities in the libraries are immutable and the routes in the app, as well as the app itself, are just functions, they're super easy to test. One of the things to be aware of, though, is that we don't have nonblocking or coroutines support in http4k yet.

Immer
ASSESS

With the increasing complexity of single-page JavaScript applications, managing state predictably is becoming more and more important. Immutability can help to ensure our applications behave consistently, but unfortunately JavaScript doesn't natively support the ability to create immutable objects. Libraries such as Immutable.js filled that gap but introduced new problems because now two kinds of objects and arrays existed in the application, the library's version and the native JavaScript ones. Immer — German for always — is a tiny package that lets you work with immutable state in a more convenient way. It's based on the copy-on-write mechanism, has a minimal API and operates on normal JavaScript objects and arrays. This means that data access is seamless and no large refactoring efforts are needed when introducing immutability to an existing codebase.

Karate
ASSESS

Given our experience that tests are the only API specifications that really matter, we're always on the lookout for new tools that might help. Karate is an API testing framework whose unique feature is that tests are written directly in Gherkin without relying on a general-purpose programming language to implement test behavior. Karate is really a domain-specific language for describing HTTP-based API tests. Although this approach is interesting and makes for some very readable specifications for simple tests, the special-purpose language for matching and validating payloads can become quite syntax-heavy and difficult to understand. It remains to be seen if more complex tests written in this style will be readable and maintainable over the long haul.

Micronaut
ASSESS

Micronaut is a new JVM framework for building microservices using Java, Kotlin or Groovy. It distinguishes itself through a small memory footprint and short startup time. It achieves these improvements by

avoiding runtime reflection for DI and proxy generation, a common shortcoming of traditional frameworks, and instead uses a DI/AOP container which performs dependency injection at compile time. This makes it attractive not just for standard server-side microservices but also in the context of, for example, the Internet of Things, Android applications and serverless functions. Micronaut uses Netty and has first-class support for reactive programming. It also includes many features that make it cloud-native friendly such as service discovery and circuit breaking. Micronaut is a very promising entrant to the full stack framework for the JVM space and we're keenly watching it.

Next.js

ASSESS

React.js has revolutionized the way most people write single-page JavaScript applications. Generally, we recommend you use Create React App throughout the application lifecycle so you don't have to configure your setup, builds and packages manually. But some developers will prefer a tool whose initial defaults reflect a sound set of opinions. Next.js is just such an opinionated framework and it is garnering quite a bit of interest among our front-end enthusiasts. Next.js simplifies routing, renders on the server side by default and streamlines dependencies and builds. We're keen to see if it lives up to expectations on our own projects.

Pose

ASSESS

Pose is a simple CSS-like animation library for React.js, React Native and Vue.js frameworks. It is a declarative motion system that combines the simplicity of CSS syntax with the power and flexibility of JavaScript animations and interactions.

react-testing-library

ASSESS

As the pace of change in JavaScript frameworks has slowed, our teams have more time to work with specific frameworks and are gaining deeper insights as a result. With React and the dominant testing framework, Enzyme, we've observed a worrying trend of unit tests becoming tightly coupled to implementation details without providing — because the focus is on shallow details — much confidence that features work as expected. These unit tests make evolving the design difficult and they shift too much responsibility up the test pyramid to functional testing. This has made us revisit the idea of subcutaneous testing. Additionally, because of its design, Enzyme has issues trying to keep up with React's development. All this has pushed us toward assessing react-testing-library as a new framework for testing React applications.

ReasonML

ASSESS

ReasonML is an interesting new language based on OCaml with a sprinkling of C-like syntax and uses JavaScript as the default compilation target. Created by Facebook, it allows embedded JavaScript snippets and JSX templating with good React integration. It aims to be approachable for JavaScript developers and leverages that ecosystem, while providing type safety in a functional language.

Taiko

ASSESS

Taiko is a node.js library with a clear and concise API to assist with chrome or chromium browser automation. You can leverage Taiko's smart selectors and write reliable tests as the structure of the web application evolves. There's no need for ID, CSS or XPath selectors or adding explicit waits (for XHR requests) in test scripts. The interactive REPL recorder comes in handy when you want to develop the tests side by side as you explore the functionality. Although you could use Taiko independently, we've had good success using it with Gauge.

Vapor

ASSESS

We're strong proponents of polyglot programming but recognize that in some cases it can make sense to focus on a single programming language. If you're heavily invested in Swift, most likely because of iOS development, and you find yourself looking for a technology to write server-side services, have a look at Vapor, a modern web framework for Swift that has gained a fair amount of popularity.

LANGUAGES & FRAMEWORKS

Micronaut is a new JVM framework for building microservices using Java, Kotlin or Groovy. It distinguishes itself through a small memory footprint and short startup time.

(Micronaut)

Taiko is a node.js library with a clear and concise API to assist with chrome or chromium browser automation. Tests written in Taiko are intended to be highly readable and maintainable.

(Taiko)

Want to stay up-to-date with all Radar-related news and insights?
Follow us on your favorite social channel or become a subscriber.

subscribe now

ThoughtWorks®

We are a global software consultancy and community of passionate purpose-led individuals. We think disruptively to deliver technology to address our clients' toughest challenges all while seeking to revolutionize the IT industry and create positive social change.

Founded 25 years ago, ThoughtWorks has grown to a company of over 6,000 people, including a products division which makes pioneering tools for software teams. ThoughtWorks has 40 offices across 14 countries: Australia, Brazil, Canada, Chile, China, Ecuador, Germany, India, Italy, Singapore, Spain, Thailand, the United Kingdom and the United States.

thoughtworks.com

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar