ThoughtWorks®

TECHNOLOGY RADAR

An opinionated guide
to technology frontiers

Volume 23

#TWTechRadar
thoughtworks.com/radar

Contributors

The Technology Radar is prepared by the ThoughtWorks Technology Advisory Board

The Technology Advisory Board (TAB) is a group of 21 senior technologists at ThoughtWorks. The TAB meets twice a year face-to-face and biweekly by phone. Its primary role is to be an advisory group for ThoughtWorks CTO, Rebecca Parsons.

The TAB acts as a broad body that can look at topics that affect technology and technologists at ThoughtWorks. With the ongoing global pandemic, we once again created this volume of the Technology Radar via a virtual event.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Crispim

Cassie Shum

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Ni Wang

Perla Villarreal

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 at 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.

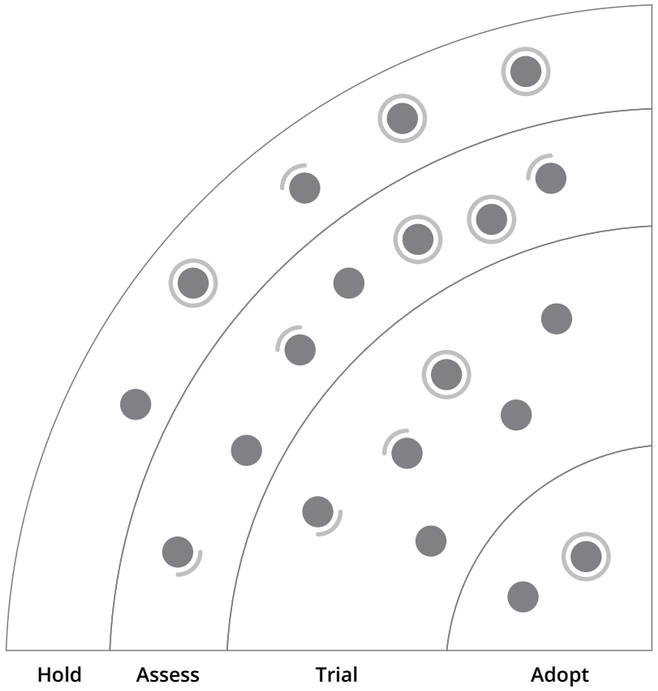
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages & frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings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

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

Radar at a glance

The Radar is all about tracking interesting things, which we refer to as blips. We organize the blips in the Radar using two categorizing elements: quadrants and rings. The quadrants represent different kinds of blips. The rings indicate what stage in an adoption lifecycle we think they should be in.

A blip is a technology or technique that plays a role in software development. Blips are things that are "in motion" — that is we find their position in the Radar is changing — usually indicating that we're finding increasing confidence in them as they move through the rings.



New

Moved in/out

No change

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.

Adopt

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

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.

Assess

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

Hold

Proceed with caution.

Themes for this edition

GraphQL Grandiosity

We see a swell in adoption of GraphQL on many teams, along with a thriving support ecosystem. It solves some common problems that are manifest in modern distributed architectures such as microservices: when developers break things into small pieces, they must often re-aggregate information to solve business requirements. GraphQL offers convenient capabilities to solve this increasingly common problem. Like all powerful abstractions, it offers trade-offs and requires careful consideration by teams to avoid long-term negative effects. For example, we've seen teams provide too much detail about the underlying implementation details via an aggregation tool, leading to unnecessary brittleness in architecture. Another short-term convenience turned long-term headache comes when teams try to use an aggregation tool to create a canonical, universal, centralized data model. We encourage teams to use GraphQL and the burgeoning tools around it; but be cautious of narrowly focused technology generalized to solve too many problems.

The Struggle with the Browser Continues

The web browser was originally designed for document browsing, but now it primarily hosts applications, and the abstraction mismatch continues to challenge developers. To overcome the many headaches inherent in this mismatch, developers keep rethinking and rechallenging established approaches for browser testing, state management and building fast and rich browser applications in general. We see several of these trends in the Radar. First, since we moved Redux to Adopt in 2017 as the default way to manage state in React applications, we now see developers either look elsewhere (Recoil) or delay the decision for a state management library. Second, Svelte has been gaining more interest, and it is challenging one of the established concepts applied by popular application frameworks such as React and Vue.js: the virtual DOM. Third, we keep seeing new tools to deal with testing in the browser: Playwright is yet another attempt at improving UI testing, and Mock Service Worker is a novel approach to decouple tests from their back-end interactions. Fourth, we continue to see the challenge of balancing developer productivity with performance, with browser-tailored polyfills aiming to move the scale in that trade-off.

Visualize All the Things

This Radar features several blips across the technology landscape with one thing in common: visualization. You'll find blips on infrastructure, data science, cloud resources and a host of other innovative visualization tools, including some very effective ways to view difficult abstractions. You'll also find discussions of interactive data tools visualization and dashboarding tools, such as Dash, Bokeh and Streamlit, as well as a host of infrastructure visualization tools, including Kiali for service mesh visualization in microservices architectures. As developer ecosystems become more complex, a picture often goes a long way toward taming the inevitable cognitive overload.

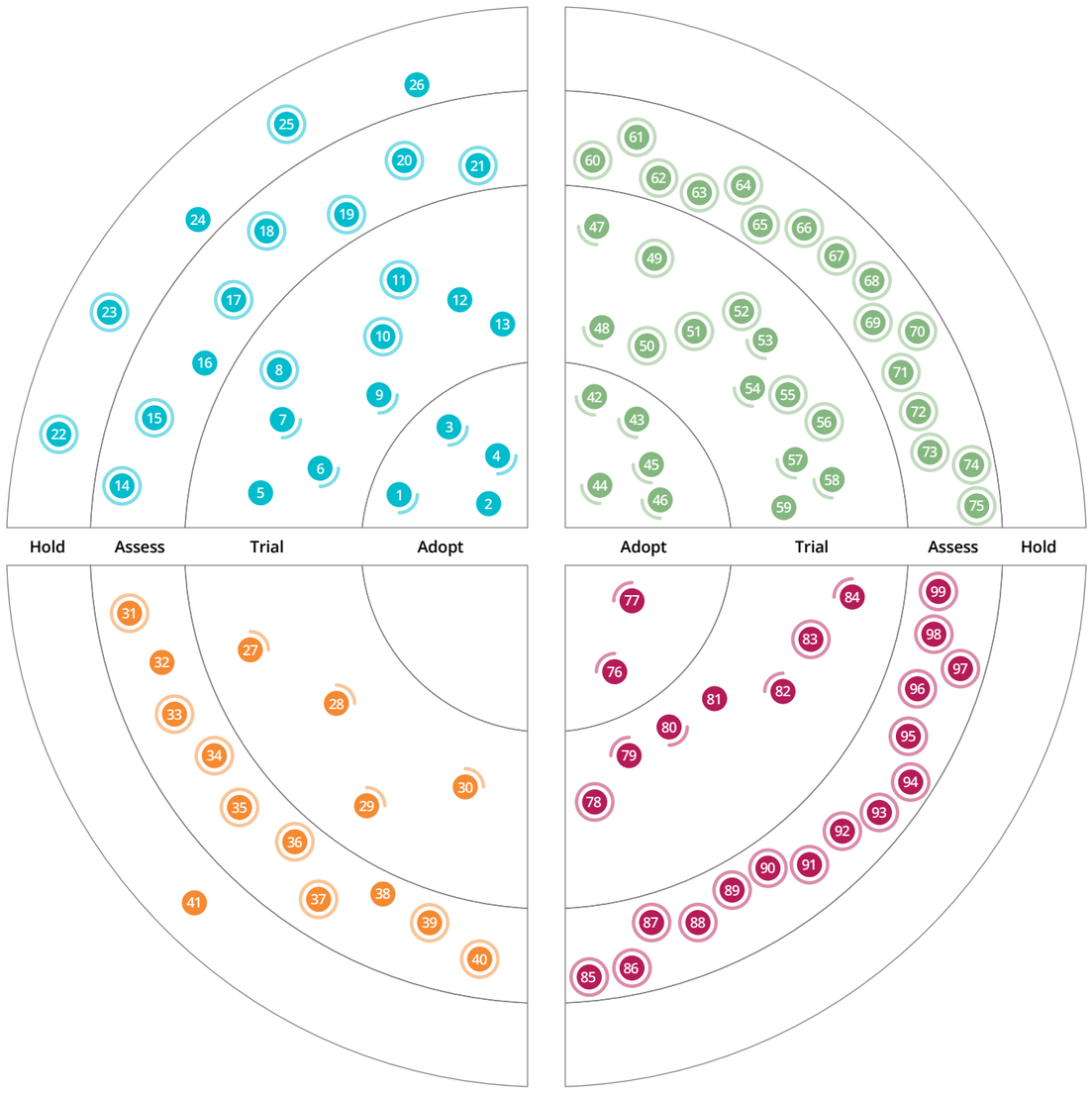
Adolescence of Infrastructure as Code

Managing infrastructure as code has become more common as organizations see the benefits of automating infrastructure, consequently creating an innovation-adoption feedback loop for the creators of tools and frameworks. Tools such as CDK and Pulumi among others offer capabilities far beyond the first generation, improving so much that we believe infrastructure as code has reached its adolescence, with both positive and negative connotations. We were pleasantly surprised at the number of blips in all quadrants reflecting positively on the increasing maturity of the ecosystem. However, we also discussed the challenges around the lack of mature patterns and the struggles many companies face as they're trying to find the best use of this capability — all of which are indicators of continuing growth toward maturity. We're hopeful that the infrastructure community will continue to learn lessons from software design, especially in terms of creating a loosely coupled deployable infrastructure.

Democratizing Programming

Several of our discussions revolve around tools and techniques that promote the democratization of programming: allowing nonprogrammers the ability to perform tasks that previously only programmers could do. For example, solutions such as IFTTT and Zapier have for long been popular in this space. We've observed an increasing use of tools such as Amazon Honeycode, a low-code environment for creating simple business applications. Although tools such as these provide fit-to-purpose programming environments, challenges arise when moving them to production-scale environments. Developers and spreadsheet wizards have long managed to find a compromise between domain-specific and traditional coding environments. The advent of more modern tools renews that discussion across wider domains, with many of the same positive and negative trade-offs.

The Radar



Techniques

Adopt
1. Dependency drift fitness function
2. Run cost as architecture fitness function
3. Security policy as code
4. Tailored service templates

Trial
5. Continuous delivery for machine learning (CD4ML)
6. Data mesh
7. Declarative data pipeline definition
8. Diagrams as code
9. Distroless Docker images
10. Event interception
11. Parallel run with reconciliation
12. Use “remote native” processes and approaches
13. Zero trust architecture

Assess
14. Bounded low-code platforms
15. Browser-tailored polyfills
16. Decentralized identity
17. Kube-managed cloud services
18. Open Application Model (OAM)
19. Secure enclaves
20. Switchback experimentation
21. Verifiable credentials

Hold
22. Apollo Federation
23. ESBs in API Gateway’s clothing
24. Log aggregation for business analytics
25. Micro frontend anarchy
26. Productionizing notebooks

Platforms

Adopt

Trial
27. Azure DevOps
28. Debezium
29. Honeycomb
30. JupyterLab

Assess
31. Amundsen
32. AWS Cloud Development Kit
33. Backstage
34. Dremio
35. DuckDB
36. K3s
37. Materialize
38. Pulumi
39. Tekton
40. Trust over IP stack

Hold
41. Node overload

Tools

Adopt
42. Airflow
43. Bitrise
44. Dependabot
45. Helm
46. Trivy

Trial
47. Bokeh
48. Concourse
49. Dash
50. jscodeshift
51. Kustomize
52. MLflow
53. Pitest
54. Sentry
55. ShellCheck
56. Stryker
57. Terragrunt
58. tfsec
59. Yarn

Assess
60. CML
61. Eleventy
62. Flagger
63. gossm
64. Great Expectations
65. k6
66. Katran
67. Kiali
68. LGTM
69. Litmus
70. Opacus
71. OSS Index
72. Playwright
73. pnpm
74. Sensei
75. Zola

Hold

Languages & Frameworks

Adopt
76. Arrow
77. jest-when

Trial
78. Fastify
79. Immer
80. Redux
81. Rust
82. single-spa
83. Strikt
84. XState

Assess
85. Babylon.js
86. Blazor
87. Flutter Driver
88. HashiCorp Sentinel
89. Hermes
90. io-ts
91. Kedro
92. LitElement
93. Mock Service Worker
94. Recoil
95. Snorkel
96. Streamlit
97. Svelte
98. SWR
99. Testing Library

Hold

TECHNOLOGY RADAR

Techniques

Techniques

Dependency drift fitness function

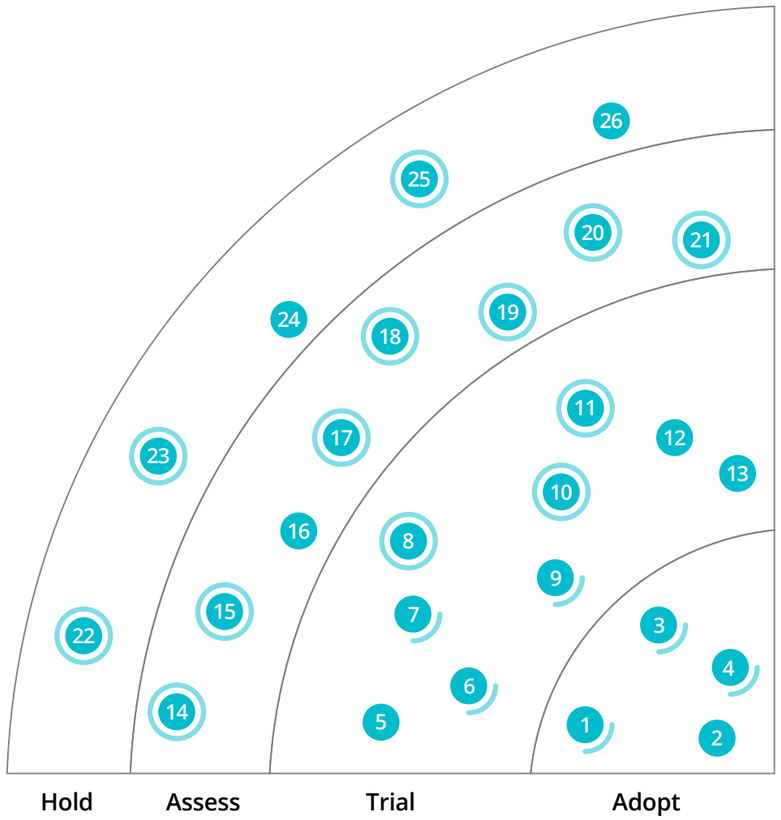
Adopt

Fitness functions introduced by evolutionary architecture, borrowed from evolutionary computing, are executable functions that inform us if our applications and architecture are objectively moving away from their desired characteristics. They're essentially tests that can be incorporated into our release pipelines. One of the major characteristics of an application is the freshness of its dependencies to other libraries, APIs or environmental components that a dependency drift fitness function tracks to flag the out-of-date dependencies that require updating. With the growing and maturing number of tools that detect dependency drifts, such as Dependabot or Snyk, we can easily incorporate dependency drift fitness functions into our software release process to take timely action in keeping our application dependencies up to date.

Run cost as architecture fitness function

Adopt

Automating the estimation, tracking and projection of cloud infrastructure's run cost is necessary for today's organizations. The cloud providers' savvy pricing models, combined with the proliferation of pricing parameters and the dynamic nature of today's architecture, can lead to surprisingly expensive run costs. For example, the price of serverless based on API calls, event streaming solutions based on traffic or data processing clusters based on running jobs, all have a dynamic nature that changes over time as the architecture evolves. When our teams manage infrastructure on the cloud, implementing run cost as architecture fitness function is one of their early activities. This means that our teams can observe the cost of running services against the value delivered; when they see deviations from what was expected or acceptable, they'll discuss whether it's time to evolve the architecture. The observation and calculation of the run cost is implemented as an automated function.



Security policy as code

Adopt

As the technology landscape is becoming more complex, concerns such as security need more automation and engineering

Adopt

1. Dependency drift fitness function
2. Run cost as architecture fitness function
3. Security policy as code
4. Tailored service templates

Trial

5. Continuous delivery for machine learning (CD4ML)
6. Data mesh
7. Declarative data pipeline definition
8. Diagrams as code
9. Distroless Docker images
10. Event interception
11. Parallel run with reconciliation
12. Use "remote native" processes and approaches
13. Zero trust architecture

Assess

14. Bounded low-code platforms
15. Browser-tailored polyfills
16. Decentralized identity
17. Kube-managed cloud services
18. Open Application Model (OAM)
19. Secure enclaves
20. Switchback experimentation
21. Verifiable credentials

Hold

22. Apollo Federation
23. ESBs in API Gateway's clothing
24. Log aggregation for business analytics
25. Micro frontend anarchy
26. Productionizing notebooks

Techniques

Data mesh marks a paradigm shift in how we manage big analytical data that attempts to address many of the known challenges of previous centralized analytical data management. There is, however, a large gap in open-source or commercial tooling to support data mesh.

(Data mesh)

practices. When building systems, we need to take into consideration security policies, which are rules and procedures to protect our systems from threats and disruption. For example, access control policies define and enforce who can access which services and resources under what circumstances; by contrast, network security policies can dynamically limit the traffic rate to a particular service.

Several of our teams have had a great experience treating security policy as code. When we say as code, we not only mean to write these security policies in a file but also to apply practices such as keeping the code under version control, introducing automatic validation in the pipeline, automatically deploying them in the environments and observing and monitoring their performance. Based on our experience and the maturity of the existing tools — including Open Policy Agent and platforms such as Istio which provide flexible policy definition and enforcement mechanisms that support the practice of security policy as code — we highly recommend using this technique in your environment.

Tailored service templates

Adopt

Since we last mentioned tailored service templates, we've seen a broader adoption of the pattern to help pave the road for organizations moving to microservices. With constant advances in observability tooling, container orchestration and service mesh sidecars, a template provides sensible defaults to bootstrap a new service, removing a great deal of setup needed to make the service work well with the surrounding infrastructure. We've had success applying product management principles to tailored service templates, treating internal developers as customers and making it easier for them to push code to production and operate it with appropriate observability. This has the added benefit of acting as a lightweight governance mechanism to centralize default technical decisions.

Continuous delivery for machine learning (CD4ML)

Trial

About a decade ago we introduced continuous delivery (CD), our default way to deliver software solutions. Today's solutions increasingly include machine-learning models and we find them no exception in adopting continuous delivery practices. We call this continuous delivery for machine learning (CD4ML). Although the principles of CD remain the same, the practices and tools to implement the end-to-end process of training, testing, deploying and monitoring models require some modifications. For example: version control must not only include code but also the data, the models and its parameters; the testing pyramid extends to include model bias, fairness and data and feature validation; the deployment process must consider how to promote and evaluate the performance of new models against current champion models. While the industry is celebrating the new buzzword of MLOps, we feel CD4ML is our holistic approach to implement an end-to-end process to reliably release and continuously improve machine-learning models, from idea to production.

Data mesh

Trial

Data mesh marks a welcome architectural and organizational paradigm shift in how we manage big analytical data. The paradigm is founded on four principles: (1) domain-oriented decentralization of data ownership and architecture; (2) domain-oriented data served as a product; (3) self-serve data infrastructure as a platform to enable autonomous, domain-oriented data teams; and (4) federated governance to enable ecosystems and interoperability. Although the principles are intuitive and attempt to address many of the known challenges of previous centralized analytical data management, they transcend the available analytical data technologies. After building data mesh for multiple clients on top of the existing tooling, we learned two things: (a) there is a large gap in open-source or commercial tooling to accelerate implementation of data mesh (for example, implementation of a universal access model to time-based polyglot data which we currently custom build for our clients) and (b) despite the gap, it's feasible to use the existing technologies as the basic building blocks.

Naturally, technology fit is a major component of implementing your organization's data strategy based on data mesh. Success, however, demands an organizational restructure to separate the data platform team, create the role of data product owner for each domain and introduce the incentive structures necessary for domains to own and share their analytical data as products.

Declarative data pipeline definition

Trial

Many data pipelines are defined in a large, more or less imperative script written in Python or Scala. The script contains the logic of the individual steps as well as the code chaining the steps together. When

faced with a similar situation in Selenium tests, developers discovered the Page Object pattern, and later many behavior-driven development (BDD) frameworks implemented a split between step definitions and their composition. Some teams are now experimenting with bringing the same thinking to data engineering. A separate declarative data pipeline definition, maybe written in YAML, contains only the declaration and sequence of steps. It states input and output data sets but refers to scripts if and when more complex logic is needed. A La Mode is a relatively new tool that takes a DSL approach to defining pipelines, but airflow-declarative, a tool that turns directed acyclic graphs defined in YAML into Airflow task schedules, seems to have the most momentum in this space.

Diagrams as code

Trial

We're seeing more and more tools that enable you to create software architecture and other diagrams as code. There are benefits to using these tools over the heavier alternatives, including easy version control and the ability to generate the DSLs from many sources. Tools in this space that we like include Diagrams, Structurizr DSL, AsciiDoctor Diagram and stables such as WebSequenceDiagrams, PlantUML and the venerable Graphviz. It's also fairly simple to generate your own SVG these days, so don't rule out quickly writing your own tool either. One of our authors wrote a small Ruby script to quickly create SVGs, for example.

Distroless Docker images

Trial

When building Docker images for our applications, we're often concerned with two things: the security and the size of the image. Traditionally, we've used container security scanning tools to detect and patch common vulnerabilities and exposures and small distributions such as Alpine Linux to address the image size and distribution performance. We've now gained more experience with distroless Docker images and are ready to recommend this approach as another important security precaution for containerized applications. Distroless Docker images reduce the footprint and dependencies by doing away with a full operating system distribution. This technique reduces security scan noise and the application attack surface. There are fewer vulnerabilities that need to be patched and as a bonus, these smaller images are more efficient. Google has published a set of distroless container images for different languages. You can create distroless application images using the Google build tool Bazel or simply use multistage Dockerfiles. Note that distroless containers by default don't have a shell for debugging. However, you can easily find debug versions of distroless containers online, including a BusyBox shell. Distroless Docker images is a technique pioneered by Google and, in our experience, is still largely confined to Google-generated images. We're hoping that the technique catches on beyond this ecosystem.

Event interception

Trial

As many more companies migrate away from their legacy systems, we feel it's worth highlighting an alternative to change data capture (CDC) as a mechanism for getting data from these systems. Martin Fowler described event interception back in 2004. In modern terms it involves forking requests on ingress to a system so that it's possible to gradually build a replacement. Often this is done by copying events or messages but forking HTTP requests is equally valid. Examples include forking events from point-of-sale systems before they're written to a mainframe and forking payment transactions before they're written to a core banking system. Both lead to the gradual replacement of parts of the legacy systems. We feel that as a technique, obtaining state changes from the source, rather than trying to recreate them postprocessing using CDC, has been overlooked which is why we're highlighting it in this issue of the Radar.

Parallel run with reconciliation

Trial

Replacing legacy code at scale is always a difficult endeavor and one that often benefits from executing a parallel run with reconciliation. In practice, the technique relies on executing the same production flow through both the old and new code, returning the response from the legacy code but comparing the results to gain confidence in the new code. Despite being an old technique, we've seen more robust implementations in recent years building on continuous delivery practices such as canary releases and feature toggles and extending them by adding an extra layer of experimentation and data analysis to compare live results. We've even used the approach to compare cross-functional results such as response time. Although we've used the technique multiple times with bespoke tooling, we certainly owe a nod to GitHub's Scientist tool, which they used to modernize a critical piece of their application and which has now been ported to multiple languages.

Techniques

We're seeing a proliferation of tools that enable you to create software architecture and other diagrams as code. Benefits include easy version control and the ability to generate the DSLs from many sources.

(Diagrams as code)

Event interception is a worthwhile alternative to change data capture when migrating away from legacy systems. Obtaining state changes from the source, rather than trying to recreate them postprocessing using CDC, has been overlooked.

(Event interception)

Techniques

Low-code or no-code platforms can solve very specific problems in very limited domains. Nevertheless, we remain skeptical about their wider applicability.

(Bounded low-code platforms)

Use “remote native” processes and approaches

Trial

As the pandemic stretches on it seems that highly distributed teams will be the “new normal,” at least for the time being. Over the past six months we’ve learnt a lot about effective remote working. On the positive side, good visual work-management and collaboration tools have made it easier than ever to collaborate remotely with colleagues. Developers, for example, can count on Visual Studio Live Share and GitHub Codespaces to facilitate teamwork and increase productivity. The biggest downside to remote work might be burnout: far too many people are scheduled for back-to-back video calls all day long, and this has begun to take its toll. While online visual tools make it easier to collaborate, it’s also possible to build complex giant diagrams that end up being very hard to use, and the security aspects of tool proliferation also need to be carefully managed. Our advice is to remember to take a step back, talk to your teams, evaluate what’s working and what’s not and change processes and tools as needed.

Zero trust architecture

Trial

While the fabric of computing and data continues to shift in enterprises — from monolithic applications to microservices, from centralized data lakes to data mesh, from on-prem hosting to polycloud, with an increasing proliferation of connected devices — the approach to securing enterprise assets for the most part remains unchanged, with heavy reliance and trust in the network perimeter: Organizations continue to make heavy investments to secure their assets by hardening the virtual walls of their enterprises, using private links and firewall configurations and replacing static and cumbersome security processes that no longer serve the reality of today. This continuing trend compelled us to highlight zero trust architecture (ZTA) again.

ZTA is a paradigm shift in security architecture and strategy. It’s based on the assumption that a network perimeter is no longer representative of a secure boundary and no implicit trust should be granted to users or services based solely on their physical or network location. The number of resources, tools and platforms available to implement aspects of ZTA keeps growing and includes: enforcing policies as code based on the least privilege and as granular as possible principles and continuous monitoring and automated mitigation of threats; using service mesh to enforce security control application-to-service and service-to-service; implementing binary attestation to verify the origin of the binaries; and including secure enclaves in addition to traditional encryption to enforce the three pillars of data security: in transit, at rest and in memory. For introductions to the topic, consult the NIST ZTA publication and Google’s white paper on BeyondProd.

Bounded low-code platforms

Assess

One of the most nuanced decisions facing companies at the moment is the adoption of low-code or no-code platforms, that is, platforms that solve very specific problems in very limited domains. Many vendors are pushing aggressively into this space. The problems we see with these platforms typically relate to an inability to apply good engineering practices such as versioning. Testing too is typically really hard. However, we noticed some interesting new entrants to the market — including Amazon Honeycode, which makes it easy to create simple task or event management apps, and Parabola for IFTTT-like cloud workflows — which is why we’re including bounded low-code platforms in this volume. Nevertheless, we remain deeply skeptical about their wider applicability since these tools, like Japanese Knotweed, have a knack of escaping their bounds and tangling everything together. That’s why we still strongly advise caution in their adoption.

Browser-tailored polyfills

Assess

Polyfills are extremely useful to help the web evolve, providing substitute implementations of modern features for browsers that don’t implement them (yet). Too often, though, web applications ship polyfills to browsers that don’t need them, which causes unnecessary download and parsing overhead. The situation is becoming more pronounced now as only a few rendering engines remain and the bulk of the polyfills target only one of them: the Trident renderer in IE11. Further, market share of IE11 is dwindling with support ending in less than a year. We therefore suggest that you make use of browser-tailored polyfills, shipping only necessary polyfills to a given browser. This technique can even be implemented as a service with Polyfill.io.

Decentralized identity
Assess

In 2016, Christopher Allen, a key contributor to SSL/TLS, inspired us with an introduction of 10 principles underpinning a new form of digital identity and a path to get there, the path to self-sovereign identity. Self-sovereign identity, also known as decentralized identity, is a "lifetime portable identity for any person, organization, or thing that does not depend on any centralized authority and can never be taken away," according to the Trust over IP standard. Adopting and implementing decentralized identity is gaining momentum and becoming attainable. We see its adoption in privacy-respecting customer health applications, government healthcare infrastructure and corporate legal identity. If you want to rapidly get started with decentralized identity, you can assess Sovrin Network, Hyperledger Aries and Indy OSS, as well as decentralized identifiers and verifiable credentials standards. We're watching this space closely as we help our clients with their strategic positioning in the new era of digital trust.

Kube-managed cloud services
Assess

Cloud providers have slowly started supporting Kubernetes-style APIs, via custom resource definitions (CRDs), for managing their cloud services. In most cases these cloud services are a core part of the infrastructure, and we've seen teams use tools such as Terraform or Pulumi to provision them. With these new CRDs (ACK for AWS, Azure Service Operator for Azure and Config Connectors for GCP) you can use Kubernetes to provision and manage these cloud services. One advantage of these Kube-managed cloud services is that you can leverage the same Kubernetes control plane to enforce the declarative state of both your application and infrastructure. The downside is that it tightly couples your Kubernetes cluster with infrastructure, so we're carefully assessing it and you should too.

Open Application Model (OAM)
Assess

We've talked a lot about the benefits of creating platform engineering product teams in support of your other product teams, but actually doing it is hard. It seems that the industry is still searching for the right abstraction in the world of infrastructure as code. Although tools such as Terraform and Helm are steps in the right direction, the focus is still on managing infrastructure as opposed to application development. There are also shifts toward the concept of infrastructure as software with new tools such as Pulumi and CDK being released. The Open Application Model (OAM) is an attempt to bring some standardization to this space. Using the abstractions of components, application configurations, scopes and traits, developers can describe their applications in a platform-agnostic way, while platform implementers define their platform in terms of workload, trait and scope. Whether the OAM will be widely adopted remains to be seen, but we recommend keeping an eye on this interesting and needed idea.

Secure enclaves
Assess

Secure enclaves, also identified as trusted execution environments (TEE), refer to a technique that isolates an environment — processor, memory and storage — with a higher level of security and only provides a limited exchange of information with its surrounding untrusted execution context. For example, a secure enclave at the hardware and OS levels can create and store private keys and perform operations with them such as encrypt data or verify signatures without the private keys leaving the secure enclave or being loaded in the untrusted application memory. Secure enclave provides a limited set of instructions to perform trusted operations, isolated from an untrusted application context.

The technique has long been supported by many hardware and OS providers (including Apple), and developers have used it in IoT and edge applications. Only recently, however, has it gained attention in enterprise and cloud-based applications. Cloud providers have started to introduce confidential computing features such as hardware-based secure enclaves: Azure confidential computing infrastructure promises TEE-enabled VMs and access through the Open Enclave SDK open-source library to perform trusted operations. Similarly, GCP Confidential VMs and Compute Engine, still in beta, allow using VMs with data encryption in memory, and AWS Nitro Enclaves is following them with its upcoming preview release. With the introduction of cloud-based secure enclaves and confidential computing, we can add a third pillar to data protection: in rest, in transit and now in memory.

Techniques

Most digital credentials today are simple data records from information systems that are easy to modify and forge and often expose unnecessary information. In recent years, we've seen the growing maturity of verifiable credentials solve this issue.

(Verifiable credentials)

Techniques

Regardless of what you call it, putting business logic in a centralized tool creates architectural coupling, decreases transparency and increases vendor lock-in with no clear upside.

(ESBs in API Gateway's clothing)

Even though we're still in the very early days of secure enclaves for enterprise, we encourage you to consider this technique, while staying informed about known vulnerabilities that can compromise the secure enclaves of the underlying hardware providers.

Switchback experimentation

Assess

Controlled experiments using A/B testing is a great way to inform decisions around product development. But it doesn't work well when we can't establish independence between the two groups involved in the A/B test — i.e., adding someone to the "A" group impacts the "B" group and vice versa. One technique to address this problem space is Switchback experimentation. The core concept here is we switch back and forth between the "A" and "B" modes of the experiment in a certain region at alternating time periods instead of both running during the same time period. We then compare the customer experience and other key metrics between the two time buckets. We've tried this to good effect in some of our projects — it's a good tool to have in our experiments toolbelt.

Verifiable credentials

Assess

Credentials are everywhere in our lives and include passports, driver's licenses and academic certificates. However, most digital credentials today are simple data records from information systems that are easy to modify and forge and often expose unnecessary information. In recent years, we've seen the continuous maturity of Verifiable Credentials solve this issue. The W3C standard defines it in a way that is cryptographically secure, privacy respecting and machine verifiable. The model puts credential holders at the center, which is similar to our experience when using physical credentials: users can put their verifiable credentials in their own digital wallets and show them to anyone at any time without the permission of the credentials' issuer. This decentralized approach also enables users to better manage their own information and selectively disclose certain information and greatly improves data privacy protection. For example, powered by zero-knowledge proof technology, you can construct a verifiable credential to prove that you are an adult without revealing your birthday. The community has developed many use cases around verifiable credentials. We've implemented our own COVID health certification with reference to the COVID-19 Credentials Initiative (CCI). Although verifiable credentials don't rely on blockchain technology or decentralized identity, this technique often works with DID in practice and uses blockchain as a verifiable data registry. Many decentralized identity frameworks are also embedded with verifiable credentials.

Apollo Federation

Hold

When we first covered GraphQL in the Radar, we cautioned that its misuse can lead to antipatterns which, in the long run, has more disadvantages than benefits. Nevertheless, we've seen an increasing interest in GraphQL among our teams because of its ability to aggregate information from different resources. This time we want to caution you about using Apollo Federation and its strong support for a single unified data graph for your company. Even though at first glance the idea of having ubiquitous concepts across the organization is tempting, we have to take into account previous similar attempts in the industry — such as MDM and canonical data model among others — that have exposed the pitfalls of this approach. The challenges can be significant, especially when the domain we find ourselves in is complex enough to create a unique unified model.

ESBs in API Gateway's clothing

Hold

We've long warned against centralized enterprise services buses and defined "smart endpoints, dumb pipes" as one of the core characteristics of a microservices architecture. Unfortunately, we're observing a pattern of traditional ESBs rebranding themselves, creating ESBs in API gateway's clothing that naturally encourage overambitious API gateways. Don't let the marketing fool you: regardless of what you call it, putting business logic (including orchestration and transformation) in a centralized tool creates architectural coupling, decreases transparency, and increases vendor lock-in with no clear upside. API gateways can still act as a useful abstraction for crosscutting concerns, but we believe the smarts should live in the APIs themselves.

Log aggregation for business analytics

Hold

Several years ago, a new generation of log aggregation platforms emerged that were capable of storing and searching over vast amounts of log data to uncover trends and insights in operational data. Splunk was the most prominent but by no means the only example of these tools. Because these platforms provide broad operational and security visibility across the entire estate of applications, administrators and developers have grown increasingly dependent on them. This enthusiasm spread as stakeholders discovered that they could use log aggregation for business analytics. However, business needs can quickly outstrip the flexibility and usability of these tools. Logs intended for technical observability are often inadequate to infer deep customer understanding. We prefer either to use tools and metrics designed for customer analytics or to take a more event-driven approach to observability where both business and operational events are collected and stored in a way they can be replayed and processed by more purpose-built tools.

Micro frontend anarchy

Hold

Since we originally introduced the term in 2016, micro frontends have grown in popularity and achieved mainstream acceptance. But like any new technique with an easy-to-remember name, it has occasionally been misused and abused. Particularly concerning is the tendency to use this architecture as an excuse to mix a range of competing technologies, tools or frameworks in a single page, leading to micro frontend anarchy. A particularly egregious form of this syndrome is using multiple frontend frameworks — for example, React.js and Angular — in the same "single-page" application. Although this might be technically possible, it is far from advisable when not part of a deliberate transition strategy. Other properties that should be consistent from team to team include the styling technique (e.g., CSS-in-JS or CSS modules) and the means by which the individual components are integrated (e.g., iFrames or web components). Furthermore, organizations should decide whether to standardize on consistent approaches or to leave it up to their teams to decide on state management, data fetching, build tooling, analytics and a host of other choices in a micro frontend application.

Productionizing notebooks

Hold

Over the last few decades computational notebooks, first introduced by Wolfram Mathematica, have evolved to support scientific research, exploration and educational workflows. Naturally, in support of data science workflows and with the likes of Jupyter notebooks and Databricks notebooks, they've become a great companion by providing a simple and intuitive interactive computation environment for combining code to analyze data with rich text and visualization to tell a data story. Notebooks were designed to provide an ultimate medium for modern scientific communication and innovation. In recent years, however, we've seen a trend for notebooks to be the medium for running the type of production-quality code typically used to drive enterprise operations. We see notebook platform providers advertising the use of their exploratory notebooks in production. This is a case of good intentions — democratizing programming for data scientists — implemented poorly and at the cost of scalability, maintainability, resiliency and all the other qualities that a long-lived production code needs to support. We don't recommend productionizing notebooks and instead encourage empowering data scientists to build production-ready code with the right programming frameworks, thus simplifying the continuous delivery tooling and abstracting complexity away through end-to-end ML platforms.

Techniques

We see a concerning tendency to use micro frontends as an excuse to mix a range of competing technologies, tools or frameworks in a single page. While this might be technically possible it is far from advisable.

(Micro frontend anarchy)

TECHNOLOGY RADAR

Platforms

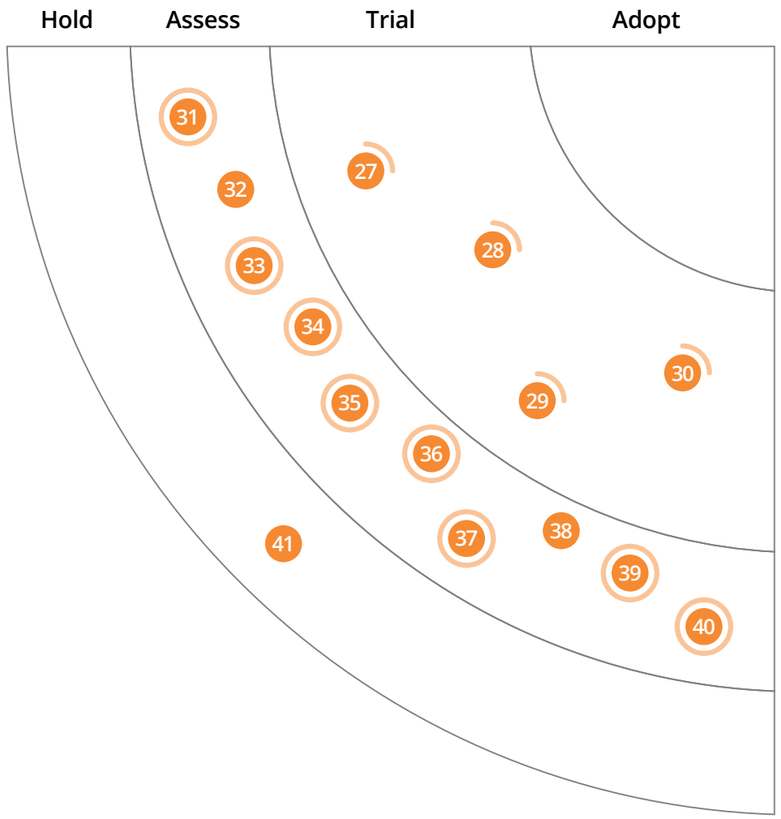
Platforms

Azure DevOps
Trial

Azure DevOps services contain a set of managed services, including hosted Git repos, CI/CD pipelines, automated testing tooling, backlog management tooling and artifact repository. We've seen our teams getting more experience in using this platform with good results, which means Azure DevOps is maturing. We particularly like its flexibility; it allows you to use the services you want even if they're from different providers. For instance, you could use an external Git repository while still using the Azure DevOps pipeline services. Our teams are especially excited about Azure DevOps Pipelines. Nevertheless, all the services offer a good developer experience that helps our teams deliver value.

Debezium
Trial

Debezium is a change data capture (CDC) platform that can stream database changes onto Kafka topics. CDC is a popular technique with multiple use cases, including replicating data to other databases, feeding analytics systems, extracting microservices from monoliths and invalidating caches. Debezium reacts to changes in the database's log files and has CDC connectors for multiple databases, including Postgres, MySQL, Oracle and MongoDB. We're using Debezium in many projects, and it has worked very well for us.



Honeycomb
Trial

Honeycomb is an observability service that ingests rich data from production systems and makes it manageable through dynamic sampling. Developers can log large amounts of rich events and later decide 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. The Honeycomb team is actively developing for a number of languages and frameworks with plugins now available for Go, Node, Java and Rails among others; other new features are being added at a rapid pace. The pricing model has also been simplified to make it more attractive. Our teams love it.

Adopt

Trial
27. Azure DevOps
28. Debezium
29. Honeycomb
30. JupyterLab

Assess
31. Amundsen
32. AWS Cloud Development Kit
33. Backstage
34. Dremio
35. DuckDB
36. K3s
37. Materialize
38. Pulumi
39. Tekton
40. Trust over IP stack

Hold
41. Node overload

Platforms

JupyterLab's interactive environment is an evolution of Jupyter Notebook that extends the original capabilities with drag-and-drop cells and tab autocompletion among other new features.

(JupyterLab)

Backstage from Spotify is an open-source platform for creating developer portals, which can help standardize the number of tools and technologies your teams are using and prevent your software ecosystem becoming fragmented and complex.

(Backstage)

JupyterLab
Trial

Since introducing JupyterLab in the Assess ring in our last issue, it has become the preferred web-based user interface for Project Jupyter for many of our data practitioners. JupyterLab use is rapidly overtaking Jupyter Notebooks, which it will eventually replace. If you're still using Jupyter Notebooks, you should give JupyterLab a try. Its interactive environment is an evolution of Jupyter Notebook: it extends the original capabilities with drag-and-drop cells and tab autocompletion among other new features.

Amundsen
Assess

Data scientists spend a large part of their time on data discovery, which means tooling to help in this space is bound to generate some excitement. Although the Apache Atlas project has become the de facto tool for metadata management, data discovery is still not easily accomplished. Enter Amundsen, which can be deployed in concert with Apache Atlas to provide a much nicer search interface for data discovery.

AWS Cloud Development Kit
Assess

For many of our teams, Terraform has become the default choice for defining cloud infrastructure. However, some of our teams have been experimenting with AWS Cloud Development Kit (AWS CDK), and they like what they've seen so far. In particular, they like the use of first-class programming languages instead of configuration files which allows them to use existing tools, test approaches and skills. Like similar tools, care is still needed to ensure deployments remain easy to understand and maintain. It currently supports TypeScript, JavaScript, Python, Java and C# and .NET. We'll continue to watch AWS CDK, especially since the AWS and HashiCorp teams recently launched a preview for Cloud Development Kit for Terraform to generate Terraform configurations and enable provisioning with the Terraform platform.

Backstage
Assess

Organizations are looking to support and streamline development environments through developer portals or platforms. As the number of tools and technologies increases, some form of standardization is becoming increasingly important for consistency so that developers are able to focus on innovation and product development instead of getting bogged down with reinventing the wheel. A centralized developer portal can offer easy discoverability of services and best practices. Backstage is an open-source platform for creating developer portals by Spotify. It is based upon software templates, unifying infrastructure tooling and consistent and centralized technical documentation. Its plugin architecture allows for extensibility and adaptability into an organization's infrastructure ecosystem.

Dremio
Assess

Dremio is a cloud data lake engine that powers interactive queries against cloud data lake storage. With Dremio, you don't have to manage data pipelines in order to extract and transform data into a separate data warehouse for predictive performance. Dremio creates virtual data sets from data ingested into a data lake and provides a uniform view to consumers. Presto popularized the technique of separating storage from the compute layer, and Dremio takes it further by improving performance and optimizing cost of operation.

DuckDB
Assess

DuckDB is an embedded, columnar database for data science and analytical workloads. Analysts spend significant time cleaning and visualizing data locally before scaling it to servers. Although databases have been around for decades, most of them are designed for client-server use cases and therefore not suitable for local interactive queries. To work around this limitation analysts usually end up using in-memory data-processing tools such as Pandas or data.table. Although these tools

are effective, they do limit the scope of analysis to the volume of data that can fit in memory. We feel DuckDB neatly fills this gap in tooling with an embedded columnar engine that is optimized for analytics on local, larger-than-memory data sets.

K3s
Assess

K3s is a lightweight Kubernetes distribution built for IoT and edge computing. It's packaged as a single binary and has minimal to no OS dependencies, making it really easy to operate and use. It uses sqlite3 as the default storage backend instead of etcd. It has a reduced memory footprint because it runs all relevant components in a single process. It also achieves a smaller binary by stripping out third-party storage drivers and cloud providers that are not relevant for the K3s use cases. For environments with constrained resources, K3s is a pretty good choice and worth considering.

Materialize
Assess

Materialize is a streaming database that enables you to do incremental computation without complicated data pipelines. Just describe your computations via standard SQL views and connect Materialize to the data stream. The underlying differential data flow engine performs incremental computation to provide consistent and correct output with minimal latency. Unlike traditional databases, there are no restrictions in defining these materialized views and the computations are executed in real time.

Pulumi
Assess

We've seen interest in Pulumi slowly but steadily rising. Pulumi fills a gaping hole in the infrastructure coding world where Terraform maintains a firm hold. While Terraform is a tried-and-true standby, its declarative nature suffers from inadequate abstraction facilities and limited testability. Terraform is adequate when the infrastructure is entirely static, but dynamic infrastructure definitions call for a real programming language. Pulumi distinguishes itself by allowing configurations to be written in TypeScript/JavaScript, Python and Go — no markup language or templating required. Pulumi is tightly focused on cloud-native architectures — including containers, serverless functions and data services — and provides good support for Kubernetes. Recently, AWS CDK has mounted a challenge, but Pulumi remains the only cloud-neutral tool in this area. We're anticipating wider Pulumi adoption in the future and looking forward to a viable tool and knowledge ecosystem emerging to support it.

Tekton
Assess

Tekton is a young Kubernetes-native platform for managing continuous integration and delivery (CI/CD) pipelines. It not only installs and runs on Kubernetes but also defines its CI/CD pipelines as Kubernetes custom resources. This means the pipelines can now be controlled by native Kubernetes clients (CLI or APIs) and can take advantage of underlying resource management features such as rollbacks. The pipeline declaration format is flexible and allows defining workflows with conditions, parallel execution paths and handling final tasks to clean up among other features. As a result, Tekton can support complex and hybrid deployment workflows with rollbacks, canary release and more. Tekton is open source and also offered as a managed service by GCP. Although the documentation has room for improvement and the community is growing, we've been using Tekton successfully for production workloads on AWS.

Trust over IP stack
Assess

Continuous challenges with how individuals and organizations establish trust digitally, over the internet, is giving rise to a new approach on how to prove identity, how to share and verify attributes needed to establish trust and how to securely

Platforms

K3s is a lightweight Kubernetes distribution built for IoT and edge computing that strips out third-party storage drivers and cloud providers that aren't relevant for these use cases.

(K3s)

Materialize is a streaming database that enables you to do incremental computation without complicated data pipelines.

(Materialize)

Platforms

This four-layer technical and governance stack aims to establish a baseline for a highly interoperable form of decentralized identity management.

(Trust over IP stack)

Node.js is wildly popular. But that doesn't make it suitable for everything — it's a poor choice, for instance, for compute-heavy workloads. We caution against the tendency to use Node.js indiscriminately or for the wrong reasons.

(Node overload)

transact. Our Radar features some of the foundational technologies such as decentralized identity and verifiable credentials that enable this new era of digital trust.

However, such a global scale change won't be possible without a standardization of a technical governance stack that enables interoperability. The new Trust over IP Foundation, part of the Linux Foundation, has set out to do just that. Taking its inspiration from how TCP/IP standardization as the narrow waist of the internet has enabled interoperability across billions of devices, the group is defining a four-layer technical and governance Trust over IP stack. The stack includes public utilities such as decentralized identifiers, decentralized identity comms to standardized protocols for agents such as digital wallets to communicate, data exchange protocols such as flows to issue and verify verifiable credentials, as well as the application ecosystems such as education, finance, healthcare, etc. If you're revisiting your identity systems and how you establish trust with your ecosystem, we suggest looking into ToIP stack and its supporting tooling, Hyperledger Aries.

Node overload

Hold

Technologies, especially wildly popular ones, have a tendency to be overused. What we're seeing at the moment is Node overload, a tendency to use Node.js indiscriminately or for the wrong reasons. Among these, two stand out in our opinion. Firstly, we frequently hear that Node.js should be used so that all programming can be done in one programming language. Our view remains that polyglot programming is a better approach, and this still goes both ways. Secondly, we often hear teams cite performance as a reason to choose Node.js. Although there are myriads of more or less sensible benchmarks, this perception is rooted in history. When Node.js became popular, it was the first major framework to embrace a nonblocking programming model which made it very efficient for IO-heavy tasks. (We mentioned this in our write-up of Node.js in 2012.) Due to its single-threaded nature, Node.js was never a good choice for compute-heavy workloads, though, and now that capable nonblocking frameworks also exist on other platforms — some with elegant, modern APIs — performance is no longer a reason to choose Node.js.

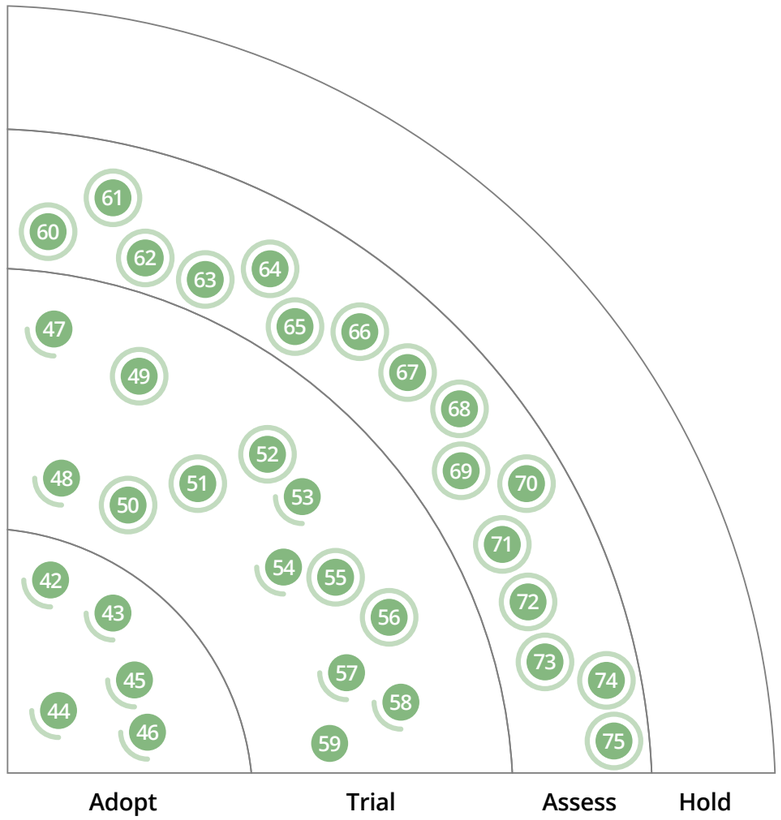
TECHNOLOGY RADAR

Tools

Tools

Airflow
Adopt

Airflow remains our most widely used and favorite open-source workflow management tool for data-processing pipelines as directed acyclic graphs (DAGs). This is a growing space with open-source tools such as Luigi and Argo and vendor-specific tools such as Azure Data Factory or AWS Data Pipeline. However, Airflow differentiates itself with its programmatic definition of workflows over limited low-code configuration files, support for automated testing, open-source and multiplatform installation, rich set of integration points to the data ecosystem and large community support. In decentralized data architectures such as data mesh, however, Airflow currently falls short as a centralized workflow orchestration.



Bitrise
Adopt

Bitrise, a domain-specific CD tool for mobile applications, continues to be a useful part of the mobile workflow, and teams really should be using it. Bitrise can build, test and deploy mobile applications all the way from developer laptop to app store publishing. It's easy to set up and provides a comprehensive set of prebuilt steps for most mobile development needs.

Dependabot
Adopt

Among the available tools for keeping dependencies up to date, Dependabot is a solid default choice in our opinion. Dependabot's integration with GitHub is smooth and automatically sends you pull requests to update your dependencies to their latest versions. It can be enabled at the organization level, so it's very easy for teams to receive these pull requests. If you're not using GitHub, you can still use the Dependabot libraries within your build pipeline. If you're interested in an alternative tool, also consider Renovate, which supports a wider range of services, including GitLab, Bitbucket and Azure DevOps.

Adopt
42. Airflow
43. Bitrise
44. Dependabot
45. Helm
46. Trivy

Trial
47. Bokeh
48. Concourse
49. Dash
50. jscodeshift
51. Kustomize
52. MLflow
53. Pitest
54. Sentry
55. ShellCheck
56. Stryker
57. Terragrunt
58. tfsec
59. Yarn

Assess
60. CML
61. Eleventy
62. Flagger
63. gossm
64. Great Expectations
65. k6
66. Katran
67. Kiali
68. LGTM
69. Litmus
70. Opacus
71. OSS Index
72. Playwright
73. pnpm
74. Sensei
75. Zola

Hold

Tools

Trivy is a vulnerability scanner for containers that ships as a stand-alone binary, making it easy to set up and run the scan locally.

(Trivy)

jscodeshift helps ease the pain of maintaining large-scale JavaScript codebases. We've found it particularly useful when maintaining design systems.

(jscodeshift)

Helm
Adopt

Helm is a package manager for Kubernetes. It comes with a repository of curated Kubernetes applications that are maintained in the official Charts repository. Since we last talked about Helm, Helm 3 has been released, and the most significant change is the removal of Tiller, the server-side component of Helm 2. The benefit of a design without Tiller is that you can only make changes to the Kubernetes cluster from the client side, that is, you can only modify the cluster according to the permissions you have as a user of the Helm command. 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.

Trivy
Adopt

Build pipelines that create and deploy containers should include container security scanning. Our teams particularly like Trivy, a vulnerability scanner for containers. We've tried Clair and Anchore Engine among other good tools in this field. Unlike Clair, Trivy doesn't only check containers but also dependencies in the codebase. Also, because Trivy ships as a stand-alone binary, it's easier to set up and run the scan locally. Other benefits of Trivy are that it's open-source software and that it supports distroless containers.

Bokeh
Trial

Bokeh is one of the principal libraries in Python for creating scientific plots and data visualizations that render in the browser via JavaScript. Such tools, compared to desktop tools that create static images, make it easy to reuse code for exploratory work in web applications. Bokeh is particularly good for this. The library is mature and full-featured. What we like about Bokeh: it's great at keeping to its concern as a presentation layer tool and not trying to take on concerns such as data aggregation (see ggplot) or web app development (such as Shiny or Dash). This makes it a joy to use when separation of concerns is important to you. Bokeh does provide web UI widgets and can run in server mode, but you can take or leave these features as you see fit. Bokeh is flexible, and it doesn't make too many assumptions about how you'll use it nor does it have many dependencies (such as pandas or notebooks).

Concourse
Trial

Implementing sustainable continuous delivery pipelines that can build and deploy production software across multiple environments requires a tool that treats build pipelines and artifacts as first-class citizens. When we first started assessing Concourse, we liked its simple and flexible model, the principle of container-based builds and the fact that it forces you to define pipelines as code. Since then, the usability has improved, and the simple model has stood the test of time. Many of our teams and clients have successfully been using Concourse for large pipeline setups over longer periods of time. We also often leverage Concourse's flexibility to run workers anywhere, for example, when hardware integration tests require a local setup.

Dash
Trial

This edition of the Radar introduces several new tools for creating web applications that help end users visualize and interact with data. These are more than simple visualization libraries such as D3. Instead, they reduce the effort necessary to build standalone analytic applications for manipulating existing data sets. Dash from Plotly is gaining popularity among data scientists for creating richly functional analytics applications in Python. Dash augments Python data libraries much like Shiny sits on top of R. These applications are sometimes referred to as dashboards, but the range of possible functionality is really much greater than the term implies. Dash is particularly suited to building scalable, production-ready applications, unlike Streamlit, another tool in this class. Consider using Dash when you need to present more sophisticated analyses to business users than a low- or no-code solution such as Tableau can provide.

jscodeshift
Trial

Maintaining large-scale JavaScript codebases is never easy, but it's especially challenging when migrating breaking changes. IDEs with refactoring capabilities may help in simple scenarios. However, when your codebase is a widely dependent library, every time you make a breaking change you have to go through a series of client codebases to make the appropriate updates — which requires human oversight and needs to be done manually. jscodeshift, a toolkit to refactor JavaScript and TypeScript, helps relieve this pain. It can parse your code to abstract syntax trees (AST) and provides an API to manipulate the tree with various transformations (e.g., adding, renaming and deleting properties from existing components) and then exports the tree as final source code. jscodeshift also comes with a simple unit testing utility, which can apply test-driven development for writing migration codemods. We've found jscodeshift to be quite helpful when maintaining design systems.

Kustomize
Trial

Kustomize is a tool to manage and customize Kubernetes manifest files. It allows you to select and patch your base Kubernetes resources before applying them to different environments and is now natively supported by kubectl. We like it because it helps keep your code DRY and in contrast to Helm (which is trying to do many things — package management, version management and so on), we find Kustomize follows the Unix philosophy: do one thing well and expect the output of every program to be input to another.

MLflow
Trial

MLflow is an open-source tool for machine-learning experiment tracking and lifecycle management. The workflow to develop and continuously evolve a machine-learning model includes a series of experiments (a collection of runs), tracking the performance of these experiments (a collection of metrics) and tracking and tweaking models (projects). MLflow facilitates this workflow nicely by supporting existing open standards and integrates well with many other tools in the ecosystem. MLflow as a managed service by Databricks on the cloud, available in AWS and Azure, is rapidly maturing and we've used it successfully in our projects. We find MLflow a great tool for model management and tracking, supporting both UI-based and API-based interaction models. Our only growing concern is that MLflow is attempting to deliver too many conflating concerns as a single platform, such as model serving and scoring.

Pitest
Trial

Traditional testing approaches focus on evaluating if our production code is doing what it's supposed to do. However, we could make mistakes in the testing code introducing incomplete or useless assertions that create a false sense of confidence. This is where mutation testing comes in; it assesses the quality of the tests themselves, finding corner cases that are hard to realize. Our teams have used Pitest for a while now, and we recommend its use in Java projects to measure the health of the test suite. In short, mutation testing introduces changes in the production code and executes the same tests a second time; if the tests are still green it means that the tests are not good and need to improve. When you're using programming languages other than Java Stryker is a good choice in this space.

Sentry
Trial

Sentry is a cross-platform application monitoring tool with a focus on error reporting. Tools like Sentry distinguish themselves from traditional logging solutions such as the ELK Stack in their focus on discovering, investigating and fixing errors. Sentry has been around for a while and supports several languages and frameworks. We've used Sentry in many projects, and it has been really useful in tracking errors, finding out if a commit actually fixed an issue and alerting us if an issue resurfaces due to a regression.

ShellCheck
Trial

Even though tooling has vastly improved in the infrastructure space, writing a shell script may make sense in some cases.

Of course, the syntax of shell scripts can only be described as arcane, and as we've less practice writing shell scripts these days, we've come to like ShellCheck, a linter for shell scripts. ShellCheck can be used from the command line, as part of a build or, even better, as an extension in many popular IDEs. The wiki contains a detailed description of several hundred issues that ShellCheck can detect, and most tools and IDEs provide a way to conveniently access the respective wiki page when an issue is found.

Stryker
Trial

Stryker is a relatively new entry in the mutation testing space. Similar to Pitest, Stryker lets you evaluate the quality of your tests. We've been using it quite successfully in JavaScript projects, but it also supports C# and Scala projects. Stryker is very user friendly and highly customizable, and we've been able to increase code coverage as well as confidence in the applications we're delivering for our clients.

Terragrunt
Trial

We've used Terraform extensively to create and manage cloud infrastructure. In our experience with larger setups, where code is divided into modules that are included in different ways, teams eventually hit a wall of unavoidable repetition caused by a lack of flexibility. We've addressed this by using Terragrunt, a thin wrapper for Terraform that implements the practices advocated by Yevgeniy Brikman's Terraform: Up and Running. We've found Terragrunt helpful because it encourages versioned modules and reusability for different environments.

Tools

Kustomize is a tool to manage and customize Kubernetes manifest files that enables you to select and patch your base Kubernetes resources before applying them to different environments.

(Kustomize)

Sentry is a cross-platform application monitoring tool with a focus on error reporting. It has helped our teams track errors, establish if a commit actually fixed an issue and alerted us if an issue resurfaces due to a regression.

(Sentry)

Tools

Flagger is a useful tool for adjusting the portion of traffic that is routed to a new version of a service — which is handy when working with service meshes and API gateways.

(Flagger)

Lifecycle hooks are another useful feature providing additional flexibility. In terms of packaging, Terragrunt has the same limitations as Terraform: there is no proper way to define packages or dependencies between packages. As a workaround, you can use modules and specify a version associated with a Git tag.

tfsec
Trial

Security is everyone's concern, and capturing risks early is always better than facing problems later on. In the infrastructure as code space — where Terraform has been an obvious choice to manage cloud environments — we now also have tfsec, a static analysis tool that scans Terraform templates to find potential security issues. Our teams have been using tfsec quite successfully. The tool is easy to set up and use, which makes it a great choice for any development team determined to mitigate security risks to prevent breaches before they happen. Its preset rules for different cloud providers, including AWS and Azure, compliment the benefits that tfsec brings to the teams that use Terraform.

Yarn
Trial

Yarn continues to be the package manager of choice for many teams. We're excited about Yarn 2, a major new release with a long list of changes and improvements. In addition to usability tweaks and improvements in the area of workspaces, Yarn 2 introduces the concept of zero-installs, which allows developers to run a project directly after cloning it. However, Yarn 2 includes some breaking changes which makes the upgrade nontrivial. It also defaults to plug'n'play (PnP) environments and at the same time doesn't support React Native in PnP environments. Teams can, of course, opt out of PnP or stay on Yarn 1. They should be aware, though, that Yarn 1 is now in maintenance mode.

CML
Assess

We've included continuous delivery for machine learning as a technique in previous Radars, and in this edition we want to highlight a promising new tool called Continuous Machine Learning (or CML) from the people who made DVC. CML aims to bring the best engineering practices of CI and CD to AI and ML teams and can help to organize your MLOps infrastructure on top of a traditional software engineering stack, instead of creating separate AI platforms. We like that they've prioritized support for DVC and see this as a good sign for this burgeoning new tool.

Eleventy
Assess

We've long liked the idea of using static site generators to avoid complexity and improve performance, whenever the use case allows it. Although Eleventy has been around for a few years, it's recently caught our attention as it's matured and previous favorites such as Gatsby.js displayed some scalability problems. Eleventy is quick to learn and easy to build sites with. We also like the ease with which you can create semantic (and therefore more accessible) markup with its templating and its simple and robust support for pagination.

Flagger
Assess

Service meshes and API gateways provide a convenient way to route traffic to a variety of microservices, all of which implement the same API interface. Flagger uses this feature to dynamically adjust the portion of traffic that is routed to a new version of a service. This is a common technique for canary releases or blue/green deployment. Flagger works in conjunction with a variety of popular proxies (including Envoy and Kong) to progressively ramp up requests to a service and report metrics on the load in order to provide fast feedback on a new release. We like that Flagger simplifies this valuable practice so that it can be more widely adopted. Although Flagger is sponsored by Weaveworks, it stands on its own with no obligation to use it in conjunction with Weaveworks' other tooling.

gossm
Assess

When connecting to server instances on AWS, it is recommended to go through a bastion host instead of a direct connection. However, provisioning a bastion host just for that purpose can be frustrating, which is why AWS Systems Manager's Session Manager provides tunneling to more comfortably connect to your servers. gossm is an open-source CLI tool that makes the use of the Session Manager even more convenient. gossm lets you leverage the security provided by Session Manager and IAM policies from your terminal using tools such as ssh and scp. It also has some capabilities that the AWS CLI is missing, including server discovery and SSH integration.

Great Expectations
Assess

With the rise of CD4ML, operational aspects of data engineering and data science have received more attention. Automated data governance is one aspect of this development. Great Expectations is a framework that enables you to craft built-in controls that flag anomalies or quality issues in data pipelines. Just as unit tests run in a build pipeline, Great Expectations makes assertions during execution of a data pipeline. This is useful not only for implementing a sort of Andon for data pipelines but also for ensuring that model-based algorithms remain within the operating range determined by their training data. Automated controls like these can help distribute and democratize data access and custodianship. Great Expectations also ships with a profiler tool to help understand the qualities of a particular data set and to set appropriate limits.

k6
Assess

We're quite excited by k6, a relatively new tool in the performance testing ecosystem with a heavy focus on developer experience. The k6 command line runner executes scripts written in JavaScript and allows you to configure the execution time and the number of virtual users. The CLI has several advanced features that let you see the current statistics before the test has finished executing, scale the number of virtual users beyond what was originally defined and even pause and resume a running test. The command line output provides a set of customizable metrics with transformers that let you visualize the results in Datadog and other observability tools. Adding checks to your scripts is an easy way to integrate performance testing into your CI/CD pipeline. For accelerated performance testing, check out the commercial version, k6 Cloud, which provides cloud scaling and additional visualizations.

Katran
Assess

Katran is a high-performance layer 4 load balancer. It's not for everyone, but if you need redundancy for layer 7 load balancers (such as HAProxy or NGINX) or need to scale load balancers to two or more servers, then we recommend assessing Katran. We see Katran as a flexible and efficient choice over techniques such as round-robin DNS over L7 load balancers or the IPVS Kernel model that network engineers usually adopt to solve similar challenges.

Kiali
Assess

Given the increased use of service mesh to deploy collections of containerized microservices, we can expect to see tools emerge that automate and simplify the administrative tasks associated with this architectural style. Kiali is one such tool. Kiali provides a graphical user interface to observe and control networks of services deployed with Istio. We've found Kiali useful for visualizing the topology of services in a network and understanding the traffic routed between them. For example, when used in conjunction with Flagger, Kiali can display requests that have been routed to a canary service release. We particularly like Kiali's ability to artificially inject network faults into a service mesh to test resilience in the face of network interruptions. This practice is all too often ignored due to the complexity of configuring and running failure tests in a complex mesh of microservices.

LGTM
Assess

Writing secure code is as important as ever, but it's only one of the many things developers have to prioritize. LGTM provides both a safety net and a means to benefit from a knowledge base of secure coding practices. It is a static code analysis tool with a focus on security that is backed by a (partially open-source) catalog of secure coding rules. The rules are implemented as queries over your codebase in the CodeQL query language. It can be used to integrate white-box security checks into your CD pipelines for Java, Go, JavaScript, Python, C# and C/C++. LGTM and CodeQL are part of the Github Security Lab.

Litmus
Assess

Litmus is a chaos engineering tool with a low barrier to entry. It allows you to inject various error scenarios into your Kubernetes cluster with minimal effort. We're particularly excited by the range of capabilities Litmus offers beyond your random pod kill, including simulating network, CPU, memory and I/O issues. Litmus also supports tailored experiments to simulate errors for Kafka and Cassandra among other common services.

Opacus
Assess

The concept of differential privacy first appeared in the Radar in 2016. Although the problem of breaking privacy through

Tools

The Great Expectations framework provides automated data governance that enables you to craft built-in controls that flag anomalies or quality issues in data pipelines.

(Great Expectations)

This is a static code analysis tool with a focus on security that is backed by a catalog of secure coding rules.

(LGTM)

Tools

Sensei makes it easy to create and distribute secure code quality guidelines.

(Sensei)

systematic model inference queries was recognized at the time, it was largely a theoretical issue since remedies were few. The industry has lacked tools to prevent this from happening. Opacus is a new Python library that can be used in conjunction with PyTorch to help thwart one type of differential privacy attack. Although this is a promising development, finding the right model and data set to which it applies has been a challenge. The library is still quite new so we're looking forward to seeing how it'll be accepted going forward.

OSS Index

Assess

It's important for a development team to identify whether the dependencies of their application have known vulnerabilities. OSS Index could be used to achieve this goal. OSS Index is a free catalog of open-source components and scanning tools designed to help developers identify vulnerabilities, understand risk and keep their software safe. Our teams are already integrating this index into pipelines via different languages, including AuditJS and Gradle plugin. The speed is fast, vulnerabilities are identified accurately and few false positives occur.

Playwright

Assess

Web UI testing continues to be an active space. Some of the folks who built Puppeteer have since moved on to Microsoft and are now applying their learnings to Playwright, which allows you to write tests for Chromium and Firefox as well as WebKit, all through the same API. Playwright has gained some attention for its support of all the major browser engines, which it currently achieves by including patched versions of Firefox and Webkit. It remains to be seen how quickly other tools can catch up, with more and more support for the Chrome DevTools Protocol as a common API for automating browsers.

pnpm

Assess

pnpm is an up-and-coming package manager for Node.js that we're looking at closely because of its higher speed and greater efficiency compared to other package managers. Dependencies are saved in a single place on the disk and are linked into the respective node_modules directories. pnpm also supports incremental optimization on file level, provides a solid API foundation to allow extension/customization and supports store server mode, which speeds up dependency download even more. If your organization has a large number of projects with the same dependencies, you may want to take a closer look at pnpm.

Sensei

Assess

Sensei from Secure Code Warrior is a Java IDE plugin that makes it easy to create and distribute secure code quality guidelines. At ThoughtWorks we often advocate for "tools over rules," that is, make it easy to do the right thing over applying checklist-like governance rules and procedures, and this tool fits this philosophy. Developers create recipes that can be easily shared with team members. These can be simple or complex and are implemented as queries targeting the Java AST. Examples include warnings for SQL injection, cryptographic weakness and many others. Another feature we like: Since it executes on code changes in the IDE, Sensei provides faster feedback than the more traditional static analysis tools.

Zola

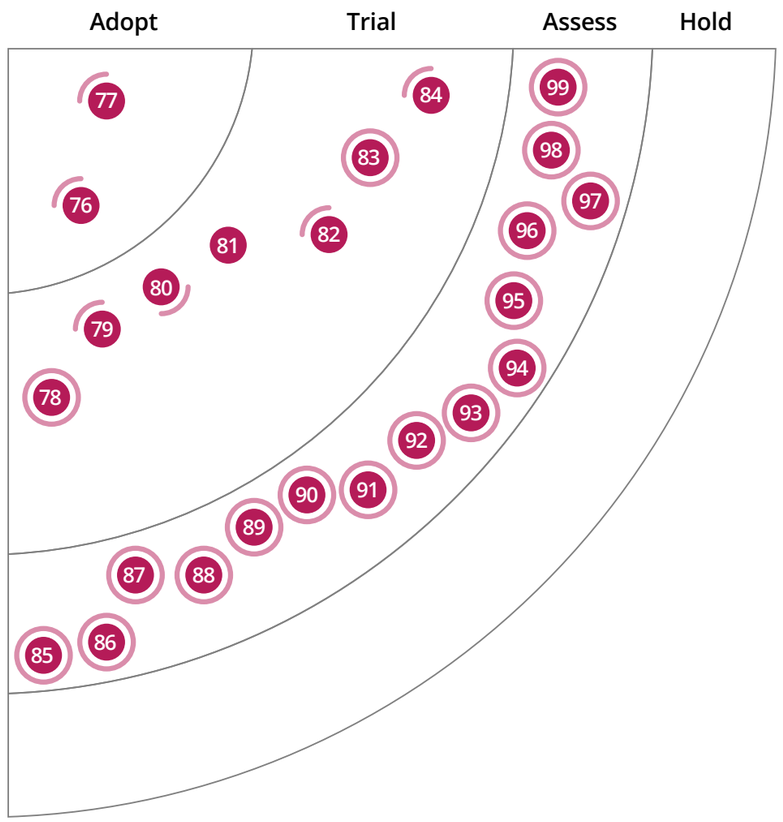
Assess

Zola is a static site generator written in Rust. As such it comes as a single executable with no dependencies, is very fast and supports all the usual things you'd expect such as Sass, content in markdown and hot reloading. We've had success building static sites with Zola and appreciate how intuitive it is to use.

TECHNOLOGY RADAR

Languages & Frameworks

Languages & Frameworks



Arrow
Adopt

Arrow is promoted as the functional companion for Kotlin's standard library. Indeed, the package of ready-to-use higher-level abstractions delivered by Arrow has proven so useful that our teams now consider Arrow a sensible default when working with Kotlin. Recently, in preparation for the 1.0 release, the Arrow team introduced several changes, including the addition of new modules but also some deprecations and removals.

jest-when
Adopt

jest-when is a lightweight JavaScript library that complements Jest by matching mock function call arguments. Jest is a great tool for testing the stack; jest-when allows you to expect specific arguments for mock functions which enables you to write more robust unit tests of modules with many dependencies. It's easy to use and provides great support for multiple matchers, which is why our teams have made jest-when their default choice for mocking in this space.

Fastify
Trial

In the case where implementation in Node.js is necessary, we see that Fastify is an option that our teams are very happy with. This web framework offers ease in handling request-response validations, support for TypeScript and a plugin ecosystem giving our teams an easier experience developing software. Although it's a good option in the Node.js ecosystem, we stand by our previous advice: don't fall into Node overload scenarios.

Immer
Trial

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 offer built-in deeply immutable data structures (see the ES Record and Tuple proposal). 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. Many of our teams now use it in their JavaScript codebases and prefer it to Immutable.js, which is why we're moving it to Trial.

Adopt
76. Arrow
77. jest-when

Trial
78. Fastify
79. Immer
80. Redux
81. Rust
82. single-spa
83. Strikt
84. XState

Assess
85. Babylon.js
86. Blazor
87. Flutter Driver
88. HashiCorp Sentinel
89. Hermes
90. io-ts
91. Kedro
92. LitElement
93. Mock Service Worker
94. Recoil
95. Snorkel
96. Streamlit
97. Svelte
98. SWR
99. Testing Library

Hold

Languages & Frameworks

We no longer consider Redux to be the default approach for state management in React applications. It is still a valuable framework but it can lead to more verbose and harder-to-follow code than alternatives.

(Redux)

XState is a simple JavaScript and TypeScript framework for creating finite state machines and visualizing them as state charts.

(XState)

Redux
Trial

We've decided to move Redux back into the Trial ring to show that we no longer consider it the default approach for state management in React applications. Our experience shows that Redux is still a valuable framework in many cases but compared to other approaches, it also leads to more verbose and harder-to-follow code. Throwing Redux Sagas into the mix usually compounds this issue. As an alternative, you can often use the features in recent versions of React to manage state effectively without an additional framework. However, we want to highlight that when you reach the point at which your simple state management solution starts to become complex, it might be worth reaching for Redux after all or perhaps even Facebook's recently published Recoil.

Rust
Trial

The Rust programming language continues to grow in popularity and has been voted Stack Overflow's "most loved" language by developers five years in a row. We like it too. It's a fast, safe and expressive language that is increasing in utility as its ecosystem grows. For example, Rust is starting to be used for data science and machine learning and can give a significant performance boost. Also, Materialize is a streaming-oriented, low-latency database written in Rust.

single-spa
Trial

single-spa is a JavaScript framework for bringing together multiple micro frontends in a single front-end application. Although we advise against micro frontend anarchy, the use of micro frontends as an excuse to mix and match multiple frameworks, single-spa supports just that. We understand that there are legitimate scenarios such as upgrading to a new revision of a framework across multiple micro frontends where integration across multiple frameworks is necessary. single-spa has been a go-to framework for micro frontend integration for our teams, and they're finding it to work well with SystemJS and managing different versions of a single dependency.

Strikt
Trial

The Kotlin ecosystem keeps growing and more libraries are taking advantage of Kotlin language features to replace their Java alternatives. Strikt is an assertion library that allows you to write test assertions in a very fluent style. It uses Kotlin features such as blocks and lambdas to help make your tests less verbose while maintaining readability. Strikt also supports building custom assertions, which can make your tests more domain specific.

XState
Trial

We've featured several state management libraries in the Radar before, but XState takes a slightly different approach. It's a simple JavaScript and TypeScript framework for creating finite state machines and visualizing them as state charts. It integrates with the more popular reactive JavaScript frameworks (Vue.js, Ember.js, React.js and RxJS) and is based on the W3C standard for finite state machines. Another notable feature is the serialization of machine definitions. One thing that we've found helpful when creating finite state machines in other contexts (particularly when writing game logic) is the ability to visualize states and their possible transitions; we like that it's really easy to do this with XState's visualizer.

Babylon.js
Assess

When we wrote about VR beyond gaming a few years ago we made no prediction on how quickly and to what extent VR solutions would be found in fields other than video gaming. In hindsight, we've certainly seen interest and adoption grow but the uptake has been slower than some of us anticipated. One reason could be tooling. Unity and Unreal are two very mature and capable engines for developing VR applications. We also highlighted Godot. However, these engines are quite unlike what most web and enterprise teams are familiar with. As we continued exploring, we realized that web-based VR solutions have come a long way and we've had positive experience with Babylon.js. Written in TypeScript and rendering its applications in the browser, Babylon.js provides a familiar experience for many development teams. Additionally, Babylon.js is open-source software, mature and well-funded, which makes it even more attractive.

Blazor
Assess

Although JavaScript and its ecosystem is dominant in the web UI development space, new opportunities are opening up with the emergence of WebAssembly. We see Blazor as an interesting option for building interactive web UIs using C#. We especially like this open-source framework because it allows running C# code in the browser on top of WebAssembly, leveraging the .NET Standard runtime and ecosystem as well as custom libraries developed in this programming language. Additionally, it can

interoperate bidirectionally with JavaScript code in the browser if needed.

Flutter Driver
Assess

Flutter Driver is an integration testing library for Flutter applications. With Flutter Driver you can instrument and drive the test suite on either real devices or emulators. Our teams continue to write unit and widget tests to ensure most of the business functionality in Flutter apps is implemented. However, for testing the actual user interaction, we're assessing Flutter Driver, and you should too.

HashiCorp Sentinel
Assess

Although we're big advocates of defining security policy as code, the tooling in this space has been fairly limited. If you're using HashiCorp products (such as Terraform or Vault) and don't mind paying for the enterprise versions, you have the option of using HashiCorp Sentinel. Sentinel is, in effect, a complete programming language for defining and implementing context-based policy decisions. For example, in Terraform it can be used to test for policy violations before applying infrastructure changes. In Vault, Sentinel can be used to define fine-grained access control on the APIs. This approach has all the benefits of encapsulation, maintainability, readability and extensibility that high-level programming languages offer, creating an attractive alternative to traditional, declarative security policy. Sentinel is in the same class of tools as Open Policy Agent but is proprietary, closed-source and only works with HashiCorp products.

Hermes
Assess

Hermes is a JavaScript engine optimized for fast start-up of React Native applications on Android. JavaScript engines such as V8 have just-in-time (JIT) compilers that profile the code at run time to produce optimized instructions. Hermes, however, takes a different approach by compiling the JavaScript code ahead of time (AOT) into an optimized bytecode. As a result you get a smaller APK image size, lean memory consumption and faster startup time. We're carefully assessing Hermes in a few React Native apps and recommend you do the same.

io-ts
Assess

We've been really enjoying using TypeScript for a while now and love the safety that the strong typing provides. However, getting data into the bounds of the type system, from say a call to a back-end service, can lead to run-time errors. One library that helps solve this problem is io-ts. It bridges the gap between compile-time type-checking and run-time consumption of external data by providing encode and decode functions. It can also be used as a custom type guard. According to our teams, it's an elegant solution to a rascal of a problem.

Kedro
Assess

In the past we've talked about the improving tooling for applying good engineering practices in data science projects. Kedro is another good addition in this space. It's a development workflow framework for data science projects that brings a standardized approach to building production-ready data and machine-learning pipelines. We like the focus on software engineering practices and good design with its emphasis on test-driven development, modularity, versioning and good hygiene practices such as keeping credentials out of the codebase.

LitElement
Assess

Steady progress has been made since we first wrote about web components in 2014. LitElement, part of the Polymer Project, is a simple library that you can use to create lightweight web components. It's really just a base class that removes the need for a lot of the common boilerplate making writing web components a lot easier. We've had early success using it on projects and are excited to see the technology maturing.

Mock Service Worker
Assess

Web applications, especially those written for internal use in enterprises, are usually written in two parts. The user interface and some business logic run in the web browser while business logic, authorization and persistence run on a server. These two halves normally communicate via JSON over HTTP. The endpoints shouldn't be mistaken for a real API; they're simply an implementation detail of an application that is split across two run-time environments. At the same time, they provide a valid seam to test the pieces individually. When testing the JavaScript part, the server side can be stubbed and mocked at the network level by a tool such as Mountebank. An alternative

Languages & Frameworks

Sentinel is a complete programming language for defining and implementing context-based policy decisions.

(HashiCorp Sentinel)

Kedro is a development workflow framework for data science projects that brings a standardized approach to building production-ready data and machine-learning pipelines.

(Kedro)

Languages & Frameworks

Created at Stanford University, Snorkel enables us to programmatically label the massive data sets that are used to train machine learning algorithms.

(Snorkel)

approach is to intercept the requests in the browser. We like the approach taken by Mock Service Worker because with service workers it uses an abstraction familiar to developers. This approach results in a simpler setup and faster test execution. However, because these tests don't test the actual network layer, you want to implement some end-to-end tests as part of a healthy test pyramid.

Recoil
Assess

More and more teams using React are reevaluating their options for state management, something we also mention in our reassessment of Redux. Now, Facebook — the creators of React — have published Recoil, a new framework for managing state, which came out of an internal application that had to deal with large amounts of data. Even though we currently do not have much practical experience with Recoil, we see its potential and promise. The API is simple and easy to learn; it feels like idiomatic React. Unlike other approaches, Recoil provides an efficient and flexible way to have state shared across an application: it supports dynamically created state by derived data and queries as well as app-wide state observation without impairing code splitting.

Snorkel
Assess

Modern ML models are very complex and require massive amounts of labeled training data sets to learn from. Snorkel started at the Stanford AI lab with the realization that manually labeling data is very expensive and often not feasible. Snorkel allows us to label training data programmatically via the creation of labeling functions. Snorkel employs supervised learning techniques to assess the accuracies and correlations of these labeling functions, and then reweighs and combines their output labels, leading to high-quality training labels. The creators of Snorkel have since come out with a commercial platform called Snorkel Flow. While Snorkel itself is no longer actively developed, it's still significant for its ideas on the use of weakly supervised methods to label data.

Streamlit
Assess

Streamlit is an open-source application framework in Python used by data scientists for building good-looking data visualization applications. Streamlit stands out from competitors such as Dash with its focus on rapid prototyping and support for a wide range of visualization libraries, including Plotly and Bokeh. For data scientists who need quick showcases during the experimentation cycle, Streamlit is a solid choice. We're using it in a few projects and like how we can put together interactive visualizations with very little effort.

Svelte
Assess

We continue to see new front-end JavaScript frameworks, and Svelte stands out as a promising new component framework. Unlike other frameworks that leverage the virtual DOM, Svelte compiles your code into vanilla framework-less JavaScript code that surgically updates the DOM directly. However, it's only a component framework; if you're planning to build feature-rich applications, consider assessing Sapper together with Svelte.

SWR
Assess

SWR is a React Hooks library for fetching remote data. It implements the stale-while-revalidate HTTP caching strategy. SWR first returns data from cache (stale), then sends the fetch request (revalidate) and finally refreshes the values with the up-to-date response. Components receive a stream of data, first stale and then fresh, constantly and automatically. Our developers have had a good experience using SWR, dramatically improving the user experience with always having data on the screen. However, we caution teams to only use SWR caching strategy when appropriate for an application to return stale data. Note that HTTP requires that caches respond to a request with the most up-to-date response held that is appropriate to the request, and only in *carefully considered circumstances* is a stale response allowed to be returned.

Testing Library
Assess

Testing Library is a family of packages for testing applications in numerous frameworks such as React, Vue, React Native and Angular among others. This set of libraries helps you test UI components in a user-centric way by encouraging you to test user behavior rather than implementation details, such as the presence of elements in the UI at a certain moment in time. One of the benefits of this mindset is more reliable tests, and this is what we call out as its main differentiator. We recommend you assess this family of libraries when testing your web applications in any framework. Although our direct experience is limited to React Testing Library and Angular Testing Library, we've been impressed with what we've seen.

ThoughtWorks®

We are a software consultancy and community of passionate purpose-led individuals, 7,000+ people strong across 43 offices in 14 countries. Over our 25+ year history, we have helped our clients solve complex business problems where technology is the differentiator. When the only constant is change, we prepare you for the unpredictable.

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®

thoughtworks.com/radar

#TWTechRadar