Technology Radar

An opinionated guide to technology frontiers

thoughtworks

Volume 26

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.

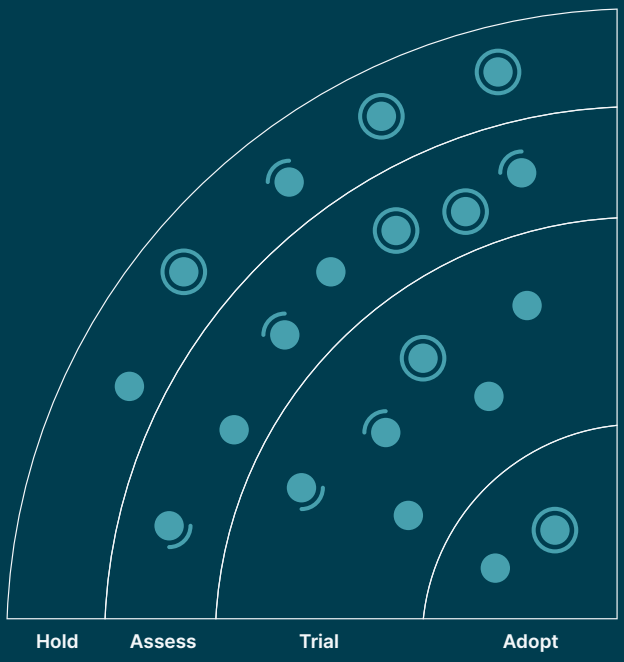
For more background on the Radar, see **thoughtworks.com/radar/faq**.

© Thoughtworks, Inc. All Rights Reserved.

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.



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.

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.

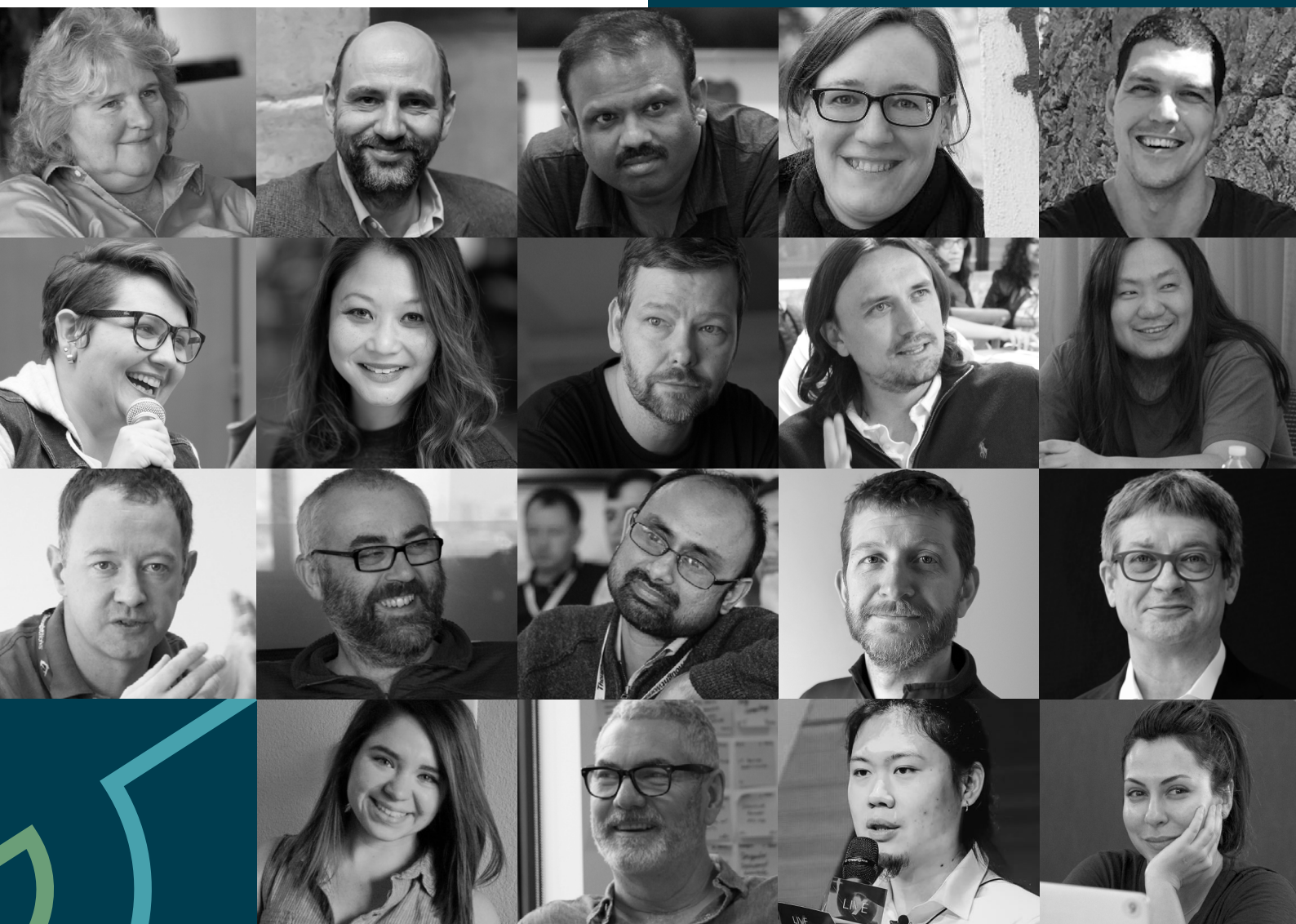
© Thoughtworks, Inc. All Rights Reserved.

Contributors

The Technology Advisory Board (TAB) is a group of 18 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 Falconi Crispim
Cassie Shum
Erik Dörnenburg
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Lakshminarasimhan Sudarshan
Mike Mason
Neal Ford
Perla Villarreal
Scott Shaw
Shangqi Liu
Zhamak Dehghani



Themes

The Bizarre Bazaar: The Changing Economics of Open-Source Software

At Thoughtworks, we've long been fans of open-source software, popularized in part by Eric Raymond's famous essay "The Cathedral and the Bazaar." Open-source software improves developer mobility and crowdsources both bug fixes and innovation. However, attempts at commercialization demonstrate the enormous economic complexity of the current ecosystem. See, for example, AWS forking Elasticsearch to OpenSearch in September 2021 in response to Elastic changing their license to require cloud service providers who profit off their work to contribute back. This shows how difficult it can be for commercial open-source software to maintain a competitive moat. (The same concern applies with free closed-source software, as we witnessed some companies exploring Docker Desktop alternatives because of Docker's ongoing effort to find the right commercial model.) Sometimes the power dynamics work in reverse: because Facebook funded Presto as an open-source product, the maintainers were able to keep the IP and rebrand it as Trino after they left the company, in effect benefiting from Facebook's investment. The situation is further muddied by the amount of critical infrastructure that isn't corporate-sponsored, where companies usually only notice how reliant they are on unpaid labor when a critical security bug is discovered (as recently happened with Log4J). In some cases, funding hobbyist maintainers through GitHub or Patreon provides enough lift to make a difference; in others it simply creates an additional feeling of responsibility on top of their day job and contributes to burnout. We continue to be strong supporters of open-source software but recognize that the economics are becoming increasingly bizarre, and there are no easy solutions to finding the right balance.

Software Supply Chain Innovations

Public instances of severe problems — the Equifax data breach, SolarWinds attack, Log4J remote zero-day vulnerability and so on — were caused by poor governance of the software supply chain. Teams now realize that responsible engineering practices include validating and governing project dependencies, and this drives a number of blips in this edition of the Radar. Entries include checklists and standards such as Supply chain Levels for Software Artifacts (SLSA), a Google-backed consortium to provide guidance on standard threats to the supply chain, and CycloneDX, another set of standards driven by the OWASP community. We also feature concrete tools such as Syft, which generates a Software Bill of Materials (SBOM) from container images. Hackers are increasingly taking advantage of the asymmetrical nature of offense and defense in the security arena — they only need to find one vulnerability, whereas defenders must secure the entire attack surface — while employing increasingly sophisticated hacking techniques. Improved supply chain security is a critical piece of our response as we work to keep systems secure.

© Thoughtworks, Inc. All Rights Reserved.

Why Do Developers Keep Implementing State Management in React?

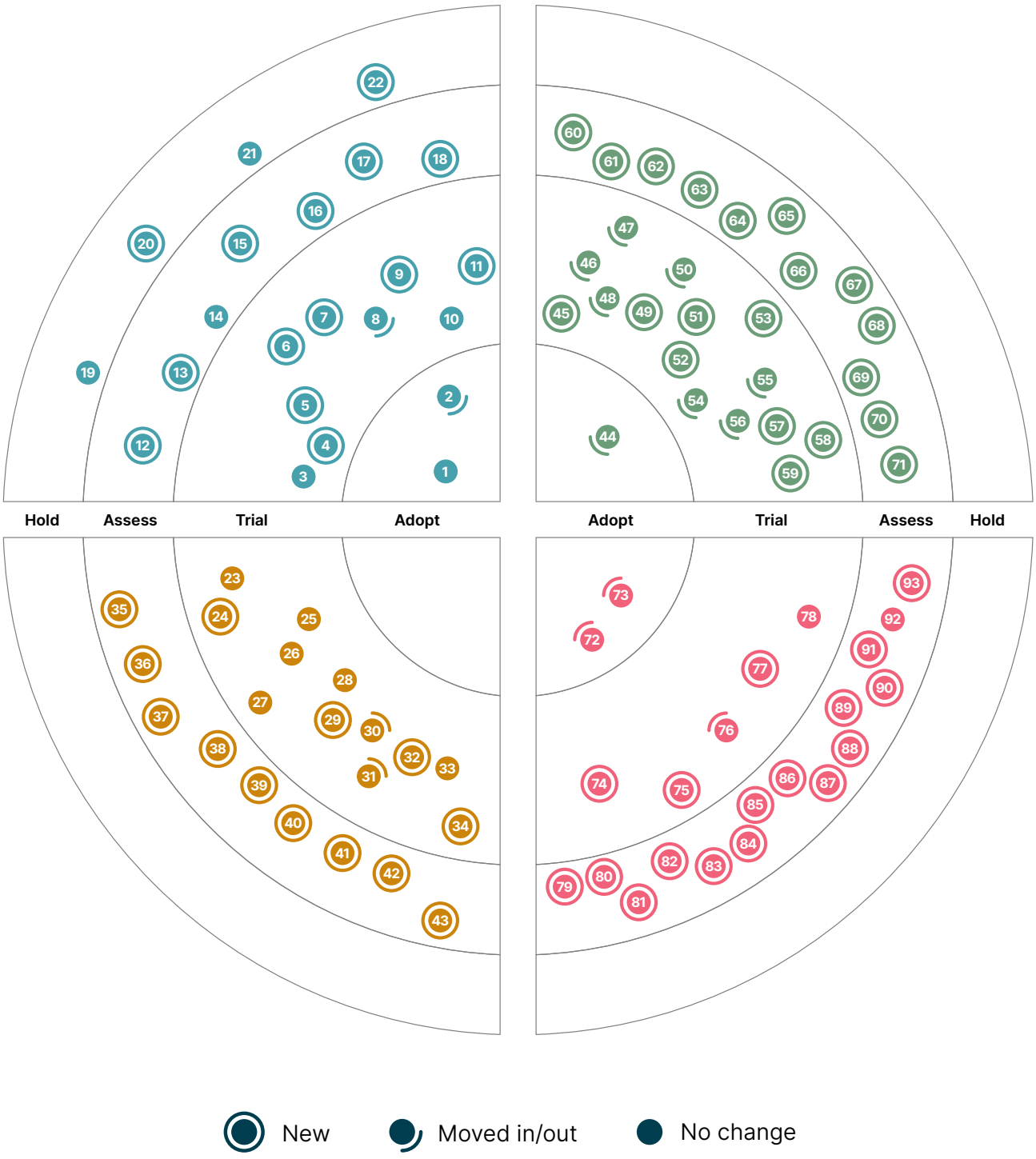
Burgeoning categories of frameworks appear to be a common pattern in the Radar: a foundational framework becomes popular, followed by a raft of tools creating an ecosystem for common deficiencies and enhancements, ending with consolidation around a few popular tools. However, React state management seems resistant to this common tendency. Ever since Redux was released, we've seen a steady stream of tools and frameworks that manage state in slightly different ways, each with a different set of trade-offs. We don't know why; we can only speculate: Is this the natural churn the JavaScript ecosystem seems to promote? Is it an underlying deficiency in React, a fun and seemingly tractable problem that encourages developers to experiment? Or is it the permanent impedance mismatch between a document-reading format (web browsers) and the interactivity (and state) required to host application development atop documents? We don't know the reason, but we look forward to the next round of attempts at solving this seemingly perpetual problem.

The Neverending Quest for The Master Data Catalog

The desire to get more value out of corporate data assets drives much of the investment we're seeing right now in digital technology. At its core, this effort is often focused on better ways to find and access all the relevant data. For nearly as long as businesses have been collecting digital data, there have been efforts to rationalize and catalog it into a single, top-down corporate directory. But time after time, this intuitively appealing notion runs up against the hard realities of complexity, redundancy and ambiguity inherent in large organizations. Recently we've noticed a renewed interest in corporate data catalogs and a surge of Radar blip proposals for clever new tools such as **Collibra** and **DataHub**. These tools can provide consistent, discoverable access to lineage and metadata across silos, but their expanding feature sets also extend to governance, quality management, publishing and more.

In contrast to this trend, there also seems to be a growing movement away from centralized, top-down data management and toward federated governance and discovery based on a data mesh architecture. This approach addresses the inherent complexity of corporate data by setting expectations and standards centrally but segregating data custodianship along business domain lines. Domain-oriented data product teams control and share their own metadata including discoverability, quality and other information. In this scenario, the catalog is just a way to surface information for searching and browsing. The resulting data catalogs are simpler and easier to maintain, reducing the need for richly featured cataloging and governance platforms.

The Radar



© Thoughtworks, Inc. All Rights Reserved.

The Radar

Techniques

Adopt

1. Four key metrics
2. Single team remote wall

Trial

3. Data mesh
4. Definition of production readiness
5. Documentation quadrants
6. Rethinking remote standups
7. Server-driven UI
8. Software Bill of Materials
9. Tactical forking
10. Team cognitive load
11. Transitional architecture

Assess

12. CUPID
13. Inclusive design
14. Operator pattern for nonclustered resources
15. Service mesh without sidecar
16. SLSA
17. The streaming data warehouse
18. TinyML

Hold

19. Azure Data Factory for orchestration
20. Miscellaneous platform teams
21. Production data in test environments
22. SPA by default

Platforms

Adopt

—

Trial

23. Azure DevOps
24. Azure Pipeline templates
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Reusable workflows in Github Actions
33. Sealed Secrets
34. VerneMQ

Assess

35. actions-runner-controller
36. Apache Iceberg
37. Blueboat
38. Cloudflare Pages
39. Colima
40. Collibra
41. CycloneDX
42. Embeddinghub
43. Temporal

Hold

—

The Radar

Tools

Adopt
44. tfsec

Trial
45. AKHQ
46. cert-manager
47. Cloud Carbon Footprint
48. Conftest
49. kube-score
50. Lighthouse
51. Metaflow
52. Micrometer
53. NUKE
54. Pactflow
55. Podman
56. Sourcegraph
57. Syft
58. Volta
59. Web Test Runner

Assess
60. CDKTF
61. Chrome Recorder panel
62. Excalidraw
63. GitHub Codespaces
64. GoReleaser
65. Grype
66. Infracost
67. jc
68. skopeo
69. SQLFluff
70. Terraform Validator
71. Typesense

Hold
—

Languages and Frameworks

Adopt
72. SwiftUI
73. Testcontainers

Trial
74. Bob
75. Flutter-Unity widget
76. Kotest
77. Swift Package Manager
78. Vowpal Wabbit

Assess
79. Android Gradle plugin - Kotlin DSL
80. Azure Bicep
81. Capacitor
82. Java 17
83. Jetpack Glance
84. Jetpack Media3
85. MistQL
86. npm workspaces
87. Remix
88. ShedLock
89. SpiceDB
90. sqlc
91. The Composable Architecture
92. WebAssembly
93. Zig

Hold
—

© Thoughtworks, Inc. All Rights Reserved.

Techniques

Adopt

1. Four key metrics
2. Single team remote wall

Trial

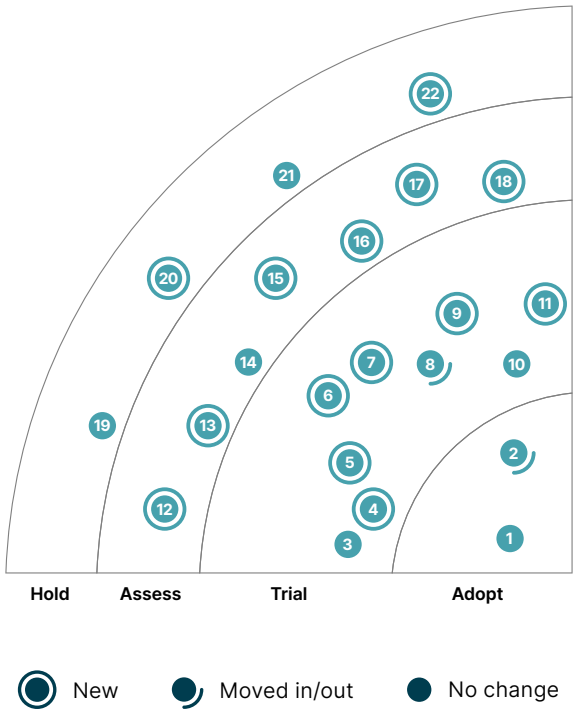
3. Data mesh
4. Definition of production readiness
5. Documentation quadrants
6. Rethinking remote standups
7. Server-driven UI
8. Software Bill of Materials
9. Tactical forking
10. Team cognitive load
11. Transitional architecture

Assess

12. CUPID
13. Inclusive design
14. Operator pattern for nonclustered resources
15. Service mesh without sidecar
16. SLSA
17. The streaming data warehouse
18. TinyML

Hold

19. Azure Data Factory for orchestration
20. Miscellaneous platform teams
21. Production data in test environments
22. SPA by default



1. Four key metrics

Adopt

To measure software delivery performance, more and more organizations are defaulting to the four key metrics as defined by the DORA research program: change lead time, deployment frequency, mean time to restore (MTTR) and change fail percentage. This research and its statistical analysis have shown a clear link between high-delivery performance and these metrics; they provide a great leading indicator for how a delivery organization as a whole is doing.

We're still big proponents of these metrics, but we've also learned some lessons. We're still observing misguided approaches with tools that help teams measure these metrics based purely on their continuous delivery (CD) pipelines. In particular when it comes to the stability metrics (MTTR and change fail percentage), CD pipeline data alone doesn't provide enough information to determine what a deployment failure with real user impact is. Stability metrics only make sense if they include data about real incidents that degrade service for the users.

We recommend always to keep in mind the ultimate intention behind a measurement and use it to reflect and learn. For example, before spending weeks building up sophisticated dashboard tooling, consider just regularly taking the DORA quick check in team retrospectives. This gives the team the opportunity to reflect on which capabilities they could work on to improve their metrics, which can be much more effective than overdetailed out-of-the-box tooling. Keep in mind that these four key metrics originated out of the organization-level research of high-performing teams, and the use of these metrics at a team level should be a way to reflect on their own behaviors, not just another set of metrics to add to the dashboard.

2. Single team remote wall

Adopt

A single team remote wall is a simple technique to reintroduce the team wall virtually. We recommend that distributed teams adopt this approach; one of the things we hear from teams who moved to remote working is that they miss having the physical team wall. This was a single place where all the various story cards, tasks, status and progress could be displayed, acting as an information radiator and hub for the team. The wall acted as an integration point with the actual data being stored in different systems. As teams have become remote, they've had to revert to looking into the individual source systems and getting an "at a glance" view of a project has become very difficult. While there might be some overhead in keeping this up-to-date, we feel the benefits to the team are worth it. For some teams, updating the physical wall formed part of the daily "ceremonies" the team did together, and the same can be done with a remote wall.

3. Data mesh

Trial

Data mesh is a *decentralized* organizational and technical approach in sharing, accessing and managing data for analytics and ML. Its objective is to create a *sociotechnical* approach that scales out getting value from data as the organization's complexity grows and as the use cases for data proliferate and the sources of data diversify. Essentially, it creates a *responsible* data-sharing model that is in step with organizational growth and continuous change. In our experience, interest in the application of data mesh has grown tremendously. The approach has inspired many organizations to embrace its adoption and technology providers to repurpose their existing technologies for a mesh deployment. Despite the great interest and growing experience in data mesh, its implementations

© Thoughtworks, Inc. All Rights Reserved.

face high cost of integration. Moreover, its adoption remains limited to sections of larger organizations and technology vendors are distracting the organizations from the hard socio aspects of data mesh — decentralized data ownership and a federated governance operating model.

These ideas are explored in ***Data Mesh, Delivering Data-Driven Value at Scale***, which guides practitioners, architects, technical leaders and decision makers on their journeys from traditional big data architecture to data mesh. It provides a complete introduction to data mesh principles and its constituents; it covers how to design a data mesh architecture, guide and execute a data mesh strategy and navigate organizational design to a decentralized data ownership model. The goal of the book is to create a new framework for deeper conversations and lead to the next phase in maturity of data mesh.

4. Definition of production readiness

Trial

In an organization that practices the "you build it, you run it" principle, a definition of production readiness (DPR) is a useful technique to support teams in assessing and preparing the operational readiness of new services. Implemented as a checklist or a template, a DPR gives teams guidance on what to think about and consider before they bring a new service into production. While DPRs do not define specific service-level objectives (SLOs) to fulfill (those would be hard to define one-size-fits-all), they remind teams what categories of SLOs to think of, what organizational standards to comply with and what documentation is required. DPRs provide a source of input that teams turn into respective product-specific requirements around, for example, observability and reliability, to feed into their product backlogs.

DPRs are closely related to Google's concept of a **production readiness review (PRR)**. In organizations that are too small to have a dedicated site reliability engineering team, or who are concerned that a review board process could negatively impact a team's flow to go live, having a DPR can at least provide some guidance and document the agreed-upon criteria for the organization. For highly critical new services, extra scrutiny on fulfilling the DPR can be added via a PRR when needed.

5. Documentation quadrants

Trial

Writing good documentation is an overlooked aspect of software development that is often left to the last minute and done in a haphazard way. Some of our teams have found **documentation quadrants** a handy way to ensure the right artifacts are being produced. This technique classifies artifacts along two axes: The first axis relates to the nature of the information, practical or theoretical; the second axis describes the context in which the artifact is used, studying or working. This defines four quadrants in which artifacts such as tutorials, how-to guides or reference pages can be placed and understood. This classification system not only ensures that critical artifacts aren't overlooked but also guides the presentation of the content. We've found this particularly useful for creating onboarding documentation that brings developers up to speed quickly when they join a new team.

© Thoughtworks, Inc. All Rights Reserved.

6. Rethinking remote standups

Trial

The term *standup* originated from the idea of standing up during this daily sync meeting, with the goal of making it short. It's a common principle many teams try to abide by in their standups: keep it crisp and to the point. But we're now seeing teams challenge that principle and rethinking remote standups. When co-located, there are lots of opportunities during the rest of the day to sync up with each other spontaneously, as a complement to the short standup. Remotely, some of our teams are now experimenting with a longer meeting format, similar to what the folks at Honeycomb call a "**meandering team sync**."

It's not about getting rid of a daily sync altogether; we still find that very important and valuable, especially in a remote setup. Instead, it's about extending the time blocked in everybody's calendars for the daily sync to up to an hour, and use it in a way that makes some of the other team meetings obsolete and brings the team closer together. Activities can still include the well-tried walkthrough of the team board but are then extended by more detailed clarification discussions, quick decisions, and taking time to socialize. The technique is considered successful if it reduces the overall meeting load and improves team bonding.

7. Server-driven UI

Trial

When putting together a new volume of the Radar, we're often overcome by a sense of déjà vu, and the technique of server-driven UI sparks a particularly strong case with the advent of frameworks that allow mobile developers to take advantage of faster change cycles while not falling foul of an app store's policies around revalidation of the mobile app itself. We've blipped about this before from the perspective of enabling mobile development to **scale across teams**. Server-driven UI separates the rendering into a generic container in the mobile app while the structure and data for each view is provided by the server. This means that changes that once required a round trip to an app store can now be accomplished via simple changes to the responses the server sends. Note, we're not recommending this approach for all UI development, indeed we've experienced some horrendous, overly configurable messes, but with the backing of behemoths such as AirBnB and Lyft, we suspect it's not only us at Thoughtworks getting tired of **everything being done client side**. Watch this space.

8. Software Bill of Materials

Trial

With continued pressure to keep systems secure and no reduction in the general threat landscape, a machine-readable Software Bill of Materials (SBOM) may help teams stay on top of security problems in the libraries that they rely on. The recent **Log4Shell** zero-day remote exploit was critical and widespread, and if teams had had an SBOM ready, it could have been scanned for and fixed quickly. We've now had production experience using SBOMs on projects ranging from small companies to large multinationals and even government departments, and we're convinced they provide a benefit. Tools such as **Syft** make it easy to use an SBOM for vulnerability detection.

9. Tactical forking

Trial

Tactical forking is a technique that can assist with restructuring or migrating from monolithic codebases to microservices. Specifically, this technique offers one possible alternative to the more common approach of fully modularizing the codebase first, which in many circumstances can take a

© Thoughtworks, Inc. All Rights Reserved.

very long time or be very challenging to achieve. With tactical forking a team can create a new fork of the codebase and use that to address and extract one particular concern or area while deleting the unnecessary code. Use of this technique would likely be just one part of a longer-term plan for the overall monolith.

10. Team cognitive load

Trial

A system's architecture mimics an organizational structure and its communication. It's not big news that we should be intentional about how teams interact — see, for instance, the Inverse Conway Maneuver. Team interaction is one of the variables for how fast and how easily teams can deliver value to their customers. We were happy to find a way to measure these interactions; we used the Team Topologies author's assessment which gives you an understanding of how easy or difficult the teams find it to build, test and maintain their services. By measuring team cognitive load, we could better advise our clients on how to change their teams' structure and evolve their interactions.

11. Transitional architecture

Trial

A transitional architecture is a useful practice used when replacing legacy systems. Much like scaffolding might be built, reconfigured and finally removed during construction or renovation of a building, you often need interim architectural steps during legacy displacement. Transitional architectures will be removed or replaced later on, but they're not just throwaway work given the important role they play in reducing risk and allowing a difficult problem to be broken into smaller steps. Thus they help with avoiding the trap of defaulting to a "big bang" legacy replacement approach, because you cannot make smaller interim steps line up with a final architectural vision. Care is needed to make sure the architectural "scaffolding" is eventually removed, lest it just become technical debt later on.

12. CUPID

Assess

How do you approach writing good code? How do you judge if you've written good code? As software developers, we're always looking for catchy rules, principles and patterns that we can use to share a language and values with each other when it comes to writing simple, easy-to-change code.

Daniel Terhorst-North has recently made a new attempt at creating such a checklist for good code. He argues that instead of sticking to a set of rules like SOLID, using a set of properties to aim for is more generally applicable. He came up with what he calls the CUPID properties to describe what we should strive for to achieve "joyful" code: Code should be composable, follow the Unix philosophy and be predictable, idiomatic and domain based.

13. Inclusive design

Assess

We recommend organizations assess inclusive design as a way of making sure accessibility is treated as a first-class requirement. All too often requirements around accessibility and inclusivity are ignored until just before, if not just after, the release of software. The cheapest and simplest way to accommodate these requirements, while also providing early feedback to teams, is to incorporate

© Thoughtworks, Inc. All Rights Reserved.

them fully into the development process. In the past, we've highlighted techniques that perform a "shift-left" for security and cross-functional requirements; one perspective on this technique is that it achieves the same goal for accessibility.

14. Operator pattern for nonclustered resources

Assess

We're continuing to see increasing use of the Kubernetes Operator pattern for purposes other than managing applications deployed on the cluster. Using the Operator pattern for nonclustered resources takes advantage of custom resource definitions and the event-driven scheduling mechanism implemented in the Kubernetes control plane to manage activities that are related to yet outside of the cluster. This technique builds on the idea of Kube-managed cloud services and extends it to other activities, such as continuous deployment or reacting to changes in external repositories. One advantage of this technique over a purpose-built tool is that it opens up a wide range of tools that either come with Kubernetes or are part of the wider ecosystem. You can use commands such as diff, dry-run or apply to interact with the operator's custom resources. Kube's scheduling mechanism makes development easier by eliminating the need to orchestrate activities in the proper order. Open-source tools such as Crossplane, Flux and Argo CD take advantage of this technique, and we expect to see more of these emerge over time. Although these tools have their use cases, we're also starting to see the inevitable misuse and overuse of this technique and need to repeat some old advice: Just because you *can* do something with a tool doesn't mean you *should*. Be sure to rule out simpler, conventional approaches before creating a custom resource definition and taking on the complexity that comes with this approach.

15. Service mesh without sidecar

Assess

Service mesh is usually implemented as a reverse-proxy process, aka sidecar, deployed alongside each service instance. Although these sidecars are lightweight processes, the overall cost and operational complexity of adopting service mesh increases with every new instance of the service requiring another sidecar. However, with the advancements in eBPF, we're observing a new service mesh without sidecar approach where the functionalities of the mesh are safely pushed down to the OS kernel, thereby enabling services in the same node to communicate transparently via sockets without the need of additional proxies. You can try this with Cilium service mesh and simplify the deployment from one proxy-per-service to one proxy-per-node. We're intrigued by the capabilities of eBPF and find this evolution of service mesh to be important to assess.

16. SLSA

Assess

As software continues to grow in complexity, the threat vector of software dependencies becomes increasingly challenging to guard against. The recent Log4J vulnerability showed how difficult it can be to even know those dependencies — many companies who didn't use Log4J directly were unknowingly vulnerable simply because other software in their ecosystem relied on it. Supply chain Levels for Software Artifacts, or SLSA (pronounced "salsa"), is a consortium-curated set of guidance for organizations to protect against supply chain attacks, evolved from internal guidance Google has been using for years. We appreciate that SLSA doesn't promise a "silver bullet," tools-only approach to securing the supply chain but instead provides a checklist of concrete threats and practices along a maturity model. The threat model is easy to follow with real-world examples of attacks, and the requirements provide guidance to help organizations prioritize actions based on levels of increasing robustness to improve their supply chain security posture. We think SLSA provides applicable advice and look forward to more organizations learning from it.

© Thoughtworks, Inc. All Rights Reserved.

17. The streaming data warehouse

Assess

The need to respond quickly to customer insights has driven increasing adoption of event-driven architectures and stream processing. Frameworks such as Spark, Flink or Kafka Streams offer a paradigm where simple event consumers and producers can cooperate in complex networks to deliver real-time insights. But this programming style takes time and effort to master and when implemented as single-point applications, it lacks interoperability. Making stream processing work universally on a large scale can require a significant engineering investment. Now, a new crop of tools is emerging that offers the benefits of stream processing to a wider, established group of developers who are comfortable using SQL to implement analytics. Standardizing on SQL as the universal streaming language lowers the barrier for implementing streaming data applications. Tools like ksqlDB and Materialize help transform these separate applications into unified platforms. Taken together, a collection of SQL-based streaming applications across an enterprise might constitute a streaming data warehouse.

18. TinyML

Assess

Until recently, executing a machine-learning (ML) model was seen as computationally expensive and in some cases required special-purpose hardware. While creating the models still broadly sits within this classification, they can be created in a way that allows them to be run on small, low-cost and low-power consumption devices. This technique, called TinyML, has opened up the possibility of running ML models in situations many might assume infeasible. For example, on battery-powered devices, or in disconnected environments with limited or patchy connectivity, the model can be run locally without prohibitive cost. If you've been considering using ML but thought it unrealistic because of compute or network constraints, then this technique is worth assessing.

19. Azure Data Factory for orchestration

Hold

For organizations using Azure as their primary cloud provider, Azure Data Factory is currently the default for orchestrating data-processing pipelines. It supports data ingestion, copying data from and to different storage types on prem or on Azure and executing transformation logic. Although we've had adequate experience with Azure Data Factory for simple migrations of data stores from on prem to the cloud, we discourage the use of Azure Data Factory for orchestration of complex data-processing pipelines and workflows. We've had some success with Azure Data Factory when it's used primarily to move data between systems. For more complex data pipelines, it still has its challenges, including poor debuggability and error reporting; limited observability as Azure Data Factory logging capabilities don't integrate with other products such as Azure Data Lake Storage or Databricks, making it difficult to get an end-to-end observability in place; and availability of data source-triggering mechanisms only to certain regions. At this time, we encourage using other open-source orchestration tools (e.g., Airflow) for complex data pipelines and limiting Azure Data Factory for data copying or snapshotting. Our teams continue to use Data Factory to move and extract data, but for larger operations we recommend other, more well-rounded workflow tools.

© Thoughtworks, Inc. All Rights Reserved.

20. Miscellaneous platform teams

Hold

We previously featured **platform engineering product teams** in Adopt as a good way for internal platform teams to operate, thus enabling delivery teams to self-service deploy and operate systems with reduced lead time and stack complexity. Unfortunately we're seeing the "platform team" label applied to teams dedicated to projects that don't have clear outcomes or a well-defined set of customers. As a result, these miscellaneous platform teams, as we call them, struggle to deliver due to high cognitive loads and a lack of clearly aligned priorities as they're dealing with a miscellaneous collection of unrelated systems. They effectively become just another general support team for things that don't fit or that are unwanted elsewhere. We continue to believe platform engineering product teams focused around a clear and well-defined (internal) product offer a better set of outcomes.

21. Production data in test environments

Hold

We continue to perceive production data in test environments as an area for concern. Firstly, many examples of this have resulted in reputational damage, for example, where an incorrect alert has been sent from a test system to an entire client population. Secondly, the level of security, specifically around protection of private data, tends to be less for test systems. There is little point in having elaborate controls around access to production data if that data is copied to a test database that can be accessed by every developer and QA. Although you can obfuscate the data, this tends to be applied only to specific fields, for example, credit card numbers. Finally, copying production data to test systems can break privacy laws, for example, where test systems are hosted or accessed from a different country or region. This last scenario is especially problematic with complex cloud deployments. Fake data is a safer approach, and tools exist to help in its creation. We do recognize there are reasons for specific elements of production data to be copied, for example, in the reproduction of bugs or for training of specific ML models. Here our advice is to proceed with caution.

22. SPA by default

Hold

We generally avoid putting blips in Hold when we consider that advice too obvious, including blindly following an architectural style without paying attention to trade-offs. However, the sheer prevalence of teams choosing a single-page application (SPA) by default when they need a website has us concerned that people aren't even recognizing SPAs as an architectural style to begin with, instead immediately jumping into framework selection. SPAs incur complexity that simply doesn't exist with traditional server-based websites: search engine optimization, browser history management, web analytics, first page load time, etc. That complexity is often warranted for user experience reasons, and tooling continues to evolve to make those concerns easier to address (although the churn in the React community around state management hints at how hard it can be to get a generally applicable solution). Too often, though, we don't see teams making that trade-off analysis, blindly accepting the complexity of SPAs by default even when the business needs don't justify it. Indeed, we've started to notice that many newer developers aren't even aware of an alternative approach, as they've spent their entire career in a framework like React. We believe that many websites will benefit from the simplicity of server-side logic, and we're encouraged by techniques like **Hotwire** that help close the gap on user experience.

© Thoughtworks, Inc. All Rights Reserved.

Platforms

Adopt

—

Trial

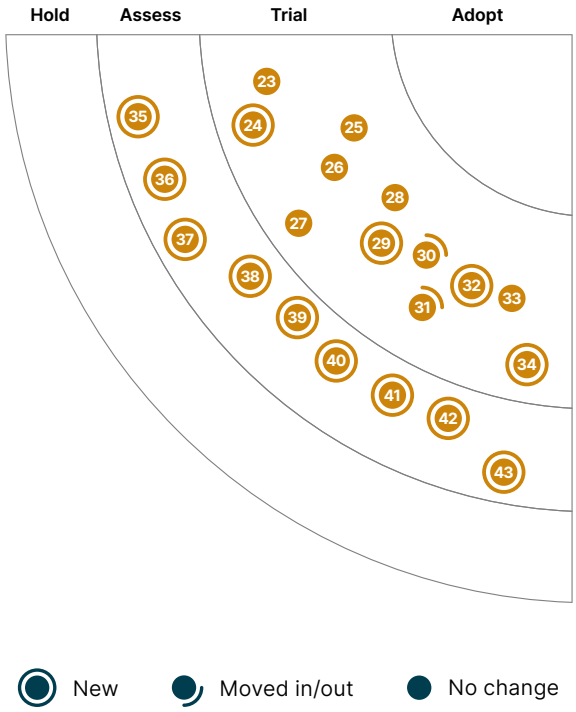
23. Azure DevOps
24. Azure Pipeline templates
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Reusable workflows in Github Actions
33. Sealed Secrets
34. VerneMQ

Assess

35. actions-runner-controller
36. Apache Iceberg
37. Blueboat
38. Cloudflare Pages
39. Colima
40. Collibra
41. CycloneDX
42. Embeddinghub
43. Temporal

Hold

—



23. Azure DevOps

Trial

As the Azure DevOps ecosystem keeps growing, our teams are using it more with success. These services contain a set of managed services, including hosted Git repos, build and deployment pipelines, automated testing tooling, backlog management tooling and artifact repository. We've seen our teams gaining 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. As the ecosystem matures, we're seeing an uptick in onboarding teams that are already on the Azure stack as it easily integrates with the rest of the Microsoft world.

24. Azure Pipeline templates

Trial

Azure Pipeline templates allow you to remove duplication in your Azure Pipeline definition through two mechanisms. With "includes" templates, you can reference a template such that it will expand inline like a parameterized C++ macro, allowing a simple way of factoring out common configuration across stages, jobs and steps. With "extends" templates, you can define an outer shell with common pipeline configuration, and with the required template approval, you can fail the build if the pipeline doesn't extend certain templates, preventing malicious attacks against the pipeline configuration itself. Along with CircleCI Orbs and the newer GitHub Actions Reusable Workflows, Azure Pipeline templates are part of the trend of creating modularity in pipeline design across multiple platforms, and several of our teams have been happy using them.

25. CircleCI

Trial

Many of our teams choose CircleCI for their continuous integration needs, and they appreciate its ability to run complex pipelines efficiently. The CircleCI developers continue to add new features with CircleCI, now in version 3.0. Orbs and executors were called out by our teams as being particularly useful. Orbs are reusable snippets of code that automate repeated processes, speed up project setup and make it easy to integrate with third-party tools. The wide variety of executor types provides flexibility to set up jobs in Docker, Linux, macOS or Windows VMs.

26. Couchbase

Trial

When we originally blipped Couchbase in 2013, it was seen primarily as a persistent cache that evolved from a merger of Membase and CouchDB. Since then, it has undergone steady improvement and an ecosystem of related tools and commercial offerings has grown up around it. Among the additions to the product suite are Couchbase Mobile and the Couchbase Sync Gateway. These features work together to keep persistent data on edge devices up-to-date even when the device is offline for periods of time due to intermittent connectivity. As these devices proliferate, we see increasing need for embedded persistence that continues to work whether or not the device happens to be connected. Recently, one of our teams evaluated Couchbase for its offline sync capability and found that this off-the-shelf capability saved them considerable effort that they otherwise would have had to invest themselves.

© Thoughtworks, Inc. All Rights Reserved.

27. eBPF

Trial

For several years now, the Linux kernel has included the extended Berkeley Packet Filter (eBPF), a virtual machine that provides the ability to attach filters to particular sockets. But eBPF goes far beyond packet filtering and allows custom scripts to be triggered at various points within the kernel with very little overhead. Although this technology isn't new, it's now coming into its own with the increasing use of microservices deployed as orchestrated containers. Kubernetes and service mesh technology such as Istio are commonly used, and they employ sidecars to implement control functionality. With new tools — Bumblebee in particular makes building, running and distributing eBPF programs much easier — eBPF can be seen as an alternative to the traditional sidecar. A maintainer of Cilium, a tool in this space, has even proclaimed the demise of the sidecar. An approach based on eBPF reduces some overhead in performance and operation that comes with sidecars, but it doesn't support common features such as SSL termination.

28. GitHub Actions

Trial

GitHub Actions has grown considerably last year. It has proven that it can take on more complex workflows and call other actions in composite actions among other things. It still has some shortcomings, though, such as its inability to re-trigger a single job of a workflow. Although the ecosystem in the GitHub Marketplace has its obvious advantages, giving third-party GitHub Actions access to your build pipeline risks sharing secrets in insecure ways (we recommend following GitHub's advice on security hardening). However, the convenience of creating your build workflow directly in GitHub next to your source code combined with the ability to run GitHub Actions locally using open-source tools such as act is a compelling option that has facilitated setup and onboarding of our teams.

29. GitLab CI/CD

Trial

If you're using GitLab to manage your software delivery, you should also look at GitLab CI/CD for your continuous integration and continuous delivery needs. We've found it especially useful when used with on-premise GitLab and self-hosted runners, as this combination gets around authorization headaches often caused by using a cloud-based solution. Self-hosted runners can be fully configured for your purposes with the right OS and dependencies installed, and as a result pipelines can run much faster than using a cloud-provisioned runner that needs to be configured each time.

Apart from the basic build, test and deploy pipeline, GitLab's product supports Services, Auto Devops and ChatOps among other advanced features. Services are useful in running Docker services such as Postgres or Testcontainer linked to a job for integration and end-to-end testing. Auto Devops creates pipelines with zero configuration which is very useful for teams that are new to continuous delivery or for organizations with many repositories that would otherwise need to create many pipelines manually.

30. Google BigQuery ML

Trial

Since we last blipped about Google BigQuery ML, more sophisticated models such as Deep Neural Networks and AutoML Tables have been added by connecting BigQuery ML with TensorFlow and Vertex AI as its backend. BigQuery has also introduced support for time series forecasting. One

© Thoughtworks, Inc. All Rights Reserved.

of our concerns previously was explainability. Earlier this year, BigQuery Explainable AI was announced for general availability, taking a step in addressing this. We can also export BigQuery ML models to Cloud Storage as a Tensorflow SavedModel and use them for online prediction. There remain trade-offs like ease of "continuous delivery for machine learning" but with its low barrier to entry, BigQuery ML remains an attractive option, particularly when the data already resides in BigQuery.

31. Google Cloud Dataflow

Trial

Google Cloud Dataflow is a cloud-based data-processing service for both batch and real-time data-streaming applications. Our teams are using Dataflow to create processing pipelines for integrating, preparing and analyzing large data sets, with Apache Beam's unified programming model on top to ease manageability. We first featured Dataflow in 2018, and its stability, performance and rich feature set make us confident to move it to Trial in this edition of the Radar.

32. Reusable workflows in Github Actions

Trial

We've seen increased interest in GitHub Actions since we first blipped it two Radars ago. With the release of reusable workflows, GitHub continues to evolve the product in a way that addresses some of its early shortcomings. Reusable workflows in Github Actions bring modularity to pipeline design, allowing parameterized reuse even across repositories (as long as the workflow repository is public). They support explicit passing of confidential values as secrets and can pass outputs to the calling job. With a few lines of YAML, GitHub Actions now gives you the type of flexibility you see with CircleCI Orbs or Azure Pipeline Templates, but without having to leave GitHub as a platform.

33. Sealed Secrets

Trial

Kubernetes natively supports a key-value object known as a secret. However, by default, Kubernetes secrets aren't really secret. They're handled separately from other key-value data so that precautions or access control can be applied separately. There is support for encrypting secrets before they are stored in etcd, but the secrets start out as plain text fields in configuration files. Sealed Secrets is a combination operator and command-line utility that uses asymmetric keys to encrypt secrets so that they can only be decrypted by the controller in the cluster. This process ensures that the secrets won't be compromised while they sit in the configuration files that define a Kubernetes deployment. Once encrypted, these files can be safely shared or stored alongside other deployment artifacts.

34. VerneMQ

Trial

VerneMQ is an open-source, high-performance, distributed MQTT broker. We've blipped other MQTT brokers in the past like Mosquitto and EMQ. Like EMQ and RabbitMQ, VerneMQ is also based on Erlang/OTP which makes it highly scalable. It scales horizontally and vertically on commodity hardware to support a high number of concurrent publishers and consumers while maintaining low latency and fault tolerance. In our internal benchmarks, we've been able to achieve a few million concurrent connections in a single cluster. While it's not new, we've used it in production for some time now, and it has worked well for us.

© Thoughtworks, Inc. All Rights Reserved.

35. actions-runner-controller

Assess

actions-runner-controller is a Kubernetes controller that operates self-hosted runners for GitHub Actions on your Kubernetes cluster. With this tool you create a runner resource on Kubernetes, and it will run and operate the self-hosted runner. Self-hosted runners are helpful in scenarios where the job that your GitHub Actions runs needs to access resources that are either not accessible to GitHub cloud runners or have specific operating system and environmental requirements that are different from what GitHub provides. In those cases where you have a Kubernetes cluster, you can run your self-hosted runners as a Kubernetes pod, with the ability to scale up or down hooking into GitHub webhook events. actions-controller-runner is lightweight and scalable.

36. Apache Iceberg

Assess

Apache Iceberg is an open table format for very large analytic data sets. Iceberg supports modern analytical data operations such as record-level insert, update, delete, time-travel queries, ACID transactions, hidden partitioning and full schema evolution. It supports multiple underlying file storage formats such as Apache Parquet, Apache ORC and Apache Avro. Many data-processing engines support Apache Iceberg, including SQL engines such as Dremio and Trino as well as (structured) streaming engines such as Apache Spark and Apache Flink.

Apache Iceberg falls in the same category as Delta Lake and Apache Hudi. They all more or less support similar features, but each differs in the underlying implementations and detailed feature lists. Iceberg is an independent format and is not native to any specific processing engine, hence it's supported by an increasing number of platforms, including AWS Athena and Snowflake. For the same reason, Apache Iceberg, unlike native formats such as Delta Lake, may not benefit from optimizations when used with Spark.

37. Blueboat

Assess

Blueboat is a multitenant platform for serverless web applications. It leverages the popular V8 JavaScript engine and implements commonly used web application libraries natively in Rust for security and performance. You can think of Blueboat as an alternative to CloudFlare Workers or Deno Deploy but with an important distinction — you have to operate and manage the underlying infrastructure. That said, we recommend you carefully assess Blueboat for your on-prem serverless needs.

38. Cloudflare Pages

Assess

When Cloudflare Workers was released, we highlighted it as an early function as a service (FaaS) for edge computing with an interesting implementation. The release of Cloudflare Pages last April didn't feel as noteworthy, because Pages is just one in a class of many Git-backed site-hosting solutions. It did have continuous previews, a useful feature not found in most alternatives. Now, though, Cloudflare has more tightly integrated Workers and Pages, creating a fully integrated Jamstack solution running on the CDN. The inclusion of a key-value store and a strongly consistent coordination primitive further enhance the attractiveness of the new version of Cloudflare Pages.

© Thoughtworks, Inc. All Rights Reserved.

39. Colima

Assess

Colima is becoming a popular open alternative to Docker for Desktop. It provisions the Docker container runtime in a Lima VM, configures the Docker CLI on macOS and handles port-forwarding and volume mounts. Colima uses containerd as runtime, which is also the runtime on most managed Kubernetes services (thus improved dev-prod parity). With Colima you can easily use and test the latest features of containerd, such as lazy loading for container images. With its good performance, we're watching Colima as a strong potential for the open-source choice alternative to Docker for Desktop.

40. Collibra

Assess

In the increasingly crowded space that is the enterprise data catalog market, our teams have enjoyed working with Collibra. They liked the deployment flexibility of either a SaaS or self-hosted instance, the wide range of functionality included out of the box, including data governance, lineage, quality and observability. Users also have the option to use a smaller subset of capabilities required by a more decentralized approach such as a data mesh. The real feather in its cap has been their often overlooked customer support, which our people have found to be collaborative and supportive. Of course, there's a tension between simple data catalogs and more full featured enterprise platforms, but so far the teams using it are happy with how Collibra has supported their needs.

41. CycloneDX

Assess

CycloneDX is a standard for describing a machine-readable Software Bill of Materials (SBOM). As software and compute fabrics increase in complexity, *software* becomes harder to define. Originating with OWASP, CycloneDX improves on the older SPDX standard with a broader definition that extends beyond the local machine dependencies to include runtime service dependencies. You'll also find implementations in several languages, an ecosystem of supporting integrations and a CLI tool that lets you analyze and change SBOMs with appropriate signing and verification.

42. Embeddinghub

Assess

Embeddinghub is a vector database for machine-learning embeddings, and quite similar to Milvus. However, with out-of-the-box support for approximate nearest neighbor operations, partitioning, versioning and access control, we recommend you assess Embeddinghub for your embedding vector use cases.

43. Temporal

Assess

Temporal is a platform for developing long-running workflows, particularly for microservice architectures. A fork of Uber's previous OSS Cadence project, it has an event-sourcing model for long-running workflows so they can survive process/machine crashes. Although we don't recommend using distributed transactions in microservice architectures, if you do need to implement them or long-running Sagas, you may want to look at Temporal.

© Thoughtworks, Inc. All Rights Reserved.

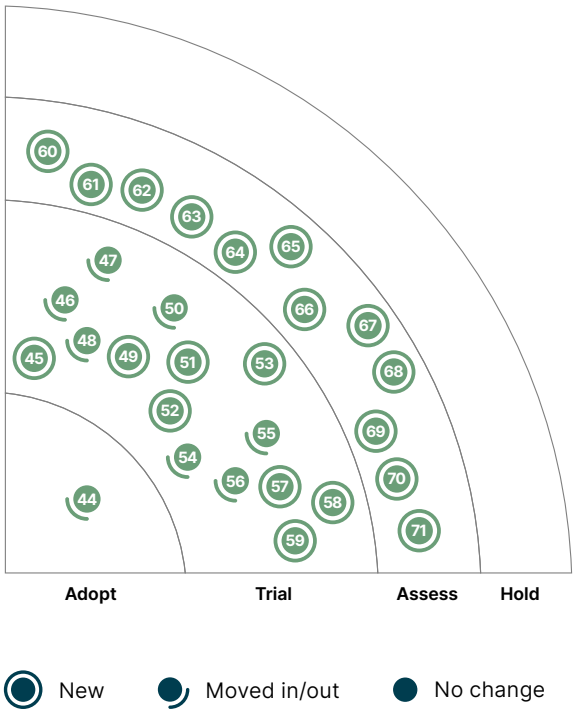
Tools

Adopt
44. tfsec

Trial
45. AKHQ
46. cert-manager
47. Cloud Carbon Footprint
48. Conftest
49. kube-score
50. Lighthouse
51. Metaflow
52. Micrometer
53. NUKE
54. Pactflow
55. Podman
56. Sourcegraph
57. Syft
58. Volta
59. Web Test Runner

Assess
60. CDKTF
61. Chrome Recorder panel
62. Excalidraw
63. GitHub Codespaces
64. GoReleaser
65. Grype
66. Infracost
67. jc
68. skopeo
69. SQLFluff
70. Terraform Validator
71. Typesense

Hold
—



44. tfsec

Adopt

For our projects using Terraform, tfsec has quickly become a default static analysis tool to detect potential security risks. It's easy to integrate into a CI pipeline and has a growing library of checks against all of the major cloud providers and platforms like Kubernetes. Given its ease of use, we believe tfsec could be a good addition to any Terraform project.

45. AKHQ

Trial

AKHQ is a GUI for Apache Kafka that lets you manage topics, topics data, consumer groups and more. Some of our teams have found AKHQ to be an effective tool to watch the real-time status of a Kafka cluster. You can, for example, browse the topics on a cluster. For each topic, you can visualize the name, the number of messages stored, the disk size used, the time of the last record, the number of partitions, the replication factor with the in-sync quantity and the consumer group. With options for Avro and Protobuf deserialization, AKHQ can help you understand the flow of data in your Kafka environment.

46. cert-manager

Trial

cert-manager is a tool to manage your X.509 certificates within your Kubernetes cluster. It models certificates and issuers as first-class resource types and provides certificates as a service securely to developers and applications working within the Kubernetes cluster. The obvious choice when using the Kubernetes default ingress controller, it's also recommended for others and much preferred over hand-rolling your own certificate management. Several of our teams have been using cert-manager extensively, and we've also found that its usability has much improved in the past few months.

47. Cloud Carbon Footprint

Trial

Cloud Carbon Footprint (CCF) is an open-source tool that uses cloud APIs to provide visualizations of estimated carbon emissions based on usage across AWS, GCP and Azure. The Thoughtworks team has successfully used the tool with several organizations, including energy technology companies, retailers, digital service providers and companies that use AI. Cloud platform providers realize that it's important to help their customers understand the carbon impact of using their services, so they've begun to build similar functionality themselves. Because CCF is cloud agnostic, it allows users to view energy usage and carbon emissions for multiple cloud providers in one place, while translating carbon footprints into real-world impact such as flights or trees planted.

In recent releases, CCF has begun to include Google Cloud and AWS-sourced optimization recommendations alongside potential energy and CO2 savings, as well as to support more cloud instance types such as GPU instances. Given the traction the tool has received and the continued addition of new features, we feel confident moving it to Trial.

© Thoughtworks, Inc. All Rights Reserved.

48. Conftest

Trial

Conftest is a tool for writing tests against structured configuration data. It relies on the Rego language from Open Policy Agent to write tests for Kubernetes configurations, Tekton pipeline definitions or even Terraform plans. We've had great experiences with Conftest — and its shallow learning curve. With fast feedback from tests, our teams iterate quickly and safely on configuration changes to Kubernetes.

49. kube-score

Trial

kube-score is a tool that does static code analysis of your Kubernetes object definitions. The output is a list of recommendations for what you can improve to make your application more secure and resilient. It has a list of predefined checks which includes best practices such as running containers with non-root privileges and correctly specifying resource limits. It's been around for some time, and we've used it in a few projects as part of a CD pipeline for Kubernetes manifests. A major drawback of kube-score is that you can't add custom policies. We typically supplement it with tools like Conftest in these cases.

50. Lighthouse

Trial

Lighthouse is a tool written by Google to assess web applications and web pages, collecting performance metrics and insights on good development practices. We've long advocated for performance testing as a first-class citizen, and the additions to Lighthouse that we mentioned five years ago certainly helped with that. Our thinking around architectural fitness functions created strong motivation for tools such as Lighthouse to be run in build pipelines. With the introduction of Lighthouse CI, it has become easier than ever to include Lighthouse in pipelines managed by various tools.

51. Metaflow

Trial

Metaflow is a user-friendly Python library and back-end service that helps data scientists and engineers build and manage production-ready data processing, ML training and inference workflows. Metaflow provides Python APIs that structure the code as a directed graph of steps. Each step can be decorated with flexible configurations such as the required compute and storage resources. Code and data artifacts for each step's run (aka task) are stored and can be retrieved either for future runs or the next steps in the flow, enabling you to recover from errors, repeat runs and track versions of models and their dependencies across multiple runs.

The value proposition of Metaflow is the simplicity of its idiomatic Python library: it fully integrates with the build and run-time infrastructure to enable running data engineering and science tasks in local and scaled production environments. At the time of writing, Metaflow is heavily integrated with AWS services such as S3 for its data store service and step functions for orchestration. Metaflow supports R in addition to Python. Its core features are open sourced.

If you're building and deploying your production ML and data-processing pipelines on AWS, Metaflow is a lightweight full-stack alternative framework to more complex platforms such as MLflow.

© Thoughtworks, Inc. All Rights Reserved.

52. Micrometer

Trial

Micrometer is a platform-agnostic library for metrics instrumentation on the JVM that supports Graphite, New Relic, CloudWatch and many other integrations. We've found that Micrometer has benefited both library authors and teams: library authors can include metrics instrumentation code in their libraries without needing to support each and every metrics system that their users are using; and teams can support many different metrics on back-end registries which enables organizations to collect metrics in a consistent way.

53. NUKE

Trial

NUKE is a build system for .NET and an alternative to either the traditional MSBuild or Cake and Fake which we've featured previously in the Radar. NUKE represents build instructions as a C# DSL, making it easy to learn and with good IDE support. In our experience, NUKE made it really simple to build automation for .NET projects. We like the accurate static code checks and hints. We also like that we can use any NuGet package seamlessly and that the automation code can be compiled to avoid problems at runtime. NUKE isn't new, but its novel approach — using a C# DSL — and our positive overall experience prompted us to include it here.

54. Pactflow

Trial

We've used Pact for contract testing long enough to see some of the complexity that comes with scale. Some of our teams have successfully used Pactflow to reduce that friction. Pactflow runs both as software as a service and as an on-prem deployment with the same features as the SaaS offering, and it adds improved usability, security and auditing on top of the open-source Pact Broker offering. We've been pleased with our use so far and are happy to see continued effort to remove some of the overhead of managing contract testing at scale.

55. Podman

Trial

As an alternative to Docker, Podman has been validated by many of our teams. Podman introduces a daemonless engine for managing and running containers which is an interesting approach in comparison to what Docker does. Additionally, Podman can be easily run as a normal user without requiring root privileges, which reduces the attack surface. By using either Open Container Initiative (OCI) images built by Buildah or Docker images, Podman can be adapted to most container use cases. Apart from some compatibility issues with macOS, our team has had generally good experiences with Podman on Linux distributions.

56. Sourcegraph

Trial

In our previous Radar, we featured two tools that search and replace code using an abstract syntax tree (AST) representation, Comby and Sourcegraph. Although they share some similarities, they also differ in several ways. Sourcegraph is a commercial tool (with a 10-user free tier). It's particularly suited for searching, navigating or cross-referencing in large codebases, with an emphasis on an interactive developer experience. In contrast, Comby is a lightweight open-source command-line tool for automating repetitive tasks. Because Sourcegraph is a hosted service, it also has the ability

© Thoughtworks, Inc. All Rights Reserved.

to continuously monitor code bases and send alerts when a match occurs. Now that we've gained more experience with Sourcegraph, we decided to move it into the Trial ring to reflect our positive experience — which doesn't mean that Sourcegraph is better than Comby. Each tool focuses on a different niche.

57. Syft

Trial

One of the key elements of improving "supply chain security" is using a Software Bill of Materials (SBOM), which is why publishing an SBOM along with the software artifact is increasingly important. Syft is a CLI tool and Go library for generating an SBOM from container images and file systems. It can generate the SBOM output in multiple formats, including JSON, CycloneDX and SPDX. The SBOM output of Syft can be used by Grype for vulnerability scanning. One way to publish the generated SBOM along with the image is to add it as an attestation using Cosign. This allows consumers of the image to verify the SBOM and to use it for further analysis.

58. Volta

Trial

When working on multiple JavaScript codebases at the same time, it's often necessary to use different versions of Node and other JavaScript tools. On developer machines, these tools are usually installed in the user account or the machine itself, which means a solution is needed to switch between multiple installations. For Node itself there's nvm, but we want to highlight Volta as an alternative that we're seeing in use with our teams. Volta has several advantages over using nvm: it can manage other JavaScript tools such as Yarn; it also has the notion of pinning a version of the toolchain on a project basis, which means that developers can simply use the tools in a given code directory without having to worry about manually switching between tool versions — Volta simply uses shims in the path to select the pinned version. Written in Rust, Volta is fast and ships as a single binary without dependencies.

59. Web Test Runner

Trial

Web Test Runner is a package within the Modern Web project, which provides several high-quality tools for modern web development with support for web standards like ES Modules. Web Test Runner is a test runner for web applications. One of its advantages compared to existing test runners is that it runs tests in the browser (which could be headless). It supports multiple browser launchers — including Puppeteer, Playwright, and Selenium — and uses Mocha by default for the test framework. The tests run pretty fast, and we like that we can open a browser window with devtools when debugging. Web Test Runner internally uses Web Dev Server which allows us to leverage its great plugin API for adding customized plugins for our test suite. Modern Web tools look like a very promising developer toolchain, and we're already using it in a few projects.

60. CDKTF

Assess

By now many organizations have created sprawling landscapes of services in the cloud. Of course, this is only possible when using infrastructure as code and mature tooling. We still like Terraform, not the least because of its rich and growing ecosystem. However, the lack of abstractions in HCL, Terraform's default configuration language, effectively creates a glass ceiling. Using Terragrunt

© Thoughtworks, Inc. All Rights Reserved.

pushes that up a bit further, but more and more often our teams find themselves longing for the abstractions afforded by modern programming languages. Cloud Development Kit for Terraform (CDKTF), which resulted from a collaboration between AWS's CDK team and Hashicorp, makes it possible for teams to use several programming languages, including TypeScript and Java, to define and provision infrastructure. With this approach it follows the lead of Pulumi while remaining in the Terraform ecosystem. We've had good experiences with CDKTF but have decided to keep it in the Assess ring until it moves out of beta.

61. Chrome Recorder panel

Assess

Chrome Recorder panel is a preview feature in Google Chrome 97 that allows for simple record and playback of user journeys. While this definitely isn't a new idea, the way in which it is integrated into Chrome allows for quick creation, editing and running of scripts. The panel also integrates nicely with the performance panel, which makes getting repeated consistent feedback on page performance easier. While record/playback style testing always needs to be used with care in order to avoid brittle tests, we think this preview feature is worth assessing, especially if you're already using the Chrome Performance panel to measure your pages.

62. Excalidraw

Assess

Excalidraw is a simple but powerful online drawing tool that our teams enjoy using. Sometimes teams just need a quick picture instead of a formal diagram, for remote teams Excalidraw provides a quick way to create and share diagrams. Our teams also like the "lo-fi" look of the diagrams it can produce, which is reminiscent of the whiteboard diagrams they would have produced when co-located. One caveat: you need to pay attention to the default security — at the time of writing, anyone who has the link can see the diagram. A paid-for version provides further authentication.

63. GitHub Codespaces

Assess

GitHub Codespaces allows developers to create development environments in the cloud and access them through an IDE as though the environment were local. GitHub isn't the first company to implement this idea; we previously blipped about Gitpod. We like that Codespaces allows environments to be standardized by using dotfiles configuration, making it quicker to onboard new team members, and that they offer VMs with up to 32 cores and 64GB memory. These VMs can be spun up in under ten seconds, potentially offering environments more powerful than a developer laptop.

64. GoReleaser

Assess

GoReleaser is a tool that automates the process of building and releasing a Go project for different architectures via multiple repositories and channels, a common need for Go projects targeting different platforms. You run the tool either from your local machine or via CI, with the tool available via several CI services thus minimizing set-up and maintenance. GoReleaser takes care of build, packaging, publishing and announcement of each release and supports different combinations of package format, package repository and source control. Although it's been around for a few years, we're surprised that more teams are not using it. If you're regularly releasing a Go codebase, this tool is worth assessing.

© Thoughtworks, Inc. All Rights Reserved.

65. Grype

Assess

Securing the software supply chain has become a commonplace concern among delivery teams, a concern that is reflected by the growing number of new tools in this space. Grype is a new lightweight vulnerability scanning tool for Docker and OCI images. It can be installed as a binary, can scan images before they're pushed to a registry, and it doesn't require a Docker daemon to run on your build agents. Grype comes from the same team that is behind Syft, which generates SBOMs in various formats from container images. Grype can consume the SBOM output of Syft to scan for vulnerabilities.

66. Infracost

Assess

One often-cited advantage of moving to the cloud is transparency around infrastructure spend. In our experience, this is often not the case. Teams don't always think about the decisions they make around infrastructure in terms of financial cost which is why we previously blipped about run cost as architecture fitness function. We're intrigued by the release of a new tool called Infracost which aims to make cost trade-offs visible in Terraform pull requests. It's open-source software and available for macOS, Linux, Windows and Docker and supports pricing for AWS, GCP and Microsoft Azure out of the box. It also provides a public API that can be queried for current cost data. Our teams are excited by its potential, especially when it comes to gaining better cost visibility in the IDE.

67. jc

Assess

In our previous Radar, we placed modern Unix commands in Assess. One of the commands featured in that collection of tools was jq, effectively a sed for JSON. jc performs a related task: it takes the output of common Unix commands and parses the output into JSON. The two commands together provide a bridge between the Unix CLI world and the raft of libraries and tools that operate on JSON. When writing *simple* scripts, for example, for software deployment or gathering troubleshooting information, having the myriad of different Unix command output formats mapped into well-defined JSON can save a lot of time and effort. As with jq, you need to make sure the command is available. It can be installed from many of the well-known package repositories.

68. skopeo

Assess

skopeo is a command line utility that performs various operations on container images and image repositories. It doesn't require a user to be root to do most of its operations nor does it require a daemon to be running. It's a useful part of a CI pipeline; we've used it to copy images from one registry to another as we promote the images. It's better than doing a pull and a push as we don't need to store the images locally. It's not a new tool, but it's useful enough and underutilized that we felt it's worth calling it out.

© Thoughtworks, Inc. All Rights Reserved.

69. SQLFluff

Assess

While linting is an ancient practice in the software world, it's had slower adoption in the data world. SQLFluff is a cross-dialect SQL linter written in Python that ships with a simple command line interface (CLI), making it easy to incorporate into a CI/CD pipeline. If you're comfortable with the default conventions, then SQLFluff works without any additional configuration after installing it and will enforce a strongly opinionated set of formatting standards; setting your own conventions involves adding a configuration dotfile. The CLI can automatically fix certain classes of violations that involve formatting concerns like whitespace or uppercasing of keywords. SQLFluff is still new, but we're excited to see SQL getting some attention in the linting world.

70. Terraform Validator

Assess

Organizations that have adopted infrastructure as code and self-service infrastructure platforms are looking for ways to give teams a maximum of autonomy while still enforcing good security practices and organizational policies. We've highlighted tfsec before and are moving it into the Adopt category in this Radar. For teams working on GCP, Terraform Validator could be an option when creating a policy library, a set of constraints that are checked against Terraform configurations.

71. Typesense

Assess

Typesense is a fast, typo-tolerant text search engine. For use cases with large volumes of data, Elasticsearch might still be a good option as it provides a horizontally scalable disk-based search solution. However, if you're building a latency-sensitive search application with a search index size that can fit in memory, Typesense is a powerful alternative and another option to evaluate alongside tools such as Meilisearch.

© Thoughtworks, Inc. All Rights Reserved.

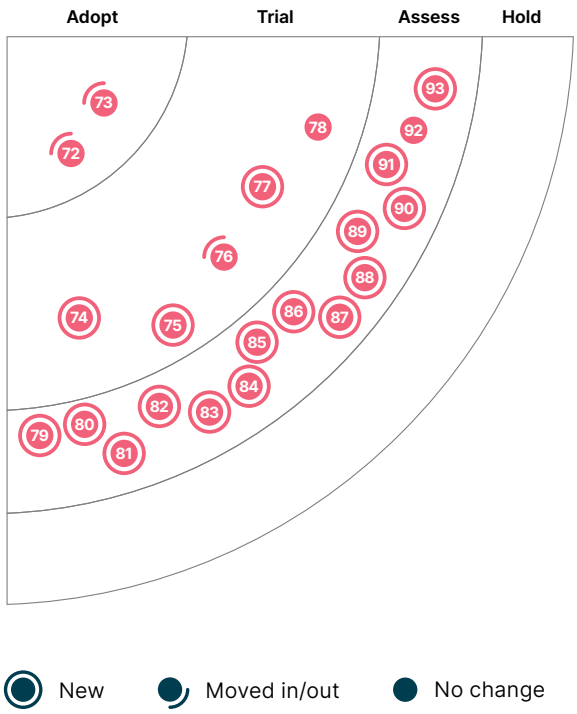
Languages and Frameworks

Adopt
72. SwiftUI
73. Testcontainers

Trial
74. Bob
75. Flutter-Unity widget
76. Kotest
77. Swift Package Manager
78. Vowpal Wabbit

Assess
79. Android Gradle plugin - Kotlin DSL
80. Azure Bicep
81. Capacitor
82. Java 17
83. Jetpack Glance
84. Jetpack Media3
85. MistQL
86. npm workspaces
87. Remix
88. ShedLock
89. SpiceDB
90. sqlc
91. The Composable Architecture
92. WebAssembly
93. Zig

Hold
—



72. SwiftUI

Adopt

When Apple introduced SwiftUI a few years ago, it was a big step forward for implementing user interfaces on all kinds of devices made by Apple. From the beginning, we liked the declarative, code-centric approach and the reactive programming model provided by Combine. We did notice, though, that writing a lot of view tests, which you still need with a model—view—viewmodel (MVVM) pattern, was not really sensible with the XCUITest automation framework provided by Apple. This gap has been closed by ViewInspector. A final hurdle was the minimum OS version required. At the time of release, only the very latest versions of iOS and macOS could run applications written with SwiftUI, but because of Apple's regular cadence of updates, SwiftUI apps can now run on practically all versions of macOS and iOS that receive security updates.

73. Testcontainers

Adopt

We've had enough experience with Testcontainers that we think it's a useful default option for creating a reliable environment for running tests. It's a library, ported to multiple languages, that Dockerizes common test dependencies — including various types of databases, queuing technologies, cloud services and UI testing dependencies like web browsers — with the ability to run custom Dockerfiles when needed. It works well with test frameworks like JUnit, is flexible enough to let users manage the container lifecycle and advanced networking and quickly sets up an integrated test environment. Our teams have consistently found this library of programmable, lightweight and disposable containers to make functional tests more reliable.

74. Bob

Trial

When building an app with React Native you sometimes find yourself having to create your own modules. For example, we've encountered this need when building a UI component library for a React Native app. Creating such a module project isn't straightforward, and our teams report success using Bob to automate this task. Bob provides a CLI to create the scaffolding for different targets. The scaffolding is not limited to core functionality but, optionally, can include example code, linters, build pipeline configuration and other features.

75. Flutter-Unity widget

Trial

Flutter is increasingly popular for building cross-platform mobile apps, and Unity is great for building AR/VR experiences. A key piece in the puzzle for integrating Unity and Flutter is the Flutter-Unity widget, which allows embedding Unity apps inside Flutter widgets. One of the key capabilities the widget offers is bi-directional communication between Flutter and Unity. We've found its performance to be pretty good as well, and we're looking forward to leveraging Unity in more Flutter apps.

76. Kotest

Trial

Kotest (previously KotlinTest) is a stand-alone testing tool for the Kotlin ecosystem that is continuing to gain traction within our teams across various Kotlin implementations — native, JVM or JavaScript. Key advantages are that it offers a variety of testing styles in order to structure the

© Thoughtworks, Inc. All Rights Reserved.

test suites and that it comes with a comprehensive set of matchers, which allow for expressive tests in an elegant internal DSL. In addition to its support for property-based testing — a technique we've highlighted previously in the Radar — our teams like the solid IntelliJ plugin and the growing community of support.

77. Swift Package Manager

Trial

Some programming languages, especially newer ones, have a package and dependency management solution built in. When it was introduced in 2014, Swift didn't come with a package manager, and so the macOS and iOS developer community simply kept using CocoaPods and Carthage, the third-party solutions that had been created for Objective-C. A couple of years later Swift Package Manager (SwiftPM) was started as an official Apple open-source project, and it then took another few years before Apple added support for it to Xcode. Even at that point, though, many development teams continued to use CocoaPods and Carthage, mostly because many packages were simply not available via SwiftPM. Now that most packages can be included via SwiftPM and processes have been further streamlined for both creators and consumers of packages, our teams are increasingly relying on SwiftPM.

78. Vowpal Wabbit

Trial

Vowpal Wabbit is a general-purpose machine-learning library. Originally created at Yahoo! Research over a decade ago, Vowpal Wabbit continues to implement new algorithms in reinforcement learning. We want to highlight Vowpal Wabbit 9.0, a major release after six years, and encourage you to plan the migration as it has several usability improvements, new reductions and bug fixes.

79. Android Gradle plugin - Kotlin DSL

Assess

Android Gradle plugin - Kotlin DSL added support for Kotlin Script as an alternative to Groovy for Gradle build scripts. The goal of replacing Groovy with Kotlin is to provide better support for refactoring and simpler editing in IDEs as well as ultimately to produce code that is easier to read and maintain. For teams already using Kotlin it also means working on the build in a familiar language. We had a team with an at least seven-year-old 450-line build script migrate within a few days. If you have large or complex gradle build scripts, then it's worth assessing whether Kotlin Script will produce better outcomes for your teams.

80. Azure Bicep

Assess

For those who prefer a more natural language than JSON for infrastructure code, Azure Bicep is a domain-specific language (DSL) that uses a declarative syntax. It supports reusable parameterized templates for modular resource definitions. A Visual Studio Code extension provides instant type-safety, intellisense and syntax checking, and the compiler allows bidirectional transpilation to and from ARM templates. Bicep's resource-oriented DSL and native integration with the Azure ecosystem make it a compelling choice for Azure infrastructure development.

81. Capacitor

Assess

We've been debating the merits of cross-platform mobile development tools for nearly as long as we've been publishing the Technology Radar. We first noted a new generation of tools in 2011 when

© Thoughtworks, Inc. All Rights Reserved.

blipping about cross-mobile platforms. Although we were skeptical of them at first, these tools have been perfected and widely adopted over the years. And nobody can debate the enduring popularity and usefulness of React Native. Capacitor is the latest generation of a line of tools starting with PhoneGap, then renamed to Apache Cordova. Capacitor is a complete rewrite from Ionic that embraces the progressive web app style for stand-alone applications. So far, our developers like that they can address web, iOS and Android applications with a single code base and that they can manage the native platforms separately with access to the native APIs when necessary. Capacitor offers an alternative to React Native, which has many years of cross-platform experience behind it.

82. Java 17

Assess

We don't routinely feature new versions of languages, but we wanted to highlight the new long-term support (LTS) version of Java, version 17. While there are promising new features, such as the preview of pattern matching, it's the switch to the new LTS process that should interest many organizations. We recommend organizations assess new releases of Java as and when they become available, making sure they adopt new features and versions as appropriate. Surprisingly many organizations do not routinely adopt newer versions of languages even though regular updates help keep things small and manageable. Hopefully the new LTS process, alongside organizations moving to regular updates, will help avoid the "too expensive to update" trap that ends with production software running on an end-of-life version of Java.

83. Jetpack Glance

Assess

Android 12 brought significant changes to app widgets that have improved the user and developer experience. For writing regular Android apps, we've expressed our preference for Jetpack Compose as a modern way of building native user interfaces. Now, with Jetpack Glance, which is built on top of the Compose runtime, developers can use similar declarative Kotlin APIs for writing widgets. Recently, Glance has been extended to support Tiles for Wear OS.

84. Jetpack Media3

Assess

Android today has several media APIs: Jetpack Media, also known as MediaCompat, Jetpack Media2 and ExoPlayer. Unfortunately, these libraries were developed independently, with different goals but overlapping functionality. Android developers not only had to choose which library to use, they also had to contend with writing adaptors or other connecting code when features from multiple APIs were needed. Jetpack Media3 is an effort, currently in early access, to create a new API that takes common areas of functionality from the existing APIs — including UI, playback and media session handling — combining them into a merged and refined API. The player interface from ExoPlayer has also been updated, enhanced and streamlined to act as the common player interface for Media3.

85. MistQL

Assess

MistQL is a small domain-specific language for performing computations on JSON-like structures. Originally built for handcrafted feature extraction of machine-learning models on the frontend, MistQL currently supports a JavaScript implementation for browsers and a Python implementation for server-side use cases. We quite like its clean composable functional syntax, and we encourage you to assess it based on your needs.

© Thoughtworks, Inc. All Rights Reserved.

86. npm workspaces

Assess

While many tools support multipackage development in the node.js world, npm 7 adds direct support with the addition of npm workspaces. Managing related packages together facilitates development, allowing you, for example, to store multiple related libraries in a single repo. With npm workspaces, once you add a configuration in a top-level package.json file to refer to one or more nested package.json files, commands like `npm install` work across multiple packages, symlinking the dependent source packages into the root node_modules directory. Other npm commands are also now workspace aware, allowing you, for example, to execute `npm run` and `npm test` commands across multiple packages with a single command. Having that flexibility out of the box decreases the need for some teams to reach for another package manager.

87. Remix

Assess

We witnessed the migration from server-side rendering website to single-page application in the browser, now the pendulum of web development seems to swing back to the middle. Remix is one such example. It's a full-stack JavaScript framework. It provides fast page loads by leveraging distributed systems and native browsers instead of clumsy static builds. It has made some optimizations on nested routing and page loading, which makes page rendering seem especially fast. Many people will compare Remix with Next.js, which is similarly positioned. We're glad to see such frameworks cleverly combining the browser run time with the server run time to provide a better user experience.

88. ShedLock

Assess

Executing a scheduled task once and only once in a cluster of distributed processors is a relatively common requirement. For example, the situation might arise when ingesting a batch of data, sending a notification or performing some regular cleanup activity. But this is a notoriously difficult problem. How does a group of processes cooperate reliably over laggy and less reliable networks? Some kind of locking mechanism is required to coordinate actions across the cluster. Fortunately, a variety of distributed stores can implement a lock. Systems like ZooKeeper and Consul as well as databases such as DynamoDB or Couchbase have the necessary underlying mechanisms to manage consensus across the cluster. ShedLock is a small library for taking advantage of these providers in your own Java code, if you're looking to implement your own scheduled tasks. It provides an API for acquiring and releasing locks as well as connectors to a wide variety of lock providers. If you're writing your own distributed tasks but don't want to take on the complexity of an entire orchestration platform like Kubernetes, ShedLock is worth a look.

89. SpiceDB

Assess

SpiceDB is a database system, inspired by Google's Zanzibar, for managing application permissions. With SpiceDB, you create a schema to model the permissions requirements and use the client library to apply the schema to one of the supported databases, insert data and query to efficiently answer questions like "Does this user have access to this resource?" or even the inverse "What are all the resources this user has access to?" We usually advocate separating the authorization policies from code, but SpiceDB takes it a step further by separating data from the policy and storing it as a graph to efficiently answer authorization queries. Because of this separation, you have to ensure

© Thoughtworks, Inc. All Rights Reserved.

that the changes in your application's primary data store are reflected in SpiceDB. Among other Zanzibar-inspired implementations, we find SpiceDB to be an interesting framework to assess for your authorization needs.

90. sqlc

Assess

sqlc is a compiler that generates type-safe idiomatic Go code from SQL. Unlike other approaches based on object-relational mapping (ORM), you continue to write plain SQL for your needs. Once invoked, sqlc checks the correctness of the SQL and generates performant Go code, which can be directly called from the rest of the application. With stable support for both PostgreSQL and MySQL, sqlc is worth a look, and we encourage you to assess it.

91. The Composable Architecture

Assess

Developing apps for iOS has become more streamlined over time, and SwiftUI moving into Adopt is a sign of that. Going beyond the general nature of SwiftUI and other common frameworks, The Composable Architecture (TCA) is both a library and an architectural style for building apps. It was designed over the course of a series of videos, and the authors state that they had composition, testing and ergonomics in mind, building on a foundation of ideas from The Elm Architecture and Redux. As expected, the narrow scope and opinionatedness is both a strength and a weakness of TCA. We feel that teams who don't have a lot of expertise in writing iOS apps, which are often teams who may be looking after multiple related codebases with different tech stacks, stand to benefit the most from using an opinionated framework like TCA, and we like the opinions expressed in TCA.

92. WebAssembly

Assess

WebAssembly (WASM) is the W3C standard that provides capabilities of executing code in the browser. Supported by all major browsers and backward compatible, it's a binary compilation format designed to run in the browser at near native speeds. It opens up the range of languages you can use to write front-end functionality, with early focus on C, C++ and Rust, and it's also an LLVM compilation target. When run in the sandbox, it can interact with JavaScript and shares the same permissions and security model. Portability and security are key capabilities that will enable most platforms, including mobile and IoT.

93. Zig

Assess

Zig is a new language that shares many attributes with C but with stronger typing, easier memory allocation, support for namespacing and a host of other features. Its syntax, however, is reminiscent of JavaScript rather than C, which some may hold against it. Zig's aim is to provide a very simple language with straightforward compilation that minimizes side-effects and delivers predictable, easy-to-trace execution. Zig also provides simplified access to LLVM's cross-compilation capability. Some of our developers have found this feature so viable, they're using Zig as a cross-compiler even though they aren't writing Zig code. Zig is a novel language and worth looking into for applications where C is being considered or already in use as well as for low-level systems applications that require explicit memory manipulation.

© Thoughtworks, Inc. All Rights Reserved.

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 is a global technology consultancy that integrates strategy, design and engineering to drive digital innovation. We are 10,000+ people strong across 49 offices in 17 countries. Over the last 25+ years, we've delivered extraordinary impact together with our clients by helping them solve complex business problems with technology as the differentiator.

/thoughtworks