ThoughtWorks®

TECHNOLOGY RADAR

An opinionated guide to technology frontiers

Vol.22

thoughtworks.com/radar

#TWTechRadar

Contributors

The Technology Radar is prepared by the ThoughtWorks Technology Advisory Board

The Technology Advisory Board (TAB) is a group of 20 or so senior technologists at ThoughtWorks. The TAB meets twice a year face-to-face and bi-weekly by phone. Its primary role is to be an advisory group for for ThoughtWorks CTO, Rebecca Parsons.

The TAB acts as a broad body that can look at topics that affect technology and technologists at ThoughtWorks. We usually create the Radar at face-to-face meetings, but given the global pandemic we've been living through, this is the first Technology Radar to be created via a virtual event.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Camilla Crispim

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Jonny LeRoy

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Ni Wang

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

About the Radar

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

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

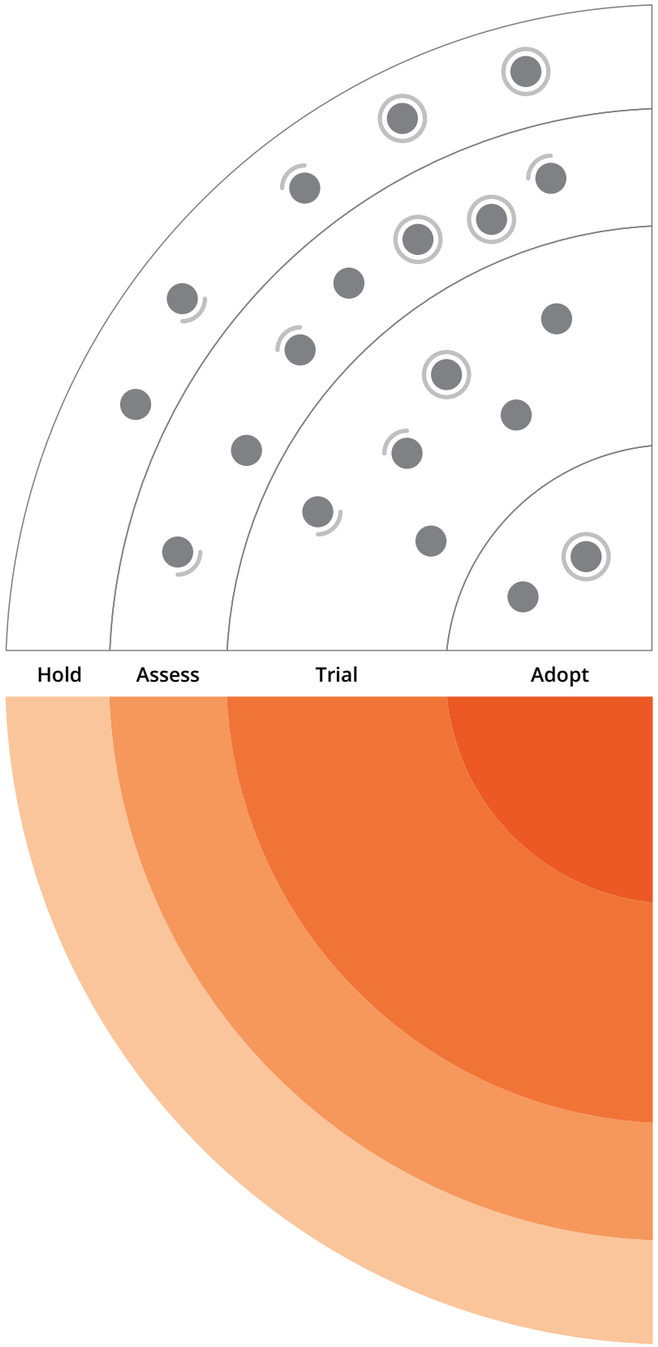
We encourage you to explore these technologies. The Radar is graphical in nature, grouping items into techniques, tools, platforms and languages & frameworks. When Radar items could appear in multiple quadrants, we chose the one that seemed most appropriate. We further group these items in four rings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

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

Radar at a glance

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

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



New

Moved in/out

No change

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

Adopt

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

Trial

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

Assess

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

Hold

Proceed with caution.

Themes for this edition

The Elephant in the Zoom

"Necessity is the mother of invention"
— Proverb

Many companies have experimented with the idea of remote working as the technology to enable it has slowly matured. But suddenly, a global pandemic has forced companies all over the world to rapidly and fundamentally change their way of working to preserve some productivity. As many have observed, "working from home" is starkly different from "being forced to work from home during a pandemic," and we think there will be a journey ahead to become fully productive in this new context.

We've never believed that creating a Radar remotely was possible, and yet here we are — this is the first Radar we've ever produced without meeting in person. Many of the proposed blips spoke to the pressing need to enable first-class remote collaboration. We didn't want to ignore the elephant in the room and not comment on the crisis, but doing a good job of remote-first collaboration is a deep and nuanced subject and certainly not all of our advice would fit in the Radar format. So alongside this edition you'll find a podcast where we discuss our experiences in creating the Radar remote-first, a written experience report including advice on remote-first productivity, a webinar covering tech strategies in a crisis and links to other ThoughtWorks materials, including our remote working playbook. We hope that these materials, together with other internet resources, will help organizations that attempt to navigate these unknown waters.

X is Software Too

We often encourage other parts of the software delivery ecosystem to adopt beneficial engineering practices pioneered by agile software development teams; we return to this topic so often because we keep finding niches where we see slow progress on this advice. For this Radar, we decided to call out again infrastructure as code as well as pipelines as code, and we also had a number of conversations about infrastructure configurations, ML pipelines and other related areas. We find that the teams who commonly own these areas do not embrace enduring engineering practices such as applying software design principles, automation, continuous integration, testing, and so on. We understand that many factors hamper fast movement for some engineering practices: complexity (both essential and accidental), lack of knowledge, political impediments, lack of suitable tooling and many others. However, the benefits to organizations that embrace agile software delivery practices are clear and worth some effort to achieve.

Data Perspectives Maturing and Expanding

A theme that spanned many blips and quadrants in this edition concerned maturity in data, particularly techniques and tools surrounding analytical data and machine learning. We note many continuing innovations in the natural language processing (NLP) space. We also welcome both the emergence and continuing maturity of full-lifecycle machine learning tool suites, combining enduring engineering practices with combinations of tools that work well in an iterative manner, showing that "machine learning is software too." Finally, for distributed architectures such as microservices, we see great interest in data mesh as a way to effectively serve and use analytical data at scale in distributed systems. As the industry thinks more diligently about how data should work in modern systems, we're encouraged by the general direction and opening perspectives in this arena and expect to see exciting innovations in the near future.

Kubernetes & Co. Cambrian Explosion

As Kubernetes continues to consolidate its market dominance, the inevitable supporting ecosystem thrives. We discussed a number of blips surrounding Kubernetes in the tools, platforms and techniques quadrants, showing just how pervasive this subject has become. For example, Lens and k9s simplify cluster management, kind helps with local testing and Gloo offers an alternative API Gateway. Hydra is an OAuth server optimized to run on Kubernetes, and Argo CD uses Kuberenetes' native desired-state management to implement a CD server. These developments indicate Kubernetes is perfectly poised to create a supporting ecosystem; it offers critical capabilities but with abstractions that are often too low level or advanced for most users. Thus, the complexity void fills with tooling to either ease the configuration and the use of Kubernetes or supply something missing from the core functionality. As Kubernetes continues to dominate, we see a rich ecosystem growing and expanding to take advantage of its strengths and address its weaknesses. As this ecosystem matures, we expect it to evolve toward a new set of higher-level abstractions offering the benefits of Kubernetes without the bewildering range of options.

The Radar

Techniques

Adopt
1. Applying product management to internal platforms
2. Infrastructure as code
3. Micro frontends
4. Pipelines as code
5. Pragmatic remote pairing
6. Simplest possible feature toggle

Trial
7. Continuous delivery for machine learning (CD4ML)
8. Ethical bias testing
9. GraphQL for server-side resource aggregation
10. Micro frontends for mobile
11. Platform engineering product teams
12. Security policy as code
13. Semi-supervised learning loops
14. Transfer learning for NLP
15. Use “remote native” processes and approaches
16. Zero trust architecture (ZTA)

Assess
17. Data mesh
18. Decentralized identity
19. Declarative data pipeline definition
20. DeepWalk
21. Managing stateful systems via container orchestration
22. Preflight builds

Hold
23. Cloud lift and shift
24. Legacy migration feature parity
25. Log aggregation for business analytics
26. Long-lived branches with Gitflow
27. Snapshot testing only

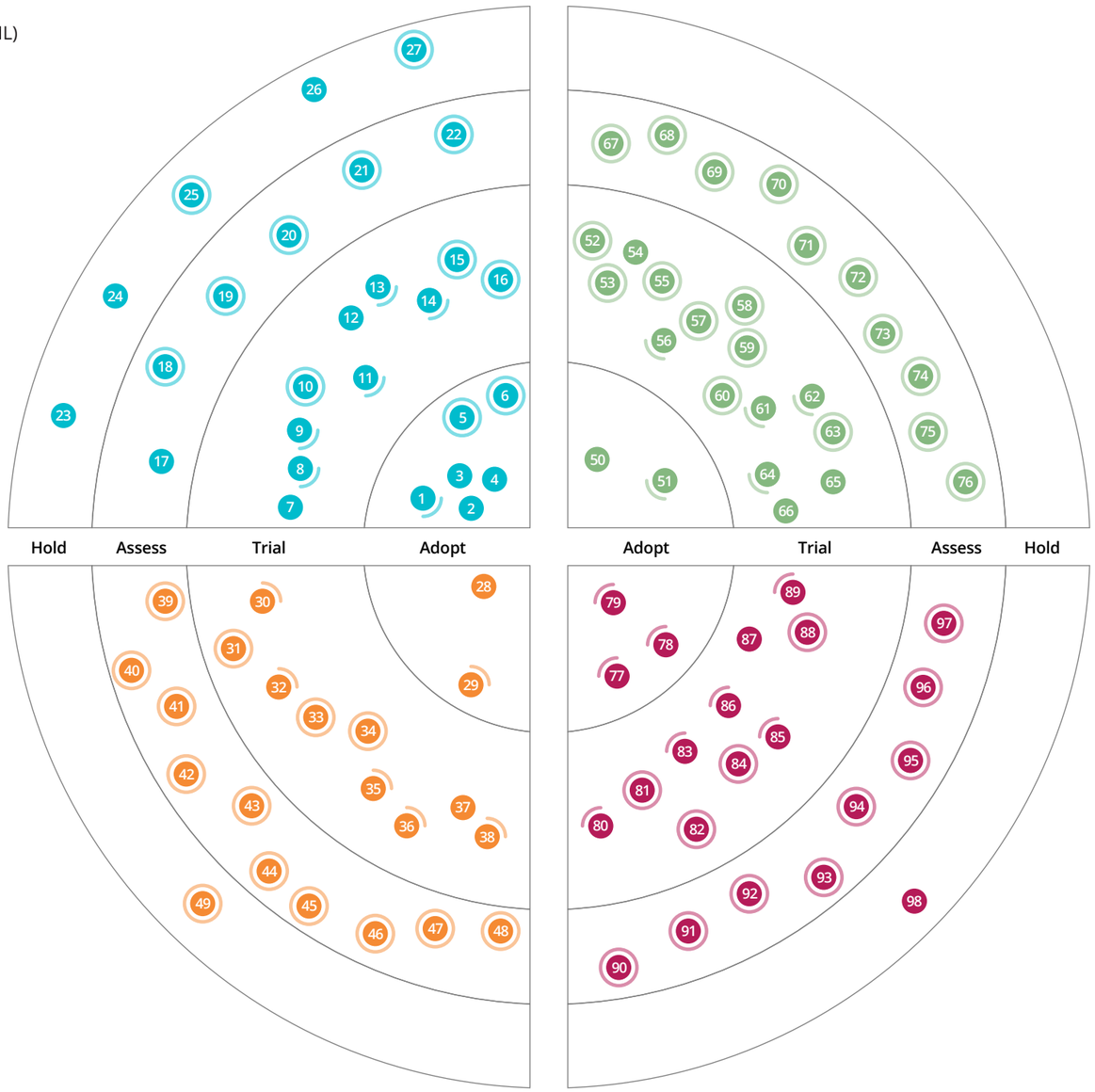
Platforms

Adopt
28. .NET Core
29. Istio

Trial
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

Assess
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

Hold
49. Node overload



Tools

Adopt
50. Cypress
51. Figma

Trial
52. Dojo
53. DVC
54. Experiment tracking tools for machine learning
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. Visual regression testing tools
66. Visual Studio Live Share

Assess
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

Hold

Languages & Frameworks

Adopt
77. React Hooks
78. React Testing Library
79. Vue.js

Trial
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

Assess
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

Hold
98. Enzyme

New Moved in/out No change

TECHNOLOGY RADAR Vol. 22

Techniques

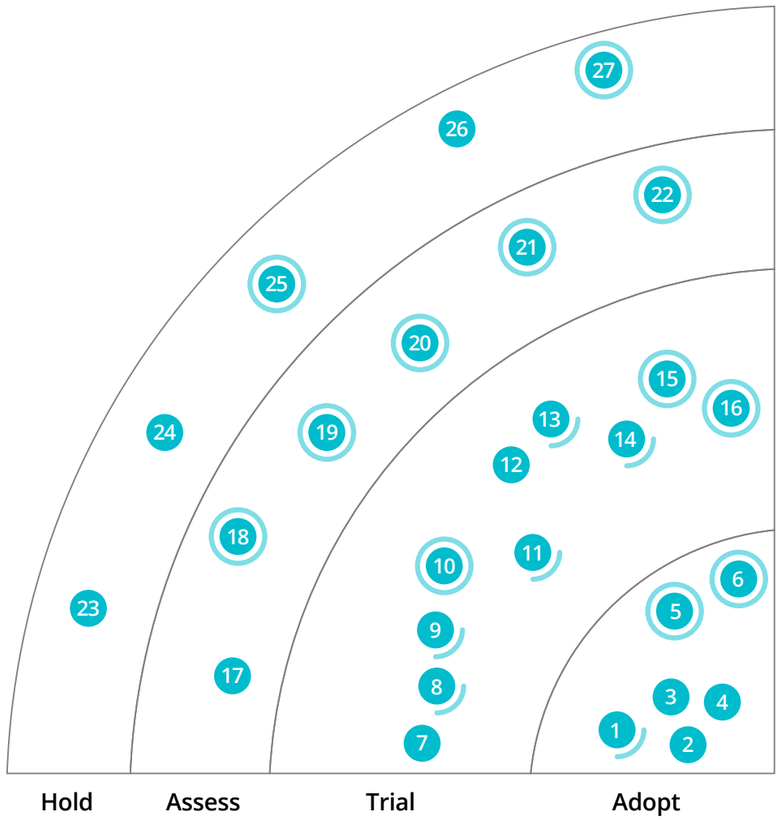
Techniques

Applying product management to internal platforms
Adopt

More and more companies are building internal platforms to roll out new digital solutions quickly and efficiently. Companies that succeed with this strategy are applying product management to internal platforms. This means establishing empathy with internal consumers (the development teams) and collaborating with them on the design. Platform product managers create roadmaps and ensure the platform delivers value to the business and enhances the developer experience. Unfortunately, we're also seeing less successful approaches, where teams create a platform in the void, based on unverified assumptions and without internal customers. These platforms, often despite aggressive internal tactics, end up being underutilized and a drain on the organization's delivery capability. As usual, good product management is all about building products that consumers love.

Infrastructure as code
Adopt

Although infrastructure as code is a relatively old technique (we've featured it in the Radar in 2011), it has become vitally important in the modern cloud era where the act of setting up infrastructure has become the passing of configuration instructions to a cloud platform. When we say "as code" we mean that all the good practices we've learned in the software world should be applied to infrastructure. Using source control, adhering to the DRY principle, modularization, maintainability, and using automated testing and deployment are all critical practices. Those of us with a deep software and infrastructure background need to empathize with and support colleagues who do not. Saying "treat infrastructure like code" isn't enough; we need to ensure the hard-won learnings from the software world are also applied consistently throughout the infrastructure realm.



Micro frontends
Adopt

We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end services — largely neutralizing the benefits of microservices. Micro frontends have continued to gain in popularity since they were first introduced. We've seen many teams adopt some form of this architecture as a way to manage the complexity of multiple developers and teams contributing to the same user experience. In June of last year, one of the originators of this technique published an introductory article that serves as a reference for micro frontends. It shows how this style

Adopt
1. Applying product management to internal platforms
2. Infrastructure as code
3. Micro frontends
4. Pipelines as code
5. Pragmatic remote pairing
6. Simplest possible feature toggle

Trial
7. Continuous delivery for machine learning (CD4ML)
8. Ethical bias testing
9. GraphQL for server-side resource aggregation
10. Micro frontends for mobile
11. Platform engineering product teams
12. Security policy as code
13. Semi-supervised learning loops
14. Transfer learning for NLP
15. Use "remote native" processes and approaches
16. Zero trust architecture (ZTA)

Assess
17. Data mesh
18. Decentralized identity
19. Declarative data pipeline definition
20. DeepWalk
21. Managing stateful systems via container orchestration
22. Preflight builds

Hold
23. Cloud lift and shift
24. Legacy migration feature parity
25. Log aggregation for business analytics
26. Long-lived branches with Gitflow
27. Snapshot testing only

Techniques

We firmly believe in pair programming. But in a post COVID-19, remote-working world, successful pairing requires a healthy dose of pragmatism to be effective.

(Pragmatic remote pairing)

We encourage you to use the simplest possible feature toggle instead of unnecessarily complex feature toggle frameworks.

(Simplest possible feature toggle)

can be implemented using various web programming mechanisms and builds out an example application using React.js. We're confident this style will grow in popularity as larger organizations try to decompose UI development across multiple teams.

Pipelines as code

Adopt

The pipelines as code technique emphasizes that the configuration of delivery pipelines that build, test and deploy our applications or infrastructure should be treated as code; they should be placed under source control and modularized in reusable components with automated testing and deployment. As organizations move to decentralized autonomous teams building microservices or micro frontends, the need for engineering practices in managing pipelines as code increases to keep building and deploying software consistent within the organization. This need has given rise to delivery pipeline templates and tooling that enable a standardized way to build and deploy services and applications. Such tools use the declarative delivery pipelines of applications, adopting a pipeline blueprint to execute the underlying tasks for various stages of a delivery lifecycle such as build, test and deployment; and they abstract away implementation details. The ability to build, test and deploy pipelines as code should be one of the evaluation criteria for choosing a CI/CD tool.

Pragmatic remote pairing

Adopt

We firmly believe that pair programming improves the quality of code, spreads knowledge throughout a team and allows overall faster delivery of software. In a post COVID-19 world, however, many software teams will be distributed or fully remote, and in this situation we recommend pragmatic remote pairing: adjusting pairing practices to what's possible given the tools at hand. Consider tools such as Visual Studio Live Share for efficient, low-latency collaboration. Only resort to pixel-sharing if both participants reside in relative geographic proximity and have high-bandwidth internet connections. Pair developers who are in similar time zones rather than expecting pairing to work between participants regardless of their location. If pairing isn't working for logistical reasons, fall back to practices such as individual programming augmented via code reviews, pull-request collaboration (but beware long-lived branches with Gitflow) or shorter pairing sessions for critical parts of the code. We've engaged in remote pairing for years, and we've found it to be effective if done with a dose of pragmatism.

Simplest possible feature toggle

Adopt

Unfortunately, feature toggles are less common than we'd like, and quite often we see people mixing up its types and use cases. It's quite common to come across teams that use heavyweight platforms such as LaunchDarkly to implement feature toggles, including release toggles, to benefit from Continuous Integration, when all you need are if/else conditionals. Therefore, unless you need A/B testing or canary release or hand over feature release responsibility to business folks, we encourage you to use the simplest possible feature toggle instead of unnecessarily complex feature toggle frameworks.

Continuous delivery for machine learning (CD4ML)

Trial

Applying machine learning to make the business applications and services intelligent is more than just training models and serving them. It requires implementing end-to-end and continuously repeatable cycles of training, testing, deploying, monitoring and operating the models. Continuous delivery for machine learning (CD4ML) is a technique that enables reliable end-to-end cycles of development, deploying and monitoring machine learning models. The underpinning technology stack to enable CD4ML includes tooling for accessing and discovering data, version control of artefacts (such as data, model and code), continuous delivery pipelines, automated environment provisioning for various deployments and experiments, model performance assessment and tracking, and model operational observability. Companies can choose their own tool set depending on their existing tech stack. CD4ML emphasizes automation and removing manual handoffs. CD4ML is our de facto approach for developing ML models.

Ethical bias testing

Trial

Over the past year, we've seen a shift in interest around machine learning and deep neural networks in particular. Until now, tool and technique development has been driven by excitement over the remarkable capabilities of these models. Currently, though, there is rising concern that these models could cause unintentional harm. For example, a model could be trained inadvertently to make profitable credit decisions by simply excluding disadvantaged applicants. Fortunately, we're seeing a growing interest in ethical bias testing that will help to uncover potentially harmful decisions. Tools such as lime, AI Fairness 360 or What-If Tool can help uncover inaccuracies that result from underrepresented groups in training data and visualization tools such as Google Facets or Facets Dive can be used to discover subgroups within a corpus

of training data. We've used lime (local interpretable model-agnostic explanations) in addition to this technique in order to understand the predictions of any machine-learning classifier and what classifiers (or models) are doing.

GraphQL for server-side resource aggregation

Trial

We see more and more tools such as Apollo Federation that can aggregate multiple GraphQL endpoints into a single graph. However, we caution against misusing GraphQL, especially when turning it into a server-to-server protocol. Our practice is to use GraphQL for server-side resource aggregation only. When using this pattern, the microservices continue to expose well-defined RESTful APIs, while under-the-hood aggregate services or BFF (Backend for Frontends) patterns use GraphQL resolvers as the implementation for stitching resources from other services. The shape of the graph is driven by domain-modeling exercises to ensure ubiquitous language is limited to subgraphs where needed (in the case of one-microservice-per-bounded-context). This technique simplifies the internal implementation of aggregate services or BFFs, while encouraging good modeling of services to avoid anemic REST.

Micro frontends for mobile

Trial

Since introducing it in the Radar in 2016, we've seen widespread adoption of micro frontends for web UIs. Recently, however, we've seen projects extend this architectural style to include micro frontends for mobile applications as well. When the application becomes sufficiently large and complex, it becomes necessary to distribute the development over multiple teams. This presents the challenge of maintaining team autonomy while integrating their work into a single app. Although we've seen teams writing their own frameworks to enable this development style, existing modularization frameworks such as Atlas and Beehive can also simplify the problem of integrating multiteam app development.

Platform engineering product teams

Trial

The adoption of cloud and DevOps — while increasing the productivity of teams who can now move more quickly with reduced dependency on centralized operations teams and infrastructure — also has constrained teams that lack the skills to self-manage a full application and operations stack. Some organizations have tackled this challenge by creating platform engineering product teams. These teams maintain an internal platform that enables delivery teams to deploy and operate systems with reduced lead time and stack complexity. The emphasis here is on API-driven self-service and supporting tools, with delivery teams still responsible for supporting what they deploy onto the platform. Organizations that consider establishing such a platform team should be very cautious not to accidentally create a separate DevOps team, nor should they simply relabel their existing hosting and operations structure as a platform. If you're wondering how to best set up platform teams, we've been using the concepts from Team Topologies to split platform teams in our projects into enablement teams, core "platform within a platform" teams and stream-focused teams.

Security policy as code

Trial

Security policies are rules and procedures that protect our systems from threats and disruption. For example, access control policies define and enforce who can access which services and resources under what circumstances; or network security policies can dynamically limit the traffic rate to a particular service. The complexity of the technology landscape today demands treating security policy as code: define and keep policies under version control, automatically validate them, automatically deploy them and monitor their performance. Tools such as Open Policy Agent or platforms such as Istio provide flexible policy definition and enforcement mechanisms that support the practice of security policy as code.

Semi-supervised learning loops

Trial

Semi-supervised learning loops are a class of iterative machine-learning workflows that take advantage of the relationships to be found in unlabeled data. These techniques may improve models by combining labeled and unlabeled data sets in various ways. In other cases they compare models trained on different subsets of the data. Unlike either unsupervised learning where a machine infers classes in unlabeled data or supervised techniques where the training set is entirely labeled, semi-supervised techniques take advantage of a small set of labeled data and a much larger set of unlabeled data. Semi-supervised learning is also closely related to active learning techniques where a human is directed to selectively label ambiguous data points. Since expert humans that can accurately label data are a scarce resource and labeling is often the most time-consuming activity in the machine-learning workflow, semi-supervised techniques lower the cost of training and make machine learning feasible for a new class of users. We're also seeing the application of weakly supervised techniques where machine-labeled data

Techniques

There is rising concern that some machine-learning models could cause unintentional harm. Fortunately, we're seeing a growing interest in ethical bias testing that will help to uncover potentially harmful decisions.

(Ethical bias testing)

Micro frontends have been widely adopted for Web UIs. Now we're seeing this architectural style come to mobile too.

(Micro frontends for mobile)

Techniques

Zero trust architectures stress the importance of securing all access and communications and enforcing policies as code based on the least privilege.

(Zero trust architecture (ZTA))

is used but is trusted less than the data labeled by humans.

Transfer learning for NLP
Trial

We had this technique in Assess previously. The innovations in the NLP landscape continue at a great pace, and we're able to leverage these innovations in our projects thanks to the ubiquitous transfer learning for NLP. The GLUE benchmark (a suite of language understanding tasks) scores have seen dramatic progress over the past couple of years with average scores moving from 70.0 at launch to some of the leaders crossing 90.0 as of April 2020. A lot of our projects in the NLP domain are able to make significant progress by starting from pretrained models from ELMo, BERT, and ERNIE, among others, and then fine-tuning them based on the project needs.

Use "remote native" processes and approaches
Trial

Distributed teams come in many shapes and setups; delivery teams in a 100% single-site co-located setup, however, have become the exception for us. Most of our teams are either multisite teams or have at least some team members working off-site. Therefore, using "remote native" processes and approaches by default can help significantly with the overall team flow and effectiveness. This starts with making sure that everybody has access to the necessary remote systems. Moreover, using tools such as Visual Studio Live Share, MURAL or Jamboard turn online workshops and remote pairing into routines instead of ineffective exceptions. But "remote native" goes beyond a lift-and-shift of co-location practices to the digital world: Embracing more asynchronous communication, even more discipline around decision documentation, and "everybody always remote" meetings are other approaches our teams practice by default to optimize for location fluidity.

Zero trust architecture (ZTA)
Trial

The technology landscape of organizations today is increasingly more complex with assets — data, functions, infrastructure and users — spread across security boundaries, such as local hosts, multiple cloud providers and a variety of SaaS vendors. This demands a paradigm shift in enterprise security planning and systems architecture, moving from static and slow-changing security policy management, based on trust zones and network configurations, to dynamic, fine-grained security policy enforcement based on temporal access privileges.

Zero trust architecture (ZTA) is an organization's strategy and journey to implement zero-trust security principles for all of their assets — such as devices, infrastructure, services, data and users — and includes implementing practices such as securing all access and communications regardless of the network location, enforcing policies as code based on the least privilege and as granular as possible, and continuous monitoring and automated mitigation of threats. Our Radar reflects many of the enabling techniques such as security policy as code, sidecars for endpoint security and BeyondCorp. If you're on your journey toward ZTA, refer to the NIST ZTA publication to learn more about principles, enabling technology components and migration patterns as well as Google's publication on BeyondProd.

Data mesh
Assess

Data mesh is an architectural and organizational paradigm that challenges the age-old assumption that we must centralize big analytical data to use it, have data all in one place or be managed by a centralized data team to deliver value. Data mesh claims that for big data to fuel innovation, its ownership must be federated among domain data owners who are accountable for providing their data as products (with the support of a self-serve data platform to abstract the technical complexity involved in serving data products); it must also adopt a new form of federated governance through automation to enable interoperability of domain-oriented data products. Decentralization, along with interoperability and focus on the experience of data consumers, are key to the democratization of innovation using data.

If your organization has a large number of domains with numerous systems and teams generating data or a diverse set of data-driven use cases and access patterns, we suggest you assess data mesh. Implementation of data mesh requires investment in building a self-serve data platform and embracing an organizational change for domains to take on the long-term ownership of their data products, as well as an incentive structure that rewards domains serving and utilizing data as a product.

Decentralized identity
Assess

Since the birth of the internet, the technology landscape has experienced an accelerated evolution toward decentralization. While protocols such as HTTP and architectural patterns such as microservices or data mesh enable decentralized implementations, identity management remains centralized. The emergence of distributed ledger technology (DLT), however, provides the opportunity to enable the concept of decentralized identity. In a decentralized identity system, entities — that is, discrete identifiable units such as people,

organizations and things — are free to use any shared root of trust. In contrast, conventional identity management systems are based on centralized authorities and registries such as corporate directory services, certificate authorities or domain name registries.

The development of decentralized identifiers — globally unique, persistent and self-sovereign identifiers that are cryptographically verifiable — is a major enabling standard. Although scaled implementations of decentralized identifiers in the wild are still rare, we're excited by the premise of this movement and have started using the concept in our architecture. For the latest experiments and industry collaborations, check out Decentralized Identity Foundation.

Declarative data pipeline definition

Assess

Many data pipelines are defined in a large, more or less imperative script written in Python or Scala. The script contains the logic of the individual steps as well as the code chaining the steps together. When faced with a similar situation in Selenium tests, developers discovered the Page Object pattern, and later many behavior-driven development (BDD) frameworks implemented a split between step definitions and their composition. Some teams are now experimenting with bringing the same thinking to data engineering. A separate declarative data pipeline definition, maybe written in YAML, contains only the declaration and sequence of steps. It states input and output data sets but refers to scripts if and when more complex logic is needed. With A La Mode, we're seeing the first open source tool appear in this space.

DeepWalk

Assess

DeepWalk is an algorithm that helps apply machine learning on graphs. When working on data sets that are represented as graphs, one of the key problems is to extract features from the graph. This is where DeepWalk can help. It uses SkipGram to construct node embeddings by viewing the graph as a language where each node is a unique word in the language and random walks of finite length on the graph constitutes a sentence. These embeddings can then be used by various ML models. DeepWalk is one of the techniques we're trialling on some of our projects where we've needed to apply machine learning on graphs.

Managing stateful systems via container orchestration

Assess

We recommend caution in managing stateful systems via container orchestration platforms such as Kubernetes. Some databases are not built with native support for orchestration — they don't expect a scheduler to kill and relocate them to a different host. Building a highly available service on top of such databases is not trivial, and we still recommend running them on bare metal hosts or a virtual machine (VM) rather than to force-fit them into a container orchestration platform.

Preflight builds

Assess

Even though we strongly advocate in favor of CI rather than Gitflow, we know that committing straight to the trunk and running the CI on a master branch can be ineffective if the team is too big, the builds are slow or flaky, or the team lacks the discipline to run the full test suite locally. In this situation a red build can block multiple devs or pairs of devs. Instead of fixing the underlying root cause — slow builds, the inability to run tests locally or monolithic architectures that necessitate many people working in the same area — teams usually rely on feature branches to bypass these issues. We discourage feature branches, given they may require significant effort to resolve merge conflicts, and they introduce longer feedback loops and potential bugs during conflict resolution. Instead, we propose using preflight builds as an alternative: these are pull request–based builds for "micro branches" that live only for the duration of the pipeline run, with the branch opened for every commit. To help automate this workflow, we've come across bots such as Bors, which automates merging to master and branch deletion in case the mini branch build succeeds. We're assessing this flow, and you should too; but don't use this to solve the wrong problem, as it can lead to misuse of branches and may cause more harm than benefit.

Cloud lift and shift

Hold

It is rather curious, that after over a decade of industry experience with cloud migration, we still feel it's necessary to call out cloud lift and shift, a practice that views the cloud simply as a hosting solution, resulting in the replication of an existing architecture, security practices and IT operational models in the cloud. This fails to realize the cloud's promises of agility and digital innovation. A cloud migration requires intentional change across multiple axes toward a cloud-native state, and depending on the unique migration

Techniques

The foundation of BitCoin — distributed ledger technology (DLT) — is enabling the emergence of decentralized identities.

(Decentralized identity)

When working on data sets that are represented as graphs, one of the key problems is to extract features from the graph. This is where DeepWalk can help.

(DeepWalk)

Techniques

Logs intended for technical observability are often inadequate to infer deep customer understanding.

(Log aggregation for business analytics)

Snapshot testing is undeniably useful when working with legacy systems. But it shouldn't be the primary test mechanism for such systems.

(Snapshot testing only)

circumstances, each organization might end up somewhere on the spectrum from cloud lift and shift to cloud native. Systems architecture, for example, is one of the pillars of delivery agility and often requires change. The temptation to simply lift and shift existing systems as containers to the cloud can be strong. While this tactic can speed up cloud migration, it falls short when it comes to creating agility and delivering features and value. Enterprise security in the cloud is fundamentally different from traditional perimeter-based security through firewalls and zoning, and it demands a journey toward zero trust architecture. The IT operating model too has to be reformed to safely provide cloud services through self-serve automated platforms and empower teams to take more of the operational responsibility and gain autonomy. Last but not least, organizations must build a foundation to enable continuous change, such as creating pipelines with continuous testing of applications and infrastructure as a part of the migration. These will help the migration process, result in a more robust and well-factored system and give organizations a way to continue to evolve and improve their systems.

Legacy migration feature parity

Hold

We find that more and more organizations need to replace aging legacy systems to keep up with the demands of their customers (both internal and external). One antipattern we keep seeing is legacy migration feature parity, the desire to retain feature parity with the old. We see this as a huge missed opportunity. Often the old systems have bloated over time, with many features unused by users (50% according to a 2014 Standish Group report) and business processes that have evolved over time. Replacing these features is a waste. Our advice: Convince your customers to take a step back and understand what their users currently need and prioritize these needs against business outcomes and metrics — which often is easier said than done. This means conducting user research and applying modern product development practices rather than simply replacing the existing ones.

Log aggregation for business analytics

Hold

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

Long-lived branches with Gitflow

Hold

Five years ago we highlighted the problems with long-lived branches with Gitflow. Essentially, long-lived branches are the opposite of continuously integrating all changes to the source code, and in our experience continuous integration is the better approach for most kinds of software development. Later we extended our caution to Gitflow itself, because we saw teams using it almost exclusively with long-lived branches. Today, we still see teams in settings where continuous delivery of web-based systems is the stated goal being drawn to long-lived branches. So we were delighted that the author of Gitflow has now added a note to his original article, explaining that Gitflow was not intended for such use cases.

Snapshot testing only

Hold

The value of snapshot testing is undeniable when working with legacy systems by ensuring that the system continues to work and the legacy code doesn't break. However, we're seeing the common, rather harmful practice of using snapshot testing only as the primary test mechanism. Snapshot tests validate the exact result generated in the DOM by a component, not the component's behavior; therefore, it can be weak and unreliable, fostering the "only delete the snapshot and regenerate it" bad practice. Instead, you should test the logic and behavior of the components emulating what users would do. This mindset is encouraged by tools in the Testing Library family.

TECHNOLOGY RADAR Vol. 22

Platforms

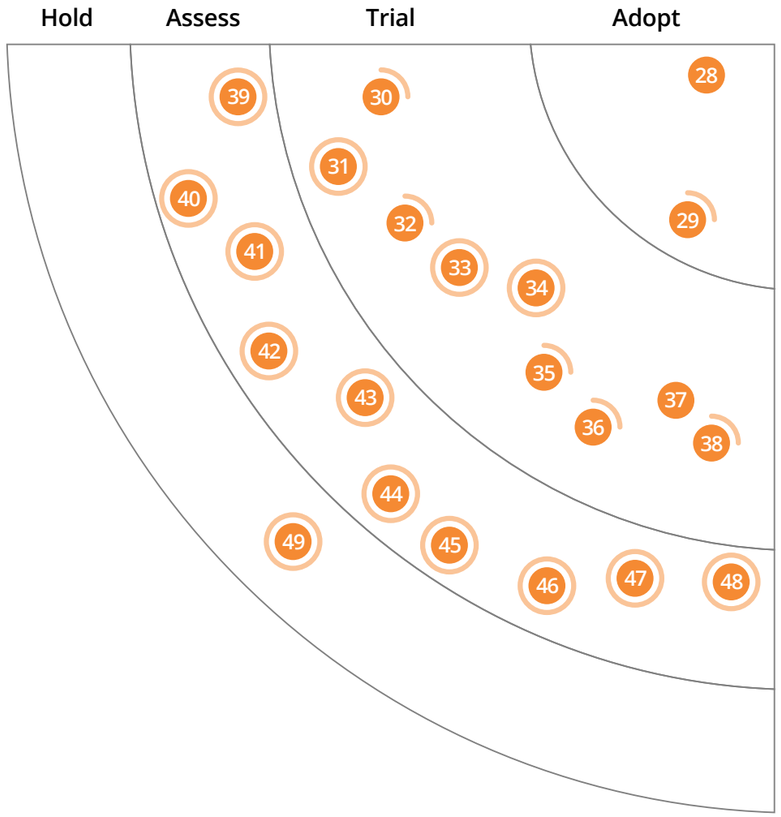
Platforms

.NET Core
Adopt

We previously had .NET Core in Adopt, indicating that it had become our default for .NET projects. But we felt it's worth again calling attention to .NET Core. With the release of .NET Core 3.*x* last year, the bulk of the features from .NET Framework have now been ported into .NET Core. With the announcement that .NET Framework is on its last release, Microsoft have reinforced the view that .NET Core is the future of .NET. Microsoft has done a lot of work to make .NET Core container friendly. Most of our .NET Core–based projects target Linux and are often deployed as containers. The upcoming .NET 5 release looks promising, and we're looking forward to it.

Istio
Adopt

If you're building and operating a scaled microservices architecture and have embraced Kubernetes, adopting service mesh to manage all cross-cutting aspects of running the architecture is a default position. Among various implementations of service mesh, Istio has gained majority adoption. It has a rich feature set, including service discovery, traffic management, service-to-service and origin-to-service security, observability (including telemetry and distributed tracing), rolling releases and resiliency. Its user experience has been improved in its latest releases, because of its ease of installation and control panel architecture. Istio has lowered the bar for implementing large-scale microservices with operational quality for many of our clients, while admitting that operating your own Istio and Kubernetes instances requires adequate knowledge and internal resources which is not for the fainthearted.



Anka
Trial

Anka is a set of tools to create, manage, distribute, build and test macOS reproducible virtual environments for iOS and macOS. It brings Docker-like experience to macOS environments: instant start, CLI to manage virtual machines and registry to version and tag virtual machines for distribution. We've used Anka to build a

Adopt
28. .NET Core
29. Istio

Trial
30. Anka
31. Argo CD
32. Crowdin
33. eBPF
34. Firebase
35. Hot Chocolate
36. Hydra
37. OpenTelemetry
38. Snowflake

Assess
39. Anthos
40. Apache Pulsar
41. Cosmos
42. Google BigQuery ML
43. JupyterLab
44. Marquez
45. Matomo
46. MeiliSearch
47. Stratos
48. Trillian

Hold
49. Node overload

Platforms

Although this technology is not new, it is now coming into its own with the increasing use of microservices deployed as orchestrated containers.

(eBPF)

The clear separation of identity from the rest of the OAuth2 framework makes it easier to integrate Hydra with an existing authentication ecosystem.

(Hydra)

macOS private cloud for a client. This tool is worth considering when virtualizing iOS and macOS environments.

Argo CD
Trial

Without making a judgment of the GitOps technique, we'd like to talk about Argo CD within the scope of deploying and monitoring applications in Kubernetes environments. Based on its ability to automate the deployment of the desired application state in the specified target environments in Kubernetes and our good experience with troubleshooting failed deployments, verifying logs and monitoring deployment status, we recommend you give Argo CD a try. You can even see graphically what is going on in the cluster, how a change is propagated and how pods are created and destroyed in real time.

Crowdin
Trial

Most of the projects with multilingual support start with development teams building features in one language and managing the rest through offline translation via emails and spreadsheets. Although this simple setup works, things can quickly get out of hand. You may have to keep answering the same questions for different language translators, sucking the energy out of the collaboration between translators, proofreaders and the development team. Crowdin is one of a handful of platforms that help in streamlining the localization workflow of your project. With Crowdin the development team can continue building features, while the platform streamlines the text that needs translation into an online workflow. We like that Crowdin nudges the teams to continuously and incrementally incorporate translations rather than managing them in large batches toward the end.

eBPF
Trial

For several years now, the Linux kernel has included the extended Berkeley Packet Filter (eBPF) virtual machine and provided the ability to attach eBPF filters to particular sockets. But *extended* BPF 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. Service-to-service communications can be complex in these systems, making it difficult to correlate latency or performance issues back to an API call. We're now seeing tools released with prewritten eBPF scripts for collecting and visualizing packet traffic or reporting on CPU utilization. With the rise of Kubernetes, we're seeing a new generation of security enforcement and instrumentation based on eBPF scripts that help tame the complexity of a large microservices deployment.

Firebase
Trial

Google's Firebase has undergone significant evolution since we mentioned it as part of a serverless architecture in 2016. Firebase is a comprehensive platform for building mobile and web apps in a way that's supported by Google's underlying scalable infrastructure. We particularly like Firebase App Distribution, which makes it easy to publish test versions of an app via a CD pipeline, and Firebase Remote Config, which allows configuration changes to be dynamically pushed to apps without needing to republish them.

Hot Chocolate
Trial

The GraphQL ecosystem and community keep growing. Hot Chocolate is a GraphQL server for .NET (Core and Classic). It lets you build and host schemas and then serve queries against them using the same base components of GraphQL — data loader, resolver, schema, operations and types. The team behind Hot Chocolate has recently added schema stitching, which allows for a single entry point to query across multiple schemas aggregated from different locations. Despite the potential to misuse this approach, our teams are happy with Hot Chocolate — it's well documented, and we're able to deliver value quickly to our clients.

Hydra
Trial

Not everyone needs a self-hosted OAuth2 solution, but if you do, have a look at Hydra — a fully compliant open source OAuth2 server and OpenID connect provider. Hydra has in-memory storage support for development and a relational database (PostgreSQL) for production use cases. Hydra as such is stateless and easy to scale horizontally in platforms such as Kubernetes. Depending on your

performance requirement, you may have to tune the number of database instances while scaling Hydra instances. And because Hydra doesn't provide any identity management solutions out of the box, you can integrate whatever flavor of identity management you have with Hydra through a clean API. This clear separation of identity from the rest of the OAuth2 framework makes it easier to integrate Hydra with an existing authentication ecosystem.

OpenTelemetry
Trial

OpenTelemetry is an open source observability project that merges OpenTracing and OpenCensus. The OpenTelemetry project includes specification, libraries, agents, and other components needed to capture telemetry from services to better observe, manage and debug them. It covers the three pillars of observability — distributed tracing, metrics and logging (currently in beta) — and its specification connects these three pieces through correlations; thus you can use metrics to pinpoint a problem, locate the corresponding traces to discover where the problem occured, and ultimately study the corresponding logs to find the exact root cause. OpenTelemetry components can be connected to back-end observability systems such as Prometheus and Jaeger among others. Formation of OpenTracing is a positive step toward the convergence of standardization and the simplification of tooling.

Snowflake
Trial

Snowflake has proven to be a robust SaaS big data storage, warehouse or lake solution for many of our clients. It has a superior architecture to scale storage, compute, and services to load, unload and use data. It's also very flexible: it supports storage of structured, semi-structured and unstructured data; provides a growing list of connectors for different access patterns such as Spark for data science and SQL for analytics; and runs on multiple cloud providers. Our advice to many of our clients is to use managed services for their utility technology such as big data storage; however, if the risk and regulations prohibit the use of managed services, then Snowflake is a good candidate for companies with large volumes of data and heavy processing workloads. Although we've been successful using Snowflake in our medium-sized engagements, we've yet to experience Snowflake in large ecosystems where data need to be owned across segments of the organization.

Anthos
Assess

We see a shift from accidental hybrid or whole-of-estate cloud migration plans to intentional and sophisticated hybrid, poly or portable cloud strategies, where organizations apply multidimensional principles to establish and execute their cloud strategy: where to host their various data and functional assets based on risk, ability to control and performance profiles; how to utilize their on-premise infrastructure investments while reducing the cost of operations; and how to take advantage of multiple cloud providers and their unique differentiated services without creating complexity and friction for users building and operating applications.

Anthos is Google's answer to enable hybrid and multicloud strategies by providing a high-level management and control plane on top of a set of open source technologies such as GKE, Service Mesh and a Git-based Configuration Management. It enables running portable workloads and other assets on different hosting environments, including Google Cloud and on-premises hardware. Although other cloud providers have comparative offerings, Anthos intends to go beyond a hybrid cloud to a portable cloud enabler using open source components, but that is yet to be seen. We're seeing a rising interest in Anthos. While Google's approach in managed hybrid cloud environments seems promising, it's not a magic bullet and requires changes in both existing cloud and on-premise assets. Our advice for clients considering Anthos is to make measured tradeoffs between selecting services from the Google Cloud ecosystem and other options, to maintain their right level of neutrality and control.

Apache Pulsar
Assess

Apache Pulsar is an open source pub-sub messaging/streaming platform, competing in a similar space with Apache Kafka. It provides expected functionality — such as low-latency async and sync message delivery and scalable persistent storage of messages — as well as various client libraries. What has excited us to evaluate Pulsar is its ease of scalability, particularly in large organizations with multiple segments of users. Pulsar natively supports multitenancy, georeplication, role-based access control and segregation of billing. We're also looking to Pulsar to solve the problem of a never-ending log of messages for our large-scale data systems where events are expected to persist

Platforms

The OpenTelemetry project includes specification, libraries, agents and other components needed to capture telemetry from services to better observe, manage and debug them.

(OpenTelemetry)

Anthos is Google's answer to enable hybrid and multicloud strategies. It enables you to run portable workloads on different hosting environments including Google Cloud and on-premise hardware.

(Anthos)

Platforms

BigQuery ML lowers the bar for using ML to make predictions and recommendations, particularly for quick explorations.

(Google BigQuery ML)

indefinitely and subscribers are able to start consuming messages retrospectively. This is supported through a tiered storage model. Although Pulsar is a promising platform for large organizations, there is room for improvement. Its current installation requires administering ZooKeeper and BookKeeper among other pieces of technology. We hope that with its growing adoption, users can soon count on wider community support.

Cosmos

Assess

The performance of blockchain technology has been greatly improved since we initially assessed this area in the Radar. However, there's still no single blockchain that could achieve "internet-level" throughput. As various blockchain platforms develop, we're seeing new data and value silos. That's why cross-chain tech has always been a key topic in the blockchain community: the future of blockchain may be a network of independent parallel blockchains. This is also the vision of Cosmos. Cosmos releases Tendermint and CosmosSDK to let developers customize independent blockchains. These parallel blockchains could exchange value through the Inter-Blockchain Communication (IBC) protocol and Peg-Zones. Our teams have had great experiences with CosmosSDK, and the IBC protocol is maturing. This architecture could solve blockchain interoperability and scalability issues.

Google BigQuery ML

Assess

Often training and predicting outcomes from machine learning models require code to take the data to the model. Google BigQuery ML inverts this by bringing the model to the data. Google BigQuery is a data warehouse designed to serve large-scale queries using SQL, for analytical use cases. Google BigQuery ML extends this function and its SQL interface to create, train and evaluate machine learning models using its data sets; and eventually run model predictions to create new BigQuery data sets. It supports a limited set of models out of the box, such as linear regression for forecasting or binary and multiclass regression for classification. It also supports, with limited functionality, importing previously trained TensorFlow models. Although BigQuery ML and its SQL-based approach lower the bar for using machine learning to make predictions and recommendations, particularly for quick explorations, this comes with a difficult trade-off: compromising on other aspects of model training such as ethical bias testing, explainability and continuous delivery for machine learning.

JupyterLab

Assess

JupyterLab is the next-generation web-based user interface for Project Jupyter. If you've been using Jupyter Notebooks, JupyterLab is worth a try; it gives you an interactive environment for Jupyter notebooks, code and data. We see it as an evolution of Jupyter Notebook: it provides a better experience by extending its original capabilities of allowing code, visualization and documentation to exist in one place.

Marquez

Assess

Marquez is a relatively young open source project for collecting and serving metadata information about a data ecosystem. It represents a simple data model to capture metadata such as lineage, upstream and downstream data processing jobs and their status, and a flexible set of tags to capture the attributes of data sets. It provides a simple RESTful API to manage the metadata which eases the integration of Marquez to other tool sets within the data ecosystem.

We've used Marquez as a starting point and easily extended it to fit our needs such as enforcing security policies as well as changes to its domain language. If you're looking for a small and simple tool to bootstrap storage and visualization of your data-processing jobs and data sets, Marquez is a good place to start.

Matomo

Assess

Matomo (formerly Piwik) is an open source web analytics platform that provides you with full control over your data. You can self-host Matomo and secure your web analytics data from third parties. Matomo also makes it easy to integrate web analytics data with your in-house data platform and lets you build usage models that are tailored to your needs.

MeiliSearch

Assess

MeiliSearch is a fast, easy-to-use and easy-to-deploy text search engine. Over the years Elasticsearch has become the popular choice for scalable text searches. However, if you don't have the volume of data that warrants a distributed solution but still want to provide a fast typo-tolerant search engine, then we recommend assessing MeiliSearch.

Stratos
Assess

Ultraleap (previously Leap Motion) has been a leader in the XR space for some time, creating remarkable hand-tracking hardware that allows a user's hands to make the leap into virtual reality. Stratos is Ultraleap's underlying haptics, sensors and software platform, and it can use targeted ultrasound to create haptic feedback in mid-air. A use case is responding to a driver's hand gesture to change the air conditioning in the car and providing haptic feedback as part of the interface. We're excited to see this technology and what creative technologists might do to incorporate it into their use cases.

Trillian
Assess

Trillian is a cryptographically verifiable, centralized data store. For trustless, decentralized environments, you can use blockchain-based distributed ledgers. For enterprise environments, however, where the cost of CPU-heavy consensus protocols is unwarranted, we recommend you give Trillian a try.

Node overload
Hold

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

Platforms

MeiliSearch is a fast, easy-to-use and easy-to-deploy text search engine. It's ideal if you don't have the volume of data that warrants a distributed solution but still want to provide a fast typo-tolerant search engine.

(MeiliSearch)

Stratos is the underlying haptics, sensors and software platform that powers XR pioneer Ultraleap (previously Leap Motion).

(Stratos)

TECHNOLOGY RADAR Vol. 22

Tools

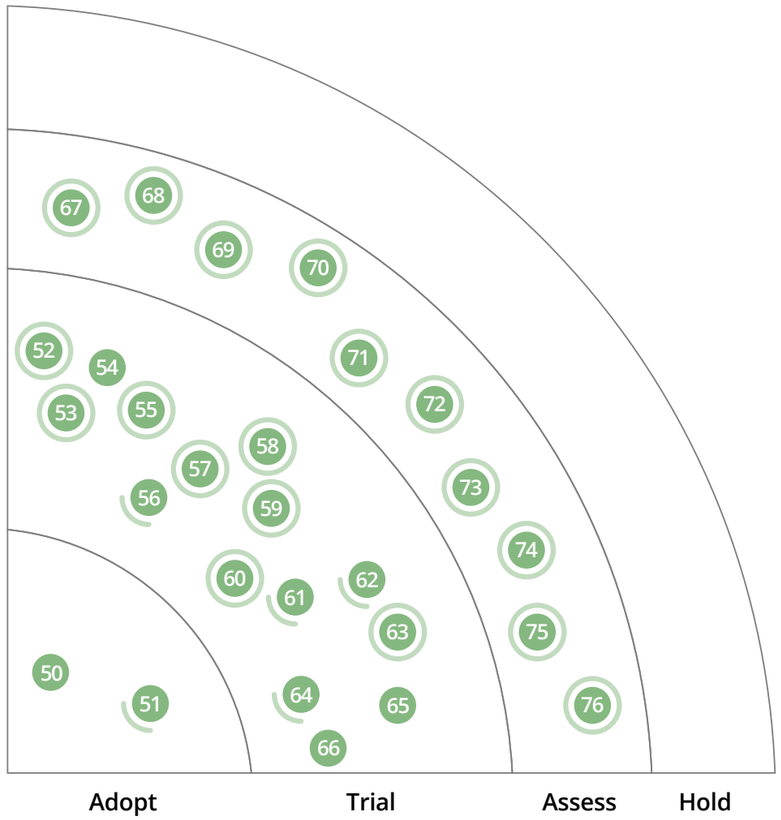
Tools

Cypress
Adopt

Cypress is still a favorite among our teams where developers manage end-to-end tests themselves, as part of a healthy test pyramid, of course. We decided to call it out again in this Radar because recent versions of Cypress have added support for Firefox, and we strongly suggest testing on multiple browsers. The dominance of Chrome and Chromium-based browsers has led to a worrying trend of teams seemingly only testing with Chrome which can lead to nasty surprises.

Figma
Adopt

Figma has demonstrated to be the go-to tool for collaborative design, not only for designers but for multidisciplinary teams too; it allows developers and other roles to view and comment on designs through the browser without the desktop version. Compared to its competitors (e.g., Invision or Sketch) which have you use more than one tool for versioning, collaborating and design sharing, Figma puts together all of these features in one tool that makes it easier for our teams to discover new ideas together. Our teams find Figma very useful, especially in remote and distributed design work enablement and facilitation. In addition to its real-time design and collaboration capabilities, Figma also offers an API that helps to improve the DesignOps process.



Dojo
Trial

A few years ago, Docker — and containers in general — radically changed how we think about packaging, deploying and running our applications. But despite this improvement in production, developers still spend a lot of time setting up development environments and regularly run into "but it works on my machine" style problems. Dojo aims to fix this by creating standard development environments, versioned and released as Docker images. Several of our teams use Dojo to streamline developing, testing and building code from local development through production pipelines.

DVC
Trial

In 2018 we mentioned DVC in conjunction with the versioning data for reproducible analytics. Since then it has become a favorite tool for managing experiments in

Adopt
50. Cypress
51. Figma

Trial
52. Dojo
53. DVC
54. Experiment tracking tools for machine learning
55. Goss
56. Jaeger
57. k9s
58. kind
59. mkcert
60. MURAL
61. Open Policy Agent (OPA)
62. Optimal Workshop
63. Phrase
64. ScoutSuite
65. Visual regression testing tools
66. Visual Studio Live Share

Assess
67. Apache Superset
68. AsyncAPI
69. ConfigCat
70. Gitpod
71. Gloo
72. Lens
73. Manifold
74. Sizzy
75. Snowpack
76. tfsec

Hold

Tools

Jaeger is an open source distributed tracing system. We've used it successfully with Istio and Envoy on Kubernetes, and like its UI.

(Jaeger)

k9s provides an interactive interface for basically everything that kubectl can do. And to boot, it's not a web application but runs inside a terminal window.

(k9s)

machine learning (ML) projects. Since it's based on Git, DVC is a familiar environment for software developers to bring their engineering practices to ML practice. Because it versions the code that processes data along with the data itself and tracks stages in a pipeline, it helps bring order to the modeling activities without interrupting the analysts' flow.

Experiment tracking tools for machine learning

Trial

The day-to-day work of machine learning often boils down to a series of experiments in selecting a modeling approach and the network topology, training data and optimizing or tweaking the model. Data scientists must use experience and intuition to hypothesize changes and then measure the impact those changes have on the overall performance of the model. As this practice has matured, our teams have found an increasing need for experiment tracking tools for machine learning. These tools help investigators keep track of the experiments and work through them methodically. Although no clear winner has emerged, tools such as MLflow and platforms such as Comet or Neptune have introduced rigor and repeatability into the entire machine learning workflow.

Goss

Trial

We mentioned Goss, a tool for provisioning testing, in passing in previous Radars, for example, when describing the technique of TDD'ing containers. Although Goss isn't always an alternative to Serverspec, simply because it doesn't offer the same amount of features, you may want to consider it when its features meet your needs, especially since it comes as a small, self-contained binary (rather than requiring a Ruby environment). A common anti-pattern with using tools such as Goss is double-entry bookkeeping, where each change in the actual infrastructure as code files requires a corresponding change in the test assertions. Such tests are maintenance heavy and because of the close correspondence between code and test, failures mostly occur when an engineer updates one side and forgets the other. And these tests rarely catch genuine problems.

Jaeger

Trial

Jaeger is an open source distributed tracing system. Similar to Zipkin, it's been inspired by the Google Dapper paper and complies with OpenTelemetry. We've used Jaeger successfully with Istio and Envoy on Kubernetes and like its UI. Jaeger exposes tracing metrics in the Prometheus format so they can be made available to other tools. However, a new generation of tools such as Honeycomb integrates traces and metrics into a single observability stream for simpler aggregate analysis. Jaeger joined CNCF in 2017 and has recently been elevated to CNCF's highest level of maturity, indicating its widespread deployment into production systems.

k9s

Trial

We continue to be ardent supporters of infrastructure as code, and we continue to believe that a robust monitoring solution is a prerequisite for operating distributed applications. Sometimes an interactive tool such as the AWS web console can be a useful addition. It allows us to explore all kinds of resources in an ad-hoc fashion without having to remember every single obscure command. Using an interactive tool to make manual modifications on the fly is still a questionable practice, though. For Kubernetes we now have k9s, which provides an interactive interface for basically everything that kubectl can do. And to boot, it's not a web application but runs inside a terminal window, evoking fond memories of Midnight Commander for some of us.

kind

Trial

kind is a tool for running local Kubernetes clusters using Docker container nodes. With kubetest integration, kind makes it easy to do end-to-end testing on Kubernetes. We've used kind to create ephemeral Kubernetes clusters to test Kubernetes resources such as Operators and Custom Resource Definitions (CRDs) in our CI pipelines.

mkcert

Trial

mkcert is a convenient tool for creating locally trusted development certificates. Using certificates from real certificate authorities (CAs) for local development can be challenging if not impossible (for hosts such as example.test, localhost or 127.0.0.1). In such situations self-signed certificates may be your only option. mkcert lets you generate self-signed certificates and installs the local CA in the system root store. For anything other than local development and testing, we strongly recommend using certificates from real CAs to avoid trust issues.

MURAL
Trial

MURAL describes itself as a "digital workspace for visual collaboration" and allows teams to interact with a shared workspace based on a whiteboard/sticky notes metaphor. Its features include voting, commenting, notes and "follow the presenter." We particularly like the template feature that allows a facilitator to design and then reuse guided sessions with a team. Each of the major collaboration suites have a tool in this space (for example, Google Jamboard and Microsoft Whiteboard) and these are worth investigating, but we've found MURAL to be slick, effective and flexible.

Open Policy Agent (OPA)
Trial

Open Policy Agent (OPA) has rapidly become a favorable component of many distributed cloud-native solutions that we build for our clients. OPA provides a uniform framework and language for declaring, enforcing and controlling policies for various components of a cloud-native solution. It's a great example of a tool that implements security policy as code. We've had a smooth experience using OPA in multiple scenarios, including deploying resources to K8s clusters, enforcing access control across services in a service mesh and fine-grained security controls as code for accessing application resources. A recent commercial offering, Styra's Declarative Authorization Service (DAS), eases the adoption of OPA for enterprises by adding a management tool, or control plane, to OPA for K8s with a prebuilt policy library, impact analysis of the policies and logging capabilities. We look forward to maturity and extension of OPA beyond operational services to (big) data-centric solutions.

Optimal Workshop
Trial

UX research demands data collection and analysis to make better decisions about the products we need to build. Our teams find Optimal Workshop useful because it makes it easy to validate prototypes and configure tests for data collection and thus make better decisions. Features such as first-click, card sorting, or a heatmap of user interaction help to both validate prototypes and improve website navigation and information display. It's an ideal tool for distributed teams since it allows them to conduct remote research.

Phrase
Trial

As mentioned in our description of Crowdin, you now have a choice of platforms to manage the translation of a product into multiple languages instead of emailing large spreadsheets. Our teams report positive experiences with Phrase, emphasizing that it's easy to use for all key user groups. Translators use a convenient browser-based UI. Managers can add new fields and synchronize translations with other teams in the same UI. Developers can access Phrase locally and from a build pipeline. A feature that deserves a specific mention is the ability to apply versioning to translations through tags, which makes it possible to compare the look of different translations inside the actual product.

ScoutSuite
Trial

ScoutSuite is an expanded and updated tool based on Scout2 (featured in the Radar in 2018) that provides security posture assessment across AWS, Azure, GCP and other cloud providers. It works by automatically aggregating configuration data for an environment and applying rules to audit the environment. We've found this very useful across projects for doing point-in-time security assessments.

Visual regression testing tools
Trial

Since we first mentioned visual regression testing tools in 2014, the use of the technique has spread and the tools landscape has evolved. BackstopJS remains an excellent choice with new features being added regularly, including support for running inside Docker containers. Loki was featured in our previous Radar. Applitools, CrossBrowserTesting and Percy are SaaS solutions. Another notable mention is Resemble.js, an image diffing library. Although most teams use it indirectly as part of BackstopJS, some of our teams have been using it to analyze and compare images of web pages directly. In general, our experience shows that visual regression tools are less useful in the early stages when the interface goes through significant changes, but they certainly prove their worth as the product matures and the interface stabilizes.

Visual Studio Live Share
Trial

Visual Studio Live Share is a suite of extensions for Visual Studio Code and Visual Studio. At a time when teams are searching for good remote collaboration options, we want to call attention to the excellent tooling here. Live Share provides a good, low-latency remote-pairing experience, and requires significantly less bandwidth than the brute-force approach of sharing your entire desktop. Importantly, developers can

Tools

OPA provides a uniform framework and language for declaring, enforcing and controlling policies for various components of a cloud-native solution.

(Open Policy Agent (OPA))

Our teams find Optimal Workshop useful because it makes it easy to validate prototypes and configure tests for data collection and thus make better decisions.

(Optimal Workshop)

Tools

AsyncAPI is an open source initiative to create a much needed event-driven and asynchronous API standardization and development tooling.

(AsyncAPI)

ConfigCat supports simple feature toggles, user segmentation, and A/B testing and has a generous free tier for low-volume use cases or those just starting out.

(ConfigCat)

work with their preferred configuration, extensions and key mappings during a pairing session. In addition to real-time collaboration for editing and debugging code, Live Share allows voice calls and sharing terminals and servers.

Apache Superset

Assess

Apache Superset is a great business intelligence (BI) tool for data exploration and visualization to work with large data lake and data warehouse setups. It works, for example, with Presto, Amazon Athena and Amazon Redshift and can be nicely integrated with enterprise authentication. Moreover, you don't have to be a data engineer to use it; it's meant to benefit all engineers exploring data in their everyday work. It's worth pointing out that Apache Superset is currently undergoing incubation at the Apache Software Foundation (ASF), meaning it's not yet fully endorsed by ASF.

AsyncAPI

Assess

Open standards are one of the foundational pillars of building distributed systems. For example, the OpenAPI (formerly Swagger) specification, as an industry standard to define RESTful APIs, has been instrumental to the success of distributed architectures such as microservices. It has enabled a proliferation of tooling to support building, testing and monitoring RESTful APIs. However, such standardizations have been largely missing in distributed systems for event-driven APIs.

AsyncAPI is an open source initiative to create a much needed event-driven and asynchronous API standardization and development tooling. The AsyncAPI specification, inspired by the OpenAPI specification, describes and documents event-driven APIs in a machine-readable format. It's protocol agnostic, so it can be used for APIs that work over many protocols, including MQTT, WebSockets, and Kafka. We're eager to see the ongoing improvements of AsyncAPI and further maturity of its tooling ecosystem.

ConfigCat

Assess

If you're looking for a service to support dynamic feature toggles (and bear in mind that simple feature toggles work well too), check out ConfigCat. We'd describe it as "like LaunchDarkly but cheaper and a bit less fancy" and find that it does most of what we need. ConfigCat supports simple feature toggles, user segmentation, and A/B testing and has a generous free tier for low-volume use cases or those just starting out.

Gitpod

Assess

You can build most software following a simple two-step process: check out a repository, and then run a single build script. The process of setting up a full coding environment can still be cumbersome, though. Gitpod addresses this by providing cloud-based, "ready-to-code" environments for Github or GitLab repositories. It offers an IDE based on Visual Studio Code that runs inside the web browser. By default, these environments are launched on the Google Cloud Platform, although you can also deploy on-premise solutions. We see the immediate appeal, especially for open source software where this approach can lower the bar for casual contributors. However, it remains to be seen how viable this approach will be in corporate environments.

Gloo

Assess

With the increasing adoption of Kubernetes and service mesh, API gateways have been experiencing an existential crisis in cloud-native distributed systems. After all, many of their capabilities (such as traffic control, security, routing and observability) are now provided by the cluster's ingress controller and mesh gateway. Gloo is a lightweight API gateway that embraces this change; it uses Envoy as its gateway technology, while providing added value such as a cohesive view of the APIs to the external users and applications. It also provides an administrative interface for controlling Envoy gateways and runs and integrates with multiple service mesh implementations such as Linkerd, Istio and AWS App Mesh. While its open source implementation provides the basic capabilities expected from an API gateway, its enterprise edition has a more mature set of security controls such as API key management or integration with OPA. Gloo is a promising lightweight API gateway that plays well with the ecosystem of cloud-native technology and architecture, while avoiding the API gateway trap of enabling business logic to glue APIs for the end user.

Lens

Assess

One of the strengths of Kubernetes is its flexibility and range of configuration possibilities along with the API-driven, programmable configuration mechanisms and command-line visibility and control using manifest files. However, that strength can also be a weakness: when deployments are complex or when managing multiple clusters, it can be difficult to get a clear picture of the overall status through command-line arguments and manifests alone. Lens attempts to solve this problem

with an integrated environment for viewing the current state of the cluster and its workloads, visualizing cluster metrics and changing configurations through an embedded text editor. Rather than a simple point-and-click interface, Lens brings together the tools an administrator would run from the command line into a single, navigable interface. This tool is one of several approaches that are trying to tame the complexity of Kubernetes management. We've yet to see a clear winner in this space, but Lens strikes an interesting balance between a graphical UI and command-line–only tools.

Manifold

Assess

Manifold is a model-agnostic visual debugger for machine learning (ML). Model developers spend a significant amount of time on iterating and improving an existing model rather than creating a new one. By shifting the focus from model space to data space, Manifold supplements the existing performance metrics with a visual characteristics of the data set that influences the model performance. We think Manifold will be a useful tool to assess in the ML ecosystem.

Sizzy

Assess

Building web applications that look just as intended on a large number of devices and screen sizes can be cumbersome. Sizzy is a SaaS solution that shows many viewports in a single browser window. The application is rendered in all viewports simultaneously and interactions with the application are also synched across the viewports. In our experience interacting with an application in this way can make it easier to spot potential issues earlier, before a visual regression testing tool flags the issue in the build pipeline. We should mention, though, that some of our developers who tried Sizzy for a while did, on balance, prefer to work with the tooling provided by Chrome.

Snowpack

Assess

Snowpack is an interesting new entrant in the field of JavaScript build tools. The key improvement over other solutions is that Snowpack makes it possible to build applications with modern frameworks such as React.js, Vue.js, and Angular without the need for a bundler. Cutting out the bundling step dramatically improves the feedback cycle during development because changes become available in the browser almost immediately. For this magic to work, Snowpack transforms the dependencies in `node_modules` into single JavaScript files in a new `web_modules` directory, from where they can be imported as an ECMAScript module (ESM). For IE11 and other browsers that don't support ESM, a workaround is available. Unfortunately, because no browser today can import CSS from JavaScript, using CSS modules is not straightforward.

tfsec

Assess

Security is everyone's concern and capturing risks early is always better than facing problems later on. In the infrastructure as code space, where Terraform is an obvious choice to manage cloud environments, we now also have tfsec, which is a static analysis tool that helps to scan Terraform templates and find any potential security issues. It comes with preset rules for different cloud providers including AWS and Azure. We always like tools that help to mitigate security risks, and tfsec not only excels in identifying security risks, it's also easy to install and use.

Tools

Lens is one of several different approaches that are trying to tame the complexity of Kubernetes management.

(Lens)

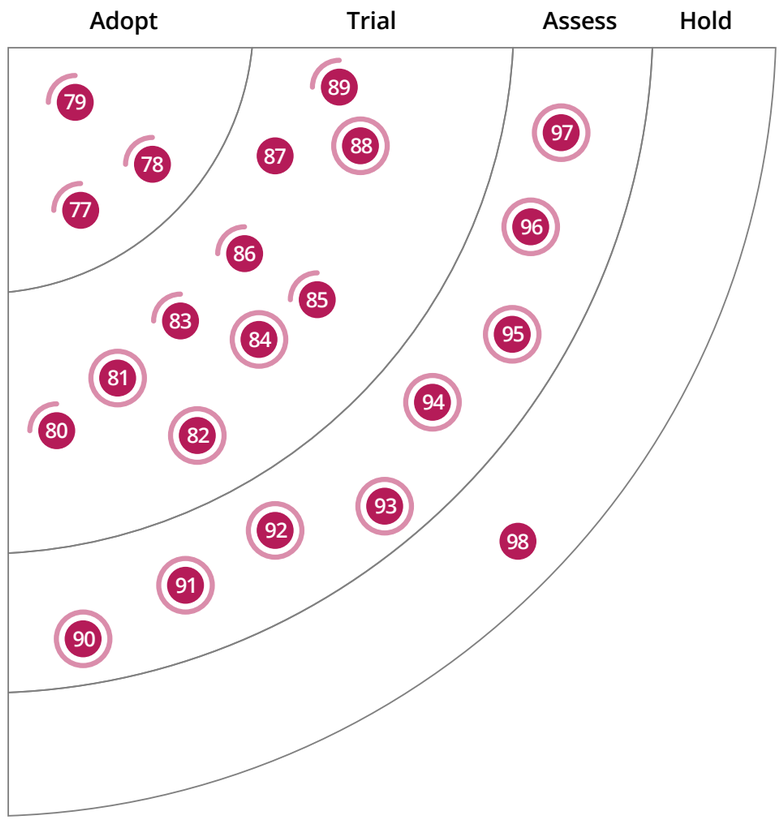
tfsec is a static analysis tool that helps to scan Terraform templates and find any potential security issues.

(tfsec)

TECHNOLOGY RADAR Vol. 22

Languages & Frameworks

Languages & Frameworks



Adopt
77. React Hooks
78. React Testing Library
79. Vue.js

Trial
80. CSS-in-JS
81. Exposed
82. GraphQL Inspector
83. Karate
84. Koin
85. NestJS
86. PyTorch
87. Rust
88. Sarama
89. SwiftUI

Assess
90. Clinic.js Bubbleprof
91. Deequ
92. ERNIE
93. MediaPipe
94. Tailwind CSS
95. Tamer
96. Wire
97. XState

Hold
98. Enzyme

React Hooks
Adopt

React Hooks have introduced a new approach to managing stateful logic; given React components have always been closer to functions than classes, Hooks have embraced this and brought state to the functions, instead of taking function as methods to the state with classes. Based on our experience, Hooks improve reuse of functionality among components and code readability. Given Hooks' testability improvements, using React Test Renderer and React Testing Library, and their growing community support, we consider them our approach of choice.

React Testing Library
Adopt

The JavaScript world moves pretty fast, and as we gain more experience using a framework our recommendations change. The React Testing Library is a good example of a framework that with deeper usage has eclipsed the alternatives to become the sensible default when testing React-based frontends. Our teams like the fact that tests written with this framework are less brittle than with alternative frameworks such as Enzyme, because you're encouraged to test component relationships individually as opposed to testing all implementation details. This mindset is brought by Testing Library which React Testing Library is part of and which provides a whole family of libraries for Angular and Vue.js, for example.

Vue.js
Adopt

Vue.js has become one of the successfully applied, loved and trusted frontend JavaScript frameworks among our community. Although there are other, well-adopted alternatives, such as React.js, the simplicity of Vue.js in API design, its clear segregation of directives and components (one file per component idiom) and its simpler state management have made it a compelling option among others.

CSS-in-JS
Trial

Since we first mentioned CSS-in-JS as an emerging technique in 2017, it has become much more popular, a trend we also see in our work. With some solid production experience under our belts,

Languages & Frameworks

Koin is a Kotlin framework that handles one of the routine problems in software development, dependency injection.

(Koin)

NestJS is a TypeScript-first framework that makes the development of NodeJS applications safer and less error prone.

(NestJS)

we can now recommend CSS-in-JS as a technique to trial. A good starting point is the styled components framework, which we mentioned in our previous Radar. Next to all the positives, though, there usually is a downside when using CSS-in-JS: the calculation of styles at runtime can cause a noticeable lag for end users. With Linaria we're now seeing a new class of frameworks that were created with this issue in mind. Linaria employs a number of techniques to shift most of the performance overhead to build time. Alas, this does come with its own set of trade-offs, most notably a lack of dynamic style support in IE11.

Exposed
Trial

Through their extended use of Kotlin, our development teams have gained experience with more frameworks designed specifically for Kotlin rather than using Java frameworks with Kotlin. Although it's been around for a while, Exposed has caught our attention as a lightweight object-relational mapper (ORM). Exposed has two flavors of database access: a typesafe internal DSL wrapping SQL and an implementation of the data access object (DAO) pattern. It supports features expected from a mature ORM such as handling of many-to-many references, eager loading, and support for joins across entities. We also like that the implementation works without proxies and doesn't rely on reflection, which is certainly beneficial to performance.

GraphQL Inspector
Trial

GraphQL Inspector lets you compare changes between two GraphQL schemas. We've cautioned against the use of GraphQL in the past, and we're happy to see some improvements in tooling around GraphQL since. Most of our teams continue to use GraphQL for server-side resource aggregation, and by integrating GraphQL Inspector in their CI pipelines, we've been able to catch potential breaking changes in the GraphQL schema.

Karate
Trial

Given our experience that tests are the only API specifications that really matter, we're always on the lookout for new tools that might help with testing. Karate is an API testing framework whose unique feature is that tests are written directly in Gherkin without relying on a general-purpose programming language to implement test behavior. It's a domain-specific language for describing HTTP-based API tests. Our teams like the readable specification that they get with this tool and recommend to keep tests with Karate in the upper levels of the testing pyramid and not overload its use by making very detailed assertions.

Koin
Trial

As Kotlin is used increasingly for both mobile and server-side development, the associated ecosystem continues to evolve. Koin is a Kotlin framework that handles one of the routine problems in software development: dependency injection. Although you can choose from a variety of dependency injection frameworks for Kotlin, our teams have come to prefer the simplicity of Koin. Koin avoids using annotations and injects either through constructors or by mimicking Kotlin's lazy initialization so that objects are injected only when needed. This is in contrast to the statically compiled Dagger injection framework for Android. Our developers like the lightweight nature of this framework and its built-in testability.

NestJS
Trial

The growth in popularity of Node.js and trends such as Node overload have led to the application of Node.js for developing business applications. We often see problems, such as scalability and maintainability, with large JavaScript-based applications. NestJS is a TypeScript-first framework that makes the development of Node.js applications safer and less error prone. NestJS is opinionated and comes with SOLID principles and an Angular-inspired architecture out of the box. When building Node.js microservices, NestJS is one of the frameworks that our teams commonly use to empower developers to create testable, scalable, loosely coupled and easily maintainable applications.

PyTorch
Trial

Our teams have continued to use and appreciate the PyTorch machine learning framework, and several teams prefer PyTorch over TensorFlow. PyTorch exposes the inner workings of ML that TensorFlow hides, making it easier to debug, and contains constructs that programmers are familiar with such as loops and actions. Recent releases have improved performance of PyTorch, and we've been using it successfully in production projects.

Rust
Trial

Rust is continuously gaining in popularity. We've had heated discussions about which is better, Rust or C++/Go, without a clear winner. However, we're glad to see Rust has improved significantly, with more built-in APIs being added and stabilized, including advanced async support, since we mentioned it in our previous Radar. In addition, Rust has also inspired the design of new languages. For example, the Move language on Libra borrows Rust's way of managing memory to manage resources, ensuring that digital assets can never be copied or implicitly discarded.

Sarama
Trial

Sarama is a Go client library for Apache Kafka. If you're developing your APIs in Go, you'll find Sarama quite easy to set up and manage as it doesn't depend on any native libraries. Sarama has two types of APIs — a high-level API for easily producing and consuming messages and a low-level API for controlling bytes on the wire.

SwiftUI
Trial

Apple has taken a big step forward with their new SwiftUI framework for implementing user interfaces on the macOS and iOS platforms. We like that SwiftUI moves beyond the somewhat kludgy relationship between Interface Builder and Xcode and adopts a coherent, declarative and code-centric approach. You can now view your code and the resulting visual interface side by side in Xcode 11, making for a much better developer experience. The SwiftUI framework also draws inspiration from the React.js world that has dominated web development in recent years. Immutable values in view models and an asynchronous update mechanism make for a unified reactive programming model. This gives developers an entirely native alternative to similar reactive frameworks such as React Native or Flutter. SwiftUI definitely represents the future of Apple UI development, and although new, it has shown its benefits. We've been having great experience with it — and its shallow learning curve. It's worth noting that you should know your customer's use case before jumping into using SwiftUI, given that it doesn't support iOS 12 or below.

Clinic.js Bubbleprof
Assess

With the aim of improving performance in our code, profiling tools are very useful to identify bottlenecks or delays in code which are hard to identify, especially in asynchronous operations. Clinic.js Bubbleprof represents visually the async operations in Node.js processes, drawing a map of delays in the application's flow. We like this tool because it helps developers to easily identify and prioritize what to improve in the code.

Deequ
Assess

There are still some tool gaps when applying good software engineering practices in data engineering. Attempting to automate data quality checks between different steps in a data pipeline, one of our teams was surprised when they found only a few tools in this space. They settled on Deequ, a library for writing tests that resemble unit tests for data sets. Deequ is built on top of Apache Spark, and even though it's published by AWS Labs it can be used in environments other than AWS.

ERNIE
Assess

In the previous edition of the Radar we had BERT — which is a key milestone in the NLP landscape. Last year, Baidu released ERNIE 2.0 (Enhanced Representation through kNowledge IntEgration) which outperformed BERT on seven GLUE language understanding tasks and on all nine of the Chinese NLP tasks. ERNIE, like BERT, provides unsupervised pretrained language models, which can be fine-tuned by adding output layers to create state-of-the-art models for a variety of NLP tasks. ERNIE differs from traditional pretraining methods in that it is a continual pretraining framework. Instead of training with a small number of pretraining objectives, it could constantly introduce a large variety of pretraining tasks to help the model efficiently learn language representations. We're pretty excited about the advancements in NLP and are looking forward to experimenting with ERNIE on our projects.

MediaPipe
Assess

MediaPipe is a framework for building MultiModal (such as video, audio, time series data, etc.), cross-platform (for example, Android, iOS, Web, and edge devices) and applied ML pipelines. It provides multiple capabilities, including face

Languages & Frameworks

Deequ is a library for writing tests that resemble unit tests for data sets which can help when automating data quality checks between different steps in a data pipeline.

(Deequ)

ERNIE provides unsupervised pretrained language models, which can be fine-tuned by adding output layers to create state-of-the-art models for a variety of NLP tasks.

(ERNIE)

Languages & Frameworks

Tailwind CSS proposes an interesting approach by providing lower-level utility CSS classes to create building blocks without opinionated styles and aiming for easy customization.

(Tailwind CSS)

Wire is a compile-time dependency injection tool that can both generate code and wire components together.

(Wire)

detection, hand tracking, gesture detection and object detection. Although MediaPipe is primarily deployed to mobile devices, it's started to show up in the browser thanks to WebAssembly and XNNPack ML Inference Library. We're exploring MediaPipe for some AR use cases and like what we see so far.

Tailwind CSS
Assess

CSS tools and frameworks offer predesigned components for fast results; after a while, however, they can complicate customization. Tailwind CSS proposes an interesting approach by providing lower-level utility CSS classes to create building blocks without opinionated styles and aiming for easy customization. The breadth of the low-level utilities allows you to avoid writing any classes or CSS on your own which leads to a more maintainable codebase in the long term. It seems that Tailwind CSS offers the right balance between reusability and customization to create visual components.

Tamer
Assess

If you need to ingest data from relational databases into a Kafka topic, consider Tamer, which labels itself "a domesticated JDBC source connector for Kafka." Despite being a relatively new framework, we've found Tamer to be more efficient than the Kafka JDBC connector, especially when huge amounts of data are involved.

Wire
Assess

The Golang community has had its fair share of dependency injection skeptics, partly because they confused the pattern with specific frameworks, and developers with a system-programming background naturally dislike runtime overhead caused by reflection. Then along came Wire, a compile-time dependency injection tool that can generate code and wire components together. Wire has no additional runtime overhead, and the static dependency graph is easier to reason about. Whether you handwrite your code or use frameworks, we recommend using dependency injection to encourage modular and testable designs.

XState
Assess

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

Enzyme
Hold

We don't always move deprecated tools to Hold in the Radar, but our teams feel strongly that Enzyme has been replaced for unit testing React UI components by React Testing Library. Teams using Enzyme have found that its focus on testing component internals leads to brittle, unmaintainable tests.

ThoughtWorks®

We are a software consultancy and community of passionate purpose-led individuals, 7000+ people strong across 43 offices in 14 countries. Over our 25+ year history, we have helped our clients solve complex business problems where technology is the differentiator. When the only constant is change, we prepare you for the unpredictable.

Want to stay up-to-date with all Radar-related news and insights?

Follow us on your favorite social channel or become a subscriber.

subscribe now

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar