ThoughtWorks®

TECHNOLOGY RADAR VOL. 21

An opinionated guide to technology frontiers

thoughtworks.com/radar
#TWTechRadar

CONTRIBUTORS

The Technology Radar is prepared by the ThoughtWorks Technology Advisory Board

This edition of the ThoughtWorks Technology Radar is based on a meeting of the Technology Advisory Board in San Francisco in October 2019

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Jonny LeRoy

Ketan Padegaonkar

Lakshminarasimhan Sudarshan

Marco Valtas

Mike Mason

Neal Ford

Ni Wang

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

ABOUT THE RADAR

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

1 ADOPT

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

2 TRIAL

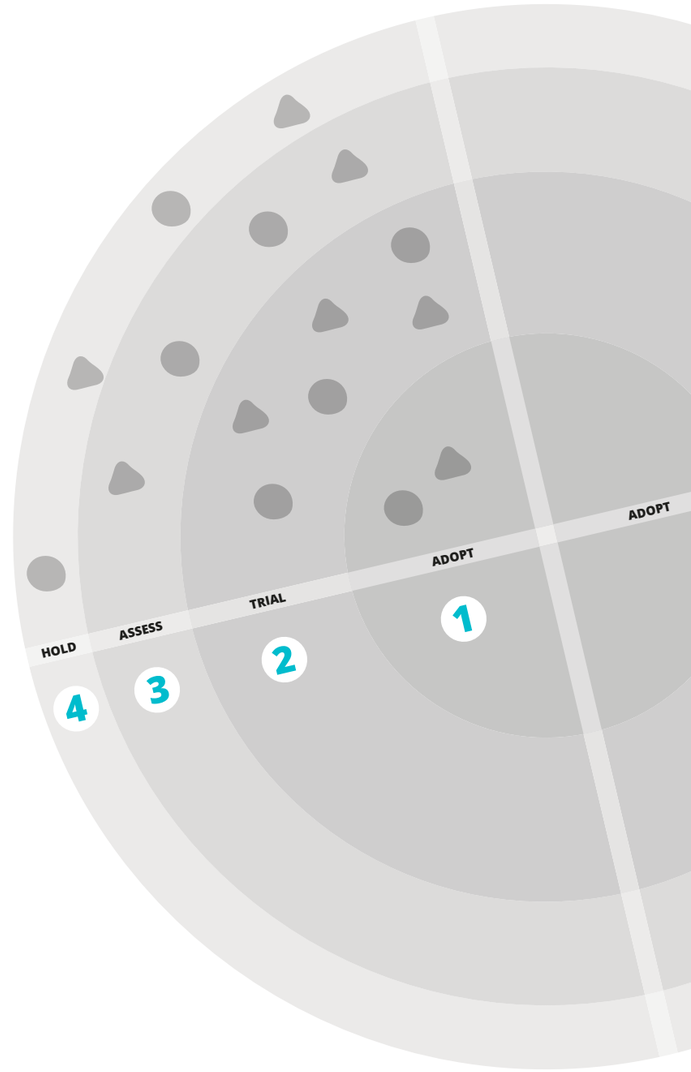
Worth pursuing. It's important to understand how to build up this capability. Enterprises can try this technology on a project that can handle the risk.

3 ASSESS

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

4 HOLD

Proceed with caution.



NEW OR CHANGED
NO CHANGE

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

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

WHAT'S NEW

Highlighted themes in this edition

Cloud: Is More Less?

As the major cloud providers have achieved near parity on core functionality, the competitive focus has moved to the extra services they can provide, encouraging them to release new offerings at breakneck speed. In their haste to compete, compete, they release new services with rough edges and incomplete features. The emphasis on speed and product proliferation, through either acquisition or hastily created services, often results not merely in bugs but also in poor documentation, difficult automation and incomplete integration with vendors' own parts. This causes frustration for teams trying to deliver software using functionality promised by the cloud provider yet constantly hitting roadblocks. Companies choose cloud vendors for a variety of factors and often at a high level in the organization. Our advice for teams: don't assume that all of your designated cloud provider's services are of equal quality, test out key capabilities and be open to alternative open-source options or a polycloud strategy, if your own time-to-market trade-offs merit the operational overhead of managing them.

Protecting the Software Supply Chain

Organizations should resist ivory tower governance rules that require lengthy manual inspection and approval; rather, automated dependency protection (Dependency drift fitness function), security (Security policy as code) and other governance mechanisms (Run cost as architecture fitness function) protect the *important* but not *urgent* parts of software projects. This topic concerning policy, compliance and governance as code reappeared multiple times in our conversations. We see a natural evolution in the software development ecosystem of increasing automation: continuous integration with automated testing, continuous delivery, infrastructure as code, and now automated governance. Building automation around cloud cost, dependency management, architectural structure and other former manual processes shows a natural evolution; we're learning how we can automate all important aspects of software delivery.

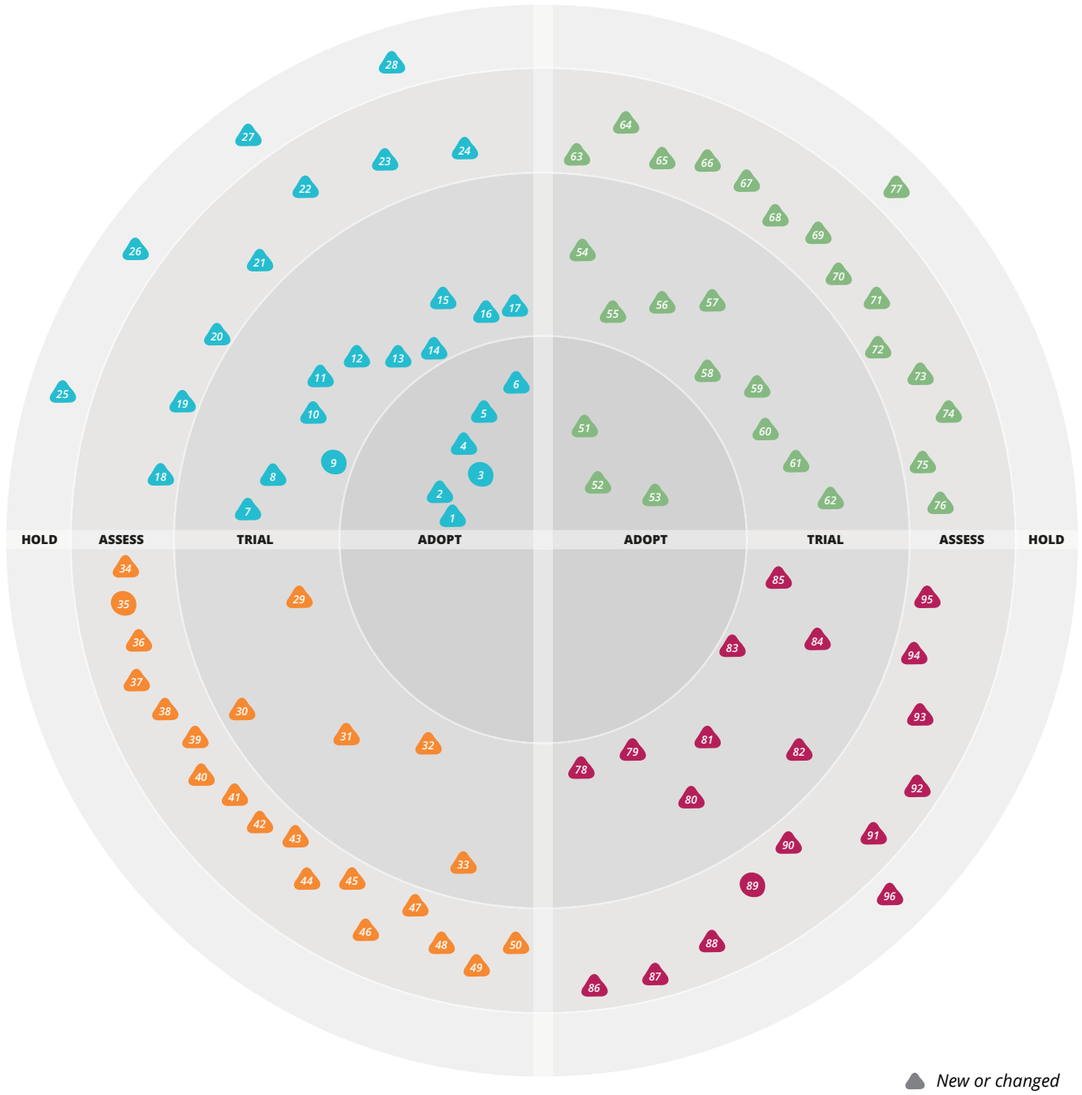
Interpreting the Black Box of ML

Machine learning often appears to discover solutions to problems that humans can't, using pattern matching, back propagation and other well-known techniques. However, despite their power, many of these models are inherently opaque, meaning that their results can't be explained in terms of logical inference. This is a problem when humans have a right to know how a decision was made or when there is a risk of introducing prejudice, sampling, algorithmic or other bias into the model. We're now seeing the emergence of tools such as What-If and techniques such as ethical bias testing that help us find the limitations and predict the output of these models. While these improvements in *interpretability* are a step in the right direction, *explaining* deep neural networks remains an elusive goal. For that reason, data scientists are beginning to regard explainability as a first class-concern when choosing a machine learning model.

Software Development as a Team Sport

Since the early days of our Technology Radar, we've warned against tools and techniques that isolate members of software teams from one another, hampering feedback and collaboration. Often, when new specializations come along, practitioners, vendors and tools insist that some part of development must be done in an isolated environment, away from the chaos of "regular" development. We reject that claim and constantly look for new ways to reengage software development as a team sport. Feedback is critical when developing something as complex as software. While projects increasingly require specialization, we strive to fit them into regular collaboration and feedback. We particularly dislike the "10x engineers" meme and prefer to focus on creating and enabling "10x teams." We see this currently playing out in how design, data science and security can be integrated into cross-functional teams and supported with solid automation. The next frontier is bringing more governance and compliance activities into the fold.

THE RADAR



TECHNIQUES

ADOPT
1. Container security scanning
2. Data integrity at the origin
3. Micro frontends
4. Pipelines for infrastructure as code
5. Run cost as architecture fitness function
6. Testing using real device

TRIAL
7. Automated machine learning (AutoML)
8. Binary attestation
9. Continuous delivery for machine learning (CD4ML)
10. Data discoverability
11. Dependency drift fitness function
12. Design systems
13. Experiment tracking tools for machine learning
14. Explainability as a first-class model selection criterion
15. Security policy as code
16. Sidecars for endpoint security
17. Zhong Tai

ASSESS
18. BERT
19. Data mesh
20. Ethical bias testing
21. Federated learning
22. JAMstack
23. Privacy-preserving record linkage (PPRL) using Bloom filter
24. Semi-supervised learning loops

HOLD
25. 10x engineers
26. Front-end integration via artifact
27. Lambda pinball
28. Legacy migration feature parity

PLATFORMS

ADOPT

TRIAL
29. Apache Flink
30. Apollo Auto
31. GCP Pub/Sub
32. Mongoose OS
33. ROS

ASSESS
34. AWS Cloud Development Kit
35. Azure DevOps
36. Azure Pipelines
37. Crowdin
38. Crux
39. Delta Lake
40. Fission
41. FoundationDB
42. GraalVM
43. Hydra
44. Kuma
45. MicroK8s
46. Oculus Quest
47. ONNX
48. Rootless containers
49. Snowflake
50. Teleport

HOLD

TOOLS

ADOPT
51. Commitizen
52. ESLint
53. React Styleguidist

TRIAL
54. Bitrise
55. Dependabot
56. Detekt
57. Figma
58. Jib
59. Loki
60. Trivy
61. Twistlock
62. Yocto Project

ASSESS
63. Aplas
64. asdf-vm
65. AWSume
66. dbt
67. Docker Notary
68. Facets
69. Falco
70. in-toto
71. Kubeflow
72. MemGuard
73. Open Policy Agent (OPA)
74. Pumba
75. Skaffold
76. What-If Tool

HOLD
77. Azure Data Factory for orchestration

LANGUAGES & FRAMEWORKS

ADOPT

TRIAL
78. Arrow
79. Flutter
80. jest-when
81. Micronaut
82. React Hooks
83. React Testing Library
84. Styled components
85. Tensorflow

ASSESS
86. Fairseq
87. Flair
88. Gatsby.js
89. GraphQL
90. KotlinTest
91. NestJS
92. Paged.js
93. Quarkus
94. SwiftUI
95. Testcontainers

HOLD
96. Enzyme

TECHNIQUES

Container security scanning
ADOPT

The continued adoption of containers for deployments, especially Docker, has made container security scanning a must-have technique and we've moved this technique into Adopt to reflect that. Specifically, containers introduced a new path for security issues; it's vital that you use tools to scan and check containers during deployment. We prefer using automated scanning tools that run as part of the deployment pipeline.

Data integrity at the origin
ADOPT

Today, many organizations' answer to unlocking data for analytical usage is to build a labyrinth of data pipelines. Pipelines retrieve data from one or multiple sources, cleanse it and then transform and move it to another location for consumption. This approach to data management often leaves the consuming pipelines with the difficult task of verifying the inbound data's integrity and building complex logic to cleanse the data to meet its required level of quality. The fundamental problem is that the source of the data has no incentive and accountability for providing quality data to its consumers. For this reason, we strongly advocate for data integrity at the origin, by which we mean, any source that provides consumable data must describe its measures of data quality explicitly and guarantee those measures. The main reason behind this is that the originating systems and teams are most intimately familiar with their data and best positioned to fix it at the source. Data mesh architecture takes this one step further, comparing consumable data to a *product*, where data quality and its objectives are integral attributes of every shared data set.

Micro frontends
ADOPT

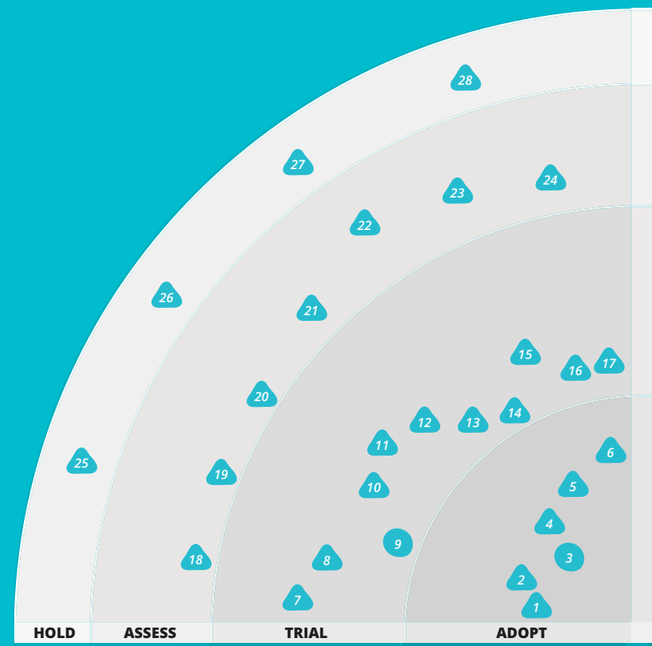
We've seen significant benefits from introducing microservices, which have allowed teams to scale the delivery of independently deployed and maintained services. Unfortunately, we've also seen many teams create a front-end monolith — a large, entangled browser application that sits on top of the back-end 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 this 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 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.

ADOPT
1. Container security scanning
2. Data integrity at the origin
3. Micro frontends
4. Pipelines for infrastructure as code
5. Run cost as architecture fitness function
6. Testing using real device

TRIAL
7. Automated machine learning (AutoML)
8. Binary attestation
9. Continuous delivery for machine learning (CD4ML)
10. Data discoverability
11. Dependency drift fitness function
12. Design systems
13. Experiment tracking tools for machine learning
14. Explainability as a first-class model selection criterion
15. Security policy as code
16. Sidecars for endpoint security
17. Zhong Tai

ASSESS
18. BERT
19. Data mesh
20. Ethical bias testing
21. Federated learning
22. JAMstack
23. Privacy-preserving record linkage (PPRL) using Bloom filter
24. Semi-supervised learning loops

HOLD
25. 10x engineers
26. Front-end integration via artifact
27. Lambda pinball
28. Legacy migration feature parity



TECHNIQUES

Automated machine learning (AutoML) tools have emerged to fill the gap between the supply and demand for data scientists that specialize in ML. These tools are a useful starting point — but produce best results when used by experts.

(AutoML)

Binary attestation is a technique to implement deploy-time security control, to cryptographically verify that a binary image is authorized for deployment.

(Binary attestation)

Pipelines for infrastructure as code

ADOPT

The use of continuous delivery pipelines to orchestrate the release process for software has become a mainstream concept. CI/CD tools can be used to test server configuration (e.g., Chef cookbooks, Puppet modules, Ansible playbooks), server image building (e.g., Packer), environment provisioning (e.g., Terraform, CloudFormation) and the integration of environments. The use of pipelines for infrastructure as code lets you find errors before changes are applied to operational environments — including environments used for development and testing. They also offer a way to ensure that infrastructure tooling is run consistently, using CI/CD agents rather than individual workstations. Our teams have had good results adopting this technique on their projects.

Run cost as architecture fitness function

ADOPT

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

Testing using real device

ADOPT

When adopting continuous delivery (CD) successfully, teams strive to make the various test environments look as close to production as possible. This allows them to avoid bugs that would otherwise only show themselves in the production environment. This remains just as valid for embedded and Internet of Things software; if we don't run our tests in realistic environments we can expect to find some bugs for the first time in production. Testing using real devices helps avoid this issue by making sure the right devices are available in the CD pipeline.

Automated machine learning (AutoML)

TRIAL

The power and promise of machine learning has created a demand for expertise that outstrips the supply of data scientists who specialize in this area. In response to this skills gap, we've seen the emergence of Automated machine learning (AutoML) tools that purport to make it easy for nonexperts to automate the end-to-end process of model selection and training. Examples include Google's AutoML, DataRobot and the H2O AutoML interface. Although we've seen promising results from these tools, we'd caution businesses against viewing them as the sum total of their machine-learning journey. As stated on the H2O website, "there is still a fair bit of knowledge and background in data science that is required to produce high-performing machine learning models." Blind trust in automated techniques also increases the risk of introducing ethical bias or making decisions that disadvantage minorities. While businesses may use these tools as a starting point to generate useful, trained models, we encourage them to seek out experienced data scientists to validate and refine the results.

Binary attestation

TRIAL

As the usage of containers, deployment of large fleet of services by autonomous teams and increased speed of continuous delivery become common practice for many organizations, the need for automated deploy-time software security controls arise. Binary attestation is a technique to implement deploy-time security control; to cryptographically verify that a binary image is authorized for deployment. Using this technique, an attestor, an automated build process or a security team signs off the binaries that have passed the required quality checks and tests and are authorized to be deployed. Services such as GCP Binary Authorization enabled by Grafeas, and tools such as in-toto and Docker Notary support creating attestations and validating the image signatures before deployment.

Continuous delivery for machine learning (CD4ML)

TRIAL

With an increased popularity of ML-based applications, and the technical complexity involved in building them, our teams rely heavily on continuous delivery for machine learning (CD4ML) to deliver such applications safely, quickly and in a sustainable manner. CD4ML is the discipline of bringing CD

principles and practices to ML applications. It removes long cycle times between training models and deploying them to production. CD4ML removes manual handoffs between different teams, data engineers, data scientists and ML engineers in the end-to-end process of build and deployment of a model served by an application. Using CD4ML, our teams have successfully implemented the automated versioning, testing and deployment of all components of ML-based applications: data, model and code.

Data discoverability

TRIAL

One of the main points of friction for data scientists and analysts, in their workflow, is to locate the data they need, make sense of it and evaluate whether it's trustworthy to use it. This remains a challenge due to the missing metadata about the available data sources and lack of adequate functionality needed to search and locate data. We encourage teams who are providing analytical data sets or building data platforms to make data discoverability a first-class function of their environments; to provide the ability to easily locate available data, detect its quality, understand its structure and lineage and get access to it. Traditionally this function has been provided by bloated data cataloguing solutions. In recent years, we've seen the growth of open-source projects that are improving developer experiences for both data providers and data consumers to do one thing really well: to make data discoverable. Amundsen by Lyft and WhereHows by LinkedIn are among these tools. What we like to see is a change in providers' behavior to intentionally share the metadata that help discoverability in favor of discoverability tools that infer partial metadata information from silos of application databases.

Dependency drift fitness function

TRIAL

Many teams and organizations have no formal or consistent way of tracking technical dependencies in their software. This issue often shows itself when that software needs to be changed, at which point the use of an outdated version of a library, API or component will cause problems or delay. Dependency drift fitness function is a technique to introduce a specific evolutionary architecture fitness function to track these dependencies over time, thus giving an indication of the possible work needed and whether a potential issue is getting better or worse.

Design systems

TRIAL

As application development becomes increasingly dynamic and complex, it's a challenge to achieve the effective delivery of accessible and usable products that are consistent in style. Design systems define a collection of design patterns, component libraries and good design and engineering practices that ensure consistency in the development of digital products. We've found design systems a useful addition to our toolbox when working across teams and disciplines in product development, because they allow teams to focus on more strategic challenges around the product itself without the need to reinvent the wheel every time they need to add a visual component. The types of components and tools you use to create design systems can vary greatly.

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, the network topology, training data and various optimizations or tweaks to the model. Because many of these models are still difficult to interpret or explain, 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 these models have become increasingly common in business systems, several different experiment tracking tools for machine learning have emerged to help investigators keep track of these experiments and work through them methodically. Although no clear winner has emerged, tools such as MLflow or Weights & Biases and platforms such as Comet or Neptune have introduced rigor and repeatability into the entire machine learning workflow. They also facilitate collaboration and help turn data science from a solitary endeavor into a team sport.

Explainability as a first-class model selection criterion

TRIAL

Deep neural networks have demonstrated remarkable recall and accuracy across a wide range of problems. Given sufficient training data and an appropriately chosen topology, these models meet and exceed human capabilities in certain select problem spaces. However, they're inherently opaque. Although parts of models can be reused through transfer learning, we're seldom able to ascribe any human-understandable

TECHNIQUES

Deep neural networks have demonstrated remarkable recall and accuracy across a wide range of problems. But as their use increases, so too does the importance of being able to explain how decisions are reached.

(Explainability as a first-class model selection criterion)

TECHNIQUES

The complexity of the technology landscape today demands we treat security policy as code; define and keep policies under version control, automatically validate them, automatically deploy them and monitor their performance.

(Security policy as code)

Zhong Tai is an approach to delivering encapsulated business models. It's designed to help a new breed of small businesses deliver first-rate services without the costs of traditional enterprise infrastructure.

(Zhong Tai)

meaning to these elements. In contrast, an explainable model is one that allows us to say how a decision was made. For example, a decision tree yields a chain of inference that describes the classification process. Explainability becomes critical in certain regulated industries or when we're concerned about the ethical impact of a decision. As these models are incorporated more widely into critical business systems, it's important to consider explainability as first-class model selection criterion. Despite their power, neural networks might not be an appropriate choice when explainability requirements are strict.

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.

Sidecars for endpoint security

TRIAL

Many of the technical solutions we build today run in increasingly complex polycloud or hybrid-cloud environments with multiple distributed components and services. Under such circumstances, we apply two security principles early in implementation: *zero trust network*, never trust the network and always verify; and the principle of *least privilege*, granting the minimum permissions necessary for performing a particular job. Sidecars for endpoint security is a common technique we use to implement these principles to enforce security controls at every component's endpoint, e.g., APIs of services, data stores or Kubernetes control interface. We do this using an out-of-process sidecar — a process or a container that is deployed and scheduled with each service sharing the same execution context, host and identity. Open Policy Agent and Envoy are tools that implement this technique. Sidecars for endpoint security minimize the trusted footprint to a local endpoint rather than the network perimeter. We like to see the responsibility of sidecar's security policy configuration left with the team that is responsible for the endpoint and not a separate centralized team.

Zhong Tai

TRIAL

Zhong Tai has been a buzzword in the Chinese IT industry for years, but it has yet to catch on in the West. At its core, Zhong Tai is an approach to delivering encapsulated business models. It's designed to help a new breed of small businesses deliver first-rate services without the costs of traditional enterprise infrastructure and enabling existing organizations to bring innovative services to market at breakneck speeds. The Zhong Tai strategy was originally proposed by Alibaba and soon followed by many Chinese Internet companies, because their business model is digital native, making it suitable to replicate for new markets and sectors. Nowadays, more Chinese firms are using Zhong Tai as a lever for digital transformation.

BERT

ASSESS

BERT stands for Bidirectional Encoder Representations from Transformers; it's a new method of pretraining language representations which was published by researchers at Google in October 2018. BERT has significantly altered the natural language processing (NLP) landscape by obtaining state-of-the-art results on a wide array of NLP tasks. Based on Transformer architecture, it learns from both the left and right side of a token's context during training. Google has also released pretrained general-purpose BERT models that have been trained on a large corpus of unlabelled text including Wikipedia. Developers can use and fine-tune these pre-trained models on their task-specific data and achieve great results. We talked about transfer learning for NLP in our April 2019 edition of the Radar; BERT and its successors continue to make transfer learning for NLP a very exciting field with significant reduction in effort for users dealing with text classification.

Data mesh

ASSESS

Data mesh is an architectural paradigm that unlocks analytical data at scale; rapidly unlocking access to an ever-growing number of distributed domain data sets, for a proliferation of consumption scenarios such as machine learning, analytics or data intensive applications across the organization. Data mesh addresses the common failure modes of the traditional centralized data lake or data platform architecture, with a shift from the centralized paradigm of a lake, or its predecessor, the data warehouse. Data mesh shifts to a paradigm that draws from modern

distributed architecture: considering domains as the first-class concern, applying platform thinking to create a self-serve data infrastructure, treating data as a product and implementing open standardization to enable an ecosystem of interoperable distributed data products.

Ethical bias testing

ASSESS

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 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 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. However, this is a developing field and we expect standards and practices specific to ethical bias testing to emerge over time.

Federated learning

ASSESS

Model training generally requires collecting data from its source and transporting it to a centralized location where the model training algorithm runs. This becomes particularly problematic when the training data consists of personally identifiable information. We're encouraged by the emergence of federated learning as a privacy-preserving method for training on a large diverse set of data relating to individuals. Federated learning techniques allow the data to remain on the users' device, under their control, yet contribute to an aggregate corpus of training data. In one such technique, each user device updates a model independently; then the model parameters, rather than the data itself, are combined into a centralized view. Network bandwidth and device computational limitations present some significant technical challenges, but we like the way federated learning leaves users in control of their own personal information.

JAMstack

ASSESS

The trend that started as backend as a service for native mobile apps many years ago is now becoming popular with web applications. We're seeing frameworks such as Gatsby.js that combine static site generation and client-side rendering with third-party APIs. Referred to as JAMstack (the JAM stands for JavaScript, API, and Markup), this approach can provide rich user experiences to web applications that rely mostly on APIs and SaaS offerings. Because the HTML is rendered either in the web browser or at build time, the deployment model is the same as fully statically generated sites, with all its benefits: the attack surface on the server is small and great performance can be achieved with low resource usage. Such deployments are also ideal for a content delivery network. In fact, we toyed with the idea of labelling this technique as *CDN first* applications.

Privacy preserving record linkage (PPRL) using Bloom filter

ASSESS

Linking records from different data providers in the presence of a shared key is trivial. However, you may not always have a shared key; even if you do, it may not be a good idea to expose it due to privacy concerns. Privacy-preserving record linkage (PPRL) using Bloom filter (a space-efficient probabilistic data structure) is an established technique that allows probabilistic linkage of records from different data providers without exposing privately identifiable personal data. For example, when linking data from two data providers, each provider encrypts its personally identifiable data using Bloom filter to get cryptographic linkage keys and then sends them to you via a secure channel. Once data is received, the records can be linked by computing similarity scores between sets of cryptographic linkage keys from each provider. Among other techniques, we found PPRL using Bloom filters to be scalable for large data sets.

Semi-supervised learning loops

ASSESS

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

TECHNIQUES

We're encouraged by the emergence of federated learning as a privacy-preserving method for training on a large diverse set of data relating to individuals.

(Federated learning)

JAMstack can provide rich user experiences to web applications that rely mostly on APIs and SaaS offerings.

(JAMstack)

TECHNIQUES

In our experience, great engineers are driven not by their own individual output, but by working in (and working to create) amazing teams.

(10x engineers)

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.

10x engineers

HOLD

The old term 10x engineer has come under scrutiny these past few months. A widely shared Twitter thread essentially suggests companies should excuse antisocial and damaging behaviors in order to retain engineers who are perceived as having immense individual output. Thankfully, many people on social media made fun of the concept, but the stereotype of the "rockstar developer" is still pervasive. In our experience, great engineers are driven not by individual output but by working in amazing teams. It's more effective to build teams of talented individuals with mixed experiences and diverse backgrounds and provide the right ingredients for teamwork, learning and continuous improvement. These 10x teams can move faster, scale more quickly and are much more resilient — without needing to pander to bad behaviors.

Front-end integration via artifact

HOLD

When teams embrace the concept of micro frontends they have a number of patterns at their disposal to integrate the individual micro frontends into one application. As always there are antipatterns, too. A common one in this case is front-end integration via artifact. For each micro frontend an artifact is built, usually an NPM package, which is pushed into a registry. A later step, sometimes in a different build pipeline, then combines the individual packages into a final package that contains all micro frontends. From a purely technical perspective this integration at build time results in a working application. However, integrating via artifact implies that for each change the full artifact needs to be rebuilt, which is time consuming and will likely have a negative impact on developer experience. Worse, this style of integrating frontends also introduces direct dependencies between the micro frontends at build time and therefore causes considerable coordination overhead.

Lambda pinball

HOLD

We've been building serverless architectures on our projects for a couple of years now, and we've noticed that it's quite easy to fall into the trap of building a distributed monolith. Lambda pinball architectures characteristically lose sight of important domain logic in the tangled web of lambdas, buckets and queues as requests bounce around increasingly complex graphs of cloud services. Typically they're hard to test as units, and the application needs must be tested as an integrated whole. One pattern we can use to avoid these pinball architectures is to draw a distinction between public and published interfaces and apply good old domain boundaries with published interfaces between them.

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.

PLATFORMS

Apache Flink
TRIAL

Apache Flink has seen increasing adoption since our initial assessment in 2016. Flink is recognized as the leading stream-processing engine and also gradually matured in the fields of batch processing and machine learning. One of Flink's key differentiator from other stream-processing engines is its use of consistent checkpoints of an application's state. In the event of failure, the application is restarted and its state is loaded from the latest checkpoint — so that the application can continue processing as if the failure had never happened. This helps us to reduce the complexity of building and operating external systems for fault tolerance. We see more and more companies using Flink to build their data-processing platform.

Apollo Auto
TRIAL

Once exclusive to tech giants, self-driving technology isn't rocket science anymore, as demonstrated by Apollo Auto. The goal of the Baidu-owned Apollo program is to become the Android of the autonomous driving industry. The Apollo platform has components such as perception, simulation, planning and intelligent control that enable car companies to integrate their own autonomous driving systems into their vehicles' hardware. The developer community is still new but with a lot of vendors joining to contribute more ports. One of our projects helped our client to complete self-driving license exams with the Apollo-based autopilot system. Apollo also provides an evolutionary architecture approach to adopt advanced features gradually, which enables us to integrate more sensors and functions in an agile, iterative way.

GCP Pub/Sub
TRIAL

GCP Pub/Sub is Google Cloud's event streaming platform. It's a popular piece of infrastructure for many of our architectures running Google Cloud Platform, including mass event ingestion, communication of serverless workloads and streaming data-processing workflows. One of its unique features is support of pull and push subscriptions: subscribing to receive all published messages available at the time of subscription or pushing messages to a particular endpoint. Our teams have enjoyed its reliability and scale and that it just works as advertised.

Mongoose OS
TRIAL

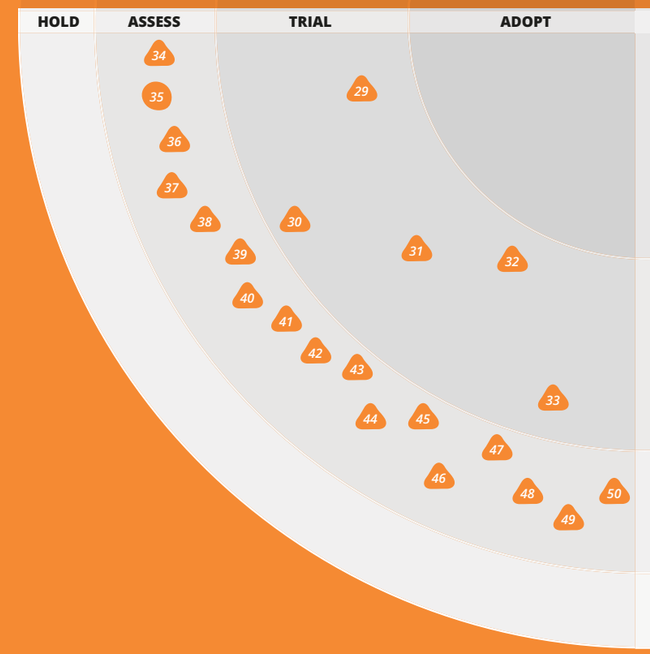
Mongoose OS remains one of our preferred open-source microcontroller operating systems and embedded firmware development frameworks. It's worth noting that Mongoose OS fills a noticeable gap for embedded software developers: the gap between Arduino firmware suitable for prototyping and bare-metal microcontrollers' native SDKs. Our teams have successfully used Cesanta's new end-to-end device management platform,

ADOPT

TRIAL
29. Apache Flink
30. Apollo Auto
31. GCP Pub/Sub
32. Mongoose OS
33. ROS

ASSESS
34. AWS Cloud Development Kit
35. Azure DevOps
36. Azure Pipelines
37. Crowdin
38. Crux
39. Delta Lake
40. Fission
41. FoundationDB
42. GraalVM
43. Hydra
44. Kuma
45. MicroK8s
46. Oculus Quest
47. ONNX
48. Rootless containers
49. Snowflake
50. Teleport

HOLD



PLATFORMS

Apache Flink is the leading stream-processing engine and is also maturing in the fields of batch processing and machine learning.

(Apache Flink)

The goal of the Baidu-owned Apollo program is to become the Android of the autonomous driving industry.

(Apollo Auto)

mDash, for small-scale greenfield hardware projects. Major Internet of Things (IoT) cloud platform providers today support the Mongoose OS development framework for their device management, connectivity, and over-the-air (OTA) firmware upgrades. Since we last reported on Mongoose OS, the number of supported boards and microcontrollers has grown to include STM, Texas Instruments and Espressif. We continue to enjoy its seamless support for OTA updates and its built-in security at the individual device level.

ROS

TRIAL

ROS (Robot Operating System) is a set of libraries and tools to help software developers create robot applications. It's a development framework that provides hardware abstraction, device drivers, libraries, visualizers, message-passing, package management and more. Apollo Auto is based on ROS. In our other ADAS simulation project, we've also used ROS's messaging system (bag). The technology isn't new, but it has regained developers' attention with the development of ADAS.

AWS Cloud Development Kit

ASSESS

For many of our teams Terraform has become the default choice for defining cloud infrastructure. However, some of our teams have been experimenting with AWS Cloud Development Kit (AWS CDK) and they like what they've seen so far. In particular, they like the use of first-class programming languages instead of configuration files which allows them to use existing tools, test approaches and skills. Like similar tools, care is still needed to ensure deployments remain easy to understand and maintain. Given that support for C# and Java is coming soon and ignoring for now some gaps in functionality, we think AWS CDK is worth watching as an alternative to other configuration file–based approaches.

Azure DevOps

ASSESS

Azure DevOps services include a set of managed services such as hosted Git repos, CI/CD pipelines, automated testing tooling, backlog management tooling and artifact repository. Azure DevOps Pipelines have been maturing over time. We particularly like its ability to define Pipelines as code and its ecosystem of extensions on the Azure DevOps marketplace. At the time of writing, our teams are still running into a few immature features, including lack of an effective UI for pipeline visualization and navigation and the inability to trigger a pipeline from artifacts or other pipelines.

Azure Pipelines

ASSESS

Azure Pipelines is a product of Azure DevOps that offers cloud-based solutions to implement pipelines as code for projects hosted in Azure DevOps Git server or other Git solution such as GitHub or Bitbucket. The interesting part of this solution is the ability to run your scripts in Linux, MacOS and Windows agents without the overhead of managing a virtual machine on your own. This represents a big step forward, especially for teams that work on Windows environments with .NET Framework solutions; we're also assessing this service for continuous delivery in iOS.

Crowdin

ASSESS

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 and the platform streamlines the text that needs translation into an online workflow. We like that Crowdin nudges the teams to continuously and incrementally incorporate translation rather than managing them in large batches toward the end.

Crux

ASSESS

Crux is an open-source document database with bitemporal graph queries. Most database systems are temporal, meaning they help us model facts along with the time at which they occurred. Bitemporal database systems let you model not just the *valid* time the fact occurred but also the *transaction* time when it was received. If you need a document store with graph capabilities for querying the content, then give Crux a try. It's currently in alpha and lacks SQL support, but you can use a Datalog query interface for reading and traversing relationships.

Delta Lake
ASSESS

Delta Lake is an open-source storage layer by Databricks that attempts to bring transactions to big data processing. One of the problems we often encounter when using Apache Spark is the lack of ACID transactions. Delta Lake integrates with the Spark API and addresses this problem by its use of a transaction log and versioned Parquet files. With its serializable isolation, it allows concurrent readers and writers to operate on Parquet files. Other welcome features include schema enforcement on write and versioning, which allows us to query and revert to older versions of data if necessary. We've started to use it in some of our projects and quite like it.

Fission
ASSESS

Kubernetes's serverless ecosystem is growing. We talked about Knative in a previous Radar; now we're seeing Fission gaining traction. Fission lets developers focus on writing short-lived functions and map them to HTTP requests while the framework handles the rest of the plumbing and automation of Kubernetes resources behind the scenes. Fission also lets you compose functions, integrate with third-party providers via web hooks and automate the management of the Kubernetes infrastructure.

FoundationDB
ASSESS

FoundationDB is an open-source multimodel database, acquired by Apple in 2015 and then open-sourced in April 2018. The core of FoundationDB is a distributed key-value store, which provides strict serializability transactions. One of the interesting aspects of FoundationDB is its concept of layers to offer additional models. These layers are essentially stateless components built on top of the core key-value store, such as the Record layer and the Document layer. FoundationDB sets a high standard with its Simulation testing where they run daily tests simulating various system failures. With its performance, rigorous testing and easy operability, FoundationDB is not just a database but can also be used by those looking to build distributed systems where they can use FoundationDB as a core primitive on which to build their system.

GraalVM
ASSESS

GraalVM is a universal virtual machine by Oracle for running applications written in JVM languages, JavaScript, Python, Ruby and R, as well as C/C++ and other LLVM-based languages. At its simplest, GraalVM can be used as a more performant VM for JVM and other supported non-JVM languages. But it also allows us to write polyglot applications with very little performance impact; and its Native Image utility (currently only available as an Early Adopter Technology) lets us compile Java code ahead of time to stand-alone executables for faster startup and less memory use. GraalVM has generated a lot of excitement in the Java community, and a host of Java frameworks (including Micronaut, Quarkus, and Helidon) are already taking advantage of it.

Hydra
ASSESS

Not everyone needs a self-hosted OAuth2 solution, but if you do, we found Hydra — a fully compliant open-source OAuth2 server and OpenID connect provider — quite useful. We really like that Hydra doesn't provide any identity management solutions out of the box; so no matter what flavor of identity management you have, it's possible to integrate it 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.

Kuma
ASSESS

Kuma is a platform-agnostic service mesh for Kubernetes, VMs and bare metal environments. Kuma is implemented as a control plane on top of Envoy and as such can instrument any Layer 4/Layer 7 traffic to secure, observe, route and enhance connectivity between services. Most of the service mesh implementations are targeted natively at the Kubernetes ecosystem which in itself is not bad but hinders the adoption of service mesh for existing non-Kubernetes applications. Rather than waiting for large platform transformation efforts to be complete, you can now use Kuma and modernize the network infrastructure.

MicroK8s
ASSESS

We talked about Kubernetes in the past and it continues to be the default choice for deploying and managing containers in production clusters. However, it's getting increasingly difficult to provide a similar experience offline for developers. Among other options, we've found MicroK8s to be quite useful. To install the MicroK8s snap, pick a release channel (stable, candidate, beta or edge), and you can get Kubernetes

PLATFORMS

GraalVM has generated a lot of excitement in the Java community, and a host of Java frameworks (including Micronaut, Quarkus, and Helidon) are already taking advantage of it.

(GraalVM)

Kuma is a platform-agnostic service mesh for Kubernetes, VMs and bare metal environments.

(Kuma)

PLATFORMS

The interoperability between tools and frameworks in the neural networks ecosystem has been a challenge. ONNX can help.

(ONNX)

Teleport is a security gateway for remotely accessing cloud native infrastructures.

(Teleport)

running with a few commands. You can also keep track of mainstream releases and choose to upgrade your setup automatically.

Oculus Quest

ASSESS

We've long tracked AR/VR (Augmented/Virtual Reality) in our Radar, but its appeal has been limited to specific platforms and tethering options. The Oculus Quest changes the game, becoming one of the first consumer mass-market standalone VR headsets that requires no tethering or support outside a smartphone. This device opens the door for a huge jump in potential exposure to VR applications, whose demand will in turn drive the market toward more aggressive innovation. We applaud the democratization of VR this device helps usher in and can't wait to see what's on the horizon.

ONNX

ASSESS

The tools and frameworks ecosystem around neural networks have been evolving rapidly. The interoperability between them, however, has been a challenge. It's not uncommon in the ML industry to quickly prototype and train the model in one tool and then deploy it in a different tool for inference. Because the internal format of these tools aren't compatible, we need to implement and maintain messy convertors to make the models compatible. The Open Neural Network Exchange format ONNX addresses this problem. In ONNX, the neural networks are represented as graphs using standard operator specifications, and together with a serialization format for trained weights, neural network models can be transferred from one tool to another. This opens up lots of possibilities, including Model Zoo, a collection of pretrained models in ONNX format.

Rootless containers

ASSESS

Ideally, containers should be managed and run by the respective container runtime without root privileges. This is not trivial but when achieved, it reduces the attack surface and avoids whole classes of security problems, notably privilege escalation out of the container. The community has discussed this as rootless containers for quite a while, and it is part of the open container runtime specification and its standard implementation runc, which underpins Kubernetes. Now, Docker 19.03 introduces rootless containers as an experimental feature. Although fully functional, the feature doesn't yet work with several other features such as cgroups resource controls and AppArmor security profiles.

Snowflake

ASSESS

We often relate data warehousing to a central infrastructure that is hard to scale and manage with the growing demands around data. Snowflake, however, is a new SQL Data Warehouse as a Service solution built from the ground up for the cloud. With a bunch of neatly crafted features such as database-level atomicity, structured and semi-structured data support, in-database analytics functions and above all with a clear separation of storage, compute and services layer, Snowflake addresses most of the challenges faced in data warehousing.

Teleport

ASSESS

Teleport is a security gateway for remotely accessing cloud native infrastructures. One of Teleport's interesting features is its ability to double as a Certificate Authority (CA) for your infrastructure. You can issue short-lived certificates and build richer role-based access control (RBAC) for your Kubernetes infrastructure (or for just SSH). With increased focus on infrastructure security it's important to keep track of changes. However, not all events require the same level of auditing. With Teleport you can stick with logging for most of the events but go the extra mile by recording the user screen for more privileged root sessions.

TOOLS

Commitizen
ADOPT

Commitizen is a simple tool to help streamline the commit process when using Git. It prompts you to provide any required fields and also formats your commit message appropriately. It supports different conventions for describing the required check-in formats, and you can add your own via an adapter. This simple tool saves time and avoids later rejections from a commit hook.

ESLint
ADOPT

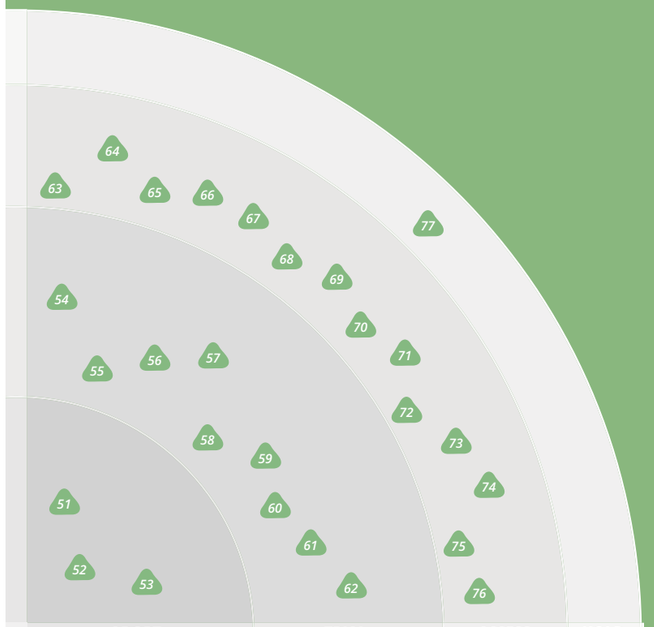
ESLint is being used as a standard in many of our projects. As a linting tool for JavaScript it has multiple rule sets, recommended rules and plugins in order to extend to frameworks or JavaScript flavors. We've seen it leveraged heavily to help teams create and enforce norms in their code by allowing for real-time analysis of code during development. It can be used to standardize coding practices by enforcing best practices and code styling, and identify vulnerabilities in your code. It does so by integrating well with most IDEs and giving live feedback while coding. It's styling rules in particular will automatically fix the linting errors, making the process seamless and effective without incurring additional development cost. Developers can quickly get up to speed with the rules thanks to the community documentation, which does a good job of explaining coding patterns. As ESLint becomes more common and powerful, it has gained traction in the industry, and this is illustrated by the TypeScript team's move to support and work with ESLint rather than investing in TSLint.

React Styleguidist
ADOPT

React Styleguidist is a development environment for React components. It includes a dev server with hot reloading capabilities and generates an HTML style guide for sharing with teams. The style guide shows a live version of all components in one place with documentation and a list of their props. We've mentioned React Styleguidist as a UI dev environment before, and over time it has become our default choice among similar tools in this space.

Bitrise
TRIAL

Building, testing and deploying mobile applications entails complex steps, especially when we consider a pipeline from source code repository to app stores. All of these steps can be automated with scripts and build pipelines in generic CI/CD tools. However, our teams have found Bitrise, a domain-specific CD tool for mobile applications, useful for mobile applications when there was no need to integrate with build pipelines for back-end systems. Bitrise is easy to set up and provides a comprehensive set of prebuilt steps for most mobile development needs.



ADOPT

51. Commitizen
52. ESLint
53. React Styleguidist

TRIAL

54. Bitrise
55. Dependabot
56. Detekt
57. Figma
58. Jib
59. Loki
60. Trivy
61. Twistlock
62. Yocto Project

ASSESS

63. Aplas
64. asdf-vm
65. AWSume
66. dbt
67. Docker Notary
68. Facets
69. Falco
70. in-toto
71. Kubeflow
72. MemGuard
73. Open Policy Agent (OPA)
74. Pumba
75. Skaffold
76. What-If tool

HOLD

77. Azure Data Factory for orchestration

TOOLS

Figma has the same functionality as other design programs such as Sketch and Invision, but it enables you to collaborate with another person.

(Figma)

If you're building a Java application and using Docker, you might consider using Google's Jib.

(Jib)

Dependabot
TRIAL

Keeping dependencies up to date is a chore, but for security reasons it's important to respond to updates in a timely manner. You can use tools to make this process as painless and automated as possible. In practical use our teams have had good experiences with Dependabot. It integrates with GitHub repositories and automatically checks dependencies for new versions. When required, Dependabot will open a pull request with upgraded dependencies.

Detekt
TRIAL

Detekt is a static code analysis tool for Kotlin. It provides code smell analysis and complexity reports based on highly configurable rule sets. It can be run from the command line and, using plugins, via Gradle, SonarQube and IntelliJ. Our teams have found great value in using Detekt to maintain high code quality. When analysis and report generation are integrated into a build pipeline, it's obviously important that the reports are checked on a regular basis and the team sets aside time to act on the findings.

Figma
TRIAL

One of the great pain points in interaction and visual design is the lack of tools built for collaboration. This is where Figma comes in. It has the same functionalities of design programs such as Sketch and Invision, but by being able to collaborate with another person at the same time, it helps you discover new ideas together with real-time collaboration capabilities. Our teams find Figma very useful, especially in remote and distributed design work enablement and facilitation. In addition to its collaboration capabilities, Figma also offers an API that helps to improve the DesignOps process.

Jib
TRIAL

Building containerized applications can require complex configurations in development environments and on build agents. If you're building a Java application and use Docker, you might consider using Google's Jib. Jib is an open-source plugin supporting both Maven and Gradle. The Jib plugin uses information from your build config to build your application directly as a Docker image without requiring a Dockerfile or Docker daemon. Jib optimizes around image layering, promising to speed up subsequent builds.

Loki
TRIAL

Loki is a visual regression tool that works with Storybook, which we mentioned previously in the context of UI dev environments. With a few lines of configuration, Loki can be used to test all UI components. The preferred mode of operation is using Chrome in a Docker container as this avoids one-pixel differences when tests are run in nonidentical environments. Our experience has been that the tests are very stable, but updates to Storybook tend to cause tests to fail with minor differences. It also seems impossible to test components which use `position: fixed` but you can work around that by wrapping the component with a `fixed`.

Trivy
TRIAL

Build pipelines that create and deploy containers should include container security scanning. Our teams particularly like Trivy, a vulnerability scanner for containers, because it's easier to set up than other tools, thanks to it shipping as a stand-alone binary. Other benefits of Trivy are that it's open-source software and that it supports distroless containers.

Twistlock
TRIAL

Twistlock is a commercial product with build-time and run-time security vulnerability detection and prevention capabilities. These capabilities span protecting VMs, container schedulers and containers to various registries and repositories that applications rely on. Twistlock has helped our teams accelerate development of regulated applications, where application infrastructure and architecture require compliance with, for example, Payment Card Industry (PCI) standards and the Health Insurance Portability and Accountability Act (HIPAA). Our teams have enjoyed the developer experience that Twistlock provides: the ability to run provisioning as code, the easy integration with other common observability platforms, and the out-of-the-box benchmarks to measure the infrastructure against industry-consensus best practices. We run Twistlock with regular runtime scans over our cloud-native applications, particularly when regulatory compliance is required.

Yocto Project

TRIAL

Increasingly we're seeing powerful Internet of Things devices that run Linux rather than a special embedded OS. In order to reduce resource usage and decrease the attack surface, it makes sense to build a custom Linux distribution that only contains the tools and dependencies needed to run the software on the device. In this context the Yocto Project has renewed relevance as a tool to create a Linux distribution tailored to the needs of a specific case. The learning curve is steep and due to its flexibility, it can be easy to do the wrong thing. However, over the many years of its existence, the Yocto Project has attracted an active community that can help. Compared to similar tools, it's easier to integrate into a CD workflow and, unlike Android Things or Ubuntu core for example, it's not tied to a specific ecosystem.

Aplas

ASSESS

It's often very difficult to get a handle on our software estates as they grow ever more complex. Aplas is a new software mapping tool that can be used to create visualizations of our software landscapes in the form of maps. The tool works by ingesting metadata about your existing systems and then displaying a map over which various views can be projected. Ingestion is either a manual process or one that can be automated via APIs. We're pretty excited to see this product evolve and to see what's possible with the automated collection of metadata. It should be possible, for example, to expose architectural fitness functions such as run cost to create visualizations of how much is being spent on cloud infrastructure. Understanding which systems talk to other systems via which technology is another problem we often face and Aplas can visualize it for us.

asdf-vm

ASSESS

asdf-vm is a command-line tool to manage runtime versions of multiple languages, per project. It's similar to other command-line version management tools, such as RVM for Ruby and nvm for Node.js, with the advantage of an extensible plugin architecture to handle multiple languages. Its list of current plugins include many languages as well as tools such as Bazel or tflint, whose runtime version you may need to manage per project.

AWSume

ASSESS

AWSume is a convenient script to manage AWS session tokens and assume role credentials from the command line. We find AWSume quite handy when we deal with multiple AWS accounts at the same time. Instead of specifying profiles individually in every command, the script reads from the CLI cache and exports them to environment variables. As a result, both the commands and AWS SDKs pick up the right credentials.

dbt

ASSESS

Data transformation is an essential part of data-processing workflows: filtering, grouping or joining multiple sources into a format that is suitable for analyzing data or feeding machine-learning models. dbt is an open-source tool and a commercial SaaS product that provides simple and effective transformation capabilities for data analysts. The current frameworks and tooling for data transformation fall either into the group of *powerful and flexible* — requiring intimate understanding of the programming model and languages of the framework such as Apache Spark — or in the group of dumb drag-and-drop UI tools that don't lend themselves to reliable engineering practices such as automated testing and deployment. dbt fills a niche: it uses SQL — an interface widely understood — to model simple batch transformations, while it provides command-line tooling that encourages good engineering practices such as versioning, automated testing and deployment; essentially it implements SQL-based transformation modeling as code. dbt currently supports multiple data sources, including Snowflake and Postgres, and provides various execution options, such as Airflow and Apache's own cloud offering. Its transformation capability is limited to what SQL offers, and it does'nt support real-time streaming transformations at the time of writing.

Docker Notary

ASSESS

Docker Notary is an OSS tool that enables signing of assets such as images, files and containers. This means that the provenance of assets can be asserted which is superuseful in regulated environments and better practice everywhere. As an example, when a container is created, it's signed by a private key and a hash, tied to the publisher's identity, stored as metadata. Once published, the provenance of the container (or other asset) can be checked

TOOLS

The Yocto Project has renewed relevance as a tool to create a Linux distribution tailored to the needs of a specific case, such as Internet of Things devices.

(Yocto Project)

Aplas is a new software mapping tool that can be used to create visualizations of our software landscapes in the form of maps.

(Aplas)

TOOLS

With increased adoption of Kubernetes as container orchestrator, the security toolset around containers and Kubernetes is evolving rapidly. Falco is one such container-native tool aimed at addressing runtime security.

(Falco)

using the image hash and the publisher's public key. There are publicly available, trusted registries such as the Docker Trusted Registry, but it's also possible to run your own. Our teams have reported some spiky edges running local Notary servers and suggest using a registry that includes Notary where possible.

Facets

ASSESS

Given the growing amount of weighty decisions that are derived from large data sets, either directly or as training input for machine learning models, it's important to understand the gaps, flaws and potential biases in your data. Google's Facets project provides two helpful tools in this space: Facets Overview and Facets Dive. Facets Overview visualizes the distribution of values for features in a data set, can show training and validation set skew and can be used to compare multiple data sets; Facets Dive is for drilling down and visualizing individual data points in large data sets, using different visual dimensions to explore the relationships between attributes. They're both useful tools in carrying out ethical bias testing.

Falco

ASSESS

With increased adoption of Kubernetes as container orchestrator, the security toolset around containers and Kubernetes is evolving rapidly. Falco is one such container-native tool aimed at addressing runtime security. Falco leverages Sysdig's Linux kernel instrumentation and system call profiling and lets us gain deep insights into system behavior and helps us detect abnormal activities in applications, containers, underlying host or Kubernetes orchestrator itself. We like Falco's capability to detect threats without injecting third-party code or sidecar containers.

in-toto

ASSESS

We're seeing increased use of Binary attestation for securing the software supply chain, particularly within regulated industries. The currently favored approaches seem to involve either building a custom system for implementing the binary verification or relying on a cloud vendor's service. We're encouraged to see the open-source in-toto enter this space. in-toto is a framework for cryptographically verifying every component and step along the path to production for a software artifact. The project includes a number of integrations into many widely used build, container auditing and deployment tools. A software supply chain tool can be a critical piece of an organization's security apparatus, so we like that as an open-source project, in-toto's behavior is transparent, and its own integrity and supply chain can be verified by the community. We'll have to wait and see if it'll gain a critical mass of users and contributors to compete in this space.

Kubeflow

ASSESS

Kubeflow is interesting for two reasons. First, it is an innovative use of Kubernetes Operators which we've spotlighted in our April 2019 edition of the Radar. Second, it provides a way to encode and version machine-learning workflows so that they can be more easily ported from one execution environment to another. Kubeflow consists of several components, including Jupyter notebooks, data pipelines, and control tools. Several of these components are packaged as Kubernetes operators to draw on Kubernetes's ability to react to events generated by pods implementing various stages of the workflow. By packaging the individual programs and data as containers, entire workflows can be ported from one environment to another. This can be useful when moving a useful but computationally challenging workflow developed in the cloud to a custom supercomputer or tensor processing unit cluster.

MemGuard

ASSESS

If your application handles sensitive information (such as cryptographic keys) as plain text in memory, there's a high probability that someone could potentially exploit it as an attack vector and compromise the information. Most of the cloud-based solutions often use hardware security modules (HSM) to avoid such attacks. However, if you're in a situation where you need to do this in a self-hosted manner without access to HSMs, then we've found MemGuard to be quite useful. MemGuard acts as a secured software enclave for storage of sensitive information in memory. Although MemGuard is not a replacement for HSMs, it does deploy a number of security tactics such as protection against cold boot attacks, avoiding interference with garbage collection and fortifying with guard pages to reduce the likelihood of sensitive data being exposed.

Open Policy Agent (OPA)

ASSESS

Defining and enforcing security policies uniformly across a diverse technology landscape is a challenge. Even for simple applications, you have to control access to their components — such as container orchestrators, services and data stores to keep the services' state — using their components' built-in security policy configuration and enforcement mechanisms.

We're excited about Open Policy Agent (OPA), an open-source technology that attempts to solve this problem. OPA lets you define fine-grained access control and flexible policies as code, using the Rego policy definition language. Rego enforces the policies in a distributed and unobtrusive manner outside of the application code. At the time of writing, OPA implements uniform and flexible policy definition and enforcement to secure access to Kubernetes APIs, microservices APIs through Envoy sidecar and Kafka. It can also be used as a sidecar to any service to verify access policies or filter response data. Styra, the company behind OPA, provides commercial solutions for centralized visibility to distributed policies. We like to see OPA mature through the CNCF incubation program and continue to build support for more challenging policy enforcement scenarios such as diverse data stores.

Pumba

ASSESS

Pumba is a chaos testing and network emulation tool for Docker. Pumba can kill, stop, remove or pause Docker containers. Pumba can also emulate networks and simulate different network failures such as delays, packet loss and bandwidth rate limits. Pumba uses the tc tool for network emulation which means it needs to be available in our containers or we need to run Pumba in a sidecar container with tc. Pumba is particularly useful when we want to run some automated chaos tests against a distributed system running on a bunch of containers locally or in the build pipeline.

Skaffold

ASSESS

Google brings us Skaffold, an open-source tool to automate local development workflows, including deployment on Kubernetes. Skaffold detects changes in source code and triggers workflows to build, tag and deploy into a K8s cluster including capturing application logs back to the command line. The workflows are pluggable with different build and deployment tools, but this comes with an opinionated default configuration to make it easier to get started.

What-If Tool

ASSESS

The machine learning world has shifted emphasis slightly from exploring what models are capable of understanding to how they do it. Concerns about introducing bias or overgeneralizing a model's applicability have resulted in interesting new tools such as What-If Tool (WIT). This tool helps data scientists to dig into a model's behavior and to visualize the impact various features and data sets have on the output. Introduced by Google and available either through Tensorboard or Jupyter notebooks, WIT simplifies the tasks of comparing models, slicing data sets, visualizing facets and editing individual data points. Although WIT makes it easier to perform these analyses, they still require a deep understanding of the mathematics and theory behind the models. It is a tool for data scientists to gain deeper insights into model behavior. Naive users shouldn't expect any tool to remove the risk or minimize the damage done by a misapplied or poorly trained algorithm.

Azure Data Factory for orchestration

HOLD

Azure Data Factory (ADF) is currently Azure's default product for orchestrating data-processing pipelines. It supports data ingestion, copying data from and to different storage types on prem or on Azure and executing transformation logic. While we've had a reasonable experience with ADF for simple migrations of data stores from on prem to cloud, we discourage the use of Azure Data Factory for orchestration of complex data-processing pipelines. Our experience has been challenging due to several factors, including limited coverage of capabilities that can be implemented through coding first, as it appears that ADF is prioritizing enabling low-code platform capabilities first; poor debuggability and error reporting; limited observability as ADF logging capabilities don't integrate with other products such as Azure Data Lake Storage or Databricks, making it difficult to get an end-to-end observability in place; and availability of data source-triggering mechanisms only to certain regions. At this time, we encourage using other open-source orchestration tools (e.g., Airflow) for complex data pipelines and limit ADF for data copying or snapshotting. We're hoping that ADF will address these concerns to support more complex data-processing workflows and prioritize access to capabilities through code first.

What-If Tool helps data scientists to dig into a model's behavior and to visualize the impact various features and data sets have on the output.

(What-If Tool)

LANGUAGES & FRAMEWORKS

Arrow
TRIAL

Arrow is a functional programming library for Kotlin, created by merging two existing popular libraries (kategory and funKTionale). While Kotlin provides building blocks for functional programming, Arrow delivers a package of ready-to-use higher-level abstractions for application developers. It provides data types, type classes, effects, optics and other functional programming patterns as well as integrations with popular libraries. Our initial positive impressions of Arrow were confirmed when using it to build applications that are now in production.

Flutter
TRIAL

Several of our teams use Flutter and really like it. It's a cross-platform framework that enables you to write native mobile apps in Dart. It benefits from Dart and can be compiled into native code and communicates with the target platform without bridge and context switching. Flutter's hot-reload feature is still impressive and provides superfast visual feedback when editing code. We're confident in recommending that you try Flutter on one of your projects.

jest-when
TRIAL

jest-when is a lightweight JavaScript library that complements Jest by matching mock function call arguments. Jest is a great tool for testing the stack; jest-when allows you to expect specific arguments for mock functions and thus lets you write more robust unit tests of modules with many dependencies.

Micronaut
TRIAL

Micronaut is a JVM framework for building services using Java, Kotlin or Groovy. It distinguishes itself through a small memory footprint and short startup time; it achieves these improvements by avoiding runtime reflection for dependency injection (DI) and proxy generation, a common shortcoming of traditional frameworks, and instead uses a DI/AOP container which performs dependency injection at compile time. This makes it attractive not just for standard server-side microservices but also in the context of, for example, the Internet of Things, Android applications and serverless functions. Micronaut uses Netty and has first-class support for reactive programming. It also includes features such as service discovery and circuit breaking that make it cloud-native friendly. Micronaut

ADOPT

TRIAL
78. Arrow
79. Flutter
80. jest-when
81. Micronaut
82. React Hooks
83. React Testing Library
84. Styled components
85. Tensorflow

ASSESS
86. Fairseq
87. Flair
88. Gatsby.js
89. GraphQL
90. KotlinTest
91. NestJS
92. Paged.js
93. Quarkus
94. SwiftUI
95. Testcontainers

HOLD
96. Enzyme

LANGUAGES & FRAMEWORKS

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

(Micronaut)

The React Testing Library has eclipsed the alternatives to become the sensible default when testing React-based frontends.

(React Testing Library)

is a very promising entrant to the full-stack framework for the JVM space, and we're seeing it in more and more projects in production, prompting us to move it to Trial.

React Hooks

TRIAL

Earlier this year, React Hooks were introduced to the popular JavaScript framework. They make it possible to use state and other React features without writing a class, offering a cleaner approach than higher-order components or render-props for use cases. Libraries such as Material UI and Apollo have already switched to using Hooks. There are some issues with testing Hooks, especially with Enzyme, which contributed to our reassessment of Enzyme as the tool of choice.

React Testing Library

TRIAL

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.

Styled components

TRIAL

Using tagged template literals styled components make it possible to put the CSS needed to style a React component directly into the JavaScript code that creates the component. This greatly reduces the pain with managing CSS and obviates the need for naming conventions or other means of avoiding naming conflicts in CSS. Developers can see the styling when looking at the component definition, and they don't have to memorize several megabytes worth of CSS. Of course, placing the CSS into the JavaScript code can make it harder to get a consistent view across the styling of different components, which is why we recommend understanding the trade-offs with this approach.

Tensorflow

TRIAL

With its 2.0 release, TensorFlow retains its prominence as the industry's leading machine learning framework. TensorFlow began as a numerical processing package that gradually expanded to include libraries supporting a variety of ML approaches and execution environments, ranging from mobile CPU to large GPU clusters. Along the way, a slew of frameworks became available to simplify the tasks of network creation and training. At the same time, other frameworks, notably PyTorch, offered an imperative programming model that made debugging and execution simpler and easier. TensorFlow 2.0 now defaults to imperative flow (eager execution) and adopts Keras as the single high-level API. While these changes modernize TensorFlow's usability and make it more competitive with PyTorch, it is a significant rewrite that often breaks backward compatibility — many tools and serving frameworks in the TensorFlow ecosystem won't immediately work with the new version. For the time being, consider whether you want to design and experiment in TensorFlow 2.0 but revert to version 1 to serve and run your models in production.

Fairseq

ASSESS

Fairseq is a sequence-to-sequence modelling toolkit by Facebook AI Research that allows researchers and developers to train custom models for translation, summarization, language modeling and other NLP tasks. For users of PyTorch, this is a good choice. It provides reference implementations of various sequence-to-sequence models; supports distributed training across multiple GPUs and machines; is very extensible; and has a bunch of pretrained models, including RoBERTa which is an optimization on top of BERT.

Flair

ASSESS

Flair is a simple Python-based framework for NLP processing. It allows users to do standard NLP tasks such as named entity recognition (NER), part-of-speech tagging (PoS), word-sense disambiguation and classification and performs well on a range

of NLP tasks. Flair presents a simple and unified interface for a variety of word and document embeddings, including BERT, Elmo and its own Flair embeddings. It also has multilingual support. The framework itself is built on top of PyTorch. We're using it in some of our projects and like its ease of use and powerful abstractions.

Gatsby.js
ASSESS

Gatsby.js is a framework to write web applications in an architectural style known as JAMstack. Part of the application is generated at build time and deployed as a static site, while the remainder of the functionality is implemented as a progressive web application (PWA) running in the browser. Such applications work without code running on the server side. Usually, though, the PWA makes calls to third-party APIs and SaaS solutions for content management, for example. In the case of Gatsby.js, all client and build time code is written using React. The framework includes some optimizations to make the web application feel fast. It provides code and data splitting out of the box to minimize load times and speeds up performance when navigating the application by prefetching resources. APIs are called via GraphQL and several plugins simplify integration with existing services.

GraphQL
ASSESS

We've seen many successful GraphQL implementations on our projects. We've seen some interesting patterns of use too, including GraphQL for server-side resource aggregation. That said, we've concerns about misuse of this framework and some of the problems that can occur. Examples include performance gotchas around N+1 queries and lots of boilerplate code needed when adding new models, leading to complexity. There are workarounds to these gotchas such as query caching. Even though it's not a silver bullet, we still think it's worth assessing as part of your architecture.

KotlinTest
ASSESS

KotlinTest is a stand-alone testing tool for the Kotlin ecosystem that our teams have come to like. It allows property-based testing, a technique we've highlighted in the Radar before. Key advantages are that it offers a variety of testing styles in order to structure the test suites and that it comes with a comprehensive set of matchers, which allow for expressive tests in an elegant internal DSL.

NestJS
ASSESS

NestJS is a server-side Node.js framework written in TypeScript. By integrating the rich ecology of the Node.js community, NestJS provides an out-of-the-box application architecture. The mental model to develop NestJS is similar to the server-side version of Angular or the TypeScript version of Spring Boot, so the learning curve for developers is low. NestJS supports protocols such as GraphQL, Websocket and ORM libraries.

Paged.js
ASSESS

When using HTML and related technologies to produce books and other print output, the question of pagination must be considered. This includes page counters, repeated elements in headers and footers, as well as mechanisms to avoid awkward page breaks. Paged.js is an open-source library that implements a series of polyfills for the Paged Media and Generated Content for Paged Media CSS modules. It is still experimental but fills an important gap in the "write once, publish everywhere" story for HTML.

Quarkus
ASSESS

Quarkus is a cloud-native, container-first framework by Red Hat for writing Java applications. It has a very fast startup time (tens of milliseconds) and has low memory utilization which makes it a good candidate for FaaS or frequent scaling up and down in a container orchestrator. Like Micronaut, Quarkus achieves this by using ahead-of-time compilation techniques to do dependency injection at compile time and avoid the runtime costs of reflection. It also works well with GraalVM's Native Image which further reduces startup time. Quarkus supports both imperative and reactive models. Along with Micronaut and Helidon, Quarkus is leading the charge on the new generation of Java frameworks which attempt to address startup performance and memory without sacrificing developer effectiveness. It's gained a lot of community attention and is worth keeping an eye on.

LANGUAGES & FRAMEWORKS

Gatsby.js is a framework to write web applications in an architectural style known as JAMstack. It provides code and data splitting out of the box to minimize load times and speeds up performance when navigating the application by prefetching resources.

(Gatsby.js)

LANGUAGES & FRAMEWORKS

Quarkus is a cloud-native, container-first framework by Red Hat for writing Java applications. It has a very fast startup time and low memory utilization.

(Quarkus)

SwiftUI
ASSESS

Apple has taken a big step forward with their new SwiftUI framework for implementing user interfaces on 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. Although SwiftUI definitely represents the future of Apple UI development, it is quite new and it will take time to smooth out the rough edges. We look forward to improved documentation and a community of developers who can establish a set of practices for testing and other engineering concerns.

Testcontainers
ASSESS

Creating reliable environments for running automated tests is a perennial problem, particularly as the number of components that modern systems depend on keeps increasing. Testcontainers is a Java library that helps mitigate this challenge by managing dockerized dependencies for your tests. This is particularly useful for spinning up repeatable database instances or similar infrastructure, but it can also be used in web browsers for UI testing. Our teams have found this library to be helpful for making integration tests more reliable with these programmable, lightweight and disposable containers.

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.

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

subscribe now

ThoughtWorks®

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

Founded 25 years ago, ThoughtWorks has grown to a company of over 7,000 people, including a products division that makes pioneering tools for software teams. ThoughtWorks has 43 offices across 14 countries: Australia, Brazil, Canada, Chile, China, Ecuador, Germany, India, Italy, Singapore, Spain, Thailand, the United Kingdom and the United States.

thoughtworks.com

ThoughtWorks®

thoughtworks.com/radar

#TWTechRadar