# Technology Radar

## Um guia com opiniões firmes sobre as fronteiras da tecnologia

/thoughtworks

Volume 26

# Sobre o Radar

Thoughtworkers são pessoas apaixonadas por tecnologia. Nós desenvolvemos, pesquisamos, testamos, contribuímos com código livre, escrevemos sobre e visamos a sua constante melhoria — para todas as pessoas. Nossa missão é liderar e promover a excelência de software e revolucionar a indústria de TI. Nós criamos e compartilhamos o Technology Radar da Thoughtworks para apoiar essa missão. O Conselho Consultivo de Tecnologia (Technology Advisory Board, ou TAB), um grupo de líderes experientes em tecnologia da Thoughtworks, é responsável por criar o Radar. O grupo se reúne regularmente para discutir a estratégia global de tecnologia da empresa e as tendências tecnológicas que impactam significativamente a nossa indústria.

O Radar captura o resultado das discussões do TAB em um formato que procura oferecer valor a uma ampla gama de indivíduos interessados, de pessoas que desenvolvem software a CTOs. O conteúdo é concebido para ser um resumo conciso.

Nós encorajamos você a explorar essas tecnologias. O Radar é gráfico por natureza, agrupando os itens em técnicas, plataformas, ferramentas, linguagens e frameworks. No caso de itens que podem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos esses itens em quatro anéis para refletir a nossa posição atual em relação a cada um.

Para mais informações sobre o Radar, veja: thoughtworks.com/radar/faq.

# Radar em um relance

A ideia por trás do Radar é rastrear coisas interessantes, que chamamos de blips. Organizamos blips no Radar usando duas categorias: quadrantes e anéis. Os quadrantes representam as diferentes naturezas dos blips. Os anéis indicam o estágio do ciclo de adoção em que consideramos que cada blip esteja.

Um blip é uma tecnologia ou técnica que desempenha um papel significativo no desenvolvimento de software. Os blips estão sempre em movimento, o que significa que suas posições no Radar estão constantemente mudando — geralmente indicando que nossa confiança tem crescido à medida que se movimentam entre os anéis.


**Adote:** Acreditamos firmemente que a indústria deva adotar esses itens. Nós os usamos quando apropriado em nossos projetos.

**Experimente:** Vale ir atrás. É importante entender como desenvolver essa capacidade. As empresas devem testar esta tecnologia em um projeto que possa lidar com o risco

**Avalie:** Vale explorar com o objetivo de compreender como isso afetará sua empresa.

**Evite:** Prossiga com cautela.

**Novo** **Mudança de anel** **Sem modificação**

Nosso Radar é um olhar para o futuro. Para abrir o espaço para novos itens, apagamos itens que não foram modificados recente, o que não é um reflexo de seu valor, mas uma solução para nossa limitação de espaço.

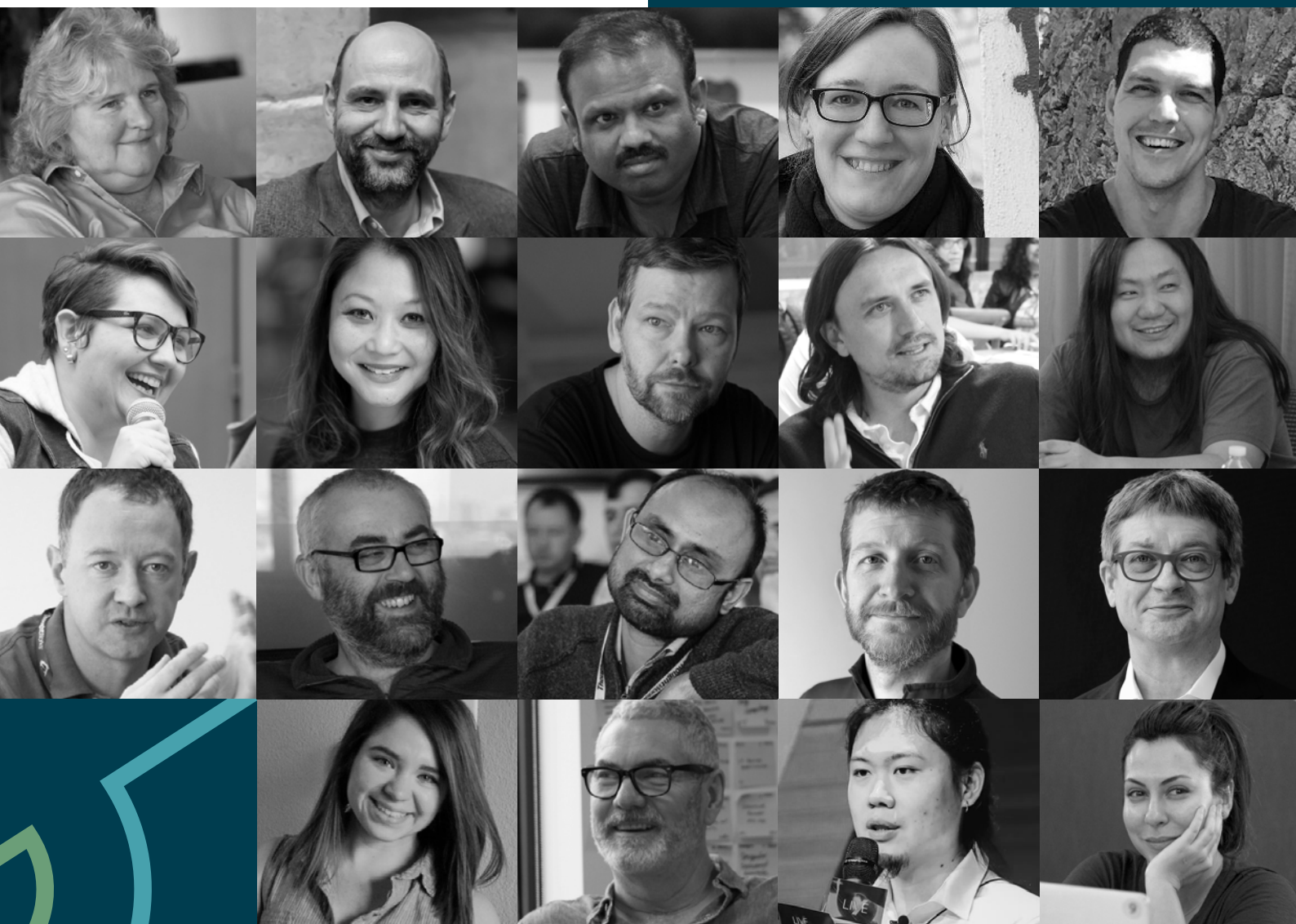
# Contribuições

O Conselho Consultivo de Tecnologia (TAB) é um grupo formado por 18 tecnologistas experientes da Thoughtworks. O TAB se reúne duas vezes por ano pessoalmente e quinzenalmente por videochamada. Sua principal atribuição é ser um grupo consultivo para a CTO da Thoughtworks, Rebecca Parsons.

O TAB atua examinando tópicos que afetam soluções de tecnologia e tecnologistas da Thoughtworks. Com o atual cenário de pandemia global, mais uma vez criamos esta edição do Technology Radar por meio de um evento virtual.

**Tradução:** Paula Ribas, Patrick Prado, Thiago Gregorio, Pietra Freitas, Rodrigo Pinheiro, Rosi Teixeira e Gregorio Melo

Rebecca Parsons (CTO)
Martin Fowler (Chief Scientist)
Bharani Subramaniam
Birgitta Böckeler
Brandon Byars
Camilla Falconi Crispim
Cassie Shum
Erik Dörnenburg
Fausto de la Torre
Hao Xu
Ian Cartwright
James Lewis
Lakshminarasimhan Sudarshan
Mike Mason
Neal Ford
Perla Villarreal
Scott Shaw
Shangqi Liu
Zhamak Dehghani



# Temas

## As transformações no mercado de software de código aberto

Na Thoughtworks, somos há muito tempo fãs do software de código aberto, em parte popularizado pelo famoso ensaio "A Catedral e o Bazar" de Eric Raymond. O software de código aberto melhora a mobilidade da pessoa desenvolvedora e colabora com correções de erros e inovações. No entanto, as tentativas de comercialização demonstram a enorme complexidade econômica do ecossistema atual. Veja, por exemplo, a AWS fazendo um fork do Elasticsearch para o OpenSearch em setembro de 2021, em resposta à mudança de licença da Elastic que passou a exigir contribuição das provedoras de serviços em nuvem que lucram com seu trabalho. Isso mostra como pode ser difícil para o software comercial de código aberto se manter competitivo (a mesma preocupação se aplica ao software livre de código fechado; observamos, por exemplo, algumas empresas explorando alternativas ao Docker Desktop devido ao esforço contínuo do Docker para encontrar o modelo comercial certo.) Às vezes, a dinâmica do poder funciona de maneira inversa: pelo fato de ter sido financiado pelo Facebook como um software aberto, a equipe responsável pelo Presto conseguiu manter a propriedade intelectual e renomeá-lo como Trino depois de deixarem a empresa, beneficiando-se efetivamente do investimento do Facebook. A situação se torna ainda mais complexa pela quantidade de infraestrutura crítica que não é patrocinada pelas empresas, que geralmente só se dão conta do quanto dependem de mão de obra não remunerada quando um erro crítico de segurança é descoberto (como aconteceu recentemente com o Log4J). Em alguns casos, o financiamento de indivíduos e times que desenvolvem como hobby fornece impulso suficiente para fazer a diferença. Em outros, isso simplesmente cria um sentimento adicional de responsabilidade para além do trabalho diário e contribui para o esgotamento. Continuamos a defender fortemente o software de código aberto, mas reconhecemos que esse mercado vem se tornando cada vez mais bizarro, e não há soluções simples para encontrar um equilíbrio ideal.

## Inovações na cadeia de fornecimento de software

Instâncias públicas de problemas graves — como o vazamento de dados da Equifax, o ataque da SolarWinds, a vulnerabilidade remota de dia zero do Log4J e assim por diante — foram causadas pela má governança da cadeia de fornecimento de software. Os times agora se dão conta de que entre as práticas responsáveis de engenharia estão a validação e a governança das dependências do projeto, que geraram várias entradas nesta edição do Radar. Entre os blips, estão listas de verificação e padrões, como **Supply chain Levels for Software Artifacts (SLSA)**, um consórcio apoiado pelo Google para fornecer orientações sobre ameaças padrão à cadeia de fornecimento e **CycloneDX**, outro conjunto de padrões impulsionado pela comunidade OWASP. Também apresentamos ferramentas concretas como **Syft**, que gera uma **lista de materiais de software (SBOM)** a partir de imagens de contêiner. Hackers estão se aproveitando cada vez mais da natureza assimétrica por trás da lógica de ataque e defesa na área de segurança – para atacar só precisam encontrar uma vulnerabilidade, enquanto para defender é necessário proteger toda a superfície de ataque –, aplicando técnicas cada vez mais sofisticadas. A segurança aprimorada da cadeia de fornecimento é uma parte crítica de nossa resposta à medida que trabalhamos para manter os sistemas seguros.

## Por que as pessoas desenvolvedoras continuam implementando gerenciamento de estado no React?

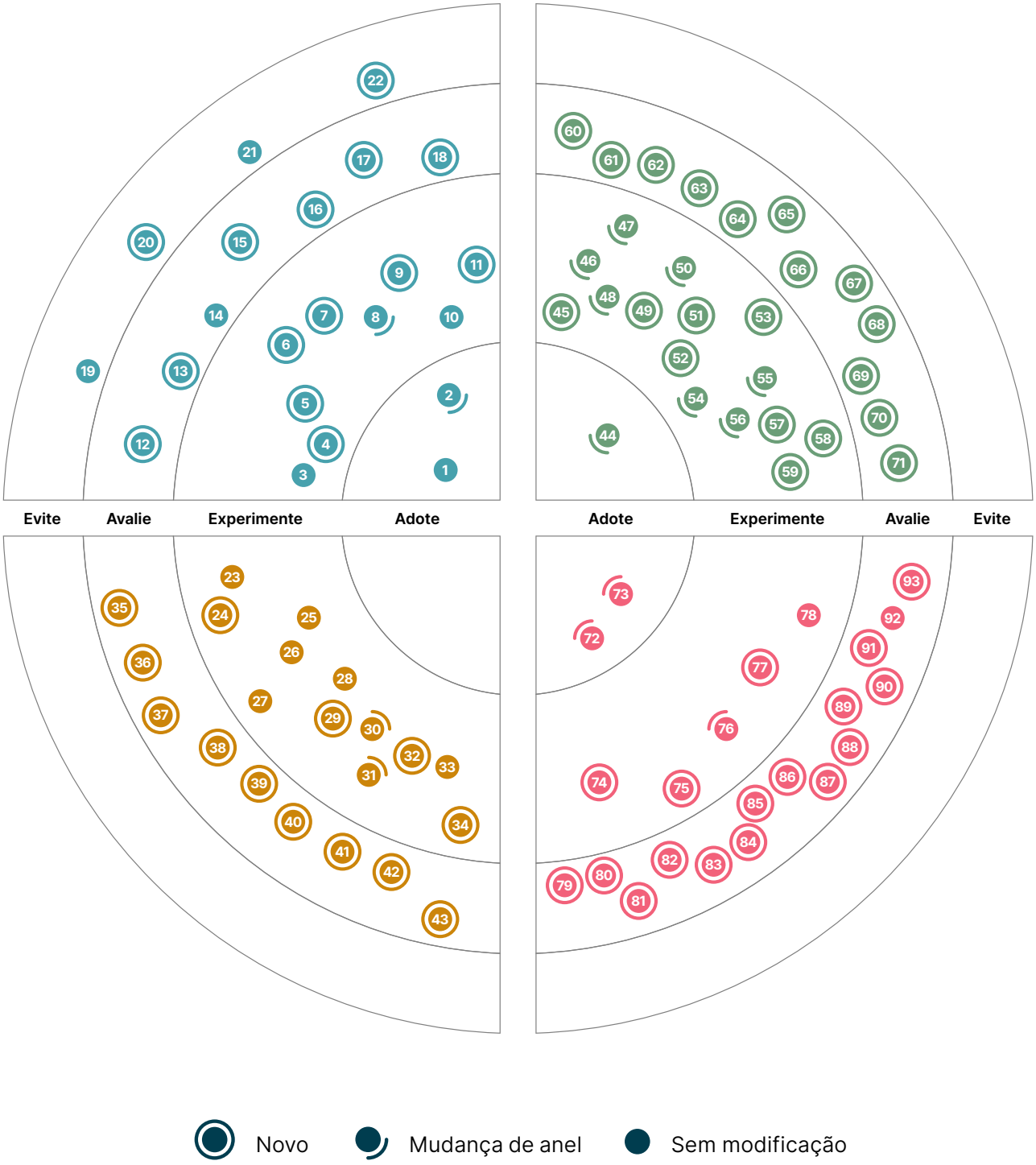
Categorias de frameworks em rápido crescimento parecem ser um padrão comum no Radar: um framework básico se torna popular, sendo seguido por uma série de ferramentas que criam um ecossistema para lidar com evoluções e deficiências comuns, finalizando com a consolidação em torno de algumas ferramentas populares. No entanto, o gerenciamento de estado do React parece resistente a essa tendência comum. Desde que o Redux foi lançado, observamos um fluxo constante de ferramentas e frameworks que gerenciam estado de maneiras ligeiramente diferentes, cada uma com um conjunto diferente de compensações. Não sabemos por quê, então podemos apenas especular: essa é a rotatividade natural que o ecossistema JavaScript parece promover? É uma deficiência subjacente no React, um problema divertido e aparentemente tratável que incentiva as pessoas desenvolvedoras a experimentar? Ou é a incompatibilidade de impedância permanente entre um formato de leitura de documentos (navegadores web) e a interatividade necessária (e o estado) para hospedar o desenvolvimento de aplicações em cima de documentos? Não sabemos o motivo, mas aguardamos com curiosidade a próxima rodada de tentativas de solução para esse problema aparentemente perpétuo.

## A busca sem fim pelo catálogo de dados mestre

O desejo de obter mais valor dos ativos de dados corporativos impulsiona grande parte do investimento em tecnologia digital que vemos hoje. Em sua essência, esse esforço geralmente se concentra em melhores maneiras de encontrar e acessar todos os dados relevantes. Por quase tanto tempo quanto as empresas coletam dados digitais, existem esforços para racionalizá-los e catalogá-los em um único diretório corporativo hierárquico. Mas, seguidamente, essa noção intuitivamente atraente esbarra na dura realidade da complexidade, redundância e ambiguidade inerentes às grandes organizações. Recentemente, notamos um interesse renovado em catálogos de dados corporativos e uma onda de submissões de blips para o Radar sugerindo novas ferramentas inteligentes, como **Collibra** e **DataHub**. Essas ferramentas podem fornecer acesso consistente e detectável à linhagem e metadados em silos, mas seus conjuntos de recursos em expansão também se estendem a governança, gerenciamento de qualidade, publicação e muito mais.

Em contraste com essa tendência, também parece haver um movimento crescente de afastamento do gerenciamento de dados centralizado e hierarquizado, em direção à governança e descoberta federadas com base em uma arquitetura de malha de dados. Essa abordagem lida com a complexidade inerente dos dados corporativos, definindo expectativas e padrões de forma centralizada, mas segregando a custódia de dados ao longo das linhas de domínio do negócio. Os times de produtos de dados orientados a domínio controlam e compartilham seus próprios metadados, incluindo capacidade de descoberta, qualidade e outras informações. Nesse cenário, o catálogo é apenas uma forma de apresentar informações para pesquisa e navegação. Os catálogos de dados resultantes são mais simples e fáceis de manter, reduzindo a necessidade de plataformas de governança e catalogação ricas em recursos.

# O Radar



# O Radar

## Técnicas

**Adote**

1. Quatro métricas fundamentais
2. Parede remota única para o time

**Experimente**

3. Malha de dados
4. Definição de prontidão para produção
5. Quadrantes de documentação
6. Repensando as standups remotas
7. UI orientada a servidor
8. Lista de materiais de software
9. Tactical forking
10. Carga cognitiva do time
11. Arquitetura transitional

**Avalie**

12. CUPID
13. Design inclusivo
14. Padrão Operador para recursos não clusterizados
15. Malha de serviços sem sidecar
16. SLSA
17. Streaming data warehouse
18. TinyML

**Evite**

19. Azure Data Factory para orquestração
20. Times de plataforma genéricos
21. Dados de produção em ambientes de teste
22. SPA como padrão

## Plataformas

**Adote**

—

**Experimente**

23. Azure DevOps
24. Modelos do Azure Pipeline
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Fluxos de trabalho reutilizáveis no Github Actions
33. Sealed Secrets
34. VerneMQ

**Avalie**

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

**Evite**

—

# O Radar

## Ferramentas

**Adote**
44. tfsec

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

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

**Evite**
—

## Linguagens e frameworks

**Adote**
72. SwiftUI
73. Testcontainers

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

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

**Evite**
—

# Técnicas

**Adote**

1. Quatro métricas fundamentais
2. Parede remota única para o time

**Experimente**

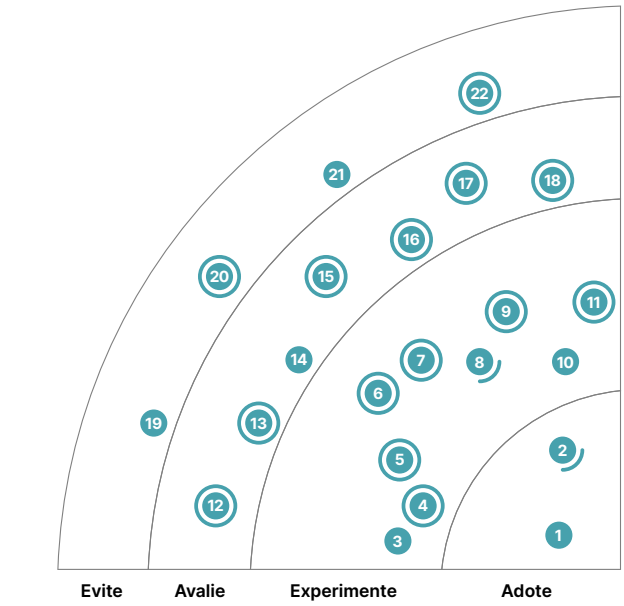
3. Malha de dados
4. Definição de prontidão para produção
5. Quadrantes de documentação
6. Repensando as standups remotas
7. UI orientada a servidor
8. f materiais de software
9. Tactical forking
10. Carga cognitiva do time
11. Arquitetura transitional

**Avalie**

12. CUPID
13. Design inclusivo
14. Padrão Operador para recursos não clusterizados
15. Malha de serviços sem sidecar
16. SLSA
17. Streaming data warehouse
18. TinyML

**Evite**

19. Azure Data Factory para orquestração
20. Times de plataforma genéricos
21. Dados de produção em ambientes de teste
22. SPA como padrão




Novo  Mudança de anel  Sem modificação

## 1. Quatro métricas fundamentais

*Adote*

Para medir o desempenho da entrega de software, cada vez mais organizações vêm adotando as quatro métricas fundamentais, definidas pelo programa **DORA research**: lead time, frequência de implantação, tempo médio de restauração (MTTR) e porcentagem de falha de alteração. A pesquisa e sua análise estatística mostram uma ligação nítida entre alta performance de entrega e essas métricas, que fornecem um ótimo indicador de desempenho para um time, ou até mesmo uma organização inteira.

Ainda defendemos fortemente o uso dessas métricas, mas aprendemos algumas lições. Ainda temos observado o uso cada vez mais frequente de abordagens de medição equivocadas, com ferramentas que ajudam os times a obter essas métricas baseando-se somente em seus canais de entrega contínua (CD). Particularmente, quando se trata de métricas de estabilidade (MTTR e porcentagem de falha de alteração), os dados do pipeline de CD por si só não fornecem informações suficientes para determinar o que é uma falha de implantação com impacto real para os usuários. As métricas de estabilidade só fazem sentido se incluírem dados sobre incidentes reais que prejudicam a experiência de uso do serviço.

Recomendamos sempre ter em mente a intenção final por trás de uma métrica, usando-a para refletir e aprender. Por exemplo, antes de passar semanas desenvolvendo ferramentas sofisticadas de dashboard, considere incluir a **verificação rápida do DORA** regularmente em retrospectivas do time. Isso dá ao time oportunidade de refletir sobre quais **recursos** podem ser trabalhados para melhorar suas métricas, o que pode ser muito mais efetivo do que usar ferramentas prontas para uso excessivamente detalhadas. Tenha em mente que essas quatro métricas principais se originaram de pesquisas de nível organizacional com times de alto desempenho, e o uso dessas métricas no nível do time deve ser uma maneira de refletir sobre os comportamentos do time, e não apenas mais um conjunto de métricas para adicionar ao dashboard.

## 2. Parede remota única para o time

*Adote*

Uma parede remota única para o time é uma técnica simples para reintroduzir a parede do time virtualmente. Recomendamos que os times distribuídos adotem essa abordagem. Uma das coisas que ouvimos de equipes que migraram para o trabalho remoto é que sentem falta de ter a parede física do time. Era um lugar único em que todos os vários cartões de histórias, tarefas, status e progresso podiam ser exibidos, funcionando como um radiador de informações e um hub para o time. A parede assumiu o papel de um ponto de integração, com os dados reais sendo armazenados em diferentes sistemas. À medida que os times se tornaram remotos, precisaram voltar a olhar para os sistemas de origem individuais, e obter uma visão rápida de um projeto tornou-se muito difícil. Embora possa haver alguma sobrecarga em mantê-la atualizada, sentimos que os benefícios para o time justificam. Para algumas equipes, a atualização da parede física fazia parte das "cerimônias" diárias que eram realizadas em conjunto, e o mesmo pode ser feito com uma parede remota.

## 3. Malha de dados

*Experimente*

A **malha de dados** é uma abordagem organizacional e técnica descentralizada de compartilhamento, acesso e gerenciamento de dados para análise e ML. Seu objetivo é criar uma abordagem *sociotécnica* que expanda a obtenção de valor dos dados à medida que a complexidade da organização cresce e à medida que os casos de uso de dados se multiplicam e as fontes de dados se diversificam. Essencialmente, a abordagem cria um modelo de compartilhamento de

dados *responsável* que acompanha o crescimento organizacional e mudança contínua. Em nossa experiência, o interesse na aplicação de malha de dados cresceu imensamente, inspirando muitas organizações a adotá-la e muitas fornecedoras de tecnologia a repensar suas ofertas existentes para incluir uma implantação de malha. Apesar do interesse e da experiência crescente em malha de dados, suas implementações enfrentam alto custo de integração. Além disso, sua adoção permanece limitada a determinadas partes de grandes organizações, e fornecedoras de tecnologia estão desviando a atenção das organizações em relação aos aspectos *sociais* inegociáveis da malha de dados – descentralização de dados e um modelo operacional de governança federada.

Essas ideias são exploradas no livro ***Data Mesh, Delivering Data-Driven Value at Scale***, que orienta profissionais de arquitetura, lideranças técnicas e responsáveis pela tomada de decisões em suas jornadas de transição da arquitetura tradicional de big data para a malha de dados. O material fornece uma introdução completa aos princípios da malha de dados e suas constituintes e aborda como projetar uma arquitetura de malha de dados, como orientar e executar uma estratégia de malha de dados e como navegar pelo design organizacional em um modelo de propriedade de dados descentralizado. O objetivo do livro é criar um novo framework para conversas mais profundas e levar a malha de dados ao próximo estágio de maturidade.

## 4. Definição de prontidão para produção

***Experimente***

Em uma organização que pratica o princípio "você constrói, você executa", uma definição de prontidão para produção (DPR) é uma técnica útil para apoiar os times na avaliação e preparação da prontidão operacional de novos serviços. Implementada como uma lista de verificação ou um modelo, uma DPR orienta os times sobre o que considerar antes de colocar um novo serviço em produção. Embora DPRs não definam objetivos específicos no nível de serviço (*service-level objectives* ou SLOs) a serem cumpridos (esses seriam difíceis de definir como tamanho único), funcionam como lembretes de quais categorias de SLOs os times devem considerar, quais padrões organizacionais devem ser cumpridos e qual documentação é necessária. DPRs fornecem uma fonte de input que os times transformam em requisitos específicos de produtos para seus respectivos casos de uso, por exemplo, observabilidade e confiabilidade para alimentar seus backlogs de produto.

As DPRs estão intimamente relacionadas ao conceito do Google de **revisão de prontidão para produção (PRR)**. Em organizações muito pequenas para uma equipe dedicada de engenharia de confiabilidade de sites, ou em organizações que se preocupam com o impacto negativo que um processo de revisão por conselho pode ter no fluxo de lançamento de um time, ter uma DPR pode ao menos fornecer alguma orientação e documentar os critérios acordados para a organização. Para novos serviços altamente críticos, uma verificação extra no cumprimento do DPR pode ser adicionada por meio de uma PRR quando necessário.

## 5. Quadrantes de documentação

***Experimente***

Escrever uma boa documentação é um aspecto negligenciado do desenvolvimento de software, muitas vezes sendo deixado para o último minuto e executado de forma casual. Alguns de nossos times encontraram nos **quadrantes de documentação** uma maneira prática de garantir que os artefatos corretos estejam sendo produzidos. Essa técnica classifica os artefatos em dois eixos: o primeiro eixo diz respeito à natureza da informação — prática ou teórica; o segundo eixo descreve o contexto em que os artefatos são usados — estudo ou trabalho. Dessa forma, são definidos quatro quadrantes nos quais artefatos como tutoriais, guias de instruções ou páginas de referência podem

ser colocados e compreendidos. Esse sistema de classificação não apenas garante que artefatos críticos não sejam esquecidos, mas também orienta a apresentação do conteúdo. Achamos essa técnica particularmente útil para criar documentação de onboarding e acelerar a integração de pessoas desenvolvedoras em um novo time.

## 6. Repensando as standups remotas

***Experimente***

O termo *standup* surgiu da ideia de ficar de pé durante essa reunião diária de sincronização, com o objetivo de torná-la curta. É um princípio comum que muitos times tentam seguir em suas standups: mantê-la objetiva e pragmática. Atualmente, no entanto, temos visto times desafiando esse princípio e repensando as standups remotas. Presencialmente, há muitas oportunidades no decorrer do dia para que as pessoas sincronizem umas com as outras de forma espontânea, de forma complementar à standup curta. Remotamente, alguns de nossos times estão experimentando um formato de reunião mais longo, semelhante ao que o pessoal da Honeycomb chama de "***meandering team sync***".

Não se trata de se livrar completamente de uma sincronização diária, ainda consideramos isso muito importante e valioso, especialmente em uma configuração remota. Em vez disso, trata-se de estender o tempo reservado nas agendas para a sincronização diária para até uma hora, e usá-lo de uma maneira que torne algumas das outras reuniões do time obsoletas e aproxime as pessoas do time. As atividades ainda podem incluir a já bastante testada revisão do quadro do time, mas sendo estendidas para discussões de esclarecimento mais detalhadas, decisões rápidas e tempo para socializar. A técnica pode ser considerada bem-sucedida se reduzir a carga geral da reunião e melhorar a conexão entre o time.

## 7. UI orientada a servidor

***Experimente***

Quando criamos uma nova edição do Radar, muitas vezes experimentamos uma sensação de *déjà vu*. A técnica de UI orientada a servidor é um caso particularmente sólido, com o advento de frameworks que permitem que pessoas desenvolvedoras mobile se beneficiem de ciclos de mudança mais rápidos, sem infringir nenhuma das políticas da loja de aplicativos em relação à revalidação do aplicativo móvel em si. Já abordamos isso anteriormente sob a perspectiva de permitir que o desenvolvimento para dispositivos móveis seja **escalado entre os times**. A interface de usuário orientada a servidor separa a renderização em um contêiner genérico no aplicativo móvel, enquanto a estrutura e os dados de cada visualização são fornecidos pelo servidor. Isso significa que as alterações que antes exigiam uma viagem de ida e volta à loja de aplicativos agora podem ser realizadas por meio de alterações simples nas respostas que o servidor envia. Observe que não estamos recomendando essa abordagem para todo o desenvolvimento de UI. Na verdade, tivemos experiências com algumas bagunças terríveis e excessivamente configuráveis, mas, com o apoio de gigantes como AirBnB e Lyft, suspeitamos que não sejamos apenas nós da Thoughtworks que nos cansamos de **tudo sendo feito do lado do cliente**. Fique de olho nesse espaço.

## 8. Lista de materiais de software

***Experimente***

Com a pressão contínua para manter os sistemas seguros e nenhum sinal de recuo no cenário geral de ameaças, uma lista de materiais de software (*software bill of materials* ou SBOM) legível por máquina pode ajudar os times a se manterem atualizados sobre os problemas de segurança em suas bibliotecas de confiança. A recente exploração remota de dia zero do **Log4Shell** foi crítica e teve amplo alcance; se os times tivessem uma SBOM pronta, poderiam ter verificado e corrigido rapidamente. Agora, temos experiências usando SBOMs em produção em projetos que vão de

pequenas empresas a grandes multinacionais e até departamentos governamentais, e temos convicção de que as listas trazem benefícios. Ferramentas como Syft facilitam o uso de uma SBOM para detecção de vulnerabilidades.

## 9. Tactical forking

*Experimente*

Tactical forking é uma técnica que pode ajudar na reestruturação ou migração de bases de código monolíticas para microsserviços. Especificamente, essa técnica oferece uma alternativa possível para a abordagem mais comum, modularizar integralmente a base de código primeiro, o que em muitas circunstâncias pode levar muito tempo ou ser muito difícil de alcançar. Com a técnica de tactical forking, um time pode criar um novo fork da base de código e usá-lo para abordar e extrair uma preocupação ou área específica enquanto exclui o código desnecessário. O uso dessa técnica seria provavelmente apenas uma parte de um plano de longo prazo para o monólito de forma geral.

## 10. Carga cognitiva do time

*Experimente*

A arquitetura de um sistema imita a estrutura organizacional e sua comunicação. Não é exatamente novidade que devemos ser intencionais em relação a como os times interagem — veja, por exemplo, a Manobra Inversa de Conway. A interação do time é uma das variáveis que determinam a rapidez e a facilidade com as quais os times podem entregar valor a clientes. Ficamos felizes em encontrar uma maneira de medir essas interações. Usamos a avaliação dos autores do livro Topologias de Time, que oferece uma compreensão de como pode ser fácil ou difícil para os times construir, testar e manter seus serviços. Medindo a carga cognitiva do time, fomos capazes de aconselhar melhor nossas clientes sobre como mudar a estrutura de seus times e evoluir suas interações.

## 11. Arquitetura transicional

*Experimente*

Uma arquitetura transicional é uma prática útil ao substituir sistemas legados. Assim como os andaimes podem ser construídos, reconfigurados e finalmente removidos durante a construção ou reforma de um edifício, muitas vezes você precisa de etapas arquiteturais provisórias durante o deslocamento do legado. As arquiteturas transicionais serão removidas ou substituídas posteriormente, mas não são trabalhos descartáveis, devido ao importante papel que desempenham para reduzir riscos e permitir que um problema difícil seja dividido em etapas menores. Dessa forma, ajudam a evitar a armadilha de adotar uma abordagem de substituição de legado “big bang”, porque você não pode alinhar pequenos passos provisórios à visão final de arquitetura. É preciso ter cuidado para garantir que o “andaime” arquitetural seja eventualmente removido, para que não se torne dívida técnica mais tarde.

## 12. CUPID

*Avalie*

Como você aborda a escrita de um bom código? Como você avalia se escreveu um bom código? Como pessoas desenvolvedoras de software, estamos sempre buscando regras, princípios e padrões interessantes que possamos usar para compartilhar linguagem e valores ao escrever código simples e fácil de alterar.

Daniel Terhorst-North recentemente executou uma nova tentativa de criar uma lista de verificação para um bom código. Ele argumenta que, em vez de se ater a um conjunto de regras como SOLID, usar um conjunto de propriedades para visar é mais aplicável. Ele criou o que chama de

propriedades CUPID para descrever o que devemos nos esforçar para alcançar um código "joyful": o código deve ser *composable*, seguir a filosofia Unix e ser previsível, idiomático e baseado em domínio.

## 13. Design inclusivo

***Avalie***

Recomendamos que as organizações avaliem o **design inclusivo** como forma de garantir que a acessibilidade seja tratada como um requisito de primeira classe. Com muita frequência, os requisitos de acessibilidade e inclusão são ignorados até pouco antes, se não logo depois do lançamento do software. A maneira mais barata e simples de acomodar esses requisitos, além de fornecer feedback antecipado aos times, é incorporá-los totalmente ao processo de desenvolvimento. Anteriormente destacamos técnicas que executam um "shift-left" para segurança e requisitos multifuncionais. Uma perspectiva dessa técnica é atingir esse mesmo objetivo para acessibilidade.

## 14. Padrão Operador para recursos não clusterizados

***Avalie***

Continuamos vendo o uso crescente do padrão **Kubernetes Operator** para outras finalidades além do gerenciamento de aplicativos implantados no cluster. O uso do padrão Operador para recursos não clusterizados se beneficia das definições de recursos personalizadas e do mecanismo de agendamento orientado a eventos implementado no plano de controle do Kubernetes para gerenciar atividades relacionadas ainda fora do cluster. Essa técnica se baseia na ideia dos **serviços em nuvem gerenciados por Kube** e a estende a outras atividades, como implantação contínua ou reação a mudanças em repositórios externos. Uma vantagem dessa técnica em relação a uma ferramenta específica é a ampla gama de ferramentas que vêm com o Kubernetes ou fazem parte de um ecossistema mais amplo. Você pode usar comandos como *diff*, *dry-run* ou *apply* para interagir com os recursos personalizados do operador. O mecanismo de agendamento do Kube facilita o desenvolvimento eliminando a necessidade de orquestrar as atividades na ordem correta. Ferramentas de código aberto como **Crossplane**, **Flux** e **Argo CD** se beneficiam desta técnica, e esperamos ver mais opções surgirem ao longo do tempo. Embora essas ferramentas tenham seus casos de uso, também estamos começando a ver o inevitável mau uso, ou uso excessivo dessa técnica, e precisamos repetir algumas recomendações antigas: só porque você *pode* fazer algo com uma ferramenta não significa que você *deva* fazer. Certifique-se de descartar abordagens convencionais mais simples antes de criar uma definição de recurso personalizada e se comprometer com a complexidade que acompanha essa abordagem.

## 15. Malha de serviços sem sidecar

***Avalie***

A **malha de serviços** é geralmente implementada como um processo de proxy reverso, também conhecido como sidecar, implantado ao lado de cada instância de serviço. Embora os sidecars sejam processos leves, o custo geral e a complexidade operacional da adoção da malha de serviços aumentam a cada nova instância do serviço que exige outro sidecar. No entanto, com os avanços no **eBPF**, estamos observando uma nova abordagem de **malha de serviços sem sidecar**, na qual as funcionalidades da malha são enviadas com segurança para o kernel do sistema operacional, permitindo assim que os serviços no mesmo nó se comuniquem de forma transparente por meio de soquetes sem a necessidade de proxies adicionais. Você pode tentar isso com **Cilium service mesh** e simplificar a implantação de um proxy por serviço para um proxy por nó. Os recursos do eBPF despertaram nosso interesse, e achamos importante avaliar essa evolução da malha de serviço.

## 16. SLSA

***Avalie***

À medida que o software continua a crescer em complexidade, proteger o vetor de ameaças de dependências de software se torna cada vez mais desafiador. A recente vulnerabilidade do Log4J mostrou o quão difícil pode ser até mesmo conhecer essas dependências — muitas empresas que não usavam o Log4J diretamente estavam inadvertidamente vulneráveis simplesmente porque outros softwares em seu ecossistema dependiam do mesmo. *Supply-chain Levels for Software Artifacts*, ou **SLSA** (pronuncia-se "salsa"), é um conjunto de orientações selecionadas por um consórcio para que as organizações se protejam contra ataques em cadeias de fornecimento, desenvolvido a partir de orientações internas que o Google vem usando há anos. Apreciamos o fato de SLSA não prometer uma abordagem "bala de prata" limitada a ferramentas para proteger a cadeia de fornecimento, em vez disso, fornecendo uma lista de verificação de ameaças e práticas concretas ao longo de um modelo de maturidade. O **modelo de ameaça** é fácil de seguir com exemplos reais de ataques e os **requisitos** fornecem orientações para ajudar as organizações a priorizar ações com base em níveis de robustez crescente para melhorar sua postura de segurança da cadeia de fornecimento. Acreditamos que SLSA fornece recomendações aplicáveis e esperamos que mais organizações aprendam com o conjunto.

## 17. Streaming data warehouse

***Avalie***

A necessidade de responder rapidamente aos insights de clientes impulsionou a crescente adoção de arquiteturas orientadas a eventos e processamento de fluxo. Frameworks como **Spark**, **Flink** ou **Kafka Streams** oferecem um paradigma no qual consumidores e produtores de eventos simples podem cooperar em redes complexas para fornecer insights em tempo real. Mas esse estilo de programação leva tempo e esforço para ser dominado e, quando implementado como aplicação de ponto único, carece de interoperabilidade. Fazer o processamento de fluxo funcionar universalmente em larga escala pode exigir um investimento significativo em engenharia. Agora, está surgindo uma nova safra de ferramentas que oferece os benefícios do processamento de fluxo para um grupo mais amplo e estabelecido de pessoas desenvolvedoras que se sentem à vontade usando SQL para implementar análises. A padronização do SQL como a linguagem de streaming universal reduz a barreira para a implementação de aplicações de dados de streaming. Ferramentas como **ksqlDB** e **Materialize** ajudam a transformar essas aplicações separadas em plataformas unificadas. Em conjunto, uma coleção de aplicações de streaming baseadas em SQL em uma empresa pode constituir um *streaming data warehouse*.

## 18. TinyML

***Avalie***

Até recentemente, a execução de um modelo de aprendizado de máquina (ML) era vista como algo caro do ponto de vista computacional e, em alguns casos, exigia hardware para fins especiais. Embora seu processo de criação ainda esteja amplamente incluído nessa classificação, os modelos podem ser construídos de maneira a permitir sua execução em dispositivos pequenos, de baixo custo e baixo consumo de energia. Essa técnica, chamada **TinyML**, abriu a possibilidade de execução de modelos de ML em situações que muitas pessoas considerariam inviáveis. Por exemplo, em dispositivos alimentados por bateria ou em ambientes desconectados, com conectividade limitada ou irregular, o modelo pode ser executado localmente sem um custo proibitivo. Se você está pensando em usar ML, mas achou que seria algo irreal devido às restrições de computação ou rede, vale a pena avaliar essa técnica.

## 19. Azure Data Factory para orquestração

*Evite*

Para organizações que usam Azure como provedor de nuvem principal, Azure Data Factory (ADF) é atualmente o padrão para orquestrar pipelines de processamento de dados. O serviço oferece suporte à ingestão de dados, copiando dados de/para diferentes tipos de armazenamento locais, ou no Azure, e executando a lógica de transformação. Embora nossa experiência com o ADF tenha sido adequada para migrações simples de armazenamentos de dados local para a nuvem, desencorajamos o uso do Azure Data Factory para orquestração de fluxos de trabalho e pipelines de processamento de dados complexos. Tivemos algum sucesso usando ADF principalmente para mover dados entre sistemas. Para pipelines de dados mais complexos, ainda há desafios, incluindo depuração e relatórios de erros pobres; observabilidade limitada, já que os recursos de log do ADF não se integram a outros produtos, como Azure Data Lake Storage ou Databricks, dificultando a implementação de uma observabilidade de ponta a ponta; e disponibilidade de mecanismos de acionamento de fontes de dados apenas para determinadas regiões. No momento, incentivamos o uso de outras ferramentas de orquestração de código aberto (por exemplo, Airflow) para pipelines de dados complexos, limitando ADF para cópia de dados ou snapshots. Nossos times continuam usando o Data Factory para mover e extrair dados, mas para operações maiores, recomendamos outras ferramentas de fluxo de trabalho mais completas.

## 20. Times de plataforma genéricos

*Evite*

Anteriormente, apresentamos **times de produto de engenharia de plataforma** em Adotar como uma boa maneira de operar para equipes de plataformas internas, permitindo assim que times de entrega habilitem o autoatendimento para implantação e operação de sistemas com tempo de espera e complexidade de stack reduzidos. Infelizmente, estamos vendo o rótulo de “time de plataforma” aplicado a equipes dedicadas a projetos que não têm resultados nítidos ou um conjunto bem definido de clientes. Como resultado, esses times de plataforma genéricos encontram dificuldades para entregar devido às altas cargas cognitivas e à falta de alinhamento de prioridades, pois estão lidando com uma coleção variada de sistemas não relacionados. Dessa forma, tornam-se efetivamente equipes genéricas de suporte geral para coisas que não se encaixam ou que são indesejadas em outros lugares. Continuamos acreditando que os times de produto de engenharia de plataforma com foco em um produto (interno) bem definido oferecem um melhor conjunto de resultados.

## 21. Dados de produção em ambientes de teste

*Evite*

Continuamos a perceber dados de produção em ambientes de teste como uma área de preocupação. Em primeiro lugar, muitos exemplos dessa prática resultaram em danos à reputação, por exemplo, quando um alerta incorreto foi enviado de um sistema de teste para toda uma base de clientes. Em segundo lugar, o nível de segurança, especificamente em torno da proteção de dados privados, tende a ser menor para sistemas de teste. Não adianta ter controles elaborados sobre o acesso aos dados de produção se esses dados forem copiados para um banco de dados de teste que pode ser acessado por todas as pessoas desenvolvedoras e QAs. Embora seja *possível* ofuscar os dados, isso tende a ser aplicado apenas a campos específicos, por exemplo, números de cartão de crédito. Por fim, copiar dados de produção para sistemas de teste pode violar as leis de privacidade, por exemplo, quando os sistemas de teste são hospedados em ou acessados de um país ou região diferente. Este último cenário é especialmente problemático com implantações de nuvem complexas. Dados falsos são uma abordagem mais segura e existem ferramentas que ajudam a criá-los. Reconhecemos que existem razões para que elementos

*específicos* dos dados de produção sejam copiados, por exemplo, para reprodução de bugs ou treinamento de modelos de ML específicos. Aqui, nosso conselho é proceder com cautela

## 22. SPA como padrão

***Evite***

Geralmente evitamos colocar blips no anel Evite quando consideramos uma recomendação muito óbvia, por exemplo, seguir cegamente um estilo de arquitetura sem prestar atenção às compensações. No entanto, a grande prevalência de times que escolhem uma aplicação de página única (SPA) por padrão quando precisam de um site nos despertou a preocupação de que as pessoas não estejam nem mesmo reconhecendo SPAs como um estilo de arquitetura — em vez disso, saltando imediatamente para a seleção de framework. SPAs incorrem em uma complexidade que simplesmente não existe em sites tradicionais baseados em servidor: otimização de mecanismo de pesquisa, gerenciamento de histórico do navegador, web analytics, tempo de carregamento da primeira página etc. Essa complexidade é frequentemente justificada por questões de experiência de uso (embora a rotatividade em torno do gerenciamento de estado na comunidade React indique o quão difícil pode ser obter uma solução de aplicação geral). Muitas vezes, porém, não vemos os times fazendo essa análise de compensação, aceitando cegamente a complexidade da prática de SPA como padrão, mesmo quando as necessidades de negócio não justificam. De fato, começamos a perceber que muitas pessoas desenvolvedoras mais inexperientes nem sequer estão cientes da existência de uma abordagem alternativa, pois passaram toda a sua carreira em um framework como o React. Acreditamos que muitos sites se beneficiarão da simplicidade da lógica do lado do servidor e técnicas que ajudam a fechar a lacuna na experiência de uso, como **Hotwire**, são encorajadoras.

# Plataformas

**Adote**

—

**Experimente**

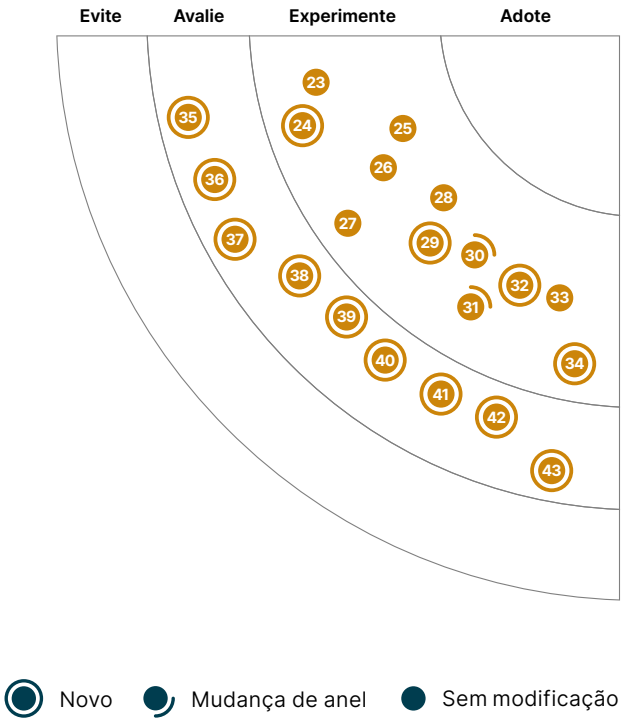
23. Azure DevOps
24. Modelos do Azure Pipeline
25. CircleCI
26. Couchbase
27. eBPF
28. GitHub Actions
29. GitLab CI/CD
30. Google BigQuery ML
31. Google Cloud Dataflow
32. Fluxos de trabalho reutilizáveis no Github Actions
33. Sealed Secrets
34. VerneMQ

**Avalie**

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

**Evite**

—



## 23. Azure DevOps

*Experimente*

À medida que o ecossistema do Azure DevOps continua crescendo, nossos times têm tido mais sucesso usando-o. O pacote contém um conjunto de serviços gerenciados, incluindo repositórios Git hospedados, pipelines de compilação e implantação, ferramentas de teste automatizadas, ferramentas de gerenciamento de backlog e repositório de artefatos. Observamos nossos times ganharem experiência no uso dessa plataforma com bons resultados, o que significa que o Azure DevOps está amadurecendo. Gostamos particularmente de sua flexibilidade, permitindo que você use os serviços que deseja, ainda que sejam de provedores diferentes. Por exemplo, você pode usar um repositório Git externo enquanto usa os serviços de pipeline do Azure DevOps. Nossos times estão especialmente empolgados com o Azure DevOps Pipelines. À medida que o ecossistema amadurece, temos visto um crescimento no onboarding de times que já adotam a stack do Azure, pelo fato de se integrar facilmente ao restante do universo Microsoft.

## 24. Modelos do Azure Pipeline

*Experimente*

Os modelos do Azure Pipeline permitem que você remova duplicações em suas definições do Azure Pipeline por meio de dois mecanismos. Com os modelos "includes", você pode fazer referência a um modelo de forma que o mesmo se expanda em linha como uma macro C++ parametrizada, possibilitando uma maneira simples de fatorar a configuração comum entre estágios, trabalhos e etapas. Com modelos "extends", você pode definir um shell externo com configuração de pipeline comum, e com a verificação do modelo necessário, você pode falhar a compilação caso o pipeline não estenda determinados modelos, evitando ataques maliciosos contra a própria configuração do pipeline. Junto com as orbs do CircleCI e os mais recentes fluxos de trabalho reutilizáveis do GitHub Actions, os modelos do Azure Pipeline fazem parte da tendência de criar modularidade no design de pipeline em várias plataformas, e vários de nossos times ficaram satisfeitos em usá-los.

## 25. CircleCI

*Experimente*

Muitos de nossos times escolhem o CircleCI para suas necessidades de integração contínua e apreciam sua capacidade de executar pipelines complexos com eficiência. O time de desenvolvimento do CircleCI continua a adicionar novos recursos à plataforma, agora na versão 3.0. Os orbs e executors foram considerados particularmente úteis por nossos times. Orbs são trechos de código reutilizáveis que automatizam processos repetidos, aceleram a configuração do projeto e facilitam a integração com ferramentas de terceiros. A ampla variedade de tipos de executor oferece flexibilidade para configurar trabalhos em Docker, Linux, macOS ou VMs Windows.

## 26. Couchbase

*Experimente*

Quando o incluímos pela primeira vez no Radar em 2013, o Couchbase era visto primordialmente como um cache persistente que evoluiu a partir de uma fusão entre Membase e CouchDB. Desde então, o Couchbase passou por um processo de melhorias constantes, com um ecossistema de ferramentas relacionadas e ofertas comerciais crescendo ao ser redor. Entre as adições ao conjunto de produtos estão o Couchbase Mobile e o Couchbase Sync Gateway. Esses recursos trabalham juntos para manter os dados persistentes nos dispositivos de borda atualizados, mesmo quando o dispositivo fica offline por períodos de tempo devido à conectividade intermitente. À medida que esses dispositivos proliferam, vemos uma necessidade crescente de persistência incorporada que continue a funcionar independentemente de o dispositivo estar conectado ou não. Recentemente,

um de nossos times avaliou o Couchbase pela perspectiva de sua capacidade de sincronização offline e descobriu que esse recurso pronto para uso economizou um esforço considerável que, normalmente, teriam que investir.

## 27. eBPF

*Experimente*

Há vários anos, o kernel do Linux inclui o Berkeley Packet Filter estendido (eBPF), uma máquina virtual que oferece a capacidade de anexar filtros a soquetes específicos. Mas o eBPF vai muito além da filtragem de pacotes, permitindo que scripts personalizados sejam acionados em vários pontos dentro do kernel com pouquíssima sobrecarga. Embora essa tecnologia não seja nova, vem se destacando agora com o uso crescente de microsserviços implantados como contêineres orquestrados. Kubernetes e tecnologias de malha de serviço como Istio são comumente usadas e empregam sidecars para implementar funcionalidade de controle. Com novas ferramentas — Bumblebee, em particular, torna a construção, a execução e a distribuição de programas eBPF muito mais simples —, eBPF pode ser visto como uma alternativa ao sidecar tradicional. Um dos responsáveis pelo Cilium, uma ferramenta neste espaço, chegou a proclamar o fim do sidecar. Uma abordagem baseada em eBPF reduz parte da sobrecarga de desempenho e operação que vem com o uso de sidecars, mas não oferece suporte a recursos comuns, como terminação SSL.

## 28. GitHub Actions

*Experimente*

O GitHub Actions cresceu consideravelmente no ano passado, provando que pode assumir fluxos de trabalho mais complexos e chamar outras ações em ações compostas, entre outras coisas. A plataforma ainda tem algumas deficiências, como a incapacidade de reativar uma única atividade em um fluxo de trabalho. Embora o ecossistema no GitHub Marketplace tenha suas óbvias vantagens, dar a ações do GitHub de terceiros acesso ao seu pipeline de compilação gera o risco do compartilhamento de segredos de maneiras inseguras (recomendamos seguir as recomendações do GitHub para fortalecimento de segurança). No entanto, a conveniência de criar seu fluxo de trabalho de compilação diretamente no GitHub ao lado do código-fonte, combinada à capacidade de executar ações do GitHub localmente usando ferramentas de código aberto como act, torna-se uma opção atraente que facilitou a configuração e o processo de onboarding em nossos times.

## 29. GitLab CI/CD

*Experimente*

Se você estiver usando GitLab para gerenciar sua entrega de software, avalie também GitLab CI/CD para suas necessidades de integração e entrega contínua. Consideramos uma opção especialmente útil quando usada com GitLab local e executores auto-hospedados, pois essa combinação contorna as dores de cabeça de autorização geralmente causadas pelo uso de uma solução baseada em nuvem. Os executores auto-hospedados podem ser totalmente configurados para seus objetivos com o sistema operacional e as dependências corretas instalados e, como resultado, os pipelines podem ser executados com muito mais rapidez do que com um executor provisionado em nuvem que precisa ser configurado a cada vez.

Além do pipeline básico de compilação, testes e implantação, o produto do GitLab oferece suporte a Serviços, Auto Devops e ChatOps, entre outros recursos avançados. Os serviços são úteis na execução de serviços do Docker, como Postgres ou Testcontainer vinculados a um trabalho para integração e teste de ponta a ponta. O Auto Devops cria pipelines com configuração zero, o que é muito útil para times novos na entrega contínua ou para organizações com muitos repositórios que, caso contrário, precisariam criar muitos pipelines manualmente.

## 30. Google BigQuery ML

*Experimente*

Desde a última vez que falamos sobre o Google BigQuery ML, modelos mais sofisticados, como Deep Neural Networks e AutoML Tables, foram adicionados ao conectar o BigQuery ML com TensorFlow e Vertex AI como back-end. O BigQuery também introduziu suporte para previsão de séries temporais. Uma das nossas preocupações, anteriormente, era quanto à explicabilidade. No início deste ano, o BigQuery Explainable AI foi anunciado com disponibilidade geral, representando um passo importante na abordagem dessa questão. Também podemos exportar modelos do BigQuery ML para o Cloud Storage como um SavedModel do Tensorflow, e usá-los para previsão online. Ainda existem desvantagens como a facilidade de "entrega contínua para aprendizado de máquina", mas com sua baixa barreira de entrada, o BigQuery ML continua sendo uma opção atraente, principalmente quando os dados já residem no BigQuery.

## 31. Google Cloud Dataflow

*Experimente*

Google Cloud Dataflow é um serviço de processamento de dados baseado em nuvem para aplicações de fluxo de dados em lote e em tempo real. Nossos times estão usando o Dataflow para criar pipelines de processamento para integrar, preparar e analisar grandes conjuntos de dados, com o modelo de programação unificado do Apache Beam para facilitar o gerenciamento. Apresentamos o Dataflow pela primeira vez em 2018, e sua estabilidade, desempenho e conjunto de recursos avançados nos deixam confiantes em movê-lo para o anel Experimente nesta edição do Radar.

## 32. Fluxos de trabalho reutilizáveis no Github Actions

*Experimente*

Observamos um interesse crescente em GitHub Actions desde que o incluímos no Radar duas edições atrás. Com o lançamento dos fluxos de trabalho reutilizáveis, o GitHub continua a evoluir o produto de forma a abordar algumas de suas fraquezas iniciais. Os fluxos de trabalho reutilizáveis no Github Actions trazem modularidade ao design do pipeline, permitindo a reutilização parametrizada mesmo em repositórios (desde que o repositório de fluxo de trabalho seja público). Suportam a passagem explícita de valores confidenciais como segredos e podem passar saídas para o trabalho de chamada. Com algumas linhas de YAML, o GitHub Actions agora oferece o tipo de flexibilidade que você vê com as orbs do CircleCI ou os modelos do Azure Pipeline, mas sem precisar sair do GitHub como plataforma.

## 33. Sealed Secrets

*Experimente*

O Kubernetes oferece suporte nativo a um objeto de chave-valor conhecido como segredo. No entanto, por padrão, os segredos do Kubernetes não são realmente secretos, sendo tratados separadamente de outros dados de chave-valor para que as precauções ou o controle de acesso possam ser aplicados separadamente. Há suporte para criptografar segredos antes de armazená-los no etcd, mas os segredos começam como campos de texto simples em arquivos de configuração. Sealed Secrets é uma combinação de operador e utilitário de linha de comando que usa chaves assimétricas para criptografar segredos de forma que só possam ser descriptografados pelo controlador no cluster. Esse processo garante que os segredos não sejam comprometidos enquanto permanecem nos arquivos de configuração que definem uma implantação do Kubernetes. Depois de criptografados, esses arquivos podem ser compartilhados com segurança ou armazenados junto com outros artefatos de implantação.

## 34. VerneMQ

*Experimente*

VerneMQ é um broker MQTT distribuído de código aberto e alto desempenho. Já incluímos outros brokers MQTT no Radar anteriormente, como Mosquitto e EMQ. Assim como EMQ e RabbitMQ, o VerneMQ também é baseado em Erlang/OTP, o que o torna altamente escalável. Pode ser escalado horizontalmente e verticalmente em hardware comum para oferecer suporte a um grande número de editores e consumidores simultâneos, mantendo baixa latência e tolerância a falhas. Em nossos benchmarks internos, conseguimos alcançar alguns milhões de conexões simultâneas em um único cluster. Embora não seja novo, já o usamos em produção há algum tempo e nos atendeu bem.

## 35. actions-runner-controller

*Avalie*

actions-runner-controller é um controlador que opera executores auto-hospedados do Kubernetes para GitHub Actions em seu cluster Kubernetes. Com essa ferramenta, você cria um recurso de execução no Kubernetes, que irá executar e operar o executor auto-hospedado. Os executores auto-hospedados são úteis em cenários em que o trabalho executado pelo GitHub Actions precisa acessar recursos que não são acessíveis aos executores de nuvem do GitHub ou têm requisitos específicos de ambientes e sistemas operacionais diferentes dos fornecidos pelo GitHub. Nos casos em que você tem um cluster do Kubernetes, pode executar seus executores auto-hospedados como um pod do Kubernetes, com a capacidade de aumentar ou diminuir a escala conectando-se a eventos webhook do GitHub. O actions-controller-runner é leve e escalável.

## 36. Apache Iceberg

*Avalie*

Apache Iceberg é um formato de tabela aberta para conjuntos de dados analíticos muito grandes. O Iceberg suporta operações de dados analíticos modernas, como inserção, atualização, exclusão em nível de registro, consultas de viagem no tempo, transações ACID, particionamento oculto e evolução completa de esquema. Oferece suporte a vários formatos de armazenamento de arquivos subjacentes, como Apache Parquet, Apache ORC e Apache Avro. Muitos mecanismos de processamento de dados suportam Apache Iceberg, incluindo mecanismos SQL como Dremio e Trino, bem como mecanismos de streaming (estruturado) como Apache Spark e Apache Flink.

Apache Iceberg se enquadra na mesma categoria que Delta Lake e Apache Hudi. Todos suportam recursos mais ou menos semelhantes, mas se diferem nas implementações subjacentes e nas listas detalhadas de recursos. Iceberg é um formato independente e não é nativo de nenhum mecanismo de processamento específico, portanto, é compatível com um número crescente de plataformas, incluindo AWS Athena e Snowflake. Pelo mesmo motivo, o Apache Iceberg, ao contrário de formatos nativos como Delta Lake, pode não se beneficiar das otimizações quando usado com o Spark.

## 37. Blueboat

*Avalie*

Blueboat é uma plataforma multitenant para aplicações web sem servidor que usa o popular mecanismo JavaScript V8 e implementa nativamente bibliotecas de aplicações web comumente usadas em Rust para segurança e desempenho. Você pode considerar Blueboat como uma alternativa para CloudFlare Workers ou Deno Deploy, mas com uma distinção importante — você precisa operar e gerenciar a infraestrutura subjacente. Dito isso, recomendamos que você avalie cuidadosamente o Blueboat para suas necessidades sem servidor no local.

## 38. Cloudflare Pages

*Avalie*

Quando Cloudflare Workers foi lançado, nós o destacamos como uma forma inicial de função como serviço (FaaS) para computação de borda com uma implementação interessante. O lançamento do Cloudflare Pages em abril do ano passado não nos pareceu tão digno de nota, porque o Pages é apenas uma entre muitas soluções de hospedagem de sites apoiadas pelo Git. É verdade que a plataforma contava com visualizações contínuas, um recurso útil não encontrado na maioria das alternativas. Agora, porém, o Cloudflare conta com uma maior integração entre Workers e Pages, criando um Jamstack em execução na CDN. A inclusão de um armazenamento de chave-valor e uma primitiva de coordenação altamente consistente torna ainda mais atrativa a nova versão do Cloudflare Pages.

## 39. Colima

*Avalie*

Colima vem se tornando uma alternativa aberta e popular ao Docker Desktop. A aplicação provisiona o tempo de execução do contêiner do Docker em uma Lima VM, configura a CLI do Docker no macOS e lida com o encaminhamento de porta e escala de volumes. Colima usa containerd como tempo de execução, que também é o tempo de execução na maioria dos serviços gerenciados do Kubernetes (o que significa uma melhor paridade dev-prod). Com Colima, você pode usar e testar facilmente os recursos mais recentes do containerd, como carregamento lento para imagens de contêiner. Com seu bom desempenho, temos visto no Colima um grande potencial como alternativa de código aberto ao Docker Desktop.

## 40. Collibra

*Avalie*

No espaço cada vez mais disputado que é o mercado de catálogos de dados corporativos, nossos times gostaram de trabalhar com Collibra, destacando a flexibilidade de implantação de uma instância SaaS ou auto-hospedada, a ampla gama de funcionalidades prontas para uso, incluindo governança de dados, linhagem, qualidade e observabilidade. Os usuários também têm a opção de usar um subconjunto menor de recursos exigidos por uma abordagem mais descentralizada, como malha de dados. Seu verdadeiro diferencial tem sido o atendimento a clientes, considerado colaborativo e solidário por nossos times. É evidente que há uma tensão entre catálogos de dados simples e plataformas corporativas mais completas, mas até o momento, os times que o utilizam estão satisfeitos com a forma como Collibra atendeu às suas necessidades.

## 41. CycloneDX

*Avalie*

CycloneDX é um padrão para descrever uma lista de materiais de software (SBOM). À medida que o software e as malhas computacionais ganham complexidade, a definição de *software* se torna mais desafiadora. Com origem no OWASP, o CycloneDX aprimora o padrão SPDX antigo com uma definição mais ampla, que se estende para além das dependências da máquina local para incluir dependências de tempo de execução de serviço. Você também encontrará implementações em várias linguagens, um ecossistema de integrações de suporte e uma ferramenta CLI que permite analisar e alterar SBOMs com assinatura e verificação apropriadas.

## 42. Embeddinghub

*Avalie*

Embeddinghub é um banco de dados vetorial para embeddings de aprendizado de máquina e bastante semelhante ao Milvus. No entanto, com suporte pronto para uso para operações do tipo *approximate nearest neighbor* (ANN), particionamento, controle de versão e controle de acesso, recomendamos que você avalie o Embeddinghub para seus casos de uso de vetor de embedding.

## 43. Temporal

*Avalie*

Temporal é uma plataforma para desenvolver fluxos de trabalho de longa duração, principalmente para arquiteturas de microsserviços. Um fork do projeto anterior OSS Cadence do Uber, Temporal conta com um modelo de sourcing de eventos para fluxos de trabalho de longa duração, de forma que possam sobreviver a falhas de processo/máquina. Apesar de não recomendarmos o uso de transações distribuídas em arquiteturas de microsserviços, se você precisar implementá-las ou usar Sagas de longa duração, pode ser interessante considerar Temporal.

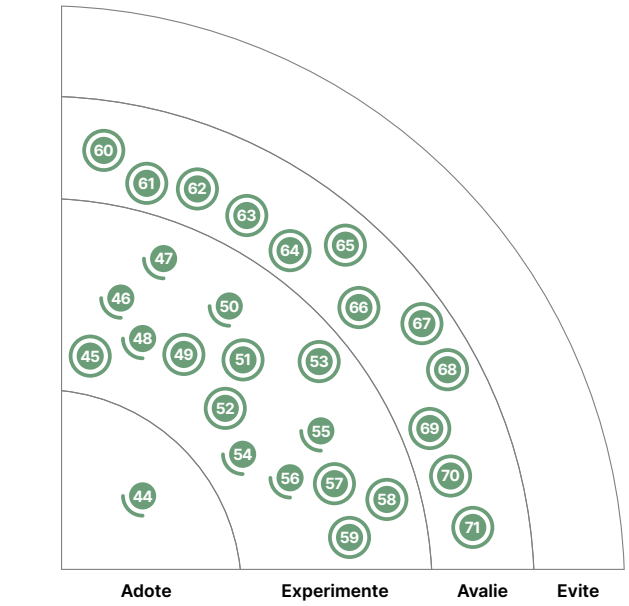
# Ferramentas

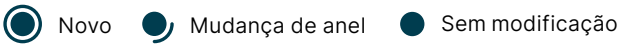
**Adote**
44. tfsec

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

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

**Evite**
—



## 44. tfsec

*Adote*

Em nossos projetos usando Terraform, tfsec tornou-se rapidamente uma ferramenta de análise estática padrão para detectar possíveis riscos de segurança. É fácil de integrar em um pipeline de CI e possui uma biblioteca crescente de verificações para todos os principais provedores de nuvem e plataformas, como Kubernetes. Dada a sua facilidade de uso, acreditamos que tfsec pode ser uma boa adição a qualquer projeto com Terraform.

## 45. AKHQ

*Experimente*

AKHQ é uma GUI para Apache Kafka que permite gerenciar tópicos, dados de tópicos, grupos de consumidores e mais. Alguns de nossos times consideram AKHQ uma ferramenta eficaz para observar o status em tempo real de um cluster Kafka. Você pode, por exemplo, navegar por tópicos em um cluster. Para cada tópico, você pode visualizar o nome, o número de mensagens armazenadas, o tamanho utilizado do disco, o horário do último registro, o número de partições, o fator de replicação com a quantidade em sincronia e o grupo de consumidores. Com opções para desserialização Avro e Protobuf, AKHQ pode ajudar a entender o fluxo de dados em seu ambiente Kafka.

## 46. cert-manager

*Experimente*

cert-manager é uma ferramenta para gerenciar seus certificados X.509 em seu cluster Kubernetes. O cert-manager modela certificados e emissores como tipos de recursos de primeira classe e fornece certificados como serviço com segurança para pessoas desenvolvedoras e aplicações que trabalham no cluster Kubernetes. Escolha óbvia ao usar o controlador de entrada padrão do Kubernetes, também é recomendado para outros casos e preferível em relação à rolagem manual de seu próprio gerenciamento de certificados. Vários de nossos times têm usado o cert-manager extensivamente, e também notamos que sua usabilidade melhorou muito nos últimos meses.

## 47. Cloud Carbon Footprint

*Experimente*

Cloud Carbon Footprint (CCF) é uma ferramenta de código aberto que usa APIs de nuvem para fornecer visualizações de emissões de carbono estimadas com base na utilização na AWS, GCP e Azure. A equipe da Thoughtworks usou a ferramenta com sucesso com várias organizações, incluindo empresas de tecnologia no setor de energia, varejo, provedores de serviços digitais e empresas que usam IA. As provedoras de plataforma em nuvem entendem que é importante ajudar clientes a entender o impacto do carbono gerado pelo uso de seus serviços, então, começaram a desenvolver funcionalidades semelhantes. Como o CCF é agnóstico de nuvem, permite que os usuários visualizem o uso de energia e as emissões de carbono de várias provedoras de nuvem em um só lugar, enquanto traduz as pegadas de carbono em impacto no mundo real, como voos ou árvores plantadas.

Em versões recentes, o CCF começou a incluir recomendações de otimização da Google Cloud e da AWS, além de possíveis economias de energia e CO2, além de oferecer suporte a mais tipos de instância de nuvem, como instâncias de GPU. Dada a tração que a ferramenta recebeu e a adição contínua de novos recursos, nós nos sentimos confiantes em movê-la para Experimente.

## 48. Conftest

*Experimente*

Conftest é uma ferramenta para escrever testes para dados de configuração estruturados. Baseia-se na Rego language da Open Policy Agent para escrever testes para configurações do Kubernetes, definições de pipeline do Tekton ou mesmo planos do Terraform. Temos tido ótimas experiências com o Confitest — e sua curva de aprendizado simples. Com feedbacks rápidos dos testes, nossos times iteram com rapidez e segurança nas alterações de configuração do Kubernetes.

## 49. kube-score

*Experimente*

kube-score é uma ferramenta que faz análise de código estático de definições de objeto do Kubernetes. A saída é uma lista de recomendações sobre o que você pode melhorar para tornar seu aplicativo mais seguro e resiliente. Conta com uma lista de verificações predefinidas que inclui práticas recomendadas, como executar contêineres com privilégios não root e especificar corretamente os limites de recursos. Já existe há algum tempo e o usamos em alguns projetos como parte de um pipeline de CD para manifestos do Kubernetes. Uma grande desvantagem do kube-score é que você não pode adicionar políticas personalizadas. Normalmente, complementamos com outras ferramentas como Conftest nesses casos.

## 50. Lighthouse

*Experimente*

Lighthouse é uma ferramenta criada pelo Google para avaliar aplicações e páginas web, coletando métricas de desempenho e insights sobre boas práticas de desenvolvimento. Há muito tempo defendemos testes de performance como elementos de primeira classe, e as adições ao Lighthouse que mencionamos há cinco anos certamente ajudaram nisso. Nosso pensamento em torno das funções de aptidão arquitetural criou uma forte motivação para que ferramentas como o Lighthouse fossem executadas em pipelines de entrega. Com a introdução do Lighthouse CI, ficou mais fácil do que nunca incluir o Lighthouse em pipelines gerenciados por várias ferramentas.

## 51. Metaflow

*Experimente*

Metaflow é uma biblioteca Python fácil de usar e um serviço de back-end que ajuda pessoas engenheiras e cientistas de dados a criar e gerenciar processamento de dados pronto para produção, treinamento de ML e fluxos de trabalho de inferência. O Metaflow fornece APIs Python que estruturam o código como um grafo direcionado de etapas. Cada etapa pode ser decorada com configurações flexíveis, como os recursos de computação e armazenamento necessários. Os artefatos de código e dados para a execução de cada etapa são armazenados e podem ser recuperados para execuções futuras ou para as próximas etapas do fluxo, permitindo que você recupere erros, repita execuções e rastreie versões de modelos e suas dependências em várias execuções.

A proposta de valor do Metaflow é a simplicidade de sua biblioteca Python idiomática: integra-se totalmente à infraestrutura de compilação e tempo de execução para permitir a execução de tarefas de engenharia e ciência de dados em ambientes de produção locais e em escala. Neste momento, o Metaflow está fortemente integrado aos serviços da AWS, como o S3, para seu serviço de armazenamento de dados e funções de etapas para orquestração. Metaflow suporta R, além de Python. Seus principais recursos são de código aberto.

Se você estiver criando e implantando pipelines de processamento de dados e ML em produção na AWS, Metaflow é uma alternativa de framework leve e completo para plataformas mais complexas, como MLflow.

## 52. Micrometer

*Experimente*

Micrometer é uma biblioteca agnóstica de plataforma para instrumentação de métricas em JVM, que suporta Graphite, New Relic, CloudWatch e muitas outras integrações. Descobrimos que o Micrometer beneficia tanto autores de bibliotecas quanto equipes: autores podem incluir código de instrumentação de métricas em suas bibliotecas sem precisar dar suporte a todo e qualquer sistema de métricas que seus usuários estejam usando, enquanto as equipes podem oferecer suporte a muitas métricas diferentes em registros de back-end, habilitando as organizações a coletar métricas de maneira consistente.

## 53. NUKE

*Experimente*

NUKE é um sistema de construção para .NET e uma alternativa ao tradicional MSBuild, ou ao Cake e o Fake, que apresentamos anteriormente no Radar. NUKE representa as instruções de construção como uma DSL C#, facilitando o aprendizado e oferecendo bom suporte a IDE. Em nossa experiência, NUKE simplificou muito a construção de automação para projetos .NET. Gostamos das verificações e dicas precisas do código estático. Também gostamos do fato de podermos usar qualquer pacote NuGet de maneira fluida, e de que o código de automação pode ser compilado para evitar problemas em tempo de execução. O NUKE não é novo, mas sua abordagem inovadora — usando uma DSL C# — e nossa experiência geral positiva nos levaram a incluí-lo aqui.

## 54. Pactflow

*Experimente*

Usamos Pact para testes de contrato por tempo suficiente para observar um pouco da complexidade que vem com a escala. Alguns de nossos times usaram com sucesso o Pactflow para reduzir esse atrito. O Pactflow é executado tanto como software como serviço quanto como implantação local, com os mesmos recursos da oferta SaaS, e adicionando usabilidade, segurança e auditoria aprimoradas sobre a oferta Pact Broker de código aberto. Nosso uso até o momento é satisfatório, e estamos felizes em ver o esforço contínuo para remover parte da sobrecarga do gerenciamento de testes de contrato em escala.

## 55. Podman

*Experimente*

Uma alternativa ao Docker, Podman foi validado por muitos de nossos times. Podman apresenta um mecanismo sem daemon para gerenciar e executar contêineres, o que é uma abordagem interessante em comparação com o que o Docker faz. Além disso, Podman pode ser executado facilmente como um usuário normal sem exigir privilégios de root, o que reduz a superfície de ataque. Usando imagens da Open Container Initiative (OCI) criadas por Buildah ou imagens do Docker, Podman pode ser adaptado à maior parte dos casos de uso de contêineres. Com exceção de alguns problemas de compatibilidade com macOS, nossos times tiveram boas experiências com Podman em distribuições Linux.

## 56. Sourcegraph

*Experimente*

Na edição anterior do Radar, apresentamos duas ferramentas que pesquisam e substituem código usando uma representação de árvore de sintaxe abstrata (AST), **Comby** e **Sourcegraph**. Embora compartilhem algumas semelhanças, também diferem de várias maneiras. Sourcegraph é uma ferramenta comercial (com opção de uso gratuito para até 10 usuários). É particularmente adequada para pesquisa, navegação ou referência cruzada em grandes bases de código, com ênfase em uma experiência interativa de desenvolvimento. Por outro lado, Comby é uma ferramenta de linha de comando leve e de código aberto para automatizar tarefas repetitivas. Como Sourcegraph é um serviço hospedado, também tem a capacidade de monitorar continuamente as bases de código e enviar alertas quando ocorre uma correspondência. Agora que ganhamos mais experiência com Sourcegraph, decidimos movê-lo para o anel Experimente para refletir nossa experiência positiva — o que não significa que seja melhor que Comby. Cada ferramenta tem como foco um nicho diferente.

## 57. Syft

*Experimente*

Um dos elementos-chave para melhorar a "segurança da cadeia de fornecimento" é usar uma **lista de materiais de software (SBOM)**. Por isso, publicar uma SBOM junto com o artefato de software é cada vez mais importante. **Syft** é uma ferramenta CLI e biblioteca Go para gerar SBOMs a partir de imagens de contêiner e sistemas de arquivos. A SBOM pode ser gerada em vários formatos, incluindo JSON, **CycloneDX** e SPDX. A saída SBOM do Syft pode ser usada pelo **Grype** para verificação de vulnerabilidades. Uma forma de publicar a SBOM gerada junto com a imagem é adicioná-la como um atestado usando **Cosign**. Isso permite que os consumidores da imagem verifiquem a SBOM e a usem para análise adicional.

## 58. Volta

*Experimente*

Ao trabalhar com várias bases de código JavaScript ao mesmo tempo, muitas vezes é necessário usar diferentes versões do Node e de outras ferramentas JavaScript. Em máquinas de pessoas desenvolvedoras, essas ferramentas geralmente são instaladas na conta do usuário ou na própria máquina, o que significa que é necessária uma solução para alternar entre várias instalações. Para o Node em si existe o nvm, mas queremos destacar **Volta** como uma alternativa que estamos vendo em uso com nossos times. Volta tem várias vantagens em relação ao uso do nvm: pode gerenciar outras ferramentas JavaScript como Yarn; também conta com a noção de fixar uma versão da cadeia de ferramentas em uma base de projeto, o que significa que as pessoas desenvolvedoras podem simplesmente usar as ferramentas em um determinado diretório de código sem ter que se preocupar em alternar manualmente entre as versões da ferramenta — Volta simplesmente usa shims no caminho para selecionar a versão fixada. Escrito em Rust, Volta é rápido e vem como um único binário sem dependências.

## 59. Web Test Runner

*Experimente*

**Web Test Runner** é um pacote dentro do projeto **Modern Web**, que fornece várias ferramentas de alta qualidade para desenvolvimento web moderno com suporte para padrões web como ES Modules. O Web Test Runner é um executor de testes para aplicações web. Uma de suas vantagens em comparação com executores de teste existentes é a execução de testes no navegador (que pode ser headless). Suporta vários launchers de navegação - incluindo **Puppeteer**, **Playwright** e Selenium — e usa Mocha como padrão para framework de testes. Os testes são executados com muita rapidez, e gostamos da possibilidade de abrir uma janela do navegador com devtools durante a depuração.

O Web Test Runner usa internamente o Web Dev Server, que nos permite aproveitar sua excelente API de plugin para adicionar plugins personalizados para nosso conjunto de testes. As ferramentas de desenvolvimento do projeto Modern Web parecem ser um conjunto muito promissor, e já estamos usando em alguns projetos.

## 60. CDKTF

*Avalie*

Até este momento, muitas organizações criaram amplas estruturas de serviços na nuvem. Evidentemente, isso só é possível ao usar infraestrutura como código e ferramentas maduras. Ainda gostamos do Terraform, principalmente por seu rico e crescente ecossistema. No entanto, a falta de abstrações em HCL, a linguagem de configuração padrão do Terraform, cria efetivamente um teto de vidro. Usar Terragrunt contribui um pouco mais para esse cenário, mas cada vez mais nossos times se veem ansiosos pelas abstrações oferecidas pelas linguagens de programação modernas. Cloud Development Kit for Terraform (CDKTF), que resultou de uma colaboração entre o time do CDK da AWS e a Hashicorp, possibilita que os times usem várias linguagens de programação, incluindo TypeScript e Java, para definir e provisionar infraestrutura. Essa abordagem segue o caminho liderado por Pulumi, mantendo-se no ecossistema Terraform. Tivemos boas experiências com o CDKTF, mas decidimos mantê-lo no anel Avalie até que saia da versão beta.

## 61. Chrome Recorder panel

*Avalie*

Chrome Recorder panel é um recurso de visualização do Google Chrome 97 que permite o registro e a reprodução simples das jornadas de usuário. Embora essa definitivamente não seja uma ideia nova, a maneira como o recurso é integrado ao Chrome possibilita a criação, edição e execução rápidas de scripts. Também se integra perfeitamente ao painel de desempenho, o que facilita a obtenção de feedback consistente e constante sobre o desempenho da página. Embora o teste de estilo de gravação/reprodução sempre deva ser usado com cuidado para evitar testes frágeis, achamos que vale a pena avaliar esse recurso de visualização, especialmente se você já estiver usando o painel Chrome Performance para medir o desempenho de suas páginas.

## 62. Excalidraw

*Avalie*

Excalidraw é uma ferramenta de desenho online simples, mas poderosa, que nossos times gostam de usar. Às vezes, as equipes precisam apenas de uma imagem rápida em vez de um diagrama formal. Para times remotos, o Excalidraw oferece uma maneira rápida de criar e compartilhar diagramas. Nossos times também gostam do visual "lo-fi" dos diagramas que podem produzir, que lembra a aparência dos diagramas que seriam criados em um quadro branco se o time estivesse reunido presencialmente. Uma ressalva: você precisa prestar atenção aos padrões de segurança. Atualmente, qualquer pessoa que tenha o link pode ver o diagrama. A versão paga fornece autenticação adicional.

## 63. GitHub Codespaces

*Avalie*

GitHub Codespaces permite que pessoas desenvolvedoras criem ambientes de desenvolvimento na nuvem e os acessem por meio de um IDE como se o ambiente fosse local. O GitHub não é a primeira empresa a implementar essa ideia; já falamos anteriormente sobre o Gitpod. Gostamos do fato de o Codespaces permitir que os ambientes sejam padronizados usando a configuração de dotfiles, tornando mais rápido o onboarding de novos membros no time, além de oferecer máquinas virtuais com até 32 núcleos e 64 GB de memória. Essas VMs podem ser ativadas em menos de dez

segundos, oferecendo potencialmente ambientes mais poderosos do que o laptop de uma pessoa desenvolvedora.

## 64. GoReleaser

*Avalie*

GoReleaser é uma ferramenta que automatiza o processo de construção e lançamento de um projeto Go para diferentes arquiteturas por meio de vários repositórios e canais, uma necessidade comum para projetos Go direcionados a diferentes plataformas. Você executa a ferramenta a partir de sua máquina local ou via CI, com a ferramenta disponível por meio de vários serviços CI, minimizando assim a configuração e a manutenção. O GoReleaser cuida da compilação, empacotamento, publicação e anúncio de cada lançamento, além de oferecer suporte a diferentes combinações de formato de pacotes, repositório de pacotes e controle de versão. Embora já exista há alguns anos, nos surpreende que mais times não estejam usando. Se você precisa lançar regularmente uma base de código Go, vale a pena avaliar essa ferramenta.

## 65. Grype

*Avalie*

Proteger a cadeia de fornecimento de software tornou-se uma preocupação comum entre os times de entrega, o que se reflete no número crescente de novas ferramentas nesse espaço. Grype é uma nova ferramenta leve de verificação de vulnerabilidades para imagens Docker e OCI. Pode ser instalado como um binário, verificar imagens antes de serem enviadas por push para um registro e não requer que um daemon do Docker seja executado em seus agentes de compilação. Grype vem da mesma equipe por trás do Syft, que gera listas de materiais de software (SBOMs) em vários formatos a partir de imagens de contêineres. Grype pode consumir a saída SBOM do Syft para verificar vulnerabilidades.

## 66. Infracost

*Avalie*

Uma vantagem de migrar para a nuvem frequentemente citada é a transparência em relação aos gastos com infraestrutura. Em nossa experiência, isso muitas vezes não é o caso. Os times nem sempre pensam nas decisões que tomam em relação à infraestrutura em termos de custo financeiro, e é por isso que mencionamos anteriormente o custo de execução como função de aptidão arquitetural. O lançamento de Infracost, uma nova ferramenta que visa tornar as compensações de custo visíveis nas pull requests do Terraform, nos intrigou. É um software de código aberto e está disponível para macOS, Linux, Windows e Docker, oferecendo suporte a precificação para AWS, GCP e Microsoft Azure prontos para uso. Também fornece uma API pública que pode ser consultada para dados de custo atuais. Nossos times estão entusiasmados com seu potencial, especialmente quando se trata de obter melhor visibilidade de custos no IDE.

## 67. jc

*Avalie*

Na edição anterior do Radar, colocamos comandos modernos do Unix no anel Avalie. Um dos comandos apresentados nessa coleção de ferramentas foi jq, efetivamente um sed para JSON. jc executa uma tarefa relacionada: pega a saída de comandos Unix comuns e analisa a saída em JSON. Os dois comandos juntos fornecem uma ponte entre o mundo Unix CLI e o conjunto de bibliotecas e ferramentas que operam em JSON. Ao escrever scripts *simples*, por exemplo, para implantação de software ou coleta de informações de solução de problemas, ter uma infinidade de diferentes formatos de saída de comando Unix mapeados em JSON bem definido pode economizar muito

tempo e esforço. Assim como acontece com jq, você precisa ter certeza de que o comando está disponível. jc pode ser instalado a partir de muitos dos repositórios de pacotes conhecidos.

## 68. skopeo

*Avalie*

skopeo é um utilitário de linha de comando que executa várias operações em imagens de contêiner e repositórios de imagens. Não requer um usuário root para a maioria de suas operações e não requer que um daemon esteja em execução. É uma parte útil de um pipeline de CI. Nós o usamos para copiar imagens de um registro para outro enquanto promovemos as imagens. É melhor do que fazer um pull e um push, pois não precisamos armazenar as imagens localmente. Não é uma ferramenta nova, mas é tão útil e subutilizada que achamos válido destacá-la.

## 69. SQLFluff

*Avalie*

Embora *linting* seja uma prática antiga no mundo do software, sua adoção é mais lenta no mundo dos dados. SQLFluff é um linter SQL de dialeto cruzado escrito em Python que vem com uma interface de linha de comando simples (CLI), facilitando a incorporação em um pipeline CI/CD. Se você estiver confortável com as convenções padrão, o SQLFluff funcionará sem nenhuma configuração adicional após instalá-lo e aplicará um conjunto de padrões de formatação fortemente opinativos. Para definir suas próprias convenções, é necessário adicionar um dotfile de configuração. A CLI pode corrigir automaticamente determinadas classes de violações que envolvem questões de formatação, como espaços em branco ou letras maiúsculas de palavras-chave. SQLFluff ainda é novo, mas estamos otimistas em ver o SQL recebendo alguma atenção no mundo do *linting*.

## 70. Terraform Validator

*Avalie*

As organizações que adotaram infraestrutura como código e plataformas de infraestrutura de autoatendimento estão procurando maneiras de dar aos times o máximo de autonomia enquanto ainda aplicando boas práticas de segurança e políticas organizacionais. Já destacamos tfsec antes e o estamos movendo para o anel Adote nesta edição do Radar. Para times que trabalham na GCP, o Terraform Validator pode ser uma opção para criar uma biblioteca de políticas, ou seja, um conjunto de restrições que são verificadas nas configurações do Terraform.

## 71. Typesense

*Avalie*

Typesense é um mecanismo de pesquisa de texto rápido e tolerante a erros de digitação. Para casos de uso que envolvem grandes volumes de dados, Elasticsearch ainda pode ser uma boa opção, pois fornece uma solução de pesquisa baseada em disco escalável horizontalmente. No entanto, se você estiver criando uma aplicação de pesquisa sensível à latência com um tamanho de índice de pesquisa que caiba na memória, Typesense é uma alternativa poderosa e mais uma opção para ser avaliada ao lado de ferramentas como Meilisearch.

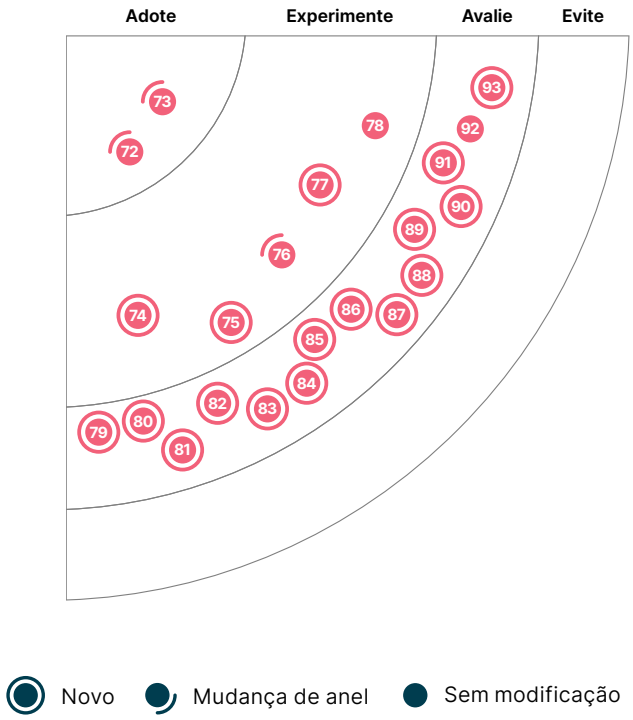
# Linguagens e frameworks

**Adote**
72. SwiftUI
73. Testcontainers

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

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

**Evite**
—



## 72. SwiftUI

*Adote*

Quando a Apple lançou o **SwiftUI** alguns anos atrás, foi um grande passo para implementação de interfaces de usuário em todos os tipos de dispositivos fabricados pela Apple. Desde o início, gostamos da abordagem declarativa e centrada no código, além do modelo de programação reativa fornecido pelo **Combine**. Percebemos, no entanto, que escrever muitos testes de visualização, o que ainda é necessário com um padrão de modelo-visualização-modelo de visualização (MVVM), não era realmente sensato com o framework de automação XCUITest fornecido pela Apple. Essa lacuna foi fechada pelo **ViewInspector**. Um obstáculo final foi a versão mínima do sistema operacional necessária. No momento do lançamento, apenas as versões mais recentes do iOS e do macOS podiam executar aplicativos escritos com SwiftUI, mas devido à cadência regular de atualizações da Apple, os aplicativos SwiftUI agora podem ser executados em praticamente todas as versões do macOS e do iOS que recebem atualizações de segurança.

## 73. Testcontainers

*Adote*

Tivemos experiências suficientes com **Testcontainers** para considerarmos uma opção padrão útil na criação de um ambiente confiável para a execução de testes. É uma biblioteca, portada para **várias linguagens**, que "dockeriza" dependências de teste comuns — incluindo vários tipos de bancos de dados, tecnologias de enfileiramento, serviços em nuvem e dependências de teste de interface do usuário, como navegadores web — com a capacidade para executar Dockerfiles personalizados quando necessário. Funciona bem com frameworks de teste como JUnit, é flexível o suficiente para permitir que os usuários gerenciem o ciclo de vida do contêiner e a rede avançada, além de configurar rapidamente um ambiente de testes integrado. Nossos times consideraram essa biblioteca de contêineres programáveis, leves e descartáveis consistentemente capaz de tornar os testes funcionais mais confiáveis.

## 74. Bob

*Experimente*

Ao desenvolver um aplicativo com React Native, às vezes você precisa criar seus próprios módulos. Encontramos essa necessidade, por exemplo, ao construir uma biblioteca de componentes de interface do usuário para um aplicativo React Native. Criar um projeto de módulo desse tipo não é simples, e nossos times têm relatado sucesso usando **Bob** para automatizar essa tarefa. Bob fornece uma CLI para criar o scaffolding para diferentes destinos. O scaffolding não se limita à funcionalidade principal, mas, opcionalmente, pode incluir código de exemplo, linters, configuração de pipeline de compilação e outros recursos.

## 75. Flutter-Unity widget

*Experimente*

Flutter é cada vez mais popular para criar aplicativos móveis multiplataforma, enquanto Unity é uma ótima opção para a criação de experiências de AR/VR. Uma peça chave no quebra-cabeça para integrar Unity e Flutter é o **Widget Flutter-Unity**, que permite incorporar aplicativos Unity dentro de widgets do Flutter. Um dos principais recursos que o widget oferece é a comunicação bidirecional entre Flutter e Unity. Descobrimos que seu desempenho também é muito bom e estamos otimistas com a perspectiva de usar o Unity em mais aplicativos Flutter.

## 76. Kotest

*Experimente*

Kotest (anteriormente KotlinTest) é uma ferramenta de testes independente para Kotlin que continua ganhando força dentro de nossos times em várias implementações Kotlin — nativas, JVM ou JavaScript. As principais vantagens são o fato de oferecer uma variedade de estilos de teste para estruturar as suítes, além de disponibilizar um conjunto abrangente de matchers que permite testes expressivos em uma DSL interna elegante. Somando-se à sua compatibilidade com testes baseados em propriedades — uma técnica que destacamos anteriormente no Radar — nossos times apreciam o sólido plugin IntelliJ e a crescente comunidade de suporte.

## 77. Swift Package Manager

*Experimente*

Algumas linguagens de programação, especialmente as mais recentes, possuem uma solução de gerenciamento de pacotes e dependências incorporada. Quando foi lançado em 2014, o Swift não vinha com um gerenciador de pacotes e, portanto, a comunidade de pessoas desenvolvedoras de macOS e iOS simplesmente continuou usando CocoaPods e Carthage, soluções de terceiros criadas para Objective-C. Alguns anos depois, Swift Package Manager (SwiftPM) foi iniciado como um projeto de código aberto oficial da Apple e alguns anos se passaram até a Apple adicionar suporte ao Xcode. Mesmo atualmente, entretanto, muitos times de desenvolvimento continuaram usando CocoaPods e Carthage, principalmente porque muitos pacotes simplesmente não estavam disponíveis via SwiftPM. Agora que a maior parte dos pacotes pode ser incluída via SwiftPM e os processos foram ainda mais simplificados para criadores e consumidores, nossos times têm contado cada vez com o SwiftPM.

## 78. Vowpal Wabbit

*Experimente*

Vowpal Wabbit é uma biblioteca de aprendizado de máquina de uso geral. Originalmente criado no Yahoo! Research há mais de uma década, Vowpal Wabbit continua a implementar novos algoritmos de aprendizado por reforço. Gostaríamos de destacar o Vowpal Wabbit 9.0, um lançamento significativo após seis anos, e incentivar você a planejar sua migração, já que a nova versão possui várias melhorias de usabilidade, novas reduções e correções de bugs.

## 79. Android Gradle plugin - Kotlin DSL

*Avalie*

O Android Gradle plugin Kotlin DSL adicionou suporte para Kotlin Script como uma alternativa aos scripts de compilação Groovy for Gradle. O objetivo de substituir o Groovy pelo Kotlin é fornecer melhor suporte para refatoração e edição mais simples em IDEs, bem como produzir código mais fácil de ler e manter. Para times que já usam Kotlin, isso também significa trabalhar na compilação em uma linguagem familiar. Um de nossos times migrou um script de compilação de 450 linhas de pelo menos sete anos de existência em poucos dias. Se você tiver scripts de compilação gradle grandes ou complexos, vale avaliar se o Kotlin Script produzirá melhores resultados para seus times.

## 80. Azure Bicep

*Avalie*

Para quem prefere uma linguagem mais natural do que JSON para código de infraestrutura, Azure Bicep é uma linguagem específica de domínio (DSL) que usa uma sintaxe declarativa e suporta modelos parametrizados reutilizáveis para definições de recursos modulares. Uma extensão Visual Studio Code fornece segurança de tipo instantânea, IntelliSense e verificação de sintaxe,

e o compilador permite transpilação bidirecional de/para modelos ARM. O DSL orientado a recursos da Bicep e a integração nativa com o ecossistema Azure o tornam uma opção atraente para o desenvolvimento de infraestrutura do Azure.

## 81. Capacitor

*Avalie*

Estamos debatendo o valor das ferramentas de desenvolvimento móvel multiplataforma há quase tanto tempo quanto publicamos o Technology Radar. Observamos pela primeira vez uma nova geração de ferramentas em 2011 ao falar sobre **plataformas cross-mobile**. Apesar de nosso ceticismo inicial, essas ferramentas foram aperfeiçoadas e amplamente adotadas ao longo dos anos. E ninguém pode questionar a popularidade duradoura e a utilidade do **React Native**. **Capacitor** é a última geração de uma linha de ferramentas que começou com PhoneGap, depois renomeada para **Apache Cordova**. Capacitor é uma reescrita completa do Ionic que adota o estilo **progressive web app** para aplicações independentes. Até o momento, nossos times de desenvolvimento gostam do fato de poderem abordar aplicações web, iOS e Android com uma única base de código, gerenciando as plataformas nativas separadamente com acesso às APIs nativas quando necessário. Capacitor surge como uma alternativa ao React Native, que por sua vez conta com muitos anos de experiência multiplataforma por trás.

## 82. Java 17

*Avalie*

Normalmente não incluímos novas versões de linguagens, mas queríamos destacar a nova versão de suporte de longo prazo (LTS) do Java, versão 17. Embora existam novos recursos promissores, como a visualização de **correspondência de padrões**, é a mudança para o novo processo LTS que deve interessar a muitos negócios. Recomendamos que as organizações avaliem as novas versões do Java à medida que forem disponibilizadas, certificando-se de adotar novos recursos e versões conforme apropriado. Surpreendentemente, muitas organizações não adotam rotineiramente versões mais recentes de linguagens, embora as atualizações regulares ajudem a manter as coisas reduzidas e gerenciáveis. Esperamos que o novo processo LTS, juntamente com migração de organizações para atualizações regulares, ajude a evitar a armadilha da mentalidade "é muito caro atualizar", que tem como resultado o software de produção sendo executado em uma versão do Java em fim de vida útil.

## 83. Jetpack Glance

*Avalie*

O Android 12 trouxe mudanças significativas nos widgets de aplicativos, melhorando a experiência do uso e de desenvolvimento. Para escrever aplicativos Android comuns, expressamos nossa preferência pelo **Jetpack Compose** como uma maneira moderna de criar interfaces de usuário nativas. Agora, com o **Jetpack Glance**, construído em cima do tempo de execução do Compose, as pessoas desenvolvedoras podem usar APIs Kotlin declarativas semelhantes para escrever widgets. Recentemente, o suporte do Glance foi **estendido** ao Tiles for Wear OS.

## 84. Jetpack Media3

*Avalie*

O Android possui hoje várias APIs de mídia: Jetpack Media, também conhecido como MediaCompat, Jetpack Media2 e ExoPlayer. Infelizmente, essas bibliotecas foram desenvolvidas de forma independente, com objetivos diferentes, mas funcionalidades sobrepostas. As pessoas desenvolvedoras de Android não apenas tiveram que escolher qual biblioteca usar, como também tiveram que lidar com a programação de adaptadores ou outro código de conexão quando os

recursos de várias APIs eram necessários. Jetpack Media3 é um esforço, disponível atualmente para acesso antecipado, para criar uma nova API que utiliza áreas comuns de funcionalidade das APIs existentes — incluindo interface de usuário, reprodução e manipulação de sessão de mídia — combinando-as em uma API mesclada e refinada. A interface do player do ExoPlayer também foi atualizada, aprimorada e simplificada para atuar como a interface do player comum para o Media3.

## 85. MistQL

*Avalie*

MistQL é uma linguagem pequena e específica de domínio para realizar cálculos em estruturas semelhantes a JSON. Originalmente desenvolvida para extração manual de recursos de modelos de aprendizado de máquina no frontend, MistQL atualmente oferece suporte a uma implementação de JavaScript para navegadores e uma implementação de Python para casos de uso do lado do servidor. Gostamos bastante de sua sintaxe funcional limpa e composta, e incentivamos você a avaliá-la com base em suas necessidades.

## 86. npm workspaces

*Avalie*

Embora muitas ferramentas ofereçam suporte ao desenvolvimento de vários pacotes no universo node.js, o npm 7 adiciona suporte direto com a adição do npm workspaces. Gerenciar pacotes relacionados de forma conjunta facilita o desenvolvimento, permitindo, por exemplo, armazenar várias bibliotecas relacionadas em um único repositório. Com o npm workspaces, uma vez que você adiciona uma configuração em um arquivo package.json de nível superior para fazer referência a um ou mais arquivos package.json aninhados, comandos como **`npm install`** funcionarão em vários pacotes, vinculando os pacotes de origem dependentes ao diretório root node_module. Outros comandos npm também estão adaptados aos espaços de trabalho, permitindo, por exemplo, executar **`npm run`** e **`npm test`** em vários pacotes com um único comando. Ter essa flexibilidade pronta para uso diminui a necessidade de alguns times de buscar outro gerenciador de pacotes.

## 87. Remix

*Avalie*

Acompanhamos a migração da renderização de sites do lado do servidor para as aplicações de página única (SPAs) no navegador, e agora o pêndulo do desenvolvimento web parece retornar ao meio. Remix é um exemplo. É um framework JavaScript full-stack que fornece carregamentos de página rápidos, aproveitando sistemas distribuídos e navegadores nativos, em vez de compilações estáticas desajeitadas. Algumas otimizações foram feitas no roteamento aninhado e no carregamento de página, o que torna a renderização da página especialmente rápida. Muitas pessoas vão comparar o Remix com o Next.js, que está posicionado de forma semelhante. Estamos felizes em ver esses frameworks combinando habilmente o tempo de execução do navegador com o tempo de execução do servidor para fornecer uma melhor experiência de uso.

## 88. ShedLock

*Avalie*

Executar uma tarefa agendada uma vez, apenas uma vez, em um cluster de processadores distribuídos é um requisito relativamente comum. Por exemplo, a situação pode surgir durante a ingestão de um lote de dados, envio de uma notificação ou executar alguma atividade de limpeza regular. Mas este é um problema notoriamente desafiador. Como um grupo de processos coopera de forma confiável em redes lentas e menos confiáveis? Algum tipo de mecanismo de bloqueio é necessário para coordenar ações em todo o cluster. Felizmente, vários armazenadores distribuídos

podem implementar um bloqueio. Sistemas como ZooKeeper e Consul, bem como bancos de dados como DynamoDB ou Couchbase, têm os mecanismos subjacentes necessários para gerenciar o consenso em todo o cluster. ShedLock é uma pequena biblioteca usada para aproveitar esses provedores em seu próprio código Java, caso você deseje implementar suas próprias tarefas agendadas. Fornece uma API para adquirir e liberar bloqueios, bem como conectores para uma ampla variedade de provedores. Se você está escrevendo suas próprias tarefas distribuídas, mas não quer assumir a complexidade de uma plataforma de orquestração inteira como Kubernetes, vale a pena dar uma olhada no ShedLock.

## 89. SpiceDB

*Avalie*

SpiceDB é um sistema de banco de dados inspirado no Zanzibar, do Google, para gerenciar permissões de aplicativos. Com SpiceDB, você cria um esquema para modelar os requisitos de permissões e usa a biblioteca cliente para aplicar o esquema a um dos bancos de dados suportados, inserir dados e realizar consultas para responder com eficiência a perguntas como "este usuário tem acesso a este recurso?" ou até mesmo o inverso "quais são os recursos aos quais esse usuário tem acesso?" Normalmente, defendemos a separação das políticas de autorização do código, mas o SpiceDB dá um passo adiante, separando os dados da política e armazenando-os como um grafo para responder com eficiência às consultas de autorização. Devido a essa separação, você deve assegurar que as alterações no armazenamento de dados primário de seu aplicativo sejam refletidas no SpiceDB. Entre outras implementações inspiradas em Zanzibar, achamos que o SpiceDB é um framework interessante para avaliar suas necessidades de autorização.

## 90. sqlc

*Avalie*

sqlc é um compilador que gera código Go idiomático de tipo seguro a partir do SQL. Ao contrário de outras abordagens baseadas em mapeamento relacional de objeto (ORM), você continua escrevendo SQL simples para suas necessidades. Uma vez invocado, o sqlc verifica a correção do SQL e gera um código Go de alto desempenho, que pode ser chamado diretamente do restante do aplicativo. Com suporte estável para PostgreSQL e MySQL, achamos que vale a pena avaliar o sqlc.

## 91. The Composable Architecture

*Avalie*

O desenvolvimento de aplicativos para iOS tornou-se mais simples ao longo do tempo, e a mudança do SwiftUI para o anel Adote é um sinal disso. Indo além da natureza geral do SwiftUI e outros frameworks comuns, The Composable Architecture (TCA) é uma biblioteca e um estilo arquitetural para criação de aplicativos. Foi projetado ao longo de uma série de vídeos, e o time responsável afirmou ter em mente composição, testes e ergonomia, baseando-se em ideias do The Elm Architecture e do Redux. Como esperado, o escopo estreito e opinativo é tanto um ponto forte quanto um ponto fraco do TCA. Acreditamos que os times que não têm muita experiência em escrever aplicativos para iOS – geralmente equipes que estão cuidando de várias bases de código relacionadas com diferentes pilhas de tecnologia –, são as que mais se beneficiam do uso de um framework opinativo como o TCA. Nós gostamos de como as opiniões são expressas no TCA.

## 92. WebAssembly

*Avalie*

WebAssembly (WASM) é o padrão W3C que fornece recursos de execução de código no navegador. Suportado por todos os principais navegadores e compatível com versões anteriores, é um formato

de compilação binária projetado para ser executado no navegador em velocidades quase nativas. WebAssembly abre a gama de linguagens que você pode usar para escrever funcionalidades de front-end, com foco inicial em C, C++ e Rust, além de ser um objetivo de **compilação LLVM**. Quando executado no sandbox, pode interagir com JavaScript e compartilhar as mesmas permissões e modelo de segurança. Portabilidade e segurança são os principais recursos que habilitarão a maioria das plataformas, incluindo dispositivos móveis e IoT.

## 93. Zig

*Avalie*

Zig é uma nova linguagem que compartilha muitos atributos com C, mas com tipagem mais forte, alocação de memória facilitada, suporte para namespaces e uma série de outros recursos. Sua sintaxe, no entanto, é uma reminiscência de JavaScript e não do C, o que pode pesar contra para algumas pessoas. O objetivo do Zig é fornecer uma linguagem muito simples com compilação direta, minimizando os efeitos colaterais e fornecendo uma execução previsível e fácil de rastrear. O Zig também oferece acesso simplificado à **capacidade de compilação cruzada** do LLVM. Algumas de nossas pessoas desenvolvedoras acharam esse recurso tão viável que estão usando o Zig como um compilador cruzado, embora não estejam escrevendo código Zig. Zig é uma linguagem nova e vale a pena considerar para aplicações nas quais C está sendo considerado ou já em uso, bem como para aplicações de sistemas de baixo nível que requerem manipulação explícita de memória.

## Quer continuar se atualizando com artigos e informações relacionadas ao Radar?

Siga nossos perfis nas redes sociais e inscreva-se gratuitamente para se tornar assinante.

Assine

A Thoughtworks é uma consultoria global de tecnologia que integra estratégia, design e engenharia de software para alavancar a inovação digital. Somos mais de 10 mil pessoas distribuídas entre 49 escritórios e em 17 países. Há mais de 25 anos, trabalhamos junto a nossas clientes para criar impacto extraordinário, usando a tecnologia como diferenciador para ajudá-las a resolver problemas de negócio complexos.

/thoughtworks