Technology Radar

Um guia com opiniões firmes sobre as fronteiras da tecnologia

/thoughtworks

Volume 27

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, ferramentas, plataformas, 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. Usamos quando apropriado em nossos projetos.

Experimente: Vale a pena 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 17 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. Esta edição do Technology Radar da Thoughtworks foi baseada em uma reunião do TAB em Barcelona, realizada em setembro de 2022.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Falconi Crispim

Erik Dörnenburg

Fausto de la Torre

Hao Xu

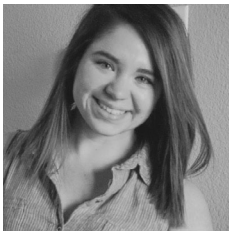
Ian Cartwright

James Lewis

Marisa Hoenig

Mike Mason

Neal Ford



Perla Villarreal

Scott Shaw

Shangqi Liu

Tradução: Paula Ribas, Thayse Onofrio, Pietra Freitas, Thiago Gregorio, Luiza Sousa, Guilherme Silveira, Beatriz Vilalta, Mayara Silva Ferreira, Marcelo Aquino, Mateus Resende, Akina Kurita e Camilla Falconi Crispim.

Temas

A popularização do aprendizado de máquina

O aprendizado de máquina (ML) já foi um domínio reservado apenas para alguns indivíduos privilegiados que tinham ferramentas e recursos para criar coisas legais. Felizmente, vemos uma integração gradual do ML à medida que o poder computacional cresce em dispositivos de todos os tamanhos, ferramentas de código aberto chegam e um maior rigor em requisitos e conscientização sobre privacidade e informações personalizadas convergem para criar um ecossistema florescente. Técnicas como **aprendizado de máquina federado** permitem modelos de ML que fornecem privacidade para informações confidenciais. O campo de **TinyML** permite que os modelos sejam executados em dispositivos com recursos limitados, movendo a inferência para a borda, o que libera recursos e melhora a privacidade de dados confidenciais. **Feature Stores** fornecem benefícios análogos ao padrão de design Model-View-Controller para desenvolvimento de aplicações, permitindo uma separação mais clara de preocupações entre curadoria de dados, treinamento de modelo e inferência. Modelos disponíveis publicamente, como **Stable Diffusion** destacam os incríveis recursos do aprendizado de máquina e as preocupações com dados de origem e ética. Os componentes de ML também estão mais fáceis de conectar do que nunca, possibilitando a criação de experiências e soluções de ML com composição criativa de modelos de negócios personalizados e modelos genéricos altamente capazes. Aplaudimos os novos recursos neste espaço e aguardamos ansiosamente os avanços futuros.

O poder das plataformas como produto

A palavra “plataforma” continua sendo uma das mais usadas em nossas reuniões do Radar, porque o conceito está bastante difundido na indústria. A palavra aparece em muitas formas diferentes, incluindo plataformas com foco em negócio ou domínio, mas também plataformas de infraestrutura ou experiência das pessoas desenvolvedoras. Fundamentalmente, a causa raiz de muitos dos problemas e decepções que as organizações experimentam com todas as plataformas é a falha em tratá-las adequadamente como produtos. Por exemplo, muitas plataformas destinadas a serem consumidas por pessoas desenvolvedoras carecem da pesquisa de usuário e da análise contextual que esperamos para outros tipos de produtos. Quem detém a propriedade da plataforma precisa validar suas suposições sobre as necessidades das pessoas desenvolvedoras e responder aos padrões reais de uso. E como qualquer bom produto, uma plataforma precisa de suporte contínuo. Deve evoluir e se adaptar em resposta às necessidades de mudança das pessoas desenvolvedoras. Além disso, papéis como gerentes de projeto e analistas de negócios geralmente têm escopos diferentes em relação a aplicações tradicionais. A metáfora da “plataforma como produto” só funciona quando totalmente adotada como uma prática e não como uma frase da moda.

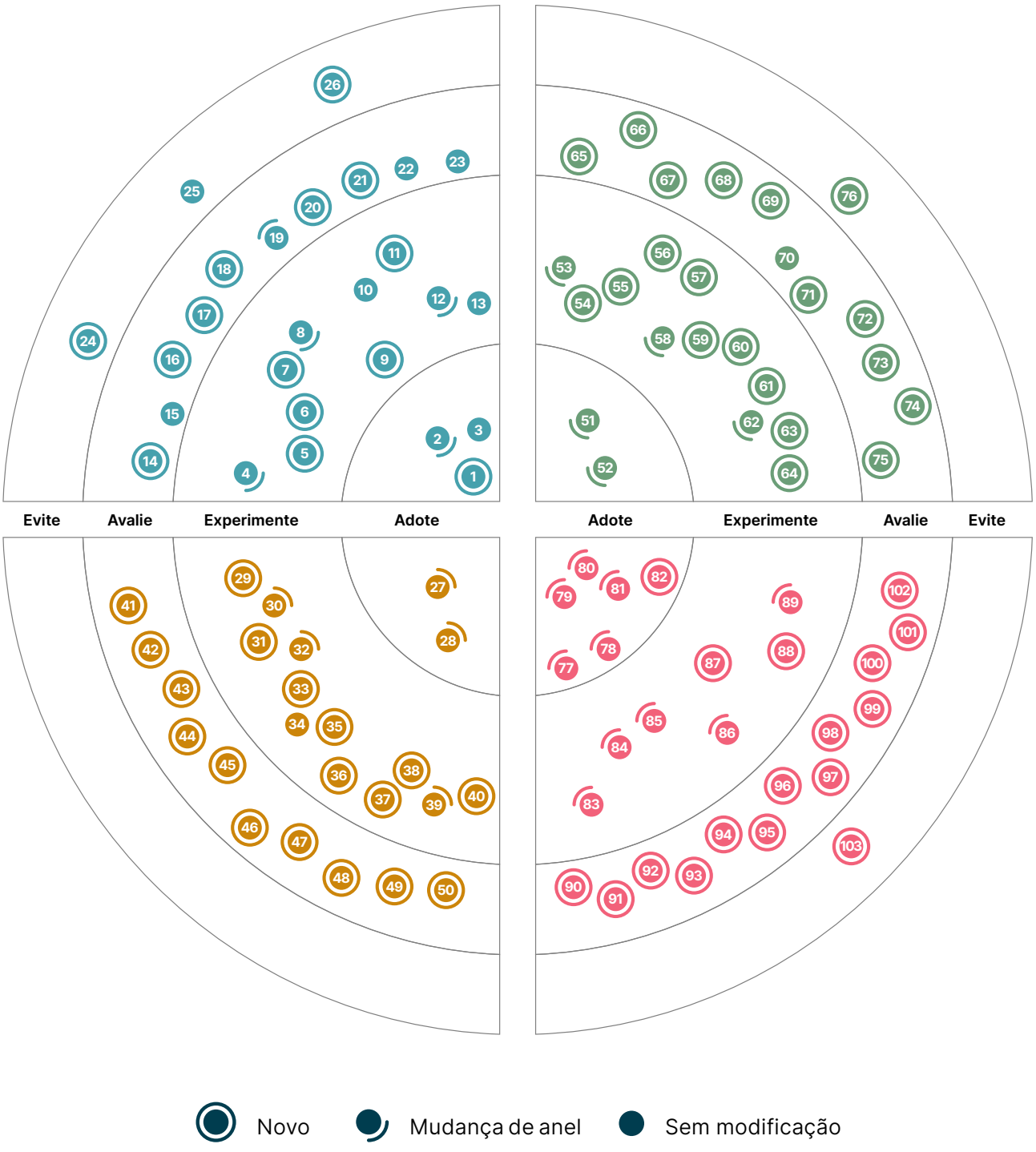
Movendo a propriedade dos dados para as bordas

Como dolorosamente sabemos, a centralização de qualquer tipo abre a possibilidade de constrição, gargalos ou exposição desnecessária. Por isso, nós nos esforçamos constantemente para encontrar novas maneiras de quebrar pontos de acoplamento centralizados, o que destacamos em vários blips em nosso Radar. Com base em pesquisas sobre tipos de dados replicados sem conflito (CRDTs), que permitem aplicações baseadas em dados sem um banco de dados centralizado, a técnica de software/aplicações locais-primeiro incentiva as pessoas desenvolvedoras a considerar construir em torno de dados ponto a ponto em vez de usar um banco de dados centralizado. Mover a propriedade dos dados para as bordas também permite que as pessoas desenvolvedoras aproveitem os recursos aprimorados nos dispositivos, conforme mostramos no tema *A popularização do aprendizado de máquina*. Por exemplo, muitos recursos, como reconhecimento facial, podem ocorrer na borda, mantendo os dados subjacentes no dispositivo para sempre.

O desenvolvimento móvel também deve ser modular

As pessoas engenheiras de software aprenderam o valor de estruturar a arquitetura de um aplicativo principalmente em torno de conceitos de domínio e funcionalidade de negócio. Preocupações técnicas – uma separação entre a interface de usuário e a lógica de domínio – ainda são importantes, mas desempenham um papel secundário. À medida que os aplicativos móveis amadurecem, geralmente ficam maiores, às vezes se tornando o que chamamos de super aplicativos, que abrangem muitos serviços e podem ser vistos como plataformas por si só. Os aplicativos que não são tão grandes, mas que adquiriram muitos recursos ao longo dos anos, geralmente também podem ser decompostos em módulos, e as empresas vêm descobrindo que os aplicativos móveis se beneficiam da mesma abordagem de modularidade. Aplicativos modulares podem ser desenvolvidos por vários times autônomos, o que traz muitos benefícios bem documentados. Além da complexidade, há a necessidade de implantação por meio de uma loja de aplicativos e a necessidade de oferecer suporte a versões nativas de iOS e Android, além de uma versão baseada na web, com mudanças sutis para acomodar cada uma. Vemos um melhor suporte de frameworks para as tensões únicas inerentes ao desenvolvimento móvel, mas de forma geral, apesar dos benefícios, muitas organizações enfrentam dificuldades para trazer uma abordagem modular ao desenvolvimento móvel.

O Radar



O Radar

Técnicas

Adote

1. Mapeamento do caminho para produção
2. Carga cognitiva do time
3. Modelagem de ameaças

Experimente

4. BERT
5. Testes de regressão visual de componentes
6. Tokens de design
7. Servidor SMTP falso para testar o envio de e-mails
8. Aprendizado de máquina federado
9. Plataforma de desenvolvimento incemental
10. Micro frontends para aplicativos móveis
11. Observabilidade para pipelines de CI/CD
12. SLSA
13. Lista de materiais de software

Avalie

14. Eficiência de carbono como uma característica arquitetural
15. CUPID
16. GitHub push protection
17. Aplicação local-first
18. Repositório de métricas
19. Interface de usuário orientada a servidor
20. SLIs e SLOs como código
21. Dados sintéticos para testes de modelos
22. TinyML
23. Credenciais verificáveis

Evite

24. Profissionais satélite sem formas de trabalho “nativamente remotas”
25. SPAs por padrão
26. Nativo de nuvem superficial

Plataformas

Adote

27. Backstage
28. Delta Lake

Experimente

29. AWS Database Migration Service
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

Avalie

41. Bun
42. Databricks Unity Catalog
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

Evite

—

O Radar

Ferramentas

Adote

51. Great Expectations
52. k6

Experimente

53. Apache Superset
54. AWS Backup Vault Lock
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto Query Language
62. Spectral
63. Serviço de Autorização Declarativa da Styra
64. xbar para monitoramento de compilação

Avalie

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness Cloud Cost Management
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

Evite

76. Serviços online para formatação ou análise de código

Linguagens e Frameworks

Adote

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

Experimente

83. Azure Bicep
84. Camunda
85. Gradle Kotlin DSL
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

Avalie

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. Cross device SDK
97. Cypress Component Testing
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. Synthetic Data Vault

Evite

103. Carbon

Técnicas

Adote

1. Mapeamento do caminho para produção
2. Carga cognitiva do time
3. Modelagem de ameaças

Experimente

4. BERT
5. Testes de regressão visual de componentes
6. Tokens de design
7. Servidor SMTP falso para testar o envio de e-mails
8. Aprendizado de máquina federado
9. Plataforma de desenvolvimento incemental
10. Micro frontends para aplicativos móveis
11. Observabilidade para pipelines de CI/CD
12. SLSA
13. Lista de materiais de software

Avalie

14. Eficiência de carbono como uma característica arquitetural
15. CUPID
16. GitHub push protection
17. Aplicação local-first
18. Repositório de métricas
19. Interface de usuário orientada a servidor
20. SLIs e SLOs como código
21. Dados sintéticos para testes de modelos
22. TinyML
23. Credenciais verificáveis

Evite

24. Profissionais satélite sem formas de trabalho “nativamente remotas”
25. SPAs por padrão
26. Nativo de nuvem superficial

1. Mapeamento do caminho para produção

Adote

Embora o mapeamento do caminho para produção tenha sido uma prática quase universal na Thoughtworks desde a programação da ***Entrega Contínua***, muitas vezes encontramos organizações que não estão familiarizadas com a prática. A atividade é mais frequentemente realizada em um workshop com um grupo multifuncional – que inclui todas as pessoas envolvidas no projeto, no desenvolvimento, no lançamento e na operação do software – em torno de um quadro branco compartilhado (ou ferramenta virtual equivalente). Primeiro, as etapas do processo são listadas e ordenadas, desde a estação de trabalho da pessoa desenvolvedora até a produção. Em seguida, uma sessão facilitada é usada para capturar mais informações e pontos problemáticos. A técnica mais comum que vemos é baseada em **mapeamento de fluxo de valor**, embora muitos **mapas de processo** sejam igualmente valiosos. A atividade é muitas vezes reveladora para muitas das pessoas que participam, pois elas identificam atrasos, riscos e inconsistências e continuam a usar a representação visual para a melhoria contínua do processo de compilação e implantação. Consideramos essa técnica tão fundamental que nos surpreendemos ao descobrir que não a havíamos incluído no Radar antes.

2. Carga cognitiva do time

Adote

A interação do time é um conceito-chave ao redesenhar uma organização para agilidade e velocidade nos negócios. Essas interações serão refletidas no software que está sendo construído (consulte a **Lei de Conway**) e indicarão o quão eficientemente os times podem entregar valor de forma autônoma a clientes. Nosso conselho é ser intencional sobre como os times são projetados e como interagem. Por acreditarmos que o design organizacional e as interações do time evoluem com o tempo, achamos que é particularmente importante medir e acompanhar a carga cognitiva do time, que indica a facilidade ou a dificuldade dos times para construir, testar e manter seus serviços. Estamos usando um **modelo** para avaliar a carga cognitiva do time com base nas ideias dos autores do ***Team Topologies***.

O impacto positivo da aplicação dos conceitos deste livro na comunicação com clientes e no redesenho das organizações continua nos impressionando. Os autores recomendam uma abordagem simples, mas poderosa para o design organizacional, identificando apenas quatro tipos de times e três modos de interação. Isso ajuda a reduzir a ambiguidade dentro da organização e fornece um glossário comum para que times, partes interessadas e liderança descrevam e desenhem o trabalho de uma equipe. Para implementar uma mudança no design da organização, projetamos a estrutura de topologias de time ideal, aplicamos quaisquer restrições técnicas/equipe (por exemplo, profissionais insuficientes) e, em seguida, concluímos com a estrutura final. Isso nos permite aconselhar melhor clientes e antecipar se estamos de fato melhorando a carga cognitiva ao comparar as estruturas de time nos estágios como estão/como devem estar.

3. Modelagem de ameaças

Adote

Continuamos recomendando que os times executem **modelagem de ameaças** — um conjunto de técnicas para ajudar a identificar e classificar ameaças potenciais durante o processo de desenvolvimento — mas queremos enfatizar que esta não é uma atividade pontual realizada apenas no início dos projetos; as equipes precisam evitar o **sanduíche de segurança**. Isso ocorre porque ao longo da vida útil de qualquer software, novas ameaças surgirão e as existentes continuarão a evoluir graças a eventos externos e mudanças contínuas nos requisitos e na arquitetura. Isso

significa que a modelagem de ameaças precisa ser repetida periodicamente — a frequência de repetição dependerá das circunstâncias e precisará considerar fatores como o custo de execução do exercício e o risco potencial para o negócio. Quando usada em conjunto com outras técnicas, como estabelecer requisitos de segurança multifuncionais para lidar com riscos comuns nas tecnologias do projeto e usar verificadores de segurança automatizados, a modelagem de ameaças pode ser um ativo poderoso.

4. BERT

Experimente

Desde a última vez que falamos sobre **BERT** (Representações de Codificador Bidirecional de Transformadores, ou *Bidirectional Encoder Representations from Transformers* em inglês) no Radar, nossos times o usaram com sucesso em alguns projetos de processamento de linguagem natural (PLN). Em uma de nossas clientes, observamos melhorias significativas quando mudamos do tokenizador BERT padrão para um tokenizador de pedaços de palavras treinado por domínio para consultas que contêm substantivos como nomes de marcas ou dimensões. Embora o PLN tenha vários novos modelos de transformadores, o BERT é bem compreendido, conta com boa documentação e uma comunidade vibrante, e continuamos a considerá-lo eficiente em um contexto de PLN empresarial.

5. Testes de regressão visual de componentes

Experimente

Testes de regressão visual são recursos úteis e poderosos para ter em sua caixa de ferramentas, mas têm um custo significativo, pois são feitos para páginas inteiras. Com o surgimento de frameworks baseados em componentes como **React** e **Vue**, também observamos a ascensão dos testes de regressão visual de componentes. Essa técnica atinge um bom equilíbrio entre valor e custo para garantir que nenhum elemento visual indesejado seja adicionado à aplicação. Em nossa experiência, o teste de regressão visual de componentes apresenta menos falsos positivos e promove um bom estilo de arquitetura. Em conjunto com ferramentas como **Vite** e o recurso **Hot Module Replacement (HMR)** do webpack, seu uso pode ser visto como uma mudança de paradigma para aplicar o desenvolvimento orientado a testes ao desenvolvimento front-end.

6. Tokens de design

Experimente

Ao ser confrontada com o desafio de usar um **sistema de design** de forma consistente em uma variedade de formatos e plataformas, a equipe da Salesforce criou o conceito de **tokens de design**. Os tokens armazenam valores, como cores e fontes, em um local centralizado. Isso possibilita **separar opções de decisões** e melhora significativamente a **colaboração entre times**. Os tokens de design não são novos, mas com a introdução de ferramentas como **Tailwind CSS** e **Style Dictionary**, temos visto tokens de design sendo usados com mais frequência.

7. Servidor SMTP falso para testar o envio de e-mails

Experimente

Usar contas de e-mail de testes ou servidores SMTP (Single Mail Transfer Protocol) de testes completos continua sendo uma prática comum de teste de software. No entanto, usar um servidor real traz o risco de que **e-mails de teste sejam enviados para pessoas reais** e muitas vezes complica o teste de integração automatizado. Tivemos sucesso ao usar um servidor SMTP falso para testar o

envio de e-mails, que registra uma solicitação para enviar um e-mail sem realmente enviá-lo. Existem várias ferramentas de código aberto neste espaço, incluindo **fake-smtp-server**, que renderiza e-mails em uma interface de usuário web para testes visuais e **mountebank**, que expõe os e-mails enviados por meio de uma API REST para testes de integração. Recomendamos explorar essa técnica para reduzir o risco e melhorar a eficiência dos testes.

8. Aprendizado de máquina federado

Experimente

Estamos observando atualmente projetos de clientes que usam aprendizado de máquina federado. Tradicionalmente, o treinamento de modelos de aprendizado de máquina (ML) exigia que os dados fossem colocados em um local centralizado onde o algoritmo de treinamento relevante pudesse ser executado. Do ponto de vista de privacidade, isso é problemático, especialmente quando os dados de treinamento contêm informações confidenciais ou de identificação pessoal. As pessoas usuárias podem relutar em compartilhar dados ou a legislação local de proteção de dados pode nos impedir de mover os dados para um local central. O ML federado é uma técnica descentralizada para treinamento em um grande conjunto diversificado de dados que permite que os dados permaneçam remotos, por exemplo, no dispositivo de uma pessoa usuária. A largura de banda da rede e as limitações computacionais dos dispositivos ainda apresentam desafios técnicos significativos, mas gostamos da maneira como o aprendizado de máquina federado deixa os usuários no controle de suas próprias informações pessoais.

9. Plataforma de desenvolvimento incremental

Experimente

Temos abordado plataformas de desenvolvimento e como construí-las em quase todas as edições do Radar desde 2017. Enquanto isso, o livro ***Team Topologies*** também fez um ótimo trabalho ao descrever uma plataforma ideal que oferece suporte a pessoas desenvolvedoras com "APIs de autoatendimento, ferramentas, serviços e conhecimento". No entanto, muitas vezes vemos times buscando adotar muito dessa visão de plataforma muito rápido. Em vez disso, construir uma plataforma de desenvolvimento incremental é a chave.

O Team Topologies recomenda sempre buscar o que chamam de "plataforma mais enxuta viável" necessária em qualquer estágio, na qual a primeira versão pode até ser apenas um conjunto de documentação em um wiki. O próximo incremento pode aumentar o nível de serviço fornecendo modelos ou permitindo que os times criem pull requests. Incrementos adicionais poderiam então introduzir APIs de autoatendimento, mas somente se valiosas. Em suma, embora tenhamos alertado contra **modelos operacionais de plataforma totalmente orientados por tickets**, passar de zero ao autoatendimento é o outro extremo. Siga seu ritmo, **trate sua plataforma como um produto** e construa-a de forma incremental.

10. Micro frontends para aplicativos móveis

Experimente

Desde sua introdução no Radar em 2016, vimos uma ampla adoção de **micro frontends** para interfaces de usuário web. Recentemente, no entanto, observamos projetos estendendo esse estilo de arquitetura para incluir também micro frontends para aplicativos móveis. Quando um aplicativo se torna suficientemente grande e complexo, torna-se necessário distribuir o desenvolvimento entre vários times. Isso apresenta uma série de desafios em torno da autonomia do time, estruturas de repositório e frameworks de integração. No passado, mencionamos **Atlas e BeeHive**, mas esses

frameworks não conseguiram ganhar força e não estão mais em desenvolvimento ativo. Abordagens mais recentes incluem **Tuist** ou **Swift Package Manager** para integrar o trabalho de vários times em um único aplicativo. Mas, em nossa experiência, os times geralmente acabam implementando seu próprio framework de integração. Embora definitivamente vejamos a necessidade de modularidade na ampliação dos times de desenvolvimento móvel, o caso dos micro frontends é menos certo. Isso porque enquanto os micro frontends implicam uma correspondência direta entre times e páginas ou componentes, essa estrutura pode acabar confundindo responsabilidades por contextos de domínio de negócio, aumentando assim a **carga cognitiva do time**. Nosso conselho é seguir o básico de um design de aplicativo bom e limpo, adotar a modularidade ao escalar para vários times e adotar uma arquitetura de micro frontend somente quando os módulos e o domínio de negócio estiverem fortemente alinhados.

11. Observabilidade para pipelines de CI/CD

Experimente

As práticas de observabilidade mudaram o foco da conversa de monitoramento de problemas bem compreendidos para apoio na solução de problemas desconhecidos em sistemas distribuídos. Tivemos sucesso ao tirar essa perspectiva do ambiente de produção tradicional, aplicando observabilidade para pipelines de CI/CD para ajudar a otimizar os gargalos de teste e implantação. Pipelines complexos criam atrito para pessoas desenvolvedoras quando são executados muito lentamente ou sofrem de não determinismo, reduzindo importantes ciclos de feedback e prejudicando a eficiência da pessoa desenvolvedora. Além disso, seu papel como infraestrutura de implantação crítica cria pontos de estresse durante os períodos de deployment rápidos, como aconteceu com várias organizações que responderam à recente vulnerabilidade do log4shell. O conceito de rastreamentos se traduz muito bem em pipelines: em vez de capturar a cascata de chamadas de serviço, os child spans capturam informações sobre cada estágio da compilação. Os mesmos gráficos em cascata usados para analisar um fluxo de chamadas em uma arquitetura distribuída também podem ser eficazes para nos ajudar a identificar gargalos em pipelines, mesmo que sejam complexos com fan-in e fan-out. Isso permite esforços de otimização com foco muito melhor definido. Embora a técnica deva funcionar com qualquer ferramenta de rastreamento, o **Honeycomb** suporta uma ferramenta chamada **buildevents** que ajuda a capturar informações sobre o rastreamento de pipelines. Uma abordagem alternativa de captura de informações já expostas por plataformas CI/CD, feita pela ferramenta de código aberto **buildviz** (desenvolvida e mantida por um Thoughtworker), permite uma investigação semelhante sem alterar as próprias configurações da etapa.

12. SLSA

Experimente

À medida que a complexidade do software continua a crescer, proteger-se do vetor de ameaças de dependências de software torna-se cada vez mais desafiador. Níveis de Cadeia de Fornecimento para Artefatos de Software (*Supply chain Levels for Software Artifacts*, em inglês), ou **SLSA** (pronuncia-se “salsa”), é uma curadoria de orientações para ajudar as organizações a se protegerem de ataques à cadeia de fornecimento, e uma evolução do conjunto de orientações interno que o Google vem usando há anos. Apreciamos que SLSA não promete uma abordagem “bala de prata” baseada apenas em ferramentas para proteger a cadeia de fornecimento, mas provê uma lista de verificação de ameaças e práticas concretas junto a um modelo de maturidade. O **modelo de ameaças** 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 crescentes para melhorar sua postura de segurança na cadeia de fornecimento. Desde que incluímos pela primeira vez no Radar, SLSA adicionou mais detalhes sobre **atestados de software** com exemplos para rastrear preocupações

como **proveniência de compilação**. Nossos times concluíram que SLSA consegue um bom equilíbrio entre a orientação para implementação e a conscientização de alto nível sobre as ameaças da cadeia de fornecimento.

13. 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 nas bibliotecas das quais dependem. Desde que a **Ordem Executiva** original foi publicada, a indústria compreendeu o que é uma SBOM e como criá-la. O National Institute of Standards and Technology (NIST), por exemplo, agora tem mais **recomendações específicas** sobre como cumprir a ordem. Temos experiência de produção usando SBOMs em projetos que vão de pequenas empresas a grandes multinacionais e até departamentos governamentais, e temos convicção de que trazem benefícios. Mais organizações e governos devem considerar exigir SBOMs para software em uso. A técnica será fortalecida por novas ferramentas que continuam surgindo, como **Firebase Android BOM** que alinha automaticamente as dependências da biblioteca de uma aplicação àquelas listadas na SBOM.

14. Eficiência de carbono como uma característica arquitetural

Avalie

A sustentabilidade é um tema que demanda a atenção das empresas. No espaço de desenvolvimento de software, sua importância aumentou, e agora estamos vendo **diferentes maneiras** para abordar este tema. Observando a pegada de carbono do desenvolvimento de software, por exemplo, recomendamos avaliar a eficiência de carbono como uma característica arquitetural. Uma arquitetura que leva em consideração a eficiência de carbono é aquela em que as escolhas de design e infraestrutura são feitas para minimizar o consumo de energia e, portanto, as emissões de carbono. As ferramentas de medição e recomendações neste espaço estão amadurecendo, tornando viável que os times considerem a eficiência de carbono juntamente com outros fatores como desempenho, escalabilidade, custo financeiro e segurança. Como quase tudo na arquitetura de software, isso deve ser considerado como uma compensação. Nosso conselho é pensar nisso como uma característica adicional em todo um conjunto de **atributos de qualidade** relevantes que são orientados e priorizados por objetivos organizacionais, e não relegados a um pequeno grupo de especialistas para refletir de forma isolada.

15. 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.

16. GitHub push protection

Avalie

A publicação acidental de segredos parece ser um problema perene, à medida que ferramentas como Talisman surgem para ajudar a resolver o problema. Até agora, os usuários do GitHub Enterprise Cloud com uma licença de segurança avançada podiam habilitar a verificação de segurança em suas contas, e quaisquer segredos (chaves de API, tokens de acesso, credenciais etc.) que fossem acidentalmente enviados acionariam um alerta. **GitHub push protection** dá um passo adiante, e traz isso para uma etapa anterior no fluxo de trabalho de desenvolvimento, impedindo que as alterações sejam enviadas se segredos forem detectados. Isso precisa ser configurado para a organização e se aplica evidentemente apenas aos detentores de licenças, mas a proteção adicional contra a publicação de segredos é bem-vinda.

17. Aplicação local-first

Avalie

Em uma aplicação centralizada, os dados no servidor são a única fonte de verdade — qualquer modificação nos dados deve passar pelo servidor. Os dados locais são subordinados à versão do servidor. Esta parece ser uma escolha natural e inevitável para permitir a colaboração entre vários usuários do software. Aplicação local-first, ou **software local-first**, é um conjunto de princípios que permite tanto colaboração quanto propriedade de dados locais. A abordagem prioriza o uso de armazenamento local e redes locais em relação a servidores em data centers remotos ou na nuvem. Técnicas como tipos de dados replicados sem conflito (CRDTs) e redes peer-to-peer (P2P) têm o potencial de ser uma tecnologia fundamental para a realização de software local-first.

18. Repositório de métricas

Avalie

Um **repositório de métricas**, às vezes chamado de headless business intelligence (BI), é uma camada que desacopla as definições de métricas a partir de seu uso em relatórios e visualizações. Tradicionalmente, métricas são definidas dentro do contexto das ferramentas de BI, mas essa abordagem leva a duplicações e inconsistências, pois diferentes times as utilizam em diferentes contextos. Ao desacoplar a definição no repositório de métricas, obtemos uma reutilização nítida e consistente em relatórios de BI, visualizações e até análises incorporadas. Esta técnica não é nova. Por exemplo, o Airbnb introduziu o **Minerva** um ano atrás. No entanto, agora estamos vendo uma tração considerável no ecossistema de dados e análises com mais ferramentas que suportam repositórios de métricas prontas para uso.

19. Interface de usuário orientada a servidor

Avalie

A interface de usuário orientada a servidor continua a ser um assunto polêmico nos círculos de discussão sobre dispositivos móveis, porque oferece às pessoas desenvolvedoras o potencial de aproveitar os ciclos de mudança mais rápidos sem entrar em conflito com as políticas das lojas de aplicativos sobre a revalidação do próprio aplicativo para dispositivos móveis. A interface de usuário orientada ao 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 um percurso de ida e volta a uma loja de aplicativos agora podem ser realizadas por meio de alterações simples nas respostas que o servidor envia. Embora alguns times de aplicativos móveis muito grandes tenham obtido bastante sucesso com essa técnica, isso também

exige um investimento substancial na construção e manutenção de um framework proprietário complexo. Tal investimento requer um caso de negócio convincente. Até que haja bons argumentos, talvez seja melhor proceder com cautela. Na verdade, nós tivemos algumas experiências com bagunças terríveis e excessivamente configuráveis que não entregaram os benefícios prometidos. Mas com o apoio de gigantes como Airbnb e Lyft, podemos verdadeiramente ver surgir alguns frameworks úteis capazes de ajudar a domar a complexidade. Fique de olho nesse espaço.

20. SLIs e SLOs como código

Avalie

Desde que o Google popularizou indicadores de nível de serviço (SLIs) e objetivos de nível de serviço (SLOs) como parte de sua prática de engenharia de confiabilidade do site (SRE), ferramentas de observabilidade como **Datadog**, **Honeycomb** e **Dynatrace** começaram a incorporar o monitoramento de SLOs em seu conjunto de ferramentas. **OpenSLO** é um padrão emergente que permite definir SLIs e SLOs como código, usando uma linguagem de especificação declarativa e independente de fornecedor com base no formato YAML usado por **Kubernetes**. Embora o padrão ainda seja bastante novo, estamos vendo uma conjuntura positiva, como é o caso da contribuição da ferramenta **slogen**, da Sumo Logic, para gerar monitoramento e painéis. Estamos otimistas com a promessa de versionar as definições de SLI e SLO no código e atualizar as ferramentas de observabilidade como parte do pipeline de CI/CD do serviço que está sendo implantado.

21. Dados sintéticos para testes de modelos

Avalie

Durante nossas discussões para esta edição do Radar, surgiram várias ferramentas e aplicações para geração de dados sintéticos. À medida que as ferramentas amadurecem, descobrimos que usar dados sintéticos para testes de modelos é uma técnica poderosa e amplamente útil. Embora não pretenda substituir os dados reais na validação do poder de discriminação dos modelos de aprendizado de máquina, os dados sintéticos podem ser usados em diversas situações. Por exemplo, podem ser usados para proteção contra falhas catastróficas do modelo em resposta a eventos que ocorrem raramente ou para testar pipelines de dados sem expor informações de identificação pessoal. Dados sintéticos também são úteis para explorar casos extremos que não possuem dados reais ou para identificar vieses de modelo. Algumas ferramentas úteis para gerar dados incluem **Faker** ou **Synth**, que geram dados de acordo com as propriedades estatísticas desejadas, e ferramentas como **Synthetic Data Vault**, que podem gerar dados que imitam as propriedades de um conjunto de dados de entrada.

22. TinyML

Avalie

Continua nos entusiasmando a técnica de **TinyML** e a capacidade de criar modelos de aprendizado de máquina (ML) projetados para serem executados em dispositivos móveis e de baixa potência. Até recentemente, a execução de um modelo de ML era vista como computacionalmente cara e, em alguns casos, exigia hardware para fins especiais. Embora a criação dos modelos ainda esteja amplamente dentro dessa classificação, os modelos agora podem ser criados de uma maneira que permita sua execução em dispositivos pequenos, de baixo custo e baixo consumo de energia. Se você está pensando em usar ML mas considerava inviável devido a restrições de computação ou rede, vale a pena avaliar essa técnica.

23. Credenciais verificáveis

Avalie

Quando as incluímos pela primeira vez no Radar, há dois anos, as credenciais verificáveis eram um padrão intrigante com algumas aplicações promissoras em potencial, mas não eram amplamente conhecidas ou entendidas fora da comunidade de entusiastas. Isso era particularmente verdadeiro quando se tratava de instituições de concessão de credenciais, como governos de estado, que seriam responsáveis pela implementação dos padrões. Dois anos e uma pandemia depois, a demanda por credenciais eletrônicas criptograficamente seguras, que respeitam a privacidade e são verificáveis por máquina cresceu e, como resultado, os governos estão começando a despertar para o potencial das credenciais verificáveis. Agora estamos começando a ver as credenciais verificáveis surgirem em nossos trabalhos para clientes do setor público. O **padrão W3C** coloca titulares de credenciais no centro, o que é semelhante à nossa experiência ao usar credenciais físicas: as pessoas usuárias podem colocar suas credenciais verificáveis em suas próprias carteiras digitais e mostrá-las a qualquer pessoa a qualquer momento sem a permissão da entidade emissora das credenciais. Essa abordagem descentralizada também permite que os usuários gerenciem melhor e divulguem seletivamente suas próprias informações, o que melhora muito a proteção da privacidade dos dados. Por exemplo, com tecnologia de prova de conhecimento zero, você pode construir uma credencial verificável para provar que é uma pessoa adulta sem revelar seu aniversário. É importante notar que, embora muitas soluções de **identidade descentralizada** baseadas em credenciais verificáveis dependam da tecnologia de blockchain, blockchain não é um pré-requisito para todas as implementações de credenciais verificáveis.

24. Profissionais satélite sem formas de trabalho “nativamente remotas”

Evite

O termo “configuração de times remotos” não descreve apenas uma configuração, mas abrange vários **padrões e nuances**. E muitos times vêm mudando os padrões recentemente. Estão saindo do modo “todo mundo sempre remoto” que foi imposto pela pandemia e mudando para um padrão de profissionais satélite (muitas em rotatividade), no qual parte do time está colocalizado enquanto outra parte fica remota. Vemos muitos times falhando ao não considerar adequadamente o que isso significa para suas formas de trabalho. Profissionais satélite sem formas de trabalho “nativamente remotas” são um retrocesso no sentido de privilegiar práticas colocalizadas. Em uma configuração com profissionais satélite, é importante ainda **usar processos e abordagens “nativamente remotos” por padrão**. Por exemplo, se a parte colocalizada do time ingressar em uma reunião, todos os membros ainda devem estar em seus laptops individuais para participar da colaboração digital ou do bate-papo da reunião. Os times precisam estar cientes do risco de excluir profissionais satélites, criando silos e sentimentos de exclusão. Se você sabe que sempre terá pelo menos um membro satélite no time, as formas padrão de trabalho devem considerar a condição remota.

25. SPAs por padrão

Evite

A prevalência de times que escolhem uma aplicação de página única (SPA) quando precisam de um site continua. Ainda nos preocupa que as pessoas não estejam reconhecendo adequadamente SPAs como um estilo arquitetural inicialmente; em vez disso, elas estão imediatamente pulando para a seleção do framework. SPAs incorrem em uma complexidade que simplesmente não existe em sites tradicionais baseados em servidor: questões como otimização de mecanismos de pesquisa, gerenciamento de histórico do navegador, web analytics e tempo de carregamento da primeira página

precisam ser abordadas. A análise e a consideração adequadas das vantagens e desvantagens são necessárias para determinar se essa complexidade é justificada por motivos de negócios ou de experiência do usuário. Muitas vezes, as equipes ignoram essa análise, aceitando cegamente a complexidade de SPAs por padrão, mesmo quando as necessidades de negócio não justificam. Ainda vemos algumas pessoas desenvolvedoras que não conhecem uma abordagem alternativa porque passaram toda a sua carreira usando um framework como o React. Acreditamos que muitos sites se beneficiarão da simplicidade da lógica do lado do servidor e técnicas como **Hotwire**, que ajudam a resolver os problemas com a experiência do usuário, nos encorajam.

26. Nativo de nuvem superficial

Evite

O termo “cloud native” foi originalmente usado para descrever arquiteturas com características que tiravam o máximo proveito da hospedagem em nuvem pública. Os exemplos incluem arquiteturas distribuídas compostas por muitos processos pequenos, sem estado e colaborativos, e sistemas com altos níveis de automação para construção, teste e implantação de aplicações. No entanto, notamos uma tendência crescente para design nativo de nuvem superficial, que simplesmente usa muitos serviços proprietários de um fornecedor de nuvem e para por aí, sem revisitar a natureza fundamentalmente monolítica, frágil ou com muitas tarefas manuais e repetitivas da aplicação. É importante lembrar que as funções sem servidor por si só não tornam uma aplicação mais resiliente ou mais fácil de manter, e que a nuvem nativa é realmente uma questão de design e não um conjunto de opções de implementação.

Plataformas

Adote

27. Backstage
28. Delta Lake

Experimente

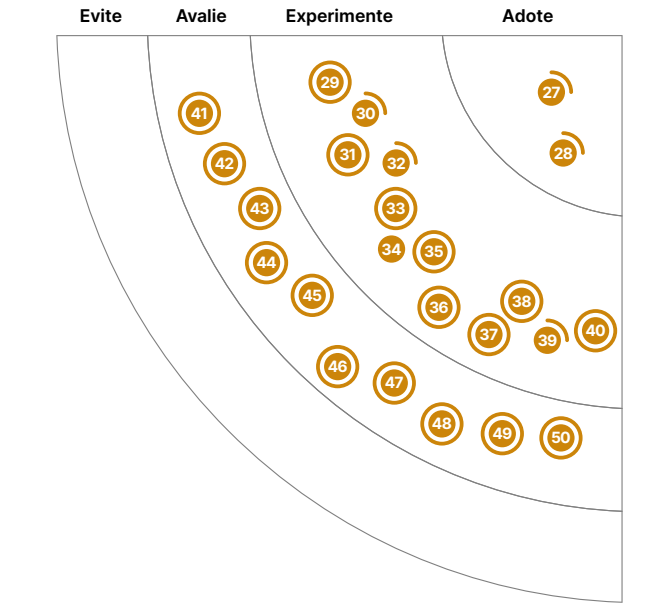
29. AWS Database Migration Service
30. Colima
31. Databricks Photon
32. DataHub
33. DataOps.live
34. eBPF
35. Feast
36. Monte Carlo
37. Retool
38. Seldon Core
39. Teleport
40. VictoriaMetrics

Avalie

41. Bun
42. Databricks Unity Catalog
43. Dragonfly
44. Edge Impulse
45. GCP Vertex AI
46. Gradient
47. IAM Roles Anywhere
48. Keptn
49. OpenMetadata
50. OrioleDB

Evite

—



Novo Mudança de anel Sem modificação

27. Backstage

Adote

Em um mundo cada vez mais digital, melhorar a eficácia das pessoas desenvolvedoras em grandes organizações costuma ser uma preocupação central da liderança sênior. Observamos valor suficiente em portais de desenvolvimento em geral, **Backstage** em particular, e por isso estamos felizes em recomendá-lo em "Adote". Backstage é uma plataforma de portal de desenvolvimento de código aberto criada pelo Spotify, que melhora a descoberta de ativos de software em toda a organização. A plataforma usa Markdown **TechDocs** que acompanham o código de cada serviço, o que equilibra muito bem as necessidades de descoberta centralizada com a necessidade de propriedade distribuída de ativos. O Backstage oferece suporte a modelos de software para acelerar novos desenvolvimentos e uma arquitetura de plug-in que permite extensibilidade e adaptabilidade no ecossistema de infraestrutura de uma organização. O **Backstage Service Catalog** usa arquivos YAML para rastrear a propriedade e metadados de todo o software no ecossistema de uma organização. Permite ainda rastrear software SaaS de terceiros, o que geralmente requer o rastreamento de propriedade.

28. Delta Lake

Adote

Delta Lake é uma **camada de armazenamento de código aberto**, implementada pela Databricks, que tenta trazer transações ACID para processamento de big data. Em nossos projetos de **malha de dados** (data mesh) ou **lago de dados** habilitado por Databricks, nossos times preferem usar o armazenamento Delta Lake em vez do uso direto de tipos de armazenamento de arquivos como **AWS S3** ou **ADLS**. Até recentemente, o Delta Lake era um produto proprietário fechado da Databricks, mas agora é código aberto e acessível a plataformas não Databricks. No entanto, nossa recomendação de Delta Lake como uma opção padrão atualmente se estende apenas a projetos Databricks que usam formatos de arquivo **Parquet**. O Delta Lake facilita os casos de uso de leitura/escrita de dados simultâneas em que a transacionalidade no nível do arquivo é necessária. Consideramos a fluida integração do Delta Lake com APIs de Apache Spark **batch** e **micro-batch** muito útil, principalmente recursos como **time travel** (acessar dados em um determinado momento ou reverter um commit), bem como suporte para gravação no **schema evolution**, embora haja algumas limitações nesses recursos.

29. AWS Database Migration Service

Experimente

Muitos de nossos times usaram com sucesso o **AWS Database Migration Service** (DMS) para migrar dados de e para a AWS. Em um de nossos projetos de transformação digital, conseguimos chegar a quase zero de tempo de inatividade para o novo sistema, pois migramos dados do Microsoft SQL Server para uma instância PostgreSQL do AWS Relational Database Service (RDS). Essas transformações envolvem muitas partes móveis que exigem planejamento e coordenação entre times multidisciplinares, mas para a migração de dados estamos muito felizes com o DMS. O serviço administra automaticamente a implantação, o gerenciamento e o monitoramento de todos os recursos necessários. Ao longo dos anos, o DMS amadureceu para oferecer suporte a várias databases **source** e **target**, e continuamos gostando.

30. Colima

Experimente

Colima está se tornando uma alternativa de código aberto popular em relação ao Docker Desktop. A plataforma provisiona o tempo de execução do contêiner Docker em uma VM Lima, configura a CLI do Docker no macOS e lida com encaminhamento de porta e escala de volume. Colima usa containerd como seu tempo de execução, que também é o tempo de execução na maioria dos serviços gerenciados do Kubernetes — melhorando a importante paridade entre desenvolvimento e produção. Com Colima, você pode usar e testar facilmente os recursos mais recentes do containerd, como carregamento lento para imagens de contêiner. Temos tido bons resultados com Colima em nossos projetos. No espaço Kubernetes, também usamos nerdctl, uma CLI compatível com Docker para containerd. Como o Kubernetes descontinuou o Docker como tempo de execução de contêiner e a maioria dos serviços gerenciados (EKS, GKE etc.) estão seguindo esse caminho, mais pessoas procurarão ferramentas nativas de contêiner, daí a importância de ferramentas como nerdctl. Em nossa opinião, o Colima está mostrando seu forte potencial e se tornando uma opção escolhida como alternativa ao Docker Desktop.

31. Databricks Photon

Experimente

A partir do Databricks 9.1 LTS (Long Term Support), foi disponibilizado um novo runtime chamado Databricks Photon, uma alternativa que foi reescrita do zero em C++. Vários de nossos times já usaram o Photon em produção e ficaram satisfeitos com as melhorias de desempenho e a consequente redução de custos. Melhorias e mudanças reais nos custos dependerão de vários fatores, como tamanho do conjunto de dados e tipos de transação. Recomendamos testar em uma carga de trabalho realista para coletar dados para uma comparação antes de tomar qualquer decisão sobre o uso do Photon.

32. DataHub

Experimente

Desde que mencionamos a detecção de dados pela primeira vez no Radar, o LinkedIn evoluiu o WhereHows para DataHub, uma plataforma de próxima geração que aborda a descoberta de dados por meio de um sistema de metadados extensível. Em vez de rastrear e extrair metadados, DataHub adota um modelo baseado em push, em que componentes individuais do ecossistema de dados publicam metadados por meio de uma API ou um fluxo para a plataforma central. Essa integração baseada em push transfere a propriedade da entidade central para times individuais, tornando-os responsáveis por seus metadados. Como resultado, usamos o DataHub com sucesso como um repositório de metadados em toda a organização e como ponto de entrada para vários produtos de dados mantidos de forma autônoma. Ao adotar essa abordagem, certifique-se de mantê-la leve e evitar o caminho escorregadio que leva ao controle centralizado sobre um recurso compartilhado.

33. DataOps.live

Experimente

DataOps.live é uma plataforma de dados que automatiza ambientes em Snowflake. Inspirado nas práticas de DevOps, DataOps.live permite que você trate a plataforma de dados como qualquer outra plataforma web, adotando integração contínua e entrega contínua (CI/CD), testes automatizados, observabilidade e gerenciamento de código. Você pode reverter as alterações imediatamente sem

afetar os dados ou recuperar-se de falhas completas e reconstruir um novo tenant do Snowflake em minutos ou horas, em vez de dias. Nossos times tiveram uma boa experiência com DataOps.live, porque nos permitiu iterar rapidamente ao criar produtos de dados no Snowflake.

34. eBPF

Experimente

Há vários anos, o kernel do Linux inclui o Berkeley Packet Filter estendido (**eBPF**), uma máquina virtual que fornece 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 muito pouca sobrecarga. A possibilidade de execução de programas em área restrita dentro do kernel do sistema operacional habilita pessoas desenvolvedoras de aplicativos a executar programas eBPF para adicionar recursos adicionais ao sistema operacional em tempo de execução. Alguns de nossos projetos exigem solução de problemas e criação de perfil no nível de chamada do sistema, e nossos times descobriram que ferramentas como **bcc** e **bpftrace** facilitam seu trabalho. A observabilidade e a infraestrutura de rede também se beneficiam do eBPF — por exemplo, o projeto **Cilium** pode implementar balanceamento de carga de tráfego e observabilidade **sem sobrecarga de sidecar** em **Kubernetes**, e **Hubble** oferece mais segurança e observabilidade de tráfego. O projeto **Falco** usa eBPF para monitoramento de segurança, e o projeto **Katran** usa eBPF para construir balanceamento de carga L4 mais eficiente. A comunidade eBPF está crescendo rapidamente e temos visto cada vez mais sinergia com o campo da observabilidade.

35. Feast

Experimente

Feast é uma **Feature Store** de código aberto para aprendizado de máquina. Tem várias propriedades úteis, incluindo a geração de conjuntos de atributos corretos em um determinado momento — para que os valores de recursos futuros propensos a erros não vazem para os modelos durante o treinamento — e suporte a fontes de dados de streaming e em lote. No entanto, atualmente suporta apenas dados estruturados com carimbo de data/hora e, portanto, pode não ser adequado se você trabalhar com dados não estruturados em seus modelos. Usamos com sucesso Feast em uma escala significativa como um repositório offline durante o treinamento do modelo e como um repositório online durante a predição.

36. Monte Carlo

Experimente

Monte Carlo é uma plataforma de observabilidade de dados. Usando modelos de aprendizado de máquina, Monte Carlo infere e aprende sobre os dados, identificando problemas e notificando os usuários quando eles surgem. A plataforma permite que nossos times mantenham a qualidade dos dados em pipelines ETL, data lakes, data warehouses e relatórios de inteligência de negócios (BI). Com recursos como painéis de monitoramento como código, catálogo de dados central e linhagem em nível de campo, nossos times consideram Monte Carlo uma ferramenta de grande valor para a governança geral de dados.

37. Retool

Experimente

Em edições anteriores, recomendamos avaliar **plataformas limitadas de baixo código** como um método para aplicar soluções de baixo código para casos de uso específicos em domínios muito

limitados. Vimos alguma tração nesse espaço, especificamente com **Retool**, uma plataforma de baixo código que nossos times usam para criar soluções para usuários internos, principalmente para consultar e visualizar dados. A plataforma permite que os times produzam soluções somente leitura não essenciais aos negócios mais rapidamente. Os principais benefícios relatados do Retool são seus componentes de interface do usuário e sua capacidade de integração rápida e fácil com fontes de dados comuns.

38. Seldon Core

Experimente

Seldon Core é uma plataforma de código aberto para empacotar, implantar, monitorar e gerenciar modelos de aprendizado de máquina no **Kubernetes**. Com suporte pronto para uso para vários frameworks de aprendizado de máquina, você pode facilmente colocar seus modelos em contêiner usando **servidores de inferência pré-empacotados**, **servidores de inferência personalizados** ou **wrappers de linguagem**. Com rastreamento distribuído por meio de **Jaeger** e explicabilidade do modelo com **Alibi**, o Seldon Core enderessa vários desafios de reta final de entrega com implantações de aprendizado de máquina, e nossos times de dados gostam disso.

39. Teleport

Experimente

Teleport é uma ferramenta para acesso de rede à infraestrutura de **confiança zero**. As configurações tradicionais exigem políticas complexas ou jump servers para restringir o acesso a recursos críticos. Teleport, no entanto, simplifica isso com um plano de acesso unificado e com controles de autorização refinados que substituem jump servers, VPNs ou credenciais compartilhadas. Implementado como um único binário com suporte imediato para vários protocolos (incluindo SSH, RDP, **Kubernetes** API, MySQL, **MongoDB** e protocolos de PostgreSQL wire), Teleport facilita a configuração e o gerenciamento de acesso seguro em ambientes Linux, Windows ou Kubernetes. Desde que o mencionamos pela primeira vez no Radar, alguns times vem usando Teleport e nossa experiência geral positiva nos levou a destacá-lo.

40. VictoriaMetrics

Experimente

A observabilidade moderna depende da coleta e agregação de um conjunto extenso de métricas granulares para entender, prever e analisar integralmente o comportamento do sistema. Mas quando aplicada a um sistema nativo de nuvem composto por muitos processos e hosts redundantes e cooperativos, a cardinalidade (ou número de séries temporais únicas) se torna difícil, porque cresce exponencialmente com cada serviço adicional, contêiner, nó, cluster etc. Ao lidar com dados de alta cardinalidade, descobrimos que **VictoriaMetrics** tem um bom desempenho. VictoriaMetrics é particularmente útil para operar **microsserviços** hospedados em **Kubernetes**, e o operador VictoriaMetrics torna mais fácil para os times a implementação do seu próprio monitoramento em forma de autoatendimento. Também gostamos de sua arquitetura em componentes e sua capacidade de continuar coletando métricas mesmo quando o servidor central não está disponível. Embora nossa equipe esteja satisfeita com VictoriaMetrics, esta é uma área em rápida evolução, e recomendamos ficar de olho em outras bancos de dados de séries temporais de alto desempenho compatíveis com **Prometheus**, como **Cortex** ou **Thanos**.

41. Bun

Avalie

Bun é um novo ambiente de execução JavaScript, semelhante ao Node.js ou Deno. Ao contrário do Node.js ou do Deno, no entanto, Bun é construído usando o JavaScriptCore do WebKit em vez do mecanismo V8 do Chrome. Projetado como um substituto imediato para o Node.js, Bun é um binário único (escrito em Zig) que atua como um empacotador, transpilador e gerenciador de pacotes para aplicativos JavaScript e TypeScript. Bun está atualmente em beta, então espere bugs ou problemas de compatibilidade com algumas bibliotecas Node.js. No entanto, foi construído desde o início com várias otimizações, incluindo inicialização rápida e renderização aprimorada do lado do servidor, e acreditamos que seja interessante avaliar.

42. Databricks Unity Catalog

Avalie

Databricks Unity Catalog é uma solução de governança de dados para ativos como arquivos, tabelas ou modelos de aprendizado de máquina em um lakehouse. Embora você encontre várias plataformas no espaço de governança de dados corporativos, se você já estiver usando outras soluções Databricks, certamente deve avaliar o Unity Catalog. Queremos destacar que, embora essas plataformas de governança geralmente implementem uma solução centralizada para melhor consistência entre espaços de trabalho e cargas de trabalho, a responsabilidade de governar deve ser federada, permitindo que times individuais gerenciem seus próprios ativos.

43. Dragonfly

Avalie

Dragonfly é um novo repositório de dados em memória compatível com Redis e APIs Memcached. Dragonfly aproveita a nova API io_uring específica do Linux para E/S e implementa novos algoritmos e estruturas de dados em cima de uma arquitetura multithread e sem compartilhamento. Por causa dessas escolhas inteligentes na implementação, Dragonfly alcança resultados impressionantes em desempenho. Embora Redis continue sendo nossa escolha padrão para soluções de armazenamento de dados na memória, consideramos Dragonfly uma opção interessante para avaliar.

44. Edge Impulse

Avalie

Em edições anteriores do Radar, escrevemos sobre TinyML — a prática de executar modelos treinados em pequenos dispositivos com sensores integrados para tomar decisões ou extrair recursos sem uma viagem de ida e volta para a nuvem. O Edge Impulse simplificou ao máximo o processo de coleta de dados do sensor, treinamento e implantação de um modelo. O Edge Impulse é uma plataforma de ponta a ponta hospedada para o desenvolvimento de modelos otimizados para execução em pequenos dispositivos de borda, como microcontroladores. A plataforma orienta a pessoa desenvolvedora por todo o pipeline, incluindo a tarefa de coletar e rotular dados de treinamento. Fica mais fácil começar a usar seu telefone celular para coleta de dados e execução do classificador, enquanto o treinamento e o refinamento do modelo acontecem no ambiente mais poderoso e hospedado na nuvem. Os algoritmos de reconhecimento resultantes também podem ser otimizados, compilados e carregados em uma ampla variedade de arquiteturas de microcontroladores. Embora o Edge Impulse seja um empreendimento comercial, a plataforma é gratuita para pessoas desenvolvedoras e torna todo o processo divertido e envolvente, mesmo para iniciantes no

aprendizado de máquina. A baixa barreira de entrada para a criação de um aplicativo funcional significa que veremos mais dispositivos de ponta com decisões inteligentes integradas.

45. GCP Vertex AI

Avalie

GCP Vertex AI é uma plataforma unificada de inteligência artificial (IA) que permite que os times criem, implantem e dimensionem modelos de aprendizado de máquina (ML). O Vertex AI inclui modelos pré-treinados, que podem ser usados diretamente, ajustados ou combinados com AutoML, bem como infraestrutura, por exemplo feature stores e pipelines para modelos de ML. Gostamos dos recursos integrados da Vertex AI, que ajudam a torná-la uma plataforma de IA coerente.

46. Gradient

Avalie

Gradient é uma plataforma para criar, implantar e executar aplicações de aprendizado de máquina, muito semelhante à Colab do Google. Os notebooks podem ser criados a partir de modelos, ajudando você a começar com PyTorch ou TensorFlow ou com aplicações como Stable Diffusion. Em nossa experiência, o Gradient é adequado para modelos com uso intenso de GPU e gostamos que o ambiente baseado em web seja persistente.

47. IAM Roles Anywhere

Avalie

IAM Roles Anywhere é um novo serviço da AWS que permite obter credenciais de segurança temporárias no IAM para cargas de trabalho como servidores, contêineres e aplicativos executados fora da AWS. Achamos isso particularmente útil em configurações de nuvem híbrida em que as cargas de trabalho são divididas entre recursos da AWS e não AWS. Em vez de criar credenciais de longa duração, com o IAM Roles Anywhere, agora você pode criar credenciais de curta duração para acessar recursos da AWS usando certificados X.509. Acreditamos que essa abordagem simplifica o padrão de acesso na nuvem híbrida e recomendamos que você confira.

48. Keptn

Avalie

Keptn é um plano de controle para entrega e operações que depende de CloudEvents para instrumentação. Como uma das técnicas que mencionamos em observalidade para pipelines de CI/CD, Keptn visualiza sua orquestração como vestígios. A definição declarativa da pipeline de entrega visa separar as intenções de SRE da implementação subjacente, contando com outras ferramentas de observabilidade, pipeline e implantação para responder aos eventos apropriados. Estamos particularmente otimistas com a ideia de adicionar verificações de objetivo de nível de serviço (SLO) como funções de aptidão arquitetural a pipelines de CI/CD: Keptn permite definir indicadores de nível de serviço (SLIs) como pares de chave-valor, com o valor representando a consulta à sua infraestrutura de observabilidade. Em seguida, a ferramenta avalia o resultado em relação aos SLOs definidos como um portão de qualidade. O Keptn adota a mesma abordagem para operações automatizadas, permitindo uma definição declarativa que especifica a intenção de dimensionar um ReplicaSet em resposta a uma degradação do tempo médio de resposta, por exemplo. Criado pela Dynatrace, o Keptn também se integra ao Prometheus e ao Datadog.

49. OpenMetadata

Avalie

Sem dúvida, **detecção de dados** tornou-se um ponto focal muito importante para as empresas, pois é um facilitador para que os dados sejam compartilhados e usados de forma eficiente por pessoas diferentes. Incluímos plataformas como **DataHub** e **Collibra** em edições anteriores do Radar, no entanto, nossos times estão constantemente avaliando opções nesse espaço e recentemente demonstraram interesse em **OpenMetadata**, uma plataforma dedicada ao gerenciamento de metadados usando padrões abertos. Nossos times gostam dessa plataforma de código aberto porque melhora a experiência de desenvolvimento devido à sua arquitetura simples, fácil implantação com foco em automação e forte foco na detecção de dados.

50. OrioleDB

Avalie

OrioleDB é um novo mecanismo de armazenamento para PostgreSQL. Nossos times usam muito o PostgreSQL, mas seu mecanismo de armazenamento foi originalmente projetado para discos rígidos. Embora existam várias opções de ajuste para hardware moderno, pode ser difícil e complicado obter os melhores resultados. OrioleDB aborda esses desafios implementando um mecanismo de armazenamento nativo da nuvem com suporte explícito para unidades de estado sólido (SSDs) e memória de acesso aleatório não volátil (NVRAM). Para experimentar o novo mecanismo, primeiro instale os patches de aprimoramento nos atuais **métodos de acesso à tabela** e depois instale o OrioleDB como uma extensão do PostgreSQL. Acreditamos que o OrioleDB tem um grande potencial para resolver vários **problemas há muito tempo pendentes no PostgreSQL** e encorajamos você a avaliá-lo cuidadosamente.

Ferramentas

Adote

51. Great Expectations
52. k6

Experimente

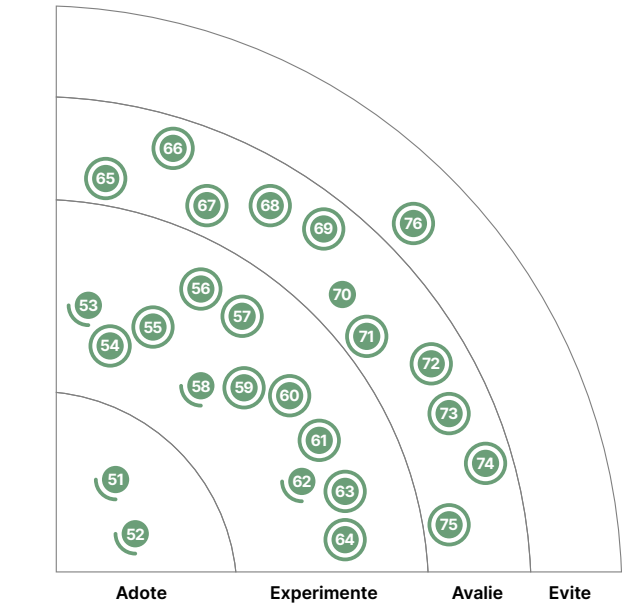
53. Apache Superset
54. AWS Backup Vault Lock
55. AWS Control Tower
56. Clumio Protect
57. Cruft
58. Excalidraw
59. Hadolint
60. Kaniko
61. Kusto Query Language
62. Spectral
63. Serviço de Autorização Declarativa da Styra
64. xbar para monitoramento de compilação

Avalie

65. Clasp
66. Databricks Overwatch
67. dbtvault
68. git-together
69. Harness Cloud Cost Management
70. Infracost
71. Karpenter
72. Mizu
73. Soda Core
74. Teller
75. Xcode Cloud

Evite

76. Serviços online para formatação ou análise de código



Novo Mudança de anel Sem modificação

51. Great Expectations

Adote

Great Expectations tornou-se um padrão sensato para nossos times no espaço de qualidade de dados, e por isso recomendamos adotá-lo — não apenas pela falta de alternativas melhores, mas também porque nossos times relataram ótimos resultados em vários projetos de clientes. Great Expectations é um framework que permite criar controles internos que sinalizam anomalias ou problemas de qualidade em pipelines de dados. Assim como os testes de unidade são executados em um pipeline de compilação, Great Expectations faz asserções durante a execução de um pipeline de dados. Gostamos de sua simplicidade e facilidade de uso — as regras armazenadas em JSON podem ser modificadas por nossa equipe de especialistas em domínio de dados sem necessariamente precisar de habilidades de engenharia de dados.

52. k6

Adote

Desde que o mencionamos pela primeira vez no Radar, o k6 se tornou uma ferramenta obrigatória para testes de desempenho. Continuamos admirando a facilidade em escrever código JavaScript para testes, mas o k6 também tem um construtor de testes de baixo código que torna ainda mais fácil brincar com a ferramenta. A documentação mostra como é simples adicionar testes de desempenho a um pipeline em várias ferramentas de CI/CD. Nossos times consideram fácil integrar ferramentas de visualização como Grafana e New Relic, que os ajudam a ajustar a infraestrutura e os aplicativos. A facilidade de desenvolvimento e o ecossistema tornam k6 uma opção atraente para investigar o comportamento de um sistema sob carga pesada.

53. Apache Superset

Experimente

Apache Superset é uma ótima ferramenta de business intelligence (BI) para exploração e visualização de dados ao trabalhar com grandes configurações de data lake e data warehouse. Suporta várias fontes de dados — incluindo AWS Redshift, BigQuery, Azure MS SQL, Snowflake e ClickHouse. Além disso, você não precisa ser uma pessoa engenheira de dados para usá-la, já que a ferramenta destina-se a beneficiar todas as pessoas engenheiras que exploram dados em seu trabalho diário. Para casos de uso exigentes, achamos fácil dimensionar o Superset implantando-o em um cluster Kubernetes. Desde a última vez que falamos sobre isso no Radar, o Superset se formou como um produto Apache e temos visto grande sucesso em vários projetos.

54. AWS Backup Vault Lock

Experimente

Ao implementar uma recuperação de desastres robusta, segura e confiável, é necessário garantir que os backups não possam ser excluídos ou alterados antes de expirarem, de forma maliciosa ou acidental. Anteriormente, com o AWS Backup, essas políticas e garantias precisavam ser implementadas manualmente. Recentemente, a AWS adicionou o recurso Vault Lock para garantir que os backups sejam imutáveis e invioláveis. AWS Backup Vault Lock aplica políticas de retenção e exclusão e impede que mesmo pessoas usuárias com privilégios administrativos de alterar ou excluir arquivos de backup. Isso provou ser uma adição valiosa e preenche um espaço anteriormente vazio.

55. AWS Control Tower

Experimente

O gerenciamento de contas com múltiplos times é um desafio na AWS, especialmente a configuração e governança. **AWS Control Tower** é uma tentativa de enfrentar esse desafio. Nossa equipe tem relatado bons resultados usando-o para gerenciar contas e controle de acesso para vários times da organização por meio de um único local centralizado.

56. Clumio Protect

Experimente

Tivemos sucesso com o **Clumio Protect** para fazer backup de dados da AWS, principalmente S3. Uma solução SaaS comercial, o Clumio Protect também pode fazer backup de uma variedade de outros serviços da AWS e armazenar os dados offline, onde não podem ser acessados pela Internet. Nossos times responsáveis por lidar com proteção e recuperação de dados em grande escala consideram o Clumio Protect fácil de configurar e manter, além de superar em muito o serviço nativo do AWS Backup quando os buckets do S3 são particularmente grandes.

57. Cruft

Experimente

Temos falado sobre **modelos de serviço personalizados** desde que identificamos **microsserviços** como uma técnica. Se uma organização se propõe a criar uma coleção de pequenos serviços que podem ser desenvolvidos, compilados, implantados e operados de forma independente, mas consistente, faz sentido dar aos times um ponto de partida sólido e alinhado ao padrão. No entanto, um dos problemas persistentes com essa abordagem é que, à medida que o modelo evolui em resposta às mudanças nos requisitos técnicos e de negócios, os projetos baseados em versões mais antigas do modelo ficam desatualizados. A adaptação para melhorias de modelo em um projeto estabelecido torna-se uma grande dor. **Cruft** tenta resolver esse problema fornecendo ferramentas para identificar e corrigir diferenças entre um projeto local e a versão mais atualizada do repositório usado como modelo. A ferramenta combina o mecanismo de modelagem **Cookiecutter** com git hashes para identificar e aplicar alterações nos modelos. Considere-o um gerenciador de pacotes para um boilerplate de projeto. Manter os modelos atualizados é um problema notoriamente difícil e antigo. Então, para nós, a solução Cruft parece boa demais para ser verdade. Com base no feedback inicial de nossa equipe, no entanto, o Cruft realmente funciona e facilita a vida de quem constrói e mantém serviços. Observamos com ansiedade como se comportará a longo prazo, mas por enquanto vale a pena checar esta ferramenta potencialmente útil.

58. Excalidraw

Experimente

Continuamos a ouvir relatos entusiasmados sobre o **Excalidraw** de nossos times, mas nossa advertência anterior sobre segurança continua valendo. Excalidraw é uma ferramenta de desenho online simples, mas poderosa. Às vezes, os times precisam apenas de uma imagem rápida em vez de um diagrama formal. Para times remotos, Excalidraw oferece uma maneira rápida de criar e compartilhar diagramas. Nossos times também gostam da aparência “lo-fi” dos diagramas que podem produzir, que lembra os diagramas de quadro branco que produziriam se estivessem trabalhando presencialmente. Em relação à segurança, no momento da redação deste blip, qualquer pessoa que tenha o link pode ver seus diagramas. Observe, no entanto, que a versão paga do Excalidraw fornece autenticação adicional e existem opções para executar um servidor localmente.

59. Hadolint

Experimente

Gostamos de divulgar as ferramentas de linting que realmente podem ajudar você a encontrar problemas, em vez de apenas exacerbar disputas de estilo na equipe. **Hadolint** é uma dessas ferramentas — ajuda a encontrar problemas comuns em Dockerfiles. Achamos que é uma opção rápida, precisa e com boa documentação. A ferramenta explica como corrigir um problema e por que é um problema em primeiro lugar, levando os autores do Dockerfile a adotar boas práticas. Aliás, o Hadolint é construído em cima do **ShellCheck**, que recomendamos por si só para verificar seus scripts de shell.

60. Kaniko

Experimente

A maioria das ferramentas e plataformas de pipeline de CI/CD atuais são construídas em contêineres como tempos de execução. Muitos de nossos times estão usando **Kaniko** para criar imagens de contêiner a partir desses pipelines baseados em contêiner. Isso vem como parte de uma tendência de distanciamento do **Docker** como padrão para execução de contêiners. Com Kaniko, você pode construir suas imagens sem usar um daemon do Docker. Isso ajuda a evitar o problema de segurança do modo "privilegiado" do Docker, que seria necessário para qualquer atividade "Docker-in-Docker". Além disso, você não precisa presumir que seu pipeline tenha acesso a um daemon do Docker em primeiro lugar, o que não pode mais ser assumido como verdade e geralmente requer configuração extra.

61. Kusto Query Language

Experimente

À medida que o trabalho com dados se torna mais comum, continuamos observando ferramentas que tentam aprimorar a linguagem SQL. **Kusto Query Language** (KQL) é um exemplo. KQL foi criada pelo Azure e traz modularidade, encapsulamento, composição, reutilização, extensibilidade e dinamismo para consultas relacionais. Nossos times gostam bastante de sua interatividade: você pode canalizar uma consulta para o operador de renderização e ver um gráfico instantaneamente. Você também pode combinar esses gráficos em painéis e obter insights de logs para executivos em minutos. Embora a linguagem KQL esteja atualmente limitada ao **Azure Data Explorer**, prevemos que a mudança no sentido de aprimorar o SQL para obter uma melhor operabilidade de dados não vai parar.

62. Spectral

Experimente

Spectral é um linter para JSON/YAML com ênfase nas especificações OpenAPI e AsyncAPI. É fornecido com um conjunto abrangente de regras prontas para usar para essas especificações, que podem evitar dores de cabeça para pessoas desenvolvedoras ao projetar e implementar APIs ou em colaboração orientada a eventos. Essas regras verificam as especificações adequadas dos parâmetros da API ou a existência de uma declaração de licença na especificação, entre outras coisas. O **CLI** facilita a incorporação do Spectral no desenvolvimento local e em pipelines de CI/CD, e a **API JavaScript** suporta casos de uso mais avançados. O **site do GitHub** tem links para conjuntos de regras reais disponíveis publicamente de empresas como Adidas, dando aos times uma vantagem inicial na adoção de suas próprias regras de linting.

63. Serviço de Autorização Declarativa da Styra

Experimente

Serviço de Autorização Declarativa da Styra (DAS) é uma ferramenta de governança e automação para gerenciar Open Policy Agent (OPA) em escala. Construída pelos criadores do OPA, a ferramenta nos permite implantar políticas em "sistemas", incluindo clusters Kubernetes, repositórios de código de infraestrutura, namespaces e mais. Mais importante, permite a análise em tempo real das decisões tomadas por um agente OPA, juntamente com a capacidade de repetição para depuração e investigação de cenários hipotéticos para alterações da política. Styra DAS também vem com um log de auditoria que pode ajudar as equipes de segurança com relatórios históricos.

64. xbar para monitoramento de compilação

Experimente

Em times remotos, sentimos falta de ter um monitor de compilação dedicado na sala. Infelizmente, as ferramentas de integração contínua (CI) mais recentes não têm suporte para o antigo formato CCTray. Como resultado, compilações quebradas nem sempre são recuperadas tão rapidamente quanto gostaríamos. Para resolver esse problema, muitos de nossos times começaram a usar xbar para monitoramento de compilação. Com xbar, pode-se executar um script para buscar o status da compilação, exibindo-o na barra de menus. Também pode ser programado para rastrear outras métricas do time, como expirações de credenciais pendentes ou até que ponto a versão de produção está atrasada em relação à versão do teste de aceitação do usuário (UAT). Evidentemente, xbar atende a um propósito mais geral, mas resolve um problema imediato e emergente causado pelo trabalho remoto. Rumps, entre outras ferramentas, pode resolver o mesmo problema.

65. Clasp

Avalie

Infelizmente, uma grande parte do mundo ainda funciona em planilhas e continuará a fazê-lo. As planilhas são a opção definitiva para permitir que qualquer pessoa construa pequenas ferramentas personalizadas sob medida para suas necessidades específicas. No entanto, quando você deseja aprimorá-las com um nível de lógica que requer código "real", a natureza de baixo código das planilhas pode se tornar uma restrição. Se você trabalha em uma empresa que, como a Thoughtworks, usa o G-Suite do Google, Clasp habilita você a aplicar pelo menos algumas práticas de Entrega Contínua ao código Apps Script. Você pode escrever o código fora do projeto Apps Script, criando opções para teste, controle de versão e pipelines de compilação. A ferramenta ainda permite que você use TypeScript. Clasp já existe há algum tempo e você não deve esperar um ambiente de programação com todos os confortos usuais, mas pode melhorar muito a experiência de usar o Apps Script.

66. Databricks Overwatch

Avalie

Databricks Overwatch é um projeto do Databricks Labs que habilita os times a analisar várias métricas operacionais de cargas de trabalho do Databricks em relação a custo, governança e desempenho, com suporte para executar experimentos "e se?". É essencialmente um conjunto de pipelines de dados que preenchem tabelas no Databricks, podendo ser analisadas usando ferramentas como notebooks. Overwatch é uma ferramenta poderosa, no entanto, ainda está em seus estágios iniciais e pode exigir algum esforço para configuração — nosso uso exigiu que pessoas arquitetas de soluções Databricks ajudassem a configurar e preencher uma tabela de referência de preços para cálculos de custo —, mas esperamos que a adoção fique mais fácil com o tempo. O nível

de análise possibilitado pelo Overwatch é mais profundo do que o permitido pelas ferramentas de análise de custos das provedoras de nuvem. Por exemplo, pudemos analisar o custo de falhas de tarefas — reconhecendo que falhar rapidamente economiza dinheiro em comparação com tarefas que falham somente perto da etapa final — e dividir o custo por vários agrupamentos (espaço de trabalho, cluster, tarefa, notebook, equipe). Também apreciamos a visibilidade operacional aprimorada, já que pudemos auditar facilmente os controles de acesso em torno das configurações do cluster e analisar as métricas operacionais, como encontrar o notebook em execução mais antigo ou o maior volume de leitura/gravação. Overwatch pode analisar dados históricos, mas seu modo em tempo real permite alertas, o que ajuda você a adicionar controles apropriados às suas cargas de trabalho do Databricks.

67. dbtvault

Avalie

Data Vault 2.0 é uma metodologia de modelagem de dados e um padrão de design destinado a melhorar a flexibilidade de data warehouses em comparação com outras abordagens populares de modelagem. O Data Vault 2.0 pode ser aplicado a qualquer armazenamento de dados, como Snowflake ou Databricks. Ao implementar warehouses do Data Vault, consideramos o pacote dbtvault para dbt uma ferramenta útil. O dbtvault fornece um conjunto de modelos jinja que geram e executam os scripts ETL necessários para preencher um warehouse do Data Vault. Embora o dbtvault tenha algumas arestas para aparar — não tem suporte para impor unicidade implícita ou executar cargas incrementais — no geral, preenche um nicho e requer configuração mínima para começar.

68. git-together

Avalie

Estamos sempre procurando maneiras de remover pequenos atritos na programação em pares, e é por isso que estamos felizes com git-together, uma ferramenta escrita em Rust que simplifica a atribuição do git commit durante o pareamento. Ao fazer um alias de `git-together` como `git`, a ferramenta permite que você adicione extensões ao `git config` que capturam informações do committer, fazendo um alias de cada committer por suas iniciais. Alterar pares (ou alternar para programação solo ou mob) requer que você execute `git with`, seguido das iniciais do par (por exemplo: `git with bb cc`), permitindo que você retome seu fluxo de trabalho git posteriormente. Toda vez que você fizer um commit, o git-together irá rotacionar o par como autor oficial que o git armazena, e irá adicionar automaticamente quaisquer outros autores ao final da mensagem do commit. A configuração pode ser armazenada no repositório, permitindo que o git-together funcione automaticamente após a clonagem de um repositório.

69. Harness Cloud Cost Management

Avalie

Harness Cloud Cost Management é uma ferramenta comercial que funciona em todas as três principais provedoras de nuvem e seus Kubernetes gerenciados para ajudar a visualizar e gerenciar os custos da nuvem. O produto calcula uma pontuação de eficiência de custo analisando recursos ociosos, bem como recursos não alocados a nenhuma carga de trabalho e usa tendências históricas para ajudar a otimizar a alocação de recursos. Os painéis destacam os picos de custo e permitem que uma pessoa usuária registre anomalias inesperadas, que são então inseridas em seu algoritmo de reforço de aprendizado em torno da detecção de anomalias. O Cloud Cost Management pode recomendar ajustes aos limites de uso de memória e CPU, com opções para otimizar custo ou desempenho. “Perspectives” permite agrupar custos com base em filtros definidos

organizacionalmente (que podem corresponder a unidades de negócios, equipes ou produtos) e automatizar a distribuição de relatórios para dar visibilidade aos gastos na nuvem. Acreditamos que o Cloud Cost Management oferece um conjunto atraente de recursos para ajudar as organizações a amadurecer suas práticas de FinOps.

70. Infracost

Avalie

Continuamos a ver as organizações migrarem para a nuvem sem entender adequadamente como rastrearão os gastos contínuos. Anteriormente, falamos sobre **custo de execução como função de aptidão arquitetural**, e **Infracost** é uma ferramenta que visa tornar essas compensações de custo de nuvem visíveis nos pull requests de arquivos Terraform. 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 prontas para uso. Também fornece uma API pública que pode ser consultada para dados de custo atuais. Continuamos observando com entusiasmo seu potencial, especialmente quando se trata de obter melhor visibilidade de custos no IDE.

71. Karpenter

Avalie

Um dos recursos fundamentais do **Kubernetes** é a habilidade de iniciar automaticamente novos pods quando se faz necessária uma capacidade adicional, e desligá-los quando as cargas diminuem. Esse escalonamento automático horizontal é um recurso útil, mas só pode funcionar se os nós necessários para hospedar os pods já existirem. Embora o **Cluster Autoscaler** possa fazer alguma expansão de cluster rudimentar desencadeada por falhas de pod, sua flexibilidade é limitada. **Karpenter**, no entanto, é um escalonador automático **Operador Kubernetes** de código aberto com mais inteligência integrada: analisa as cargas de trabalho atuais e as restrições de agendamento de pod para selecionar automaticamente um tipo de instância apropriado e, em seguida, iniciá-lo ou interrompê-lo conforme necessário. Karpenter é um operador com o mesmo espírito de ferramentas como **Crossplane**, que pode provisionar recursos de nuvem fora do cluster. Karpenter é um companheiro atraente para os serviços de auto escalonamento oferecidos nativamente por provedores de nuvem com seus clusters Kubernetes gerenciados. Por exemplo, a AWS agora oferece suporte ao Karpenter como uma alternativa de primeira classe em seu serviço EKS Cluster Autoscaler.

72. Mizu

Avalie

Mizu é um visualizador de tráfego de API para **Kubernetes**. Ao contrário de outras ferramentas, Mizu não requer instrumentação ou alterações de código. É executado como um **DaemonSet** para injetar um contêiner a nível de nó em seu cluster Kubernetes, e executa operações semelhantes a tcpdump. Achamos útil como ferramenta de depuração, pois pode observar todas as comunicações da API em vários protocolos (REST, gRPC, **Kafka**, AMQP e **Redis**) em tempo real.

73. Soda Core

Avalie

Soda Core é uma ferramenta de qualidade e observabilidade de dados de código aberto. Falamos sobre **Great Expectations** anteriormente no Radar, e Soda Core é uma alternativa com uma diferença fundamental — você expressa as validações de dados em uma DSL chamada

SodaCL (anteriormente chamada de Soda SQL) em oposição às funções do Python. Depois que as validações são escritas, podem ser executadas como parte de um pipeline de dados ou agendadas para serem executadas programaticamente. À medida que nossos negócios tornam-se cada vez mais orientados a dados, é fundamental manter a qualidade dos dados, e recomendamos que você avalie o Soda Core.

74. Teller

Avalie

Teller é um gerenciador de segredos universal de código aberto para pessoas desenvolvedoras que garante que as variáveis de ambiente corretas sejam definidas ao iniciar uma aplicação. No entanto, não é um cofre em si — é uma ferramenta CLI que se conecta a uma variedade de fontes, desde provedoras de segredos em nuvem a soluções de terceiras, como HashiCorp Vault para arquivos de ambiente local. Teller tem funcionalidade adicional de buscar por segredos guardados em cofre em seu código, redigir segredos dos logs, detectar diferenças entre provedoras de segredos e sincronizar entre elas. Dada a sensibilidade de acessar segredos, não podemos enfatizar o suficiente a necessidade de proteger a cadeia de fornecimento para dependências de código aberto, mas apreciamos a facilidade de uso da CLI em ambientes de desenvolvimento local, pipelines de CI/CD e automação de implantação.

75. Xcode Cloud

Avalie

Xcode Cloud ié uma ferramenta de CI/CD incorporada ao Xcode e usada para criar, testar e implantar aplicativos da Apple. Fornece uma experiência integrada com ferramentas familiares para pessoas desenvolvedoras da Apple, como Xcode, App Store Connect e TestFlight. Com base na experiência de nossa equipe, ela faz um bom trabalho ao simplificar a configuração do pipeline e provisionar perfis e certificados. Esta ferramenta é bastante recente e a maioria de nossos times de desenvolvimento móvel ainda está usando o mais maduro Bitrise. Ainda assim, achamos que vale a pena avaliar e acompanhar seu progresso.

76. Serviços online para formatação ou análise de código

Evite

Anteriormente, chamamos a atenção para dados de produção em ambientes de teste, e agora queremos destacar outra prática comum que precisa ser abordada com cuidado ou até mesmo interrompida por completo: serviços online para formatação ou análise de código. Existem muitos sites úteis para formatação ou análise de formatos como JSON e YAML, bem como sites que abordam tutoriais de código ou produzem métricas de código online. É necessário muito cuidado ao usá-los. Colar um bloco de JavaScript, JSON ou similar em um site desconhecido pode facilmente criar problemas de segurança e privacidade, além de exportar dados pessoais inadvertidamente para uma jurisdição diferente. Esses sites nunca devem ser usados com dados de produção e devem ser abordados com cautela em todas as outras circunstâncias.

Linguagens e Frameworks

Adote

77. io-ts
78. Kotest
79. NestJS
80. React Query
81. Swift Package Manager
82. Yjs

Experimente

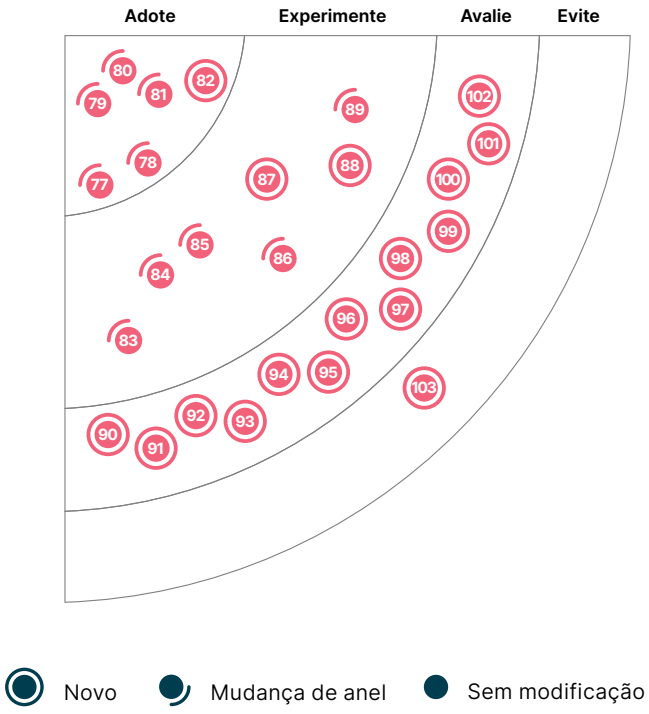
83. Azure Bicep
84. Camunda
85. Gradle Kotlin DSL
86. Jetpack Media3
87. Ladle
88. Moshi
89. Svelte

Avalie

90. Aleph.js
91. Astro
92. BentoML
93. Carbon Aware SDK
94. Cloudscape
95. Connect
96. Cross device SDK
97. Cypress Component Testing
98. JobRunr
99. Million
100. Soketi
101. Stable Diffusion
102. Synthetic Data Vault

Evite

103. Carbon



77. io-ts

Adote

Nossos times desenvolvendo em TypeScript estão considerando o valor de io-ts inestimável, especialmente ao interagir com APIs que acabam resultando na criação de objetos com tipos específicos. Ao trabalhar com o TypeScript, colocar dados dentro dos limites do sistema de tipos (ou seja, das APIs mencionadas) pode levar a erros de tempo de execução que podem ser difíceis de encontrar e depurar. O io-ts preenche a lacuna entre a verificação de tipo em tempo de compilação e o consumo de dados externos em tempo de execução, fornecendo funções de codificação e decodificação. Dadas as experiências de nossos times e a elegância de sua abordagem, achamos que vale a pena adotar o io-ts.

78. Kotest

Adote

Kotest (anteriormente KotlinTest) é uma ferramenta de teste independente para o ecossistema Kotlin que é amplamente usada entre nossos times em várias implementações Kotlin — nativas, JVM ou JavaScript. Suas principais vantagens são o fato de oferecer uma variedade de estilos de teste para estruturar suítes de teste e um conjunto abrangente de matchers, que permitem testes expressivos em uma DSL interna elegante. Além de seu suporte para testes baseados em propriedades, nossos times gostam do sólido plug-in IntelliJ e da comunidade de suporte. Muitas de nossas pessoas desenvolvedoras o consideram sua primeira escolha e recomendam a quem ainda usa JUnit em Kotlin considerem mudar para o Kotest.

79. NestJS

Adote

No passado, alertamos sobre o uso excessivo de Node.js e ainda mantemos cautela em relação aos motivos para escolhê-lo. No entanto, em cenários adequados nos quais o Node.js é necessário para criar aplicações back-end, nossos times relatam que o NestJS é uma boa opção para habilitar pessoas desenvolvedoras a criar aplicaçõs escaláveis, pouco acopladas e de fácil manutenção nas empresas. NestJS é um framework que utiliza TypeScript como padrão, o que torna o desenvolvimento de aplicações Node.js mais seguro e menos propenso a erros. O NestJS é opinativo e vem com princípios SOLID e uma arquitetura inspirada em Angular pronta para uso.

80. React Query

Adote

React Query é frequentemente descrito como a biblioteca de busca de dados que faltava para React. Buscar, armazenar em cache, sincronizar e atualizar o estado do servidor é um requisito comum em muitas aplicações React e, embora os requisitos sejam bem compreendidos, obter a implementação correta é notoriamente difícil. O React Query fornece uma solução direta usando hooks. Funciona em conjunto com bibliotecas de busca de dados assíncronas existentes como axios, Fetch e GraphQL uma vez que são construídos com base em promises. Como uma pessoa desenvolvedora de aplicativos, você simplesmente passa uma função que resolve seus dados e deixa todo o resto para o framework. Gostamos do fato de ser pronto para uso, mas ainda oferecer muitas opções de configuração quando necessário. As ferramentas de desenvolvimento, infelizmente ainda não disponíveis para React Native, também ajudam as pessoas desenvolvedoras novas no framework a entender seu funcionamento. Para React Native, você pode usar um plugin de ferramentas de desenvolvimento de terceiros utilizando Flipper. Em nossa experiência, a versão 3 do React Query trouxe a estabilidade necessária para ser utilizada em produção com nossas clientes.

81. Swift Package Manager

Adote

Quando lançado em 2014, o Swift não vinha com um gerenciador de pacotes. Mais tarde, o Swift Package Manager foi criado como um projeto de código aberto oficial da Apple, e essa solução continuou a se desenvolver e amadurecer. Nossos times confiam cada vez mais no SwiftPM já que a maioria dos pacotes podem ser incluídos por meio dele, e os processos para criadores e consumidores de pacotes foram simplificados. No Radar anterior, recomendamos experimentar, mas agora acreditamos que faz sentido selecioná-lo como padrão ao iniciar novos projetos. Para projetos existentes usando ferramentas como CocoaPods ou Carthage, pode valer a pena fazer um experimento rápido para avaliar o nível de esforço em migrar e verificar se todas as dependências estão disponíveis.

82. Yjs

Adote

Os algoritmos de tipo de dados replicados livres de conflito (CRDT) são comprovadamente capazes de distribuir e mesclar alterações automaticamente entre pares sem conflitos. Mas, na prática, mesmo para dados pequenos o suficiente, esses algoritmos geralmente exigem uma quantidade significativa de memória para rastrear todas as alterações feitas por diferentes pares, tornando-os, então, impraticáveis. Yjs é uma implementação de CRDT cuidadosamente otimizada que mantém o consumo de memória em um nível razoável para grandes conjuntos de dados e milhões de modificações. Também fornece ligações para editores de texto populares, o que reduz bastante o custo de criação de ferramentas colaborativas.

83. Azure Bicep

Experimente

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 oferece suporte a templates parametrizados reutilizáveis para definições de recursos modulares. Uma extensão do Visual Studio Code fornece segurança de tipo instantânea, intellisense e verificação de sintaxe, enquanto o compilador permite a transpilação bidirecional de e para templates do Azure Resource Manager (ARM). O DSL orientado a recursos do Bicep e a integração nativa com o ecossistema do Azure o tornam uma opção atraente para o desenvolvimento de infraestrutura do Azure.

84. Camunda

Experimente

Desde que mencionamos Camunda pela última vez, vimos muitos de nossos times e clientes usarem a plataforma, tornando-a um dos nossos mecanismos de fluxo de trabalho preferidos nos casos em que um mecanismo de fluxo de trabalho é um bom ajuste para o problema em questão. Camunda oferece mecanismos de fluxo de trabalho e decisão que podem ser integrados como uma biblioteca em seu código Java. Isso facilita o teste, o versionamento e a refatoração de fluxos de trabalho, aliviando algumas das desvantagens de outros mecanismos de fluxo de trabalho baixo código. Vimos Camunda ser usado até mesmo em ambientes com requisitos de alto desempenho. Os times também gostam da facilidade de integração com Spring Boot e sua interface de usuário agradável.

85. Gradle Kotlin DSL

Experimente

Anteriormente, falamos sobre o plugin Android Gradle Kotlin DSL, ou Gradle Kotlin DSL, que adicionou suporte para **Kotlin** Script como alternativa ao **Groovy** para projetos Android usando scripts de compilação **Gradle**. O objetivo de substituir o Groovy pelo Kotlin é fornecer melhor suporte para refatoração e edição mais simples em IDEs e, em última análise, produzir código que seja mais fácil de ler e manter. Para times que já usam Kotlin, isso também significa trabalhar na compilação em uma linguagem familiar. Nesse momento, sugerimos experimentar Kotlin DSL como uma linguagem alternativa aos projetos Groovy para Gradle em geral, especialmente se você tiver scripts de compilação Gradle grandes ou complexos. Muitos IDEs agora incluem suporte para a migração de projetos existentes. Algumas ressalvas permanecem, e sugerimos verificar a **documentação** para obter os detalhes mais atualizados, incluindo os pré-requisitos. Um de nossos times tinha um script de compilação de pelo menos sete anos e 450 linhas que foi migrado com sucesso em poucos dias.

86. Jetpack Media3

Experimente

O Android tinha 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 tinham que escolher qual biblioteca usar, como também tinham que lidar com a escrita de adaptadores ou outro código de conexão quando os recursos de várias APIs eram necessários. **Jetpack Media3** é uma API que usa áreas comuns de funcionalidade das APIs existentes — incluindo interface de usuário, reprodução e manipulação de sessão de mídia — e as combina 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. Após uma fase de acesso antecipado, o Media3 está agora em beta. Embora seu primeiro lançamento esteja próximo, já tivemos experiências positivas ao usá-lo em aplicativos.

87. Ladle

Experimente

À medida que o **Storybook** cresceu em popularidade, tornou-se cada vez mais um Frankenstein. Se tudo o que você realmente quer é isolar e testar seus componentes do React UI, então **Ladle** é a alternativa. Ladle suporta a maioria das APIs do Storybook (os arquivos MDX ainda não são suportados) e pode ser usado como um substituto imediato. É leve e tem melhor integração com **Vite**. Também fornece APIs simples e limpas que podem ser facilmente integradas a outros frameworks de teste.

88. Moshi

Experimente

Temos ouvido que nossos times baseados em **Kotlin** estão buscando alternativas para frameworks Java, como GSON, ao lidar com JSON. Embora já exista há algum tempo, o **Moshi** surgiu como o framework preferido para muitas dessas equipes. É fácil migrar do GSON, e Moshi fornece suporte nativo para tipos não nulos de Kotlin e parâmetros padrão. Moshi torna o trabalho com JSON mais rápido e fácil. Se você estiver usando um framework Java de dentro do Kotlin para lidar com JSON, recomendamos experimentar o Moshi.

89. Svelte

Experimente

Entre os frameworks de componentes web, **Svelte** se destaca ao mover a reatividade do navegador para o compilador. Em vez de otimizar as atualizações do DOM usando um DOM virtual e truques de otimização do navegador, Svelte compila seu código em um código JavaScript clássico, sem frameworks, que atualiza o DOM cirúrgica e diretamente. Além dos benefícios de desempenho em tempo de execução, isso também permite que o Svelte otimize a quantidade de código que o navegador precisa baixar sem sacrificar recursos para pessoas desenvolvedoras. Além disso, é comprovadamente eficiente e com baixo consumo de bateria em aplicações web em dispositivos móveis, pois menos código precisa ser executado no próprio navegador. Além dos benefícios de desempenho, nossos times apreciaram sua curva de aprendizado amigável e os benefícios de manutenção decorrentes de **escrever menos código**. O próprio Svelte é apenas o framework de componentes, mas o **SvelteKit** adiciona recursos para construir aplicações web completas.

90. Aleph.js

Avalie

Certamente não faltam frameworks para construir aplicações web em JavaScript/**TypeScript**. Apresentamos muitos deles no Radar, mas o que diferencia o **Aleph.js** nesse campo populoso é o fato de ter sido desenvolvido para ser executado em **Deno**, o novo runtime do lado do servidor criado pelo desenvolvedor original do **Node**. Isso coloca o Aleph.js em uma base moderna que endereça várias deficiências e problemas com o Node. Aleph.js ainda é novo — está se aproximando da versão 1.0 no momento da redação deste blip — mas já oferece uma sólida experiência de desenvolvimento, incluindo hot module replacement. Com o Deno agora muito além de sua **versão 1.0**, esta é uma escolha moderna para projetos que podem lidar com riscos.

91. Astro

Avalie

É difícil de acreditar, mas em 2022, a comunidade de desenvolvimento continua lançando novos frameworks interessantes para construir aplicações web. **Astro** é um framework recente de aplicações com várias páginas e de código aberto, que renderiza HTML no servidor e minimiza a quantidade de JavaScript enviada pela rede. Astro parece particularmente adequado para sites orientados a conteúdo que consomem muitas fontes diferentes. Gostamos do fato de que, embora o Astro incentive o envio apenas de HTML, ainda suporta — quando apropriado — selecionar componentes ativos escritos no framework JavaScript front-end de sua escolha. Astro faz isso por meio de sua **arquitetura da ilha**. Ilhas são regiões de interatividade dentro de uma única página onde o JavaScript necessário é baixado apenas quando preciso. Astro é relativamente novo, mas parece oferecer suporte a um ecossistema crescente de pessoas desenvolvedoras e código. Vale acompanhar seu desenvolvimento.

92. BentoML

Avalie

BentoML é um framework python-primeiro para servir modelos de aprendizado de máquina em produção em escala. Os modelos fornecidos são independentes de seu ambiente; todos os artefatos de modelo, código-fonte e dependências são encapsulados em um formato autocontido chamado Bento. É como ter seu modelo "como um serviço". Pense no BentoML como o **Docker** para modelos de ML: gera imagens de VM com APIs pré-programadas prontas para implantação e inclui recursos

que facilitam para testar essas imagens. BentoML pode ajudar a acelerar o esforço inicial de desenvolvimento, facilitando o início dos projetos, e é por isso que o incluímos em Avalie.

93. Carbon Aware SDK

Avalie

Ao olhar para a redução da pegada de carbono de uma aplicação – as emissões de dióxido de carbono causadas indiretamente pela execução do software – a atenção geralmente é direcionada para tornar o software mais eficiente. O pensamento é claro: um software mais eficiente precisa de menos eletricidade e menos servidores, reduzindo as emissões da geração de eletricidade e fabricação dos servidores. Uma estratégia adicional é tornar a aplicação consciente do carbono. Isso ocorre porque a mesma carga de trabalho nem sempre tem a mesma pegada de carbono. Por exemplo, quando executada em um data center em um clima mais frio, é necessária menos energia para o ar condicionado; ou, quando executada em um momento em que há mais energia renovável disponível (mais sol, ventos mais fortes), é necessária menos eletricidade de fontes baseadas em carbono. Com **Carbon Aware SDK**, as pessoas engenheiras de software podem consultar fontes de dados para descobrir opções menos intensivas em carbono para uma determinada carga de trabalho e, em seguida, movê-la para um local diferente ou executá-la em um horário diferente. Isso faz sentido para grandes cargas de trabalho que não são sensíveis ao tempo nem à latência, como treinar um modelo de aprendizado de máquina. Embora o SDK e as fontes de dados disponíveis ainda não sejam muito abrangentes, acreditamos que é hora de começar a analisar como podemos tornar nossos sistemas conscientes de carbono.

94. Cloudscape

Avalie

Cloudscape é um sistema de design de código aberto que não apenas possui um rico conjunto de componentes, mas também 35 padrões de interação e representação de conteúdo. Além disso, ele usa **design tokens** para criação de temas e fornece wrappers de elementos para todos os componentes, o que simplifica bastante o teste unitário. Isso o destaca em relação a outros sistemas de design por aí.

95. Connect

Avalie

Connect é uma família de bibliotecas para criar APIs HTTP - e gRPC - compatíveis com navegadores. Semelhante ao gRPC, você escreve o esquema de buffer de protocolo e implementa a lógica do aplicativo, e o Connect gera código para lidar com empacotamento, roteamento, compactação e negociação de tipo de conteúdo. No entanto, o Connect tenta melhorar o gRPC de várias maneiras. Isso inclui suporte nativo para gRPC-Web sem um proxy de tradução; interoperabilidade com roteadores ou middleware de terceiros, porque **connect-go** é construído sobre net/http (diferente do grpc-go); e clientes type-safe totalmente gerados com a ergonomia do código feito à mão. Preferimos principalmente o REST e não somos grandes fãs da abordagem RPC para criar APIs. Dito isso, o Connect parece resolver algumas de nossas preocupações com RPCs e incentivamos você a avaliá-lo.

96. Cross device SDK

Avalie

À medida que os dispositivos inteligentes continuam a se incorporar em nossas vidas, estamos começando a ver novos casos de uso surgindo em vários dispositivos. O exemplo clássico é um

texto que começamos a ler em um telefone, mas preferimos terminar em um tablet. Outros exemplos incluem traçar uma rota de ciclismo em um laptop e depois transferir os dados para o computador de uma bicicleta para facilitar a navegação ou usar um telefone celular como webcam. Esses casos de uso exigem tipos muito específicos de recursos, como a descoberta de dispositivos próximos, comunicação segura e sessões em vários dispositivos. A Apple começou a introduzir esses recursos há algum tempo em seus próprios SDKs, e agora o Google lançou a primeira amostra de seu Cross device SDK. Embora a amostra tenha várias limitações – por exemplo, apenas telefones e tablets são suportados e apenas dois dispositivos por vez – a tecnologia é empolgante e pode ser utilizada à medida que é desenvolvida ao longo do tempo.

97. Cypress Component Testing

Avalie

Cypress Component Testing fornece um workbench de componentes testável para criar e testar componentes de interface de usuário rapidamente. Você pode escrever testes de regressão visual de componentes com a mesma API que você escreve testes de interface de usuário de ponta a ponta (E2E). Embora ainda em beta, o teste de componentes será o recurso mais importante no Cypress 10.

98. JobRunr

Avalie

JobRunr é uma biblioteca para processamento de trabalho em segundo plano em Java e uma alternativa ao agendador Quartz. Nossos times gostaram de usar o painel integrado do JobRunr, que é fácil de usar e permite o monitoramento e agendamento de tarefas em segundo plano. JobRunr é de código aberto e gratuito para uso comercial. Para recursos como migração e recuperação de trabalhos, no entanto, você precisa obter uma licença paga.

99. Million

Avalie

Million é uma nova biblioteca JavaScript com DOM virtual. Semelhante a Svelte, utiliza o compilador, Vite, para criar pequenos pacotes JavaScript com desempenho de renderização excepcional. A biblioteca Million é fornecida como um único pacote NPM com vários módulos — incluindo router, jsx-runtime e um módulo para compatibilidade do React para criar aplicações de página única. Embora React tenha popularizado o DOM virtual há uma década, é fascinante ver inovações neste espaço.

100. Soketi

Avalie

Soketi é um servidor WebSockets de código aberto. Se sua aplicação for compatível com o protocolo Pusher, você poderá conectar o Soketi diretamente, pois ele implementa integralmente o Protocolo Pusher v7. Achamos o suporte beta para Cloudflare Workers particularmente interessante porque abre as portas para o uso de WebSockets na borda da rede.

101. Stable Diffusion

Avalie

O DALL·E da OpenAI chamou a atenção de todo mundo com sua capacidade de criar imagens a partir de prompts de texto. Agora, o Stable Diffusion oferece o mesmo recurso, mas, fundamentalmente, é de código aberto. Qualquer pessoa com acesso a uma placa de vídeo poderosa

pode experimentar o modelo e qualquer pessoa com recursos de computação **suficientes** pode recriar o modelo por conta própria. Os resultados são **surpreendentes**, mas também levantam questões significativas. Por exemplo, o modelo é treinado em pares imagem-texto obtidos por meio de uma **ampla coleta da internet** e, portanto, refletirá preconceitos sociais, o que significa que possivelmente irá produzir conteúdo ilegal, perturbador ou, no mínimo, indesejável. Stable Diffusion agora inclui um **classificador de segurança** baseado em IA. No entanto, dada a sua natureza de código aberto, as pessoas podem desabilitar o classificador. Finalmente, artistas notaram que, com as dicas certas, o modelo é capaz de imitar seu estilo artístico. Isso levanta questões sobre as implicações éticas e legais de uma IA capaz de imitar artistas.

102. Synthetic Data Vault

Avalie

Synthetic Data Vault (SDV) é um ecossistema de bibliotecas de geração de dados sintéticos que pode aprender a distribuição de um conjunto de dados para gerar dados sintéticos com o mesmo formato e propriedades estatísticas que a fonte. No passado, falamos sobre as desvantagens de usar **dados de produção em ambientes de teste**. No entanto, as nuances da distribuição de dados em produção dificilmente podem ser replicadas manualmente, resultando em defeitos e surpresas. Acreditamos que o SDV e ferramentas semelhantes podem resolver essa lacuna gerando dados semelhantes à produção para **tabela única**, **tabela múltipla complexa** e **séries temporais multivariadas**. Embora o SDV não seja novo, gostamos bastante e decidimos destacá-lo.

103. Carbon

Evite

Temos visto algum interesse na linguagem de programação **Carbon**. Isso não é uma surpresa: a linguagem tem o apoio do Google e se apresenta como uma sucessora natural do C++. Em nossa opinião, o C++ não pode ser substituído com rapidez suficiente, pois as pessoas engenheiras de software mostraram, nas últimas décadas, que escrever código C++ seguro e livre de erros é extremamente difícil e demorado. Embora Carbon seja um conceito interessante com foco na migração do C++, sem um compilador que esteja funcional, está nitidamente longe de ser utilizável, e existem outras linguagens de programação modernas que são boas escolhas se você deseja migrar do C++. É muito cedo para dizer se o Carbon se tornará o sucessor natural do C++, mas, na perspectiva atual, recomendamos que os times analisem **Rust** e **Go** em vez de adiar uma migração porque estão esperando a chegada do Carbon.

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

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

Assine

A Thoughtworks é uma consultoria global de tecnologia que integra estratégia, design e engenharia para alavancar a inovação digital. Somos mais de 12 mil pessoas distribuídas entre 50 escritórios e em 18 países. Há mais de 25 anos, trabalhamos junto com nossas clientes para criar impacto extraordinário, usando a tecnologia como diferenciador para ajudá-las a resolver problemas de negócio complexos.

/thoughtworks