Technology Radar

Um guia com opiniões firmes sobre as fronteiras da tecnologia

thoughtworks

Volume 25

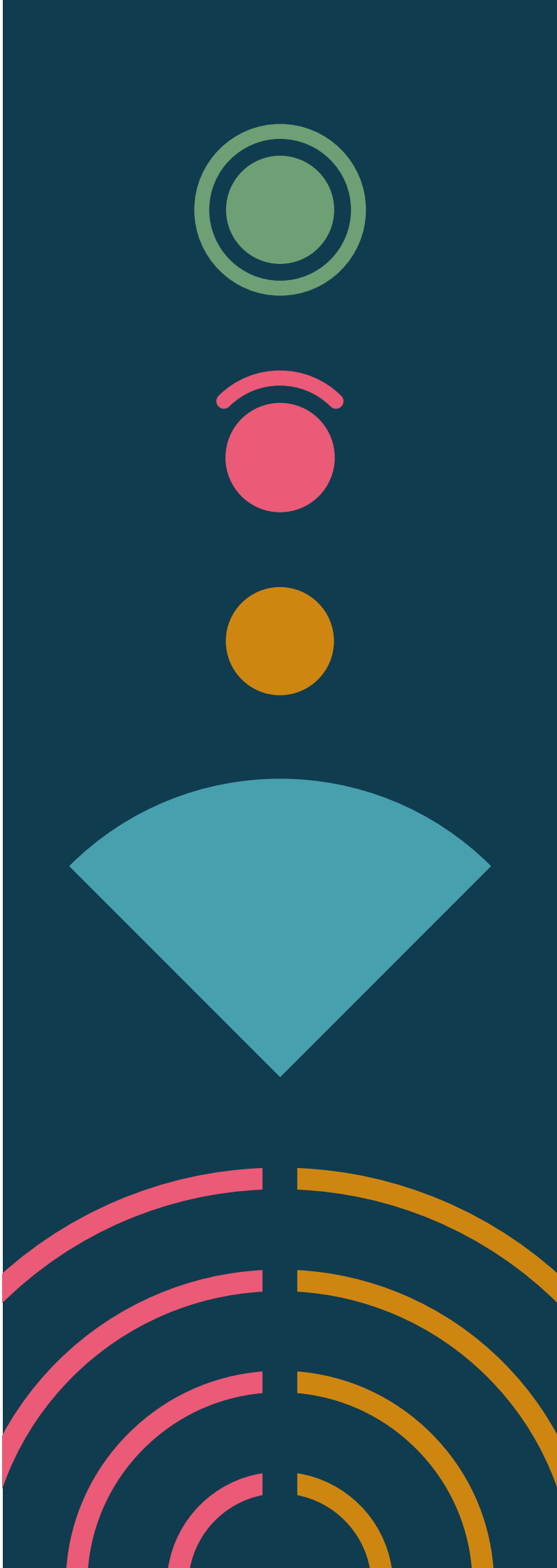
Sobre o Radar

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 e linguagens & 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 deveria adotar esses itens. Nós os usamos quando são apropriados 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.

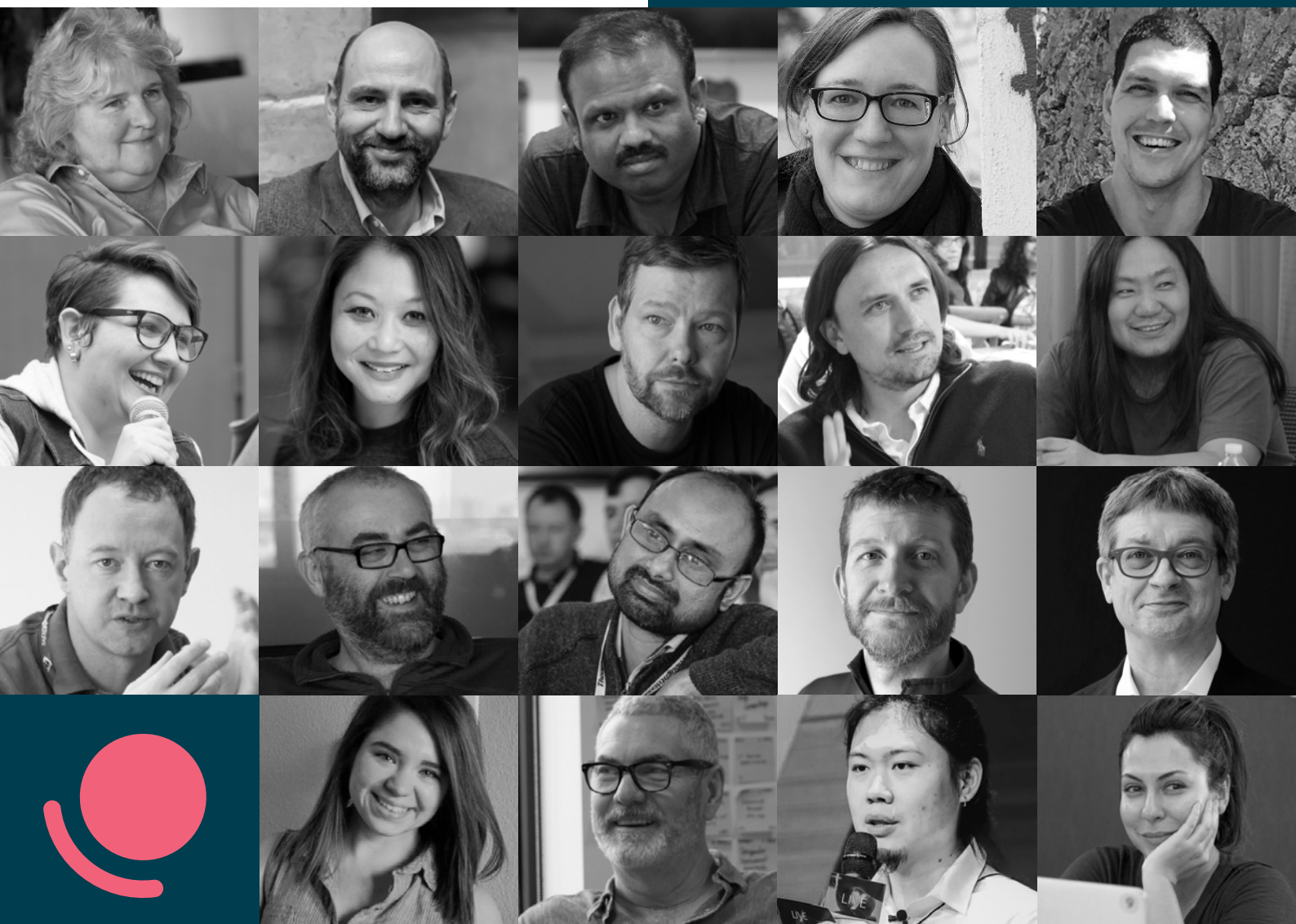
Contribuições

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

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

Tradução: Paula Ribas, Marcelo Viana, Patrick Prado, Ingrid Murielem, Gregorio Melo, Thayse Onofrio, Mauricio Silva, Thiago Gregorio

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



Temas

Adaptando Kafka

Discutimos uma série de tópicos para esta edição do Radar (alguns dos quais eventualmente não entraram no corte final) abordando como as equipes têm usado ferramentas para se adaptar de/para Kafka. Algumas dessas ferramentas permitem interfaces mais tradicionais para o Kafka (como **ksqlDB**, **Confluent Kafka REST Proxy** e Nakadi), enquanto outras são projetadas para fornecer serviços extras, como interfaces gráficas e complementos para orquestração. Suspeitamos que parte da razão por trás desta abundância de ferramentas é a alta complexidade de algumas das partes do Kafka, combinada com o aumento de sua presença em organizações que precisam adaptá-lo a arquiteturas e processos existentes. Algumas equipes acabam tratando **Kafka como um ESB de próxima geração** — um exemplo do tema *Os perigos da conveniência* —, mas outras usam Kafka para fornecer acesso universal a eventos de negócio à medida que acontecem. Essas organizações reconhecem que às vezes é mais fácil ter uma infraestrutura centralizada com adaptações nas arestas, e tentam evitar que as coisas saiam do controle com design e governança cuidadosos. Em todo caso, isso mostra que Kafka continua seguindo em direção ao status de um padrão de fato para mensagens de publicação/assinatura assíncronas em grandes volumes.

Os perigos da conveniência

Um antipadrão tão antigo quanto o Radar é a tendência das equipes de inserir comportamento em seu ecossistema em pontos de conexão convenientes, mas inadequados, que resultam a longo prazo em dívidas técnicas e coisas piores. Os exemplos são abundantes, incluindo o uso de um banco de dados como ponto de integração, o uso de **Kafka** como orquestrador global, a mistura de lógica de negócio com código de infraestrutura, e assim por diante. O desenvolvimento de software moderno oferece muitas opções de lugares para as pessoas desenvolvedoras esconderem comportamento, e equipes inexperientes ou imprudentes muitas vezes se arriscam ao não considerar cuidadosamente as consequências de longo prazo do acoplamento inadequado. Estruturas de equipe inadequadas e outros desvios da **Lei de Conway** também não ajudam. À medida que os sistemas de software se tornam mais complexos, as equipes de desenvolvimento devem demonstrar diligência para criar e manter arquitetura e design de forma cuidadosa, e não com decisões precipitadas por conveniência. Frequentemente, pensar sobre a testabilidade de uma abordagem específica afasta as equipes de algumas dessas decisões potencialmente problemáticas. O software tende a ganhar complexidade quando deixado por conta própria. A combinação entre um design cuidadoso e, talvez ainda mais importante, uma governança contínua, garante que a pressão do cronograma ou uma das várias outras forças disuptivas não levem as equipes a tomar decisões convenientes, mas inadequadas.

A Lei de Conway ainda vale

Muitas pessoas arquitetas citam a Lei de Conway, uma observação feita nos anos 1960 de que as estruturas de comunicação das equipes influenciam o design, para justificar mudanças na organização da equipe. Descobrimos em vários blips incluídos nesta edição que a estrutura de equipe de uma organização continua sendo um habilitador-chave quando bem administrada, e um sério impedimento quando mal administrada. Os exemplos que discutimos incluem a necessidade de pensar no produto que envolve as equipes de plataforma, em vez de tratá-las como tiradoras de pedidos; Topologias de time e o crescente reconhecimento da carga cognitiva do time em relação à efetividade; além do novo framework desenvolvido em torno da produtividade de pessoas programadoras, chamado SPACE. As organizações fazem grandes investimentos em ferramentas, mas muitas encontram maiores ganhos de produtividade prestando atenção nas pessoas que criam o software e o que as torna efetivas em uma determinada organização.

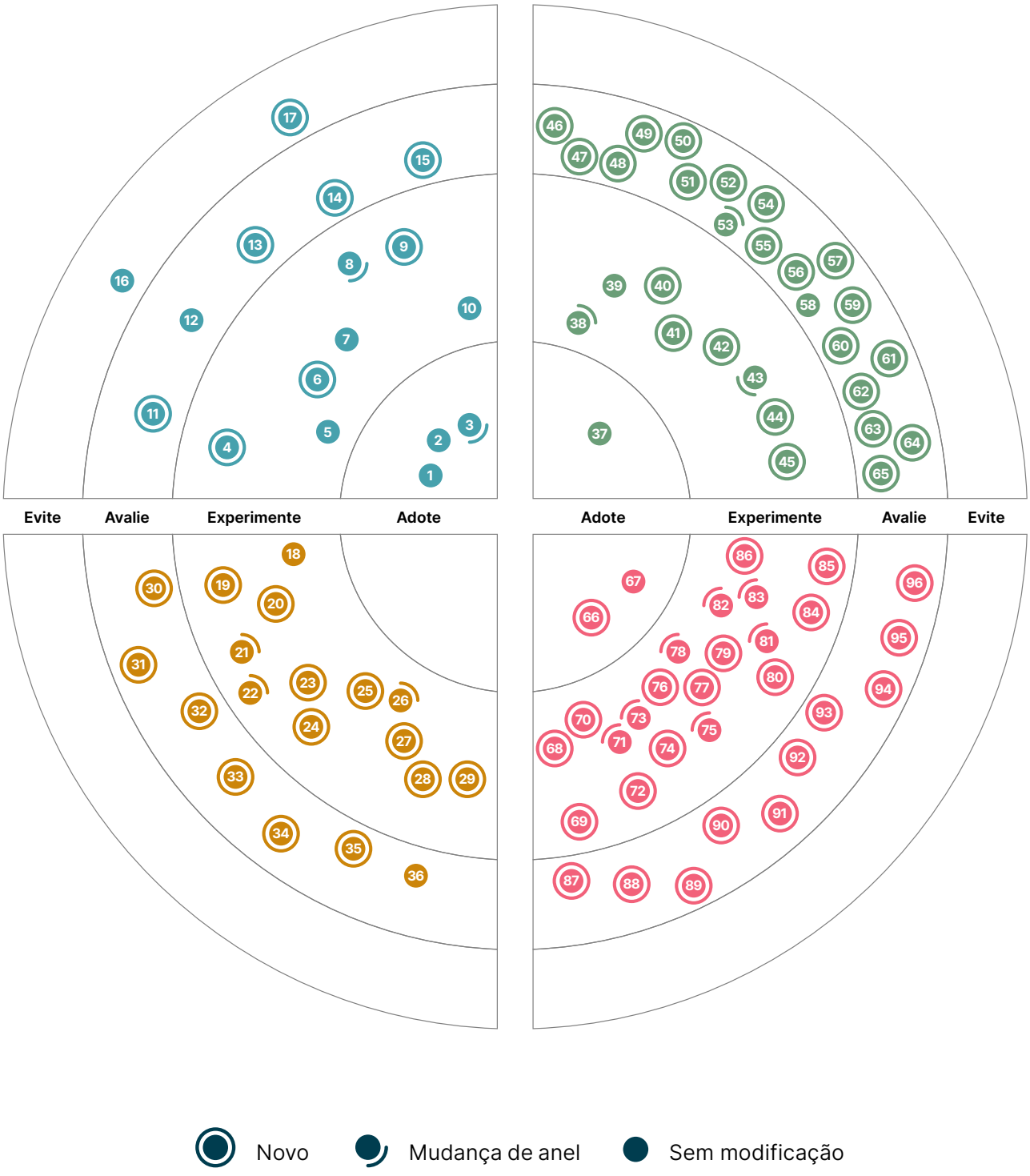
Tecnologia inteligente de que não deveríamos precisar

Muitas pessoas no mundo do software valorizam soluções inteligentes para problemas complexos, mas muitas vezes essas soluções inteligentes são resultado de complexidade acidental autoinfligida. Vários exemplos desse fenômeno existem hoje, incluindo a prática infeliz, mas comum, de manter código de orquestração ou de coordenação em locais inadequados. Por exemplo, vemos ferramentas de gerenciamento de fluxo de trabalho inteligentes, como Airflow ou Prefect sendo usadas avidamente para gerenciar pipelines de dados complexos por meio de orquestração. Encontramos uma série de ferramentas que contornam os problemas causados por monorepos, como Nx e muitas outras. As equipes muitas vezes não percebem que estão dobrando ou triplicando a complexidade de maneira desnecessária, sem parar para olhar o quadro geral e questionar se a solução atual é pior do que o problema. Em vez de recorrer a mais tecnologia para resolver um problema, as equipes devem analisar a causa raiz, abordar a complexidade essencial subjacente e corrigir a rota. Data mesh é um exemplo de abordagem que lida com suposições organizacionais e técnicas subjacentes que resultam em pipelines de dados e ferramentas excessivamente complexas.

Menos plataformas de tecnologia no Radar

Notamos uma queda significativa no número de blips relacionados à plataformas nesta edição do Radar, o que atribuímos à consolidação de alguns padrões do setor: a maioria das empresas já escolheu suas fornecedores de nuvem, que adotam como padrão principalmente Kubernetes para orquestração de contêineres e Kafka para mensagens de alto desempenho. Isso significa que as plataformas não importam mais? Ou estamos experimentando o equivalente a um ciclo de negócio com alternância entre períodos de expansão e contração — observamos períodos semelhantes de inovação rápida seguidos de estagnação (o que Stephen Jay Gould chamou de “equilíbrio pontuado”) em tecnologias de banco de dados, por exemplo. Talvez o setor tenha entrado em um período de relativa calmaria à medida que as organizações assimilam a mudança tectônica para a nuvem e aguardam a próxima onda de inovação disruptiva.

O Radar



O Radar

Técnicas

Adote

1. Quatro métricas fundamentais
2. Times de produto de engenharia de plataforma
3. Arquitetura de confiança zero

Experimente

4. Protocolos bilíngues CBOR/JSON
5. Malha de dados
6. Documentação viva em sistemas legados
7. Micro frontends para aplicativos móveis
8. Programação em grupo remota
9. Parede remota única para o time
10. Carga cognitiva do time

Avalie

11. Âncoras espaciais de RA
12. Hotwire
13. Padrão Operador para recursos não clusterizados
14. Huddle remota espontânea
15. Lista de materiais de software

Evite

16. Revisão por pares significa pull request
17. Dados de produção em ambientes de teste

Plataformas

Adote

—

Experimente

18. Backstage
19. ClickHouse
20. Confluent Kafka REST Proxy
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. OPA Gatekeeper para Kubernetes
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

Avalie

30. Azure Cognitive Search
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

Evite

—

O Radar

Ferramentas

Adote
37. fastlane

Experimente
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

Avalie
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. Comandos Unix modernos
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

Evite
—

Linguagens e frameworks

Adote
66. Jetpack Compose
67. React Hooks

Experimente
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplatform Mobile
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

Avalie
87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. Qiankun
94. React Three Fiber
95. Tauri
96. Transloco

Evite
—

Técnicas

Adote

1. Quatro métricas fundamentais
2. Times de produto de engenharia de plataforma
3. Arquitetura de confiança zero

Experimente

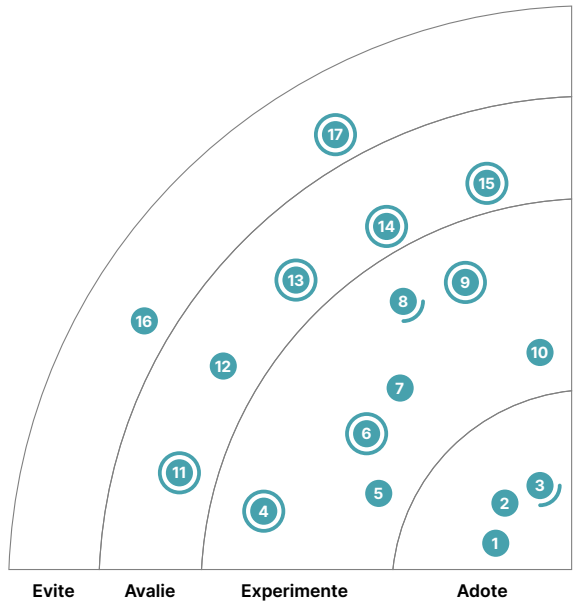
4. Protocolos bilíngues CBOR/JSON
5. Malha de dados
6. Documentação viva em sistemas legados
7. Micro frontends para aplicativos móveis
8. Programação em grupo remota
9. Parede remota única para o time
10. Carga cognitiva do time

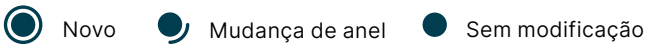
Avalie

11. Âncoras espaciais de RA
12. Hotwire
13. Padrão Operador para recursos não clusterizados
14. Huddle remota espontânea
15. Lista de materiais de software

Evite

16. Revisão por pares significa pull request
17. Dados de produção em ambientes de teste





1. Quatro métricas fundamentais

Adote

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

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

E como qualquer métrica, recomendamos sempre ter em mente a intenção final por trás, usando-as para refletir e aprender. Por exemplo, antes de passar semanas desenvolvendo ferramentas sofisticadas de dashboard, considere incluir a **verificação rápida do DORA** regularmente em retrospectivas do time. Isso dá ao time oportunidade de refletir sobre quais **recursos** podem ser trabalhados para melhorar suas métricas, o que pode ser muito mais efetivo do que usar ferramentas prontas para uso excessivamente detalhadas.

2. Times de produto de engenharia de plataforma

Adote

Continuamos vendo times de produto de engenharia de plataforma como um padrão sensato. O insight principal é que são **times de produto** como quaisquer outros, embora tenham foco em clientes da plataforma interna. Portanto, é fundamental ter clientes e produtos claramente definidos, usando as mesmas disciplinas da engenharia e formas de trabalhar de qualquer outra equipe de produto (com foco externo) — os times de plataforma não são especiais nesse aspecto. Desaconselhamos fortemente a prática de apenas renomear equipes internas existentes como “times de plataforma”, mas sem alterar as formas de trabalho e estruturas organizacionais. Ainda somos grandes fãs dos conceitos de **topologias de time** ao considerar a melhor forma de organizar equipes de plataforma. Consideramos os times de produto de engenharia de plataforma uma abordagem padrão e um elemento facilitador significativo para a TI de alto desempenho.

3. Arquitetura de confiança zero

Adote

Continuamos ouvindo relatos sobre empresas descobrindo que sua segurança estava seriamente comprometida devido ao excesso de confiança no perímetro de rede “seguro”. Uma vez que esse perímetro externo é violado, os sistemas internos provam estar mal protegidos, sendo suscetíveis a ataques capazes de implantar rapidamente e facilidade ferramentas automatizadas de extração de dados e ransomware que muitas vezes permanecem não detectadas por longos períodos. Isso nos leva a recomendar a arquitetura de confiança zero (ZTA) como um padrão agora sensato.

ZTA é uma mudança de paradigma na arquitetura e estratégia de segurança. É baseada na suposição de que um perímetro de rede não representa mais um limite seguro, e nenhuma confiança implícita deve ser concedida a usuários ou serviços com base apenas em sua localização física ou de rede.

O número de recursos, ferramentas e plataformas disponíveis para implementar aspectos da ZTA continua crescendo e inclui a aplicação de **políticas como código** com base nos princípios de privilégios mínimos e o mais granulares possível, monitoramento contínuo e mitigação automatizada de ameaças, **malha de serviço** para impor o controle de segurança de aplicação a serviço e serviço a serviço, implementação de **atestado binário** para verificar a origem dos binários, e **enclaves seguros**, além da criptografia tradicional para reforçar os três pilares da segurança de dados: em trânsito, em repouso e na memória. Para uma introdução ao tópico, consulte a publicação **NIST ZTA** e o artigo do Google sobre **BeyondProd**.

4. Protocolos bilíngues CBOR/JSON

Experimente

Embora já exista há algum tempo, estamos vendo cada vez mais casos de uso em que a especificação **CBOR** para intercâmbio de dados faz sentido — especialmente em ambientes contendo vários tipos de aplicações que se comunicam umas com as outras: serviço a serviço, navegador a serviço e assim por diante. Uma coisa que achamos útil com **Borer**, uma implementação Scala de um codificador/decodificador CBOR, é a capacidade de clientes negociarem conteúdo entre a representação binária e o antigo formato JSON. É muito útil ter uma versão em texto visível em um navegador, bem como um formato binário conciso. Prevemos protocolos bilíngues CBOR/JSON ganhando popularidade com o aumento contínuo da IoT e da computação de borda, além de outras situações em que o ambiente é fortemente restrito.

5. Malha de dados

Experimente

Cada vez mais, observamos uma incompatibilidade entre o que as organizações orientadas por dados desejam alcançar e o que as atuais arquiteturas de dados e estruturas organizacionais permitem. As organizações desejam incorporar tomada de decisão baseada em dados, aprendizado de máquina e análise de dados em muitos aspectos de seus produtos e serviços e operações internas, essencialmente, buscando aumentar cada aspecto de seu cenário operacional com inteligência orientada por dados. No entanto, ainda temos um longo caminho a percorrer antes de incorporar dados analíticos, incluindo definições de acesso e gerenciamento, nos domínios e operações de negócios. Hoje, todos os aspectos do gerenciamento de dados analíticos são mantidos fora dos domínios operacionais de negócio, com a equipe de dados e os monólitos de gerenciamento de dados: data lakes e data warehouses. Data mesh (ou **malha de dados**) é uma abordagem sociotécnica descentralizada para remover a dicotomia entre dados analíticos e operação de negócio. Seu objetivo é incorporar o compartilhamento e o uso de dados analíticos em cada domínio operacional do negócio e fechar a lacuna entre os planos operacional e analítico. É baseada em quatro princípios: propriedade de dados por domínio, dados como produto, plataforma de dados self-service e governança computacional federada.

Nossos times têm implementado a **arquitetura de malha de dados**, criando novas abstrações arquiteturais, como quantum de produto de dados para encapsular código, dados e política como uma unidade autônoma de compartilhamento de dados analíticos incorporado em domínios operacionais. Nossos times também vem criando recursos de plataforma de dados self-service para gerenciar o ciclo de vida dos quanta de produtos de dados de maneira declarativa, conforme descrito no livro ***Data Mesh***. Apesar de nossos avanços técnicos, ainda estamos enfrentando atritos com o uso de tecnologias existentes em uma topologia de malha de dados, sem mencionar a resistência dos domínios de negócio em abraçar o compartilhamento e o uso de dados como uma responsabilidade de primeira classe em algumas organizações.

6. Documentação viva em sistemas legados

Experimente

A documentação viva, que vem da comunidade de desenvolvimento orientado por comportamento (BDD), é muitas vezes considerada um privilégio para bases de código mantidas com especificações executáveis. Descobrimos que essa técnica também pode ser aplicada a sistemas legados. A falta de conhecimento do negócio é um obstáculo comum encontrado pelos times que executam a modernização do sistema. O código é geralmente a única fonte confiável da verdade devido à rotatividade de pessoas e à documentação existente desatualizada. Portanto, é muito importante restabelecer a associação entre a documentação e o código, difundindo o conhecimento do negócio entre o time ao assumir um sistema legado. Na prática, primeiro tentaríamos ir para a base de código e aprofundar nossa compreensão do negócio por meio de uma limpeza simples e uma refatoração segura. Durante o processo, será preciso adicionar anotações ao código para que a documentação viva possa ser gerada automaticamente mais tarde. Isso é muito diferente de fazer BDD em projetos green-field, mas é um bom começo em sistemas legados. Com base na documentação gerada, o passo seguinte seria converter algumas das especificações em testes de automação executáveis de alto nível. Fazendo isso iterativamente, eventualmente, você poderá obter documentação viva em sistemas legados intimamente associada ao código e parcialmente executável.

7. Micro frontends para aplicativos móveis

Experimente

Desde que os incluímos no Radar em 2016, vimos uma ampla adoção de micro frontends para interfaces de usuário web. Recentemente, entretanto, observamos projetos expandindo esse estilo arquitetural para também incluir micro frontends para aplicativos móveis. Quando o aplicativo se torna suficientemente grande e complexo, torna-se necessário distribuir o desenvolvimento entre vários times. Isso introduz o desafio de manter a autonomia dos times, ao mesmo tempo integrando seu trabalho em um único aplicativo. Alguns times escrevem seus próprios frameworks para habilitar esse estilo de desenvolvimento e, anteriormente, mencionamos Atlas e Beehive como possíveis formas de simplificar o problema de integração do desenvolvimento de aplicativos por múltiplos times. Mais recentemente, vimos times usando React Native para esse mesmo objetivo. Cada micro frontend React Native é mantido em seu próprio repositório, onde pode ser compilado, testado e implantado separadamente. A equipe responsável pelo aplicativo de forma geral pode, então, agregar esses micro frontends criados por diferentes times em um único release do aplicativo.

8. Programação em grupo remota

Experimente

Continuamos vendo muitos times trabalhando e colaborando remotamente. Para essas equipes, a programação em grupo remota é uma técnica que vale experimentar. A programação em grupo (mob programming) remota permite que os times rapidamente se agrupem em torno de um problema ou de parte do código sem as restrições físicas de poder acomodar apenas um determinado número de pessoas ao redor de uma estação de pareamento. Os times podem colaborar rapidamente em um problema ou código sem ter que se conectar a um grande display, reservar uma sala de reunião física ou usar um quadro branco.

9. Parede remota única para o time

Experimente

Com o crescimento do uso de times remotos distribuídos, uma das coisas de que as pessoas relatam sentir falta é a parede física da equipe. É um único lugar onde todos os vários cartões de histórias, tarefas, status e progresso podem ser exibidos, funcionando como uma espécie de radiador de informações e hub para o time. Frequentemente, a parede era um ponto de integração, com os dados reais sendo armazenados em vários sistemas diferentes. À medida que os times se tornaram remotos,

foi necessário voltar a olhar individualmente para os sistemas de fonte, e obter uma visão "rápida" de um projeto se tornou muito difícil. Uma parede remota única para o time é uma técnica simples para reintroduzir a parede de equipes virtualmente. Embora possa haver alguma sobrecarga para mantê-la atualizado, sentimos que os benefícios para os times valem a pena. Para algumas equipes, a atualização da parede física fazia parte das "cerimônias" diárias que o time fazia em conjunto, e o mesmo pode ser feito com uma parede remota.

10. Carga cognitiva do time

Experimente

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

11. Âncoras espaciais de RA

Avalie

Muitas aplicações de realidade aumentada (RA) dependem do conhecimento de localização e orientação do dispositivo do usuário. O padrão é usar soluções baseadas em GPS, mas as âncoras espaciais, uma técnica mais recente para atender a esse requisito, também é uma opção a se considerar. As âncoras espaciais trabalham com a imagem gravada pela câmera do dispositivo, usando recursos de imagem e sua posição relativa no espaço 3D para reconhecer uma localização no mundo real. Para este local, uma âncora correspondente é criada no espaço de RA. Embora as âncoras espaciais não possam substituir todas as âncoras baseadas em marcadores e GPS, elas oferecem mais precisão do que a maioria das soluções baseadas em GPS e são mais resilientes a diferentes ângulos de visão do que as âncoras baseadas em marcadores. Nossa experiência atualmente se limita a Cloud Anchors para Android do Google, que funcionou bem para nós. De forma atípica, o Google também oferece Cloud Anchors para iOS e com Azure Spatial Anchors, a Microsoft oferece suporte a ainda mais plataformas.

12. Hotwire

Avalie

Depois de lançar com sucesso seu aplicativo de e-mail HEY como uma aplicação do lado do servidor, o Basecamp comunicou a migração de seu principal produto, Basecamp 3, para Hotwire. À medida que as organizações adotam cada vez mais aplicações de página única (SPAs) para novos desenvolvimentos web, seguimos otimistas com o Hotwire nadando contra a corrente. Ao contrário dos SPAs, os aplicativos Hotwire mantêm a maior parte da lógica e da navegação no servidor, contando com uma quantidade mínima de JavaScript do navegador. O Hotwire modulariza as páginas HTML em um conjunto de componentes (chamados Turbo Frames) que podem ser carregados pelo padrão lazy loading, fornecer contextos independentes e enviar atualizações HTML para esses contextos com base nas ações do usuário. SPAs oferecem responsividade inegável ao usuário, mas a simplicidade da programação web tradicional do lado do servidor combinada com ferramentas de navegador modernas oferece uma visão revigorada do equilíbrio entre a efetividade da pessoa desenvolvedora e a capacidade de resposta do usuário.

13. Padrão Operador para recursos não clusterizados

Avalie

Temos observado o uso crescente do padrão Operador do Kubernetes para outros fins além do gerenciamento de aplicativos implantados no cluster. O uso do Padrão Operador para recursos não clusterizados se beneficia das vantagens oferecidas pelas definições personalizadas de recursos e o mecanismo de agendamento orientado a eventos implementado no plano de controle do Kubernetes para gerenciar atividades relacionadas ainda fora do cluster. Esta técnica se baseia na ideia de serviços em nuvem gerenciados por Kube e a estende para outras atividades, como implantação contínua ou reação a mudanças em repositórios externos. Uma vantagem dessa técnica em relação a uma ferramenta criada para um fim específico é que ela abre uma ampla gama de ferramentas que vêm com Kubernetes ou fazem parte de um ecossistema mais amplo. Você pode usar comandos como **`diff`**, **`dry-run`** ou **`apply`** para interagir com os recursos personalizados do operador. O mecanismo de agendamento do Kube torna o desenvolvimento mais fácil, eliminando a necessidade de orquestrar as atividades na ordem adequada. Ferramentas de código aberto como Crossplane, Flux e ArgoCD usam essa técnica e esperamos ver outras surgindo com o tempo.

14. Huddle remota espontânea

Avalie

Temos observado inovação contínua em ferramentas de colaboração remota. O novo recurso de Huddles no Slack fornece uma experiência semelhante ao Discord de chamadas de áudio persistente, que os usuários podem acessar a qualquer momento. Gather fornece uma maneira criativa de emular um escritório virtual com avatares e vídeo. Os IDEs oferecem recursos de colaboração direta para pareamento e depuração de erros: mencionamos o Visual Studio Live Share anteriormente e nesta edição incluímos Code With Me da JetBrains. À medida que as ferramentas continuam a desenvolver modalidades de colaboração além de videoconferência, vemos cada vez mais equipes participando de momentos de huddle remota espontânea, recriando a espontaneidade de conversas informais em contraste com a intencionalidade de agendar uma reunião no Zoom ou no Microsoft Teams. Não esperamos que a riqueza da comunicação face a face seja recriada por meio de ferramentas digitais, mas vemos uma melhora na efetividade dos times remotos ao proporcionar vários canais de colaboração, em vez de depender de um conjunto de ferramentas para tudo.

15. Lista de materiais de software

Avalie

Em maio de 2021, a Casa Branca dos EUA publicou sua ordem executiva para melhorar a segurança cibernética da nação. O documento apresenta vários mandatos técnicos relacionados a itens que apresentamos em edições anteriores do Radar, como arquitetura de confiança zero e digitalização de conformidade automatizada usando política de segurança como código. Grande parte do documento é dedicado a melhorar a segurança da cadeia de suprimentos de software. Um item em particular que chamou nossa atenção foi o requisito de que o software governamental deveria conter uma lista de materiais de software (Software Bill of Materials ou SBOM) legível por máquina, definida como “um registro formal contendo os detalhes e as relações da cadeia de suprimentos de vários componentes usados no desenvolvimento de software.” Em outras palavras, a lista deve detalhar não apenas os componentes enviados, mas também as ferramentas e os frameworks usados para entregar o software. Esse pedido tem o potencial de inaugurar uma nova era de transparência e abertura no desenvolvimento de software. Isso, sem dúvida, terá um impacto sobre quem cria software como profissão. Muitos, senão todos os produtos de software produzidos hoje, contêm componentes de código aberto ou os empregam no processo de desenvolvimento. Frequentemente, o público consumidor não tem como saber qual versão de qual pacote pode ter um impacto na segurança de seu produto. Em vez disso, contam com os alertas de segurança e patches oferecidos pela fornecedora de varejo. Essa ordem executiva garantirá que uma descrição explícita de todos

os componentes seja disponibilizada ao público consumidor, habilitando-o a implementar seus próprios controles de segurança. E como a SBOM é legível por máquina, esses controles podem ser automatizados. Notamos que esse movimento também representa uma mudança no sentido de adotar o software de código aberto e abordar de forma prática os riscos e benefícios de segurança oferecidos.

16. Revisão por pares significa pull request

Evite

Algumas organizações parecem acreditar que a revisão por pares significa pull request. Elas consideram que a única maneira de se obter uma revisão por pares do código é por meio de um pull request. Vimos essa abordagem criar gargalos significativos para o time, além de degradar significativamente a qualidade do feedback, à medida que pessoas revisoras sobrecarregadas passam a simplesmente rejeitar as solicitações. Embora seja possível argumentar que esta é uma maneira de demonstrar a conformidade regulamentar da revisão do código, uma de nossas clientes foi informada de que esse argumento era inválido, pois não havia evidências de que o código foi realmente lido por alguém antes da aceitação. Pull requests são apenas uma maneira de gerenciar o fluxo de trabalho de revisão de código. Encorajamos as pessoas a considerarem outras abordagens, especialmente quando houver necessidade de ensinar e transmitir feedback com cuidado.

17. Dados de produção em ambientes de teste

Evite

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

Plataformas

Adote

—

Experimente

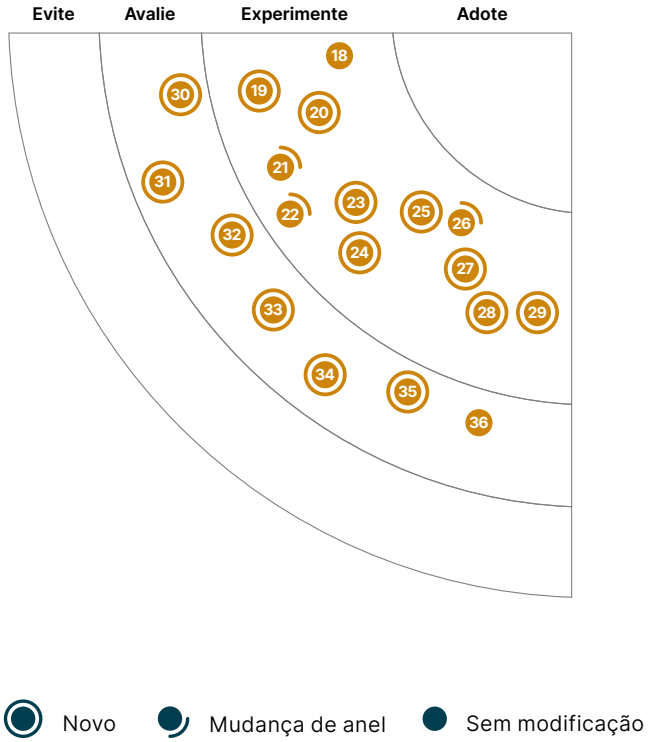
18. Backstage
19. ClickHouse
20. Confluent Kafka REST Proxy
21. GitHub Actions
22. K3s
23. Mambu
24. MirrorMaker 2.0
25. OPA Gatekeeper para Kubernetes
26. Pulumi
27. Sealed Secrets
28. Vercel
29. Weights & Biases

Avalie

30. Azure Cognitive Search
31. Babashka
32. ExternalDNS
33. Konga
34. Milvus 2.0
35. Thought Machine Vault
36. XTDB

Evite

—



18. Backstage

Experimente

À medida que o foco em melhorar a experiência e a efetividade das pessoas desenvolvedoras aumenta nas organizações, vemos **Backstage** ganhar popularidade, junto com a adoção de portais de desenvolvimento. Essas organizações procuram oferecer suporte e otimizar seus ambientes de desenvolvimento de software. Conforme o número de ferramentas e tecnologias aumenta, alguma forma de padronização se torna cada vez mais importante para a consistência, de modo que as pessoas desenvolvedoras possam se concentrar na inovação e no desenvolvimento de produtos, em vez de se prenderem a reinventar a roda. Backstage é uma plataforma de portal de desenvolvimento de código aberto criada pelo Spotify. É baseada em modelos de software, unificando ferramentas de infraestrutura e documentação técnica consistente e centralizada. A arquitetura de plug-in permite extensibilidade e adaptabilidade no ecossistema de infraestrutura de uma organização. Vamos seguir observando de perto o novo **catálogo de serviços do Backstage**, atualmente em alfa, que mantém o controle de propriedade e metadados para software como um todo no ecossistema de uma organização.

19. ClickHouse

Experimente

ClickHouse é um banco de dados de processamento analítico online colunar (OLAP) de código aberto para análise em tempo real. Começou como um projeto experimental em 2009 e desde então amadureceu para um banco de dados analítico de alto desempenho linearmente escalável. Seu mecanismo de processamento de consulta eficiente, junto com a compactação de dados, torna-o adequado para executar consultas interativas sem pré-agregação. Usamos ClickHouse e seu desempenho tem nos impressionado bastante.

20. Confluent Kafka REST Proxy

Experimente

Kafka é um padrão comum para arquiteturas orientadas a eventos, mas adaptá-lo a ambientes legados introduz uma incompatibilidade de impedância. Em alguns casos, tivemos sucesso em minimizar a complexidade do legado usando **Confluent Kafka REST Proxy**. O proxy permite que as pessoas desenvolvedoras acessem Kafka por meio de uma interface HTTP, o que é útil em ambientes que dificultam o uso do protocolo nativo Kafka. Por exemplo, fomos capazes de consumir eventos emitidos com SAP simplesmente fazendo a equipe SAP invocar um comando HTTP POST por meio de uma chamada de função remota SAP pré-configurada, evitando a necessidade de executar o spin up de uma abstração Java em SAP (e de uma equipe para gerenciá-la). O proxy tem muitos recursos, embora, como acontece com qualquer ferramenta de adaptação, recomendamos cautela e uma visão clara das compensações envolvidas. Acreditamos que o proxy seja valioso quando permite que produtores legados enviem eventos, mas é preciso ter mais cuidado ao criar consumidores de eventos por meio do proxy conforme a abstração fica mais complexa. O proxy não muda o fato de que os consumidores Kafka têm estado, o que significa que as instâncias do consumidor criadas por meio da API REST estão vinculadas a um proxy específico, e a necessidade de fazer uma chamada HTTP para consumir mensagens de um tópico muda a semântica padrão de eventos Kafka.

21. GitHub Actions

Experimente

Apesar de nossa recomendação de cautela na última vez em que o mencionamos, temos visto um entusiasmo contínuo com o **GitHub Actions**. O que dissemos antes ainda é válido: o GitHub Actions ainda não substitui completamente CI/CD para fluxos de trabalho complexos. O orquestrador não pode, por exemplo, disparar novamente uma única tarefa de um fluxo de trabalho, chamar outras ações dentro de uma ação composta ou oferecer suporte a uma biblioteca compartilhada. Além disso,

embora o ecossistema no GitHub Marketplace ofereça vantagens óbvias, dar a ações do GitHub de terceiros acesso ao seu pipeline de compilação gera o risco do compartilhamento de segredos de maneiras inseguras (recomendamos seguir as recomendações do GitHub sobre fortalecimento de segurança). Apesar dessas preocupações, a conveniência de criar seu fluxo de trabalho de compilação diretamente no GitHub ao lado do seu código-fonte é uma opção atraente para alguns times, e act ajuda a executar ações do GitHub localmente. Como sempre, recomendamos uma avaliação clara das vantagens e desvantagens envolvidas, mas alguns de nossos times estão satisfeitos com a simplicidade do GitHub Actions.

22. K3s

Experimente

K3s é uma distribuição leve do Kubernetes desenvolvida para IoT e computação de borda. Com K3s, você obtém os benefícios de um Kubernetes em conformidade com todas as regulamentações necessárias, mas com a sobrecarga operacional reduzida. Suas melhorias incluem back-ends de armazenamento leves (sqlite3 como padrão em vez de etcd), um único pacote binário com dependências mínimas de sistema operacional e consumo de memória reduzido, tornando K3s adequado para ambientes com recursos limitados. Usamos K3s em máquinas de ponto de venda e estamos muito contentes com nossa decisão.

23. Mambu

Experimente

Mambu é uma plataforma SaaS de serviços bancários em nuvem que permite que clientes criem e alterem de forma simples e flexível seus produtos bancários e empréstimos. Ao contrário de outras plataformas de core banking prontas para uso, que só permitem adaptação com integração embutida no código, Mambu foi projetada para ofertas financeiras em constante mudança. A plataforma vem com um fluxo de trabalho opinativo, ao mesmo tempo fornecendo uma abordagem baseada em API para personalizar a lógica de negócios, processos e integrações. Atualmente, temos vários projetos usando o Mambu. Com sua escalabilidade baseada em nuvem e recursos altamente personalizáveis, está se tornando um padrão sensato de sistema de domínio para construir produtos financeiros.

24. MirrorMaker 2.0

Experimente

Construído usando o framework Kafka Connect, MirrorMaker 2.0 (também conhecido como MM2) resolve muitas deficiências de ferramentas em abordagens anteriores de replicação do Kafka. O MM2 pode georreplicar dados de tópicos e metadados entre clusters com sucesso, incluindo offsets, grupos de consumidores e linhas de comando de autorização (ACLs). MM2 preserva o particionamento e detecta novos tópicos e partições. Gostamos de sua capacidade de preparar uma migração de cluster ao longo do tempo, uma abordagem que pode ser útil na migração de um cluster local para um cluster em nuvem. Depois de sincronizar os tópicos e grupos de consumidores, primeiro migramos clientes para o novo local do cluster, depois migramos produtores para o novo local e finalmente desligamos o MM2 e descomissionamos o cluster antigo. Também vimos o MM2 sendo usado em cenários de recuperação de desastres e alta disponibilidade.

25. OPA Gatekeeper para Kubernetes

Experimente

OPA Gatekeeper para Kubernetes é um webhook de admissão personalizável para Kubernetes que impõe políticas executadas pelo Open Policy Agent (OPA). Estamos usando esta extensão da plataforma Kubernetes para adicionar uma camada de segurança aos clusters, fornecendo mecanismos de governança automatizados que garantem que as aplicações sejam compatíveis com as políticas definidas. Nossos times gostam especialmente de sua capacidade de personalização:

usar CustomResourceDefinitions (CRD) nos permite definir ConstraintTemplates e Constraints que tornam a definição de regras e objetos (por exemplo, implantações, tarefas, tarefas cron) e namespaces sob avaliação uma tarefa fácil.

26. Pulumi

Experimente

Temos visto um número cada vez maior de times usando Pulumi em várias organizações. Pulumi preenche uma lacuna no mundo da programação de infraestrutura, onde o Terraform mantém uma posição firme. Embora o Terraform seja um standby testado e aprovado, sua natureza declarativa sofre com recursos de abstração inadequados e testabilidade limitada. O Terraform é adequado quando a infraestrutura é totalmente estática, mas as definições de infraestrutura dinâmica exigem uma linguagem de programação real. Pulumi se distingue por permitir que as configurações sejam escritas em TypeScript/JavaScript, Python e Go — sem necessidade de linguagem de marcação ou modelagem. Pulumi tem seu foco fortemente voltado para arquiteturas nativas de nuvem — incluindo contêineres, funções sem servidor e serviços de dados — e fornece bom suporte para Kubernetes. Recentemente, o AWS CDK se mostrou um forte concorrente, mas Pulumi continua a ser a única ferramenta neutra de nuvem nesta área.

27. Sealed Secrets

Experimente

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

28. Vercel

Experimente

Desde que avaliamos pela primeira vez o JAMstack, temos visto cada vez mais aplicações web desse mesmo estilo. No entanto, quando a infraestrutura para construir sites dinâmicos tradicionais e serviços de back-end é muito pesada para o JAMstack, nossos times optam pelo Vercel. Vercel é uma plataforma em nuvem para hospedagem estática de sites. E o mais importante: fornece um fluxo de trabalho perfeito para desenvolver, visualizar e enviar sites JAMstack. A configuração da implantação é bastante simples. Com a integração com o GitHub, cada commit de código ou pull request pode acionar uma nova implantação de site que tem uma URL para visualização, o que acelera muito o feedback do desenvolvimento. Vercel também usa o CDN para escalar e acelerar os locais de produção. Vale a pena mencionar que a equipe por trás do Vercel também é responsável por outro framework popular, o Next.js.

29. Weights & Biases

Experimente

Weights & Biases é uma plataforma de aprendizado de máquina (ML) para construir modelos com mais rapidez, por meio de rastreamento de experimentos, controle de versão de conjunto de dados, visualização de desempenho e gerenciamento de modelo. Você pode integrá-la com seu código

de ML existente e acessar rapidamente métricas em tempo real, logs de terminal e estatísticas do sistema transmitidas ao dashboard para análise posterior. Nossos times têm usado Weights & Biases, e nos agrada sua abordagem colaborativa para a construção de modelos.

30. Azure Cognitive Search

Avalie

Azure Cognitive Search oferece pesquisa como um serviço para aplicações que exigem pesquisa de texto em conteúdo heterogêneo. Também fornece APIs baseadas em push ou pull para fazer upload e indexar imagens, texto não estruturado ou conteúdo de documento estruturado, com limitações nos tipos de fonte de dados baseados em pull com suporte. O serviço fornece ainda APIs em REST e .NET SDK para executar consultas de pesquisa, usando uma linguagem de consulta simples ou consultas mais poderosas Apache Lucene, com consultas de escopo de campo, pesquisa difusa, pesquisa de infixo e sufixo curinga, pesquisa de expressão regular, entre outros recursos. Usamos com sucesso a Azure Cognitive Search em conjunto com outros serviços Azure, incluindo a pesquisa de conteúdo carregado do Cosmos DB.

31. Babashka

Avalie

Até hoje, considerando todas as ferramentas de desenvolvimento e infraestrutura à nossa disposição, muitas vezes chegamos a um ponto em que precisamos de um script para colar várias coisas ou automatizar uma tarefa recorrente. Os atuais favoritos para escrever esses scripts são bash e Python, mas estamos felizes em informar que há uma opção nova e interessante: Clojure. Isso é possível com Babashka, um tempo de execução Clojure completo implementado com GraalVM. Babashka vem com bibliotecas que cobrem a maioria dos casos de uso nos quais você usaria uma ferramenta de script, e o carregamento de outras bibliotecas também é possível. O uso do GraalVM traz os tempos de inicialização dentro do alcance das ferramentas nativas e também torna o Babashka uma das poucas opções para um ambiente de script multithread, para aqueles raros casos em que é necessário.

32. ExternalDNS

Avalie

ExternalDNS sincroniza entradas e serviços Kubernetes com provedores DNS externos, preenchendo uma lacuna anteriormente ocupada por kops dns-controller, Zalando's Mate ou route53-kubernetes — os dois últimos foram substituídos por ExternalDNS. A ferramenta torna os recursos internos do Kubernetes detectáveis por meio de servidores DNS públicos, removendo uma etapa às vezes manual para atualizar os registros DNS quando um host de entrada ou endereço de IP do serviço muda. A ferramenta oferece suporte a uma grande lista de provedores de serviços DNS prontos para uso, com mais sendo adicionados por meio do suporte da comunidade. Como diz a velha piada, é sempre DNS.

33. Konga

Avalie

Konga é uma IU de código aberto para administrar Kong API Gateway, anteriormente incluído no Radar no anel Experimente. Nossos times gostaram da configuração rápida e do rico conjunto de recursos, que permite experimentar e testar configurações com facilidade. E o fato de ser um software de código aberto alivia as preocupações em relação a custos de licenciamento.

34. Milvus 2.0

Avalie

Milvus 2.0 é um banco de dados vetorial de código aberto e nativo de nuvem, criado para busca e gerenciamento de vetores de embedding gerados por modelos de aprendizado de máquina e redes neurais. A plataforma suporta vários **índices vetoriais** para pesquisa de vizinhos mais próximos aproximados (ANN) em vetores de embedding de áudio, vídeo, imagem ou quaisquer dados não estruturados. Milvus 2.0 é um banco de dados relativamente novo e recomendamos que você o avalie para suas necessidades de busca por similaridade.

35. Thought Machine Vault

Avalie

É raro incluirmos software comercial de prateleira no Radar, principalmente se tratando de uma plataforma de core banking. No entanto, **Thought Machine Vault** (nenhuma conexão com a Thoughtworks) é um exemplo de um produto desta classe projetado para suportar boas práticas de engenharia de software, como desenvolvimento orientado a testes, entrega contínua e infraestrutura como código. As pessoas desenvolvedoras definem produtos bancários no Vault escrevendo contratos inteligentes em código Python. Isso é muito diferente da abordagem padrão sem código, em que a personalização é feita por meio de interfaces gráficas ou arquivos de configuração proprietários, ou ambos. Como os produtos são definidos em código Python regular, as pessoas desenvolvedoras têm acesso a uma variedade de ferramentas, como estruturas de teste e controle de versão, para garantir que seu trabalho seja seguro e preciso. Gostaríamos que mais plataformas de serviços financeiros fossem projetadas com a eficácia de pessoas desenvolvedoras em mente.

36. XTDB

Avalie

XTDB é um banco de dados de documentação de código aberto, com consultas de grafo bitemporais. Suporta nativamente dois eixos de tempo para cada registro: tempo *válido*, quando um fato ocorre, e tempo de *transação*, quando um fato é processado e registrado pelo banco de dados. O suporte para bitemporalidade é um benéfico em vários cenários, incluindo casos de uso analíticos executando consultas com reconhecimento de tempo; auditoria de mudanças históricas em fatos; suporte a arquiteturas de dados distribuídas que precisam garantir consultas point-in-time globalmente consistentes como **data mesh (malha de dados)**; e preservação da imutabilidade dos dados. O XTDB recebe informações em forma de documentos, expressas no formato Extensible Data Notation (EDN), um subconjunto da linguagem Clojure. O XTDB suporta grafos e também consultas SQL, e é extensível por meio de uma camada de API REST e Kafka Connect, entre outros módulos. Estamos otimistas em ver um crescimento na adoção de XTDB e a adição de recursos como suporte para transações e SQL.

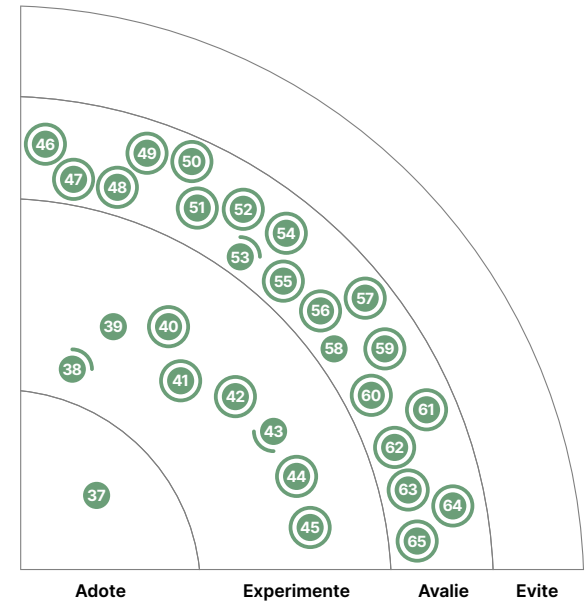
Ferramentas

Adote
37. fastlane

Experimente
38. Airflow
39. Batect
40. Berglas
41. Contrast Security
42. Dive
43. Lens
44. Nx
45. Wav2Vec 2.0

Avalie
46. cert-manager
47. Cloud Carbon Footprint
48. Code With Me
49. Comby
50. Conftest
51. Cosign
52. Crossplane
53. gopass
54. Micoo
55. mob
56. Comandos Unix modernos
57. Mozilla Sops
58. Operator Framework
59. Pactflow
60. Prefect
61. Proxyman
62. Regula
63. Sourcegraph
64. Telepresence
65. Vite

Evite
—



Novo Mudança de anel Sem modificação

37. fastlane

Adote

O release de aplicativos para iOS envolve uma etapa de assinatura de código. Embora seja suportado pelo conjunto de ferramentas da Apple, o processo pode ser complicado, sujeito a erros e repleto de surpresas inesperadas. Ficamos felizes em informar que **fastlane**, que já era nossa escolha de ferramenta para automação do processo de release de aplicativos móveis, oferece uma solução melhor: **match** se integra ao fluido processo oferecido por fastlane, implementando uma **nova abordagem** de gerenciamento de assinatura de código para times. Em vez de armazenar as chaves de assinatura entre as chaves do macOS da pessoa desenvolvedora — que é a abordagem padrão —, a nova abordagem gira em torno do armazenamento de chaves e certificados em um repositório Git. Além de facilitar o onboarding de novos membros do time e a configuração de novas máquinas de desenvolvimento, em nossa experiência, também é o método mais simples para integrar assinatura em pipelines de entrega contínua.

38. Airflow

Experimente

Nos últimos anos, observamos o surgimento de ferramentas de gerenciamento de fluxo de trabalho genéricas e específicas de domínio. Os fatores motivadores por trás desse aumento incluem o aumento do uso de pipelines de processamento de dados e a automação do processo de desenvolvimento de modelo de aprendizado de máquina (ML). **Airflow** é uma das primeiras ferramentas de orquestração de tarefas de código aberto que popularizou a definição de gráficos acíclicos direcionados (DAGs) como código, uma melhoria em relação a uma configuração de pipeline em XML/YAML. Embora o Airflow continue sendo uma das ferramentas de orquestração mais amplamente adotadas, recomendamos que você avalie outras ferramentas com base em sua situação específica. Por exemplo, você pode escolher **Prefect**, que oferece suporte a tarefas de processamento de dados dinâmicas como uma preocupação de primeira classe, com funções Python genéricas como tarefas; **Argo** caso você prefira uma integração estreita com Kubernetes; **Kubeflow** ou **MLflow** para fluxos de trabalho específicos de ML. Devido ao surgimento de novas ferramentas, combinadas com algumas das deficiências do Airflow (como a falta de suporte nativo para fluxos de trabalho dinâmicos e sua abordagem centralizada para agendar pipelines), não recomendamos mais o Airflow como ferramenta de orquestração padrão.

Acreditamos que com o aumento do uso de streaming em análises e pipelines de dados, bem como o gerenciamento de dados por meio de uma **malha de dados descentralizada**, a necessidade de ferramentas de orquestração para definir e gerenciar pipelines de processamento de dados complexos é reduzida.

39. Batect

Experimente

Batect continua ganhando força entre nossos times de desenvolvimento e é considerada por muitos uma abordagem padrão para configurar ambientes de desenvolvimento e teste locais. Esta ferramenta de código aberto (que por acaso é desenvolvida por um Thoughtworker) torna mais fácil a tarefa de configurar e compartilhar um ambiente de construção baseado em **Docker**. Batect se torna então o ponto de entrada para seu sistema de compilação, substituindo o onipresente script go como base para uma abordagem "**check out and go**". O Batect continua a evoluir em resposta ao feedback de pessoas desenvolvedoras e recentemente adicionou suporte para o BuildKit do Docker e conclusão de tabulação do shell.

40. Berglas

Experimente

Berglas é uma ferramenta para gerenciar segredos na Google Cloud Platform (GCP). Anteriormente, recomendamos a técnica de segredos como serviço para armazenar e compartilhar segredos em arquiteturas distribuídas modernas. GCP oferece o Secret Manager para essa finalidade, e Berglas funciona bem com o Secret Manager. Isso é especialmente útil para os serviços da GCP que ainda não têm integração direta com o Secret Manager; a alternativa em tais casos seria escrever código ou scripts personalizados. O Berglas é fornecido como uma ferramenta de linha de comando e uma biblioteca, e ambos também são úteis em casos de uso além de segredos como serviço. O autor do Berglas, que também o responsável pelo HashiCorp Vault, trabalha no Google atualmente; no entanto, Berglass não é um produto oficial do Google.

41. Contrast Security

Experimente

Contrast Security oferece uma plataforma de segurança com vários componentes, incluindo teste de segurança de aplicativo estático (SAST), teste de segurança de aplicativo interativo (IAST), verificação de código aberto e autoproteção de aplicativo de tempo de execução (RASP). A ferramenta já existe há alguns anos e já a usamos em vários projetos. Uma das coisas de que gostamos bastante na plataforma Contrast é sua análise de tempo de execução de bibliotecas; ajuda a identificar bibliotecas que não estão em uso, o que por sua vez ajuda nossos times a priorizar vulnerabilidades e potencialmente descartar bibliotecas não utilizadas. Isso é particularmente relevante, dada a importância crescente de proteção da cadeia de suprimentos de software. Também gostamos bastante de seu componente IAST — descobrimos que é efetivo em nosso pipeline de entrega contínua (CD), reduzindo falsos positivos e detectando uma boa variedade de vulnerabilidades.

42. Dive

Experimente

Dive é uma ferramenta para analisar imagens Docker que ajuda a explorar cada camada da imagem e identificar o que mudou em cada uma. Dive estima a *eficiência da imagem* e o *espaço desperdiçado* em uma imagem e pode ser integrado ao pipeline de integração contínua (CI) para reprovar o build com base na pontuação de eficiência ou quantidade de espaço desperdiçado. Nós a usamos em alguns projetos e Dive provou ser uma ferramenta útil — especialmente se você estiver criando imagens com tolerância muito baixa para ferramentas adicionais ou consumo de espaço.

43. Lens

Experimente

Nossos times seguem relatando bons resultados ao usar Lens para visualizar e gerenciar seus clusters Kubernetes. Rotulado como um "IDE para Kubernetes", Lens possibilita interagir com o cluster sem ter que memorizar comandos ou manifestar estruturas de arquivo. O Kubernetes pode ser complexo, e entendemos que uma ferramenta para visualizar métricas de cluster e cargas de trabalho implantadas pode economizar tempo e reduzir parte do trabalho envolvido na manutenção de um cluster Kubernetes. Em vez de ocultar a complexidade atrás de uma interface de apontar e clicar, Lens reúne as ferramentas que um admin executaria na linha de comando. Mas tenha cuidado ao fazer alterações interativamente em um cluster em execução usando qualquer mecanismo. Geralmente preferimos que as mudanças de infraestrutura sejam implementadas no código para que possam ser repetidas, testadas e menos sujeitas a erros humanos. No entanto, Lens se destaca como uma ferramenta completa para navegar interativamente e compreender o status de seu cluster.

44. Nx

Experimente

Ao longo dos anos, discutimos várias vezes se deveríamos incluir monorepos no Radar. Em todas as oportunidades, concluímos que os dilemas introduzidos pelos monorepos requerem uma discussão com muitas nuances, e que a técnica é "muito complexa para ser capturada em um blip". Agora, estamos observando um interesse crescente em monorepos na comunidade JavaScript, por exemplo, para construir aplicativos compostos de micro frontends, conforme discutido neste episódio de podcast. Se esta é uma boa ideia ou não depende muito da sua situação e certamente não queremos dar uma recomendação geral. O comentário que gostaríamos de fazer é sobre as ferramentas. Em nossos times, vemos um distanciamento do Lerna e uma forte preferência por usar Nx para gerenciar JavaScript com base em monorepos.

45. Wav2Vec 2.0

Experimente

Wav2Vec 2.0 é um framework de aprendizagem auto-supervisionada para reconhecimento de fala. Com este framework, o modelo é treinado em duas fases. Inicialmente, no modo auto-supervisionado, usando dados não rotulados e tentando obter a melhor representação de fala possível. E em seguida, usando o ajuste fino supervisionado, momento em que os dados rotulados ensinam o modelo a prever palavras ou fonemas específicos. Usamos o Wav2Vec e consideramos sua abordagem bastante poderosa para construir modelos de reconhecimento automático de voz para idiomas regionais com disponibilidade limitada de dados rotulados.

46. cert-manager

Avalie

cert-manager é uma ferramenta para gerenciar certificados X.509 em seu cluster Kubernetes. cert-manager modela certificados e emissores como tipos de recursos de primeira classe e fornece certificados como serviço com segurança para pessoas desenvolvedoras e aplicações trabalhando no cluster Kubernetes. Com suporte integrado para Let's Encrypt, HashiCorp Vault e Venafi, cert-manager é uma ferramenta interessante para avaliar para gerenciamento de certificados.

47. Cloud Carbon Footprint

Avalie

Stakeholders esperam cada vez mais que as empresas prestem contas dos impactos ambientais de suas decisões, conforme evidenciado pelo aumento de investimentos ambientais, sociais e de governança corporativa (ESG) e do ativismo de profissionais em relação às mudanças climáticas. A migração para a nuvem oferece potencial para um uso mais efetivo da energia — os provedores de nuvem têm muito mais escala para justificar o investimento em fontes de energia sustentáveis e P&D —, mas a desvantagem das abstrações de software para usuários da nuvem é que essas abstrações também escondem o impacto da energia, pois os data centers são ocultados e financiados por outra empresa. Cloud Carbon Footprint, uma nova ferramenta de código aberto, usa APIs de nuvem para fornecer visualizações de emissões de carbono estimadas com base no uso em AWS, GCP e Azure. A ferramenta usa heurísticas como Etsy Cloud Jewels para estimar o uso de energia e fontes de dados públicos para converter o uso de energia em emissões com base na intensidade de carbono da rede de energia subjacente da região da nuvem (A GCP já publica esses dados). Os painéis da ferramenta atuam como radiadores de informações, permitindo que os indivíduos tomadores de decisão modifiquem as configurações para cortar custos e emissões ao mesmo tempo. A ligação das regiões da nuvem à intensidade de carbono da rede subjacente funciona como um incentivo para mudar as cargas de trabalho sujas para regiões com fontes de energia mais sustentáveis.

48. Code With Me

Avalie

A ferramenta de programação colaborativa da JetBrains, Code With Me, tem ganhado popularidade à medida que mais times usam uma variedade de ferramentas JetBrains neste contexto remoto-primeiro. Junto com outras ferramentas de colaboração remota, como o Visual Studio Live Share da VSCode, Code With Me oferece aos times de desenvolvimento uma experiência aprimorada com pareamento e colaboração remotos. Vale explorar as habilidades do Code With Me para convidar colegas de equipe para projetos de IDE e colaborar em tempo real. No entanto, vimos algumas limitações em relação à refatoração de forma fluida e alguns problemas em ambientes de alta latência. Continuaremos observando essa ferramenta neste espaço.

49. Comby

Avalie

Incluímos nesta edição do Radar duas ferramentas que buscam e substituem código usando uma representação de árvore de sintaxe abstrata (AST). Ambas ocupam um espaço semelhante a jscodeshift, mas contêm analisadores para uma ampla gama de linguagens de programação. Embora compartilhem algumas semelhanças, também diferem de várias maneiras. Uma dessas ferramentas, Comby, é única em sua interface de linha de comando simples projetada no espírito das ferramentas Unix, como **awk** e **sed**. Enquanto os comandos Unix são baseados em expressões regulares que operam em texto correspondente, Comby emprega uma sintaxe padrão específica para construções de linguagem de programação e analisa o código antes de pesquisar. Isso ajuda as pessoas desenvolvedoras a buscar grandes bases de código para padrões estruturais. Assim como **sed**, Comby pode substituir os padrões combinados com novas estruturas. Isso é útil para automatizar alterações indiscriminadas em grandes bases de código ou para realizar alterações repetitivas em um conjunto de repositórios de microsserviços. Como essas ferramentas são relativamente novas, esperamos ver uma variedade de usos criativos que ainda não foram descobertos.

50. Conftest

Avalie

Conftest é uma ferramenta para escrever testes para dados de configuração estruturados que usa a Rego language do Open Policy Agent para escrever testes de configuração para Kubernetes, além de definições de pipeline Tekton ou até mesmo planos Terraform. As configurações são uma parte crítica da infraestrutura, e encorajamos você a avaliar o uso de Conftest para verificar hipóteses e obter feedback rápido.

51. Cosign

Avalie

Cosign é uma ferramenta de assinatura e verificação de contêiner. Parte do Sigstore — um projeto sob o guarda-chuva Cloud Native Computing Foundation (CNCF) que visa simplificar a assinatura e a transparência do software —, Cosign suporta não apenas imagens Docker e Open Container Initiative (OCI), mas também outros artefatos que podem ser armazenados em um registro de contêiner. Já falamos sobre o Docker Notary, que também atua neste espaço. O Notary v1, entretanto, tem algumas desvantagens: não é nativo de registro e precisa de um servidor de Notary separado. O Cosign evita esse problema e armazena as assinaturas no registro ao lado de uma imagem. Atualmente, tem integrações com GitHub actions e Kubernetes usando um Webhook com outras integrações no pipeline. Usamos Cosign em alguns de nossos projetos e parece bastante promissor.

52. Crossplane

Avalie

Crossplane é mais uma entrada na classe de ferramentas implementadas pelo padrão Kubernetes Operator, mas com efeitos colaterais que vão além do cluster Kubernetes. Na última edição do Radar, mencionamos a técnica de serviços de nuvem gerenciados por Kube, e o Crossplane faz exatamente isso. A ideia é aproveitar o plano de controle do Kubernetes para provisionar serviços em nuvem dos quais sua implantação depende, mesmo se eles forem implantados no próprio cluster. Os exemplos incluem instâncias de banco de dados gerenciado, balanceadores de carga ou políticas de controle de acesso. Essa ferramenta é notável por dois motivos. Primeiro, porque demonstra o ambiente de execução poderoso e flexível do plano de controle do Kubernetes subjacente. Não há limite real para a gama de recursos personalizados com suporte. Em segundo lugar, o Crossplane fornece uma alternativa às opções usuais Terraform, CDK ou Pulumi. Crossplane vem ainda com um conjunto de provedores predefinidos para os principais serviços em nuvem que cobrem os serviços mais comumente provisionados. Crossplane não tenta ser uma ferramenta de infraestrutura como código (IaC) de uso geral, mas sim um complemento das cargas de trabalho que estão sendo implantadas no Kubernetes. Frequentemente associado à prática do GitOps, Crossplane é autônomo e permite que você permaneça dentro do ecossistema Kubernetes quando for necessário gerenciar a nuvem externa de recursos. No entanto, Crossplane não ajuda no provisionamento do Kubernetes; você precisará de pelo menos uma outra ferramenta IaC para inicializar o cluster.

53. gopass

Avalie

gopass é um gerenciador de senhas para equipes, construído em GPG e Git. É descendente de pass e adiciona vários recursos, incluindo pesquisa interativa e vários armazenamentos de senha em uma única árvore. Desde que mencionamos o gopass pela primeira vez, nossos times o têm usado em vários projetos, às vezes estendendo-o além de seus limites. Um recurso que fazia falta era a capacidade de descontinuar segredos. A capacidade de descoberta já era um problema, mas a incapacidade de marcar segredos como “não mais em uso” agravou esse problema. O maior problema, porém, era a escala. Quando você tem times com mais de 50 pessoas usando o mesmo repositório por vários anos, descobrimos que o repositório pode crescer para vários gigabytes de tamanho. Criptografar novamente os segredos durante a integração de novos membros pode levar mais de meia hora. A questão subjacente parece ser que em nossos times tudo muda o tempo todo: as pessoas vêm e vão, os segredos são girados, a arquitetura evolui, novos segredos são adicionados, os antigos não são mais necessários. gopass parece funcionar bem, mesmo para um grande número de usuários, quando há menos mudanças.

54. Micoo

Avalie

Micoo é uma nova participante no concorrido espaço de ferramentas de regressão visual. É uma solução de código aberto e independente, fornecendo imagens Docker para permitir uma configuração de ambiente fácil e rápida. A ferramenta também fornece clientes diferentes para Node.js, Java e Python, bem como um plug-in Cypress, para uma integração simples com a maioria das estruturas ou soluções de teste de automação de interface do usuário de frontend comuns. Embora Micoo não forneça todas as funcionalidades que algumas das soluções baseadas em SaaS ou outras soluções comerciais fornecem, nossos times têm usado extensivamente e tiveram experiências positivas. Eles destacaram especialmente que a ferramenta funciona para aplicativos móveis e desktop, bem como para a web.

55. mob

Avalie

Às vezes, você se depara com uma ferramenta que não sabia que precisava até então. mob é uma dessas ferramentas. Em um mundo onde a programação remota em pares se tornou a norma para muitos times, ter uma ferramenta que permite uma troca fluida entre pares ou em um grupo mais amplo como parte de uma sessão de programação em grupo é super útil. mob esconde toda a parafernália de controle de versão atrás de uma interface de linha de comando que torna a participação nas sessões de programação em grupo mais simples. Também fornece instruções específicas para participar remotamente, por exemplo, "roubar o compartilhamento de tela" no Zoom em vez de encerrar o compartilhamento de tela, garantindo que o layout do vídeo não mude para demais participantes. Uma ferramenta útil e recomendações ponderadas, como não gostar?

56. Comandos Unix modernos

Avalie

Existem muitas razões para amar o Unix, mas a que afetou profundamente nossa indústria foi a filosofia Unix de construir aplicações que "fazem uma coisa e a fazem bem". Os comandos Unix incorporam essa filosofia. Um conjunto de pequenas funções que podem ser combinadas para criar soluções mais complexas. Nos últimos anos, pessoas programadoras contribuíram para um conjunto crescente de comandos Unix modernos. Essas versões modernas buscam ser menores e mais rápidas, geralmente escritas em Rust. Incluem recursos adicionais, como destaque de sintaxe e utilizam recursos de terminais modernos. Visam oferecer suporte nativo para pessoas desenvolvedoras, integrando-se bem com **`git`** e reconhecendo os arquivos de código-fonte. Por exemplo, `bat` é um substituto para **`cat`** com paginação e destaque de sintaxe; `exa` é uma substituição para **`ls`** com informações de arquivo estendidas e `ripgrep` é uma substituição mais rápida do **`grep`**, que por padrão ignora arquivos gitignore, binários e ocultos. O repositório Modern Unix faz referência a alguns desses comandos. Estamos gostando de usar esses comandos Unix. Recomendamos que você experimente-os para melhorar sua experiência de linha de comando. No entanto, evitamos usá-los em scripts como substitutos dos utilitários de linha de comando padrão fornecidos em distribuições de sistema operacional padrão, porque reduzem a portabilidade dos scripts em execução em outras máquinas.

57. Mozilla Sops

Avalie

Os segredos de texto simples verificados no controle de origem (geralmente Github) são um dos erros de segurança mais comuns cometidos por pessoas desenvolvedoras. Por esta razão, achamos útil apresentar Mozilla Sops, uma ferramenta para criptografar segredos em arquivos de texto que nossos times de desenvolvimento consideram útil em situações nas quais é impossível remover segredos de repositórios de código legados. Mencionamos muitas ferramentas desse tipo anteriormente (Blackbox, git-crypt), mas Sops conta com vários recursos que o diferenciam. Por exemplo, a ferramenta se integra a keystores gerenciadas em nuvem, como AWS e GCP Key Management Service (KMS) ou Azure Key Vault como fontes de chaves de criptografia. Também funciona em várias plataformas e oferece suporte a chaves PGP. Isso permite um controle de acesso refinado a segredos em uma base de arquivo por arquivo. Sops simplifica a chave de identificação em texto para que os segredos ainda possam ser localizados e difundidos pelo git. Sempre apoiamos qualquer coisa que facilite a segurança para as pessoas desenvolvedoras; no entanto, lembre-se de que você não precisa manter segredos no controle de origem em primeiro lugar. Consulte o blip sobre desacoplar gerenciamento de segredos do código-fonte em nossa edição de novembro de 2017.

58. Operator Framework

Avalie

Continuamos vendo a adoção do Kubernetes em cenários novos e inovadores. Por exemplo, observamos o Kubernetes sendo estendido para **gerenciar recursos em execução fora de seu cluster**, ou em **múltiplos provedores de infraestrutura**, ou sendo usado no gerenciamento de aplicações com estado além do escopo original do Kubernetes. Essas extensões são possíveis usando o padrão **Kubernetes Operator**: construir controladores Kubernetes que tenham o conhecimento específico do domínio do recurso personalizado que gerenciam. Por exemplo, um operador que gerencia uma aplicação com estado pode usar as primitivas Kubernetes para automatizar as tarefas específicas de uma aplicação além de sua implantação, como restaurar, fazer backup e atualizar seu banco de dados.

Operator Framework é um conjunto de ferramentas de código aberto que simplifica a construção e o gerenciamento do ciclo de vida dos **operadores Kubernetes**. Embora existam **vários frameworks** para ajudar na construção de operadores Kubernetes, o Operator Framework continua sendo uma boa escolha. O conjunto de ferramentas oferece suporte ao gerenciamento avançado do ciclo de vida do operador usando seu módulo **Operator Lifecycle Manager**; suporta várias linguagens para construir o código do operador em si usando seu **Operator SDK** e fornece um **catálogo** para publicar e compartilhar os operadores. Se você planeja criar operadores Kubernetes, recomendamos que experimente o Operator Framework para acelerar seu desenvolvimento de maneira confiável.

59. Pactflow

Avalie

Para organizações com ecossistemas de API maiores e mais complexos, especialmente aquelas que já usam **Pact**, achamos que vale avaliar se **Pactflow** pode ser útil. Pactflow gerencia o fluxo de trabalho e a implantação contínua de testes escritos em Pact, reduzindo a barreira para **testes de contrato orientados ao consumidor**. A complexidade da coordenação entre vários produtores e vários consumidores distintos pode se tornar proibitiva. Vimos algumas equipes investirem esforços significativos em soluções manuais para esse problema e achamos que vale a pena avaliar se o Pactflow pode cuidar disso para você.

60. Prefect

Avalie

Prefect é uma ferramenta de gerenciamento de fluxo de trabalho de dados que facilita a adição de semântica, como novas tentativas, mapeamento dinâmico, armazenamento em cache e notificações de falha em pipelines de dados. Você pode marcar funções Python como tarefas e encadeá-las por meio de chamadas de função para construir o fluxo de dados. A API Python combinada com uma coleção de tarefas predefinidas para operações de dados comuns torna Prefect uma opção notável para avaliar considerando suas necessidades de pipeline de dados.

61. Proxyman

Avalie

Pode não ser uma ferramenta que você precisa usar todos os dias, mas quando você está tentando diagnosticar um problema de rede desagradável, a capacidade de recorrer a um proxy de depuração HTTP rico em recursos é muito útil. **Proxyman** é uma dessas ferramentas. Vários de nossos times têm o usado já há algum tempo como um substituto direto para **Charles** específico para macOS, e realmente gostam de sua interface simplificada e gerenciamento de certificados.

62. Regula

Avalie

Um dos principais princípios da infraestrutura como código (IaC) é o teste automatizado. Se tivermos uma pirâmide de teste sólida com boa cobertura de nível de código na parte inferior, podemos produzir uma infraestrutura melhor e mais segura. Infelizmente, as ferramentas para auxiliar neste espaço têm sido esparsas. **Conftest** é frequentemente usado para testar código Terraform JSON e HCL, mas é uma ferramenta de uso geral. **Regula** é uma alternativa atraente. Semelhante ao Conftest, Regula verifica a conformidade do código da infraestrutura aplicando regras escritas na linguagem Rego do Open Policy Agent, mas também fornece um conjunto de primitivas especificamente para validar as configurações da infraestrutura. Como ambas as ferramentas são baseadas na linguagem Rego, as regras do Regula podem ser executadas pelo Conftest. No entanto, Regula vem com sua própria ferramenta de linha de comando para executar testes como parte de um pipeline, sem dependência de Conftest ou OPA. Nossos times de desenvolvimento descobriram que o Regula economiza tempo e produz um código de teste muito mais legível, sustentável e sucinto. Ainda assim, ambas as ferramentas validam apenas o código de infraestrutura. Um conjunto completo também deve testar a infraestrutura para garantir que o código está sendo interpretado com precisão.

63. Sourcegraph

Avalie

Outra ferramenta de pesquisa de código baseada em árvore de sintaxe abstrata que ganhou nossa atenção é **Sourcegraph**. Ao contrário de **Comby**, que tem código aberto, Sourcegraph é uma ferramenta comercial (com um plano gratuito para até 10 usuários). Sourcegraph é particularmente adequado para pesquisa, navegação ou referência cruzada em grandes bases de código. A versão hospedada na nuvem pode ser acessada pelo site do Sourcegraph e é projetada para pesquisar repositórios de código aberto disponíveis publicamente. Enquanto Comby é uma ferramenta de linha de comando leve para automatizar tarefas repetitivas, a ênfase de Sourcegraph está em ferramentas de desenvolvimento interativas para compreender e navegar em grandes bases de código. Ao contrário da interface semelhante ao **sed** do Comby, o recurso de reescrita de código automatizado do Sourcegraph é conduzido a partir de uma IU, permitindo que os usuários revisem as alterações antes de serem feitas. Como o Sourcegraph é um serviço hospedado, também tem a capacidade de monitorar continuamente as bases de código e enviar alertas quando ocorre uma correspondência.

64. Telepresence

Avalie

Telepresence é uma ferramenta que ajuda a reduzir o ciclo de feedback de mudanças que geralmente requerem uma implantação para testes adequados. As pessoas desenvolvedoras podem usá-la para conectar um processo em execução em sua máquina local a um cluster Kubernetes remoto. Isso dá ao processo local acesso aos serviços e recursos do cluster remoto, e o serviço local também pode substituir temporariamente um dos serviços do cluster.

Em situações nas quais a configuração da integração de serviço se tornou complicada, Telepresence pode aumentar a produtividade das pessoas desenvolvedoras e possibilitar testes locais mais efetivos. No entanto, adquirir o hábito de usar uma ferramenta inteligente como essa pode trazer problemas maiores. Por exemplo, se você está usando Telepresence porque configurar todas as dependências necessárias para o desenvolvimento local se tornou uma tarefa impossível, é importante que você examine a complexidade da sua configuração e da arquitetura. Se for a única maneira de fazer testes de integração de serviço, considere **testes de contrato** ou outras formas automatizadas de teste de integração.

65. Vite

Avalie

O feedback rápido é crucial para uma boa experiência de desenvolvimento. Nada interrompe mais o fluxo de desenvolvimento do que ter que esperar um ou dois minutos antes de obter feedback sobre as últimas alterações de código. Infelizmente, com as aplicações crescendo em tamanho e complexidade, as ferramentas de build para pipelines de front-end populares muitas vezes não são mais rápidas o suficiente. Anteriormente, incluímos **esbuild**, que oferece uma melhoria de desempenho significativa, porque é implementado em uma linguagem compilação-para-nativa em vez de JavaScript. **Vite**, que é construído em cima do esbuild, oferece **melhorias significativas** em relação a outras ferramentas. Consiste em duas partes principais: um servidor de desenvolvimento que fornece aprimoramentos de recursos ricos em módulos ES nativos, como Hot Module Replacement (HMR) extremamente rápido e um comando de construção que agrupa seu código com Rollup. Vite depende de módulos ES e, ao contrário da maioria das ferramentas mais antigas, não oferece shimming ou polyfills, o que significa que não é compatível com navegadores mais antigos que não suportam módulos ES. Nos casos em que navegadores mais antigos precisavam ser suportados, alguns de nossos times usaram Vite durante o desenvolvimento e outras ferramentas para builds de produção.

Linguagens e frameworks

Adote

66. Jetpack Compose
67. React Hooks

Experimente

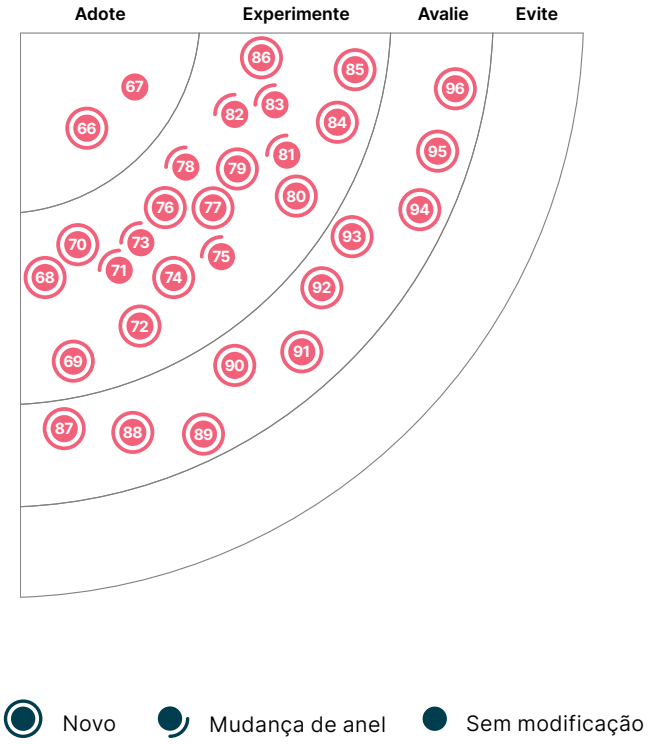
68. Arium
69. Chakra UI
70. DoWhy
71. Gatsby.js
72. Jetpack Hilt
73. Kotlin Multiplatform Mobile
74. lifelines
75. Mock Service Worker
76. NgRx
77. pydantic
78. Quarkus
79. React Native Reanimated 2.0
80. React Query
81. Tailwind CSS
82. TensorFlow Lite
83. Three.js
84. ViewInspector
85. Vowpal Wabbit
86. Zap

Avalie

87. Headless UI
88. InsightFace
89. Kats
90. ksqlDB
91. Polars
92. PyTorch Geometric
93. Qiankun
94. React Three Fiber
95. Tauri
96. Transloco

Evite

—



66. Jetpack Compose

Adote

Em uma mudança que reflete a introdução do SwiftUI pela Apple, o Google introduziu o Jetpack Compose como uma abordagem nova e bem diferente para construir interfaces de usuário para aplicativos Android modernos. O Compose traz ferramentas mais poderosas e uma API Kotlin intuitiva. Na maioria dos casos, menos código é necessário, e ficou mais fácil criar interfaces de usuário em tempo de execução em vez de definir uma IU estática que pode ser preenchida com dados. Com Compose Multiplatform e Kotlin Multiplatform as pessoas desenvolvedoras agora têm um kit de ferramentas unificado para construir aplicações nativas para desktop, web e Android. O Wear OS 3.0+ também está incluído e com suporte para iOS já presente no Kotlin Multiplatform Mobile, é provável que o iOS seja compatível com Compose no futuro.

67. React Hooks

Adote

React Hooks introduziram uma nova abordagem para gerenciar lógica com estado. Considerando que os componentes React sempre estiveram mais próximos das funções do que das classes, os Hooks adotaram isso e trouxeram estado para as funções, em vez de usar classes para levar a função ao estado com métodos. Outro elemento básico do gerenciamento de estado em aplicações React é o Redux, e já observamos que a biblioteca está sob escrutínio, o que sugere que às vezes a complexidade do Redux não vale a pena e, em tais casos, uma abordagem simples usando Hooks é preferível. Fazer rollout disso pode rapidamente se tornar complicado caso você esteja trabalhando individualmente; portanto, recomendamos considerar uma combinação de React Context e os hooks useContext e useReducer, como explicado neste blog post.

68. Arium

Experimente

Arium é um framework de teste automatizado para aplicativos 3D escritos em Unity. Os testes funcionais são uma parte importante de uma pirâmide de teste saudável. Arium, que é construído como um wrapper no framework Unity Test, permite escrever testes funcionais para aplicativos 3D em várias plataformas. Nós o usamos com sucesso em alguns de nossos projetos.

69. Chakra UI

Experimente

Chakra UI é uma biblioteca de componentes de IU para React.js projetada para acessibilidade. Gostamos especialmente por seus recursos de acessibilidade, incluindo modo escuro e compatibilidade com as diretrizes da Web Accessibility Initiative – Accessible Rich Internet Applications (WAI-ARIA). Além disso, é fácil de testar e customizar, o que contribui para uma boa experiência de desenvolvimento, acelerando o processo de desenvolvimento de soluções de IU em ambientes de produção.

70. DoWhy

Experimente

DoWhy é uma biblioteca Python para realizar inferência e análise causais de ponta a ponta. Embora os modelos de aprendizado de máquina possam fazer previsões com base em dados factuais, explorando a correlação de variáveis presentes no momento, são insuficientes em cenários em que precisamos questionar e *se* e *por que*: *e se uma variável mudasse? Qual seria o impacto no resultado?* A inferência causal é uma abordagem para responder a essas perguntas, estimando o efeito causal, ou seja, a magnitude pela qual um resultado mudaria, se mudássemos uma das variáveis causais. Esta abordagem é aplicada quando não podemos chegar à resposta por meio de observações e coleta de dados de testes A/B — devido ao custo dos experimentos ou limitações.

A biblioteca DoWhy estima o efeito causal com base em um processo que usa os fatos e dados coletados no passado, bem como as suposições que podem ser feitas conhecendo o domínio. DoWhy usa um processo de quatro etapas de modelagem do gráfico de relações causais com base em suposições, identificando uma causa para um resultado, estimando o efeito causal e, finalmente, desafiando essas suposições, refutando o resultado. Usamos essa biblioteca com sucesso em produção e é uma das bibliotecas comumente usadas em casos de uso de estimativa causal.

71. Gatsby.js

Experimente

Embora vários frameworks prometam a mesma facilidade de desenvolvimento e escalabilidade típicas de geradores de sites estáticos, continuamos tendo boas experiências com **Gatsby.js**. Particularmente, nós o usamos para construir e implantar sites que podem ser escalados para um grande número de usuários, sem precisarmos nos preocupar com o planejamento de capacidade ou a infraestrutura de implantação. Nossos times de desenvolvimento também ficaram impressionados com o foco na acessibilidade, suporte para navegadores antigos e o fato de poderem reutilizar sua experiência com **React.js**. Em resumo, sentimos que Gatsby amadureceu bem e é uma escolha sólida neste espaço.

72. Jetpack Hilt

Experimente

Jetpack Hilt chegou recentemente à versão 1.0 e podemos dizer que tivemos boas experiências com o framework. O Jetpack Hilt fornece extensões para integrar Hilt com várias outras bibliotecas AndroidX, como WorkManager e Navigation, expandindo ainda mais o alcance do Hilt para oferecer às pessoas desenvolvedoras uma maneira padrão de incorporar a injeção de dependência **Dagger** em aplicativos Android. Já destacamos **Koin** como um framework de injeção de dependência nativo do Kotlin anteriormente no Radar, e recomendamos não tentar substituí-lo em uma base de código extensa já existente. No entanto, para iniciar novos projetos, Hilt parece ser o caminho a se seguir atualmente.

73. Kotlin Multiplatform Mobile

Experimente

Para muitas organizações, o desenvolvimento móvel multiplataforma vem se tornando uma opção forte, especialmente à medida que a experiência de ponta a ponta de construção de aplicativos móveis multiplataforma se torna mais agradável e eficiente. **Kotlin Multiplatform Mobile** (KMM) é um SDK fornecido pela JetBrains que aproveita os **recursos multiplataforma** em **Kotlin** e inclui ferramentas e recursos projetados para otimizar a experiência de desenvolvimento. Com o KMM, você escreve o código uma vez para a lógica de negócio e o core do aplicativo em Kotlin, e em seguida o compartilha com aplicativos Android e iOS. Você pode escrever código específico para plataforma apenas quando necessário, por exemplo, para aproveitar as vantagens dos elementos nativos da IU, e o código específico é mantido em visualizações diferentes para cada plataforma. Estamos mudando o KMM para o anel Experimente devido à sua **rápida evolução** e vemos algumas organizações usando como padrão.

74. lifelines

Experimente

lifelines é uma biblioteca para análise de sobrevivência em Python. Originalmente desenvolvida para eventos de nascimento e morte, evoluiu para uma biblioteca de análise de sobrevivência completa para prever qualquer tempo de duração. Além dos casos de uso médico (como responder: *por quanto tempo essa população vive em média?*), nós o usamos no varejo e na indústria para responder a perguntas como: *há quanto tempo o usuário está inscrito em um serviço?*; ou *quando devemos realizar a próxima manutenção preventiva?*

75. Mock Service Worker

Experimente

Aplicações web, especialmente as que se destinam a uso interno em empresas, geralmente são escritas em duas partes. A interface do usuário e alguma lógica de negócio são executadas no navegador web, enquanto a maior parte da lógica de negócio, além de autorização e persistência, são executadas em um servidor. Essas duas metades normalmente se comunicam via JSON sobre HTTP. Os endpoints não devem ser confundidos com uma API real, são simplesmente um detalhe de implementação de uma aplicação dividida em dois ambientes de tempo de execução. Ao mesmo tempo, fornecem uma costura válida para testar as peças individualmente. Ao testar a parte do JavaScript, o lado do servidor pode ser um mock ou stub no nível da rede usando uma ferramenta como **Mountebank**. **Mock Service Worker** oferece uma abordagem alternativa para interceptar solicitações no navegador. Isso também simplifica os testes manuais. Como Mountebank, o Mock Service Worker é executado fora do navegador como um processo Node.js para testar interações de rede. Além das interações REST, também executa mocks de APIs GraphQL — um bônus, já que pode ser complexo simular manualmente no nível da rede com GraphQL.

76. NgRx

Experimente

Gerenciamento de estado em aplicações React tem sido um tópico recorrente no Radar, e recentemente esclarecemos nossa posição sobre **Redux**, um framework popular neste espaço. **NgRx** é, em essência, Redux para **Angular** . É um framework para construir aplicações reativas com Angular, fornecendo maneiras de gerenciar o estado e isolar os efeitos colaterais. Nossos times relatam que usar o NgRx foi simples, principalmente porque é construído com **RxJS**, e destacam um trade-off semelhante àquele que já conhecemos do Redux: adicionar gerenciamento de estado reativo traz uma complexidade adicional que só compensa em aplicações maiores. A experiência de desenvolvimento é aprimorada por esquemas, uma biblioteca de scaffolding, além de um conjunto de ferramentas que permitem o rastreamento visual de estado e depuração de viagem no tempo.

77. pydantic

Experimente

Originalmente, as anotações de tipo foram adicionadas ao Python para oferecer suporte à análise estática. No entanto, considerando o quão amplamente as anotações de tipo, e as anotações em geral, são usadas em outras linguagens de programação, era apenas uma questão de tempo antes que as pessoas desenvolvedoras começassem a usar as anotações de tipo do Python para outros fins. **pydantic** se enquadra nesta categoria. A biblioteca permite que você use anotações de tipo para validação de dados e gerenciamento de configurações em tempo de execução. Quando os dados chegam como, digamos, um documento JSON e precisam ser analisados em uma estrutura Python complexa, pydantic garante que os dados recebidos correspondam aos tipos esperados ou relata um erro caso não correspondam. Embora você possa usar pydantic diretamente, muitas pessoas desenvolvedoras têm o usado como parte do **FastAPI**, um dos frameworks Python para web mais populares. Na verdade, o uso de pydantic em FastAPI é considerado tão indispensável que uma alteração recentemente proposta para Python, com o objetivo de reduzir o custo de carregamento de código anotado na memória, **foi reconsiderada** porque teria quebrado o uso de anotações de tipo em tempo de execução.

78. Quarkus

Experimente

Avaliamos **Quarkus** há dois anos, e agora nossos times já têm mais experiência usando-a. Quarkus é uma stack Java nativa do Kubernetes, adaptada para OpenJDK HotSpot e **GraalVM**. Nos últimos dois anos, Quarkus conectou as melhores bibliotecas do mundo Java e simplificou a configuração

do código, proporcionando a nossos times uma experiência de desenvolvimento muito boa. Quarkus tem um tempo de inicialização muito rápido (dezenas de milissegundos) e um baixo consumo de memória RSS; isso ocorre por causa de sua abordagem de compilação **contêiner-primeiro**, que usa técnicas de compilação antecipada para fazer injeção de dependência em tempo de compilação e, assim, evita a execução custos de tempo de reflexão. Nossos times também tiveram que lidar com as desvantagens: Quarkus leva quase dez minutos para compilar em nosso pipeline; alguns recursos que dependem de anotações e reflexão (como ORM e serializador) também são limitados. Em parte, essas desvantagens são resultado do uso do GraalVM. Portanto, se sua aplicação não está rodando para FaaS, usar Quarkus com HotSpot também é uma boa escolha.

79. React Native Reanimated 2.0

Experimente

Quando queremos animações em aplicativos **React Native**, **React Native Reanimated 2.0** é o caminho a seguir. Anteriormente, usamos Reanimated 1.x, mas havia problemas relacionados à complexidade da linguagem declarativa Reanimated e também alguns custos de desempenho adicionais relacionados à inicialização e comunicação entre o thread React Native JavaScript e o thread de IU. Reanimated 2.0 é uma tentativa de reimaginar a maneira de executar animações no thread de interface do usuário, permitindo programar as animações em JavaScript e executá-las no thread de IU usando uma nova API chamada **animation worklets**. A biblioteca faz isso gerando um contexto JavaScript secundário no thread de interface do usuário que, então, é capaz de executar funções JavaScript. Estamos usando em nossos projetos React Native que precisam de animações e gostamos bastante.

80. React Query

Experimente

React Query é frequentemente descrita como a biblioteca de busca de dados para React que faltava. 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. React Query fornece uma solução simples e direta usando hooks. As pessoas desenvolvedoras de aplicativos podem simplesmente passar uma função que resolve seus dados e deixar todo o resto para o framework. Gostamos de sua funcionalidade out-of-the-box, mas também das muitas possibilidades de configuração quando necessário. As ferramentas para pessoas desenvolvedoras, infelizmente ainda não disponíveis para React Native, ajudam na compreensão de como o framework funciona, beneficiando quem está começando a usá-lo. Em nossa experiência, a versão 3 do framework trouxe a estabilidade necessária para ser usada em produção em nossos projetos de clientes.

81. Tailwind CSS

Experimente

Nossos times de desenvolvimento continuaram sendo produtivos com Tailwind CSS e estão impressionados com sua capacidade de escala para grandes equipes e bases de código. **Tailwind CSS** oferece uma abordagem alternativa para ferramentas e frameworks CSS que reduz a complexidade por meio de classes CSS de utilitário de nível inferior. As classes CSS de Tailwind podem ser facilmente personalizadas para se adequar à identidade visual de qualquer cliente. Também descobrimos que o framework trabalha particularmente bem com **Headless UI**. O CSS de Tailwind permite que você evite escrever quaisquer classes ou CSS por conta própria, o que leva a uma base de código mais sustentável a longo prazo. Tailwind CSS parece oferecer o equilíbrio certo entre capacidade de reutilização e personalização para criar componentes visuais.

82. TensorFlow Lite

Experimente

Desde que mencionamos pela primeira vez o **TensorFlow Lite** no Radar em 2018, nós o usamos em vários produtos e temos o prazer de informar que o framework funciona como anunciado. O caso de uso padrão é integrar modelos pré-treinados em aplicativos móveis, mas o TensorFlow Lite também oferece suporte ao aprendizado no dispositivo, o que expande as possibilidades de aplicação. Os vários exemplos no site mostram muitas áreas comuns de aplicação, como classificação de imagens e detecção de objetos, mas também sugerem novas formas de interação usando, por exemplo, estimativa de pose e reconhecimento de gestos.

83. Three.js

Experimente

Mencionamos pela primeira vez o **Three.js** no Radar no anel Avalie em 2017. Desde então, esta biblioteca de renderização 3D para a web evoluiu e melhorou rapidamente. As APIs WebGL padrão foram aprimoradas e o Three.js adicionou suporte para WebXR, tornando-o uma ferramenta viável para a criação de experiências imersivas. Ao mesmo tempo, o suporte do navegador para renderização 3D e APIs de dispositivos WebXR melhorou, tornando a web uma plataforma cada vez mais atraente para conteúdo 3D. Embora existam outras bibliotecas de renderização 3D, nossos times passaram a preferir Three.js, especialmente quando pareado com **React Three Fiber** para abstrair alguns dos detalhes de baixo nível. Descobrimos que as pessoas desenvolvedoras ainda precisam estar cientes dos problemas de desempenho e, às vezes, precisam reestruturar os dados para otimizar a velocidade de renderização.

84. ViewInspector

Experimente

Ao criar uma interface de usuário com **SwiftUI**, a ideia é construir um modelo de visualização que possa ser mapeado facilmente para os elementos da IU. Nesses casos, a maioria dos testes pode ser feita no modelo, usando frameworks de teste de unidade padrão, o que torna esses testes fáceis de escrever e rápidos de executar. Para testar as ligações entre o modelo e as visualizações, as pessoas desenvolvedoras geralmente usam **XCUITest**, um framework de automação de teste que inicia o aplicativo completo e controla remotamente a interface. Funciona e os testes são razoavelmente estáveis, mas demoram muito para serem executados.

Para uma abordagem mais rápida para escrever testes de unidade para SwiftUI, tente **ViewInspector**, um framework de código aberto que usa a API de reflexão pública do Swift para acessar as visualizações subjacentes criadas por SwiftUI. Com o ViewInspector, um teste simplesmente instancia uma visualização SwiftUI, localiza os elementos da interface que precisam ser testados e, em seguida, faz afirmações contra esses elementos. As interações básicas, como taps, também podem ser testadas. Como muitos frameworks de teste de IU, fornece uma API para localizar elementos de interface, seja especificando um caminho por meio da hierarquia de visualização ou usando um conjunto de métodos localizadores. Esses testes são geralmente mais simples do que os XCUITests e são executados com muito mais rapidez. Como uma palavra de cautela, no entanto, dada a facilidade com que os testes podem ser escritos usando ViewInspector, você pode acabar tendo que lidar com a tentação de testar demais a interface. Testar mapeamentos um-para-um simples é apenas um registro de partidas dobradas. E embora o ViewInspector torne mais fácil testar o código SwiftUI, lembre-se de manter a maior parte da lógica no modelo.

85. Vowpal Wabbit

Experimente

Vowpal Wabbit é uma biblioteca de aprendizado de máquina de uso geral. Mesmo que tenha sido originalmente criada no Yahoo! Research há mais de uma década, ainda queremos mencioná-lo para destacar que continua sendo onde muitas das mais novas técnicas de aprendizado de máquina são adicionadas primeiro. Se você tiver interesse em aprendizado de máquina, fique de olho nas inovações do Vowpal Wabbit. Observe também que a Microsoft mostrou um interesse mais profundo no Vowpal Wabbit nos últimos anos, empregando um grande contribuidor e integrando-o em suas ofertas do Azure, por exemplo em seu **designer de aprendizado de máquina** e no **Personalizer**.

86. Zap

Experimente

Zap é uma biblioteca de registro estruturada de alto desempenho para GoLang, mais rápida do que a implementação padrão de log e outras bibliotecas de log. Zap tem um logger "bonito", fornecendo uma interface estruturada e no estilo **`printf`**, bem como uma implementação (ainda) mais rápida com apenas a interface estruturada. Nossos times a têm usado amplamente em grande escala e estão felizes em recomendá-la como sua solução ideal.

87. Headless UI

Avalie

Headless UI é uma biblioteca de componentes sem estilo para React.js ou Vue.js, desenvolvida pelas mesmas pessoas que criaram o Tailwind CSS. Nossos times de desenvolvimento gostam do fato de não terem que personalizar ou contornar os estilos padrão que vêm com outras bibliotecas de componentes. A rica funcionalidade e acessibilidade total dos componentes, combinadas com o estilo sem atrito, permitem que as pessoas desenvolvedoras se concentrem de forma mais produtiva no problema de negócio e na experiência do usuário. Como seria esperado, Headless UI também funciona bem com as classes CSS do Tailwind.

88. InsightFace

Avalie

InsightFace é uma caixa de ferramentas de código aberto para análise profunda de face em 2D e 3D, baseada principalmente em PyTorch e MXNet. InsightFace usa alguns dos métodos mais recentes e precisos para detecção, reconhecimento e alinhamento faciais. Particularmente, a ferramenta nos interessa por contar com uma das melhores implementações para ArcFace, um modelo de aprendizado de máquina de ponta que detecta as semelhanças entre duas imagens. InsightFace com ArcFace recebeu uma pontuação de precisão de 99,83% no conjunto de dados Labeled Faces in the Wild (LFW). Estamos testando no contexto da desduplicação facial e vimos resultados promissores.

89. Kats

Avalie

Kats é um framework leve para realizar análises de séries temporais, recentemente lançado pelo Facebook Research. A análise de séries temporais é uma área importante na ciência de dados, abrangendo os domínios do problema de previsão, detecção (incluindo a detecção de sazonalidades, outliers e pontos de mudança), extração de características e análise multivariada. Normalmente, tendemos a ter diferentes bibliotecas para diferentes técnicas em uma análise de série temporal. Kats, entretanto, pretende ser um balcão único para análises de séries temporais e fornece um conjunto de algoritmos e modelos para todos os domínios de problemas de análise de séries temporais. Anteriormente mencionamos Prophet, também do Facebook Research, que é um dos modelos que Kats implementa para previsão. Temos boas expectativas para testar Kats em problemas envolvendo análises de séries temporais.

90. ksqlDB

Avalie

Se você estiver usando Apache Kafka e criando aplicações de processamento de fluxo, ksqlDB é um ótimo framework para escrever aplicações simples usando instruções semelhantes a SQL. ksqlDB não é um banco de dados SQL tradicional. No entanto, permite que você use instruções semelhantes a SQL leves para construir novos streams ou tabelas usando como base os tópicos Kafka existentes. As consultas podem extrair dados, semelhante à leitura de um banco de dados tradicional, ou enviar resultados para a aplicação quando ocorrem mudanças incrementais. Você pode optar por executá-lo como um servidor autônomo nativamente, como parte da instalação existente do Apache Kafka ou como um serviço totalmente gerenciado na Confluent Cloud. Estamos usando ksqlDB em casos de uso de processamento de dados simples. Em casos de uso mais complexos, nos quais uma aplicação requer código de programação além de consultas SQL algébricas, continuamos usando frameworks de processamento de dados como Apache Spark ou Apache Flink tendo Kafka como base. Recomendamos experimentar ksqlDB nos cenários em que a simplicidade da aplicação permita.

91. Polars

Avalie

Polars é uma biblioteca de estruturas de dados em memória implementada em Rust. Ao contrário de outras estruturas de dados (como Pandas), Polars é multithread e segura para operações paralelas. Os dados em memória são organizados no formato Apache Arrow para fornecer operações analíticas eficientes e interoperabilidade com outras ferramentas. Se você tem familiaridade com Pandas, vai se adaptar rapidamente às ligações Python de Polars. Acreditamos que Polars, com implementação em Rust e ligações Python, é uma estrutura de dados em memória com bom desempenho para ser avaliada conforme suas necessidades analíticas.

92. PyTorch Geometric

Avalie

PyTorch Geometric é uma biblioteca de extensão de aprendizagem profunda geométrica para PyTorch. O aprendizado profundo geométrico visa construir redes neurais que possam aprender com dados não euclidianos, como grafos. Abordagens de aprendizado de máquina (ML) baseado em rede gráfica têm despertado interesse crescente na modelagem de redes sociais e nos campos biomédicos, especificamente na descoberta de medicamentos. PyTorch Geometric fornece uma biblioteca fácil de usar para projetar problemas complicados de rede de grafos, como a representação da estrutura de proteínas. Possui suporte para GPU e CPU e inclui uma boa coleção de algoritmos de ML baseados em grafos fundamentados em pesquisas recentes.

93. Qiankun

Avalie

Micro frontends continuaram ganhando popularidade desde que foram introduzidos pela primeira vez. No entanto, é fácil cair na anarquia de micro frontends se os times não conseguirem manter a consistência em uma aplicação, desde a técnica de estilo até a gestão de estado. Qiankun, que significa céu e terra em chinês, é uma biblioteca JavaScript construída para fornecer uma solução pronta para uso com esse fim. Qiankun é baseado em single-spa, portanto, permite que diferentes frameworks coexistam em uma única aplicação. Também fornece isolamento de estilo e sandbox de JavaScript para garantir que o estilo ou estado das microaplicações não interfiram um no outro. Qiankun recebeu alguma atenção na comunidade; nossos times também estão o avaliando, com a expectativa de que possa suportar uma depuração mais amigável.

94. React Three Fiber

Avalie

Com o crescente interesse e — e a viabilidade — de aplicações 3D e realidade estendida (XR) em navegadores web, nossos times vem experimentando **React Three Fiber** para o desenvolvimento de experiências 3D na web. React Three Fiber é uma biblioteca que pega o componente React.js e o modelo de estado e os converte em objetos 3D renderizados com o **Three.js**. Essa abordagem abre a programação 3D da web para o grupo mais amplo de pessoas desenvolvedoras já familiarizadas com React.js e o rico ecossistema de ferramentas e bibliotecas que o cercam. No entanto, ao desenvolver aplicações com React Three Fiber, nossos times geralmente precisam manipular a cena 3D de maneira imperativa. Isso não combina bem com o paradigma do componente reativo fornecido pelo React. Não há como escapar da necessidade de entender os mecanismos básicos de renderização 3D. Ainda não há consenso sobre o React Three Fiber oferecer abstração suficiente para garantir o aprendizado de suas idiossincrasias, ou se é melhor apenas trabalhar com o Three.js diretamente.

95. Tauri

Avalie

Tauri é uma alternativa a **Electron** para criar aplicações desktop usando uma combinação de ferramentas **Rust** e HTML, CSS e JavaScript renderizados no WebView do sistema. Ao contrário do Electron, que empacota o Chromium, as aplicações desenvolvidas com Tauri usam o WebView subjacente, ou seja, WebKit no macOS, WebView2 no Windows e WebKitGTK no Linux. Essa abordagem tem compensações interessantes: por um lado, você obtém binários pequenos e rápidos nas aplicações; por outro lado, você precisa verificar as peculiaridades de compatibilidade entre WebViews de sistemas diferentes.

96. Transloco

Avalie

Transloco é uma biblioteca do Angular para construir aplicativos multilíngues. Pode ser usado em modelos e oferece uma função para cobrir casos de uso mais complexos. Como as traduções são carregadas sob demanda em tempo de execução, Transloco facilita a implementação da troca de idioma no navegador web. Também cobre a localização de números, datas e muito mais usando template pipes.

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 10 mil pessoas distribuídas entre 48 escritórios e em 17 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