/thoughtworks

TECHNOLOGY RADAR

Um guia com opiniões firmes
sobre as fronteiras da tecnologia

Volume 24

#TWTechRadar
thoughtworks.com/radar

Contribuições

O Technology Radar é produzido pelo Conselho Consultivo de Tecnologia da Thoughtworks

O Conselho Consultivo de Tecnologia (TAB) é um grupo formado por 20 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 tecnologias 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.

Rebecca Parsons (CTO)

Martin Fowler (Chief Scientist)

Bharani Subramaniam

Birgitta Böckeler

Brandon Byars

Camilla Crispim

Cassie Shum

Erik Dörnenburg

Evan Bottcher

Fausto de la Torre

Hao Xu

Ian Cartwright

James Lewis

Lakshminarasimhan Sudarshan

Mike Mason

Neal Ford

Perla Villarreal

Rachel Laycock

Scott Shaw

Shangqi Liu

Zhamak Dehghani

Tradução: Angelica de Oliveira, Camila Bastos, Camilla Crispim, Edlaine Zamora, Gregório Melo, Luiza Souza, Paula Ribas, Ricardo Cavalcanti, Tania Gonzales e Thayse Onofrio

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 pessoas interessadas, de pessoas que desenvolvem software a CTOs. O conteúdo é concebido para ser um resumo conciso.

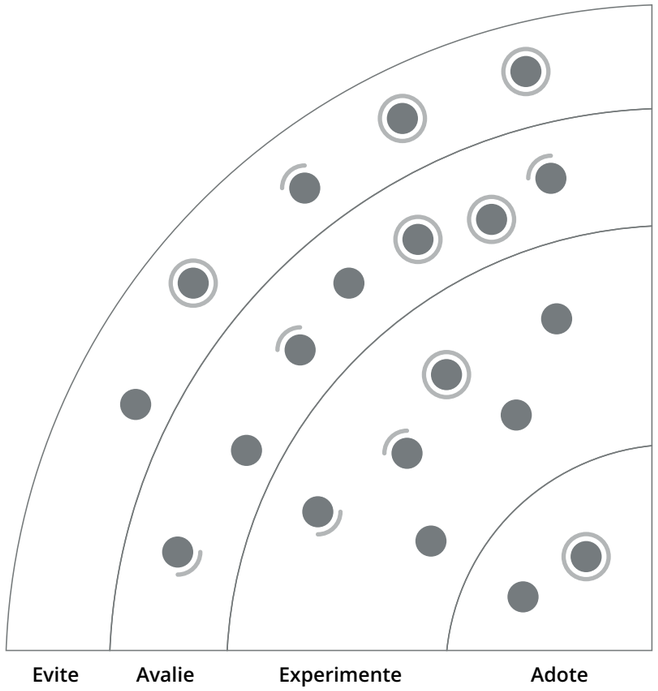
Nós encorajamos você a explorar essas tecnologias. O Radar é gráfico por natureza, agrupando os itens em técnicas, ferramentas, plataformas e linguagens & frameworks. No caso de itens que podem ser classificados em mais de um quadrante, escolhemos aquele que parece mais adequado. Além disso, agrupamos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

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, o que chamamos de blips. Organizamos os blips no Radar usando duas categorias: quadrantes e anéis. Os quadrantes representam as diferentes naturezas dos blips. Os anéis indicam em que estágio do ciclo de adoção consideramos que cada blip esteja.

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



Novo

Modificado

Sem modificação

Nosso Radar é um olhar para o futuro. Para abrir espaço para novos itens, apagamos itens em que não houve mudança recente, o que não é uma representação de seu valor, mas uma solução para nossa limitação de espaço.

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 experimentar esta tecnologia em um projeto que possa lidar com o risco.

Avalie

Vale a pena explorar com o objetivo de compreender como isso afetará sua empresa.

Evite

Prossiga com cautela.

Temas desta edição

Times de plataforma aumentam a velocidade de chegada ao mercado

Cada vez mais, as organizações vêm adotando um conceito de times de plataforma: a ideia é estabelecer um grupo dedicado a criar e fornecer suporte a recursos de plataforma internas — abordagens nativas de nuvem, entrega contínua, observabilidade moderna, padrões AuthZ/N, malha de serviços, entre outros — e, em seguida, usar esses recursos para acelerar o desenvolvimento de aplicações, reduzir a complexidade operacional e melhorar o tempo de chegada ao mercado. Apresentamos esta técnica pela primeira vez no Radar em 2017, e sua maturidade crescente é bem-vinda. Mas com essa maturidade crescente, também começamos a descobrir antipadrões que as organizações devem evitar. Por exemplo, "uma plataforma para reger todas as outras" pode não ser a abordagem ideal. Uma abordagem "big platform up front" pode levar anos para entregar valor, enquanto apostar na abordagem "construa e as pessoas usarão" pode acabar sendo um esforço desperdiçado. Em vez dessas, usar uma abordagem de mentalidade de produto pode ajudar a evidenciar o que suas plataformas internas devem oferecer, dependendo da base de clientes de cada uma delas. As empresas que colocam seus times de plataforma atrás de um sistema de tickets, como um silo de operações à moda antiga, encontram as mesmas desvantagens da priorização sem alinhamento: feedback e resposta lentos, contenção de alocação de recursos e outros problemas já conhecidos provocados pelo silo. Também temos visto várias novas ferramentas e padrões de integração para times e tecnologias surgindo, permitindo uma divisão mais eficaz de ambos.

Conveniência consolidada acima das melhores opções

À medida que as práticas de engenharia que incluem automação, escala, entre outras metas modernas, se tornam mais comuns entre os times de desenvolvimento, vemos a integração de ferramentas de desenvolvimento correspondentes em muitas plataformas, principalmente no espaço da nuvem. Por exemplo, repositórios de artefatos, controle de versão, pipelines de CI/CD, wikis e ferramentas semelhantes geralmente eram escolhidas a dedo pelos times de desenvolvimento e agrupados à la carte. Agora, plataformas de entrega como Azure DevOps e ecossistemas como GitHub englobam muitas dessas categorias de ferramentas. Embora o nível de maturidade varie entre as ofertas de plataforma, o apelo de uma "loja que tem de tudo" para ferramentas de entrega é inegável. De forma geral, parece que os pontos positivos e negativos se concentram entre ter stacks consolidadas de ferramentas, que oferecem maior conveniência às pessoas desenvolvedoras e menos rotatividade, embora o conjunto de ferramentas raramente represente a melhor escolha possível.

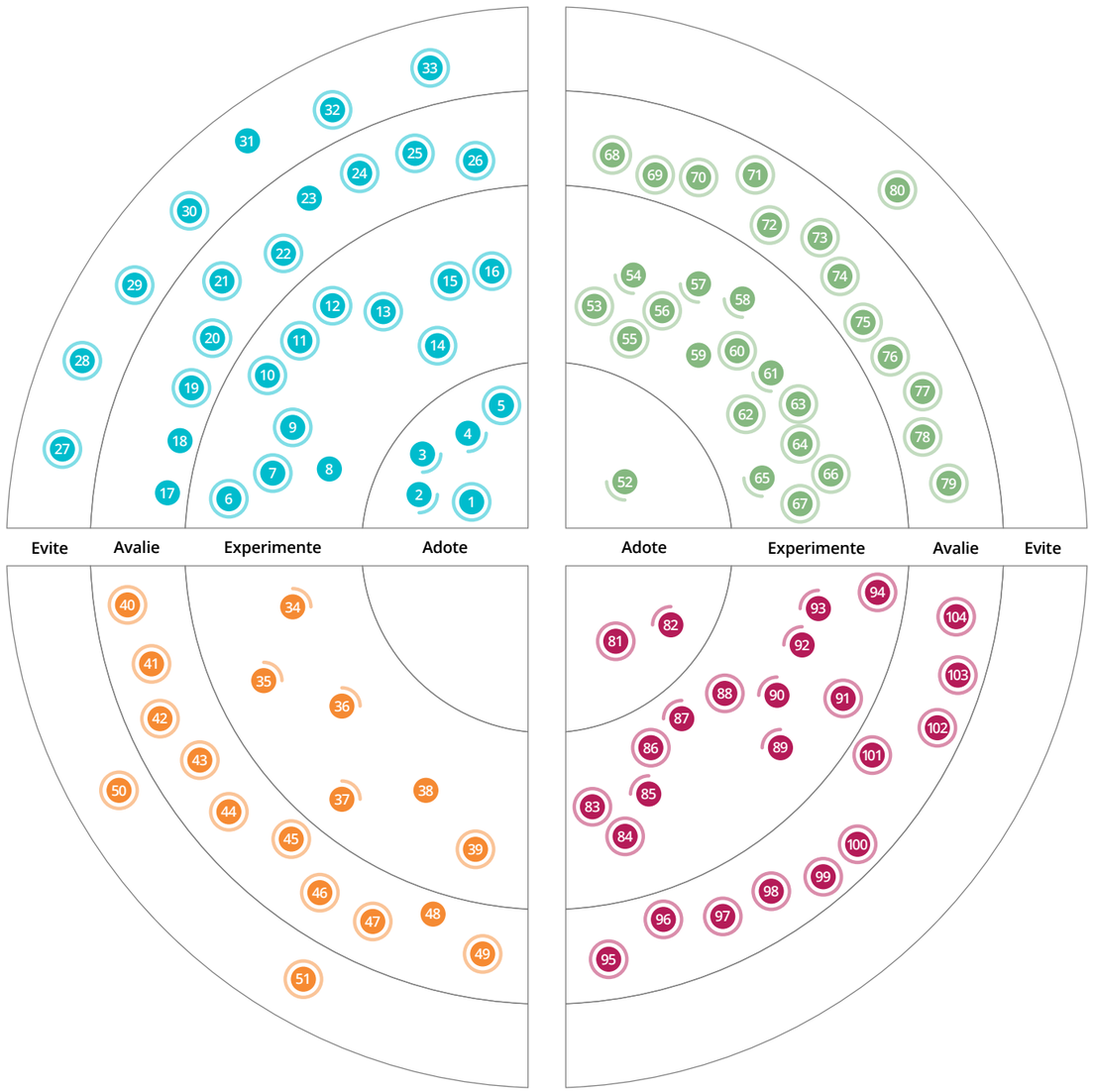
Permanentemente muito complexo para capturar em um blip

Na nomenclatura do Radar, o status final após a discussão de muitos tópicos complexos é "TCTB - too complex to blip" (muito complexo para capturar em um blip): são itens que desafiam a classificação, porque oferecem uma série de prós e contras, uma grande quantidade de nuances quanto à aplicabilidade da recomendação ou da ferramenta, ou outros motivos que nos impedem de resumir nossas opiniões em poucas frases. Frequentemente, esses tópicos acabam virando artigos, podcasts e seguindo outros destinos que não o Radar. Algumas de nossas conversas mais ricas concentram-se nesses tópicos: são importantes, mas complexas, impossibilitando um único ponto de vista sucinto. Vários desses tópicos são recorrentes, reunião após reunião — e, criticamente, em vários de nossos engajamentos com clientes —, eventualmente caindo na categoria TCTB, incluindo monorepos, diretrizes de orquestração para arquiteturas distribuídas e modelos de branching, entre outros. Para quem está se perguntando por que esses tópicos importantes não são incluídos no Radar, não é por falta de reconhecimento ou desejo de nossa parte. Como acontece com muitos tópicos no desenvolvimento de software, são muitas as concessões necessárias para permitir recomendações claras e sem ambiguidade. Às vezes, encontramos partes menores dentro dos tópicos maiores, sobre as quais podemos oferecer recomendações que são incluídas no Radar, mas os tópicos mais amplos se mantêm permanentemente repletos de nuances e complexidade para serem resumidos no Radar.

Compreendendo o contexto para acoplamento arquitetural

Um tópico recorrente em praticamente todas as reuniões (veja "Permanentemente muito complexo para capturar em um blip") é o nível apropriado de acoplamento na arquitetura de software entre microsserviços, componentes, gateways de API, hubs de integração, front-ends e assim por diante. Praticamente em todos os lugares em que dois pedaços de software podem se conectar, pessoas arquitetas e desenvolvedoras lutam para encontrar o nível correto de acoplamento — muitas recomendações comuns encorajam o desacoplamento extremo, mas isso dificulta a construção de fluxos de trabalho. O acoplamento em arquitetura de software traz muitas considerações importantes: como as coisas estão conectadas, a compreensão do acoplamento semântico inerente a cada domínio de problema, como as coisas chamam umas às outras ou como funciona a transacionalidade (às vezes em combinação com outras características arquiteturais complicadas como escalabilidade). O software não pode existir sem algum nível de acoplamento fora dos sistemas monolíticos únicos. Chegar ao conjunto certo de compromissos para determinar os tipos e níveis de acoplamento torna-se uma habilidade crítica para lidar com arquiteturas modernas. Observamos práticas inadequadas, como a geração de código para bibliotecas cliente, e boas práticas, como o uso criterioso dos padrões de BFF. No entanto, uma recomendação geral neste domínio é inútil, e balas de prata não existem. Invista tempo e esforço na compreensão dos fatores em jogo ao tomar essas decisões caso a caso, em vez de buscar uma solução que seja genérica, mas inadequada.

O Radar



Técnicas

Adote
1. Expansão e contração de API
2. Entrega contínua para aprendizado de máquina (CD4ML)
3. Sistemas de design
4. Times de produto de engenharia de plataforma
5. Abordagem de rotação de contas de serviços

Experimente
6. Sandboxes na nuvem
7. Contextual bandits
8. Imagens Docker sem distribuição
9. Ethical Explorer
10. Renovação de legado baseada em hipóteses
11. Abordagem simples para RFCs
12. ML mais simples possível
13. Injeção de SPA
14. Carga cognitiva do time
15. Xcodeproj gerenciado por ferramenta
16. Tipos compartilhados de UI/BFF

Avalie
17. Plataformas limitadas de baixo código
18. Identidade descentralizada
19. Radiador de desvio de implantação
20. Criptografia homomórfica
21. Hotwire
22. Import maps para micro frontends
23. Open Application Model (OAM)
24. Web analytics com foco em privacidade
25. Programação em grupo remota
26. Computação multipartidária segura

Evite
27. GitOps
28. Times de plataforma em camadas
29. Requisitos ingênuos de complexidade de senha
30. Revisão por pares significa pull request
31. SAFe™
32. Separação da propriedade do código e do pipeline
33. Modelos operacionais de plataforma orientados a tickets

Plataformas

Adote

Experimente
34. AWS Cloud Development Kit
35. Backstage
36. Delta Lake
37. Materialize
38. Snowflake
39. Fontes variáveis

Avalie
40. Apache Pinot
41. Bit.dev
42. DataHub
43. Feature Store
44. JuiceFS
45. API do Kafka sem Kafka
46. NATS
47. Opstrace
48. Pulumi
49. Redpanda

Evite
50. Azure Machine Learning
51. Produtos caseiros de infraestrutura como código (IaC)

Ferramentas

Adote
52. Sentry

Experimente
53. axe-core
54. dbt
55. esbuild
56. Flipper
57. Great Expectations
58. k6
59. MLflow
60. OR-Tools
61. Playwright
62. Prowler
63. Pyright
64. Redash
65. Terratest
66. Tuple
67. Why Did You Render

Avalie
68. Buildah e Podman
69. GitHub Actions
70. Graal Native Image
71. HashiCorp Boundary
72. imgcook
73. Longhorn
74. Operator Framework
75. Recomendador
76. Remote - WSL
77. Spectral
78. Yelp detect-secrets
79. Zally

Evite
80. AWS CodePipeline

Linguagens & Frameworks

Adote
81. Combine
82. LeakCanary

Experimente
83. Angular Testing Library
84. AWS Data Wrangler
85. Blazor
86. FastAPI
87. io-ts
88. Kotlin Flow
89. LitElement
90. Next.js
91. On-demand modules
92. Streamlit
93. SWR
94. TrustKit

Avalie
95. .NET 5
96. bUnit
97. Dagster
98. Flutter para Web
99. Jotai e Zustand
100. Kotlin Multiplatform Mobile
101. LVGL
102. React Hook Form
103. River
104. Webpack 5 Module Federation

Evite

TECHNOLOGY RADAR

Técnicas

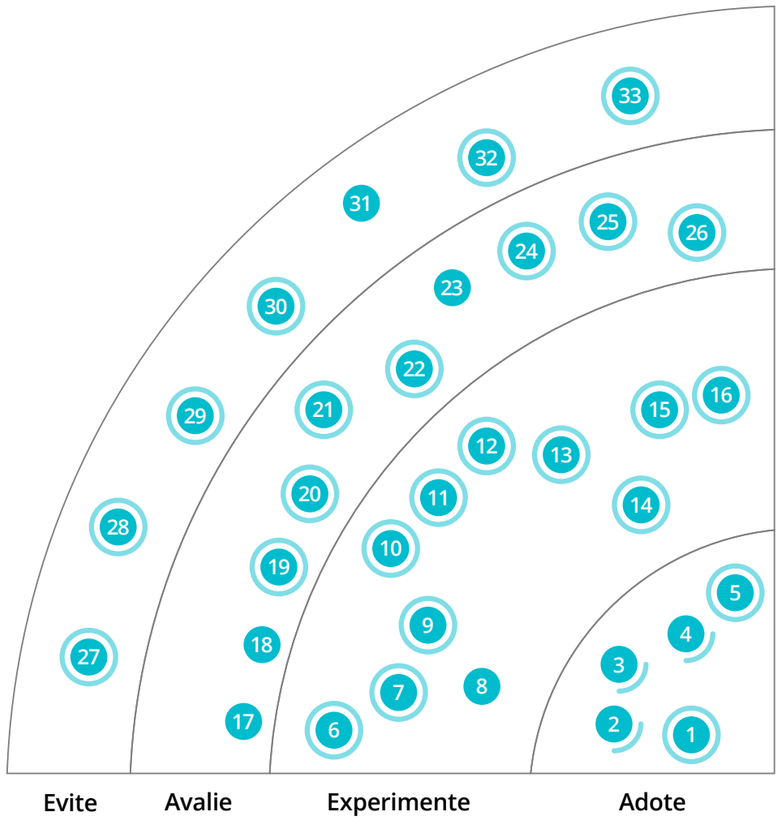
Técnicas

Expansão e contração de API
Adote

O padrão expansão e contração de API , também chamado de mudança paralela, é familiar para muitas pessoas, especialmente quando usado com bancos de dados ou código. No entanto, vemos baixa adoção quando se trata de APIs. Especificamente, temos visto esquemas de controle de versão complexos e mudanças significativas em cenários nos quais uma simples expansão, seguida por uma contração, seriam suficientes. Um exemplo seria adicionar primeiro a uma API enquanto um elemento existente é descontinuado e, em seguida, remover os elementos depreciados depois que a base de clientes for movida para o esquema mais recente. Essa abordagem requer alguma coordenação e visibilidade de clientes da API, talvez por meio de uma técnica como teste de contrato orientado a clientes.

Entrega contínua para aprendizado de máquina (CD4ML)
Adote

Vemos a entrega contínua para aprendizado de máquina (CD4ML) como um bom ponto de partida para qualquer solução de aprendizado de máquina que esteja sendo implantada em produção. Muitas organizações têm se tornado mais dependentes de soluções de aprendizado de máquina, tanto em ofertas para clientes quanto em operações internas. Portanto, faz sentido que os negócios apliquem as lições e boas práticas capturadas pela entrega contínua (CD) para soluções de aprendizado de máquina (ML).



Sistemas de design
Adote

À medida que o desenvolvimento de aplicações se torna cada vez mais dinâmico e complexo, é um desafio entregar produtos acessíveis e utilizáveis com estilo consistente. Isso vale particularmente para organizações maiores, com vários times trabalhando em produtos diferentes. Os sistemas de design definem uma coleção de padrões de design, bibliotecas de componentes e boas práticas de design e engenharia que garantem produtos digitais consistentes. Construídos com base nos guias de estilo corporativos do passado, os sistemas de design oferecem bibliotecas e documentos compartilhados que são fáceis de encontrar e usar. Geralmente, a orientação é escrita como código e mantida sob controle de versão para que o guia seja menos ambíguo e mais fácil de manter do que documentos simples. Os sistemas de design tornaram-se uma abordagem padrão para se trabalhar entre múltiplos times e disciplinas no desenvolvimento de produtos, porque permitem que os times se concentrem. Eles podem abordar desafios estratégicos em torno do próprio produto sem reinventar a roda toda vez que um novo componente visual for necessário.

Adote
1. Expansão e contração de API
2. Entrega contínua para aprendizado de máquina (CD4ML)
3. Sistemas de design
4. Times de produto de engenharia de plataforma
5. Abordagem de rotação de contas de serviços

Experimente
6. Sandboxes na nuvem
7. Contextual bandits
8. Imagens Docker sem distribuição
9. Ethical Explorer
10. Renovação de legado baseada em hipóteses
11. Abordagem simples para RFCs
12. ML mais simples possível
13. Injeção de SPA
14. Carga cognitiva do time
15. Xcodeproj gerenciado por ferramenta
16. Tipos compartilhados de UI/BFF

Avalie
17. Plataformas limitadas de baixo código
18. Identidade descentralizada
19. Radiador de desvio de implantação
20. Criptografia homomórfica
21. Hotwire
22. Import maps para micro frontends
23. Open Application Model (OAM)
24. Web analytics com foco em privacidade
25. Programação em grupo remota
26. Computação multipartidária segura

Evite
27. GitOps
28. Times de plataforma em camadas
29. Requisitos ingênuos de complexidade de senha
30. Revisão por pares significa pull request
31. SAFe™
32. Separação da propriedade do código e do pipeline
33. Modelos operacionais de plataforma orientados a tickets

Técnicas

Nomeado em referência às "bandits" (máquinas caça-níquel), o algoritmo investiga diferentes opções para aprender mais sobre os resultados esperados e equilibra tirando proveito das opções que funcionarem bem.

(Contextual bandits)

Times de produto de engenharia de plataforma
Adote

Conforme observado em um dos temas desta edição, a indústria vem ganhando cada vez mais experiência com times de produto de engenharia de plataforma , que criam e oferecem suporte a plataformas internas. Essas plataformas são usadas pelos times de uma organização e aceleram o desenvolvimento de aplicações, reduzem a complexidade operacional e melhoram o tempo de chegada ao mercado. Com o aumento da adoção, também ficam mais evidentes os padrões bons e ruins dessa abordagem. Ao criar uma plataforma, é fundamental definir clientes e produtos que serão beneficiados, em vez de construí-la no vácuo. Advertimos contra times de plataforma em camadas que simplesmente preservam silos de tecnologia existentes sob o rótulo de "times de plataforma", e também contra modelos operacionais de plataforma orientados a tickets. Ainda defendemos fortemente o uso de conceitos da abordagem de topologias de time quando pensamos sobre a melhor forma de organizar times de plataforma. Consideramos os times de produto de engenharia de plataforma uma abordagem padrão e um facilitador significativo para a TI de alta performance.

Abordagem de rotação de contas de serviços
Adote

Aconselhamos fortemente as organizações a certificarem-se — quando realmente for necessário usar contas de serviços em nuvem — de rotacionar as credenciais. Rotação é um dos três Rs da segurança. É muito comum que as organizações se esqueçam dessas contas até que um incidente ocorra. Como resultado, contas com permissões desnecessariamente amplas permanecem em uso por longos períodos, sem planejamento para substituí-las ou rotacioná-las. A aplicação regular da rotação de contas de serviços em nuvem também permite exercer o princípio do menor privilégio.

Sandboxes na nuvem
Experimente

À medida que a nuvem se torna cada vez mais uma commodity e a capacidade de criar sandboxes na nuvem se torna mais simples e disponível em escala, nossos times tendem a preferir ambientes de desenvolvimento somente na nuvem (em vez de locais) para reduzir a complexidade da manutenção. Temos observado que o conjunto de ferramentas para executar simulação local de serviços nativos da nuvem limita a confiança no build de desenvolvimento e nos ciclos de teste. Portanto, temos procurado nos concentrar na padronização de sandboxes de nuvem em vez de executar componentes nativos de nuvem em máquinas de pessoas desenvolvedoras. Isso impulsiona boas práticas de infraestrutura como código como função motriz e bons processos de integração para provisionar ambientes de sandbox. Existem riscos associados a essa transição, pois pressupõe-se que as pessoas desenvolvedoras terão dependência absoluta da disponibilidade do ambiente de nuvem, o que pode desacelerar o ciclo de feedback de desenvolvimento. Recomendamos enfaticamente a adoção de práticas de governança enxutas para padronização desses ambientes de sandbox, especialmente no que se refere à segurança, IAM e implantações regionais.

Contextual bandits
Experimente

Contextual bandits é um tipo de aprendizado por reforço, adequado para problemas que envolvem o dilema entre explorar investigando e explorar tirando proveito. Nomeado em referência às "bandits" — como são informalmente chamadas as máquinas caça-níquel, em inglês —, o algoritmo investiga diferentes opções para aprender mais sobre os resultados esperados e equilibra tirando proveito das opções que funcionarem bem. Usamos com sucesso essa técnica em cenários com poucos dados disponíveis para treinar e implantar outros modelos de ML. O fato de podermos adicionar contexto à relação investigar e tirar proveito torna a técnica adequada para uma ampla variedade de casos de uso, incluindo testes A/B, recomendações e otimizações de layout.

Imagens Docker sem distribuição
Experimente

Ao criar imagens do Docker para nossas aplicações, geralmente temos duas preocupações: a segurança e o tamanho da imagem. Tradicionalmente, usamos ferramentas de escaneamento de segurança de contêiner para detectar e corrigir vulnerabilidades e exposições comuns, e pequenas distribuições como Alpine Linux para tratar do tamanho da imagem e do desempenho da distribuição. Mas com o aumento das ameaças à segurança, eliminar todos os vetores de ataque possíveis se tornou mais importante do que nunca. É por isso que as imagens Docker sem distribuição estão se tornando a escolha padrão para contêineres de implantação. Imagens do Docker sem distribuição reduzem o rastro e as dependências ao eliminar uma distribuição completa do sistema operacional. Essa técnica reduz o ruído da verificação de segurança e a superfície de ataque da aplicação. São menos vulnerabilidades a serem corrigidas e, como bônus, essas imagens menores são mais eficientes. O Google publicou um conjunto de imagens de contêiner sem distribuição para diferentes linguagens. Você pode criar imagens de aplicações sem distribuição usando a ferramenta de construção do Google Bazel ou simplesmente usar Dockerfiles de vários estágios (multistage). Observe que os contêineres sem distribuição, por padrão, não têm um shell para depuração. No entanto, você pode encontrar facilmente versões de depuração de contêineres sem distribuição online, incluindo um shell BusyBox. O uso de imagens do Docker sem distribuição é uma técnica pioneira do Google e, em nossa experiência, ainda é muito restrita a imagens geradas pela empresa. Ficaríamos mais confortáveis se houvesse mais de uma

provedora. Além disso, cuidado ao aplicar Trivy ou scanners de vulnerabilidade semelhantes, uma vez que contêineres sem distribuição são suportados apenas em versões mais recentes.

Ethical Explorer
Experimente

O grupo por trás do Ethical OS — Omidyar Network, que se define como um empreendimento de transformação social, criado pelo fundador do eBay Pierre Omidyar — lançou um nova iteração chamada Ethical Explorer. O novo pacote Ethical Explorer baseia-se nas lições aprendidas com o uso do Ethical OS e adiciona outras questões para times de produto considerarem. O kit, que pode ser baixado gratuitamente e impresso em cartões para discussão, contém sugestões de perguntas abertas para várias "zonas de risco" , incluindo vigilância ("alguém pode usar nosso produto ou serviço para rastrear ou identificar outras pessoas usuárias?"), desinformação, exclusão, viés algorítmico, vício, controle de dados, agentes malfeitores e poder desproporcional. O guia incluído tem atividades e workshops, ideias para iniciar conversas e dicas para obter adesão organizacional. Embora tenhamos um longo caminho a percorrer como indústria para melhor representar as externalidades éticas de nossa sociedade digital, tivemos algumas conversas produtivas usando o Ethical Explorer e a ampliação da consciência sobre a importância das decisões de produto abordando questões sociais nos encoraja.

Renovação de legado baseada em hipóteses
Experimente

Frequentemente, recebemos pedidos para atualizar, modernizar ou remediar sistemas legados que não foram originalmente criados por nós. Às vezes, questões técnicas como melhorar o desempenho ou a confiabilidade exigem nossa atenção. Uma abordagem comum para resolver essas questões é criar "histórias técnicas" usando o mesmo formato de uma história de usuário, mas com um resultado técnico em vez de comercial. Entretanto, as tarefas técnicas costumam ser difíceis de estimar, demoram mais do que o previsto ou acabam não tendo o resultado desejado. Um método alternativo e melhor sucedido é a renovação de legado baseada em hipóteses. Em vez de caminhar em direção a um backlog padrão, o time assume a propriedade de um resultado técnico mensurável e estabelece coletivamente um conjunto de hipóteses sobre o problema. Em seguida, o time conduz experimentos iterativos, com tempo pré-definido, para verificar ou refutar cada hipótese seguindo a ordem de prioridade. O fluxo de trabalho resultante é otimizado para reduzir a incerteza em vez de seguir um plano em direção a um resultado previsível.

Abordagem simples para RFCs
Experimente

À medida que as organizações avançam em direção à arquitetura evolutiva, é importante capturar decisões sobre design, arquitetura, técnicas e formas de trabalho dos times. O processo de coleta e agrupamento de feedbacks que levará a essas decisões começa com os Request for Comments (RFCs). RFCs são uma técnica para coletar ideias de contexto, design e arquitetura e colaborar com os times para tomar decisões considerando contexto e consequências. Recomendamos que as organizações adotem uma abordagem simples, usando um modelo padronizado para os times e controle de versão para capturar RFCs.

É importante coletá-los em uma auditoria dessas decisões, para beneficiar futuros membros dos times e para capturar a evolução técnica e comercial da organização. Organizações maduras têm usado RFCs em times autônomos para estimular uma melhor comunicação e uma colaboração mais eficiente, especialmente quando se trata de decisões relevantes para times multidisciplinares.

ML mais simples possível
Experimente

Todas as principais provedoras de nuvem oferecem uma gama impressionante de soluções de aprendizado de máquina (ML). Essas ferramentas poderosas podem fornecer muito valor, mas têm um custo. Existe o custo puramente de execução cobrado pelas provedoras de nuvem para esses serviços. Além disso, existe uma espécie de imposto operacional. Essas ferramentas complexas precisam ser compreendidas e operadas e, a cada nova ferramenta adicionada à arquitetura, essa carga tributária aumenta. Em nossa experiência, os times costumam escolher ferramentas complexas porque subestimam o poder de ferramentas mais simples, como a regressão linear. Muitos problemas de ML não exigem uma GPU ou redes neurais. Por esse motivo, defendemos o ML mais simples possível , usando ferramentas e modelos simples e algumas centenas de linhas de Python na plataforma de computação que você tem em mãos. Busque ferramentas complexas apenas quando puder demonstrar a necessidade delas.

Injeção de SPA
Experimente

O padrão da figueira estranguladora costuma ser a estratégia padrão para a modernização de legados, com o novo código envolvendo o antigo e absorvendo lentamente a capacidade de lidar com todas as funcionalidades necessárias. Esse tipo de abordagem "de fora para dentro" funciona bem para vários sistemas legados, mas agora que temos experiência suficiente com aplicações de página única (SPA), de forma que elas próprias se tornem sistemas legados, temos visto a abordagem oposta, "de dentro para fora", usada para substituí-las. Em vez de envolver o sistema legado, incorporamos desde o início a nova SPA ao documento HTML que contém a antiga, e deixamos que ela expanda lentamente em funcionalidade. Os frameworks de SPA nem precisam ser os mesmos, desde que os usuários possam tolerar o impacto no desempenho

Técnicas

Quando se trata de modernização de aplicações de página única (SPAs), em vez de envolver o sistema legado, incorporamos a nova SPA ao documento HTML que contém a antiga, e deixamos ela expandir lentamente sua funcionalidade.

(Injeção de SPA)

Técnicas

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. O autor do livro Topologias de Time desenvolveu uma avaliação para medir essas interações que chamamos de carga cognitiva do time.

(Carga cognitiva do time)

provocado pelo tamanho aumentado da página (por exemplo, incorporando um novo aplicativo React dentro de um antigo AngularJS. A injeção de SPA permite remover iterativamente a antiga SPA até que a nova assuma completamente. Considerando que uma figueira estranguladora pode ser vista como um tipo de parasita que usa a superfície externa estável da árvore hospedeira para se sustentar até que se enraíze e a própria hospedeira morra, esta abordagem seria como injetar um agente externo no hospedeiro, contando com a funcionalidade da SPA original até que esta possa assumir completamente.

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.

Xcodeproj gerenciado por ferramenta
Experimente

Muitas de nossas pessoas desenvolvedoras que programam iOS no Xcode costumam ter dores de cabeça porque o arquivo Xcodeproj muda a cada mudança no projeto. O formato de arquivo Xcodeproj não é legível por seres humanos, portanto, tentar lidar com conflitos de merge é bastante complicado e pode levar à perda de produtividade e ao risco de atrapalhar todo o projeto — se algo der errado com o arquivo, o Xcode não funcionará corretamente e as pessoas desenvolvedoras muito provavelmente ficarão bloqueadas. Em vez de tentar fazer merge e corrigir manualmente ou versionar o arquivo, recomendamos que você use uma abordagem de Xcodeproj gerenciado por ferramenta: defina a configuração do projeto Xcode em YAML (XcodeGen, Struct), Ruby (Xcake) ou Swift (Tuist). Essas ferramentas geram o arquivo Xcodeproj com base em um arquivo de configuração e na estrutura do projeto. Como resultado, os conflitos de merge no arquivo Xcodeproj ficarão no passado e, ainda que eventualmente aconteçam no arquivo de configuração, serão muito mais simples de resolver.

Tipos compartilhados de UI/BFF
Experimente

Com TypeScript se tornando uma linguagem comum para desenvolvimento de front-end e Node.js se tornando a tecnologia preferida para BFF, estamos observando um aumento no uso de tipos compartilhados de UI/BFF. Nesta técnica, um único conjunto de definições de tipo é usado para definir tanto os objetos de dados retornados por consultas de front-end quanto os dados servidos para satisfazer essas consultas pelo servidor de back-end. Normalmente, abordaríamos essa prática com cautela devido ao acoplamento desnecessariamente forte criado entre os limites do processo. No entanto, muitos times consideram que os benefícios dessa abordagem superam os riscos de um acoplamento forte. Uma vez que o padrão BFF funciona melhor quando o mesmo time é dono do código da UI e do BFF, geralmente armazenando ambos os componentes no mesmo repositório, o par UI/BFF pode ser visto como um único sistema coeso. Quando o BFF oferece consultas fortemente tipadas, os resultados podem ser ajustados às necessidades específicas do front-end, em vez de reutilizar uma única entidade de uso geral que deve atender às necessidades de vários consumidores e conter mais campos do que necessário. Isso reduz o risco de expor acidentalmente dados que o usuário não deveria ver, evita a interpretação incorreta do objeto de dados retornado e torna a consulta mais expressiva. Essa prática é bastante útil quando implementada com io-ts para aplicar a segurança de tipo em tempo de execução.

Plataformas limitadas de baixo código
Avalie

Uma das decisões mais complexas que as empresas enfrentam no momento é a adoção de plataformas de baixo código ou sem código, ou seja, plataformas que resolvem problemas muito específicos em domínios muito limitados. Muitas fornecedoras estão ocupando agressivamente este espaço. Os problemas que vemos com essas plataformas normalmente estão relacionados à incapacidade de aplicar boas práticas de engenharia, como controle de versão. Testar também é normalmente muito difícil. No entanto, notamos algumas novas participantes interessantes no mercado — incluindo Amazon Honeycode, que facilita a criação de aplicações simples de gerenciamento de tarefas ou eventos, e Parabola, para fluxos de trabalho em nuvem do tipo IFTTT. Por esse motivo, estamos incluindo novamente as plataformas limitadas de baixo código nesta edição. No entanto, continuamos com dúvidas sobre sua aplicabilidade mais ampla, uma vez que essas ferramentas, assim como algumas espécies de plantas, têm por característica avançar para além de seus limites e se emaranhar com outras. Por isso, ainda recomendamos cautela em sua adoção.

Identidade descentralizada
Avalie

Em 2016, Christopher Allen, um importante contribuidor no espaço de SSL/TLS, nos inspirou com a introdução de 10 princípios sustentando uma nova forma de identidade digital e um caminho para chegar lá, o caminho para a identidade auto-soberana. A identidade auto-soberana, também conhecida como identidade descentralizada , é uma "identidade portátil

vitalícia para qualquer pessoa, organização ou coisa que não dependa de nenhuma autoridade centralizada e jamais possa ser removida", de acordo com o padrão Trust over IP. A aplicação da identidade descentralizada vem crscendo e se tornando mais possível. Vemos sua adoção em aplicações de saúde para clientes, infraestruturas de saúde governamentais e identidades jurídicas corporativas. Se você deseja adotar rapidamente a identidade descentralizada, pode avaliar sistemas de suporte como Sovrin Network, Hyperledger Aries e Indy, bem como padrões para identificadores descentralizados e credenciais verificáveis. Estamos observando este espaço de perto, enquanto ajudamos nossa base de clientes com seus posicionamentos estratégicos na nova era de confiança digital.

Radiador de desvio de implantação

Avalie

Um radiador de desvio de implantação torna o desvio de versão visível para um software implantado em vários ambientes. As organizações que usam implantações automatizadas podem exigir aprovações manuais para ambientes próximos à produção, o que significa que o código nesses ambientes pode estar atrasado em várias versões em relação à versão em desenvolvimento. Essa técnica torna esse atraso visível por meio de um painel simples que mostra o quão desatualizado está cada componente implantado em cada ambiente. Isso ajuda a destacar o custo de oportunidade de um software concluído que ainda não esteja em produção, ao mesmo tempo chamando atenção para riscos, como correções de segurança ainda não implantadas.

Criptografia homomórfica

Avalie

A criptografia homomórfica refere-se a uma classe de métodos de criptografia que permitem que cálculos (como pesquisa e aritmética) sejam realizados diretamente em dados criptografados. O resultado de tais cálculos permanece na forma criptografada, que posteriormente pode ser descriptografada e revelada. Embora o problema da criptografia homomórfica tenha sido proposto pela primeira vez em 1978, uma solução não havia sido construída até 2009. Com avanços no poder da computação e a disponibilidade de bibliotecas de código aberto fáceis de usar — incluindo SEAL, Lattigo, HElib e criptografia parcialmente homomórfica em Python — a criptografia homomórfica está se tornando viável em aplicações do mundo real. Os cenários mais animadores incluem casos de uso de preservação de privacidade, em que a computação pode ser terceirizada para uma parte não confiável, por exemplo, executando computação em dados criptografados na nuvem ou permitindo que uma terceira parte agregue resultados de aprendizado de máquina federado intermediário criptografado homomorficamente. Além disso, a maioria dos esquemas de criptografia homomórfica são considerados seguros contra computadores quânticos, e esforços estão em andamento para padronizar a criptografia homomórfica. Apesar de suas limitações atuais, especificamente, desempenho e viabilidade dos tipos de cálculos, a criptografia homomórfica merece atenção.

Hotwire

Avalie

Hotwire (HTML over the wire) é uma técnica para construir aplicações web. As páginas são construídas a partir de componentes, mas ao contrário das aplicações de página única (SPAs) modernas, o HTML dos componentes é gerado no lado do servidor e, em seguida, enviado pela rede ("over the wire") para o navegador. A aplicação possui apenas uma pequena quantidade de código JavaScript no navegador para juntar os fragmentos HTML. Nossos times, e sem dúvida outros times também, experimentaram essa técnica depois que as solicitações assíncronas web ganharam suporte para vários navegadores por volta de 2005, mas por várias razões, ela nunca ganhou muita força. Hoje, a abordagem Hotwire usa um navegador web moderno e recursos HTTP para atingir a velocidade, a capacidade de resposta e a natureza dinâmica de SPAs. Ela adota um design de aplicação web mais simples, localizando a lógica no servidor e mantendo o código do lado do cliente simples. A equipe da Basecamp lançou alguns frameworks Hotwire que alimentam sua própria aplicação, incluindo Turbo e Stimulus. O Turbo inclui um conjunto de técnicas e frameworks para acelerar a capacidade de resposta da aplicação, evitando o recarregamento da página inteira, visualização da página do cache e decomposição da página em fragmentos com aprimoramentos progressivos mediante solicitação. O Stimulus foi projetado para aprimorar o HTML estático no navegador, conectando objetos JavaScript aos elementos da página no HTML.

Import maps para micro frontends

Avalie

Ao compor uma aplicação a partir de vários micro frontends, alguma parte do sistema precisa decidir quais micro frontends carregar e de onde carregá-los. Até o momento, nós criávamos soluções customizadas ou usávamos um framework mais abrangente, como o single-spa. Agora, existe um novo padrão, import maps, que ajuda em ambos os casos. Nossas primeiras experiências mostram que import maps para micro frontends permite uma separação organizada de responsabilidades. O código JavaScript indica o que importar e uma pequena tag de script na resposta HTML inicial especifica de onde carregar os frontends. Esse HTML é obviamente gerado no lado do servidor, o que torna possível usar alguma configuração dinâmica durante sua renderização. De muitas maneiras, essa técnica nos lembra dos caminhos de vinculadores/carregadores para bibliotecas Unix dinâmicas. No momento, import maps são compatíveis apenas com o Chrome, mas com o polyfill SystemJS, eles podem ser usados de forma mais ampla.

Técnicas

Organizações que usam implantações automatizadas podem exigir aprovações manuais para ambientes próximos à produção, provocando atrasos no código em relação à versão em desenvolvimento. Um radiador de desvio de implantação torna esse atraso visível por meio de um painel simples.

(Radiador de desvio de implantação)

Técnicas

A computação multipartidária segura resolve o problema da computação colaborativa, protegendo a privacidade entre partes que não confiam umas nas outras sem envolver uma terceira.

(Computação multipartidária segura)

Open Application Model (OAM)
Avalie

O Open Application Model (OAM) é uma tentativa de padronização para o espaço de modelagem de plataformas de infraestrutura como produtos. Usando abstrações de componentes, configurações de aplicações, escopos e características, as pessoas desenvolvedoras podem descrever aplicações de forma agnóstica de plataforma, enquanto quem implementa a plataforma pode defini-la em termos de carga de trabalho, característica e escopo. Desde a última vez que falamos sobre o OAM, acompanhamos de perto uma de suas primeiras implementações, a KubeVela. A KubeVela está perto de lançar a versão 1.0, e temos curiosidade em saber se implementações como essa podem justificar a promessa por trás da ideia do OAM.

Web analytics com foco em privacidade
Avalie

Web analytics com foco em privacidade é uma técnica para coletar web analytics sem comprometer a privacidade do usuário final, mantendo-o verdadeiramente anônimo. Uma consequência espantosa da conformidade com o Regulamento Geral de Proteção de Dados (GDPR) é a decisão tomada por muitas organizações de degradar a experiência de uso com processos complexos de consentimento de cookies, especialmente quando o usuário não consente imediatamente à configuraçãpadrão de "todos os cookies". Web analytics com foco em privacidade tem o benefício duplo de observar o espírito e o texto do GDPR, ao mesmo tempo evitando a necessidade de introduzir formulários intrusivos de consentimento. Uma implementação dessa abordagem pode ser vista no Plausible.

Programação em grupo remota
Avalie

A programação em grupo é uma técnica que nossos times consideram mais fácil de ser executada remotamente. A programação em grupo remota permite que os times 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 reservar uma sala de reunião ou usar um quadro branco.

Computação multipartidária segura
Avalie

A computação multipartidária segura (MPC) resolve o problema da computação colaborativa protegendo a privacidade entre partes que não confiam umas nas outras. Seu objetivo é calcular com segurança um problema acordado sem uma terceira parte confiável, com cada parte obrigatoriamente participando do resultado do cálculo, sem que ele possa ser obtido por outras entidades. Uma ilustração simples para a MPC é o problema dos milionários, no qual duas pessoas querem entender qual delas é a mais rica, mas nenhuma delas quer compartilhar seu patrimônio líquido real com a outra, nem confiar em uma terceira parte. As abordagens de implementação da MPC variam: os cenários podem incluir compartilhamento de segredos, transferência inconsciente, circuitos truncados ou criptografia homomórfica. Algumas soluções comerciais de MPC que surgiram recentemente (por exemplo, Antchain Morse) afirmam ajudar a resolver os problemas de compartilhamento de segredos e aprendizado de máquina seguro em cenários como investigação multipartidária de crédito conjunto e intercâmbio de dados de registros médicos. Embora essas plataformas sejam atraentes do ponto de vista de marketing, ainda precisamos entender se são de fato úteis.

GitOps
Evite

Sugerimos certo cuidado ao abordar GitOps, especialmente em relação às estratégias de branching. GitOps pode ser visto como uma forma de implementar infraestrutura como código que envolve sincronização contínua e aplicação de código de infraestrutura Git em vários ambientes. Quando usado com uma infraestrutura de "branch por ambiente", as alterações são promovidas de um ambiente para o outro por meio de merges. Embora tratar o código como a única fonte de verdade seja uma abordagem correta, vemos branches por ambiente provocarem desvios de ambientes e, eventualmente, merges de configuração como código específicos do ambiente tornando-se problemáticos ou parando completamente. Isso é muito semelhante ao que vimos no passado com branches de longa duração com GitFlow.

Times de plataforma em camadas
Evite

A explosão de interesse em torno das plataformas de software gerou muito valor para as organizações, mas o caminho para a construção de um modelo de entrega baseado em plataforma está repleto de potenciais becos sem saída. É comum em meio ao entusiasmo com novos paradigmas ver técnicas mais antigas ressurgindo repaginadas para incorporar o novo vernáculo, contribuindo para perdermos de vista as razões pelas quais superamos essas técnicas em primeiro lugar. Para ver um exemplo dessa repaginação, veja nosso blip sobre os tradicionais ESBs disfarçados de gateways de API na edição anterior do Radar. Outro exemplo que estamos vendo é a reformulação da abordagem de divisão de times por camada de tecnologia, mas chamando-os de times de plataformas. No contexto de construção de uma aplicação, costumava ser comum ter um time de front-end separado do time de lógica de negócio, por sua vez separado do time de dados. Vemos analogias a esse modelo quando as organizações segregam os recursos da plataforma entre times dedicados a um negócio ou camada de dados. Graças à Lei de Conway, sabemos que a organização de times de recursos de plataforma em torno de recursos de negócios é um modelo mais eficaz, dando aos times a propriedade de ponta a ponta do recurso,

incluindo a propriedade dos dados. Isso ajuda a evitar as dores de cabeça com gerenciamento de dependências de times de plataforma em camadas , com o time de front-end aguardando o time de lógica de negócio aguardando o time de dados para conseguir concluir qualquer coisa.

Requisitos ingênuos de complexidade de senha

Evite

As políticas de senha são comuns em muitas organizações hoje. No entanto, ainda vemos organizações exigindo senhas que incluam uma variedade de símbolos, números, letras maiúsculas e minúsculas, além de caracteres especiais. Esses são requisitos ingênuos de complexidade de senha , que levam a uma falsa sensação de segurança, pois as pessoas optam por senhas mais inseguras porque a alternativa é difícil de lembrar e digitar. De acordo com as recomendações do NIST (National Institute of Standards and Technology), o principal fator na força da senha é o tamanho. Portanto, as pessoas devem escolher senhas compostas por frases longas com um requisito máximo de 64 caracteres (incluindo espaços). Essas senhas são mais seguras e fáceis de memorizar.

Revisão por pares significa pull request

Evite

Algumas organizações parecem acreditar que a revisão por pares significa pull request. Elas consideram que a única forma de se obter uma revisão do código por pares é um pull request. Vimos essa abordagem criar gargalos significativos, além de degradar significativamente a qualidade do feedback à medida que pessoas revisoras sobrecarregadas passam a rejeitar as solicitações. Embora possa-se argumentar que 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 tinha sido lido por alguém antes da aceitação. Pull requests são apenas uma forma de gerenciar o fluxo de revisão. Recomendamos considerar outras abordagens, especialmente quando houver necessidade de ensinar e dar feedback.

SAFe™

Evite

Nosso posicionamento sobre "ser ágil antes de fazer ágil" e nossas opiniões sobre o tema não devem ser uma surpresa. Mas já que o SAFe™ (Scaled Agile Framework®), segundo relatório da Gartner de maio de 2019, é o framework ágil corporativo mais considerado e usado e, como vemos cada vez mais empresas passando por mudanças organizacionais, achamos que era hora de ampliar a conscientização sobre esse assunto novamente. Nós nos deparamos com organizações tendo dificuldades com os processos super padronizados e em fases super controladas do SAFe. Esses processos criam atritos na estrutura organizacional e no modelo operacional. O framework também pode promover silos na organização, impedindo que as plataformas se tornem de fato habilitadoras de recursos de negócios. O controle top-down gera desperdício no fluxo de valor e desestimula a criatividade dos talentos de engenharia, limitando a autonomia e a experimentação nos times. Em vez de medir esforço e se concentrar em cerimônias padronizadas, recomendamos uma abordagem e uma governança mais enxutas e orientadas a valor — como EDGE —, além de uma avaliação de carga cognitiva do time, para identificar os tipos de time e determinar como podem interagir melhor.

Scaled Agile Framework® e SAFe™ são marcas registradas da Scaled Agile, Inc.

Separação da propriedade do código e do pipeline

Evite

O ideal, especialmente quando os times praticam DevOps, é que o pipeline de implantação e o código pertençam ao mesmo time. Infelizmente, ainda vemos separação entre propriedade do código e do pipeline em algumas organizações, com a configuração do pipeline de implantação ficando sob responsabilidade do time de infraestrutura. Essa prática resulta em atrasos, barreiras para melhorias, além de falta de sendo de propriedade e envolvimento do time de desenvolvimento nas implantações. Uma das causas disso pode ser a separação do time. Outra pode ser o desejo de manter processos e funções de "gatekeeper". Embora possa haver razões legítimas para usar essa abordagem (por exemplo, controle regulatório), de forma geral, a consideramos trabalhosa e inútil.

Modelos operacionais de plataforma orientados a tickets

Evite

Um dos objetivos principais de uma plataforma deve ser reduzir processos baseados em tickets a um mínimo absoluto, pois eles criam filas no fluxo de valor. Infelizmente, vemos que as organizações ainda não insistem o suficiente nesse objetivo, resultando em um modelo operacional de plataforma orientado a tickets. Isso é ainda mais frustrante quando os processos baseados em tickets são colocados à frente de plataformas construídas em cima de recursos de autoatendimento orientados a API de fornecedoras de nuvem pública. Um autoatendimento com poucos tickets no início é difícil e desnecessário, mas deve ser o objetivo.

O excesso de dependência na burocracia e a falta de confiança estão entre as causas da relutância em abandonar processos baseados em tickets. Implementar verificações e alertas automatizados é uma forma de romper a dependência nos processos com tickets, por exemplo, dando aos times visibilidade sobre seus custos de execução e colocando barreiras automatizadas para evitar explosões de custo acidentais. Implemente uma política de segurança como código e use scanners de configuração ou analisadores como Recommender para ajudar os times a fazer a coisa certa.

Técnicas

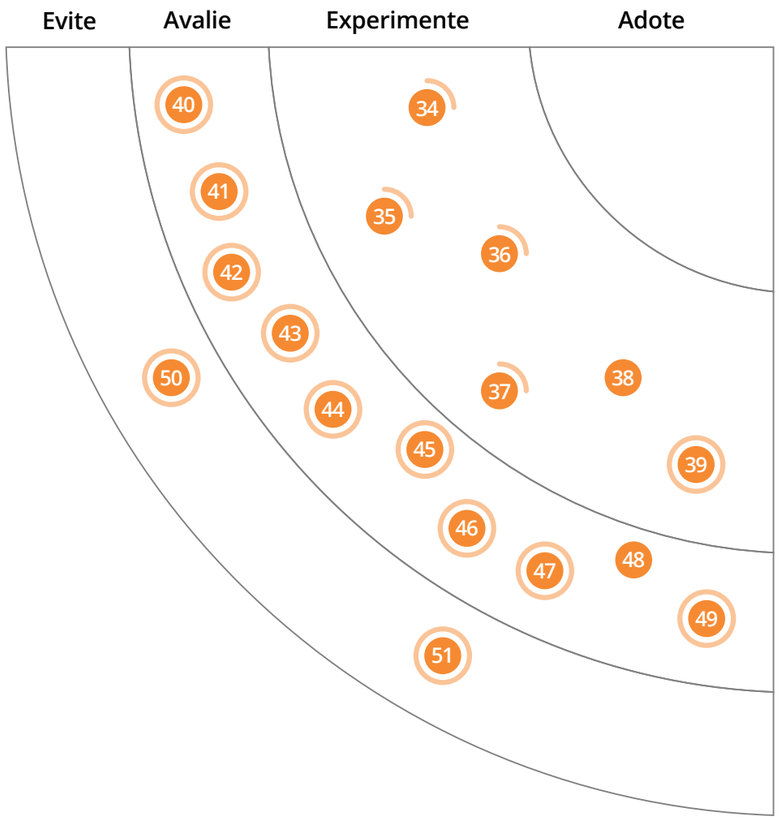
Algumas organizações parecem acreditar que a revisão por pares significa pull resuest. Vimos essa abordagem criar gargalos significativos, além de degradar a qualidade do feedback.

(Revisão por pares significa pull request)

TECHNOLOGY RADAR

Plataformas

Plataformas



AWS Cloud Development Kit
Experimente

Muitos de nossos times que já usam AWS descobriram que o AWS Cloud Development Kit (AWS CDK) é um padrão sensato da AWS para habilitar o provisionamento de infraestrutura. Particularmente, eles gostam do uso de linguagens de programação de primeira classe em vez de arquivos de configuração, o que permite aos times usar as ferramentas existentes, abordagens e habilidades de teste. Como acontece com ferramentas semelhantes, ainda é necessário cuidado para garantir que as implantações permaneçam fáceis de entender e manter. O kit atualmente suporta TypeScript, JavaScript, Python, Java, C # e .NET. Novos provedores estão sendo adicionados ao núcleo do CDK. Também usamos o AWS Cloud Development Kit e o HashiCorp's Cloud Development Kit for Terraform para gerar configurações do Terraform e habilitar o provisionamento usando a plataforma Terraform com sucesso.

Backstage
Experimente

Continuamos observando aumento no interesse e no uso de Backstage e na adoção de portais de desenvolvimento, à medida que as organizações buscam apoiar e otimizar seus ambientes de desenvolvimento. Conforme o número de ferramentas e de tecnologias aumenta, alguma forma de padronização se torna cada vez mais importante para consistência e para que as pessoas desenvolvedoras possam se concentrar na inovação e no desenvolvimento de produtos, em vez de terem que reinventar a roda. Backstage é uma plataforma de código aberto do Spotify para a criação de portais de desenvolvimento. É baseada em modelos de software, ferramentas de infraestrutura unificadas e documentação técnica consistente e centralizada. Sua arquitetura de plugins permite extensibilidade e adaptabilidade no ecossistema de infraestrutura de uma organização.

Delta Lake
Experimente

Delta Lake é uma camada de armazenamento de código aberto implementada pelo Databricks, que tenta levar transações ACID para o processamento de big data. Em nossos projetos de lago de dados ou malha de dados habilitados pelo Databricks, nossos times continuam preferindo usar o armazenamento Delta Lake em vez do uso direto de tipos de armazenamento de arquivos, como S3 ou ADLS. Claro, isso se limita a projetos que usam plataformas de armazenamento que suportam Delta Lake ao usar formatos de arquivo Parquet. O Delta Lake facilita os casos de uso simultâneos de leitura/gravação de dados em que a transacionalidade no nível de arquivo é necessária. Achamos a impecável integração do Delta Lake com a API de processamento em lote e microlote do Apache Spark muito úteis, principalmente recursos como versionamento — que possibilita acessar dados em um determinado momento ou reverter um commit — bem como suporte a evolução de esquemas, embora existam algumas limitações nessas funcionalidades.

Adote

Experimente
34. AWS Cloud Development Kit
35. Backstage
36. Delta Lake
37. Materialize
38. Snowflake
39. Fontes variáveis

Avalie
40. Apache Pinot
41. Bit.dev
42. DataHub
43. Feature Store
44. JuiceFS
45. API do Kafka sem Kafka
46. NATS
47. Opstrace
48. Pulumi
49. Redpanda

Evite
50. Azure Machine Learning
51. Produtos caseiros de infraestrutura como código (IaC)

Plataformas

O LinkedIn evoluiu o WhereHows para DataHub, uma geração seguinte da plataforma que lida com a detecção de dados por meio de um sistema de metadados extensível.

(DataHub)

Materialize
Experimente

Materialize é um banco de dados de streaming que permite fazer computação incremental sem pipelines de dados complicados. Basta descrever seus cálculos por meio de visualizações SQL padrão e conectar o Materialize ao fluxo de dados. A ferramenta subjacente de fluxo de dados diferencial executa computação incremental para fornecer saída consistente e correta com latência mínima. Ao contrário dos bancos de dados tradicionais, não há restrições na definição dessas visualizações materializadas e os cálculos são executados em tempo real. Usamos Materialize, junto com Spring Cloud Stream e Kafka, para consultar fluxos de eventos em busca de insights em sistemas distribuídos orientados a eventos, e gostamos bastante da configuração.

Snowflake
Experimente

Desde que mencionamos Snowflake pela última vez no Radar, ganhamos mais experiência com a plataforma, bem como com a malha de dados como uma alternativa para warehouses e lagos de dados. Snowflake continua a impressionar com recursos como viagem no tempo, clonagem de cópia zero, compartilhamento de dados e com o seu marketplace. Não encontramos nada de que não tenhamos gostado nele, o que fez nossas pessoas consultoras, de forma geral, o preferirem às alternativas. O Redshift está se movendo em direção à separação entre armazenamento e computação, o que tem sido um ponto forte do Snowflake, mas ainda assim o Redshift Spectrum não é tão conveniente e flexível, em parte pela limitação imposta pela herança do Postgres (a propósito, ainda gostamos do Postgres). As consultas federadas podem ser um motivo para usar o Redshift. Quando se trata de operação, o Snowflake é muito mais simples de executar. BigQuery, que é outra alternativa, é muito fácil de operar, mas em uma configuração com várias nuvens, o Snowflake é uma escolha melhor. Também podemos dizer que usamos o Snowflake com sucesso com GCP, AWS e Azure.

Fontes variáveis
Experimente

Fontes variáveis são uma maneira de evitar a necessidade de encontrar e incluir arquivos de fonte separados para diferentes pesos e estilos. Tudo fica em um arquivo de fonte e você pode usar as propriedades para selecionar o estilo e o peso necessários. Embora não seja algo novo, ainda vemos sites e projetos que poderiam se beneficiar dessa abordagem simples. Se você tiver páginas que incluem diferentes variações da mesma fonte, sugerimos experimentar a abordagem de fontes variáveis.

Apache Pinot
Avalie

Apache Pinot é um armazenamento de dados OLAP distribuído, construído para fornecer análises em tempo real com baixa latência. Ele pode ingerir de fontes de dados em lote (como Hadoop HDFS, Amazon S3, Azure ADLS ou Google Cloud Storage), bem como fontes de dados de stream (como Apache Kafka). Se a necessidade for uma análise de baixa latência voltada para o usuário, as soluções SQL-on-Hadoop não oferecem a baixa latência necessária. Mecanismos OLAP modernos como Apache Pinot (ou Apache Druid e Clickhouse, entre outros) podem atingir latência muito menor e são particularmente adequados em contextos em que análises rápidas, como agregações, são necessárias em dados imutáveis, possivelmente, com ingestão de dados em tempo real. Construído originalmente pelo LinkedIn, o Apache Pinot entrou na incubação da Apache no final de 2018 e, desde então, adicionou uma arquitetura de plugin e suporte a SQL, entre outros recursos importantes. O Apache Pinot pode ser bastante complexo de operar e tem muitas partes móveis, mas se seus volumes de dados forem grandes o suficiente e você precisar de capacidade de consulta de baixa latência, recomendamos que avalie o Apache Pinot..

Bit.dev
Avalie

Bit.dev é uma plataforma colaborativa hospedada em nuvem para componentes de UI extraídos, modularizados e reutilizados com Bit. Web Components já existem há algum tempo, mas construir uma aplicação de front-end moderna juntando pequenos componentes independentes extraídos de outros projetos nunca foi fácil. É aqui que o Bit é útil. É uma ferramenta de linha de comando elegante que permite extrair um componente de uma biblioteca ou de um projeto existente. Você pode construir seu próprio serviço em cima do Bit para colaboração de componentes, ou usar o Bit.dev.

DataHub
Avalie

Desde que mencionamos pela primeira vez a detecção de dados no Radar, o LinkedIn evoluiu o WhereHows para DataHub, uma geração seguinte da plataforma que lida com a detecção de dados por meio de um sistema de metadados extensível. Em vez de rastrear e extrair metadados, o DataHub adota um modelo baseado em push, no qual componentes individuais do ecossistema de dados publicam metadados por meio de uma API ou de um stream para a plataforma central. Essa integração baseada em push transfere a propriedade da entidade central para times individuais, tornando-os responsáveis por seus metadados. À medida que mais empresas tentam se tornar orientadas por dados, ter um sistema que ajuda na descoberta de dados e no entendimento da qualidade e da linhagem dos dados é fundamental, e recomendamos que você avalie essa capacidade no DataHub.

Feature Store
Avalie

Feature Store é uma plataforma de dados específica de aprendizado de máquina que aborda alguns dos principais desafios enfrentados hoje pela engenharia de recursos, por meio de três capacidades fundamentais: (1) uso de pipelines de dados gerenciados para remover dificuldades de pipelines conforme novos dados chegam; (2) catalogação e armazenamento de dados de recursos para promover descoberta e colaboração de recursos entre modelos; (3) veiculação de dados de recursos de maneira consistente durante o treinamento e a interferência.

Desde que o Uber revelou sua plataforma Michelangelo, muitas organizações e startups criaram suas próprias versões de lojas de recursos. Exemplos incluem Hopsworks, Feast e Tecton. Vemos potencial na Feature Store e recomendamos que você a avalie cuidadosamente.

JuiceFS
Avalie

JuiceFS é um sistema de arquivos POSIX distribuído e de código aberto, construído com base em Redis e armazenamento de objetos. Ao construir novas aplicações, nossa recomendação sempre foi interagir diretamente com o armazenamento de objetos, sem passar por outra camada de abstração. No entanto, JuiceFS pode ser uma opção se você estiver migrando aplicações legadas que dependam de sistemas de arquivos POSIX tradicionais para a nuvem.

API do Kafka sem Kafka
Avalie

À medida que mais empresas se voltam para eventos como uma forma de compartilhar dados entre microsserviços, coletar analytics ou alimentar lagos de dados, Apache Kafka tornou-se a plataforma favorita para apoiar um estilo arquitetural orientado a eventos. Embora o Kafka fosse um conceito revolucionário em mensagens persistentes escaláveis, muitas partes móveis são necessárias para fazê-lo funcionar, incluindo o ZooKeeper, brokers, partições e mirrors. Embora possam ser particularmente difíceis de implementar e operar, elas oferecem grande flexibilidade e poder quando necessário, especialmente em uma escala empresarial. Por causa da alta barreira de entrada apresentada em todo o ecossistema Kafka, damos as boas-vindas à recente explosão de plataformas que oferecem a API do Kafka sem Kafka. Entradas recentes como Kafka on Pulsar e Redpanda oferecem arquiteturas alternativas, e o Azure Event Hubs para Kafka oferece alguma compatibilidade com APIs de produtor e consumidor do Kafka. Alguns recursos do Kafka, como a biblioteca cliente de streams, não são compatíveis com esses brokers alternativos, portanto, ainda há razões para escolher o Kafka entre as alternativas. Resta saber, entretanto, se as pessoas desenvolvedoras de fato irão adotar essa estratégia, ou se é apenas uma tentativa da concorrência de afastar usuários da plataforma Kafka. Em última análise, talvez o impacto mais duradouro do Kafka seja a conveniência do protocolo e da API fornecidos a clientes.

NATS
Avalie

NATS é um sistema de filas de mensagens rápido e seguro com uma variedade incomum de recursos e targets de implantação em potencial. À primeira vista, você pode se perguntar por que o mundo precisaria de outro sistema de filas de mensagens. As filas de mensagens existem em várias formas há quase tanto tempo quanto as empresas usam computadores, e passaram por anos de refinamento e otimização para várias tarefas. Mas o NATS tem várias características interessantes e é único em sua capacidade de escalar, que vai de controladores incorporados a superclusters globais hospedados em nuvem. Particularmente, intriga-nos a intenção do NATS de oferecer suporte a um fluxo contínuo de dados de dispositivos móveis e IoT e por meio de uma rede de sistemas interconectados. No entanto, alguns problemas complexos precisam ser resolvidos, como garantir que consumidores vejam apenas as mensagens e os tópicos aos quais têm acesso permitido, especialmente quando a rede ultrapassa os limites organizacionais. O NATS 2.0 introduziu um framework de controle de segurança e acesso que oferece suporte a clusters multitenant, onde as contas restringem o acesso do usuário a filas e tópicos. Escrito em Go, o NATS foi adotado principalmente pela comunidade da linguagem Go. Embora existam clientes para praticamente todas as linguagens de programação amplamente utilizadas, o cliente Go é de longe o mais popular. No entanto, algumas de nossas pessoas desenvolvedoras descobriram que todas as bibliotecas cliente de linguagem tendem a refletir as origens Go da base de código. Aumentar a largura de banda e a capacidade de processamento em pequenos dispositivos sem fio significa que o volume de dados que as empresas devem consumir em tempo real só aumentará. Avalie o NATS como uma plataforma possível para transmitir esses dados internamente e entre empresas.

Opstrace
Avalie

Opstrace é uma plataforma de observabilidade de código aberto destinada a ser implantada na própria rede da pessoa usuária. Se não usamos soluções comerciais como Datadog (por exemplo, por questões de custo ou residência de dados), a única solução é construir sua própria plataforma composta de ferramentas de código aberto. Isso pode exigir muito esforço — o Opstrace destina-se a preencher essa lacuna. Ele usa APIs de código aberto e interfaces como Prometheus e Grafana e adiciona recursos, como TLS e autenticação. O núcleo do Opstrace executa um cluster Cortex para fornecer API Prometheus escalável, bem como um cluster Loki para os logs. É uma plataforma relativamente nova e ainda carece de recursos quando comparada a soluções como Datadog ou SignalFX. Ainda assim, é uma adição promissora a este espaço e vale a pena ficar de olho.

Plataformas

À medida que mais empresas se voltam para eventos como uma forma de compartilhar dados entre microsserviços, coletar análises ou alimentar lagos de dados, o Apache Kafka se tornou uma plataforma favorita. Por causa da alta barreira de entrada apresentada por todo o ecossistema Kafka, damos as boas-vindas à recente explosão de plataformas que oferecem a API do Kafka sem Kafka.

(API do Karfa sem Kafka)

Plataformas

Às vezes, as organizações constroem frameworks ou abstrações baseadas em produtos externos para atender a necessidades específicas, imaginando que a adaptação trará mais benefícios do que os produtos existentes. Entretanto, elas subestimam o esforço necessário para manter essas soluções em evolução e, após um curto período de tempo, percebem que a versão original está em um estado muito melhor.

(Produtos caseiros de infraestrutura como código (IaC))

Pulumi
Avalie

Temos visto o interesse no Pulumi crescer de forma lenta, mas contínua. O Pulumi preenche uma lacuna no mundo da programação de infraestrutura, no qual o Terraform mantém uma posição sólida. Embora o Terraform seja uma ferramenta testada e aprovada, 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. O Pulumi tem foco fortemente voltado para arquiteturas nativas da nuvem — incluindo contêineres, funções sem servidor e serviços de dados — e oferece bom suporte para Kubernetes. Recentemente, o AWS CDK surgiu como desafiante, mas Pulumi continua a ser a única ferramenta neutra em nuvem nesta área. Estamos antecipando uma adoção mais ampla do Pulumi no futuro e otimistas por uma ferramenta viável e um ecossistema de conhecimento emergente para apoiá-lo.

Redpanda
Avalie

Redpanda é uma plataforma de streaming que fornece uma API compatível com Kafka, permitindo que ela se beneficie do ecossistema sem ter que lidar com as complexidades de uma instalação Kafka. Por exemplo, o Redpanda simplifica as operações enviando-as como um único binário e evitando a necessidade de uma dependência externa como o ZooKeeper. Em vez disso, ele implementa o protocolo Raft e realiza testes abrangentes para validar se foi implementado corretamente. Um dos recursos do Redpanda (disponível apenas para clientes corporativos) é a transformação em linha WebAssembly (WASM), usando um mecanismo WASM incorporado. Isso permite que as pessoas desenvolvedoras criem transformadores de eventos em sua linguagem de escolha e compilem para WASM. O Redpanda também oferece latências de cauda muito reduzidas e maior rendimento devido a uma série de otimizações. Redpanda é uma alternativa interessante ao Kafka e vale a pena avaliar.

Azure Machine Learning
Evite

Observamos anteriormente que as provedoras de nuvem disponibilizam cada vez mais serviços no mercado. Também documentamos nossas preocupações de que às vezes os serviços são disponibilizados antes de estarem totalmente prontos. Infelizmente, em nossa experiência, Azure Machine Learning se enquadra na última categoria. Um dos vários novos participantes no campo de plataformas limitadas de baixo código, o Azure ML promete mais conveniência para cientistas de dados. No final das contas, entretanto, ele não cumpre sua promessa. Na verdade, nossas pessoas cientistas de dados ainda acham mais fácil trabalhar com Python. Apesar de esforços significativos, encontramos dificuldades para escalar e a falta de documentação adequada mostrou ser mais um problema. Por essas razões, movemos a plataforma para o anel Evite.

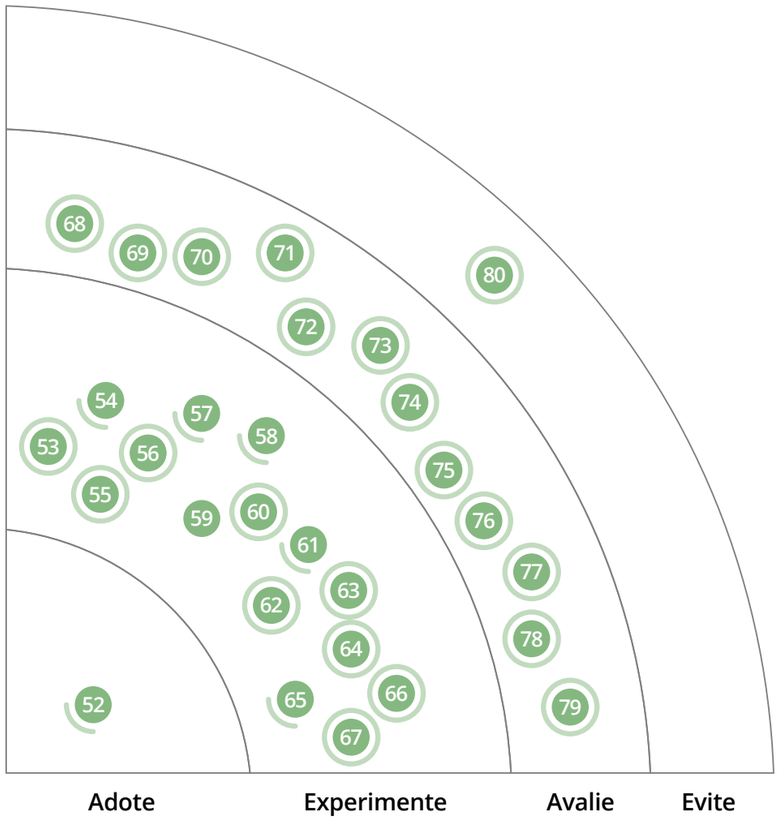
Produtos caseiros de infraestrutura como código (IaC)
Evite

Produtos que são suportados por empresas ou comunidades estão em constante evolução, pelo menos aqueles que ganham força na indústria. Às vezes, as organizações tendem a construir frameworks ou abstrações baseadas em produtos externos para atender a necessidades muito específicas, imaginando que a adaptação trará mais benefícios do que os produtos existentes. Estamos observando organizações tentando criar produtos caseiros de infraestrutura como código (IaC) com base nos existentes. Essas organizações subestimam o esforço necessário para manter essas soluções em evolução de acordo com suas necessidades e, após um curto período de tempo, percebem que a versão original está em muito melhor estado do que a sua. Há ainda casos em que a abstração do produto externo reduz as capacidades originais. Embora tenhamos visto histórias de sucesso de organizações criando soluções caseiras, queremos alertar sobre essa abordagem, pois o esforço necessário não é desprezível e uma visão de produto de longo prazo é necessária para ter os resultados esperados.

TECHNOLOGY RADAR

Ferramentas

Ferramentas



Sentry
Adote

Sentry tornou-se a escolha padrão para muitos de nossos times no que se refere a relatórios de erros de front-end. A conveniência de recursos como agrupamento de erros ou definição de padrões para descartar erros com determinados parâmetros ajuda a lidar com a enxurrada de erros provenientes de muitos dispositivos de usuários finais. Integrar o Sentry em seu pipeline de CD permite a você carregar mapas de origem para uma depuração de erros mais eficiente, e ajuda a rastrear facilmente quais erros ocorreram em qual versão do software. Também apreciamos o fato de que, embora o Sentry seja primordialmente uma oferta de SaaS, seu código-fonte está disponível para o público e pode ser usado gratuitamente para casos de uso menores e auto-hospedagem.

axe-core
Experimente

Tornar a web inclusiva requer muita atenção para garantir que a acessibilidade seja considerada *e* validada em todos os estágios da entrega de software. Muitas das ferramentas de teste de acessibilidade populares são projetadas para executar testes após a finalização de uma aplicação web. Como resultado, os problemas são detectados tardiamente e geralmente são mais difíceis de corrigir, acumulando-se como dívidas. Em nosso trabalho interno recente nos sites da Thoughtworks, incluímos o mecanismo de teste de acessibilidade (a11y) de código aberto axe-core como parte de nossos processos. Ele forneceu aos membros do time um feedback inicial sobre a adesão às regras de acessibilidade, mesmo durante os primeiros incrementos. No entanto, nem todos os problemas podem ser encontrados por meio da inspeção automatizada. Para estender a funcionalidade do axe-core, temos o axe DevTools disponível comercialmente, incluindo uma funcionalidade que orienta os membros do time por meio de testes exploratórios para a maioria dos problemas de acessibilidade.

dbt
Experimente

Desde a última vez que escrevemos sobre dbt, nós temos usado em alguns projetos e gostamos do que vimos. Por exemplo, gostamos do fato de que o dbt torna a parte da transformação de pipelines ELT mais acessível para consumidores de dados, em contraste com a prática de apenas pessoas engenheiras de dados construindo pipelines. O dbt faz isso ao mesmo tempo em que incentiva boas práticas de engenharia, como controle de versão, implantação e testes automatizados. SQL continua a ser a língua franca do mundo dos dados (incluindo bancos de dados, warehouses, motores de consulta, lagos de dados e plataformas analíticas) e a maioria desses sistemas oferece suporte até certo ponto. Isso permite que o dbt seja usado nesses sistemas para transformações, apenas criando adaptadores. O número de conectores nativos cresceu para incluir Snowflake, BigQuery, Redshift e Postgres, assim como a variedade de plugins da comunidade. Vemos ferramentas como o dbt ajudando as plataformas de dados a ampliar sua capacidade de "autoatendimento".

Adote
52. Sentry

Experimente
53. axe-core
54. dbt
55. esbuild
56. Flipper
57. Great Expectations
58. k6
59. MLflow
60. OR-Tools
61. Playwright
62. Prowler
63. Pyright
64. Redash
65. Terratest
66. Tuple
67. Why Did You Render

Avalie
68. Buildah e Podman
69. GitHub Actions
70. Graal Native Image
71. HashiCorp Boundary
72. imgcook
73. Longhorn
74. Operator Framework
75. Recomendador
76. Remote - WSL
77. Spectral
78. Yelp detect-secrets
79. Zally

Evite
80. AWS CodePipeline

Ferramentas

OR-Tools é um conjunto de softwares de código aberto para resolver problemas de otimização combinatória. Os problemas de otimização têm um conjunto muito grande de soluções possíveis, e ferramentas como essa são bastante úteis na busca da melhor solução.

(OR-Tools)

esbuild
Experimente

Sempre procuramos encontrar ferramentas que sejam capazes de encurtar o ciclo de feedback do desenvolvimento de software. esbuild é um exemplo. Conforme a base de código do front-end cresce, geralmente enfrentamos um tempo de alguns minutos de empacotamento. Como um empacotador JavaScript otimizado para velocidade, o esbuild pode reduzir este tempo por um fator de 10 a 100. Ele é escrito em Golang e usa uma abordagem mais eficiente no processo de análise, impressão e geração de mapa de origem que supera significativamente as ferramentas de construção, como Webpack e Parcel em tempo de compilação. O esbuild pode não ser tão abrangente quanto essas ferramentas na transformação da sintaxe JavaScript. No entanto, isso não tem impedido que muitos de nossos times migrem para o esbuild como padrão.

Flipper
Experimente

Flipper é um depurador de aplicação móvel extensível. Ele oferece suporte para criação de perfis, inspeção de layout interativa, visualizador de log e um inspetor de rede para aplicações iOS, Android e React Native. Em comparação com outras ferramentas de depuração para aplicações móveis, achamos o Flipper leve, rico em recursos e fácil de configurar.

Great Expectations
Experimente

Escrevemos sobre Great Expectations na edição anterior do Radar. Continuamos gostando e o movemos para o anel Avalie nesta edição. Great Expectations é um framework que permite criar controles integrados que sinalizam anomalias ou problemas de qualidade em pipelines de dados. Assim como os testes de unidade são executados em um pipeline de compilação, o Great Expectations faz afirmações durante a execução de um pipeline de dados. Gostamos de sua simplicidade e facilidade de uso — as regras armazenadas em JSON podem ser modificadas por especialistas em domínio de dados sem necessariamente precisar de habilidades de engenharia de dados.

k6
Experimente

Tivemos um pouco mais de experiências em testes de desempenho com k6 desde que o incluímos pela primeira vez no Radar, e com bons resultados. Nossos times têm gostado do foco na experiência de desenvolvimento e na flexibilidade da ferramenta. Embora seja fácil de começar a usar o k6 sozinho, ele realmente se destaca por sua facilidade de integração em um ecossistema de desenvolvimento. Por exemplo, usando o adaptador Datadog, um time foi capaz de visualizar rapidamente o desempenho em um sistema distribuído, identificando preocupações significativas antes de liberar o sistema em produção. Outro time, com a versão comercial do k6, conseguiu usar a extensão do Azure pipelines marketplace para fazer testes de desempenho em seu pipeline de CD e obter relatórios do Azure DevOps com pouco esforço. Como o k6 oferece suporte a limites que permitem asserções de teste automatizadas prontos para uso, é relativamente fácil adicionar um estágio ao pipeline para detectar a degradação do desempenho de novas mudanças, adicionando um poderoso mecanismo de feedback para pessoas desenvolvedoras.

MLflow
Experimente

MLflow é uma ferramenta de código aberto para rastreamento de experimentos de aprendizado de máquina e gerenciamento de ciclo de vida. O fluxo de trabalho para desenvolver e evoluir continuamente um modelo de aprendizado de máquina inclui uma série de experimentos (uma coleção de execuções), rastreamento de desempenho desses experimentos (uma coleção de métricas) e rastreamento e ajuste de modelos (projetos). O MLflow facilita esse fluxo de trabalho muito bem, suportando os padrões abertos existentes e se integrando bem com muitas outras ferramentas no ecossistema. O MLflow como um serviço gerenciado por Databricks na nuvem, disponível em AWS e Azure, está amadurecendo rapidamente e temos usado com sucesso em nossos projetos. Consideramos o MLflow uma ótima ferramenta para gerenciamento e rastreamento de modelos, com suporte para modelos de interação baseados em UI e API. Nossa única e crescente preocupação é que o MLflow tenta entregar muitas questões conflitantes como uma única plataforma, por exemplo, modelos de serviço e pontuação.

OR-Tools
Experimente

OR-Tools é um conjunto de softwares de código aberto para resolver problemas de otimização combinatória. Os problemas de otimização têm um conjunto muito grande de soluções possíveis, e ferramentas como OR-Tools são bastante úteis na busca da melhor solução. Você pode modelar o problema em qualquer uma das linguagens suportadas — Python, Java, C# ou C++ — e escolher entre vários solvers de código aberto ou comerciais suportados. Usamos com sucesso o OR-Tools em vários projetos de otimização com programação inteira e inteira mista.

Playwright
Experimente

Playwright permite escrever testes de Web UI para Chromium, Firefox, e também para WebKit, tudo por meio da mesma API. A ferramenta ganhou alguma atenção por seu suporte a todos os principais motores de busca de navegadores, algo que ela consegue fazer ao incluir versões corrigidas do Firefox e do Webkit. Continuamos ouvindo relatos de experiências positivas com Playwright, em particular sobre sua estabilidade. Os times também acharam simples migrar do Puppeteer, que tem uma API muito semelhante.

Prowler
Experimente

Celebramos o aumento da disponibilidade e a maturidade das ferramentas de análise de configuração de infraestrutura: Prowler ajuda os times a analisar suas configurações de infraestrutura da AWS e melhorar a segurança com base nos resultados. Embora Prowler já exista há algum tempo, a ferramenta evoluiu muito nos últimos anos e achamos valiosa sua capacidade de habilitar os times a assumir a segurança com um curto ciclo de feedback. Prowler categoriza as análises do benchmarking AWS CIS em diferentes grupos (gerenciamento de identidade e acesso, registro, monitoramento, rede, nível CIS 1, nível CIS 2, EKS-CIS), e inclui muitas análises que ajudam a obter informações sobre conformidade com PCI DSS e GDPR.

Pyright
Experimente

Embora duck typing seja certamente visto como um recurso interessante por muitas pessoas que programam em Python, às vezes — especialmente para bases de código maiores — a verificação de tipo também pode ser útil. Por esse motivo, várias anotações de tipo são propostas como Python Enhancement Proposals (PEPs) e Pyright é um verificador de tipos que funciona com essas anotações. Além disso, ele fornece inferências de tipo e algumas proteções que entendem construções de fluxo de código condicional. Projetado para grandes bases de código, o Pyright é rápido e suas verificações de modo de observação acontecem de forma incremental conforme os arquivos são alterados para encurtar ainda mais o ciclo de feedback. O Pyright pode ser usado diretamente na linha de comando, mas integrações para VS Code, Emacs, vim, Sublime e possivelmente outros editores também estão disponíveis. Em nossa experiência, o Pyright é preferível em relação a alternativas como mypy.

Redash
Experimente

A adoção de uma filosofia DevOps do tipo "você constrói, você executa" significa que os times devem ter maior atenção às métricas técnicas e de negócios que podem ser extraídas dos sistemas que são implantados. Frequentemente, descobrimos que as ferramentas analíticas são de difícil acesso para a maioria das pessoas desenvolvedoras e, portanto, o trabalho de capturar e apresentar as métricas é deixado para outros times — bem depois que as funcionalidades são enviadas às pessoas usuárias finais. Nossos times consideram o Redash muito útil para consultar métricas de produtos e criar dashboards que possam ser usados por pessoas desenvolvedoras em geral, reduzindo os ciclos de feedback e concentrando todo o time nos resultados do negócio.

Terratest
Experimente

O Terratest chamou nossa atenção no passado como uma opção interessante para testes de infraestrutura. Desde então, nossos times o têm usado e estão bastante entusiasmados com a estabilidade e a experiência oferecida. Terratest é uma biblioteca Golang que torna mais fácil escrever testes automatizados para código de infraestrutura. Usando ferramentas de infraestrutura como código, como Terraform, você pode criar componentes de infraestrutura reais (como servidores, firewalls ou balanceadores de carga) para implantar aplicações neles e validar o comportamento esperado usando Terratest. Ao final do teste, o Terratest pode "desimplantar" as aplicações e limpar os recursos. Isso o torna muito útil para testes de ponta a ponta de sua infraestrutura em um ambiente real.

Tuple
Experimente

Tuple é uma ferramenta relativamente nova, otimizada para programação em pares remota e projetada para preencher a lacuna que o Slack deixou no mercado após abandonar o Screenhero. Embora a ferramenta ainda exiba alguns problemas crescentes — a disponibilidade da plataforma está limitada ao Mac OS por enquanto (com suporte para Linux em breve) e existam algumas peculiaridades de UI a serem resolvidas — tivemos uma boa experiência de uso considerando essas restrições. Ao contrário de ferramentas de compartilhamento de vídeo e tela de uso geral, como o Zoom, o Tuple oferece suporte a controle duplo com dois cursores de mouse, e ao contrário de opções como o Visual Studio Live Share, não está vinculado a um IDE. Tuple oferece ainda suporte a chamadas de voz e vídeo, compartilhamento de área de transferência e menor latência do que ferramentas de uso geral. Isso tudo, além do recurso que permite desenhar e apagar na tela de seu par com facilidade, torna o Tuple uma ferramenta muito intuitiva e amigável para pessoas desenvolvedoras.

Why Did You Render
Experimente

Ao trabalhar com React, muitas vezes encontramos situações em que nossa página fica muito lenta porque alguns componentes são renderizados novamente quando não deveriam ser. Why Did You Render é uma biblioteca que ajuda a detectar por que um componente está sendo renderizado novamente. Ela faz isso aplicando um monkey patch no React. Nós a usamos em alguns de nossos projetos para depurar problemas de desempenho com grandes resultados.

Ferramentas

Tuple é uma ferramenta relativamente nova, otimizada para programação em pares remota e projetada para preencher a lacuna que o Slack deixou no mercado após abandonar o Screenhero.

(Tuple)

Ferramentas

imgcook é um produto SaaS da Alibaba que pode transformar de forma inteligente vários arquivos de design — Sketch, PSD e até mesmoimagens estáticas — em código front-end,

(imgcook)

Buildah e Podman
Avalie

Embora o Docker tenha se tornado o padrão sensato para a conteinerização, temos visto novos players neste espaço que estão chamando nossa atenção. É o caso de Buildah e Podman, que são projetos complementares para construção de imagens (Buildah) e execução de contêineres (Podman) usando uma abordagem sem usuário root em várias distribuições Linux. O Podman apresenta um mecanismo sem daemon para gerenciar e executar contêineres, que é uma abordagem interessante em comparação com o que o Docker faz. O fato de o Podman poder usar imagens Open Container Initiative (OCI) criadas pelo Buildah, ou imagens Docker, torna essa ferramenta ainda mais atraente e fácil de usar.

GitHub Actions
Avalie

Ferramentas de compilação e servidores de CI estão entre as mais antigas e usadas em nosso conjunto de ferramentas. Elas vão desde simples serviços hospedados em nuvem até servidores de pipeline complexos e definidos por código que oferecem suporte a frotas de máquinas de compilação. Dada a nossa experiência e a ampla gama de opções já disponíveis, adotamos inicialmente uma postura de ceticismo quando o GitHub Actions foi introduzido como mais um mecanismo para gerenciar o fluxo de compilação e integração. Mas a possibilidade de começar com um comportamento pequeno e facilmente personalizado para as pessoas desenvolvedoras significa que o GitHub Actions está se movendo em direção à categoria padrão de projetos menores. É difícil argumentar contra a conveniência de ter a ferramenta de compilação integrada diretamente no repositório do código-fonte. Uma comunidade entusiasmada surgiu em torno desse recurso e isso significa que uma ampla gama de ferramentas e fluxos de trabalho criados com a contribuição de pessoas usuárias está disponível. Fornecedores de ferramentas também têm demonstrado interesse via GitHub Marketplace. No entanto, ainda recomendamos que você proceda com cautela. Embora o código e o histórico do Git possam ser exportados para hosts alternativos, um fluxo de trabalho de desenvolvimento baseado no GitHub Actions não pode. Além disso, você deve usar seu bom senso para determinar quando um projeto é grande ou complexo o suficiente para justificar uma ferramenta de pipeline com suporte independente. Mas para começar a trabalhar rapidamente em projetos menores, vale a pena considerar o GitHub Actions e seu ecossistema em crescimento.

Graal Native Image
Avalie

Graal Native Image é uma tecnologia que compila código Java em um binário nativo do sistema operacional — na forma de um executável estaticamente vinculado ou uma biblioteca compartilhada. Uma imagem nativa é otimizada para reduzir o consumo de memória e o tempo de inicialização de uma aplicação. Nossos times têm usado com sucesso imagens nativas do Graal, executadas como pequenos contêineres do Docker na arquitetura sem servidor, em que a redução do tempo de inicialização é importante. Embora projetada para uso com linguagens de programação como Go ou Rust, que compilam nativamente e exigem binários menores e tempos de inicialização mais curtos, a imagem nativa do Graal pode ser igualmente útil para times que têm outros requisitos e desejam usar linguagens baseadas em JVM.

O Graal Native Image Builder, *native-image*, oferece suporte a linguagens baseadas em JVM — como Java, Scala, Clojure e Kotlin — e cria executáveis em vários sistemas operacionais como Mac OS, Windows e várias distribuições Linux. Uma vez que ele requer uma suposição de mundo fechado, na qual todo o código é conhecido no tempo de compilação, é necessária uma configuração adicional para recursos como *reflection* ou *dynamic class loading*, uma vez que os tipos não podem ser deduzidos apenas no tempo de compilação do código.

HashiCorp Boundary
Avalie

O HashiCorp Boundary reúne uma rede segura e os recursos de gerenciamento de identidade necessários para intermediar o acesso a seus hosts e serviços em um só lugar, combinando nuvem e recursos locais, se necessário. O gerenciamento de chaves pode ser feito integrando o serviço de sua escolha, seja de uma fornecedora de nuvem ou de algo como o HashiCorp Vault. O HashiCorp Boundary oferece suporte a um número crescente de provedores de identidade e pode ser integrado a partes de sua configuração de serviços para ajudar a definir permissões, não apenas no host, mas também em um nível de serviço. Por exemplo, ele permite que você controle as restrições de acesso a um cluster do Kubernetes, e a extração de catálogos de serviço de várias fontes está nos planos. Tudo isso fica fora do caminho das pessoas usuárias finais de engenharia, que obtêm a experiência de shell com a qual estão acostumadas, com uma conexão segura por meio da camada de gerenciamento de rede do Boundary.

imgcook
Avalie

Você se lembra do projeto de pesquisa pix2code, que mostrou como gerar código automaticamente a partir de capturas de tela da GUI? Agora, há uma versão desta técnica em formato de produto. imgcook é um produto SaaS da Alibaba que pode transformar de forma inteligente vários arquivos de design (Sketch/PSD/imagens estáticas) em código de front-end. A Alibaba precisa personalizar um grande número de páginas de campanha durante o festival de compras Double Eleven. Geralmente são páginas para uso específico nesse período que precisam ser desenvolvidas rapidamente. Por meio do método de aprendizado profundo, o design de UX é inicialmente processado em código de front-end e, em seguida, ajustado por uma pessoa desenvolvedora. Nosso time está avaliando esta tecnologia: embora o processamento de imagem ocorra no lado do

servidor, enquanto a interface principal está na web, o imgcook fornece ferramentas que podem se integrar com o design de software e o ciclo de vida do desenvolvimento. O imgcook pode gerar código estático, bem como algum código de componente de vinculação de dados, caso você defina uma DSL. A tecnologia ainda não é perfeita: designers precisam consultar certas especificações para melhorar a precisão da geração de código (que ainda precisa ser ajustada por pessoas desenvolvedoras posteriormente). Sempre temos cautela com a geração de código mágica, porque o código gerado costuma ser difícil de manter a longo prazo, e o imgcook não é exceção. Mas se você limitar o uso a um contexto específico, como páginas para campanhas específicas, vale a pena tentar.

Longhorn
Avalie

Longhorn é um sistema de armazenamento de bloco distribuído para Kubernetes. Existem muitas opções de armazenamento persistente para Kubernetes. Ao contrário da maioria, no entanto, o Longhorn é construído desde o início para fornecer backups e snapshots incrementais, facilitando assim a execução de um armazenamento replicado para Kubernetes não hospedado em nuvem. Com o recente suporte experimental para ReadWriteMany (RWX), você pode até montar o mesmo volume para acesso de leitura e gravação em vários nós. Escolher o sistema de armazenamento certo para Kubernetes não é uma tarefa trivial, e recomendamos que você avalie o Longhorn com base em suas necessidades.

Operator Framework
Avalie

Operator Framework é um conjunto de ferramentas de código aberto que simplifica a construção e o gerenciamento do ciclo de vida dos operadores Kubernetes. O padrão de operador do Kubernetes, originalmente introduzido pelo CoreOS, é uma abordagem para encapsular o conhecimento de operação de uma aplicação usando recursos nativos do Kubernetes. Inclui *recursos* a serem gerenciados e *código do controlador* , garantindo que os recursos correspondam ao estado de desejado. Essa abordagem foi usada para estender o uso do Kubernetes ao gerenciamento de várias aplicações, especialmente aquelas com estado, nativamente. O Operator Framework tem três componentes: Operator SDK, que simplifica a construção, os testes e o empacotamento de operadores Kubernetes; o Gerenciador de ciclo de vida, para instalar, gerenciar e atualizar os operadores; e um catálogo, para publicar e compartilhar operadores de terceiras partes. Nossos times consideram o Operator SDK particularmente poderoso no desenvolvimento rápido de aplicações nativas do Kubernetes.

Recomendador
Avalie

O número de serviços oferecidos pelas grandes provedoras de nuvem continua crescendo, mas a conveniência e a maturidade das ferramentas que ajudam a usar esses serviços com segurança e eficiência também vem crescendo. O Recomendador é um serviço da Google Cloud que analisa seus recursos e fornece recomendações sobre como otimizá-los com base em seu uso real. O serviço consiste em uma variedade de "recomendadores" em áreas como segurança, uso de computação ou economia de custos. O Recomendador do IAM, por exemplo, ajuda a implementar melhor o princípio do menor privilégio, apontando permissões que nunca são de fato usadas e, portanto, são potencialmente amplas demais.

Remote - WSL
Avalie

Nos últimos anos, o Subsistema Windows para Linux (WSL) surgiu algumas vezes em nossas discussões. Apesar de gostarmos do que vimos, incluindo as melhorias no WSL 2, ele nunca foi incluído no Radar. Nesta edição, queremos destacar uma extensão para Visual Studio Code que melhora muito a experiência de trabalho com o WSL. Ainda que os editores baseados em Windows sempre pudessem acessar os arquivos em um sistema de arquivos WSL, eles desconheciam o ambiente Linux isolado. Com a extensão Remote - WSL, o Visual Studio Code torna-se ciente do WSL, permitindo às pessoas desenvolvedoras iniciar um shell do Linux. Isso também permite a depuração de binários em execução no WSL do Windows. O IntelliJ da Jetbrains também tem recebido melhorias constantes em seu suporte para WSL.

Spectral
Avalie

Um padrão que vimos se repetir nesta publicação é que ferramentas estáticas de verificação de erros e estilo surgem rapidamente após uma nova linguagem ganhar popularidade. Essas ferramentas são genericamente conhecidas como linters — em homenagem ao clássico e amado utilitário do Unix, *lint*, que analisa estaticamente código em C. Gostamos dessas ferramentas porque elas detectam os erros cedo, antes mesmo que o código seja compilado. A instância mais recente desse padrão é Spectral, um linter para YAML e JSON. Embora Spectral seja uma ferramenta genérica para esses formatos, seus alvos principais são OpenAPI (a evolução do Swagger) e AsyncAPI. O Spectral vem com um conjunto abrangente de regras prontas para usar para especificações que podem evitar dores de cabeça para pessoas desenvolvedoras ao projetar e implementar APIs ou colaboração orientada a eventos. Essas regras verificam as especificações adequadas aos parâmetros da API ou a existência de uma declaração de licença na especificação, entre outras coisas. Embora essa ferramenta seja uma adição bem-vinda ao fluxo de desenvolvimento de APIs, ela levanta a questão: uma especificação não executável deve ser tão complexa a ponto de exigir uma técnica de verificação de erros projetada para linguagens de programação? Talvez as pessoas desenvolvedoras devam se concentrar em escrever código, e não especificações.

Ferramentas

O Recomendador é um serviço da Google Cloud que analisa seus recursos e fornece recomendações sobre como otimizá-los com base em seu uso real.

(Recomendador)

Ferramentas

Zally é um linter OpenAPI minimalista que ajuda a garantir que uma API esteja em conformidade com o guia de estilo de API do time.

(Zally)

Yelp detect-secrets
Avalie

Yelp detect-secrets é um módulo Python para detectar segredos em uma base de código, fazendo a verificação de arquivos em um diretório em busca de segredos. Ele pode ser usado como um hook de pré-commit do Git ou para executar uma verificação em vários locais dentro do pipeline de CI/CD. Ele vem com uma configuração padrão que o torna muito fácil de usar, mas pode ser modificado para atender às suas necessidades. Você também pode instalar plugins personalizados para adicionar às pesquisas heurísticas padrão. Em comparação com ofertas semelhantes, consideramos que essa ferramenta detecta mais tipos de segredos com sua configuração padrão.

Zally
Avalie

Conforme o ecossistema de especificações de API amadurece, temos visto mais ferramentas construídas para automatizar as verificações de estilo. Zally é um linter OpenAPI minimalista que ajuda a garantir que uma API esteja em conformidade com o guia de estilo de API do time. Pronto para uso, ele valida em relação a um conjunto de regras desenvolvido para o guia de estilo de API da Zalando, mas também suporta um mecanismo de extensão do Kotlin para desenvolver regras personalizadas. O Zally inclui uma UI da web que fornece uma interface intuitiva para entender as violações de estilo e uma CLI que facilita a conexão ao pipeline de CD.

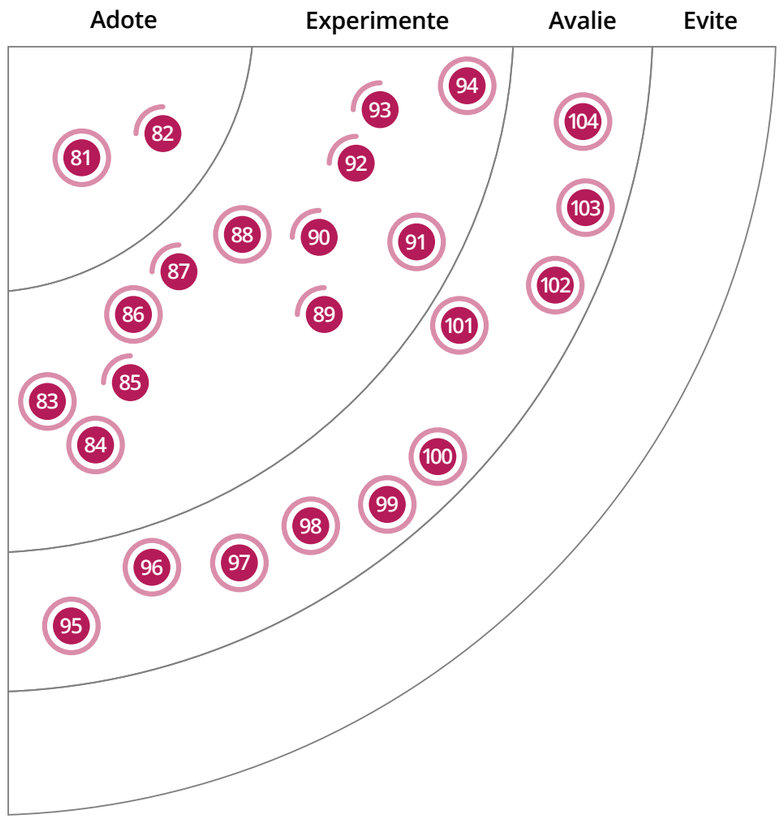
AWS CodePipeline
Evite

Com base nas experiências de vários dos times da Thoughtworks, sugerimos abordar AWS CodePipeline com cautela. Especificamente, descobrimos que, uma vez que os times vão além de pipelines simples, essa ferramenta pode se tornar difícil de trabalhar. Embora possa parecer que o resultado é um "sucesso rápido" ao começar trabalhar com AWS, sugerimos dar um passo atrás e verificar se o AWS CodePipeline realmente atende às suas necessidades de longo prazo, por exemplo, pipeline fan-out e fan-in ou implementações mais complexas e cenários de teste com dependências e gatilhos não triviais.

TECHNOLOGY RADAR

Linguagens & Frameworks

Linguagens & Frameworks



Combine
Adote

Há muito tempo, incluímos ReactiveX — uma família de frameworks de código aberto para programação reativa — no anel Adote do Radar. Em 2017, mencionamos a adição do RxSwift, que trouxe a programação reativa para o desenvolvimento iOS com Swift. Desde então, a Apple introduziu sua própria visão da programação reativa na forma do framework Combine. Combine se tornou nossa escolha padrão para aplicativos que suportam iOS 13 como um destino de implantação aceitável. É mais fácil de aprender do que RxSwift e se integra muito bem com SwiftUI. Se você está planejando converter uma aplicação existente de RxSwift para Combine ou trabalhar com os dois no mesmo projeto, você pode se interessar em conferir o RxCombine.

LeakCanary
Adote

Nossos times de desenvolvimento para dispositivos móveis atualmente veem LeakCanary como uma boa escolha padrão para desenvolvimento Android. Ele detecta vazamentos irritantes de memória no Android, é extremamente simples de integrar e fornece notificações com rastreamento claro da causa do vazamento. O LeakCanary pode economizar horas de tédio investigando erros de falta de memória em vários dispositivos, e recomendamos que você o adicione ao seu conjunto de ferramentas.

Angular Testing Library
Experimente

Continuamos desenvolvendo aplicações web em JavaScript e seguimos gostando da abordagem da Testing Library para testar aplicações, e seguimos explorando e ganhando experiência com seus pacotes — além da React Testing Library. A Angular Testing Library traz todos os benefícios de sua família ao testar componentes de UI de uma forma centrada no usuário, impulsionando testes de fácil manutenção com foco principalmente no comportamento do usuário, em vez de testar os detalhes de implementação da UI. Embora sua documentação seja insuficiente, a Angular Testing Library fornece bons exemplos de testes que nos ajudaram a começar mais rápido em vários casos. Tivemos grande sucesso com esta biblioteca de testes em nossos projetos Angular e aconselhamos você a experimentar esta sólida abordagem de testes.

AWS Data Wrangler
Experimente

A AWS Data Wrangler é uma biblioteca de código aberto que estende os recursos do Pandas para a AWS, conectando frames de dados a serviços da AWS relacionados a dados. Além do Pandas,

Adote
81. Combine
82. LeakCanary

Experimente
83. Angular Testing Library
84. AWS Data Wrangler
85. Blazor
86. FastAPI
87. io-ts
88. Kotlin Flow
89. LitElement
90. Next.js
91. On-demand modules
92. Streamlit
93. SWR
94. TrustKit

Avalie
95. .NET 5
96. bUnit
97. Dagster
98. Flutter para Web
99. Jotai e Zustand
100. Kotlin Multiplatform Mobile
101. LVGL
102. React Hook Form
103. River
104. Webpack 5 Module Federation

Evite

Linguagens & Frameworks

FastAPI é um framework web de alto desempenho, moderno e rápido para construir APIs com Python 3.6 ou posteriores.

(FastAPI)

a biblioteca usa Apache Arrow e Boto3 para expor várias APIs para carregar, transformar e salvar dados em lagos de dados e data warehouses. Uma limitação importante é que você não pode executar grandes pipelines de dados distribuídos com esta biblioteca. No entanto, você pode aproveitar os serviços de dados nativos — como Athena, Redshift e Timestream — para fazer o trabalho pesado e extrair dados a fim de expressar transformações complexas que são a dequadas para frames de dados. Usamos o AWS Data Wrangler em produção e, dessa forma, ele permite que você se concentre em escrever transformações sem gastar muito tempo na conectividade com os serviços de dados da AWS.

Blazor
Experimente

Embora JavaScript e seu ecossistema sejam dominantes no espaço de desenvolvimento de web UI, novas oportunidades estão aparecendo com o surgimento de WebAssembly. O Blazor continua chamando nossa atenção ao produzir bons resultados para nossos times, que estão criando interfaces de usuário ricas e interativas usando C# em cima de WebAssembly.
O fato de nossos times poderem usar C# no front-end também permite que compartilhem código e reutilizem as bibliotecas existentes. Isso, junto com as ferramentas existentes para depuração e testes, como bUnit, é o que faz valer a pena experimentar esse framework de código aberto.

FastAPI
Experimente

Temos visto mais times adotando Python como a linguagem preferida para construir soluções, não apenas para ciência de dados, mas também para serviços de back-end. Nesses cenários, temos tido boas experiências com FastAPI — um framework web de alto desempenho, moderno e rápido para construir APIs com Python 3.6 ou posteriores. Além disso, este framework e seu ecossistema incluem recursos como documentação de API usando OpenAPI, que permite que nossos times se concentrem nas funcionalidades do negócio e criem APIs REST rapidamente, tornando FastAPI uma boa alternativa às soluções existentes neste espaço.

io-ts
Experimente

Temos gostado muito de usar o TypeScript há algum tempo, e adoramos a segurança que a tipagem forte oferece. No entanto, colocar dados dentro dos limites do sistema de tipos, por exemplo, de uma chamada para um serviço de back-end, pode levar a erros de tempo de execução. Uma biblioteca que ajuda a resolver esse problema é io-ts. Ela preenche a lacuna entre a verificação de tipo em tempo de compilação e o consumo em tempo de execução de dados externos, fornecendo funções de codificação e decodificação. Também pode ser usada como uma proteção de tipos personalizados. À medida que ganhamos mais experiência com io-ts, nossas impressões inicialmente positivas vêm sendo confirmadas, e continuamos gostando da elegância de sua abordagem.

Kotlin Flow
Experimente

A introdução de corrotinas no Kotlin abriu portas para várias inovações — o Kotlin Flow é uma delas, integrando-se diretamente à biblioteca de corrotinas. É uma implementação de Reactive Streams com base em corrotinas. Ao contrário do RxJava, os fluxos são uma API Kotlin nativa semelhante à API de sequência familiar, com métodos que incluem `map` e `filter`. Assim como as sequências, os fluxos s ão *frios*, o que significa que os valores da sequência são construídos apenas quando necessário. Tudo isso torna a programação multithread muito mais simples e fácil de entender do que outras abordagens. O método `toList`, previsivelmente, converte um fluxo em uma lista, que é um padrão comum em testes

LitElement
Experimente

O progresso tem sido constante desde que escrevemos sobre Web Components em 2014. LitElement, parte do Polymer Project, é uma biblioteca simples que pode ser usada para criar componentes web leves. Na verdade, é apenas uma classe base que elimina a necessidade de boa parte do boilerplate comum, tornando a programação de componentes web muito mais fácil.
Tivemos sucesso usando-o em projetos e, à medida que vemos a tecnologia amadurecendo e a biblioteca sendo bem aceita, LitElement vem se tornando mais comumente usado em nossos projetos baseados em Web Components.

Next.js
Experimente

Tivemos mais algumas experiências usando Next.js para bases de código React desde a última vez que escrevemos sobre ele. Next.js é um framework opinativo de configuração zero que inclui roteamento simplificado, compilação automática e empacotamento com Webpack e Babel, recarregamento rápido para um fluxo de trabalho de desenvolvimento conveniente, entre outros recursos. Ele fornece renderização do lado do servidor por padrão, melhora a otimização do mecanismo de pesquisa e o tempo de carregamento inicial, e oferece

suporte à geração estática incremental. Recebemos relatos de experiências positivas de times usando Next.js e, devido à sua grande comunidade, seguimos otimistas em relação à evolução do framework.

On-demand modules
Experimente

On-demand modules para Android é um framework que permite que APKs personalizados contendo apenas a funcionalidade necessária sejam baixados e instalados em um aplicativo adequadamente estruturado. Pode valer a pena testá-lo com aplicativos maiores, nos quais a velocidade de download pode ser um problema, ou caso um usuário tenda a usar apenas parte da funcionalidade na instalação inicial. Ele também pode simplificar o manuseio de vários dispositivos sem exigir APKs diferentes. Um framework semelhante está disponível para iOS.

Streamlit
Experimente

Streamlit é um framework de código aberto em Python, usado por cientistas de dados para construir aplicações de dados interativas. Ajustar modelos de aprendizado de máquina leva tempo. Em vez de ir e voltar na aplicação principal (que usa esses modelos), encontramos valor em construir protótipos autônomos com rapidez no Streamlit e coletar feedback durante os ciclos de experimentação. O Streamlit se destaca em relação a concorrentes como Dash por seu foco em prototipagem rápida e suporte para uma ampla gama de bibliotecas de visualização, incluindo Plotly e Bokeh. Temos usado em alguns projetos e gostamos da possibilidade de criar visualizações interativas com pouquíssimo esforço.

SWR
Experimente

Nossos times descobriram que, quando usada em circunstâncias apropriadas, a biblioteca React Hooks SWR pode resultar em um código mais limpo e um desempenho muito superior. A SWR implementa a estratégia de cache HTTP stale-while-revalidate, primeiro retornando dados do cache (stale), depois enviando a solicitação de busca (revalidate) e, finalmente, atualizando os valores com a resposta atualizada. Aconselhamos os times a usar a estratégia de cache SWR apenas quando for esperado que uma aplicação retorne dados desatualizados. Note que o HTTP requer que os caches respondam a uma solicitação com a resposta mais atualizada. Somente em *circunstâncias cuidadosamente consideradas* uma resposta desatualizada pode ser retornada.

TrustKit
Experimente

A fixação de chave pública SSL pode ser complicada. Se você selecionar uma política errada ou não tiver um PIN de backup, sua aplicação irá parar de funcionar inesperadamente. É aqui que TrustKit é útil — é um framework de código aberto que torna a fixação de chave pública SSL mais fácil para aplicações iOS. Há um framework equivalente para Android também. Escolher a estratégia de fixação correta é um tema com muitas nuances, e você pode encontrar mais detalhes sobre isso no guia de primeiros passos. Temos usado o TrustKit em vários projetos em produção, e ele tem funcionado bem.

.NET 5
Avalie

Nós não incluímos cada nova versão do .NET no Radar, mas o .NET 5 representa um passo significativo para unir o .NET Core e o .NET Framework em uma única plataforma. As organizações devem começar a desenvolver uma estratégia para migrar seus ambientes de desenvolvimento — uma mistura fragmentada de frameworks dependendo do destino da implantação — para uma única versão do .NET 5 ou 6 quando estiver disponível. A vantagem dessa abordagem será uma plataforma de desenvolvimento comum, independentemente do ambiente pretendido: Windows, Linux, dispositivos móveis multiplataforma, (com Xamarin), ou navegador, (usando Blazor). Embora o desenvolvimento poliglota continue sendo a abordagem preferida para empresas com a cultura de engenharia necessária para suportá-lo, outras acharão mais eficiente padronizar em uma única plataforma de desenvolvimento .NET. Por enquanto, queremos manter isso no anel Avalie para observarmos a performance do framework unificado final no .NET 6.

bUnit
Avalie

bUnit é uma biblioteca de testes para Blazor que facilita a criação de testes para componentes do Blazor em frameworks de teste de unidade existentes, como NUnit, xUnit ou MSUnit. bUnit fornece uma fachada em torno do componente, permitindo que ele seja executado e testado dentro do paradigma familiar de teste de unidade, permitindo assim feedback e testes muito rápidos do componente de forma isolada. Se você estiver desenvolvendo para o Blazor, recomendamos que você adicione o bUnit à sua lista de ferramentas para experimentar.

Linguagens & Frameworks

Streamlit é um framework de código aberto em Python, usado por cientistas de dados para construir aplicações de dados interativas.

(Streamlit)

Linguagens & Frameworks

Estas bibliotecas de gerenciamento de estado para React pretendem ser pequenas e simples de usar e, talvez não por coincidência, ambos os nomes são traduções da palavra estado em japonês e alemão, respectivamente.

(Jotai e Zustand)

Dagster
Avalie

Dagster é um framework de orquestração de dados de código aberto para aprendizado de máquina, analytics e pipelines de dados ETL simples. Ao contrário de outros frameworks orientados a tarefas, Dagster reconhece os dados que fluem pelo pipeline e pode fornecer segurança de tipagem. Com essa visão unificada de pipelines e ativos produzidos, Dagster pode agendar e orquestrar Pandas, Spark, SQL ou qualquer outra coisa que Python possa invocar. O framework é relativamente novo, e recomendamos que você avalie seus recursos para pipelines de dados.

Flutter para Web
Avalie

Até o momento, o Flutter ofereceu suporte principalmente a aplicações iOS e Android nativas. No entanto, a visão da equipe do Flutter é apoiar a construção de aplicações em todas as plataformas. O Flutter para Web é um passo nessa direção — nos permite construir aplicações para iOS, Android e navegador a partir da mesma base de código. Ele já está disponível há mais de um ano em "Beta", mas com o lançamento recente do Flutter 2.0, o Flutter para Web atingiu a estabilidade. Na versão inicial do suporte para web, a equipe do Flutter está se concentrando em aplicações web progressivas (PWAs), aplicações de página única (SPAs) e expandindo os aplicativos móveis existentes para a web. A aplicação e o código do framework (todos em Dart) são compilados para JavaScript em vez do código de máquina ARM, que é usado para aplicativos móveis. O mecanismo web do Flutter oferece a escolha entre dois renderizadores: um renderizador HTML, que usa HTML, CSS, Canvas e SVG, e um renderizador CanvasKit que usa WebAssembly e WebGL para renderizar comandos do Skia na tela do navegador. Alguns de nossos times começaram a usar o Flutter para Web e gostaram dos resultados iniciais.

Jotai e Zustand
Avalie

Na edição anterior do Radar, comentamos sobre o início de uma fase de experimentação com gerenciamento de estado em aplicações React. Movemos o Redux de volta ao anel Experimente, registrando que não é mais nossa escolha padrão, e mencionando o Recoil do Facebook. Neste volume, queremos destacar Jotai e Zustand: ambas são bibliotecas de gerenciamento de estado para React, ambas pretendem ser pequenas e simples de usar e, talvez não por coincidência, ambos os nomes são traduções da palavra *estado* em japonês e alemão, respectivamente. Além dessas semelhanças, no entanto, há diferenças em seu design. O design do Jotai é mais próximo ao do Recoil, pois o estado consiste em átomos armazenados na árvore de componentes React, enquanto o Zustand armazena o estado fora do React em um único objeto de estado, de forma muito semelhante à abordagem adotada pelo Redux. Os autores do Jotai fornecem uma lista de verificação útil para decidir quando usar qual.

Kotlin Multiplatform Mobile
Avalie

Seguindo a tendência de desenvolvimento móvel multiplataforma, o Kotlin Multiplatform Mobile (KMM) é um novo participante neste espaço. KMM é um SDK fornecido pela JetBrains que aproveita os recursos multiplataforma do Kotlin e inclui ferramentas e funcionalidades projetadas para tornar a experiência de construção ponta a ponta de aplicativos móveis multiplataforma mais agradável e eficiente. Com o KMM, você escreve o código uma vez para a lógica de negócios e o núcleo do aplicativo em Kotlin, e depois o compartilha com os aplicativos Android e iOS. Você escreve o código específico da plataforma apenas quando necessário, por exemplo, para aproveitar as vantagens dos elementos nativos da UI, e o código específico destes elementos é mantido em diferentes *views* para cada plataforma. Embora ainda em Alpha, o Kotlin Multiplatform Mobile está evoluindo rapidamente. Certamente ficaremos de olho nele, e você também deveria

LVGL
Avalie

Com a crescente popularidade de dispositivos smart home e wearables, a demanda por interfaces gráficas de usuário (GUIs) intuitivas está aumentando. No entanto, se você trabalha com desenvolvimento de dispositivos embarcados, e não Android/iOS, o desenvolvimento de GUI pode exigir muito esforço. Uma biblioteca gráfica embarcada de código aberto, a LVGL vem ganhando cada vez mais popularidade. A LVGL foi adaptada para plataformas embarcadas convencionais, como NXP, STM32, PIC, Arduino e ESP32. Ela tem um footprint de memória muito pequeno: 64 kB de flash e 8 kB de RAM são suficientes para fazê-la funcionar, e ela pode ser executada sem problemas em vários MCUs Cortex-M0 de baixo consumo de energia. A LVGL suporta tipos de entrada como touchscreen, mouse e botões, e contém mais de 30 controles, incluindo TileView adequado para relógios inteligentes. A licença do MIT escolhida não restringe o uso empresarial e comercial. O feedback de nossos times sobre essa ferramenta tem sido positivo, e um de nossos projetos usando LVGL já está em produção, mais especificamente na manufatura de pequenos lotes.

React Hook Form
Avalie

Criar formulários para a web continua sendo um dos desafios constantes do desenvolvimento front-end, em particular com React. Muitos de nossos times que trabalham com React têm usado o Formik para tornar essa tarefa mais fácil, mas alguns estão avaliando o React Hook Form como uma alternativa potencial. Os React Hooks já existiam quando o React Hook Form foi criado, então ele pôde usá-los como um conceito de primeira classe: o framework registra e rastreia os elementos do formulário como componentes não controlados por meio de um hook, reduzindo significativamente a necessidade de uma nova renderização. Ele também é bem leve em tamanho e na quantidade de código boilerplate necessária.

River
Avalie

No centro de muitas abordagens de aprendizado de máquina está a criação de um modelo a partir de um conjunto de dados de treinamento. Depois que um modelo é criado, ele pode ser usado repetidamente. No entanto, o mundo não é estático e, muitas vezes, o modelo precisa ser alterado conforme novos dados se tornam disponíveis. A simples reexecução da etapa de criação do modelo pode ser lenta e cara. O aprendizado incremental resolve esse problema, tornando possível aprender a partir de fluxos de dados de forma incremental para reagir às mudanças com mais rapidez. Como bônus, os requisitos de computação e memória são menores e previsíveis. Em nossas implementações, tivemos uma boa experiência com o framework River, mas até o momento adicionamos verificações, às vezes manuais, após atualizações no modelo.

Webpack 5 Module Federation
Avalie

O lançamento da funcionalidade Webpack 5 Module Federation foi muito aguardado por pessoas desenvolvedoras de arquiteturas de micro frontends. A funcionalidade apresenta uma maneira mais padronizada de otimizar a forma como as dependências de módulo e o código compartilhado são gerenciados e carregados. A funcionalidade de module federation permite a especificação de módulos compartilhados, o que ajuda na redução de duplicação de dependências entre micro frontends, carregando o código usado por vários módulos apenas uma vez. Também permite distinguir entre módulos locais e remotos, quando os módulos remotos não são realmente parte do build em si, mas carregados de forma assíncrona. Comparado às dependências de tempo de compilação, como pacotes npm, isso pode simplificar significativamente a implantação de uma atualização de módulo com muitas dependências downstream. Esteja ciente, porém, que isso requer que você empacote todos os seus micro front-ends com o Webpack, ao contrário de abordagens como import maps, que podem eventualmente se tornar parte do padrão W3C.

Linguagens & Frameworks

Os modelos de aprendizado de máquina geralmente precisam ser alterados à medida que novos dados se tornam disponíveis. O aprendizado incremental torna possível aprender a partir de fluxos de dados de forma incremental para responder às mudanças mais rapidamente. Em nossas implementações, tivemos boas experiências com River, uma biblioteca Python para aprendizado de máquina online.

(River)

/thoughtworks

Somos uma consultoria global de software e uma comunidade de indivíduos apaixonados e guiados por propósitos, com mais de 9.000 pessoas em 48 escritórios distribuídos em 17 países. Em nossos mais de 27 anos de história, ajudamos clientes a resolver problemas de negócio complexos, usando a tecnologia como diferencial. Quando a mudança é a única constante, nós preparamos você para o imprevisível.

Quer se atualizar com artigos e informações relacionadas ao Radar?

Siga nossos perfis nas redes sociais e aproveite para se tornar assinante.

assine

/thoughtworks

thoughtworks.com/radar

#TWTechRadar